Бусыгин Б.С.
Коротенко Г.М.
Коротенко Л.М.

Учебник

Введение в современную информатику

www.programmer.dp.ua

г. Днепропетровск, НГУ, 2004 год

Книга является победителем VI-го Областного межвузовского конкурса на «Лучшие научные, учебные, учебно-методические и художественно-публицистические издания», 2006 г.

Книга утверждена Министерством образования и науки, молодёжи и спорта Украины в качестве учебника для ВУЗов.

кафедра
Программного обеспечения компьютерных систем
www.programmer.dp.ua

Сведения про авторов

Автор	*e-mail*
Бусыгин Борис Сергеевич доктор технических наук, профессор кафедры геоинформационных систем НГУ	
Коротенко Григорий Михайлович доктор технических наук, профессор кафедры геоинформационных систем НГУ	gkorotenko@rambler.ru
Коротенко Леонид Михайлович кандидат технических наук, доцент кафедры программного обеспечения компьютерных систем НГУ	leonid_korotenko@ukr.net

Подробнее о книге на сайте:

www.programmer.dp.ua/books/

УДК 004.4
ББК 32.97
Б92

Материалы книги утверждены в качестве учебника для студентов высших учебных заведений (письмо Министерства образования и науки Украины № 14/18-2-1733 от 16.07.04)

Рецензенты:

Научно-методическая комиссия по компьютерным наукам Научно-методического совета Министерства образования и науки Украины.

О.Ф. Приставка, д-р техн. наук, профессор (Днепропетровский национальный университет, профессор кафедры математического обеспечения и электронных вычислительных машин).

И.В. Жуковицкий, д-р техн. наук, профессор (Днепропетровский государственный технический университет железнодорожного транспорта, заведующий кафедрой электронных вычислительных машин).

Бусыгин Б.С., Коротенко Г.М., Коротенко Л.М.
Б92 Введение в современную информатику: – 559 с.

Учебник для студентов компьютерных специальностей. – Днепропетровск: Национальный горный университет, 2004.

УДК 004.4
ББК 32.97

СОДЕРЖАНИЕ

Введение

Если бы за последние 25 лет авиационная промышленность развивалась столь же стремительно, как вычислительная техника, то Боинг-767 можно было бы сегодня приобрести за 500 долларов и облететь на нем земной шар за 20 минут, израсходовав при этом 19 литров горючего.

Журнал «Scientific American», 1985 г.

Сегодня никого не удивляет тот факт, что информационный джинн, стремительно ворвавшись в современное общество, резко ускорил процесс воспроизводства знаний. С момента своего появления около пятидесяти лет назад, информатика стала определяющей технологией нашего времени. Достижения в области информационных технологий за короткое время стали одной из основных движущих сил современного общества. Человечество с трудом поспевает за этими темпами развития.

Вместе с тем, любой из нас, подключившись через модем компьютера к всемирной сети Интернет, сразу же погружается в сплав многоуровневых, новейших, интенсивно развивающихся программно-аппаратных сетевых технологий. Скорее всего, на Вашем персональном компьютере с процессором, имеющем частоту не менее 1,7 Ггц (производства Intel, а может и AMD), уже стоит операционная система Windows XP (кстати, задумайтесь, как давно она заменила у Вас предыдущие версии 95, 98. Millenium?), а набор устройств Вашего компьютера измеряется уже десятками единиц (видеокарта, аудио колонки, микрофон, TV-тьюнер, FM-тьюнер, аудио- и видео-проигрыватели, привод CD-RW, сканер, принтер, флэш-память) и этот список Вас уже не удивляет, а его можно продолжать и продолжать…

И если через несколько лет у Вас возникнет желание ещё раз открыть эту книгу, а может она просто случайно попадёт к Вам в руки, то Вы сможете убедиться, как изменилась первая часть списка и пополнилась вторая.

Уже на 2002 год, по мнению специалистов, из 132 сформировавшихся тем в совокупности знаний по информатике, «ядро» обязательных составляли 64![1] Вся совокупность знаний по информатике условно может быть разделена на 14 областей (см. таблицу 1).

[1] Рекомендации по преподаванию информатики в университетах: Пер. с англ. –СПб.: 2002. –372с.

Таблица 1. Структуризация знаний по информатике на 2002 год

Дискретные структуры (DS)	Графика и визуализация (GV)
Основы программирования (PF)	Интеллектуальные системы (IS)
Алгоритмы и теория сложности (AL)	Управление информацией (IM)
Архитектура и организация ЭВМ (AR)	Методы вычислений (CN)
Операционные системы (OS)	Человеко-машинное взаимодействие (HC)
Распределённые (сетевые) вычисления (NC)	Социальные и профессиональные вопросы программирования (SP)
Языки программирования (PL)	Программная инженерия (SE)

Многие процессы, влияющие на информатику, связаны с прогрессом в развитии новых современных технологий. Как эволюционные, так и революционные изменения в информационных технологиях повысили важность многих тем и, в особенности, следующих:

❶ WWW и его приложения (программирование процессов обработки и обслуживания ресурсов "Всемирной паутины" (Web-сайтов, Web-узлов, порталов и т.д.)).

❷ Использование программных интерфейсов приложений (API) (межплатформенное взаимодействие и переносимость программных компонентов).

❸ Сетевые технологии, в частности, базирующиеся на TCP/IP (взаимодействие сетей с разными протоколами, операционными системами и уровнями взаимодействия: локальные, глобальные и пиринговые).

❹ Беспроводные технологии и их программирование (WAP) (совместное функционирование и обмен информацией между компьютерными и телефонными системами, не связанных линиями коммуникации).

❺ Геоинформационные системы и технологии (ГИС) (глобальное, всемирное, совместное накапливание, обработка, взаимодействие и представление колоссальных объёмов информации, собранной о Земле и инфраструктуре пространственных объектов, расположенных на ней).

❻ Графика и мультимедиа (совместное кодирование, представление, обработка, запись и вывод аудио- и видеоинформации).

❼ Встроенные системы (конструирование и программирование микропроцессоров, встраиваемых в современную технику: автомобили, бытовую, радио-, видео-, электро- и многую другую).

❽ Реляционные, объектно-ориентированные и XML-ориентированные базы данных (базы данных, формируемые и управляемые с помощью разных механизмов представления данных и обработки объектов).

❾ Способность к взаимодействию (интероперабельность – interoperability) (программирование средств обеспечения доступа практически к любым источникам информации: библиотечным ресурсам, компьютерам, радиотелефонам, ноутбукам, локальным сетям, Internet, WWW и др.).

❿ Объектно-ориентированное и компонентно-ориентированное программирование (ООР и СОР) (создание компьютерных систем и языков программирования, обеспечивающих адекватное представление в информационных системах объектов и сущностей реального мира, а также возможности их обработки).

❶❶ Надёжность программного обеспечения (создание механизмов обеспечения бесперебойного выполнения программных процессов представления и обработки данных в компьютерных и информационных системах любого уровня).

❶❷ Анализ и управление контентом (Topic Maps) (разработка и внедрение теоретических методов управления смысловым поиском требуемой информации в WWW и Internet).

❶❸ Безопасность и криптография (защита персональной, корпоративной и информации в Internet от хакерских атак и несанкционированного доступа, а также от компьютерных вирусов любых разновидностей).

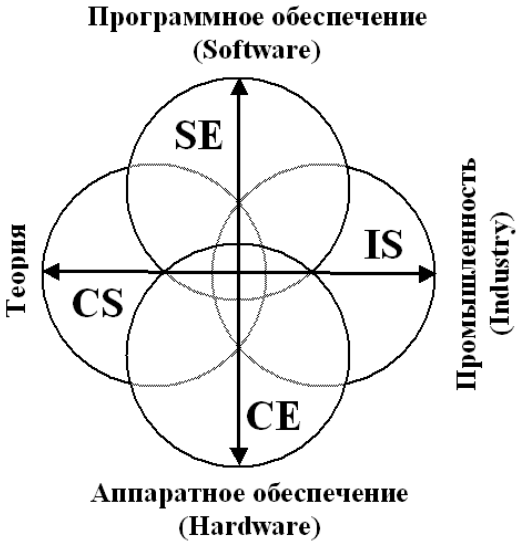


Рис. 1. Пересечение некоторых компьютерных областей: Computer Science (CS), Software Engineering (SE), Information System (IS), Computer Engineering (CE)

Одно из возможных представлений пересечения и взаимодействия наиболее крупных современных компьютерных направлений[2] представлено на рисунке 1. Несомненно, что один из наиболее ёмких секторов рынка компьютерных технологий принадлежит информационным системам, работающим в корпоративном пространстве и опирающимся на научные разработки (CS), программную инженерию (SE) и аппаратную технологическую базу (CE).

Не будет преувеличением сказать, что подавляющая часть тем информатики зиждется на использовании тех или иных языков программирования, как единственно возможном инструменте управления разнообразными компонентами сложных многоуровневых компьютерных систем и самим информационным контентом. И хотя рынок программных продуктов постоянно пополняется всё новыми и новыми приложениями, покрыть поле пользовательских интересов они не в состоянии…

На начальных этапах освоения их возможностей, как правило, пользователь применяет инструменты стандартного интерфейса приложения (API), а затем, освоившись, применяет встроенный язык системы (типа *Visual*

[2] Training Course SE MSF.NET / V. Pavlov, M. Boyko, D. Malenko, O. Biloborod'ko // .NET ***Technologies'2004*** workshop proceedings. Copyright UNIAN Agency – Science Press, Plzen, Czech Republic. –.5 p.

Basic for Application). И действительно, вряд ли Вас надолго увлечёт процесс использования чужих программ (кстати, ведь и их кто-то создавал в своё время!). А ещё большую неудовлетворённость вызывает следование многочисленным руководствам для «чайников» типа: «Подведите курсор мыши к меню окна приложения такого-то, выберите требуемую команду и щёлкните левой кнопкой мыши ОК!». И неизбежно наступит момент, когда Вам самому захочется направить ход действий компьютера по своему усмотрению. А это и будут Ваши первые шаги в программировании и в программирование.

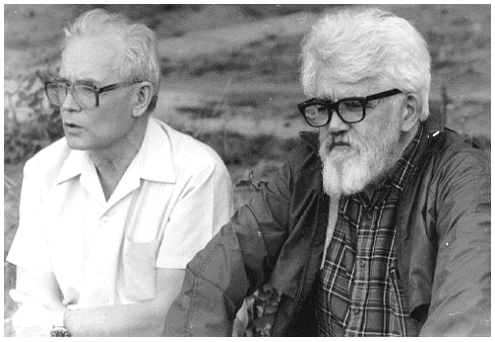


Рис. 2.
Андрей Петрович Ершов (1931-1988), советский математик, академик АН СССР (слева) и Джон МакКарти (отец языка Лисп, США)

Но, прежде чем вступать в эту таинственную и увлекательную область человеческой деятельность, задумайтесь над словами академика А.П. Ершова (рис. 2), сказанными на Объединенной вычислительной конференции Американской федерации обществ по обработке информации в Атлантик-Сити (США) ещё в 1972 г., в его выступлении под названием «О человеческом и эстетическом факторах в программировании»: «Программирование становится массовой профессией. Однако надо иметь в виду, что сейчас это, пожалуй, самая трудная из всех массовых профессий, причем, к сожалению, эта трудность не признана в должной мере».

Эта трудность заключается в том, что именно программисты непосредственно упираются в пределы человеческого познания в виде алгоритмически неразрешимых проблем и глубоких тайн работы головного мозга.

Трудность состоит ещё и в том, что собственный стек программиста должен быть не в 56 позиций глубины, как это обнаружили психологи у среднего человека, а той же глубины, что и стек в его очередной задаче, подлежащей программированию, плюс еще как минимум две-три позиции.

Трудность программирования заключается также и в том, что программист должен обладать способностью первоклассного математика к абстракции и логическому мышлению в сочетании с эдисоновским талантом сооружать все, что угодно, из нуля и единицы. Он должен сочетать аккуратность бухгалтера с проницательностью разведчика, фантазию автора детективных романов с трезвой практичностью экономиста. А, кроме того, программист должен иметь вкус к коллективной работе, понимать интересы пользователя и многое, многое другое».

Это одна сторона дела, но есть ещё и другая. А.П. Ершов обратил внимание присутствующих на очень важный аспект, связанный с проблемами программирования: «Во время пребывания в 1970 г. в Соединенных Штатах на меня произвели очень большое впечатление новые идеи профессоров Массачусетского технологического института Марвина Минского и Сеймура

Пейперта в области обучения детей. Они выбросили в корзину ходячее представление о том, что дети учатся бессознательно методом подражания. Минский и Пейперт доказывают, что человек чему-то научается только в том случае, если у него в голове складывается блок-схема действия, выделены подпрограммы и проложены информационные связи. Профессор Пейперт навсегда обратил меня в свою веру на примере жонглирования двумя мячами, когда, апеллируя к моим способностям программиста, он за десять минут научил меня тому, чего бы я сам не сделал и за несколько часов. Таким образом, человек неизмеримо усилит интеллект, если сделает частью своей натуры способность планировать свои действия, вырабатывать общие правила и способ их применения к конкретной ситуации, организовывать эти правила в осознанную и выразимую структуру, – одним словом, сделается программистом.

Творческая и конструктивная природа программирования не требует особых доказательств. В своей творческой природе оно идет намного дальше большинства других профессий, приближаясь к математике и писательскому делу. В большинстве других профессий мы лишь «приручаем» при помощи сил природы те или иные физические или биологические явления, не обязательно постигая их сущность. В программировании же мы в некотором смысле идем до конца. Один из тезисов современной теории познания: «мы знаем что-то, если можем это запрограммировать» – очень выпукло характеризует этот максимализм программирования.

Рис.3 Интерфейс одной из систем электронного дистанционного образования (e-Learning) в Web-браузере

В этом и заключается фундаментальная роль основных понятий информатики в формировании способностей и интеллекта современного человека и в вытекающей из этой роли необходимости внедрения компьютеров в учебный процесс» (рис. 3).

И если после этого Вы еще не передумали заниматься программированием, то перед Вами встанет вопрос: «По какой книге и какой язык программирования следует в первую очередь изучать»? Ответ неоднозначен, но, к счастью, может быть «разложен на составляющие». Условно его можно разбить на три части.

1. В принципе, программировать можно на любом языке программирования. В классической работе Бома и Джакопини (опубликована в журнале Communications of the ACM в мае 1966 г.) было показано, что для программы, в которой всякий модуль проектировался с единственным входом и

единственным выходом, а программа является множеством вложенных модулей, каждый из которых также имеет один вход и один выход, может быть реализована в языке, включающем только две основные управляющие конструкции. Фактическая реализация этих конструкций может быть различной в различных языках программирования. Принцип, изложенный в статье Бома и Джакопини, называемый «структурной теоремой» имеет фундаментальное значение и составляет основу большинства реализаций алгоритмов при проектировании программ.

По Бому и Джакопини для построения программы требуется три основных составляющих блока (рис.4):

❶ функциональный блок или линейная последовательность операторов;

❷ условный оператор (конструкция принятия двоичного или дихотомичного решения);

❸ конструкция обобщённого цикла (цикл с параметром, с предусловием, с постусловием и т.д.).

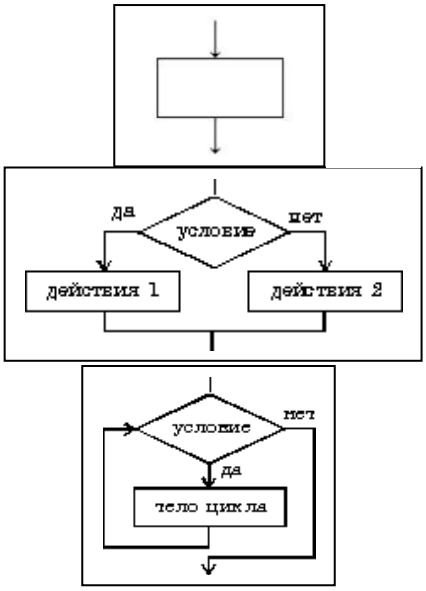


Рис. 4. Три основных блока в программе

Тогда для всякой программы, составленной из указанных конструкций, всегда может быть получено ещё и доказательство её правильности!

2. Если Вы намереваетесь сразу решать конкретные задачи в конкретной организации, то здесь при выборе языка программирования, Вам необходимо (и обязательно придётся!) учитывать массу факторов: накопленный опыт организации, специфику решаемой задачи, имеющееся аппаратное обеспечение, наличие лицензионных программных продуктов, установленные сроки выполнения проекта и т.д., что и позволяет уже остановить свой выбор на подходящих средствах реализации проекта. Тем более, что хорошему программисту для уверенности в завтрашнем дне требуется знание ***нескольких*** языков программирования.

3. Но, если Вы решили начать с освоения систематического и научного подхода к программированию, т.е. научиться реализовывать большие программы со сложными алгоритмами и структурами данных, то, несомненно, Вам нужно начинать с языка Турбо Паскаль.

Сам язык Паскаль появился в Цюрихе в «Школе программирования», во главе которой стоит Никлаус Вирт (1968-1999 гг. – профессор информатики в университете г. Цюрих, Швейцария) (рис. 5), обобщивший свой многолетний авторский опыт обучения программированию. Созданный специально с педагогическими целями Паскаль оказался крайне удачным не только в силу того, что ему просто научиться, но и как основа обсуждения языков программирования вообще. Язык проектировался с учётом простоты написания соответствующего транслятора, а описание Паскаля, занимающее около 30

страниц текста, представляет собой яркий пример систематического и научного подхода к обучению построению программ. Для сравнения попробуйте заглянуть в книгу Бьёрна Страуструпа по C++ и, как говорится, почувствуйте разницу!

Рис. 5.
Никлаус Вирт перед тренировочным полётом на боевом истребителе

Никлаус Вирт первым сформулировал тезис о том, что программы представляют собой, в конечном счёте, конкретные формулировки абстрактных ***алгоритмов***, применяемые к обработке спроектированных структур ***данных***. Это и нашло своё отражение в языке Паскаль. Так, освоив конструкцию описания массива:

```
Array[1..N] of Real,
```

Вы далее узнаете, что конструкция `1..N` представляет собой ограниченный тип и может быть представлена самостоятельным идентификатором (именем, названием), например Border и тогда то же описание будет выглядеть так:

```
Array[Border] of Real.
```

Плавный переход от простых типов данных к составным позволяет научиться проектировать пользовательские типы данных и алгоритмы их обработки, легко перемещаясь по уровням абстракций, что является основным в программировании. Это должно позволить Вам программировать или быстро научиться программировать практически на любом из основных языков программирования, который(е) Вам придется использовать в будущем на рабочем месте.

Важно отметить, что в программировании, в отличие от математики, нет формул, подставив в которые некоторые исходные данные можно получить гарантированный результат и в этом состоит его объективная трудность. Но зато здесь имеется известный набор приёмов и примеров их использования. Фрагменты кода и примеры программ, конечно, многому могут научить, но не следует забывать и о возможностях направлять ход мыслей программиста, т.е. предлагать ему уже апробированные алгоритмы проектирования программ и проектирования алгоритмов. Тем более, что известный во всём мире специалист в области компьютерных технологий Дональд Кнут (США)[3] (рис. 6)

[3] Профессор Компьютерных наук Стэндфордского университета Дональд Кнут (США), обладатель премии Тьюринга в 1974 г., член Британского компьютерного общества с 1980 г., Почётный член организации IEEE с 1982 г. Член Американской академии Искусства и Науки, Национальной американской академии Наук, Национальной Американской Инженерной Академии, зарубежных академий: Французской Академии наук (l'Académie des Sciences (Paris)) и Норвежской Научной академии (Det Norske Videnskaps-Akademi (Oslo)). Опубликовал 160 статей и 19 книг, имеет много научных и государственных наград разных стран мира, является почётным профессором Оксфордского, Парижского и Петербургского университетов, а также многих колледжей мира.

разработал и издал три тома наиболее употребительных вычислительных алгоритмов.

Таким образом, в основе программирования лежит творческий акт и основная задача при обучении состоит в развитии способностей человека творчески мыслить. И здесь нужно не столько рассказывать, сколько показывать, как делается «то» или «это». Здесь большую роль играет выбор материала и подбор примеров, в которых подчёркиваются основополагающие принципы программирования.

Рис. 6. Професор Дональд Кнут (США), разработавший и напечатавший три книги вычислительных алгоритмов для комьютеров

Как тонко заметил Н. Вирт: «Программирование – это искусство конструирования». Конечно, научить конструкторской или изобретательской деятельности весьма трудно. Но можно воспользоваться старым как мир способом. Подобно тому, как ребёнок учится говорить, складывая из букв слова, а из слов предложения, программист «складывает» алгоритмы программных модулей, из модулей – программы, а из программ – системы[4]…

«Вхождение» в Паскаль существенно облегчается наличием компактной интегрированной среды программирования Турбо Паскаль (ТП). Ещё одной из причин начать свои первые шаги в программировании на Турбо Паскале, является его многофункциональная среда разработки, которая включает текстовый редактор для ввода программ, заботливый компилятор, который в случае наличия ошибки в тексте Вашей программы, устанавливает курсор в этом месте и сообщает номер и название этой ошибки, а также и дебаггер, то есть отладчик программ, который позволяет не только отлаживать, но и исследовать написанные Вами программы. Простота его использования позволит Вам сосредоточиться на основных тонкостях программирования в отличие от монстрообразных систем быстрой разработки программ, изучение возможностей которых требует отдельных (а иногда и напряжённых) усилий. Лучше всего эту ситуацию иллюстрирует история, приведенная Джозефом Фоксом в его книге «Программное обеспечение и его разработка»: «Когда в начале 1970-х гг. мы успешно завершили первую рабочую версию новой программной системы управления воздушными перевозками для компьютеров IBM (рис.7), мы испытали чувство огромного удовлетворения (группа из 500-т программистов работала над её созданием 7 лет).

[4] Бортовое программное обеспечение (ПО) ***всех*** российских спутников связи пишется на языке Модула-2, разработанном Н.Виртом в 1979 г.после создания языка Паскаль в 1970 г.

Система была отправлена в Джексонвилл, шт. Флорида, для испытаний в рабочих условиях в ночную смену в Центре управления авиаперевозками. Однако диспетчеры из Джексонвилла отказались пользоваться ею – они заявили, что она «ненадёжна».

«Ненадёжность» диспетчерской авиаслужбы требует особого внимания. Нам удалось решить проблему ***исключением*** из состава системы ***значительной доли функций***, которые ранее были туда включены. Следующий вариант программы, поставленной в Джексонвилле, содержал гораздо меньше системных возможностей. И диспетчерам он понравился. А потом постепенно, очень медленно мы стали добавлять функции, уже запрограммированные и оттестированные нами ранее.

Рис. 7. Компьютер IBM System/360, 1974 г.

Понятно, что если Вам предложат пуд мороженного, трудно ожидать, что Вы сможете проглотить его за один приём. Это не получится».

Вероятно, Вам будет интересно узнать, что авторы книги за свою жизнь «вкусили» все прелести программистского ремесла. Ещё на студенческой скамье в 1970 г. пришлось программировать в машинных кодах для электронной вычислительной машины (ЭВМ) «Урал-1». На преддипломной практике в Институте кибернетики АН УССР им. академика Глушкова программировали в кодах трёхадресной машины М-220. Обучаясь в аспирантуре и затем, работая там же, программировали для ЭВМ БЭСМ-6 на языках Алгол, Фортран и Автокоде «Мадлен», а для ЕС ЭВМ (линия IBM, рис.7) – на языках PL/1, Fortran и Assembler. Уже в эпоху IBM PC, работая в высших учебных заведениях, программировали на языках Турбо Паскаль, Quick Basic, Пролог, Ассемблер, Лисп. В настоящее время предпочтение отдают языкам Турбо Паскаль, C++, Visual Basic for Application, Python. Указанный языковой слалом «выверил и разметил» сам процесс развития информационных технологий, а отнюдь не эксцентричность в избрании подхода к изучению серии языков.

Идея создания этого учебного пособия возникла в процессе общения авторов с преподавателями и студентами Национального горного университета, а также учителями и школьниками Региональных учебных центров НГУ в Павлограде, Комсомольске, Александрии и Днепрорудном в 2003-2004 г.г. Опыт этих встреч показывает, что, несмотря на наличие большого числа учебных пособий и специальной литературы по информатике, изучение языка программирования в отрыве от истории появления персональных компьютеров, основных тенденций развития информационных технологий и изменения парадигмы создания программ, получается очень «плоским». Кроме того, известно, что специалисты, осваивавшие в студенческие годы учебный курс «с

картинками» и «без картинок», обычно принципиально отличаются друг от друга по ***стилю мышления*** в дальнейшем. Поэтому авторы уделили особое внимание подкреплению изложения материала иллюстративными материалами.

Следует также иметь в виду, что определённую трагикомичность ситуации с обучением информатике на начальных курсах вуза придаёт тот факт, что персональный компьютер практически превратился во многих семьях в настольный бытовой прибор для поддержки детских игр (рис.8), а простота инсталляции операционной системы (ОС) Windows (спасибо Биллу Гейтсу и его команде!) позволяет "юным дарованиям" самостоятельно устанавливать эту ОС на ПК, а затем с лёгкостью пользоваться всевозможными игровыми, интернет- и мультимедийными программами, что у простых (и не очень) пользователей рождает эйфорию полного владения информационными технологиями. Но одно дело осваивать буквально в течение десятка минут приложения, которые и ***создавались*** в расчёте на их ***максимальную дружественность*** неподготовленному пользователю, а другое – ***проектировать и разрабатывать такие программы***...

Рис. 8. Фрагмент мультимедийной игры

Основная часть популярной (и очень нужной!) литературы по овладению компьютерной грамотностью направлена на описание аппаратных и программных компонентов компьютерных систем. Стремительное их развитие быстро «старит» излагаемый в них материал. Поэтому в предлагаемом пособии особое внимание было уделено освещению ряда вопросов, которые с одной стороны слабо освещены в известной литературе, а с другой стороны – относятся к разряду фундаментальных (то есть не стареющих).

К примеру, простой вопрос: «Что происходит в процессе включения компьютера?» – вызывает сильное замешательство и ставит в тупик многих юных компьютерщиков. Поэтому авторы почли своим долгом заглянуть внутрь персонального компьютера и снабдили книгу подробным глоссарием, содержащим более 1000-и терминов из области компьютерных технологий, без которого многие вещи ускользают из поля зрения при изучении глубин и основ информатики.

И до сих пор продолжается нестихающая полемика о том, «что» и «как» следует в первую очередь излагать в информационных курсах (см. Приложение 1), поэтому на наш взгляд тема эта неиссякаема. Если же говорить

в целом о цели данного пособия, то кроме непосредственных задач обучения программированию авторы стремились показать становление информатики не как нечто «данное свыше», а как борьбу идей, которые выдвигали и отстаивали живые и несомненно чрезвычайно талантливые люди.

Конечно, ряд глав покажется при первом прочтении достаточно сложными для начинающих. На первых порах их можно пропустить и перейти непосредственно к теме, освещающей особенности языка Турбо Паскаль. Но, возвращаясь снова и снова к не во всём понятным главам, Вы убедитесь, что многие вещи не так трудны, как это кажется на первый взгляд.

Поскольку учебное пособие не монография, ссылки на литературу в тексте не приводились, а сам список литературы разбит по главам. За небольшим исключением, в большинстве своём, это любимые книги авторов, которые хочется перечитывать неоднократно. Несмотря на относительную молодость информатики как науки, в ней уже имеются свои классики. Их книги написаны живым и образным языком, полны реальных программистских историй, которые составляют фольклор профессии. Без этого фольклора бывает трудно понять многие тонкости ремесла программиста. Надеемся, что они вызовут и у читателей соответствующий интерес. Учитывая широту «разброса» освещаемых вопросов и достаточно высокий уровень абстракций, заключённых в некоторых представлениях, в таких случаях ссылки на соответствующие источники и понятия даются непосредственно в сносках к тексту.

Рис. 9. Один из первых персональных компьютеров, созданный и выпускаемый в виде чемодана компанией Osborne Computers (США). Вес в собранном виде 10 кг (1981 г.).

Предлагаемое учебное пособие начинается главой о рождении персонального компьютера по нескольким причинам (рис.9). Во-первых, мы считаем её весьма увлекательной и полезной для вхождения в атмосферу стремительного развития компьютерных технологий, где раздел о процессе включения ПК служит мостиком к пониманию сложности решённых при этом задач и поводом углубиться в фундаментальные вопросы обработки данных и командной основы работы компьютера. Во-вторых, акцент везде делается на фокусирование Вашего внимания на точные формулировки в сфере существующего многообразия в определении многих терминов, обилие которых вызвало появления множества компьютерных словарей и Web-сайтов, толкующих многочисленные неологизмы, которые возникают лавинообразно…

Знание и представление значений постоянно возникающих терминов нужно в первую очередь потому, что как заявил руководитель японских программистов на одной из конференций в 1977 г.: «Наиболее важным языком, который нужно знать программисту, является не JCL или ПЛ/1, а японский язык». Комментарии, как говорится, излишни! (Хотя знания, в том числе и языка UML отнюдь не помешают (рис.10).

Во второй главе описываются основные тенденции в развитии информационных технологий и в относящихся к ним фундаментальных представлениях. Вам предлагается погрузиться в глубины разнообразия понятия интерфейса и рассмотреть, как он менялся от DOS к Windows…

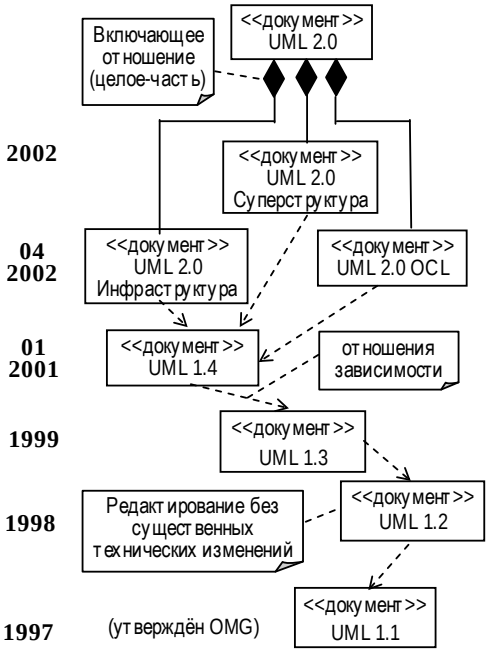


Рис. 10. Диаграмма, описывающая развитие языка UML с 1997 г. по 2002 г., выполненная на самом языке UML

Другая тема, которая заставит Вас по новому взглянуть на изучение Турбо Паскаля, это развитие языков программирования: какие из них и где применяются и в чём их принципиальное отличие, сколько их нужно знать и т.п. Разрабатывая свои первые программы на ТП, Вы должны уметь увидеть перспективы развития своих усилий в области разработки программных систем. Другой, на наш взгляд редко освещаемой темой, является проблема разнообразия типов приложений и компьютерных архитектур, в рамках которых они будут функционировать после создания. А раздел о языке UML нацелит Вас на освоение представлений этого важного средства моделирования процесса разработки программных систем, опирающегося на объектно-ориентированные представления реального мира для организации своего мышления и деятельности в области проектирования и разработки ПО.

В главах, посвящённых особенностям языка Турбо Паскаль и программирования на нём, основное внимание было сосредоточено:

–на понимании абстракций, заключённых в представлениях разных типов чисел, не имеющих аналогов в других сферах деятельности человека (к примеру, где в окружающей жизни можно встретить такое число: ***-2.34e-12***);

–на взаимодействии уровней абстракций при использовании логических: ***значений, переменных, операций, выражений*** и т.д.

–на алгоритмических особенностях конструирования программ вычисления бесконечных рядов и произведений;

–на уникальных возможностях исследования функционирования программ и их фрагментов средствами дебаггера среды Турбо Паскаль;

–на особенностях практического применения в программах сложных типов данных: массивов, записей, файлов, множеств, очередей и ряда других.

В конце каждой главы для закрепления знаний по основным вопросам и самоконтроля приводятся вопросы и задания. На наш взгляд, полезно выполнение не только заданий, но и примеров из глав, посвящённых языку Турбо Паскаль, поскольку в программировании важно не только созерцать фрагменты кода, а проделывать всё «своими руками». Конечно, на первых порах, не всё у Вас будет получаться, но программист мужает именно в борьбе с компилятором…

Несколько слов о Приложениях. Сюда вынесены вопросы преобразования чисел из десятичной системы в двоичную, восьмеричную, шестнадцатиричную и обратно. Здесь также располагается таблица характеристик всех наиболее известных языков программирования. Отдельно приводятся определения названий сетей разных уровней, которые повсеместно применяются, но в таком сочетании в известных авторам источниках не объединялись. Несколько частей приложений отведены для справочных сведений о характеристиках системы Турбо Паскаль (сочетаниях клавиш и типах применяемых данных), а также таблица кодов ASCII.

Рис. 11.
Фредерик П. Брукс

Особо следует выделить глоссарий, являющийся одним из главных компонентов пособия. Собранный по крупицам из материалов различных и многочисленных отечественных и зарубежных источников, он может также служить самостоятельным инструментом по оказанию помощи начинающим по проникновению в мир стремительно развивающихся информационных технологий. Кроме того, в нём собран ряд редких и/или многозначных терминов, что углубляет представления о компьютерном мире. Более глубокое знакомство с глоссарием кроме всего прочего, позволит лишний раз убедиться в глубоком проникновении англоязычных фундаментальных информационных терминов не только в наш язык, но и в наши абстрактные представления об окружающем нас мире и его проявлениях (о чём заявляют даже продвинутые специалисты)[5].

Завершая введение, хочется вспомнить слова архитектора и руководителя разработки одной из крупнейших программных систем – операционной системы OS/360 – профессора университета Северной Каролины (США)

[5] Terminology, and naming things in general, is always difficult. Web Services Architecture Working Group (http://www.w3.org/2002/ws/arch/). Дословно: «Создание терминологии и именование сущностей вообще и всегда протекают чрезвычайно трудно», Документ Рабочей группы Архитектуры Web-сервисов организации W3C, 2002 г.

Фредерика П. Брукса (рис. 11), автора произведения, ставшего классикой, под названием «Мифический человеко-месяц или как создаются программные комплексы[6]»: «Почему программирование доставляет удовольствие? Как вознаграждаются все усилия профессионала?».

Потому, пишет Ф.П. Брукс, что это удовольствие работать с очень гибким материалом. Программист, как и поэт, работает почти исключительно головой. Он строит свои зáмки в воздухе и из воздуха только силой своего воображения. Очень редко материал для творчества допускает такую гибкость, такую возможность столь частых улучшений и переделок и такими простыми средствами позволяет осуществлять громадные замыслы.

Материал поэта – слова, и результат – те же слова; но в отличие от стихотворца, программист создаёт программный продукт, реальный в том смысле, что сам программист движется и работает, производя видимый результат, отличный от него самого. Он печатает результаты, чертит рисунки, производит звуки, управляет движением руки. Волшебство мифов и легенд стало явью в наши дни. Вы печатаете на клавиатуре заклинание, и вот экран дисплея оживает, показывая объекты, которых не было и могло не быть никогда!

Программирование доставляет нам радость. потому что позволяет удовлетворить стремление к творчеству, глубоко заложенное в каждом из нас, и разделить это чувство радости с другими».

Поскольку это учебное пособие является практическим руководством, в нём содержится множество примеров с пошаговыми инструкциями и подробными указаниями. При этом используются следующие соглашения.

Слова, на которых мы хотим акцентировать Ваше внимание, будут выделяться ***жирным курсивом***.

Если необходимо нажать клавишу **Ctrl** или **Alt** одновременно с какой-либо клавишей, названия этих клавиш в тексте объединяются знаком «плюс». Например, «Компилируйте и запускайте свою программу, используя **Ctrl+F9**». Такой же шрифт используется для выделения элементов интерфейса интегрированной среды разработки Турбо Паскаль, например, «**Save file as**».

Фрагменты кода программ и идентификаторы в учебном пособии напечатаны моноширинным шрифтом, например «`Program MyFirst;`».

Литература к Введению

1. Рекомендации по преподаванию информатики в университетах: Пер. с англ. –СПб.: 2002. –372с. (Русский перевод образовательного стандарта «IEEE/ACM Computing Curricula 2001: Computer Science»: WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://se.math.spbu.ru/cc2001/)

2. Преподавание информатики: потерянная дорога. Никлаус Вирт. Приветствие на открытии Международной конференции по преподаванию

[6] Только в Росси эта книга выдержала девять (!) изданий с 1979 г. по 1999г..

информатики ITiCSE. г. Аархус (Дания), 24 июня 2002 г. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL:
http://www.inr.ac.ru/~info21/greetings/wirth_doklad_rus.htm

3. О человеческом и эстетическом факторах в программировании. Академик А.П. Ершов. Статья написана на основе речи автора, произнесенной на Объединенной вычислительной конференции Американской федерации обществ по обработке информации (Атлантик-Сити, США, 16-18 мая 1972 г.). WEB-сайт (Электронн. Ресурс) / Способ доступа: URL:
http://www.uriit.ru/portal/jsp/Pages/RUSSIAN/Structure/centr/cpress/pages/inform/5.htm

4. А.П.Ершов «Программирование – вторая грамотность». Брошюра (Препринт № 293). Дата: 01.04.81. Ротапринт ВЦ СО АН СССР, Новосибирск 90. -19с.

5. Böhm C., Jacoipini G. Flow Diagrams, Turing Machines, and Languages with Only Two Formulation Rules. Communications of the ACM, May 1966.

6. Йенсен К., Вирт Н. Паскаль. Руководство для пользователя и описание языка: Пер. с англ. – М.: Финансы и статистика, 1982. – 151 с.

7. Вирт Н. Алгоритмы + структуры данных = программы: Пер. с англ.. – М.: Мир, 1985. – 406 с.

8. Программирование в среде Turbo Pascal 7.0. / Марченко А.И., Марченко Л.А. / Под ред. В.П. Тарасенко: – 5-е изд. перераб. и доп. – К.: ВЕК+, 1999. –464 с.

9. Фокс Дж. Программное обеспечение и его разработка: Пер. с англ.. – М.: Мир, 1985. – 368 с.

10. Страуструп Б. Язык программирования С++, спец. изд.: Пер. с англ.. – М.; СПб.: «Издательство БИНОМ» – «Невский Диалект», 2002. – 1099 с.

11. Фредерик Брукс. Мифический человеко-месяц или как создаются программные комплексы.: Пер. с англ. — СПб.: Символ-Плюс, 1999, – 304 с.

12. Йодан Э. Структурное проектирование и конструирование программ: Пер. с англ. – М.: Мир, 1979. – 415 с.

13. Громов Г.Р. Национальные информационные ресурсы: проблемы промышленной эксплуатации. – М.: Наука, 1984. – 240 с.

14. Виноградова Н.В. Русский компьютерный сленг. // Энергия, 2003, №9. –С.65-68.

15. Дал О., Дийкстра Э., Хоор К. Структурное программирование: Пер. с англ. –М.: Мир, 1975. –256 с.

16. Дийкстра Э. Дисциплина программирования: Пер. с англ. / Под ред. Э.З. Любимского. –М.: Мир, 1978. –278 с.

17. Лингер Р., Миллс Х., Уитт Б. Теория и практика структурного программирования: Пер. с англ.. –М.: Мир, 1982. –406 с.

Бесполезны такие произведения, которые не возбуждают стремление к подражанию: доблестными делами люди не только восхищаются, но и желают подражать тем, кто их совершает.

Плутарх (46 – 120 р.н.е.)

1. ДОЛГИЙ ПУТЬ К ПЕРСОНАЛЬНОМУ КОМПЬЮТЕРУ

1.1 Источники "информационного взрыва"

За точку отсчета развития человеческой цивилизации обычно принимают время, когда люди начали создавать предметы труда и охоты. Тайна "похищения огня" теряется в веках, но вся последующая история технического прогресса от овладения огнём до открытия ядерной энергии - это история последовательного подчинения человеку всё более могущественных сил природы: тягловые животные, ветряные и водяные двигатели, тепловые двигатели, атомная энергетика. Стоящая перед человеком задача была предельно проста: умножать различными инструментами и машинами мускульную силу человека. В то же время попытки создания инструментов, усиливающих природные возможности человека по обработке информации, начиная с камешков абака и до механической машины Беббиджа измерялись отдельными всплесками человеческих судеб, фактов, музейных механизмов и устройств, настойчиво предвещавших невиданное развитие этой отрасли вознесения человека на новые вершины знаний.

Рис. 1.1. Клинопись на глиняной табличке

На самых ранних этапах формирования трудовых коллективов для синхронизации выполняемых действий человеку потребовались кодированные сигналы общения, сложность которых быстро возрастала с повышением сложности трудового процесса. Эту задачу человеческий мозг решал эволюционно без каких-либо искусственно созданных инструментов: просто развивалась и постоянно совершенствовалась человеческая речь.

Речь и оказалась первым носителем человеческих знаний, накапливающихся в устных рассказах и преданиях, передававшихся из поколения в поколение. Первой технологической поддержкой этого важного процесса явилось создание письменности. Начатый тогда процесс поиска и совершенствования носителей информации (а также инструментов для её регистрации) продолжается до сих пор: камень, кость, дерево, глина (рис.1.1), папирус, шёлк, бумага,

люминофор, магнитные и оптические носители информации и т.д.

Одновременно с розвитием процесса накопления знаний в человеческом обществе шёл процесс формирования обособленной профессиональной группы, для которой сначала основным, а потом и единственным "служебным занятием" становится работа с информацией. Этот живой барьер начал разрушаться только после изобретения книгопечатания и овладения обществом новой технологией - ***грамотностью***. В середине XV в. почти одновременно И. Гутенберг (рис.1.2) в Германии и И. Фёдоров в России изобрели печатный станок, который ускорил темпы накопления профессиональных знаний.

Рис. 1.2. Изобретатель первого печатного станка И. Гутенберг (1397-1468)

По подсчетам науковедов, общая сумма человеческих знаний изменялась раньше очень медленно: в 1800 г. она удваивалась каждые 50 лет, к 1950 г. удваивалась каждые 10 лет, а к 1970 г. - каждые 5 лет. Это явление, получившее название "информационный взрыв", указывается среди симптомов, свидетельствующих о начале века информатизации и включающих:

❶ быстрое сокращение времени удвоения объема накопленных научных знаний;

❷ превышение материальными затратами на хранение, передачу и переработку информации аналогичных расходов на энергетику.

И если в начале XX века экономическая мощь государств измерялась ***материальными ресурсами***, то в его конце впервые в истории человечества основным ***предметом труда*** в промышленно розвитых странах стоновится ***ИНФОРМАЦИЯ***. Инструментом для работы с новым предметом труда и стал персональный компьютер, положивший начало освоения обществом новой технологии – ***компьютерной грамотности***.

1.2. Компьютер: от идеи – до реализации

Существует множество версий истории появления персонального компьютера. Иначе и быть не могло. Ведь путь к нему вместил в себя массу поисков и открытий на протяжении многих веков. И всё же, попытаемся кратко остановиться на некоторых вехах зарождения компьютера.

Из всех изобретателей счетных машин прошлого, наверное, ближе всего к формулировке основных принципов построения и организации вычислительного процесса в компьютере в современном его понимании подошел англичанин Чарльз Бэббидж (рис. 1.3). В своё время он прославился не только остротой ума, но и своими чудачествами. Бэббидж был одним из основателей Королевского астрономического общества, автором всевозможных сочинений на самые различные темы – от политики и математики до

технологии производства. В течении 13 лет этот эксцентричный гений заведовал кафедрой математики Кембриджского университета, хотя ни разу не появился в нём и не прочитал там ни одной лекции.

Рис. 1.3.
Чарльз Беббидж
(1791-1871 р.)

Наивысшим достижением Чарльза Бэббиджа была разработка принципов действия и устройства компьютера, за целое столетие до того, как появилась техническая возможность его реализации. Один из создателей первого американского компьютера "Марк-1" молодой гарвардский математик Говард Эйкен говорил в 1943 году, что для него описания Аналитической машины, оставленного самим Бэббиджем, оказалось более чем достаточно. "Если бы Бэббидж жил на 75 лет позже, – заявил он, – я бы остался без работы".

Энергия, авторитет и умение убеждать позволили Чарльзу Бэббиджу добиться в 1822 году государственного финансирования разработки и создания «Разностной машины», предназначенной для расчетов и печати больших математических таблиц. Осуществлению этого проекта все свои незаурядные математические и литературные способности посвящала урожденная Огаста Ада Байрон графиня Лавлейс (рис. 1.4).

Рис. 1.4. Первая женщина-программист, графиня Ада Лавлейс

Говоря об «Аналитической машине», Бэббидж отмечал: "…графиня Лавлейс, по-видимому, понимает её лучше меня, а уж объясняет её устройство во много-много раз лучше". Графиня Лавлейс помогала Бэббиджу прояснять его собственные идеи, воодушевляла его, глубоко интересуясь его работой и заражая своим энтузиазмом. Она прекрасно поняла революционную сущность разрабатываемой машины и то, что это действительно был «математический станок», изначально как-бы бесмысленный, но способный выполнить любую программу, переведенную на язык перфокарт. Именно поэтому, в её честь был назван один из известнейших и поныне языков программирования реального времени ADA.

Несмотря на то, что результат не был достигнут, в процессе работы Ч. Бэббидж пришел к идее создания еще более мощной машины. Его новое детище– «Аналитическая машина», в отличие от своей предшественницы, должна была не просто решать математические задачи определенного типа, а выполнять разнообразные вычислительные операции в соответствии с инструкциями, задаваемыми оператором. По замыслу это была

"машина самого универсального характера" – в действительности не что иное, как первый универсальный программируемый компьютер. К тому времени, когда в 1833 году расходы на постройку «Разностной машины» достигли 17 000 фунтов стерлингов, терпение британских чиновников иссякло и государственное финансирование было прекращено. Ведь если «Разностная машина» имела сомнительные шансы на успех, то «Аналитическая» – и вовсе выглядела фантастикой. Размером с железнодорожный локомотив, конструкция представляла собой внушительное нагромождение стальных, медных и деревянных деталей, часовых механизмов, приводимых в действие паровым двигателем. Малейшая нестабильность какой-нибудь крошечной детали приводила бы к стократно усиленным нарушениям в других частях механизма, что делало нереальным этот проект. Ведь, как известно, для реализации научно-технической идеи требуется выполнение по крайней мере трех основных условий:

1. Идея не должна противоречить известным законам науки.
2. В её реализации должна быть остро заинтересована значительная часть общества (иными словами, должен "созреть" социальный заказ).
3. Должен быть достигнут тот уровень технологии и общественного производства, который обеспечивает эффективную реализацию не только конкретных элементов, но и других заложенных в идею технических принципов..

Лишь к середине XIX в. появляются открытия и изобретения, приближающие рождение компьютера. Уильям Остин Барт первым в Америке (1829 г.) получает патент на неуклюжую, но работоспособную пишущую машинку, не вызвавшую тогда интереса у промышленников. А знакомая сейчас любому пользователю компьютерная клавиатура с раскладкой QWERTY (так на ней расположены первые шесть латинских букв), появилась лишь в 1874 году. Но именно с нее началось победное шествие этого шумного орудия труда по столам секретарш малых и крупных организаций. Кстати, клавишу Shift, для переключения верхнего и нижнего регистров, добавили к пишущим машинкам только в 1878 году, а до того времени заглавные буквы располагались на клавиатуре отдельно.

Рис. 1.5. Электронная лампа

Экспериментируя с электрической лампой в 1883 г., Эдисон вводит в вакуумный баллон дополнительный платиновый электрод, подает напряжение и, к своему удивлению, обнаруживает, что между электродом и угольной нитью протекает ток. Поскольку в тот момент главной целью Эдисона было продление срока службы лампы накаливания, этот результат его заинтересовал мало, но патент предприимчивый американец все-таки получил. Явление, известное нам как термоэлектронная эмиссия, тогда получило название "эффект Эдисона" и на какое-то время забылось. В электронную лампу его идея материализовалась позднее (рис. 1.5).

Социальный заказ на компьютеры созрел в разгар второй мировой войны –

требовались сложные артиллерийские баллистические расчеты. И вот, с благословения командования военно-морского флота США, при финансовой и технической поддержке фирмы IBM, началась разработка вычислительной машины, в основу которой легли непроверенные идеи XIX в. и надежные технологии XX в.

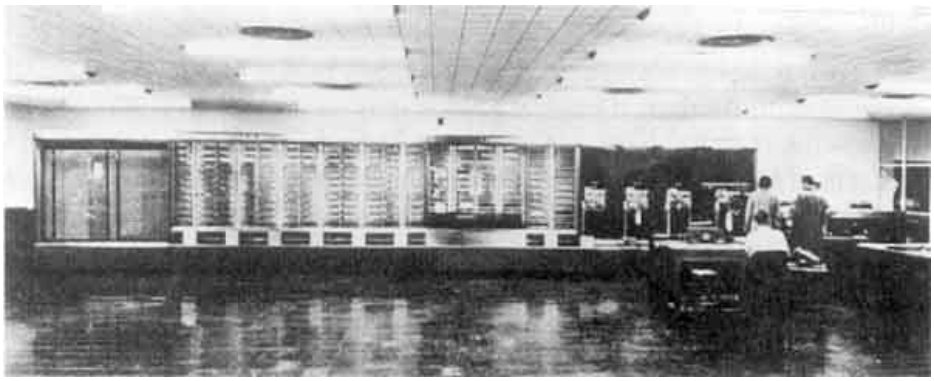


Рис. 1.6. Машина "Марк-1"

В начале 1943 года начало свою работу творение Говарда Эйкена – вычислительная машина "Марк-1" (рис. 1.6), достигавшая в длину почти 17 м и в высоту более 2,5 м, содержащая 750 000 деталей, соединенных проводами общей протяженностью около 800 км. "Марк-1" мог "перемалывать" числа длиной до 23 разрядов (рис. 1.7). На операции сложения и вычитания тратилось 0,3 с, а на умножение 3 с. За день машина выполняла вычисления, на которые раньше уходило полгода (!). В ней, как и в ее предшественницах, использовались простые электромеханические переключатели (реле), широко применяемые в то время в телефонной связи, а команды машине в виде программ набивались на перфоленте (рис. 1.8).

24.712946369421639419437

Рис. 1.7. Двадцатитрёхразрядное дробное десятичное число (т.е. число, которое содержит двадцать три десятичных цифры)

Война продолжалась, военные разработки требовали ускорения и за дело взялись сотрудники вычислительного центра Пенсильванского университета Джон У. Мочли и Джон Преспер Экерт. Их усилия не остались незамеченными и 9 апреля 1943 года армия США заключила с университетом, где они работали, контракт на сумму в 400 000 долл., предусматривающий создание лампового компьютера "Эниак" (рис. 1.9).

Рис. 1.8. Перфолента

В конце 1945 года 30-тонный гигант был собран и продемонстрирован представителям прессы. По своим размерам (около 6 м в высоту и 26 м в длину) этот компьютер более чем вдвое превосходил по быстродействию "Марк-1" Говарда Эйкена и стоил 500 000 долл. Одновременно работавшие 17 000 электронных ламп увеличили его быстродействие в 1 000 раз! Это был прорыв.

Дальнейшее совершенствование компьютерной техники пошло по пути повышения надежности работы, миниатюризации электронных устройств и сокращения энергозатрат. Инженеры из фирмы Bell Labs Уильям Шокли, Джон Бардин и Уолтер Брэттен в 1947 году изобрели транзистор (рис. 1.10). Заметка в New York Times про это событие была помещена 1 июля 1948 года в самом конце небольшого раздела "Новости радио" рядом с объявлением о времени трансляции передачи "В ритме вальса". После почти трехлетних исследований, потребовавших около миллиона долларов, фирма Bell Labs получила первый полупроводниковый усилитель, кстати, не произведший особого впечатления на представителей средств массовой информации.

С появлением транзисторов ламповые компьютеры морально устарели. В 1959 году фирма IBM создала свой первый полностью транзисторный большой универсальный компьютер (мэйнфрейм), способный выполнять 229 тыс. арифметических операций в секунду. Такие мэйнфреймы позволили военно-воздушным силам США создать систему раннего предупреждения о нападении баллистических ракет. В 1964 году на основе двух мэйнфреймов модели 7090 американская авиакомпания SABRE впервые применила автоматизированную систему продажи и бронирования авиабилетов в 65 городах мира.

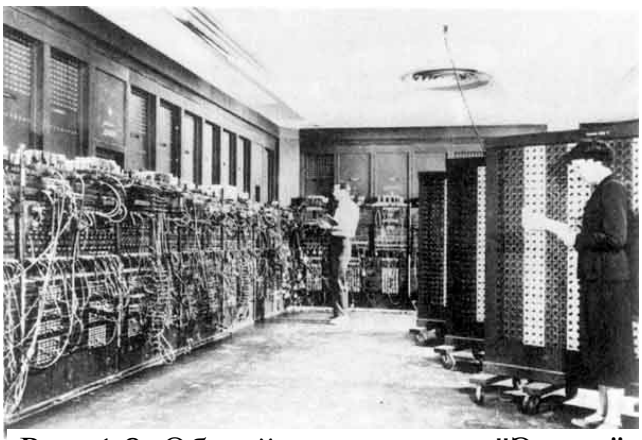


Рис. 1.9. Общий вид машины "Эниак", 1946 г.

Первые германиевые транзисторы стоили до 8 долл. за штуку, в то время как цена лампы не превышала 75 центов, так как германий был и остаётся довольно редким элементом, получаемым по особой технологии и по себестоимости дороже золота. Однако, в 1954 г. американский физик Гордон Тил изготовил транзистор из кремниевого кристалла, вместо германиевого. Это был очередной шаг вперёд, так как кремний самый распространенный на Земле (после кислорода) химический элемент. Совершенствование технологий производства транзисторов тоже способствовало снижению их стоимости.

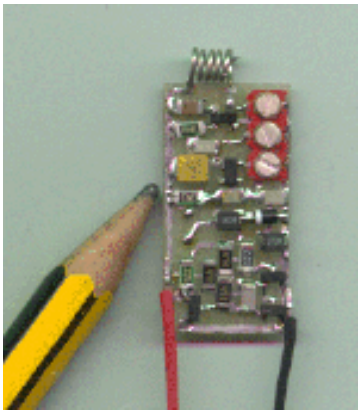


Рис. 1.10. Плата с транзисторами, 1947 г.

А интенсивные исследования продолжались. Гениальная идея пришла в голову уроженцу Канзаса Джеку Килби. Решение возникло в июле 1956 г., когда сотрудники компании Texas Instruments отправились в двухнедельный летний отпуск, а он как новичок, не заслуживший на это права, остался в лаборатории практически один. Килби понял: элементы электрических схем

резисторы и конденсаторы можно делать тоже не только из того же материала, что и транзисторы, но и изготавливать все компоненты на одной полупроводниковой пластине. После нескольких месяцев длительных разъяснений он убедил в правильности своей идеи скептически настроенного шефа, изготовив первый опытный образец. Первая в мире интегральная схема (ИС) представляла собой тонкую германиевую пластинку площадью в 1 $см^2$. Фирма сообщила о рождении нового устройства в январе 1959 г. А чтобы продемонстрировать потенциальные возможности новой техники, компания построила для военно-воздушных сил США компьютер объёмом около 40 $см^3$ на 587 ИС. Его размеры оказались в 150 раз меньше, чем у машин старого образца.

1.3. Скачок в развитии вычислительной техники

В конце 60-х годов большая группа талантливых инженеров-электроников, покинув фирму Fairchild Semiconductors, создали более 50-ти компаний для разработки и создания ***интегральных схем*** (ИС). На то время ИС, вскоре прозванные "чипами", представляли собой кристаллы кремния, содержащие все необходимые для работы электронных схем компоненты: транзисторы, резисторы и конденсаторы. По мере уменьшения размеров отдельных компонентов на кристалле, количество их на одном чипе возрастало с невероятной быстротой, примерно удваиваясь каждый год. Например, в 1964 году на кристалле размером 7 $см^2$ умещалось 10 транзисторов и других компонентов, а к 1970 г. в кристалле того же размера содержалось уже не менее 100 элементов приблизительно при той же стоимости ИС.

Рис. 1.11. Основатели фирмы Intel

Поэтому неудивительно, что бизнес-план объемом в одну страничку, который собственноручно напечатал Боб Нойс в 1969 г. на "текстовом процессоре" того времени – пишущей машинке – вызвал нешуточный интерес у Артура Рока, финансиста из Сан-Франциско. Полагаясь на высочайшую репутацию основателей фирмы Intel Энди Гроува, Гордона Мура и Роберта Нойса (рис. 1.11) и их глубокие знания, мистер Рок в течение двух дней предоставил им необходимые деньги – ни много, ни мало – два с половиной миллиона долларов!

Корпорация Intel начинала свою деятельность с изготовления микросхем оперативной памяти. Первым серийным изделием Intel была "микросхема 3101" – 64-разрядная статическая оперативная память. Это изделие имело некоторый успех. Однако надо признать: то особое место, которое Intel заняла в компьютерной индустрии нынешнего времени связано с совсем другими микросхемами (в 2001 году в ежегодно публикуемом списке 500 крупнейших компаний США американского журнала Fortune

компания Intel занимала 41-ое место с годовой прибылью в 33 млрд. долларов). По данным же 1997 года, фирма Intel произвела кроме процессоров, ещё и системных плат для ПК больше, чем все остальные восемь их призводителей вместе взятых, общим числом 30 млн. штук на сумму более $3,6 млрд. Эти цифры означают, что каждые восемь персональных компьютеров (ПК) из десяти в настоящее время содержат материнские платы производства Intel.

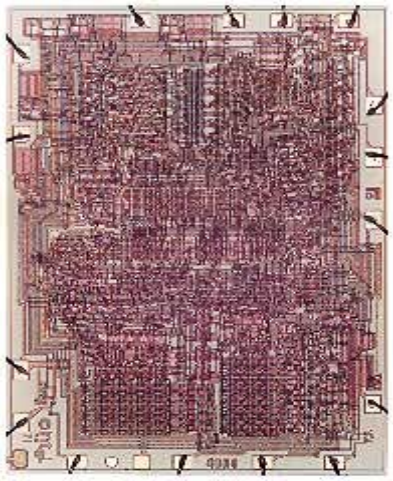


Рис. 1.12.
Внешний вид ИС Intel 4004

А все началось с одной замечательной идеи инженера Intel по имени Тед Хофф. Вместо двенадцати специализированных микросхем, заказанных японской фирмой Busicom для карманных калькуляторов, создать одну универсальную, которая смогла бы их заменить. Как оказалось, это решение изменило историю, позволив сделать программируемый интеллектуальный процессор настолько дешевым, что его можно было применять для бытовых целей, и столь универсальным (мощным), что каждый мог иметь свой индивидуальный (персональный) компьютер.

Через 9 месяцев напряженной работы в 1971 году появился первый в мире микропроцессор – Intel 4004 (рис. 1.12). В нем содержалось 2 300 полупроводниковых транзисторов, и он свободно умещался на ладони. Крошечный процессор 4004 имел такое же быстродействие, как огромный компьютер ENIAC, занимающий объем 85 кубических метров и имевший 17 000 вакуумных ламп (сравните рисунок 1.9 и рисунок 1.12).

В этот момент руководство Intel поняло, что сделано выдающееся изобретение. Но, к несчастью, авторские права на это изобретение принадлежали не им, а фирме Busicom (Япония), заказавшей и оплатившей расходы на разработку. В результате переговоров Intel выкупил у Busicom все права на микропроцессор 4004 за 60 тысяч долларов. "Теперь вопрос состоял в том, сможем ли мы продавать процессоры кому-либо, кроме Busicom... и найдется ли достаточно много других применений для этих микросхем, чтобы они стали приносить заметную прибыль", – вспоминал впоследствии Роберт Нойс

Фирме Intel пришлось взяться за непростую задачу: разъяснить инженерному миру огромный потенциал использования программируемых микропроцессоров во множестве областей. Компании пришлось проводить семинары для инженеров, публиковать рекламные объявления и продавать справочные руководства по микропроцессорам. Один из сотрудников Intel вспоминал: "Бывали недели, за которые Intel продавал больше справочной документации, чем самих микропроцессоров". Одновременно совершенствовалась и конструкция самого чипа.

В 1974 году появилась новая модель – Intel 8008. Мощность этого

процессора, по сравнению с его предшественником, возросла вдвое. По сообщению журнала Radio Electronics, известный энтузиаст вычислительных технологий Дон Ланкастер применил процессор 8008 в разработке прототипа персонального компьютера – устройства, которое упомянутый журнал назвал "гибридом телевизора и пишущей машинки". Использовалось оно в качестве терминала ввода-вывода.

Рис. 1.13.
Первый ПК Altair

И только следующий шаг оказался самым удачным. Процессор Intel 8080 1974 г. выпуска стал "мозгом" первого персонального компьютера Altair (рис. 1.13), названного по имени звезды, к которой был запущен межпланетный корабль Энтерпрайз из американского телесериала "Космическая одиссея". По нынешним меркам его параметры были смехотворны – оперативная память имела объём всего 256 байт, а экрана не было и в помине. Более того, машина не имела ни клавиатуры, ни других внешних устройств. Тем не менее, десятки тысяч экземпляров комплекта для самостоятельной сборки Altair, по цене $397, разошлись всего за несколько месяцев. И хотя его возможности были довольно ограничены, на только что появившемся рынке ПК впервые образовался дефицит. Из соображений экономии многие покупатели приобретали компьютер в виде набора деталей, а затем собирали его собственными силами. Чтобы собранный таким образом компьютер работал как надо, от его владельца требовались немалые познания и практические навыки в электронике.

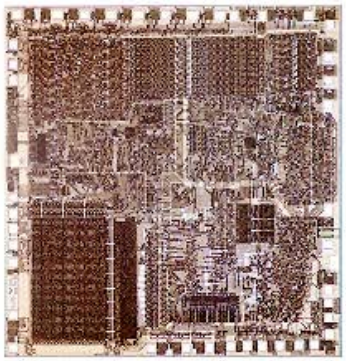


Рис. 1.14.
Intel 8088

При работе пользователи вводили программы и данные в двоичной форме, щелкая набором маленьких ключей, которые могли занимать два положения – верхнее и нижнее (что соответствовало двоичным кодам 1 и 0). Результаты считывали также в двоичных кодах – по светящимся и темным лампочкам. Большинство покупателей заказывали компьютерный набор по почте, узнав о нём лишь из журнальной статьи. Их не пугали технические трудности и недостатки Altair. Энтузиасты писали собственные программы для машины и дополняли её различными устройствами.

1978 год ознаменовался для фирмы Intel выпуском процессоров-бестселлеров – Intel 8086 и 8088 (рис. 1.14). К тому времени, образованная в 1924 г. Германом Холлеритом фирма IBM (International Business Machines), занималась производством крупных вычислительных машин для экономических и научных расчётов. В 1969-71 годах компьютеры IBM обеспечивали высадку американских астронавтов на Луну, а в 1973 году фирма

выполнила заказ NASA на поставку компьютерного оборудования для программы «Союз-Аполлон». Впоследствии IBM принял участие и в программе полетов космических челноков «Шаттл». К концу 1960-х годов IBM господствовала на рынке электронных вычислительных машин, где объем сбыта ее продукции превышал 3 млрд. долларов ежегодно. В 2001 году в списке журнала ″Fortune″ 500 крупнейших фирм США IBM занимала почетное 8-е место с доходом 88 млрд. долл. в год.

Рис. 1.15. IBM PC XT, 1981 г.

В 1980 году руководство IBM обратило внимание на зарождающийся рынок микрокомпьютеров и приняло революционное решение о создании собственной модели персонального компьютера. При конструировании был применен принцип открытой архитектуры: составные части были универсальными, что позволяло модернизировать компьютер по частям. С целью уменьшения затрат на создание персонального компьютера IBM использовала для своего детища разработки других фирм, в частности, программное обеспечение фирмы Microsoft и микропроцессор фирмы Intel. "Мозгом" вновь созданного ″хита″ сезона – IBM PC XT (рис. 1.15) стал процессор 8088, крупную партию которых, приобрело у Intel специально образованное подразделение корпорации IBM по разработке и производству персональных компьютеров.

Параметры нового ПК были уже достаточно внушительны – операционная система MS-DOS 1.0. фирмы Microsoft, процессор Intel 8088 с частотой 4,77 МГц, 64 Кбайт оперативной пемяти, расширяемой до 640 Кбайт. Появление IBM PC в 1981 году породило лавинообразный спрос на персональные компьютеры, которые стали теперь орудием труда людей самых разных профессий. Наряду с этим возник гигантский спрос на программное обеспечение и компьютерную периферию. На этой волне возникли сотни новых фирм, занявших свои ниши компьютерного рынка.

С этого момента началось победное шествие ПК по всему миру. Появилось множество клонов (IBM-подобных компьютеров других фирм-производителей). Их характеризовала программная и аппаратная совместимость с IBM PC. Немногие из участников марафона продолжают ныне начатую в начале 80-х гонку. В первой сотне наиболее удачливых, кроме IBM, в 2001 году находились: Hewlett-Packard (19 место, доход 48 млрд. долл.), Compaq Computer (27 место, доход 42 млрд. долл.), Lucent Technologies (28-е место, доход 41 млрд. долл.), Motorola (34-е место, доход 37 млрд. долл.), Dell Computer (48-е место, доход 31 млрд. долл.), Microsoft (79-е место, доход 22 млрд. долл).

Развитие вычислительной техники и информационных технологий вывело компьютеры из больших машинных залов на рабочие столы пользователей, сделав их предметом первой необходимости практически во всех сферах труда

– орудием для упрощения рутинных операций и получения новой информации. Этому способствовало открытие «феномена персонального компьютера и персональных вычислений», которое в США связывают с именем Стива Джобса – основателя и руководителя фирмы Apple Computer (рис. 1.16).

Рис. 1.16.
Стив Джобс,
основатель Apple
Computer

В 1980 году Джобс определил этот тип электронной вычислительной машины (ЭВМ), как индивидуальный инструмент для усиления природных возможностей человеческого разума. В середине 1981 года Джобс сформулировал концепцию ПК с помощью простой аналогии. "Однажды, – писал он, – мне довелось разглядывать список биологических видов, расположенных по уровню эффективности, с которой они используют свою мускульную энергию для передвижения. На первом месте по эффективности в этом списке находится кондор (рис.1.17) – он почти не шевелит крыльями при полете, а человек пребывает в нижней трети этого обширного списка.

В то же время, из исследований биологов известно, что человек, который едет на велосипеде, по эффективности использования мускульной энергии намного превосходит всех известных животных, включая кондора". Другими словами, персональный компьютер выполняет для человека те же функции повышения эффективности, что и велосипед, но в иной, мыслительной сфере человеческих возможностей, позволяя заметно увеличить эффективность его интеллектуальной деятельности.

Рис. 1.17
Кондор в полёте

Из всего этого Стив Джобс делает вывод, что основное назначение ПК заключается в том, чтобы освободить человека от гнета рутинной обработки информации, оставляя ему "…делать то, что он может делать значительно лучше, чем любой из созданных им приборов***: концептуально мыслить".***

Выпускник вуза на производстве для решения поставленных перед ним задач, как правило, выходит "один на один" с персональным компьютером. И здесь от него требуется глубокое овладение компьютерной грамотностью, базирующейся на знаниях не только взаимодействия программных и аппаратных средств самого ПК (рис. 1.18), но и последнего с окружающим его реальным миром, без чего невозможно ожидать максимальной отдачи от столь полезного и наукоёмкого устройства под названием "персональный компьютер".

С невероятной скоростью изменяются технологии и концепции комплексирования компьютеров и их элементов, но общие подходы в

конструировании моделей данных и программных компонент уже сформировались.

Рис. 1.18. ПК фирмы Apple ″образца″ 2003 года.

Поэтому важно усвоить фундаментальные представления об информационных процессах в компьютерных системах для дальнейшего успешного применения информационных технологий в повседневной практике каждого специалиста.

1.4. Развитие операционных систем для персонального компьютера

Когда в 1981 году был выпущен первый промышленный вариант ПК, никто не мог предположить, что к 2000 году 86% всех ПК во всём мире будут работать под управлением операционных систем фирмы Microsoft, а сама фирма – в 2001 году в списке журнала Fortune расположится на 79 месте. Этому способствовало ежегодное увеличение её доходов и прибыли на 30%, что, после уплаты всех налогов, в 2002 г. составляло 250 000 долл. в год на одного служащего (а в среднем в американской промышленности – 17 000 долл. в год).

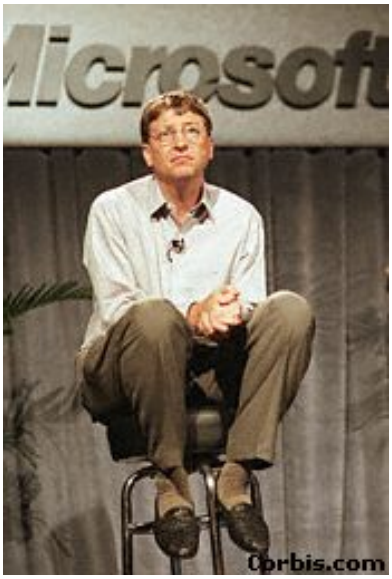


Рис. 1.19 Основатель Microsoft Билл Гейтс

Как свидетельствует одно из многочисленных жизнеописаний, дело начиналось так....

Уильям Генри Гейтс III (рис. 1.19) родился в 1955 году в Сиэтле (шт. Вашингтон, США) в семье юриста и школьной учительницы. Родители определили Билла в престижную частную школу, где открылись его склонность к математике и незаурядные способности к программированию.

Там же произошло знакомство с Полом Алленом – оказалось, что и тот и другой увлекались научной фантастикой. Чуть позже в местном вычислительном центре образовался детский кружок, в который немедленно записались Пол и Билл. Интерес к программированию повлек за собой привычку тратить всё свободное время (а часто и несвободное) на постижение программистских премудростей. После перехода в восьмой класс, Билл впервые приступил к созданию программ для своей родной школы, за что последняя платила

машинным временем, которое, в свою очередь, использовалось для новых разработок.

Последний класс мистер Гейтс аккуратно делил между школой и работой программистом в аэрокосмической корпорации TRW, где получал 20 тысяч долларов ежегодно.

В 1973 году свои объятия юному Биллу открыл Гарвард, куда вскоре из Вашингтонского университета перешел и Пол Аллен.

Как-то раз, завладев журналом Popular Elerctonics, Гейтс и Аллен выудили оттуда статью о настольном компьютере Altair. Его производитель компания MITS из г. Альбукерка (шт. Нью-Мексико, США), желала приобрести язык программирования, который позволил бы специалистам с ним работать. Пол и Билл немедленно связались с MITS и поведали, что на данный момент обладают готовой версией языка Бейсик, которого, скажем откровенно, у них не было и в помине. После договоренности о встрече, на всё про всё Билл и Пол получили в своё распоряжение три недели, по прошествии которых должен был состояться очный контакт с представителями MITS. Теперь оставалась сущая безделица – сочинить обещанный интерпретатор, не имея на руках самого компьютера Altair. Однако выход был найден незамедлительно! На имевшемся под руками большом компьютере была эмулирована система команд новинки – процессора Intel 8080. Полученный таким образом интерпретатор языка Basic Билл Гейтс и Пол Аллен продали MITS. Труд компаньонов был потрачен не зря. Ведь в продаже каждая копия программы от MITS стоила "всего" 500 долл. (рис. 1.20).

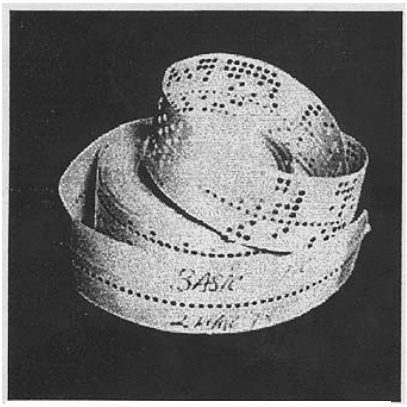


Рис. 1.20 Перфолента с кодом интерпретатора языка BASIC, написанный Б. Гейтсом и П. Алленом для ПК Altair 8800 в 1975 г.

Успех окрылил Гейтса и в 1975 году он досрочно расстался с гостеприимными стенами Гарварда, не потрудившись закончить учебу и получить диплом. Вдвоем с Полом Алленом они перекочевали в штат Нью-Мексико, где и была 4-го апреля 1975 г. зарегистрирована компания Microsoft (Microcomputer Software). На первых порах Пол настаивал на специализации в области аппаратного обеспечения, однако Билл предложил сделать упор на разработку программ, не без основания полагаясь на успех программного обеспечения в компьютерной индустрии.

Движимый естественными амбициями, Гейтс приступил к набору достойной команды. Он планомерно объезжал колледжи и университеты в поисках одаренных студентов, честно предупреждая об изнурительной работе, гарантируя при этом полную свободу творчества. Энтузиасты попадались крайне редко, но их всё же хватило,чтобы набрать необходимый состав. Поначалу дела у фирмы шли не очень гладко. Компания не могла позволить себе найм менеджера по сбыту продукции – этим

занималась мать Билла, без малейшего смущения предлагавшая программы Microsoft таким огромным корпорациям, как IBM и AT&T.

Мало помалу, лицензии на усовершенствованный BASIC стали покупать такие фирмы как DTS, General Electric, NCR, Citibank, Radio Shack, Apple и даже IBM. Неудивительно, что 4 апреля 1979 г. BASIC 8080 стал первым официально отмеченым программным продуктом фирмы и получил премию ICP (American College of Physicians) в 1 миллион долларов. Эта награда, врученная Полу Аллену, явилась заслуженной наградой Microsoft за нелёгкий труд на ниве создания программного обеспечения. Прояснилось новое стратегическое направление и 18 июня 1979 г. Microsoft представляет новый BASIC для компьютерных систем на микропроцессоре 8086. Это был первый язык программирования высокого уровня, который появился для 16-битных машин.

И неудивительно, что выбор ″голубого гиганта″ IBM на разработку базовой операционной системы (ОС) для всех своих будущих ПК пал именно на Microsoft, так как эта составная часть компьютера является одной из наисложнейших и важнейших в его работе (рис. 1.21).

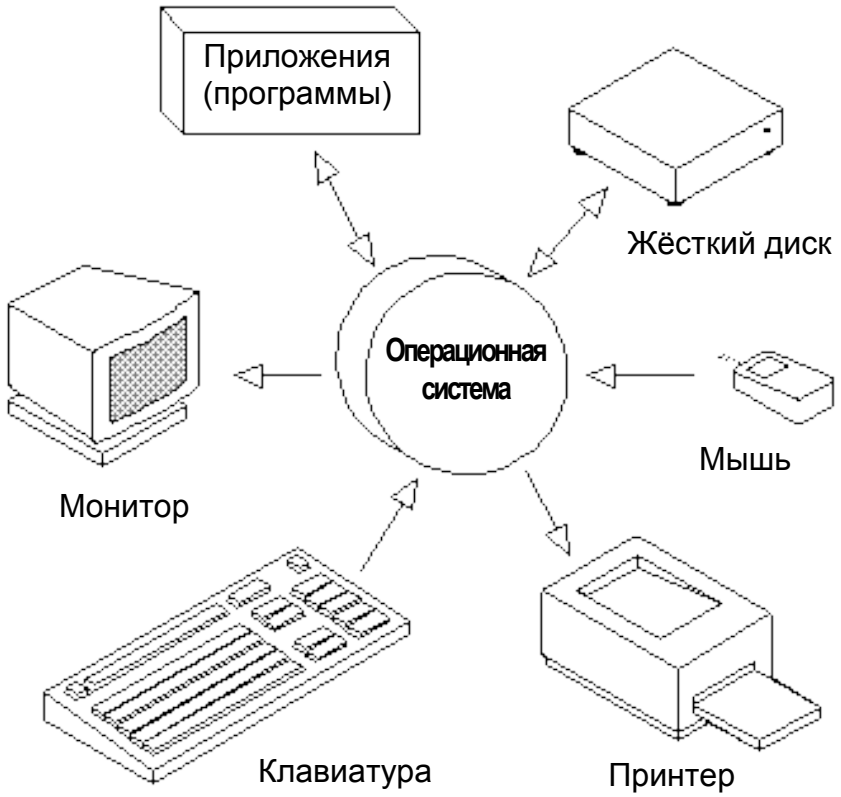


Рис. 1.21. Области применения управленческих
и ″исполнительских обязанностей″ современной операционной системы

Задача для Microsoft была непростой, ведь к тому моменту компания разрабатывала и поставляла только компиляторы и интерпретаторы для языков программирования и насчитывала всего 39 человек. Естественно, что подходящей операционной системы у Билла ещё не было. Первое, что пришло

ему в голову – это порекомендовать IBM обратиться к сопернику Microsoft, Digital Research, которая уже имела в своём распоряжении довольно популярную систему CP/M, установленную на многих 8-разрядных компьютерах. Однако, немного поразмыслив, Гейтс тут же среагировал и руководству IBM была направлена целая диссертация о необходимости перехода на более мощный 16-рязрядный процессор 8080 от Intel. Видимо, текст оказался убедительным, вследствие чего CP/M была отвергнута, и договор на разработку новой дисковой операционной системы достался Microsoft.

В условиях сжатых донельзя сроков Пол Аллен установил контакт с маленькой компанией Seattle Computer Products, располагавшей подходящей дисковой операционной системой 86-DOS, нуждавшейся в серьёзной оптимизации для установки на IBM PC, выпуск в продажу которого должен был состояться уже через месяц. Microsoft поступила мудро, не только купив 86-DOS, но и пригласив на работу её создателя Тима Паттерсона. Гейтс и компания активно доводили до ума выкупленную ДОС, работая «по 25 часов в сутки», откуда пошла известная традиция хранить в шкафах фирмы спальные мешки для круглосуточно работающих программистов. В результате 2 августа 1981 года IBM представила свой ПК, в котором использовались программные продукты от Microsoft Inc: 16-битная дисковая операционная система MS-DOS 1.0 и языки программирования BASIC, COBOL и Pascal.

Начался быстрый рост количества инсталляций (установок) MS-DOS. За первые 16 месяцев лицензию у Microsoft купили 50 производителей подобных компьютеров. Одновременно рос уровень производства ПК и у других фирм. Таким образом, хотя IBM и сохраняло лидерство в производстве ПК, монополистом уже считаться не могла. Были образованы и начали активно развиваться Compaq, Hewlett-Packard, Texas Instruments.

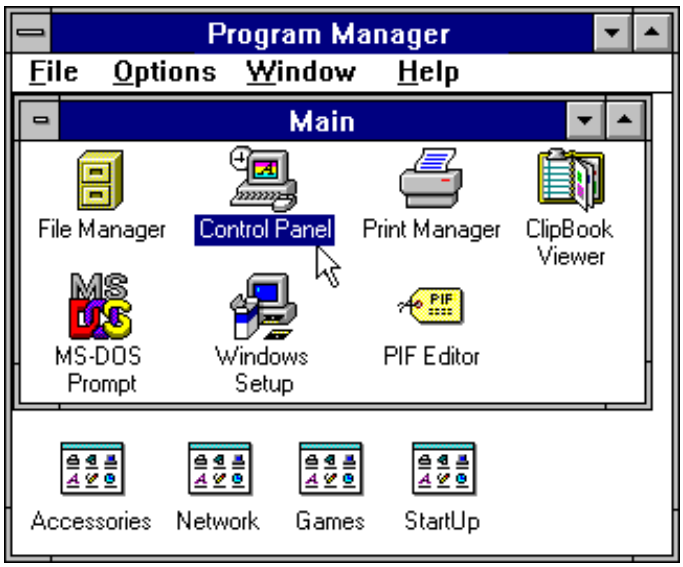


Рис. 1.22. Интерфейс первой графической ОС Windows 3.1

И тут, уловив актуальность момента, Билл Гейтс сделал гениальный ход, призвав ведущих производителей ПК создавать продукцию, рассчитанную на полную совместимость с машинами IBM для того, чтобы любая программа, написанная для IBM ПК, могла быть использована для всех этих компьютеров. Дар убеждения Гейтса привел к тому, что MS-DOS стали скупать оптом и в розницу, чуть ли не вставая за ней в очередь. В результате такого ажиотажа на 80 % всех ПК в мире была установлена именно MS-DOS,

благодаря чему и возникло понятие "IBM-совместимость компьютеров".

10 сентября 1983 года Microsoft впервые анонсировала Windows, появившуюся как графический ***интерфейс*** для операционной системы MS-DOS, представляющий собой универсальную операционную среду для работы с прикладными программами (рис. 1.22).

В 1990 г. было заключено соглашение между Microsoft и Intel по созданию стандарта Wintel (аббревиатура Windows + Intel) (рис. 1.23), который стал популярным благодаря ПК на базе чипов, производимых Intel, работающих под управлением ОС Microsoft Windows. К середине 2003 г. операционные системы Windows-9x/NT/200x были установлены на 91% компьютеров во всём мире.

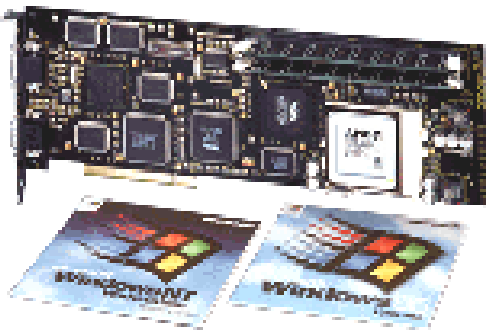


Рис. 1.23. Объединение технологий Microsoft и Intel

Сполучення даної ОС з могутніми процесорами, що досягли до 2003 р. швидкодії 3,2 ГГц і супермісткими жорсткими дисками ємністю від 60 ГГб і більше створює унікальні можливості для рішення численних задач у різних галузях практичної діяльності людини.

Сочетание данной ОС с мощными процессорами, которые достигли к 2003 году быстродействия 3,2 ГГц с супервместительными жёсткими дисками ёмкостью от 60 ГГб и больше создаёт уникальные воможности для решения многочисленных задач в разных областях практической деятельности человека.

По данным аналитиков Ovum, пятёрка лидеров продаж программного обеспечения для персональных и коропоративных вычислений в 2003 г. выглядела следующим образом: Microsoft ($25,9 млрд.), IBM ($13,1 млрд.), Oracle ($6,9 млрд.), SAP ($6,8 млрд.) и Hewlett-Packard ($2,6 млрд.).

Понимая громадное значение развития и продвижения информационных технологий во все сферы государственной и промышленной деятельности ряд стран начинают резко активизировать действия по разворачиванию производства программного обеспечения (ПО) в сфере информационно-компьютерных технологий (ИКТ). Так, по сравнению с приведенными выше цифрами, по данным тех же аналитиков, в 2005 г. предполагается, что экспорт програмного обеспечения из Индии в Америку и Европу будет превышать $35 млрд., что составит 5–7% всего её совокупного ВВП. Кстати, вклад всего российского сектора производства ПО в экономику страны в 2003 году не превышал и 1%.

На таком фоне быстрый подъём Индии должен стать главным фактором её стремительного экономического роста. Здесь она уже сейчас имеет несравненные преимущества перед другими странами, которые только начали работать в этом направлении. Ведь успехи Индии обеспечиваются не только

благодаря большому количеству сертифицированных специалистов, высокой дисциплине работников при низкой стоимости их труда, но и полным отсутствием языкового барьера, так как английский язык в Индии давно получил статус официального и стал языком общения в сфере ИКТ. Поэтому индийцы работают сейчас практически в каждой компании Силиконовой долины. В самой же Индии, уже сегодня, около 60% оборота ПО – экспортные операции, на которые работают 400 компаний. На долю двадцати самых крупных из них приходится 67% всего оборота. При этом 57% индийского экспорта ПО идёт в США, 22% – в западноевропейские страны и только 21% – в другие страны мира.

О потенциале страны говорят такие цифры. В конце 1990-х годов в Индии действовали 226 университетов, в которых обучались более 6,5 млн. студентов, что составляло около 6% молодёжи страны в возрасте 17-23 лет (в 1972 г. здесь было только 83 университета и 9 институтов, обучавших 2,6 млн. студентов). По количеству англоязычных дипломированных специалистов в сфере инфомационных технологий Индия находится на втором месте в мире после США, а в индийских компаниях над разработкой ПО трудятся около трёхсот тысяч специалистов. Таким образом, информационная составляющая играет важную роль в развитии многих стран.

Вопросы

1. Какие носители использовались ранее и используются сейчас для сохранения и передачи информации?
2. В каких видах существуют и передаются знания?
3. Благодаря чему постоянно ускоряются темпы накопления знаний?
4. Почему язык, письменность, печатный станок и другие средства фиксирования необходимой человеку информации называются технологиями?
5. Почему английскому изобретателю Чарльзу Беббиджу не удалось реализовать свою вычислительную машину?
6. Какие технологии последовательно сменяли одна другую при конструировании электронно-вычислительных машин на этапах их развития?
7. Какие технологические новинки создали возможность построения первых персональных компьютеров?
8. Какую роль играет операционная система в работе компьютера и в работе пользователя?

Если ничего не получается, прочти же, наконец, инструкцию!

Журнал CW/K №42/98

2. ИЗЯЩЕСТВО ПРОЦЕССА ВКЛЮЧЕНИЯ ПК

2.1 Универсальность комплектации персонального компьютера

Рис. 2.1. Кнопки управления ПК

И элементы конструкции, и программное обеспечение ПК постоянно совершенствуются и эволюционируют. Производители стараются приблизить эксплуатационные характеристики ПК к уровню домашних бытовых приборов. Но, если нажатие кнопки "***Вкл***." на панели телевизора тривиально для нас по своєму эффекту, то нажатие кнопки "*Power*" на системном блоке ПК (рис. 2.1) связано с процессами, которые требуют некоторого осмысления.

Фундаментальное концептуальное отличие ПК от первых электронно-вычислительных машин кроется в многоликом и ёмком слове "***персональный***". Мало того, что в компьютерном комплекте, который представляет собой работоспособный ПК, компонуется более десятка разнообразных электронных устройств, объединённых общими для всех них соединителями и формфакторами плат. В любой момент и любое из них может быть заменено на соответствующее по характеристикам другое из громадного количества существующих сейчас унифицированных частей практически от любой фирмы-производителя (табл.2.1).

Таблица 2.1.

Компоненты, необходимые для сборки современного ПК:

Аппаратные компоненты	Состав системной (материнской) платы
Системная (материнская) плата Процессор Память (оперативная память) Корпус Блок питания Дисковод для гибких дисков Жёсткий диск Накопитель CD-ROM, CD-RW или DVD-ROM Клавиатура Мышь Видеоадаптер Монитор (дисплей) Звуковая карта Акустическая система	Гнездо процессора Преобразователи напряжения питания процессора Набор микросхем системной логики системной платы Кэш-память второго уровня (кэш L2) Гнездо памяти SIMM или DIMM Разъёмы (слоты) шины ROM BIOS Батарея для питания системных часов и CMOS Микросхема ввода-вывода

Десятки тысяч программных продуктов разного назначения могут выполняться на этом конкретном наборе аппаратных средств. Такая взаимозаменяемость аппаратных компонентов и интероперабельность (переносимость) программных средств, стали возможны благодаря тому, что идеология внутреннего построения и концепция функционирования ПК постоянно согласовываются и стандартизуются всеми заинтересованными фирмами-разработчиками *hardware* и *software*. ***Усилиями сотен тысяч фирм-производителей общий принцип запуска и функциониования ПК унифицирован полностью.***

2.2 BIOS всему "голова"

Обычно, после сборки и настройки, ничего не должно помешать Вашему ПК приступить к выполнению своих прямых обязанностей – решать для Вас Ваши задачи. Поэтому алгоритм начала его работы "запаян" в чипе, называющемся ***постоянным запоминающим устройством*** (ПЗУ, ROM – Read Only Memory), который размещён на одной из плат внутри системного блока, а сама ***программа***, которая реализует этот алгоритм, называется ***базовой системой ввода - вывода*** ПК (*BIOS* – Basic Input/Output System)[7] (рис. 2.2). Этот чип как Минотавр (получеловек-полузверь), наполовину относится к программному, а наполовину – к техническому обеспечению, одновременно являясь и частью аппаратуры, и ***частью любой операционной системы*** и, в том числе, и MS DOS.

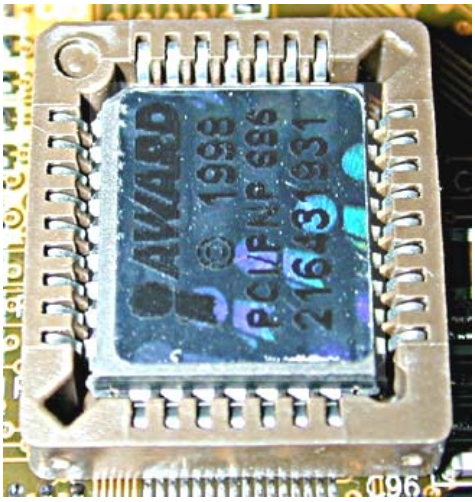


Рис. 2.2 BIOS фирмы AWARD Software

При изготовлении таких чипов в настоящее время получили распространение схемы с перемычками в виде плавких вставок-предохранителей, которые можно избирательно "пережечь" с помощью внешнего источника тока достаточной силы, получив при этом память с соответсвующей направленностью в работе (Programmable Read Only Memory). Целенаправленное пережигание формирует наперёд заданные свойства такого устройства в виде неизменяемых кодов программных компонентов, драйверов устройств, подпрограмм обработки данных и т.д[8]. Нелишне напомнить, что только в микропроцессоре за период в 30 лет с 1971 года число транзисторов увеличилось с

[7] BIOS – группа программ, которые работают непосредственно с базовыми аппаратными средствами компьютера и с некоторыми периферийными устройствами, выполняя основные, фундаментальные задачи в системе, то есть обмен данными на уровне байта между процессором и клавиатурой, экраном, дискетой, жёстким диском и т.д.

[8] По способу программирования выпускаемые ПЗУ делятся на следующие три основных типа: (1) Программируемые в процессе изготовления. (2) Однократно программируемые у заказчика. (3) Программируемые у заказчика с возможностью перепрограммирования.

2 300 (Intel 4004) до 45 млн. (Intel Pentium 4). А частота его работы возросла с 0,108 Мгц до 3,2 Ггц (3 200 Мгц!), что вызвало необходимость внесения соответствующих изменений в конструкцию и в состав программных компонетов BIOS.

Как правило, BIOS для современных системных плат разрабатывается одной из нескольких фирм, которые специализируются на этом: Phoenix Technology, Award Software, American Megatrends Inc. (AMI) и некоторые другие.

BIOS в большинстве IBM-совместимых компьютеров выполняет пять основных функций:

❶ *POST(Power-OnSelf Test)* – самотестирование при включении питания процессора, памяти, набора микросхем системной логики, видеоадаптера, контроллеров диска, дисковода, клавиатуры и других жизненно важных компонентов системы.

❷ Запуск и выполнение ***программы установки параметров ПК*** – *Setup BIOS*, осуществляющей установку параметров аппаратуры компьютерной системы для загружаемой DOC или ОС. Эта программа запускается при нажатии определённой клавиши (или комбинации клавиш) во время выполнения процедуры POST до загрузки операционной системы (рис. 2.3).

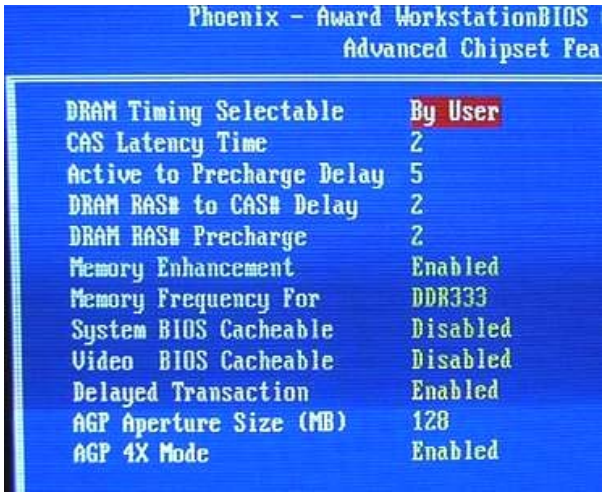


Рис. 2.3 Интерфейс программы Setup BIOS

❸ Реализация ***начальной загрузки системы***, то есть выполнение поиска на дисковых устройствах ПК главного загрузочного сектора *(Master Boot Record)*. Если последние два байта этого сектора (его сигнатура) равны *55ah*, данный код выполняется.

❹ Предоставление набора *BIOS–**драйверов***[9], предназначенных для взаимодействия операционной системы и аппаратного обеспечения при загрузке системы. При запуске DOS или Windows в так называемом защищённом режиме в работе устройств ПК также используются только драйверы устройств из BIOS.

❺ ***Обработка прерываний***[10] при обмене данными ПК с внешними устройствами.

Так как при каждом очередном включении компьютера неизвестна текущая конфигурация системы (некоторые устройства могут быть на

[9] Драйвер – программа, которая управляет работой некоторого внешнего устройства (мышь, клавиатура, принтер и т.д.); драйвер как правило является интерфейсом между конкретным устройством и программами ввода-вывода ОС. Наиболее характерным примером драйвера служит программа KEYRUS.COM, которая кириллизирует клавиатуру и монитор для обеспечения русскоязычного интерфейса пользователя с ПК.

[10] Прерывание – обрыв нормальной последовательности выполнения инструкций в работе компьютера. Прерывание вызывает автоматическую передачу управления на заранее определённый адрес в памяти, где расположена последовательность команд, выполнение которых и составляет процесс прерывания.

некоторое время отключены или отсоединены, а другие находиться в неработоспособном состоянии), то первой важной задачей BIOS является автоматическое тестирование, или проще говоря "опрос" всего подключенного к ПК оборудования и занесение в определённое место памяти информации об устройствах, которые принимают участие в выполнении текущей работы. Автотестирование выполняется программой POST (Power-On Self Test – автотест при включении электрического питания).

Рис. 2.4. Кнопка RESET ПК

При рестартировании (повторном запуске) ПК с помощью нажатия комбинации клавиш Ctrl-Alt-Del еще раз выполняется программа POST, но уже без тестирования памяти. Этот способ перезапуска имеет специальное название – "тёплый" перезапуск и может выполняться с помощью специальной кнопки "Reset" (рис. 2.4), которая находится на передней панели системного блока ПК. Другой способ – "холодный" перезапуск, который реализуется путём выключения и повторного включения питания ПК. Оба этих способа являются спасением в случае "зависания" компьютера (а вернее его операционой системы), по обстоятельствам, которые не зависят от пользователя.

Рис. 2.5. Внешний вид платы ПК с BIOS

Первая информация, которая появляется на экране компьютера после его включения – это название фирмы-изготовителя BIOS, а потом и сама BIOS на экране дисплея показывает список устройств, которые подключены в данном сеансе работы и их характеристики. Большая часть времени самотестирования уходит на проверку работоспособности элементов оперативной памяти: чем больше микросхем памяти установлено в ПК, тем дольше идёт процесс тестирования. При обнаружении каких нибудь неполадок, BIOS (рис. 2.5) выводит на экран ПК соответствующие сообщения (обычно в виде условного кода ошибки) и извещает об этом пользователя звуковым сигналом. Дальнейшая работа машины при этом завершается и пользователю необходимо принять меры к устранению несправностей, которые были обнаружены. Иногда причиной ошибки может послужить простое нарушение контакта (например, при сильном изгибе кабеля, который присоединяет клавиатуру, или при частичном выходе из гнезда одной из печатных плат, которые вставляются внутрь системного блока). Такие ошибки ликвидируются легко, но некоторые неисправности могут потребовать замены соответствующих узлов либо целых устройств.

По окончании процесса тестирования, начинается этап "раскрутки" операционной системы (bootstrapping). Буквально этот термин переводится как "поднятие себя за голенища сапог или за шнурки ботинок". Однако речь идет не об известном подвиге Барона Мюнхаузена, а о гениальной находке разработчиков компьютерной техники – "втаскивании" ОС в компьютер усилиями схемы BIOS (и её программ!). Это она, как бы на ощупь, начинает обшаривать возможные места присутствия небольшой, но очень важной программы начальной загрузки (т.н. *Master Boot record)* на одном из нескольких дисковых устройств компьютера, которая и инициирует дальнейшую загрузку DOS (Disc Operating Sysytem) или ОС.

Зачем же это необходимо? Рассмотрим реальную ситуацию. Вы приобрели компьютер и ещё не успели привести в рабочее состояние его жесткий диск. Т.е. не отформатировали[11] его и не установили на нём требуемую Вам операционную систему. Как же в этом случае начать работу?

Рис. 2.6. Вставка дискеты в слот дисковода

Нужно простым движением вставить в флоппи-дисковод системную дискету (рис. 2.6), т.е. дискету с записанной на неё ОС DOS и нажать кнопку Power. Заметим, что на любую дискету при форматировании в Boot sector всегда записывается Master Boot Record, а на системную – ещё и все модули DOS. Поэтому, когда Вы вставите эту системную дискету в флоппи-дисковод, при включении ПК умный BIOS найдёт на ней Master Boot Record и загрузит его в оперативное запоминающее устройство (ОЗУ). Естественно, что "любая" дискета Вас не спасёт – ведь один Master Boot Record в поле не воин! Поэтому хороший хозяин всегда должен иметь в запасе ещё одну системную дискету с каким-либо файл-менеджером (Volcov Commander, Far Manager или каким-либо другим). В компьютере обычно предусмотрено (и это удобно), что роль системного диска в ПК может играть по выбору гибкий или жёсткий магнитный диск. Поэтому BIOS сначала делает попытку осуществить чтение с гибкого диска (рис. 2.7).

Если гибкий (флоппи) диск в момент включения ПК вставлен в слот дисковода, то именно он и считается системным. Если же гибкий диск отсутствует, то BIOS обращается к жесткому диску, считая системным его (рис. 2.7). Понятно, что к этому моменту он должен быть уже отформатирован и содержать все необходимые компоненты устанавливаемой операционной системы (MS-DOS, Windows 9x/200x/XP или какой либо другой).

Следует отметить, что последовательность поиска программы начальной загрузки можно менять, добавляя для этого новые устройства (допустим, CD-ROM и т.д.), выполнив соответствующие изменения в установках программы

[11] Форматирование – программно управляемое нанесение на поверхность дисков участков стандартной длины (секторов) для последующей записи файлов.

Setup BIOS. Важно также и то, что ***системная BIOS*** (т.е. та, которая находится в чипе, укреплённом на плате в системном блоке) содержит драйверы основных компонентов ПК (клавиатуры, дисковода, жёсткого диска, последовательного и параллельного портов и т.д.), необходимых для начального запуска компьютера.

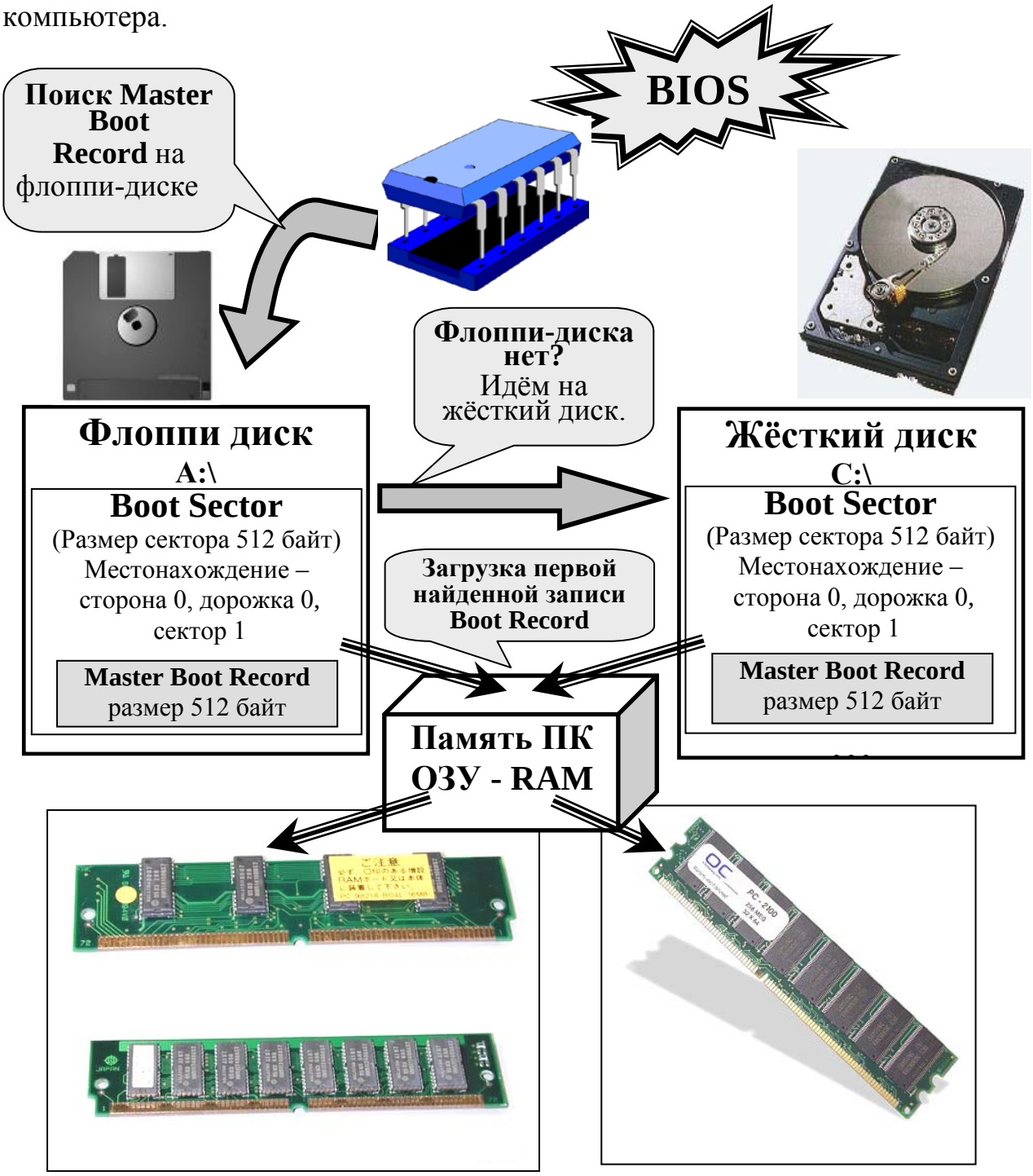


Рис. 2.7. Процесс поиска на дисках и размещения в ОЗУ ПК программы начальной загрузки Master Boot Record (начальная запись)

По мере появления всё новых и новых устройств (видеоадаптеров, накопителей CD-ROM, жёстких дисков с интерфейсом SCSI и т.д.) их процедуры инициализаци не добавлялись в системную BIOS. Острая необходимость в таких устройствах в процессе запуска компьютера приводит к

необходимости загрузки соответствующих драйверов непосредственно с дисков во время установки операционной системы. Это, в первую очередь относится к звуковым адаптерам, сканерам, принтерам, устройствам PC Card (PCMCIA) и т.д.

Однако, некоторые устройства крайне необходимы на начальном этапе запуска компьютера. Например, для отображения информации на экране жидкокристаллического монитора необходима активизация соответствующего видеоадаптера, но его поддержка не встроена в системную BIOS. Кроме того, сейчас существует большое количество различных видеоадаптеров и все их драйверы невозможно разместить в BIOS. В таких случаях необходимые драйверы размещаются в микросхеме второй BIOS, размещаемой на плате самого этого устройства.

А системная BIOS, при загрузке, ищет вторую, специализированную BIOS требуемого видеоадаптера и загружает её в ОЗУ до запуска операционной системы. Такое расположение и использование второй BIOS предотвращает необходимость постоянной модернизации основной системной BIOS при появлении новых моделей устройств и особенно тех, которые используются на начальных этапах включения компьютера.

Поскольку в целом принципы загрузки различных операционных систем с помощью BIOS подобны между собой, то дальнейшее описание процессов, которые сопровождают подготовку компьютера к дальнейшей работе, будет проведено на примере операционной системы MS DOS.

Итак, после самотестирования устройств ПК, важнейшей задачей BIOS является поиск записи Master Boot Record, размещения её в памяти ПК (ОЗУ) и передача на неё управления с целью выполнения. А уже Master Boot Record, в свою очередь, начинает процесс загрузки модулей операционной системы. Когда же Вы уже приступили к работе на компьютере, включается пятая важная функция BIOS – обслуживание прерываний ОС, которые производятся программными или аппаратными средствами с целью выполнения операций обмена информацией между устройствами и узлами ПК. Прерывания можно разделить на три основные группы: аппаратные, логические и программные. Источниками ***аппаратных*** прерываний являются падения напряжения питания, нажатия клавиш клавиатуры, поступление очередного импульса от счётчика времени (таймера), возникновение специальных сигналов от накопителей на гибких или жёстких дисках и др.

Логические, или процессорные, прерывания возникают при разных нестандартных ситуациях в работе центрального процессора (CPU – Central Processing Unit) — деление на нуль, переполнение регистров, появление «точки останова» и др.

Программные прерывания — наиболее обширная категория. Вырабатываются они, когда одна программа хочет получить определенный сервис со стороны другой программы. Причём этот сервис, как правило, связан с обменами данных между компонентами аппаратных средств, а так же работой программ с ними.

Вернёмся, однако, к процессу загрузки DOS. Наверное, лучше всего его можно описать формулой: "дедка за репку, бабка за дедку и т.д.". Только в нашем случае это будет выглядеть приблизительно так: "BIOS за Boot Record, Boot Record за файл IO.SYS, IO.SYS за COMMAND.COM и т.д. (рис. 2.8).

Поскольку BIOS находится в постоянном запоминающем устройстве, изменения прерываний, которые в ней реализованы, представляются достаточно сложной задачей. Поэтому разработчиками был предложен механизм, который позволяет дополнять возможности работы BIOS с новой аппаратурой.

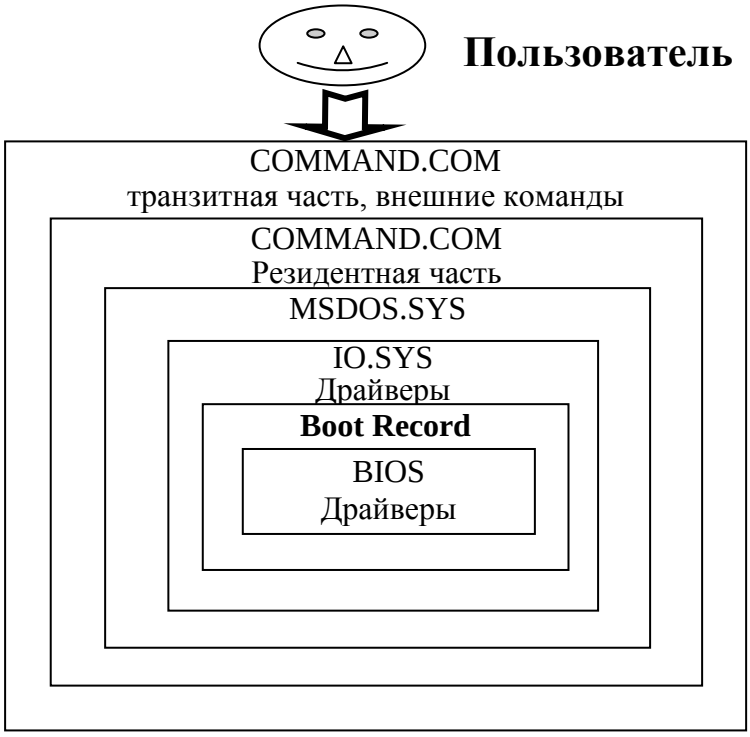


Рис. 2.8. Иерархия элементов модульной структуры MS-DOS

Для этого в DOS предназначен модуль IO.SYS, который называется модулем расширения BIOS. В него можно вводить новые прерывания, которые "перекрывают" (заменяют) старые, либо подключать программы-драйверы для обеспечения работы с любыми новыми устройствами, либо работать по-новому со старыми.

Таким образом, IO.SYS загружает в ОЗУ Boot Record вместе с модулем обработки прерываний DOS под названием MSDOS.SYS. Чтобы упростить работу Boot Record оба этих модуля располагаются, начиная с первого сектора системного диска. Первым из этих двух модулей начинает работу IO.SYS, который внимательно читает и обрабатывает информацию, находящуюся в файле CONFIG.SYS. В нём сообщается о необходимости подключения новых драйверов внешних устройств, а также конфигурируется операционная обстановка. CONFIG.SYS редактируется (изменяется) пользователем заранее как простой текстовый файл и его содержимое будет рассмотрено далее.

Завершив чтение и обработку файла CONFIG.SYS, модуль расширения BIOS (IO.SYS) передает управление на загруженный к этому моменту в ОЗУ модуль обработки прерываний DOS под названием MSDOS.SYS, в котором

производятся системные установки и подготовка к загрузке в ОЗУ командного процессора COMMAND.COM, находящегося пока на системном диске. После этого управление возвращается в модуль расширения BIOS *IO.SYS*, который производит загрузку командного процессора (файл COMMAND.COM) с диска в ОЗУ и передаёт ему управление дальнейшей работой устройств ПК.

Командный процессор COMMAND.COM при загрузке исполняет командный файл AUTOEXEC.BAT и в отличие от своих требовательных собратьев Boot Record, IO.SYS и MSDOS.SYS может располагаться на системном диске в любом месте. При этом он рассматривается как обычная программа. Это он печатает строку-приглашение на экране дисплея и выполняет все пожелания пользователя, задаваемые в виде команд DOS. Его основные функции заключаются в следующем:

❶ анализировать разнообразные команды, которые вводятся пользователями с клавиатуры или из командного файла с расширением .BAT;

❷ загружать в ОЗУ и выполнять внешние программы DOS и разнообразные прикладные программы (файлы с расширеннем .COM и .EXE) (рис. 2.9).

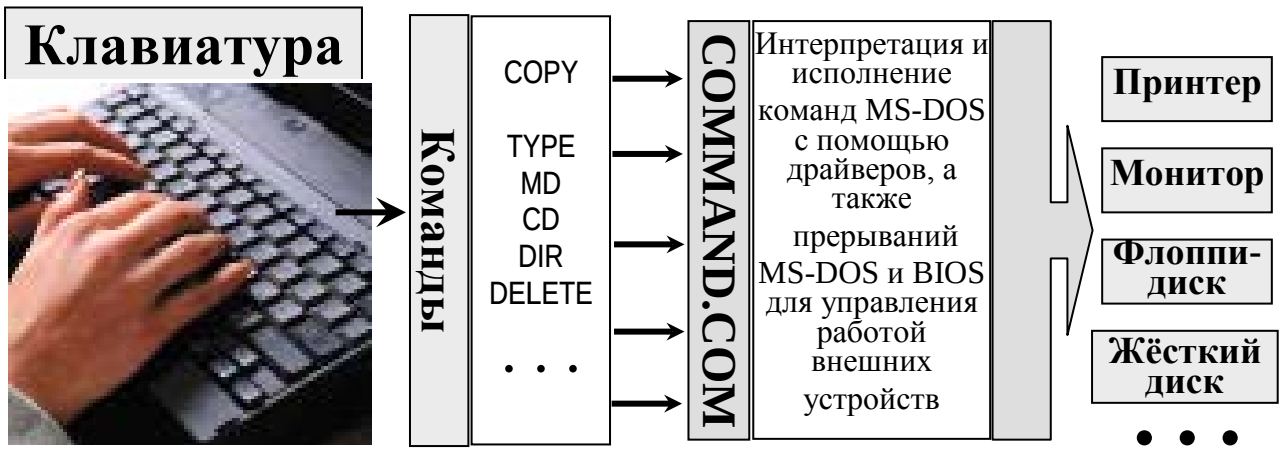


Рис. 2.9. Путь команд от их ввода с клавиатуры до выполнения их ОС MS DOS

Образно выражаясь, компьютер можно представить существом, которое общается с внешним миром с помощью своих периферийных (внешних) устройств. Его органы чувств – это клавиатура, мышь и сканер (рис.2.9), а средства обратной связи – принтеры, дисплеи, диски и другие периферийные устройства, посредством которых осуществляется контакт компьютера с внешним миром конкретных объектов и пользователем.

Все эти устройства, по сути, являются буферами для приёма, накопления, переработки и передачи потоков информации, циркулирующих между всеми устройствами компьютера. Кстати, многие из них при этом имеют ***существенно различные скорости*** обработки, передачи и отображения данных (к примеру, скорость обработки команд микропроцессором – 3,2 ГГц, а принтер может выводить десяток или более строчек в секунду и т.д.) (рис.2.10).

Устройства ввода получают из внешнего мира команды и связанные с ними данные, которые поступают в память для обработки. Устройства вывода

получают вычисленные результаты и передают их пользователю или другому устройству для вывода или сохранения. Точки контакта между устройствами ввода - вывода и микропроцессором называются ***портами ввода-вывода*** (рис. 2.11).

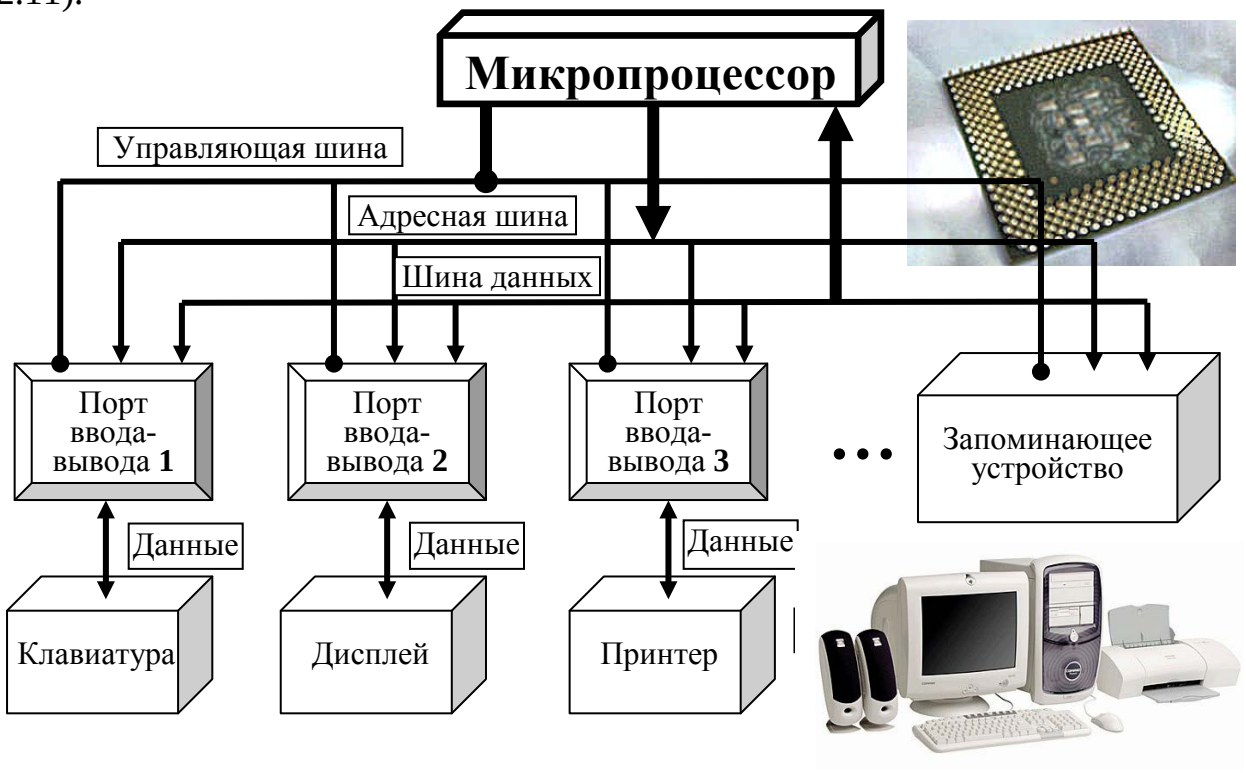


Рис. 2.10. Организация взаимодействия между устройствами компьютера

В соответствии с общепринятыми соглашениями ***портом ввода*** называется любой источник данных, а ***портом вывода*** – приёмник данных. ***Шины*** – это информационные каналы, которые соединяют устройства внутри компьютера.

Рис. 2.11. Коннекторы портов ввода-вывода ПК и сам порт (внизу)

По управляющим шинам подаются команды на выполнение разнообразных действий. Адресные шины служат для выбора необходимых портов и размещения в памяти информации, которая передаётся. Понятно, что по шине данных передаются потоки данных.

Порты ввода-вывода имеют свои адреса в памяти (ОЗУ), так что к одному микропроцессору может быть подключено несколько устройств ввода-вывода. По этому адресу в памяти находится поле ячеек, через которые и проводится обмен информацией между устройствами и пользователем. Процессор черпает информацию из одного места (например, с клавиатуры) и передаёт её в другое место (например, на дисплей).

Таким образом, мы приходим к осмыслению того факта, что мозг компьютера –

это процессор, сердце – таймер, который задаёт ритм работы всего его организма, кровеносные сосуды, которые несут информацию – это шины адресов и данных, а органы чуств – это порты, которые принимают информацию (управляющие сигналы и данные) от внешних устройств и отсылают её назад.

Рис. 2.12. Представление имён папок (каталогов) на китайском языке

Каждый раз, когда мы нажимаем или отпускаем одну из клавиш клавиатуры ПК, схемы клавиатуры генерируют однобайтное число, которое называется скэн-кодом, который показывает только одно – нажата клавиша или отпущена, поскольку скэн-коды нажатия и отпускания клавиши разные. При этом, ни один из скэн-кодов ещё не связан с определённым символом конкретного языка, а лишь отмечает местонахождение клавиши на клавиатуре. Ведь за клавиатурой может работать и англичанин, и китаец (рис. 2.12), и финн, и, конечно же – украинец. Кроме того, известно, что население Земли говорит сегодня более чем на 3 000 разных языков, а пишет только на 100 из них. На английском языке говорят 1 млрд. 400 млн. жителей планеты, из которых только 400 млн. считают его своим родным.

Присваивание клавише символа того или иного языка – это работа подпрограмм (прерываний) ROM BIOS с помощью соответствующих портов, а отображение родного для пользователя языка выполняются дополнительными драйверами клавиатуры (PROKEY, KEYRUS и др.).

Подводя итоги выщесказанному, можно сказать, что конечным этапом процесса включения компьютера является полная и окончательная загрузка операционной системы DOS (или любой другой) в его оперативную память. Этот процесс называется ***начальной загрузкой системы*** (или перезагрузкой).

В ходе этого процесса:

❶ проверяется правильность работы устройств компьютера;

❷ в оперативную память компьютера загружается ОС DOS или компоненты другой операционной системы;

❸ происходит настройка DOS на выбранные параметры конфигурации и те устройства, ***которые подключены к компьютеру в этот текущий момент***;

❹ выполняются команды и программы, которые указаны пользователем в файле *autoexec.bat;*

❺ на экран выводится приглашение DOS, которое указывает, на то, что DOS готова к приёму команд пользователя (рис.2.13), либо на экране отображается интерфейс соответствующей графической ОС.

2.3. Как операционная система управляет процессом ввода-вывода

BIOS – это термин, который используется для описания базовой системы ввода-вывода. По сути, BIOS представляет собой "***промежуточный слой***" между программной и аппаратной частями системы, которая состоит из комбинации всех типов BIOS, а также драйверами устройств, которые загружаются для данной конфигурации ПК.

```
C:\Documents and Settings\Gm>cd \

C:\>_
```

Рис. 2.13. Приглашение ОС DOS для ввода очередной команды работы с диском С:

Часть BIOS, которая размещена в микросхеме на системной плате или платах адаптеров, зовётся *firmware*[12]. Стандартная PC-совместимая система складывается из нескольких слоёв, которые связаны между собой (рис. 2.14).

На этом рисунке показаны два разных компьютера, у которых используется уникальная BIOS в роли интерфейса между разным аппаратным обеспечением, операционной системой и её приложением (программой пользователя). То есть BIOS в каждом сеансе работы настраивается на взаимодействие с каждой конкретной конфигурацией аппаратных средств любого ПК. Таким образом, на этих компьютерах может быть установлено самое разное оборудование: процессоры разных производителей, жёсткие либо гибкие диски, мониторы и другие устройства, используя которые можно запускать ***одинаковое программное обеспечение***.

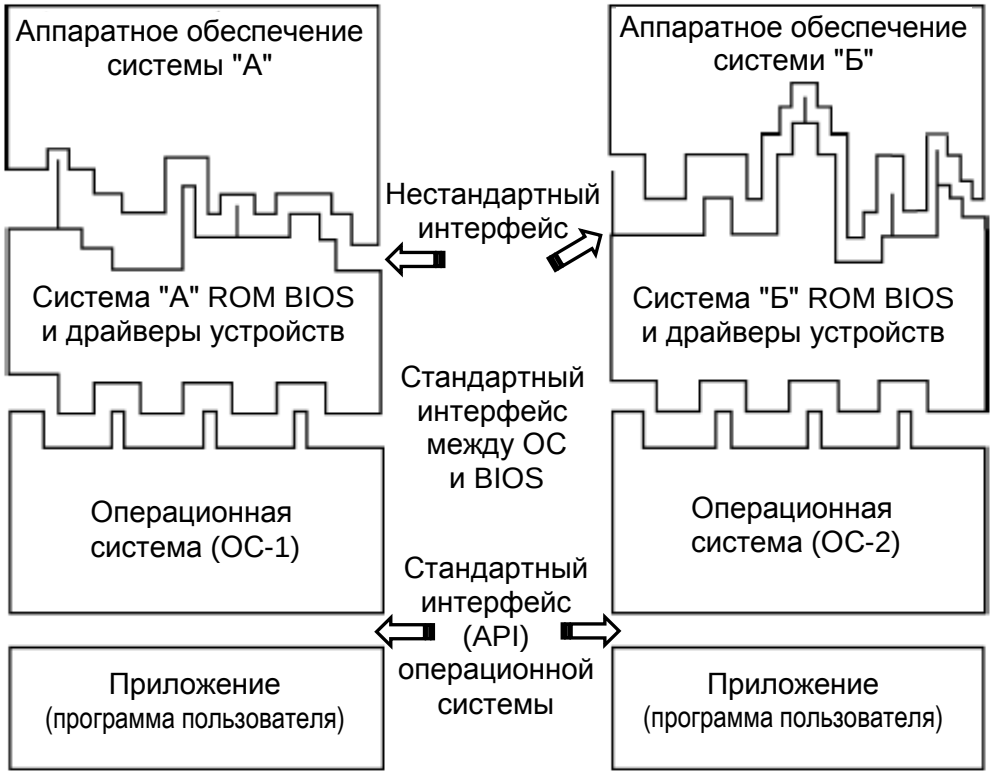


Рис. 2.14. Условное деление ПК на несколько слоёв

[12] firmware – программно-аппаратное обеспечение, программно-аппаратные средства, аппаратно-программное обеспечение, т.е. комплекс совместно взаимодействующих при работе програмно-аппаратных компонент.

Связь между приложениями и операционной системой осуществляется с помощью соответствующего стандартного для всех приложений API (Application Programming Interface). Этот Интерфейс Прикладного Программирования определяет, например, как выполняется запись или считывание данных на диск, их печать и другие функции.

Поскольку приложение не зависит от присоединённого аппаратного обеспечения (т.е. имеющихся внешних и электронных устройств), то и все его вызовы требуемых функций обрабатывает операционная система, которая уже располагает информацией об установленном оборудовании и через BIOS обращается непосредственно к его компонентам. Эта связь поддерживается драйверами соответствующих компонентов оборудования. Причём каждой операционной системе – DOS, Windows9x, Windows NT, Windows 2000, OS/2, Linux или другой – для одного и того же устройства необходимы свои драйверы. Как видно из рисунка 2.14, приложения и операционная система идентичны у большинства компьютеров, а BIOS «подстраивается» под конкретную конфигурацию аппаратного обеспечения и, независимо от установленного оборудования, обеспечивает стандартный интерфейс для работы с ними операционной системы. Таким образом, BIOS представляет собой интерфейс между аппаратурой конкретного ПК и установленной на нём операционной системой.

Как правило, прикладная программа не работает напрямую с аппаратурой, а пользуется услугами операционной системы. Исключение составляют случаи, когда пользователь самостоятельно реализует доступ к устройствам или их компонентам из своей программы. Для IBM совместимых ПК, как правило, используется механизм драйверов. Однако драйверы ОС не всегда обращаются прямо к аппаратуре. Обычно они вызывают функции BIOS, и уже BIOS выполняет все действия по вводу-выводу на уровне своих прерываний. Естественно, что BIOS содержит программы обслуживания только стандартных устройств ввода-вывода, а нестандартные обслуживаются собственными драйверами.

Использование BIOS как дополнительного интерфейса между драйверами стандартных устройств и аппаратурой резко повышает "живучесть" используемой ОС на "не совсем совместимых" с IBM PC компьютерах других производителей. Это возможно в связи с тем, что производители совместимых компьютеров учитывают в программах BIOS особенности постоянно появляющихся образцов нового оборудования. В этом случае и ОС, и программа пользователя, тем более, не видят никаких отличий новых компонентов от старых.

С другой стороны, пользователь может легко дополнять ОС своими собственными драйверами, которые составлены для нестандартных устройств, либо заменять стандартные драйверы и функции BIOS. При этом нужно особо отметить, что поскольку драйвер должен учитывать все детали конструкции каждого нового устройства и работать в режиме реального времени, хотя бы часть его должна быть написана на ***машинно-ориентированном языке программирования***.

2.4 Управление устройствами с помощью драйверов

Управление внешними устройствами – это одна из важнейших функций любой операционной системы. Система должна обеспечивать эффективный и удобный доступ к периферийным устройствам, а также возможность унифицированной, независимой от вновь создавемых устройств разработки программного обеспечения. Все новые устройства требуют некоторой дополнительной программной поддержки, обеспечиваемой т.н. драйверами устройств, связывающими их с операционной системой и с программами, работающими под её управлением. Для выполнения специфических операций по обмену данными ПК с новым устройством соответствующий драйвер следует стандартному набору правил взаимодействия с ОС. Он считывается с диска и присоединяется к операционной системе ***в процессе её начального вызова***. Часть драйверов, необходимых для работы ПК ***после загрузки ОС***, последняя, в соответствии с указанными в специальных областях жёсткого диска данными, загружает в ОЗУ позднее. Информация о том, что должен быть загружен драйвер конкретного устройства, к примеру, в ОС DOS предоставляется ей в файле CONFIG.SYS.

Рис. 2.15. Разные устройства компьютера

Номенклатура вновь создаваемых внешних устройств чрезвычайно велика. К ним можно отнести и новые типы внешней памяти (Zip, Jazz, стриммеры, магнитные диски повышенной ёмкости и др.), беспроводные клавиатуры и манипуляторы типа мышь (рис. 2.15), тьюнеры, видео приставки, цифровые фотокамеры, микрофоны и многое другое.

Большинство из них обмениваются данными с процессором асинхронно, то есть через неровные отрезки времени. Несмотря на это, технологии создания соответствующих драйверов решают практически любые задачи. Рассмотрим процесс взаимодействия комплекса устройств, которые поддерживаются соответствующими драйверами.

Итак, любое внешнее устройство характеризируется уникальным унифицированным интерфейсом обмена данными с компьютером, а также набором внутренних команд. В их состав, как правило, входят следующие:

❶ инициализации, которые приводят устройства в готовность к работе;

❷ управления компонентами (механическими, электрическими, электронными и др.) данного устройства;

❸ управления обменом данных (т.е. пересылкой) от компьютера к устройству и назад;

❹ завершения процесса совместной работы (очистка регистров, буферов, сброса флагов, отключения питания элементов, которые завершили работу и т.д.).

Каждая версия операционной системы концептуально разрабатывается один раз, а внешние устройства каждый год во всём мире появляются десятками тысяч. Поэтому представляется важным со стороны ОС рассматривать внешнее устройство как некоторый абстрактно обобщённый объект, который имеет неизменный интерфейс, то есть унифицированные средства доступа к нему и обмена данными с ним. А все тонкости, которые относятся к специфике конструкции и функционирования реального устройства, разработчики стремятся «упрятать» в тело драйвера, который программируется.

При реализации указанной концепции для доступа к внешнему устройству в DOS и в многих других ОС используется универсальная абстракция файла. Важно отметить, что, как правило, файл практически в большинстве операционных систем представляет собой комплексную структуру, которая включает (рис. 2.16):

❶ имя, с расширением из трёх символов, которое содержится в каталоге соответствующей дисковой системы (к примеру, FAT, FNTS и др.);

❷ байт атрибутов файла (файл только для чтения, системный и т.д.);

❸ время и дату создания файла или его модификации;

❹ размер файла в байтах;

❺ ссылка на первый кластер[13] магнитного диска, с которого начинается размещение всего файла;

❻ в таблице размещения файлов размещаются номера кластеров, в которых файл размещается целиком (!) и т.д.

Кроме обычных файлов или каталогов, которые реально занимают память (место) на магнитных дисках, файловая система содержит так называемые специальные файлы, для которых, как и для настоящих файлов, отводятся отдельные (логические) имена, но которым на самом деле соответствуют внешние устройства. Такое решение позволяет естественным образом работать в одном и том же интерфейсе с любым файлом или внешним устройством. (На самом деле, в некоторых случаях использование нестандартных внешних устройств нередко может выходить за границы стандартного интерфейса). Понятно, что простое объявление внешнего устройства специальным файлом не даёт возможности работать с этим устройством, если не создан и

[13] Кластер – группа блоков (секторов) диска, которые объединяются в единое целое и затем рассматриваются системой как комплексная единица записи. В MS DOS и некоторых других операционных системах– минимальная единица деления дискового пространства. Состоит из одного или нескольких соседних секторов. Размер сектора, как правило, кратен степени числа 2. Может иметь значения: 124, 256, 512 или больше килобайт.

соответствующим образом не подключён к системе специальный программный код, который отвечает специфике взаимодействия с данным устройством.

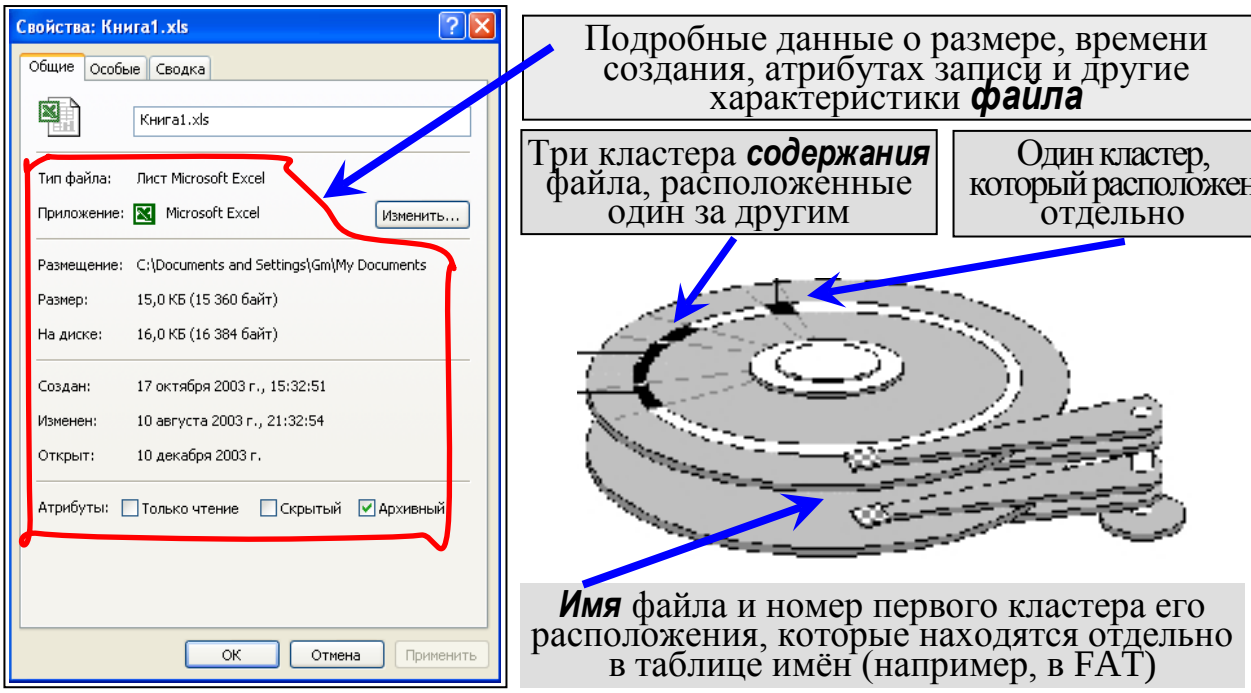


Рис. 2.16. Комплексные характеристики, которые описывают файл, хранящийся на магнитном диске

Как и у большинства современных операционных систем, такого рода ***программный код*** в любой ОС DOS зовётся ***драйвером устройства*** (в этом контексте слово драйвер лучше всего понимать в значении "управляющая программа"). Так же, как и в любой другой системе, драйвер устройства – это многовходовый программный модуль (со своими статическими данными), который должен и умеет инициировать работу с устройствами. То есть выполнять ***обмены данными*** (т.н. двунаправленные их пересылки между пользователем и устройством), которые заказывает пользователь, терминировать работу с устройством и обрабатывать прерывания, поступающие от него.

Файл с драйвером устройства имеет почти стандартный формат исполняемой программы с добавлением некоторой идентифицирующей информации, характерной именно для драйвера. В большинстве своём, существует два типа драйверов устройств: ***драйверы символьных устройств***, которые подобно клавиатуре, экрану дисплея, принтеру или коммуникационному порту работают с последовательным потоком символов и ***драйверы блочных устройств***, которые подобно дисководу читаю и пишут произвольные блоки данных, для ссылки на которые используются некоторые разновидности адреса блока. Для идентификации символьных устройств используются логические имена (например, LPT1:, COM1:, PRN). Поэтому такие устройства системой могут рассматриваться как файлы. Для идентификации блочных устройств используются буквы, присваиваемые

операционной системой и аналогичные идентификаторам дисководов A, B, C, D, E и т.д.

Как пример использования имён файлов и устройств, можно рассмотреть команду ОС DOS «*COPY*», предназначенную для копирования данных из одной области их хранения в другую. В зависимости от операндов, указываемых в её адресной части может меняться и смысл её работы (к примеру, разные варианты адресных имён областей источников данных, имён устройств приёмников данных, изменение их последовательности и т.д.) (рис. 2.17, 2.18).

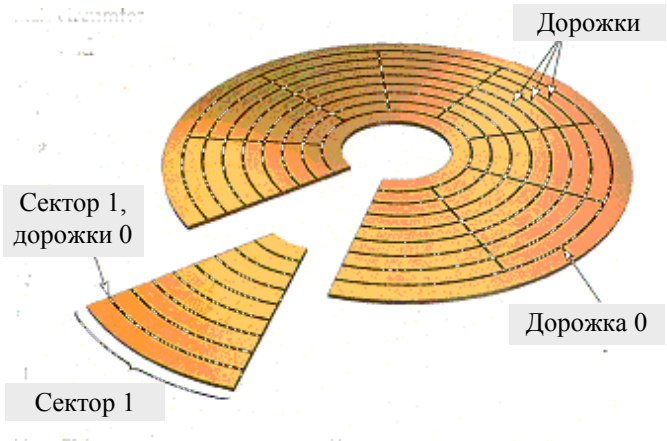


Рис. 2.17. Размещение секторов и дорожек на поверхности пластины магнитного диска

№ пп	Имя команды	Первый объект переписи (источник данных)	Второй объект переписи (приёмник данных)
1	**COPY**	Источник данных	Приёмник данных
2	**COPY**	Имя файла	Адрес, куда его необходимо записать
3	**COPY**	Путь к файлу, который переписывается	Адрес конечной записи
4	**COPY**	Имя каталога	Адрес назначения
5	**COPY**	Имя устройства, которое передаёт данные	Имя файла, который получает данные
6	**COPY**	Имя файла, из которого извлекаются данные	Имя устройства, куда данные выводятся/размещаются
7	**F5**	Выделенный объект или группа объектов	Каталог либо имя устройства
8	**COPY** (из контекстного меню объектов рабочего стола Windows)	Выделенный объект или группа объектов	Место назначения: папка, пиктограмма устройства (диска, принтера и др.), и даже (!!!) ***документ приложений*** *MS Office*

Рис. 2.18. Некоторые трактовки использования команды копирования данных *COPY* в разном контексте

Обычно, команда *COPY* обеспечивает перепись объектов с одного места диска (источника данных) на другое (приёмник данных). Объектами

переписи, как правило, выступают файлы или целые каталоги (они же директории или папки).

Но в указанном пользователем порядке следования операндов в адресной части команды, источниками и приёмниками данных могут быть также и конкретные устройства, которым присваиваются логические имена *CON, PRN* и др. либо меняется контекст операции переписи (табл. 2.2).

Таблица 2.2

Использование устройств, как источников или приёмников объектов файлового типа

Текущий диск	Команда и её адресные части	Результат выполненного действия
C:\>	**COPY** A:\PROG C:\FRAG	Каталог PROG с диска A: перепишется в каталог FRAG на диске C:
C:\>	**COPY** **CON** FILE1.txt	Данные, которые вводятся с клавиатуры, вводятся в файл с именем FILE1.txt. (**CON** – файл-источник данных). После окончания ввода текстовых данных в файл FILE1.txt необходимо нажать сочетание клавиш Ctrl+Z, что приведёт к его закрытию с добавлением в его конце признака конца файла.
C:\>	**COPY** FILE1.txt **CON** **TYPE** FILE1.txt	Текстовые данные, которые содержатся в файле с именем FILE1.txt, выводятся на экран дисплея компьютера (**CON** – файл приёмник данных)
C:\>	**COPY** FILE1.txt **PRN**	Копирование текстовых данных из файла FILE1.txt на любой принтер, подключённый к ПК

Таким образом, если логическое имя CON располагается на первом месте в команде COPY, то оно трактуется как логическое имя системного устройства ввода, то есть ***клавиатуры***. После завершения ввода текстовых данных в открытый командой COPY файл с заданным именем и последующего закрытия этого файла сочетанием клавиш [Ctrl+Z], что является командой прекращения приёма данных при записи на диск, можно просмотреть содержание этого файла на экране дисплея. В этом последнем случае, CON – логическое имя устройства вывода, то есть в данном случае ***дисплея***, хотя по сути, действие, которое декларируется – обозначает вывод (перепись) одного файла (FILE1.txt) в другой (CON). При необходимости вывода текстового файла на печатающее устройство (***принтер***), достаточно указать логическое имя PRN. Концепция абстрактного файла в этом случае выдерживается полностью, так как указывая для вывода данных стандартное логическое имя CON (или любое другое) мы абсолютно не задумываемся о призводителе реально используемого физического устройства (к примеру, принтера фирм XEROX, Hewlett-Packard, Canon и др.), его быстродействии (10, 15, 20 строк и выше в минуту), его конструкции (матричный, струйный, лазерный чёрно-белый, лазерный цветной и др.).

Коцептуальность и универсальность подобного подхода заключается в том, что во всех вышеуказанных случаях, пользователь абсолютно ничего не должен знать (и, как правило, и не знает!) о типе используемого им устройства (беспроводное, лазерное, формат записи, тип носителя, уровень быстродействия, название фирмы производителя и т.д.). Всё это уже решено на уровне BIOS, драйверов всех известных (!) и будущих (!) устройств, а также операционной системы, использующей предоставляемые BIOS и драйверами интерфейсы нижнего уровня работы с аппаратурой ПК. Такой подход полностью сосответствует условиям интероперабельности[14] во взаимодействии компонентов информационных систем между собой и пользователем.

Вопросы.

1. Как называются основные аппаратные компоненты компьютера?
2. Какие компоненты входят в состав материнской платы?
3. Что представляет собой базовая система ввода - вывода ПК (BIOS – Basic Input/Output System)?
4. Какие пять основных функций выполняет BIOS после включения ПК?
5. Чем ″тёплый″ перезапуск ПК отличается от ″холодного″?
6. Каковы основные функции записи Master Boot Record?
7. Где может размещаться Master Boot Record?
8. Что такое порт ПК?
9. Что такое шина ПК?
10. Что представляет собой драйвер и для чего он служит?
11. Каковы основные характеристики файла?
12. Что общего у файла с внешним устройством компьютера и в чём их различия?

[14] Интероперабельность – взаимная возможность/способность информационных систем или компьютеров обмениваться сообщениями, выполнять программы или пересылать данные между их разными функциональными блоками таким образом, чтобы пользователь при этом практически ничего не должен был бы знать об особенностях этих блоков. Поддерживается средствами развитых многоуровневых интерфейсов.

Оператор (языка) – базовая единица *действия* в языках программирования и алгоритмических языках.

Программа – упорядоченная последовательность *команд*, подлежащая обработке.

ГОСТ.

3. КОМАНДНАЯ ОСНОВА РАБОТЫ КОМПЬЮТЕРА

3.1. Роль команд в процессе управления компьютером

Персональные компьютеры (ПК) сегодня можна увидеть на рабочем месте специалиста практически любой профессии. Они стали неотделимой частью производственной деятельности организаций и предприятий, расширяя область промышленной обработки данных. Такая ситуация возникла по многим причинам, и в первую очередь потому, что ПК стал доступным, миниатюрным и удобным устройством для ввода, хранения, обработки и вывода разнообразной информации практически во всех известных предметных областях[1] (рис. 3.1). Более того, первый вопрос при приёме специалиста на работу в организацию любой формы собственности формулируется очень просто: "Компьютером владеете?". Так что, речь идёт о новой форме грамотности – ***компьютерной грамотности***.

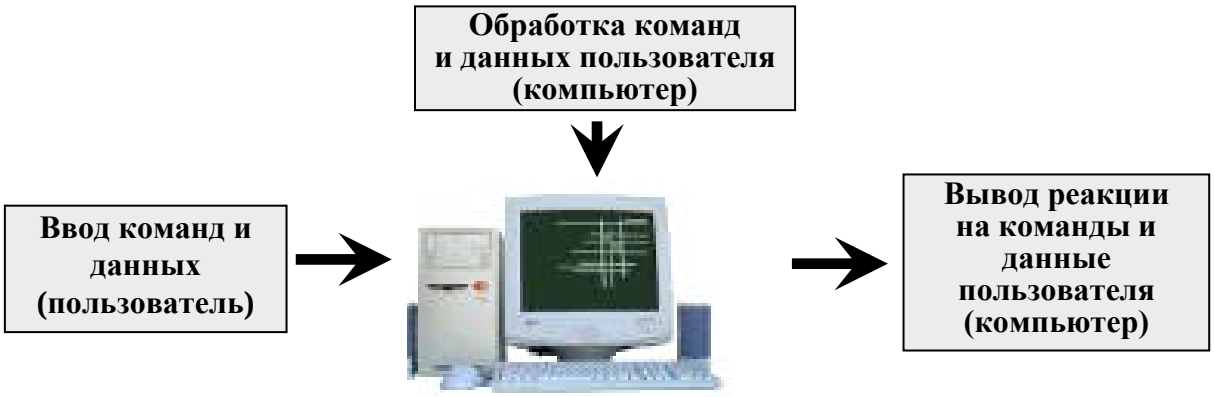


Рис. 3.1. Процесс решения задачи пользователем на компьютере

И чтобы овладеть этой грамотностью в условиях, когда функции компьютерной техники постоянно совершенствуются и усложняются, необходимо хорошо представлять особенности его работы и принципы функционирования.

Как известно, процесс взаимодействия пользователя с ПК в процессе решения задачи основывается на нескольких ключевых моментах.

Во-первых, при роботе с компьютером им необходимо постоянно ***управлять***, то есть направлять его действия в необходимое русло. Ибо, если

[1] Предметная область – класс задач, которые решаются программным средством или программной системой.

после его включения не выполнять никаких действий, сам компьютер не выполнит ни одной конкретной задачи, пока не дооождётся конкретных действий (***команд***) пользователя. ***Последние вводятся, как правило, путём нажатия клавиш на клавиатуре ПК или кнопок мыши, после подведения курсора к управляющим элементам приложений или операционной системы.*** Эти действия принуждают те компоненты, которые их получили, в свою очередь, порождать командные стимулы на уровне этих программных компонентов и так далее вниз по иерархии. Чтобы вникнуть в суть процессов, которые происходят, необходимо рассмотреть самый верхний уровень представления понятия и термина "команда" для ПК, который отвечает уровню взаимодействия последнего с пользователем.

Как правило, под ***управлением*** понимают процесс целенаправленного воздействия на систему, который обеспечивает повышение уровня её организованности, с целью достижения того или другого полезного результата. При этом любая система управления разделяется на две подсистемы: ***управляющую*** и ту, которой ***управляют***. Связь между управляющей подсистемой и той, которой управляют, называется прямой связью. Такая связь существует всегда. Противоположная по направлению связь называется обратной. Понятие обратной связи является фундаментальным[2] в технике, природе и обществе. Опыт показывает, что управление без сильных обратных связей не эффективно, так как не обладает свойством к самообнаружению ошибок, формулированию проблем, не позволяет использовать возможности саморегулирования системы, а таже опыт и знания специалистов. В комплексе ***пользователь-компьютер*** прямой связью является воздействие на компьютер с целью получения решения некоторой задачи, а обратной – сообщения о ходе её решения (рис. 3.2).

Рис. 3.2. Управляющие связи, существующие при взаимодействии пользователя с компьютером

[2] Фундаментальный – нечто, составляющее основную часть или основу чего-либо. Обычно применяется в качестве характеристики основополагающих элементов наук или прикладных отраслей знаний. К примеру, фундаментальные исследования, фундаментальные понятия.

Во-вторых, осуществление прямых и обратных связей между двумя системами реализуется на уровне ***интерфейса***[3] (в данном случае – интерфейса компьютера), который обеспечивает ***интероперабельность***[4] между системами, которые взаимодействуют, то есть между ***пользователем*** и ***ПК***. Основные задачи поддержки интерфейса ПК возлагаются на операционную систему и будут обсуждены далее в главе 4. Интерфейс приложений, которые работают под управлением ОС, создаются их разработчиками и зависят от конкретных задач, которые решаются с их помощью.

И, наконец, ***в-третьих***, как управление компьютером, так и обеспеченне работы его и всех его компонентов обеспечивается путём выполнения разнообразных ***команд, из которых формируются программы или последовательности наборов команд для описания действий компьютера, которые приводят к решению задач пользователя***.

Так, во многих современных приложениях (MS Word, MS Excel, MS Access и др.) решение любой типовой задачи реализуется ***последовательностью щелчков левой кнопки мыши на названиях меню, командах из выпадающих списков, кнопках стандартной панели инструментов и других управляющих элементах***, которые, в своей совокупности и выполнении в определённой последовательности, играют, по сути, роль ***программы, составленной из «элементов языка» данного приложения*** (рис.3.3).

ЗАДАЧА	**ПРОГРАМА ДЕЙСТВИЙ.**
Отформатировать диапазон ячеек B4:F16 в денежном формате.	1. Выделите диапазон ячеек **B4:F16**. 2. В меню ***Формат*** выберите команду ***Ячейки***. 3. Щёлкните по вкладке ***Число***, если она не активна. 4. В списке ***Числовые форматы*** выберите *денежный формат*. 5. Установите в поле ***Число десятичных знаков*** значение 0. 6. Щёлкните **ОК**, чтобы вернуться на лист.

Рис. 3.3. Программа (действий) для форматирования диапазона ячеек в приложении (языке) MS Excel

За время длительного периода развития компьютеров термин ***команда*** вобрал в себя множество значений и приобрёл не меньше оттенков. Вне компьютерного мира слово ***команда*** имеет два основных значения:

❶ коллектив людей, которые объединены общей целью (футбольная, пожарная и другие команды);

❷ устное приказание выполнить какое-либо действие (например, налево, направо, стой, кругом и т. д.).

[3] Интерфейс – средство объединення (стыковки) двух или более частей нескольких систем.

[4] Интероперабельность – способность обмениваться сообщениями, выполнять программы или пересылать данные между разными функциональными элементами систем.

В информатике ***значений*** и ***смыслов*** слова ***команда*** Вы можете найти значительно больше, чем во всех других сферах деятельности человека. Они присутствуют на разных ***уровнях***[5] существования и представления компонентов ***компьютерных*** систем и ***информационных*** потоков и означают инициализацию того или другого процесса, происходящего на стадиях ***проектирования, реализации*** или ***работы*** многочисленных элементов ПК.

3.2. Физический и логический уровни применения команд

Следует отметить, что наиболее важными в информационных технологиях являются ***физический*** и ***логический*** уровни представления компонентов компьютера. Последний превратился за последние 20 лет в технологически и технически чрезвычайно сложный комплекс разнообразных взаимодействующих между собой устройств: клавиатуры, мыши, разнообразных чипов и чипсетов, микропроцессоров, дисководов для съёмных и несъёмных носителей информации, дисплея, принтера, сканера и многих других устройств (рис. 3.4).

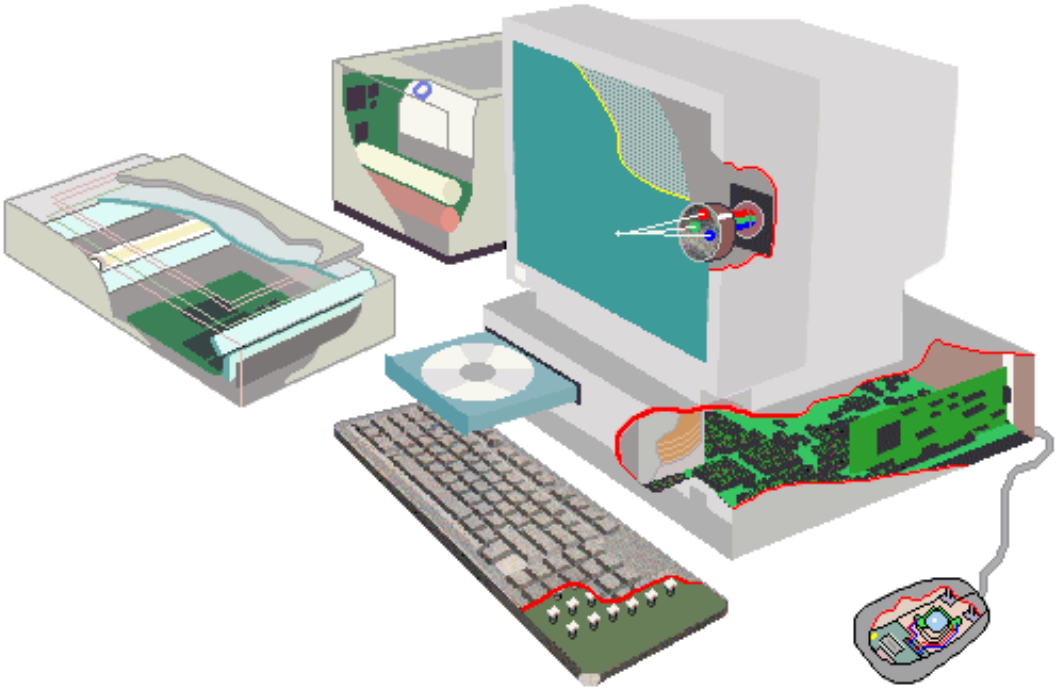


Рис. 3.4. Персональный компьютер в комплексе

Физический уровень определяет конкретную технологическую реализацию того или иного устройства или элемента. Например, дисплей ПК может быть конструктивно реализован на электронно-лучевой трубке (рис. 3.4) или на основе жидкокристаллической или плазменной технологий.

[5] Специалисты рассматривают девять значений понятия ***уровень***, который является основным для концепции иерархии систем. Термин «уровень» может означать: 1) степень вообще; 2) степень сложности; 3) степень глубины аналитического исследования; 4) возникновение (естественной, животной) организации более высокого уровня по сравнению с имеющейся; 5) систему взаимосвязанных свойств или переменных; 6) разряд; 7) слой; 8) основной слой; 9) уровень.

Внешнее запоминающее устройство компьютера может представлять собой флоппи-дисковод или жёсткий диск, и каждый со своей системой команд доступа, количеством секторов, дорожек, цилиндров и т.д. Представьте себе ситуацию, когда Вам для того, чтобы переписать файл с одного устройства на другое будет необходимо вникать во все технические тонкости отличий принципиально различных дисковых устройств, и к тому же, самому писать программы обмена между ними.

Для упрощения понимания и повышения безопасности использования устройств ПК, при работе пользователя с ОС вводится понятие логического уровня. Это позволяет абстрагироваться от всех технических и программных тонкостей работы с этими устройствами и оперировать, например, именами логических дисковых устройств в ОС MS DOS (A:, C:, D: и т.д.), или графическими представлениями этих же дисковых устройств на рабочем столе ОС Windows. Таким образом, можно сказать, что логический уровень – это некий абстрактный или концептуальный[6], то-есть виртуальный[7] уровень, который отображает некий образ, а не реальные физические объекты таким образом скрывает принципы их реализации.

Это означает, что когда Вы обращаетесь к логическому имени флоппи-диска или жёсткого диска для Вас не имеют значения ни их конструктивные особенности, ни тонкости технической реализации (может быть только объём свободного дискового пространства!) (рис. 3.5).

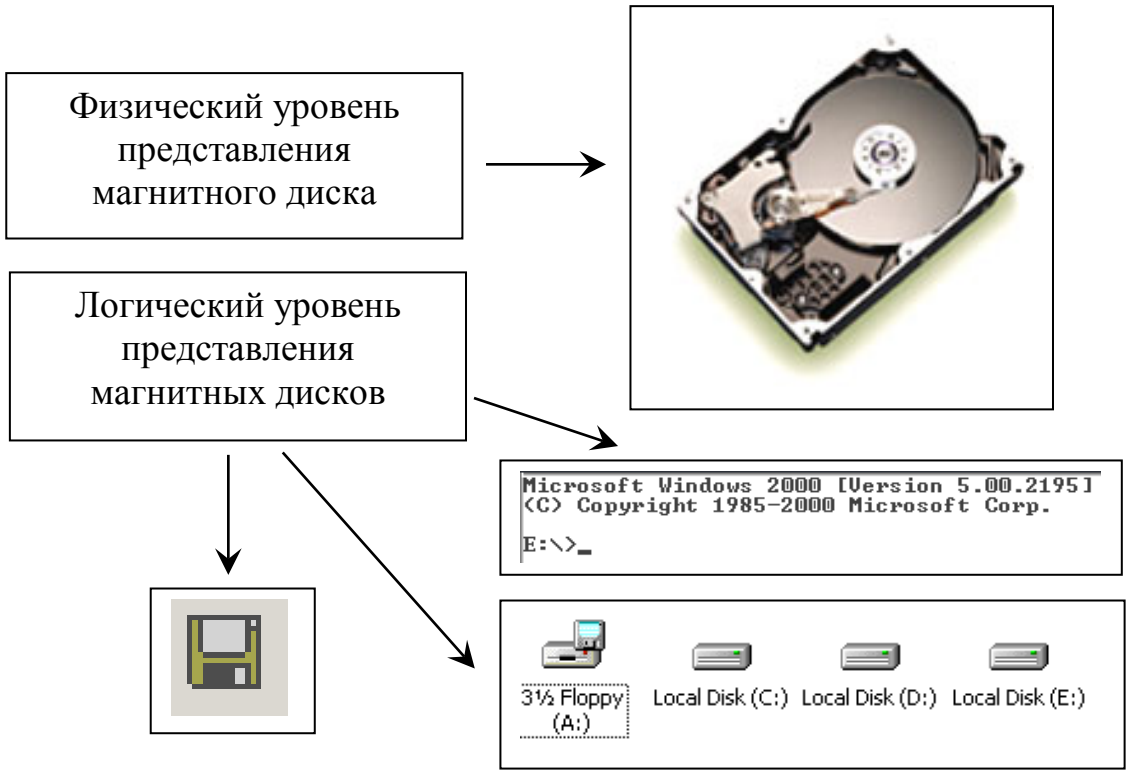


Рисунок 4.5. Уровни логического представления магнитных дисков в ПК для обращения к ним и работы с ними

[6] Концепция – определённый способ понимания, трактовка какого-либо предмета, явления, процесса.
[7] Виртуальный – не имеющий физического воплощения или воспринимаемый иначе, чем реализован.

Таким образом, логическим именем диска может служить абстрактное обозначение устройства компьютера в виде дополнительного текстового и/или графическог имени/обозначения, которые приписываются операционной системой для удобства их использования. Логическое имя даёт возможность пользоваться объектом не углубляясь в особенности его физической реализации. Например, каждый ***логический*** диск ПК является частью ***физического*** жёсткого диска, которая рассматривается как отдельный жёсткий диск со своим именем накопителя (рис. 3.6).

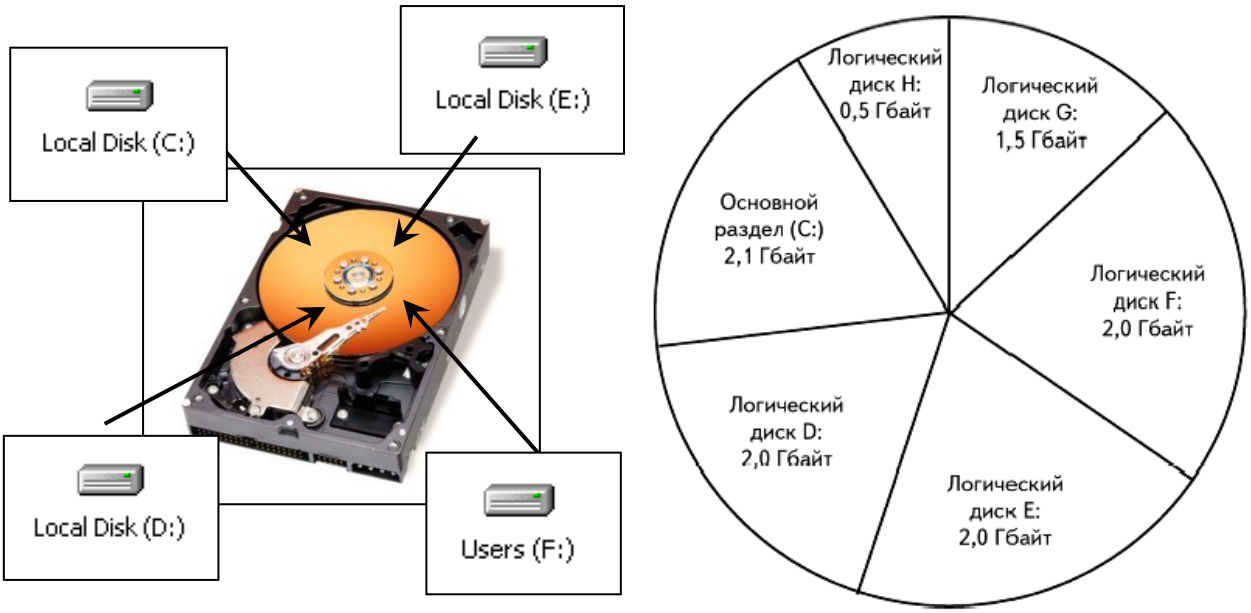


Рис. 3.6. Размещение логических дисков на физическом диске

3.3. Команды физического и логического уровней

Многие компоненты и устройства ПК содержат в своём составе разнообразные интегральные схемы, самая большая из которых в ПК обычно является процессором. Со времён изобретения первого в мире микропроцессора (МП) Intel 4004 в 1971 году, в котором содержалось 2 300 полупроводниковых транзисторов прошло уже достаточно много времени. Но и до сих пор технически процессор реализуется в виде большой интегральной схемы (БИС), структура которой постоянно усложняется, а количество функциональных элементов (типа диод или транзистор) на ней постоянно возрастает (от 5 миллионов в процессоре Pentium II до 42 млн. транзисторов в Pentium 4). И это не предел. В конце 2002 года инженеры компании Intel открыли совершенно новый тип кремниевых транзисторов, скорость переключения которых примерно в 1000 раз превосходит аналогичный показатель

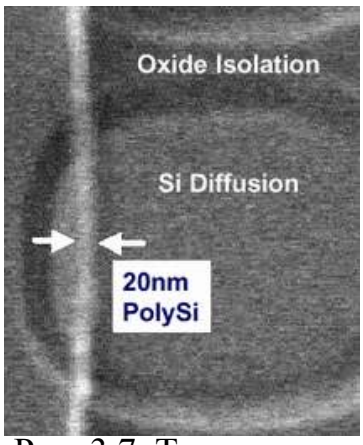


Рис. 3.7. Транзистор, являющийся элементом интегральной и электрической схем, представляющих микропроцессор

транзисторов, используемых в современных чипах. Новый транзистор имеет ширину всего в 80 атомов (20 нм), что позволит уместить в будущем на одном кристалле свыше миллиарда таких элементов (рис. 3.7).

Несмотря на такой невероятный рост концентрации электронных элементов в процессоре, работа всех этих транзисторов в электрических схемах и сами электрические схемы практически не изменились со времён их изобретения.

Командами открытия и закрытия транзисторов на уровне электрических схем и цепей являются электрические ***сигналы***[8], которые называются ***электрическими импульсами***[9] (рис. 3.8). Таким образом, необходимо чётко представлять, что на самом нижнем уровне абстракции, то есть на физическом уровне конструкции элементов ПК, понятие ***команда*** относится к средствам управления электронными компонентами устройств, таких как ***процессор***[10], ***чипсет***[11] или ***материнская плата***, ***схемы памяти*** и других, которые построены на триодах, диодах, сопротивлениях, проводниках и т.д. (рис. 3.8 и 3.9).

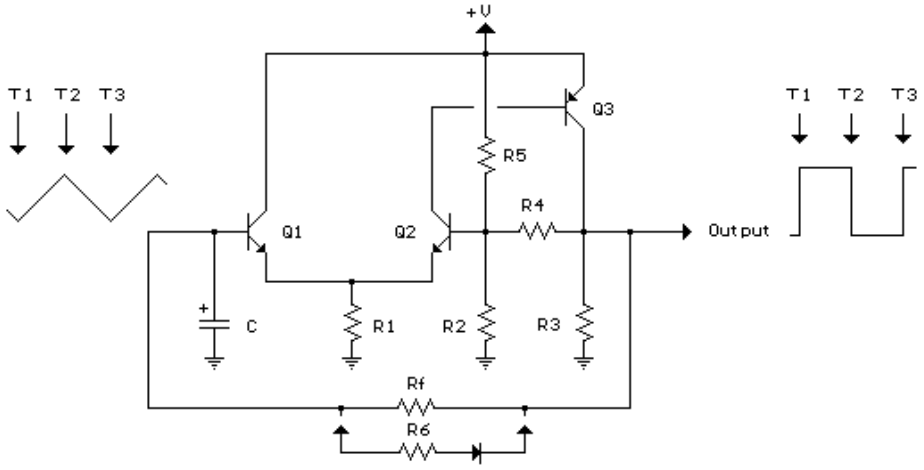
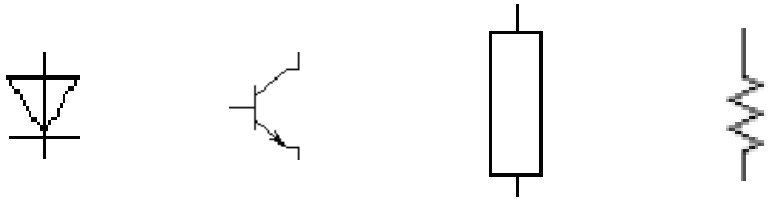


Рис. 3.8. Электрическая схема преобразования пилообразного сигнала в прямоугольные импульсы



А) Б) В) Г)

Рисунок 4.9. Обозначения диода (А), транзистора (Б) и сопротивлений (В,Г)

[8] Сигнал – изменяющийся во времени физический процесс, отражающий передаваемое сообщение.

[9] Электрический импульс – кратковременное отклонение напряжения или тока от некоторого постоянного значения.

[10] Процессор (микропроцессор) – это микросхема, реализующая функции центрального процессора персонального компьютера.

[11] Чипсет – набор микросхем материнской (системной) платы, реализующих архитектуру компьютера.

Импульсы обеспечивают не только слаженное взаимодействие, включение и выключение самих устройств и блоков, построенных на этих устройствах, но и моделируют собственно основу функционирования ПК – цифровые значения 0 и 1.

Проектирование компьютерных и микропроцессорных устройств начинается на логическом уровне путём разработки логических схем работы нового устройства по заданным требованиям. Сложные логические системы строятся из ***типовых логических узлов***, представляющих собой схемы из логических вентилей. К типовым логическим узлам, играющим основополагающую роль в микропроцессорных и компьютерных системах, относятся: ***триггеры*** (рис. 3.10), ***регистры, счётчики, дешифраторы, селекторы, схемы выполнения арифметических операций (сумматоры и вычитатели), а также системы шин***.

Транзисторы и другие элементы микропроцессоров набираются в логические схемы, элементами которых являются так называемые *вентили*, сочетания сигналов на входах и выходах которых реализуют модель поведения или протекания того или иного реального процесса. Рисунки с изображениями элементарных логических схем и их связей называются логическими диаграммами (рис 4.10), в которых при разработке логики работы компьютера и его устройств сообщения моделируются в виде ***сигналов (команд)***, подаваемых на их входы.

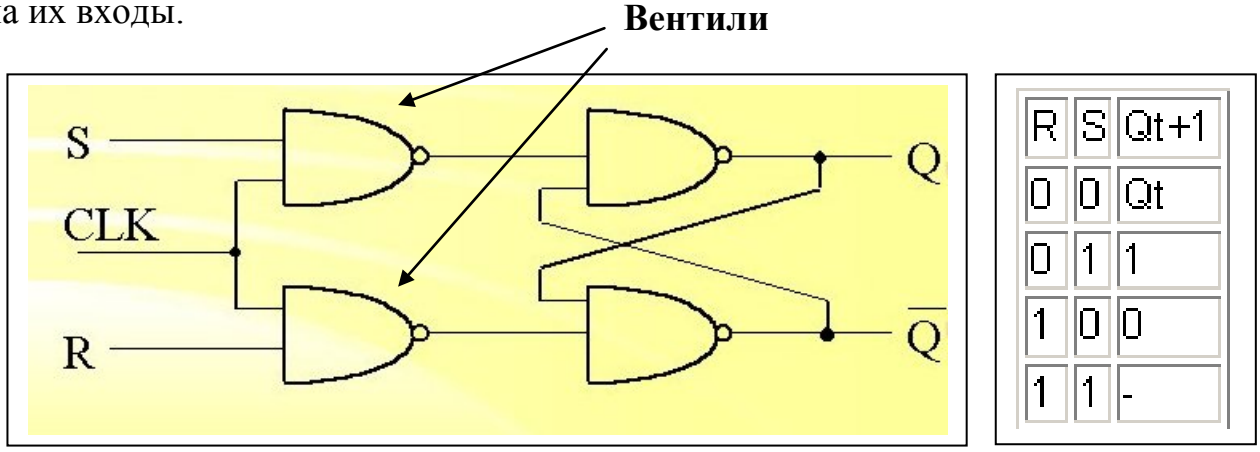


Рис. 3.10. Логическая схема RS-триггера и таблица сигналов и состояний на его входах

Одними из основных элементов МП являются ***триггеры*** – простейшие цифровые схемы последовательного типа. В зависимости от алгоритма работы, который реализуется, триггер может иметь установочные, информационные или управляющие входы. Он также имеет два устойчивых состояния: Q=1 и Q=0 (см. рис.3.10) и поэтому его иногда называют бистабильной схемой. В каком из этих состояний окажется триггер, зависит от сигналов на входах триггера и от его предыдущего состояния, то есть он имеет память. Таким образом, командами логического уровня для этого и других подобных устройств являются ***значения*** (сигналы) 0 и 1 на входах R и S (рис. 3.10).

Следует помнить, что логический (вентильный) уровень представления триггера, на физическом уровне может реализовываться транзисторами и резисторами с помощью различных технологий и из различных, вообще говоря, материалов.

С момента изобретения в 1959 году интегральных схем МП или как их ещё называют «чипов», лавинообразно растёт интерес к созданию на их базе электронных устройств, способных выполнять сложные функции управления. После внедрения микропроцессоров в производство карманных калькуляторов они начали широко применяться и в других отраслях. Основными достоинствами этих устройств, стали ***низкая стоимость*** и способность выполнять ***программы***. Другими словами, микропроцессор является ***программируемым*** логическим устройством, изготовленным в монолитном кристалле. Сам по себе МП не может решить ту или иную конкретную задачу. Для её решения его нужно запрограммировать, обеспечить необходимыми данными и соединить с другими устройствами. Например, МП может управлять светофорами на сложном перекрёстке, обеспечивая их переключение через заданные промежутки времени (рис. 3.11).

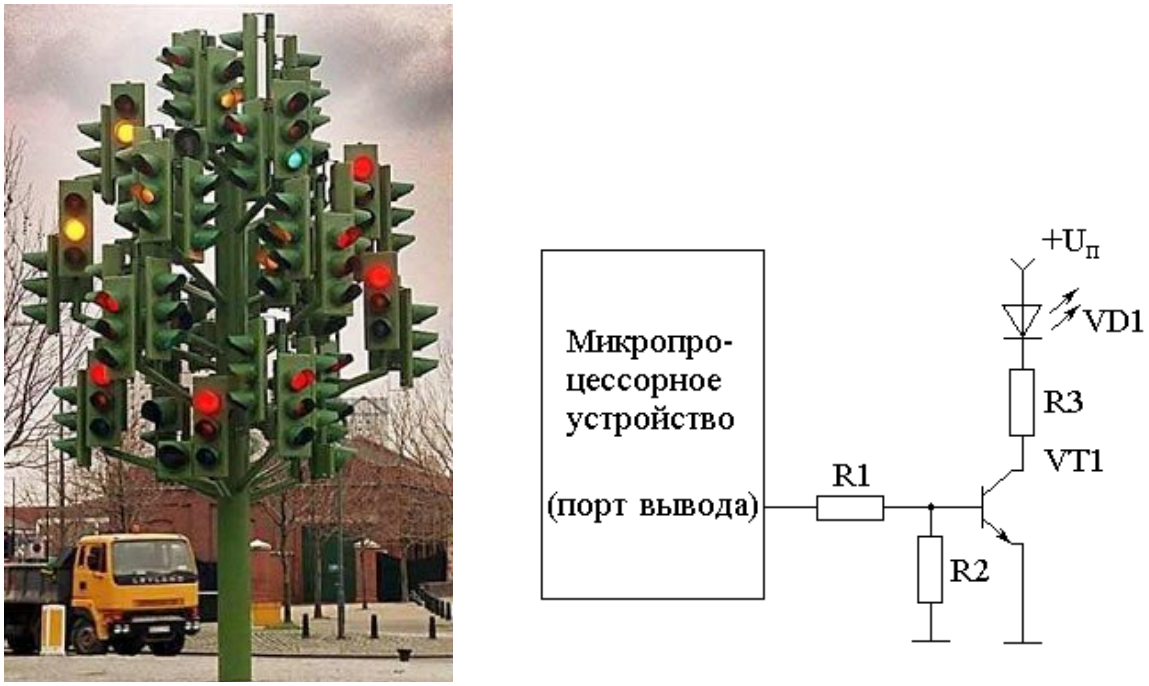


Рис. 3.11. Светофор на сложном участке дорожной развязки в Англии, управляемый с помощью микропроцессора и схема типового его подключения

3.4. Команды программных уровней и уровней работы с ОС

Следующим, более высоким уровнем представления команд, являются ***предписания, которые определяют элементарные шаги выполнения программируемых действий в работе конкретных устройств***, например,

запись, считывание, пересылка данных и так далее в регистрах и других блоках процессора. Для их обработки в интегральных схемах служат ***программный счетчик (счётчик команд), стек и регистр команд***. Понятно, что эти команды также управляют и работой всех других устройств и компонентов устройств ПК.

Команда микропроцессора обычно состоит из кода операции (КОП) и так называемой адресной части, где могут указываться адреса операндов в оперативной памяти (ОП, ОЗУ). Операндами называются данные, над которыми команда производит какое либо действие. Как правило, в адресной части команд указываются не сами операнды, а их адреса в операционных устройствах ПК (регистрах, сумматорах, оперативной памяти и др.).

Форматы команд достаточно сильно зависят от конструкции процессора. Для процессора, разработанного по архитектуре[12] Фон-Неймана, которой соответствуют все большие чипы фирмы Intel, команды строятся следующим образом (рис. 3.12).

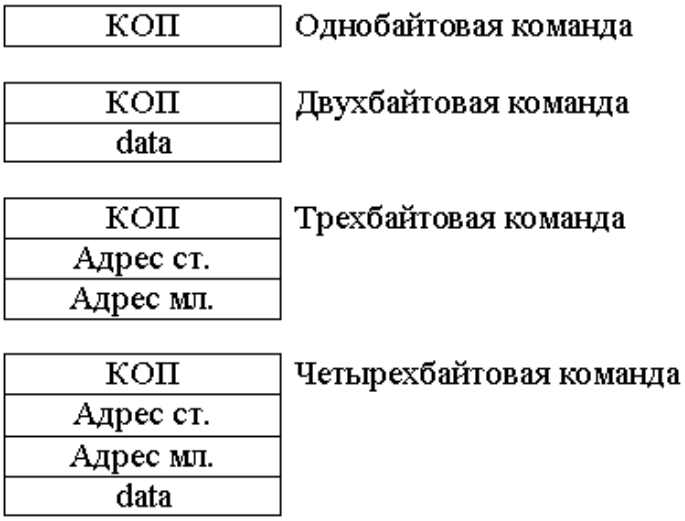


Рис. 3.12. Построение команд процессоров Intel

Внутренние команды процессора являются основой ***архитектуры вычислительной системы*** и, соответственно, ***компьютерной платформы***[13]. На их базе строятся практически все языки высокого уровня (ЯВУ), такие как C, Pascal, C++, Object Pascal, Java и многие другие. В микропроцессоре команды имеют цифровую кодировку (рис. 3.13), а данные обрабатываются в двоичной, восьмеричной, шестнадцатиричной и десятичной системах счисления (см. также Приложение 2).

[12] Архитектура – описание вычислительной системы на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд и средств пользовательского интерфейса, организации памяти и системы адресации, операций ввода-вывода и управления и т.д.

[13] Платформа – совокупность аппаратных средств, программного обеспечения и интерфейсов, используемых в конкретных компьютерах. Обычно платформа определяется применяемой операционной системой и процессором.

```
75CBFF75CAFB75CDFF75CCFB75985275
C834D28875D24A75D32075D50175330E
75345975A60175A11375D13230DFFD20
8929E5DB9533C2D71200F8E5DA9534C2
D71200F8E5DD1200F8E5DC1200F89001
181200D8B2B41200CF80CEC289E5DB94
```

Рис. 3.13.Фрагмент программы компьютера в цифровом виде (16-ричный код).

Таким образом, каждой команде соответствует некоторый цифровой код (к примеру, 01, 02 и т.д.), который специальным устройством ***дешифратором*** интерпретируется в соответствующую последовательность элементарных действий (посылок импульсов) исполняющим устройствам (триггерам, сумматорам, регистрам и др.).

Если для кода операции используется восьмибитное слово (байт), то при помощи этого слова можно закодировать 256 операций. В процессе разработки системы команд нового устройства для операции может быть назначен любой код. Именно системой команд и определяется конкретное семейство процессоров. Однобайтные команды позволяют работать с внутренними программно доступными регистрами процессора. При этом для выполнения одной и той же операции над ***разными регистрами*** процессора назначаются ***разные коды команд.***

С целью удобства написания программ для конкретных устройств компьютера разрабатываются специальные языки (программирования) нижнего уровня (ЯНУ) – ассемблеры. Точно так же называют и программу или пакет программ, которая осуществляет трансляцию (преобразование) исходного текста программы (исходный модуль) в машинные коды (объектный модуль). Далее, как правило, полученный код трансформируется в загрузочный модуль, т.е. программу, представленную в виде пригодном для загрузки и выполнения. Такой файл содержит программу в машинном коде, информацию для настройки адресов в памяти ПК и имеет расширение **.**EXE.

В ассемблерных программах, цифровым кодам команд, приводятся в соответствие мнемонические (текстовые) обозначения (к примеру, ADD-сложить, LOOP –цикл, MOV–переслать и т.д) с соответствующими адресными частями. Таким образом, для полного обозначения команды используется мнемоническое обозначение операции и символические имена используемых ею операндов, которые перечисляются через запятую. При этом в большинстве процессоров операнд приёмник данных записывается первым, а операнд источник данных – вторым.

Например:

```
MOV   R0, A   ;Скопировать содержимое регистра A в регистр R0
```

Инструкция (команда) сложения содержимого двух регистров для 16-разрядного процессора Intel 80286 на ассемблере выглядит следующим образом:

```
ADD   AX,BX   ;Сложить содержимое регистров AX и BX
```

Команда организации цикла имеет вид:

```
LOOP  MET1    ; MET1 – імя метки начала цикла
```

Команда сдвига влево на 4 бита содержимого регистра AX показана ниже.

```
SHL   AX,4    ; AX імя регистра
```

Важность языка ассемблера состоит ещё и в том, что, например, в действительности команда пересылки данных MOV – это целое семейство машинных команд микропроцессора. Для разных типов адресов и операндов, существует от семи (Intel 8080) и более различных вариантов данной команды. Ассемблер порождает правильную машинную запись, основываясь на типах операндов, которые указывает программист. И это одна из причин, по которой ассемблер требует для операндов назначения типов, т.е. ассемблер должен знать, что представляет собой каждый операнд – регистр, байт памяти, слово памяти, сегментный регистр, или что-либо другое.

Основой формирования программ для выполнения на компьютерах задач из разных предметных областей являются языки (программирования) высокого уровня (ЯВУ). В их структуру обязательно входят ***команды обработки данных, которые именуются также операторами программы или предложениями языка программирования***. Сюда можно отнести операторы (команды): цикла, условные, выбора и некоторые другие. Каждый из таких операторов реализуется в компьютере в виде набора базовых машинных команд процессора. Перевод каждого оператора и всей программы в целом в наборы соответствующих машинных команд осуществляется ***компилятором*** или ***интерпретатором*** используемого языка программирования.

Компилятором называется программное средство для преобразования текста программы из описания на входном языке (языке программирования) в ее представление на выходном языке (в машинных командах). Программная система, анализирующая команды или операторы программы и немедленно выполняющая их называется интерпретатором. Достаточно часто перевод программы при трансляции осуществляется сначала на язык ассемблера, а затем уже в машинные команды.

Например, рассмотрим простое математическое выражение:

f(n) = n! (вычислить значение факториала числа *n*)

Операторная реализация этого выражения на языке Турбо Паскаль будет выглядеть следующим образом:

```
...
Fct := 1;
For i := 1 to  n  do
 Fct := Fct * i;
...
```

Команды (инструкции) языка ТП

Фрагмент программы на языке Турбо Паскаль, который реализует вычисление факториала числа *n* и занесение этого значения в переменную Fct (ячейку с именем Fct)

Так как работу и взаимодействие пользователя с устройствам компьютера организует ***операционная система***, она должна учитывать, что на каждой компьютерной платформе существует свой, ***стандартный набор команд машинного уровня*** для управления вычислительной системой (компьютером) в целом. В данном контексте ***командой*** является предписание компьютеру или устройству выполнить определённый шаг на пути решения задачи. На рисунке 3.14 приведен фрагмент приведенной выше программы вычисления факториала на языке ассемблера для процессора платформы Intel.

	Код операции	**Адресная часть**	**Комментарии**
START:	mov	i, 1	; ***Заносим значение*** 1 ***в ячейку*** i
	mov	Fct, 1	; ***Заносим значение*** 1 ***в ячейку*** *Fct*
	mov	CX, n	; ***В регистр*** CX ***заносим*** n
	sub	CX, i	; ***Вычисляем разность и помещаем значение*** n-i ***в*** CX
	mov	AX, Fct	; ***В регистр*** AX ***заносим значенние*** Fct
	mul	i	; ***Там же в*** AX ***умножаем*** Fct ***на*** i
	mov	Fct, AX	; ***Возвращаем призведение в*** Fct
	loop	START	; ***Если содержимое*** CX ***не равно нулю, оператор*** loop ***отнимает из него единицу и переходит на метку*** START
	…		; ***В противном случае цикл завершается и управление передаётся следующей за циклом команде***
Примечание: mov, sub, mul и loop – команды языка ассемблера.			

Рис. 3.14. Фрагмент программы вычисления факториала на языке ассемблера

В отличие от команд разных программных уровней – команды операционной системы и приложений, которые работают под её управлением, имеют разнообразную внешнюю форму, хотя практически все они также работают с устройствами ПК (!).

В командной строке операционной системы MS–DOS команды представлены специальными (ключевыми) словами, которые понимает

программа или система (например, команды MS DOS: *MD*, *CD*, *COPY*, *DIR* и т.д.) (рис. 3.15).

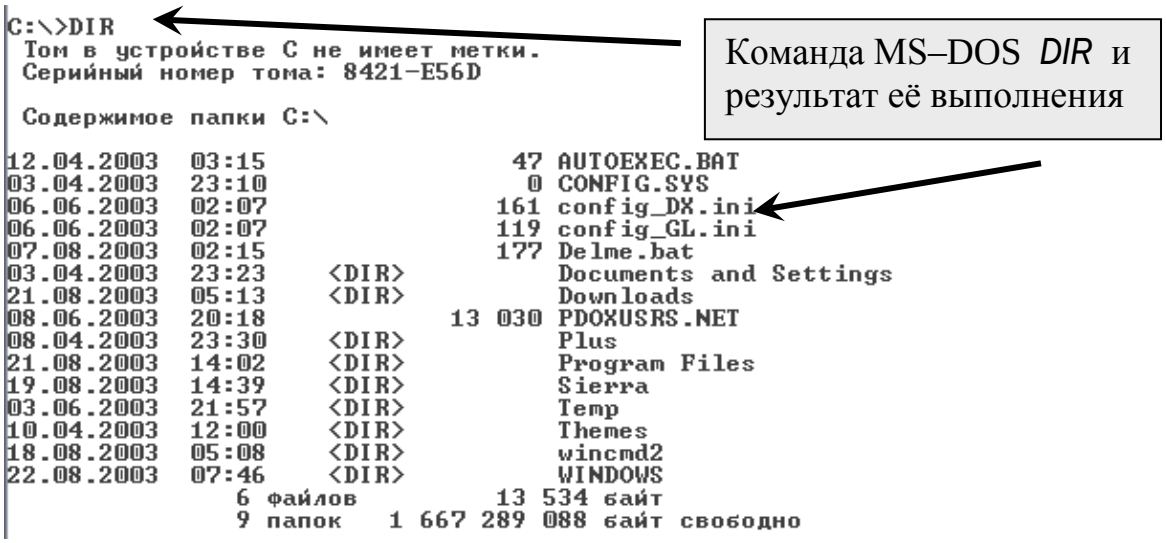


Рис. 3.15. Результат выполнения команды MS DOS *DIR*.

Ряд важных системных команд ПК представлен функциональными клавишами (Tab, Esc, Shift, Ctrl, Alt) или их сочетаниями (Ctrl+Alt+Del, Ctrl+Tab и т.д.).

Команды разных версий операционной системы Windows реализуются в виде пиктограмм и элементов, расположенных в разных частях экрана. Другие выбираются из меню и списков, которые раскрываются после щелчков мышью на соответствующих управляющих элементах (рис. 3.16).

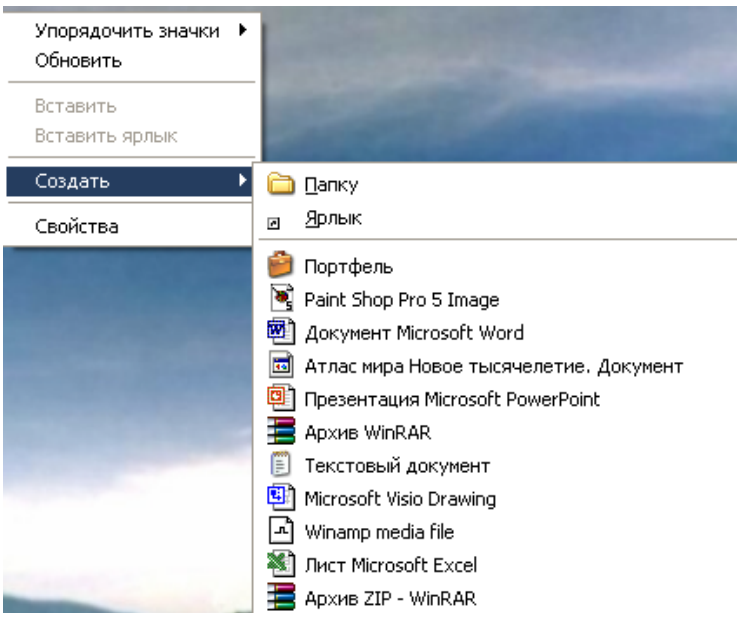


Рис. 3.16. Команды из списка команды ***Создать*** контекстного меню рабочего стола ОС Windows

Команды приложений ОС Windows реализуются в виде кнопок, команд меню и многими другими графическими объектами (комбобоксы, чекбоксы, радиокнопки и др.), которые располагаются на экране ПК (рис. 3.17, 3.18).

Рис. 3.17. Кнопки стандартной панели инструментов приложений MS Office

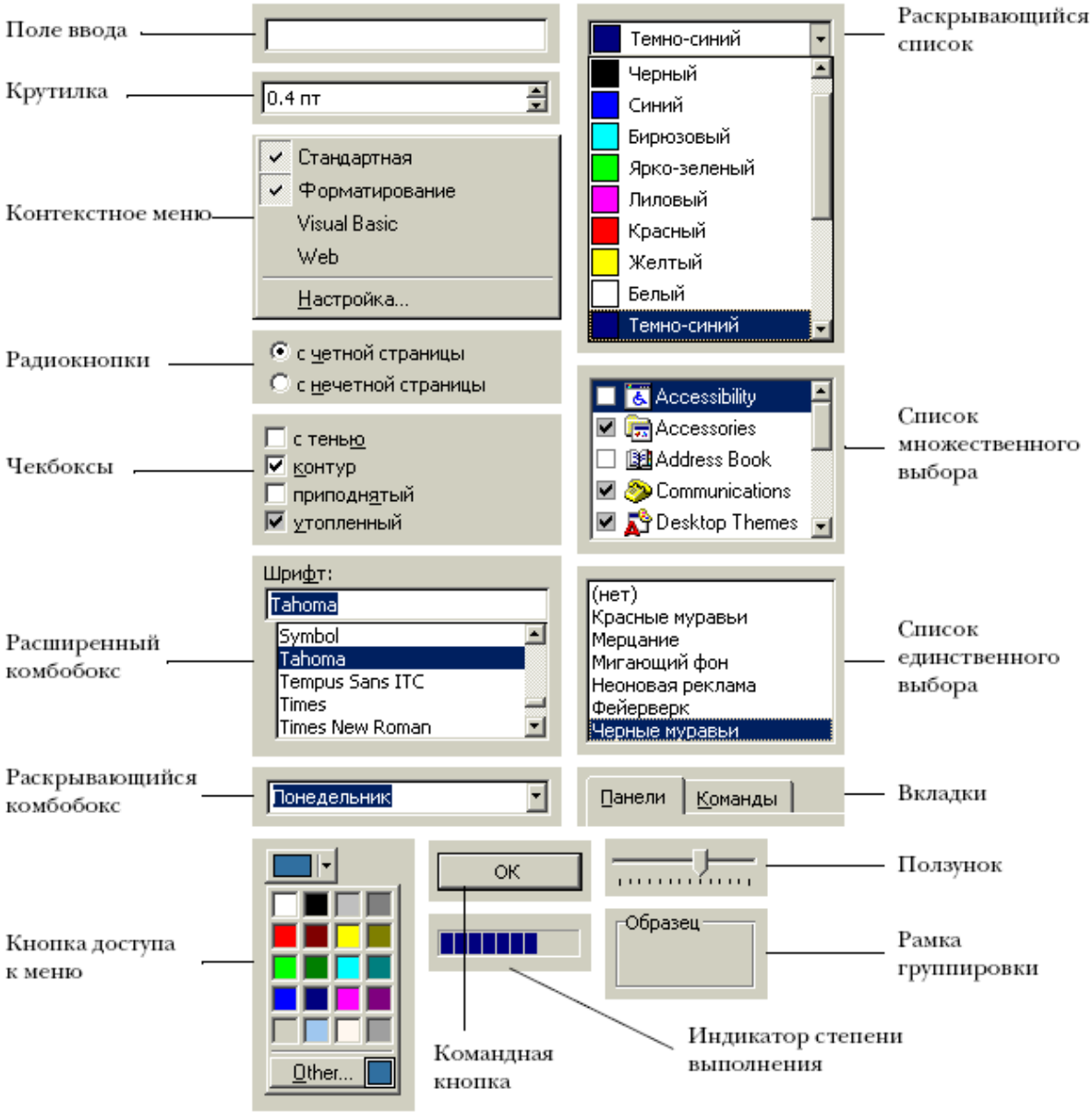


Рис. 3.18. Вид разнообразных ***команд*** работы с операционными системами и приложениями, которые реализованы в стандарте графического интерфейса пользователя (GUI)

Таким образом, ***набор команд***, который разработан для работы с приложениями (программами) под управлением ОС, а также ***правила их использования***, называются в комплексе ***интерфейсом пользователя*** и может меняться от программы к программе. Сами команды реализуются разными средствами, но по своей сути являются ***макрокомандами***, которые содержат

большое количество других элементарных команд, а также обращений к подпрограммам и прерываниям. Чем интерфейс является более развитым с точки зрения его удобства для пользователя, тем больше он считается «дружественным».

Все действия и операции, которые доступны в виде макрокоманд (то есть больших команд), функций приложений и библиотек программ, пользователь может применять путём простых щелчков мышью на кнопках и выбора команд из выпадающих списков и некоторых других действий, которые предусмотрены интерфейсом ОС. В таблице 3.1 представлены некоторые виды команд, которые используются и функционируют на разных уровнях иерархии управления компонентами и объектами при разработке, использовании и работе элементов компьютерных устройств.

Таблица 3.1.

Команды в структуре ПК

№	Наименование команды	Уровень (логич./физич.)	Устройства, которые управляются командами	Уровень представления
1	Импульс	Физический	Транзисторы, диоды	Электрический
2	Значение (0,1)	Логический	Вентили	Логический
3	Значение (0,1)	Логический	Логические диаграммы	Логический
4	Сигнал (реализующий значения 0 и 1)	Логический	Логические схемы триггера и др. устройств	Логический
5	Импульс (реализующий значения 0 и 1)	Физический	Физические схемы триггера и др. устройств	Электрический
6	Команда процессора	Физический	Элементы устройств компьютера	Электрический
7	Команда языка ассемблера	Логический	Элементы устройств и устройства компьютера	Логический Текстовый Данные
8	Команда алгоритмического языка (ЯВУ) (оператор, инструкция)	Логический	Элементы устройств и устройства компьютера	Логический Текстовый Данные
9	Команда ЯВУ с графическим интерфейсом* (меню, кнопка и др.).	Логический	Элементы устройств и устройства компьютера, другие компьютеры	Логический Текстовый Графический Данные
10	Команда управления операционной системой (текстовая строка, сочетание клавиш, меню, кнопки, списки и др.)	Логический	Устройства компьютера	Логический Текстовый Графический Данные

Примечание (*): Языки высокого уровня (ЯВУ) с графическим интерфейсом являются, как правило, интегрированными средами разработки (ИСР) (IDE-Integrated Development Environment) либо средствами быстрой разработки приложений (RAD-Rapid Application Development). В последнее время некоторые из этих средств стали называть языками: Delphi, C#, Excel и др. Кроме того, их ещё относят к языкам так называемого четвёртого поколения – 4GL-языки.

Всё то, что представляется ещё нерешённым или не реализованным в функциях существующих операционных систем и приложений, либо требует дополнительной автоматизации или специальной настройки, как правило, программируется непосредственно пользователем или профессиональными программистами. Таким образом, главный вектор приложения сил последних направляется в программирование всего того, что ещё не реализовано. И несмотря на колоссальный рост количества разнообразных приложений (по некоторым данным только к 1990 году для компьютеров Apple было разработано более 13 тыс. прикладных программ), необходимость в применении пользователями языков программирования не только не уменьшилась, но продолжает постоянно увеличиваться. Кстати, одним из важнейших приложений программирования является автоформализация знаний специалистов разных прикладных областей.

Вопросы.

1. Что такое управление?
2. Для чего необходимо управлять компьютером?
3. Приведите примеры представления физических уровней систем?
4. Приведите примеры представлений логических уровней систем?
5. Какие задания выполняют логические имена устройств?
6. Приведите примеры физических команд?
7. Что играет роль команд в языках программирования?
8. Какие команды операционной системы Windows Вы знаете?
9. Какими устройствами компьютера и с помощью каких команд можно управлять?

Интерфейс – именованное множество операций, описывающих поведение элемента.

Компьютер – это интерфейс доступа к цифровым данным.

Язык UML.

4. КОНЦЕПЦИИ ИНТЕРФЕЙСА

4.1.Задачи и функции интерфейса

Рис. 4.1. Скоро с компьютером можно будет разговаривать

Усаживаясь за ПК, как правило, человек не задумывается о том, насколько сложный процесс осуществляется при трансформации информации в цепочке "мысль – результат" (рис. 5.1, 5.2). Ведь для того, чтобы организовать процесс решения задачи по обработке информации, приходится предварительно сформулировать "задание машине" в терминах и действиях понятных самой машине, то есть программно. И если интерфейс между мыслительными процессами человека и его двигательными функциями природой организован изначально, то с компьютером дело обстоит значительно сложнее, так как он состоит из комплекса разнородных и достаточно сложных устройств. Поэтому, приходится на границе взаимодействия не только человека с компьютером, но и на границах взаимодействия всех его компонентов решать проблемы согласования их работы на уровне соответствующих интерфейсов, как программных, так и аппаратных.

Рис. 4.2. Переходные состояния информации при её обработке на ПК

Под границами взаимодействия, естественно, понимаются участки пространства, на которых происходит передача данных от одной сущности к другой (например, разъём является участком, где спрягаются входной штеккер внешнего устройства и гнездо выхода устройства, к которому подключается последнее).

Другими словами, основная задача интерфейса – обеспечение ***двусторонней*** передачи данных, а значит и информации.

Само по себе, сложное и многофункциональное англоязычное (!) понятие "***интерфейс***" достаточно многозначно. Энциклопедия предлагает следующие трактовки и смысловые оттенки этого термина (рис. 4.3).

ИНТЕРФЕЙС (как существительное, обозначает)

1) область, где встречаются и взаимодействуют две системы;

2) пользовательский интерфейс, состоящий из набора кнопок, списков команд (меню), команд операционной системы, форматов графических дисплеев и других устройств, поддерживаемых компьютером или программой. Графический интерфейс пользователя (ГИП)(GUI-Graphic User Interface) обеспечивает пользующемуся им более или менее "образно ориентированный" (picture-oriented) способ взаимодействовать с новыми техническими средствами и технологиями. GUI обычно более удобен или дружественен при работе с компьютерными системами.

3) программный интерфейс состоит из набора утверждений, функций, опций и других средств, выражающих программные инструкции и данные, предусмотренные программой или языком программирования для использования их программистом.

4) физический или логический интерфейс поддерживает соединение любого устройства с любым соединением или другим устройством.

5) как глагол (соединять, связывать), означает взаимодействие с другим пользователем или объектом. По отношению к аппаратным средствам интерфейс означает создание физического соединения, позволяющего двум частям оборудования сообщаться или работать вместе эффективно.

Рис. 4.3. Многозначность понятия ***интерфейс***

Рис. 4.4. Интерфейс дороги – дорожные знаки

Чтобы лучшее понять содержание и некоторые отдельные значения этого термина рассмотрим интерфейс автомобильной дороги (рис. 4.4). Разработанный совместными усилиями специалистов разных стран мира он понятен всем автомобилистам потому, что система дорожных знаков изучается во время прохождения соответствующих курсов. Глядя на тот или иной дорожный знак, водитель понимает, что ожидает его дальше на дороге и какие действия необходимо выполнить во время движения по разным её участкам. Более того, проезжая на автомобиле по автостраде в любой

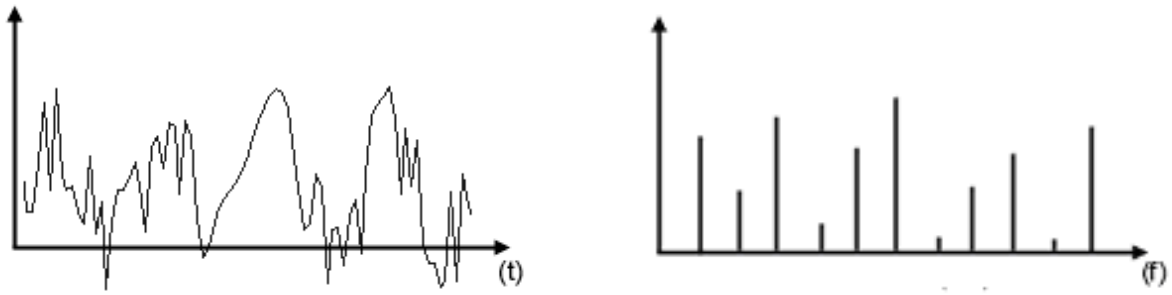
стране, ему нет необходимости знать язык этой страны, так как общие для водителей всех стран дорожные знаки и являются языком взаимодействия с системой под названием "дорога".

Природа компьютера значительно сложнее многих других систем и объектов, с которыми человеку приходится взаимодействовать. К тому же она является чрезвычайно динамичной во многих отношениях.

Во-первых, конфигурация и состав устройств ПК очень изменчивы. Сюда же следует отнести разнообразность и многофункциональнисть программных компонентов, которые используются.

Во-вторых, природа, масштабы изменений в форме существования и размерах объёмов ***данных*** и ***информации***, которыми обмениваются пользователь и компьютер, иногда трудно поддаются осмыслению и формализации (на данном этапе развития науки).

В самом общем приближении, можно сказать, что ***данными для компьютера являются зарегистрированные объективные сигналы или факты.*** И те, и другие вводятся в его память в виде последовательностей цифровых дискретных (а не аналоговых!) сигналов (рис. 4.5), или текстовых символов с какого-либо их носителя или другого устройства (клавиатуры, жёсткого или гибкого диска, CD-ROM, базы данных, цифровых или аналоговых датчиков и т.д.).



а) **б)**

Рис. 4.5. Пример аналогового сигнала (а) и
его дискретного (оцифрованного) образа (б)

Примерами сигналов могут служить данные, которые передаются спутниками с планет Солнечной системы, куда они были посланы. Сюда же можно отнести и фотографии в цифровой форме, температурные режимы на поверхности планет, которые исследуются и многое другое.

Примерами зарегистрированных фактов могут служить результаты переписи населения или сведения о геологических характеристиках Земли, определённые в конкретный промежуток времени и введенные в компьютер. В свою очередь, каждая дискретная величина (дискретный знак, цифра или число) представляются в схемах и устройствах компьютера (в том числе и запоминающих) также в виде дискретных двоичных сигналов 0 и 1 (рис. 4.6).

Информацией же, является ***субъективная динамическая составляющая*** результата обработки данных.

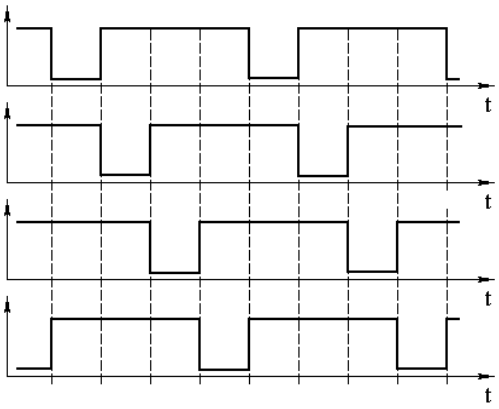


Рис. 4.6. Представление дискретных двоичных сигналов 0 и 1, которое реализуется в виде синхронных импульсов (0-нет импульса, 1-есть импульс)

Вообще говоря, ***информацией называются сведения, которые неизвестны до их получения и являются объектом хранения, передачи и обработки***. То есть, информацией является ***динамический результат*** применения субъективных методов обработки к объективным данным, который ***осознан получателем***. Если результата (осознания) нет, это значит, что информация не воспринята. Например, чтоб узнать, как работает то или иное сложное устройство, необходимо открыть инструкцию и начать её ***читать*** (динамическая составляющая процесса получения информации). Однако, если инструкция напечатана на китайском языке, никакой информации человеку получить не удастся, если она не владеет этим языком (рис.4.7).

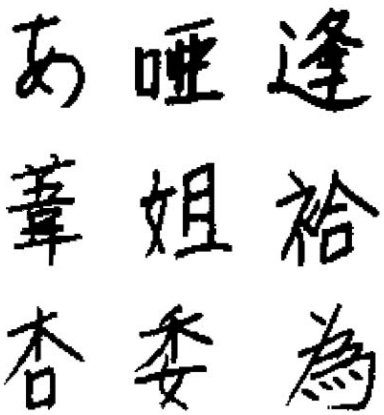


Рис. 4.7. Китайские иероглифы (сложны в переводе и произношении).

Данные всегда объективны, даже если и несут в себе неправдивую информацию. В какой бы форме они не были бы выражены, это всегда зарегистрированные сигналы, то есть объекты материальной природы. А вот информация может быть как объективной, так и субъективной, как истинной, так и ложной. Она не является объектом материальной природы. Свойства информации зависят не только от содержания данных, но и от свойств методов, с помощью которых она получена, и более всего от согласования представления информации с возможностями её получателя (реципиента).

При этом не следует забывать, что практически вся информация, которая сейчас накапливается на современных носителях информации запоминается, сохраняется, а потом используется и обрабатывается в виде нулей и единиц. Поэтому, компьютер является интерфейсом доступа к использованию всего этого множества данных, которые продолжают постоянно накапливаться! (рис. 4.8).

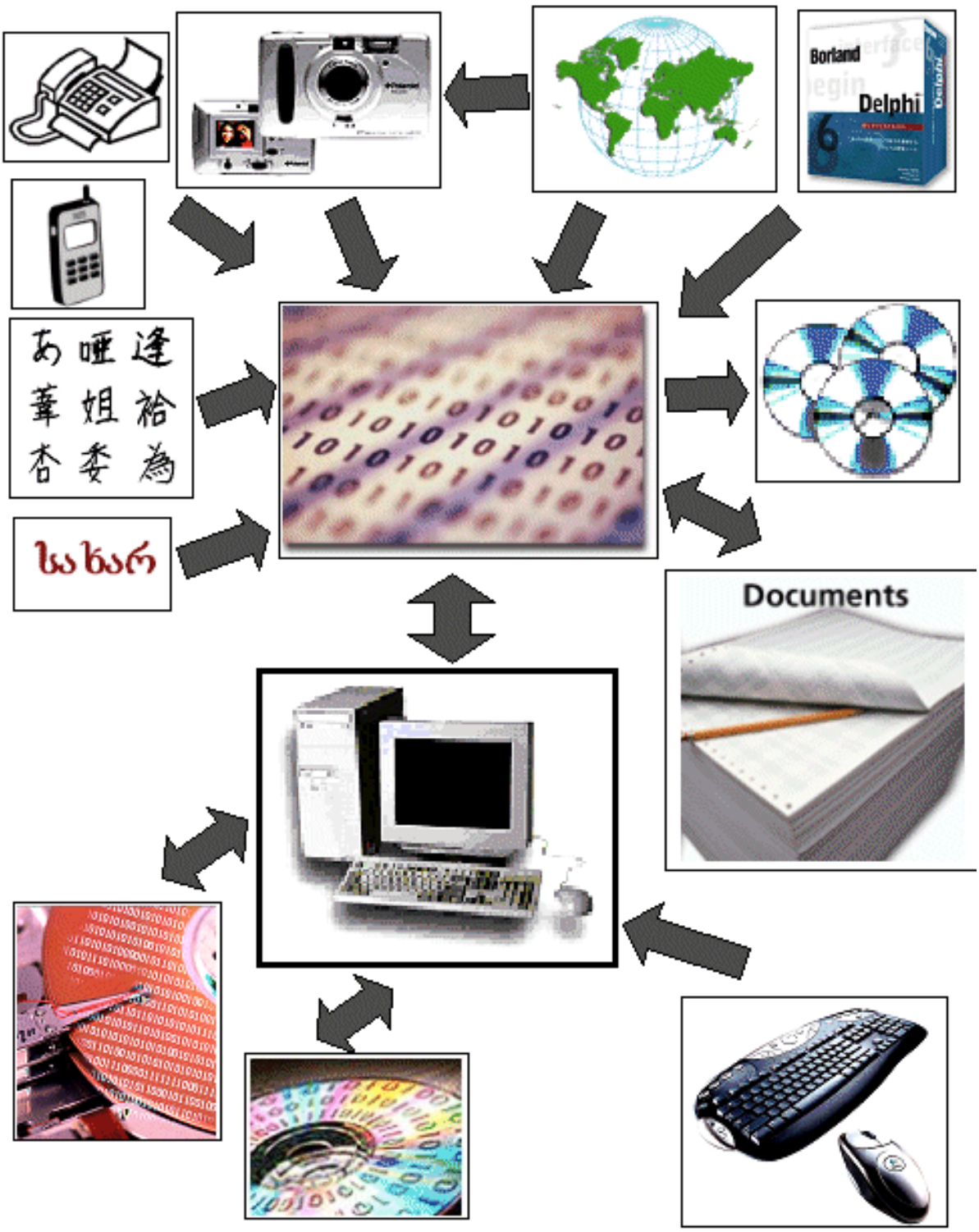


Рис. 4.8. Двоичные цифры *0* и *1* – универсальный хранитель и передатчик информации и данных

Какие бы данные не вводились в компьютер: текст на любом языке, звук, изображение, фотографии, даннные о разнообразных объектах Земли– все это преобразуется в процессе ввода в компьютер только в ***нули*** *(0)* ***и единицы*** *(1)*. В таком же виде всё это находится на гибких и жёстких магнитных дисках, лазерных компакт- дисках и обрабатывается в устройствах ПК программами, которые представлены тоже в виде цифр 0 и 1.

Объём информации, необходимый для нормального функционирования современного общества растёт приблизительно пропорционально квадрату (!!!) уровня развития промышленного потенциала ведущих стран мира. Доля рабочей силы, которая вовлечена в процессы накопления и обработки информации уже давно и многократно превышает долю рабочей силы, занятой непосредственно в материальном производстве.

В данном контексте, ***процесс взаимодействия пользователя с компьютером***, в самом общем виде можно представить следующим образом (рис.4.9). Пользователь вводит требуемую ему информацию (для записи на диск, обработки и т.д.), а также и для управления компьютером, чередуя ***данные*** и ***команды***. Компьютер, следуя указаниям, выполняет задания, отображая для пользователя на экране дисплея результаты работы и разнообразные сообщения (текущее состояние, сообщения об ошибках, подсказки и др.)

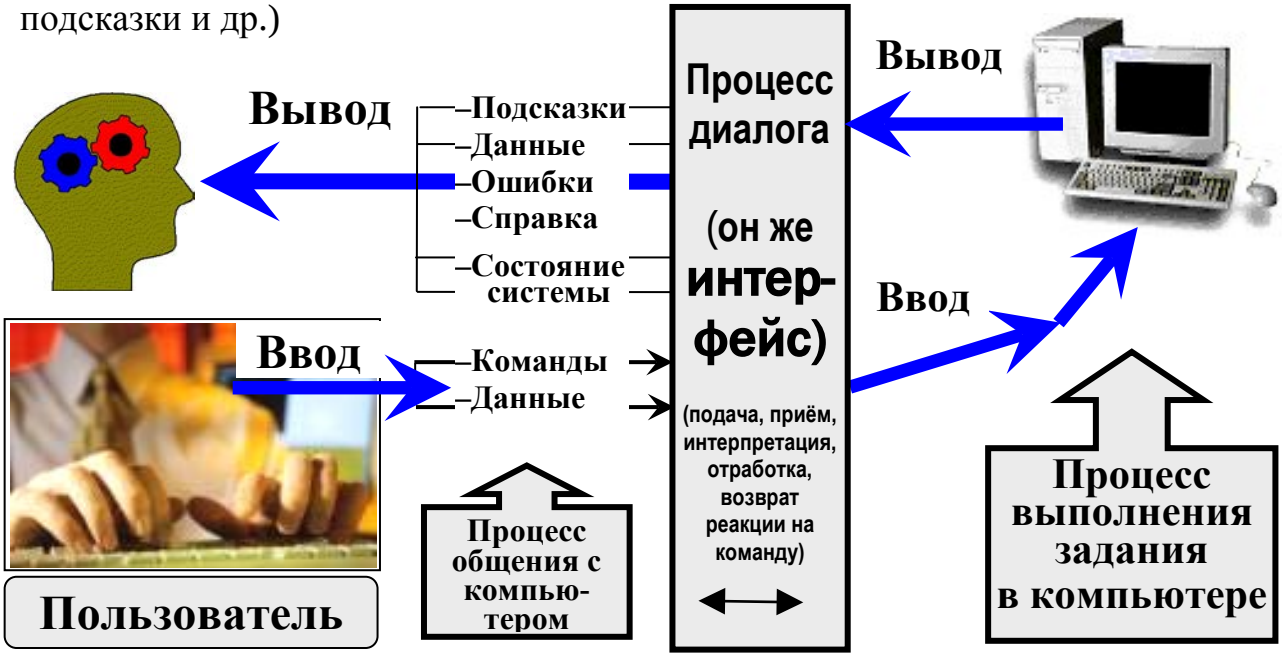


Рис. 4.9. Процесс диалога пользователя с компьютером

Что главное в этом процессе? Чтобы ответить на этот вопрос, необходимо чётко представлять о каком взаимодействии идёт речь и каковы его содержательные компоненты. Очевидно, что в случае адекватной реакции со стороны другой системы, необходимо присутствие следующих концептуальных составляющих:

❶ наличие информационных потоков с обеих сторон;
❷ соглашение о протоколе[1] обмена данными между ними;
❸ соблюдение спецификаций[2] протокола обмена данными;
❹ обеспечение интерфейса взаимодействия в рамках ввода-вывода[3] со стороны обеих систем.

[1] Протокол обмена данными – набор правил и соглашений, определяющих форматы данных и процедуры передачи, для обмена информацией между взаимодействующими системами (процессами).
[2] Спецификация – полное описание требований.
[3] Ввод-вывод – обмен данными под управлением компьютера.

4.2. Принципы формирования интерфейса пользователя

Вообще говоря, UI (User Interface – интерфейс пользователя) в компьютерных системах состоит из экрана, мыши, клавиатуры и поддерживается операционной системой. В современных ПК интерфейс пользователя может включать динамики, микрофоны, а также, в случае необходимости применения специализированных устройств, из ручек их управления, датчиков, фиксирующих направление взгляда либо чувствительных к прикосновению мест на экране. Такие экраны часто именуются сенсорными и впервые были применены в копировальных аппаратах фирмы Xerox, которые практически не имели механических (тумблерных) или электрических (кнопочных) элементов управления. Пользователь видел на жидкокристаллическом экране изображения кнопок с расшифровкой их функционального назначения (рис. 4.10).

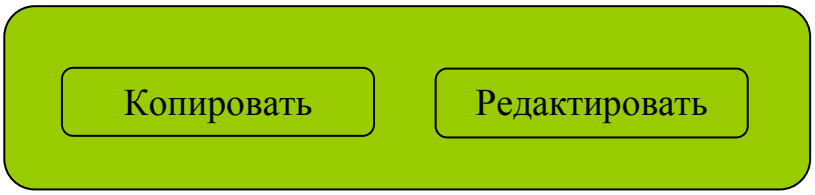


Рис. 4.10 Кнопки жидкокристаллического экрана, прикосновение к которым, раскрывает меню (кнопки) следующих уровней

Нажатие пальцем в месте расположения требуемой кнопки вызывало раскрытие нового, следующего экрана, с изображением кнопок более низкого уровня и так далее. Вследствие применения такого подхода, аппараты Xerox значительно превосходили по простоте управления аппараты многих других фирм, панели управления которых, были густо усеяны множеством кнопок, клавиш, тумблеров, индикаторов работоспособности и т.д (а это всё, по своей сути, команды пользователя на выполнение определённых действий и реакция системы на них!). В некоторых из подобных копировальных устройств, число таких регуляторов доходило до восьмидесяти элементов. С учётом наличия под каждым из них текстовых подписей, расшифровывающих их функциональное назначение, освоение интерфейса таких аппаратов представляло собой непростую задачу и требовало много времени.

В начале своего развития операционные системы обеспечивали как пакетный, так и диалоговый режим работы с пользователем. В пакетном режиме операционная система автоматически исполняет заданную последовательность команд, записываемую предварительно в так называемые командные "бэт-файлы" с расширением .BAT (рис. 4.11).

Суть же диалогового или интерактивного[4] режима заключается в том, что операционная система находится в состоянии ожидания команд пользователя и,

[4] Інтерактивний – режим, у котрому користувач задає програмі команди і уводить дані у період її роботи. Такий режим як правило припускає обмін текстовими командами (запитами) і відповідями (запрошеннями).

плучив её, начинает выполнение требуемых действий, а выполнив, возвращает отклик и ожидает поступления очередной команды. Способность операционной системы прервать текущую работу и отреагировать на события, вызванные пользователем с помощью управляющих устройств, воспринимается нами как диалоговый (интерактивный) режим работы.

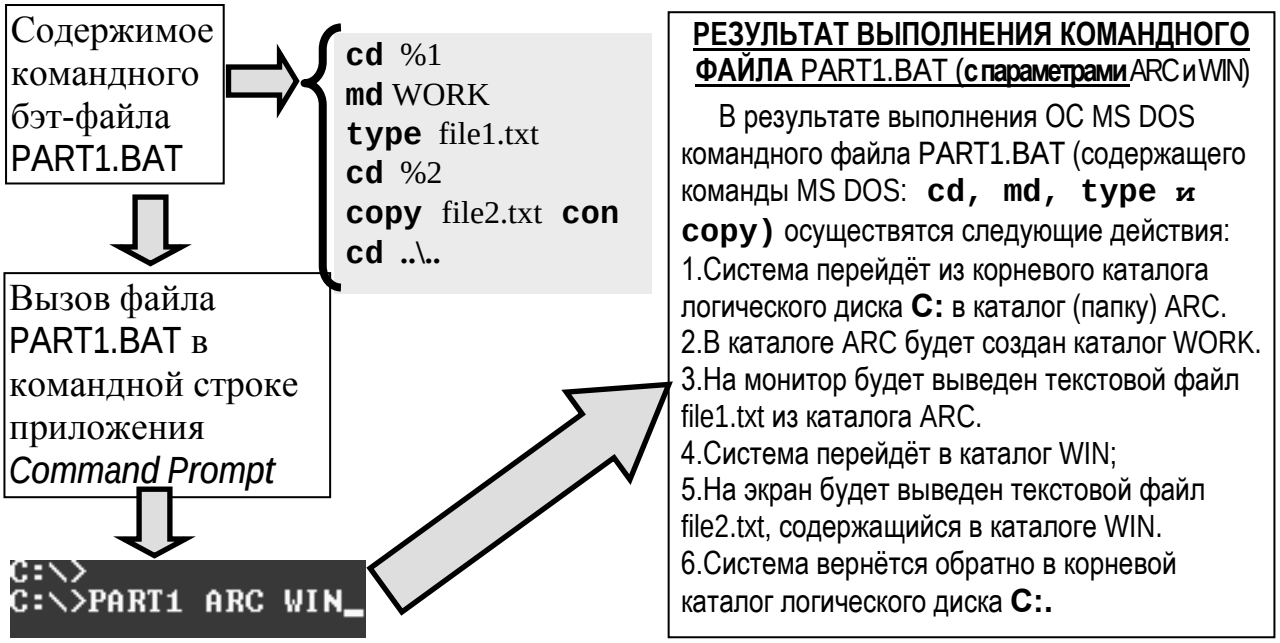


Рис. 4.11 Процесс управления компьютером с помощью команд, объединённых в командном файле, использующем подстановочные параметры ARC и WIN

Современные операционные системы могут одновременно обеспечивать и интерфейс командной строки, и графический интерфейс пользователя. В первом случае устройством управления является клавиатура, с помощью которой вводятся команды. Эти ***команды*** (в том числе их имена (названия) и выполняемые действия которых) пользователь должен ***предварительно выучить и научиться применять*** (!!!), вводятся в поле командной строки, где их можно также и редактировать. Отправка команды на исполнение ОС осуществляется после нажатия клавиши Enter. Для компьютеров платформы IBM PC интерфейс командной строки обеспечивается семейством операционных систем под общим названием MS-DOS (версий от 1.0 до 6.2).

За последние годы методы организации интерфейса в системе "человек–компьютер" получили значительное развитие и приобрели определённую логическую завершённость. Тем не менее, в операционной системе ПК Windows сохранён изначальный интерфейс командной строки, окно которой открывается после выбора из меню[5] кнопки ***Пуск*** команды ***Командная строка (Command prompt)*** (рис. 4.12).

[5] Меню – список свойств объектов и применяемых к ним команд (т.е. способов преобразования) в операционной системе и в работающих под её управлением приложениях.

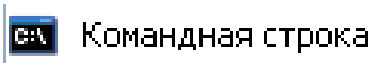


Рис. 4.12. Команда ***Командная строка*** из меню кнопки ***Пуск***

В открывшемся окне могут выполняться все команды MS-DOS (рис. 4.13).

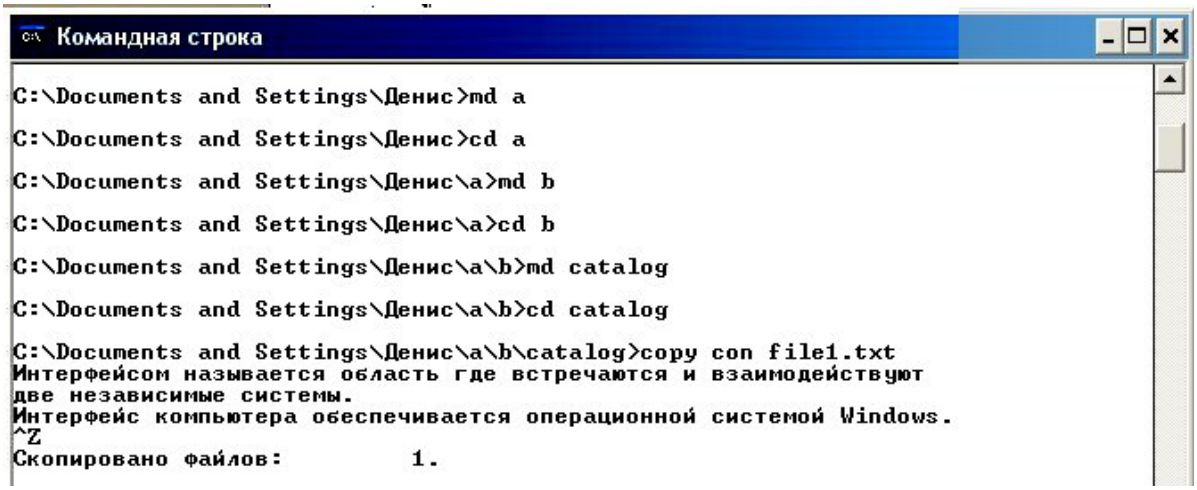


Рис. 4.13. Создание каталогов и текстовых файлов в ОС Windows XP с помощью команд MS-DOS

Графический интерфейс пользователя обеспечивается операционными системами Windows xx (разных версий), в которых в качестве органа управления кроме клавиатуры может использоваться мышь или адекватное устройство позиционирования. Доступ к устройствам и их свойствам может быть осуществлён из окна ***Панель управления*** (Control panel) (рис. 4.14)

Рис. 4.14. Интерфейс доступа к устройствам компьютера и их свойствам в ОС Windows

Таким образом, работа с графической операционной системой основана на взаимодействии активных и пассивных экранных элементов управления. В качестве активного элемента управления выступает указатель мыши (курсор) – графический объект, перемещение которого на экране синхронизировано с перемещением мыши. В качестве пассивных элементов управления приложениями выступают следующие графические элементы управления: окна, экранные кнопки, значки, переключатели, флажки, раскрывающиеся списки, строки меню и многие другие. Программно, доступ к элементам интерфейса осуществляется на уровне баблиотеки компонетов Windows API.

Для создания такого интуитивно понятного и комфортного интерфейса сначала необходима ***метафора***[6], отталкиваясь от которой можно начинать реализацию требуемых от него функций. Ведь для пользователя, что называется, "с улицы", обычно непосвященного во все тонкости аппаратных и программных средств компьютера, необходимо все "болты и гайки" операционной системы упрятать внутрь удобной и привлекательной упаковки, которой и является Windows !. В основу WIMP-интерфейсов легли три следующие метафоры: "конкретный объект" (WIMP), "что видишь – то и получишь" (WYSIWYG) и "рабочий стол" (Desktop) (рис. 4.15).

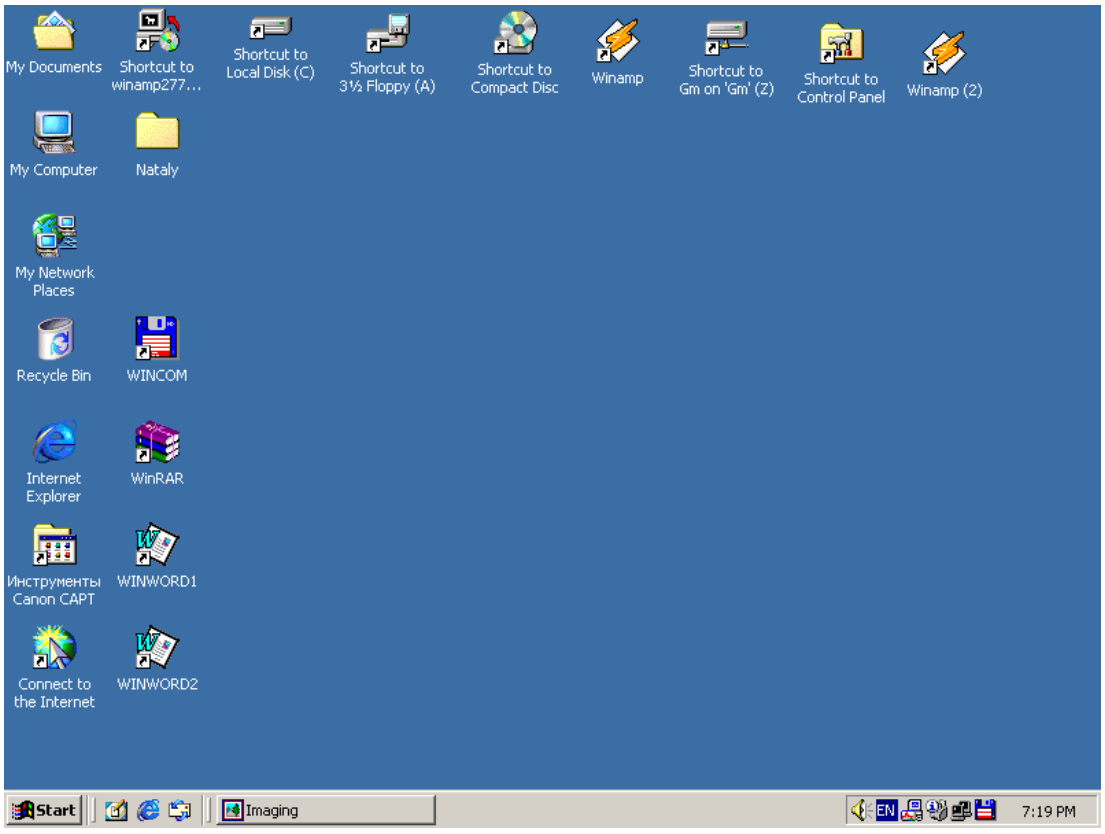


Рис. 4.15. Рабочий стол (Desktop) операционной системы Windows 2000 с пиктограммами папок, внешних и внутренних устройств ПК

[6] Метафора – (от греч. metaphora – перенос) – троп или механизм речи, состоящий в употреблении слова, обозначающего некоторый класс предметов, явлений и т.п., для охарактеризования или наименования объекта, входящего в другой класс, либо наименования другого класса объектов, аналогичного данному, в каком-либо смысле. В расширительном смысле термин "метафора" применяется к любым видам употребления слов в непрямом значении.

Аббревиатура *WIMP* представляет следующие понятия:

W (windows – окна) – информация представляется пользователю на экране дисплея (монитора) в виде одного или нескольких окон;

I (icons – иконки (пиктограммы)[7]) – объекты, с которыми система и пользователь имеют дело и представляются пиктограммно[8] в виде иконок;

M (mouse – мышь) – выборка объектов для работы производится с помощью манипулятора типа "мышь";

P (pop-up – всплывать) – означает раскрытие контекстных[9] меню, которые автоматически всплывают на экране при щелчке правой кнопки мыши в любой момент работы с системой в любом месте экрана (рис. 4.16).

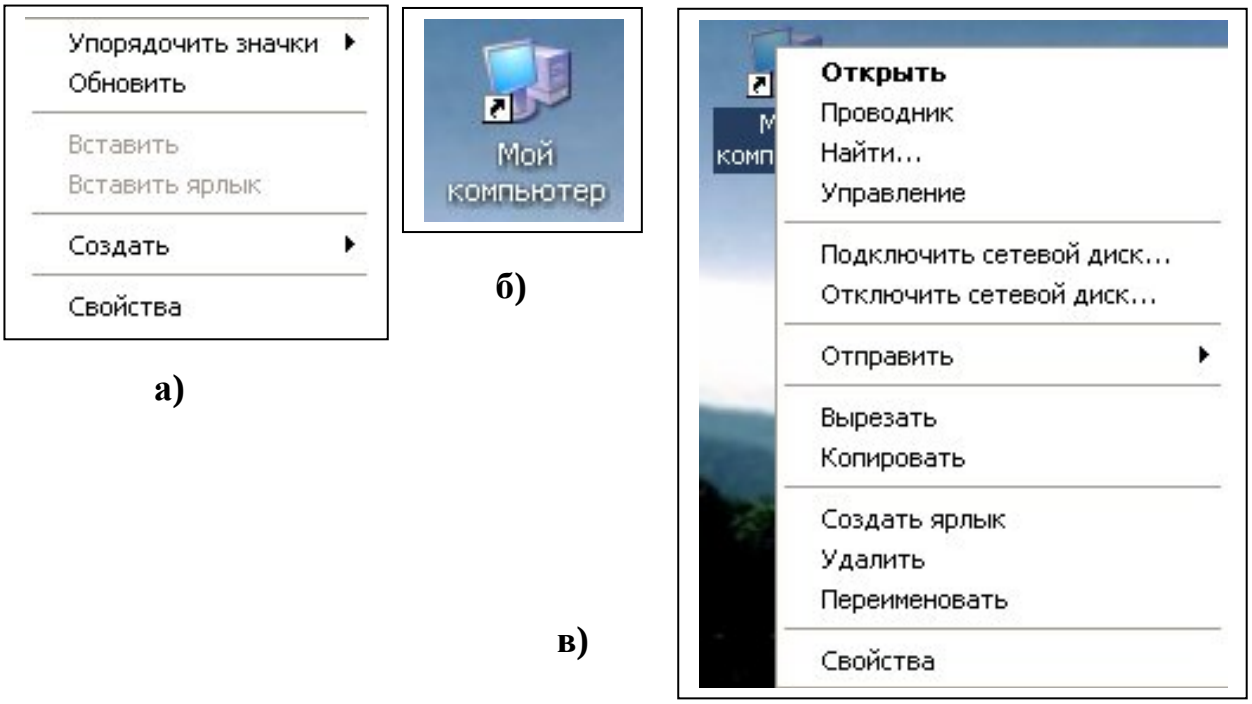


Рис. 4.16. Контекстное меню Рабочего стола (а).
Объект ***Мой компьютер*** (б) и соответствующее ему контекстное меню (в)

Метафора "конкретный объект" введена для того, чтобы пользователю не требовалось изучать ни семантику применяемой компьютерной системы (имена команд, компоненты ПК и их связи с объектами решаемых задач), ни синтаксис команд компьютерной системы, а пользовался только графическими представлениями физических объектов, то есть их пикторамами (иконками).

На рабочем столе метафорой файлов являются пачки бумаг или папок, а каталогов – ящики в картотеке, которые выдвигаются и т.д. Канцелярские

[7] Иконка – неотъемлемый атрибут любой кнопки или файла в операционной системе Windows, позволяющий легко распознать тип объекта либо конкретный объект. Более точно тип файла определяется по его расширению (.DOC, .EXE и т.д.). Значки могут храниться в отдельных файлах с расширением .ICO, в программных файлах (.EXE), в динамически формируемых библиотеках (.DLL) и т.д.

[8] Пиктограмма – графическое представление (изображение) объекта на экране компьютера (аналог – иконка). Пиктограммы имеют: компьютер (значок «Мой компьютер»), файлы, логические диски, принтеры и т.д.

[9] Контекстное меню – меню, открываемое операционной системой или приложением в результате щелчка правой кнопкой мыши по некоторому объекту. Такие меню содержат команды, применяемые в контексте операционной обстановки в данный момент работы с данным объектом.

операции предполагают наличие физических действий над этими объектами. Так, для запоминания файла, человек собирает пачку деловых бумаг подходит к шкафу для хранения, открывает ящик и кладёт пачку внутрь. При переименовании файла человек сотрёт старое имя и напишет новое, а запись файла в новый каталог может быть выполнена перетаскиванием его с помощью мыши на иконку требуемого каталога и т.д.

Метафора WYSIWYG (What You See Is What You Get-"что видишь – то и получишь") описывает такие интерфейсы, в которых фактический эффект любого действия немедленно отображается на дисплее. Это означает, что экран должен имитировать средства печати, и если пользователь хочет напечатать часть текста курсивом, он и на экране должен быть набран курсивом. Если файл уничтожается, пользователь видит, что файл исчезает из изображенного на экране списка файлов, т.е. интерфейс эффективно обеспечивает информацию о статусе объекта, подтверждая, что действие было выполнено.

Метафора "рабочий стол" позволяет пользователю иметь перед собой все необходимые для работы документы и легко переключаться с одного на другой, т.е. "тасовать бумаги на столе" и совершать с ними все необходимые действия. На таком столе легко уживаются средства для работы с текстами, электронные таблицы для финансового планирования и различные вспомогательные средства: калькулятор, часы, дневник, записная книжка и т.д.

Интерфейсом доступа к объектам и компонентам компьютера служат их ярлыки[10], представляемые операционной системой соответствующими иконками. Располагаемые в необходимых местах на рабочем столе, в каталогах и других доступных местах они позволяют пользователю оперативно связывать разнородные компоненты компьютера. К примеру, перемещение ярлыка документа текстового редактора Word (расположенного одновременно и в некотором каталоге и на Рабочем столе Windows (тоже модели каталога!)) на иконку принтера приведёт к печати файла (рис. 4.17). Естественно, если принтер подключён к данному ПК!

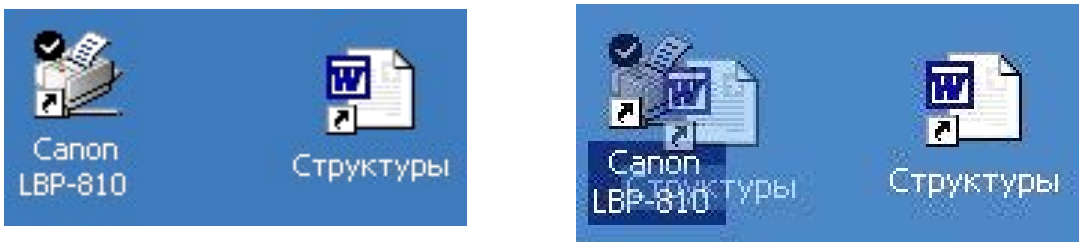


Рис. 4.17. Перетаскивание иконки документа на иконку принтера с целью печати документа

Перетаскивание иконки того же документа на иконку флоппи-дисковода осуществит его запись на дискету (если она вставлена в щель дисковода!). Одно простое движение мыши инициализирует в недрах ОС последовательность

[10] Ярлык – файл, содержащий указатель (ссылку) на некоторый объект в дереве ресурсов – другой файл, папку, принтер и т.д. Обеспечивает непосредственный доступ к объекту из другой папки, в частности с рабочего стола. Имеет расширение .LNK и распознаётся по загнутой стрелке в левом нижнем углу его значка. Роль значков выполняют также PIF и URL файлы.

программных действий по считыванию файла с жёсткого диска в память, а затем перепись его из памяти на гибкий диск.

Интерфейсом доступа пользователя (и ОС) к информации, размещённой в файлах осуществляется на уровне расширений их имён. При инсталляции очередного программного продукта, расширения обрабатываемых им файлов (.DOC, .XLS, .EXE, .MP3 и т.д.) заносятся в системный реестр ОС и им в соответствие ставятся породившие их приложения (продукты): .DOC – приложение MS WORD, .XLS – приложение MS Excel и т.д. В каталогах, где они размещаются, таким файлам приформировываются соответствующие значки (рис. 4.18).

Рис. 4.18. Значки файлов приложений MS Excel, MS Visio и MS Word, отображаемые в каталогах (папках) Windows XP

После двойного щелчка мышью по значку (или имени) требуемого файла операционная система, в соответствии с записью в реестре, вызывает функционально ответственное за работу с данным файлом приложение. Оно открывает файл и отображает его содержимое в поддерживаемом им формате: документ, таблицу, музыку, видеофильм и др. В реестре ОС указываются также программы, поддерживающие несколько форматов файлов данных (а всего в мире к 2004 г. насчитывалось более 5 000 разных форматов файлов данных). В частности архиватор файлов RAR имеет возможность раскрывать файлы, заархивированные в формате ZIP. Приложение MS Word раскрывает документы с расширениями .txt, .rtf, .doc и некоторые другие.

Если пользователь щёлкает мышью по файлу, расширение которого неизвестно операционной системе, то на экран выводится окно диалога, где нужно самостоятельно подобрать приложение, которое может «справиться» с содержимым неизвестного системе файла (рис. 4.19).

Исходные (текстовые в ASCII-кодах) файлы интегрированной среды разработки Турбо Паскаль в ПК с ОС Windows, где развёрнута среда быстрой разработки Delphi, представляются в окнах со значками исходных текстов этой среды (Delphi Source File), а исходные тексты Delphi – со значками файлов Delphi Project (рис. 4.20).

Программы на языке Турбо Паскаль с некоторыми ограничениями могут открываться и выполняться в окне консольных приложений (Console Application) среды Delphi. Значительно проще выполняются у этом окне аналогичные или более сложные по своей функциональности программы на языке Object Pascal (языке самой среды). Для кодирования программ необходимо выполнить в Delphi последовательность следующих команд: *File|New|Other* и в окне диалога *New Items* выбрать объект *Console Application*.

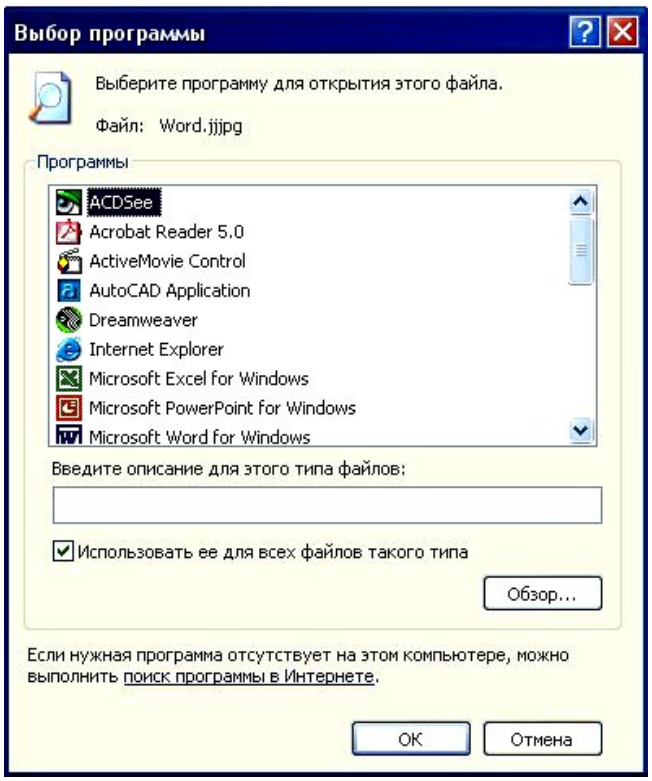


Рис. 4.19. Окно диалога операционной системы Windows XP для поиска «неопознанного» файла с расширением имени .jjjpg

а) **б)** **в)**

Рис. 4.20. Иконки (пиктограммы) файлов в ОС Windows XP:
а) драйвера кириллизатора KEYRUS с расширением .COM,
б) текста Турбо Паскаль KURSOWRT с расширением .PAS
в) текста консольного приложения WWProject2 с расширением .DPR

В то же время, наряду с командным и графическим интерфейсами управления работой устройств компьютера, физический интерфейс между устройствами ПК обеспечивается разнообразными коннекторами[11], штеккерами, гнёздами, сокетами[12] и слотами[13] (рис. 4.21).

[11] Коннектор – соединитель многоконтактный, (штепсельный) разъем. Средство соединения взаимозаменяемых частей (компонентов) компьютера. В частности, шестое поколение процессоров Pentium отличается большим разнообразием разъёмов-конструктивов. Одних только коннекторов имеется 4 типа: сокет 8, слот 1, слот 2 и сокет-370. Следует отметить, что в технологиях фирмы Intel, производителя данных марок процессоров, термины слот и сокет употребляются в более широком смысле. Они обозначают спецификацию электрических, программных и механических интерфейсов.

[12] Сокет – гнездо; (соединительная) панель; розетка (гнездовая часть разъемного соединения).

[13] Слот —1) гнездо (в т.ч. и для платы), вход, розетка; 2) разъем для элементов памяти. Как правило, это название используется для разъемов, куда "вставляются" платы расширения, в том числе модули типа SIMM и DIMM. Разъемы, куда "втыкаются" ножки (чипов либо разъемов "противоположного пола"), называются сокет (socket).

4.3. Конструкции и назначение физических (аппаратных) интерфейсов

Физические (аппаратные) интерфейсы играют большую роль в процессе комплектования базовой основы персонального компьютера. Они определяют: типы стыка, уровни сигналов, импедансы, синхронизацию и много других параметров каналов связи между разнообразными устройствами (рис.4.21).

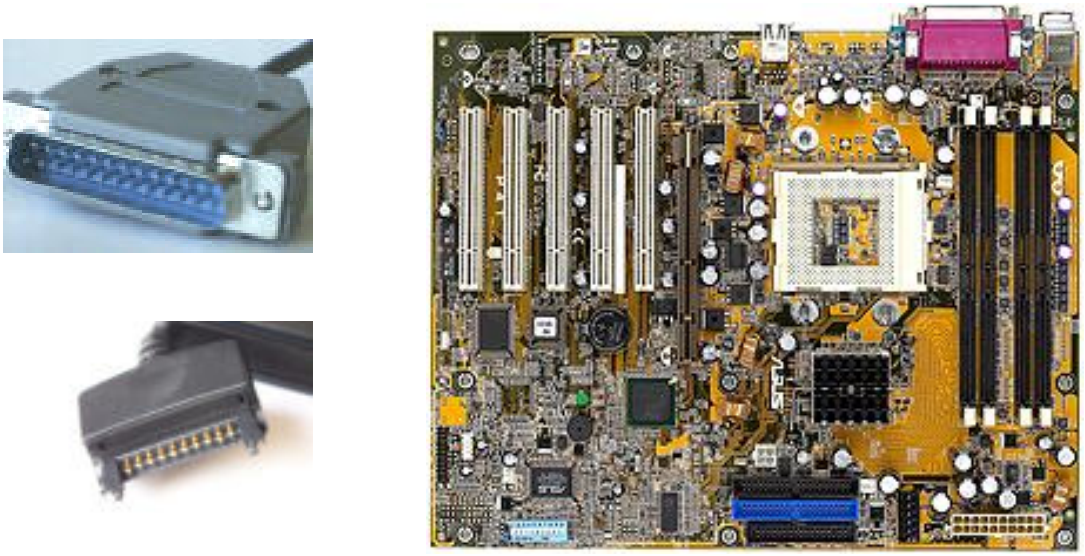


Рис. 4.21. Коннекторы, слоты и сокеты ПК

Интерфейс самого процессора с внешними для него, а также внешними и внутренними для системного блока устройствами организован через порты[14] ввода–вывода. Обычно порт представляет собой точку подключения внешнего устройства компьютера (принтера, сканера и т.д.) к внутренней шине процессора (рис. 4.22). Таким образом, программа или устройство могут посылать данные в порты или получать их из портов для обработки процессором. Для многоканальных устройств порт представляет группу линий ввода-вывода (шин[15]), по которым происходит передача информации между центральным процессором (ЦП) и соответствующим устройством ввода-вывода, как правило, по одному биту на каждую линию. Большинство устройств ввода-вывода информации подключаются к компьютеру с помощью внешней шины. Такие шины принято называть параллельными (LPT1) или последовательными (COM) портами. Обычно разъёмы внешних шин располагаются на задней панели

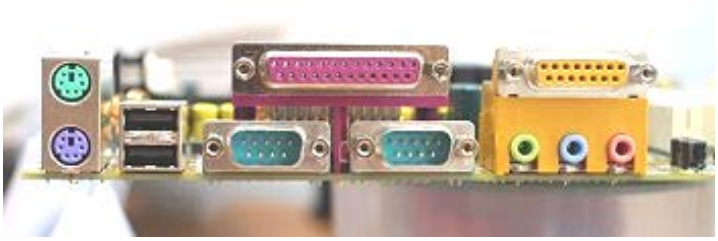


Рис. 4.22 Порты входа-выхода ПК

[14] Порт – точка подключения внешнего устройства к внутренней шине процессора. Таким образом, программа или устройство могут посылать данные в порты или получать их из портов.

[15] Шина – группа линий электрических соединений, обеспечивающих передачу данных и управляющих сигналов между компонентами компьютера.

системного блока компьютера. Большое количество разнообразных внешних устройств и соответствующих им разнотипных интерфейсных механических соединителей, которое многократно выросло за последнее время, затрудняет пользователям их применение и заставляет производителей маркировать соответствующие вилки и розетки идентифицирующими их логотипами (значками или пиктограммами). К примеру, разъёмы, так называемого, порта USB (Universal Serial Bus – Универсальная Последовательная Шина) имеют логотип в виде разветвляющейся структуры (рис. 4.23). Сама система передачи данных USB была разработана в 1995 году фирмами Compaq, DEC, IBM, Intel, Microsoft, NEC и Northern Telecom. Порт USB практически нивелировал существующие ранее существенные различия между последовательным и параллельным портами ПК и имеет следующие характеристики:

❶ Скорость передачи данных 1,5 – 12 Мбит/с.

❷ Максимальная длина шнура - 5 метров.

❸ До 127 внешних устройств, которые подключены одновременно.

❹ Подключение устройств без выключения компьютера (так называемое "горячее" подключение), а также подключение в режиме Plug&Play[16].

Рис. 4.23. Пиктограмма обозначения и разъёмы (вилки и розетки) последовательного порта USB

Другим чрезвычайно распространённым интерфейсом внешних шин является стандарт SCSI (Small Computer System Interface –читается «скази»). Важнейшим преимуществом этого интерфейса является то, что можно подключить к ПК до 8-ми периферийных устройств, имея всего один слот расширения (рис. 4.24).

[16] Технология PLUG & PLAY обеспечивает независимость подключаемых устройств от конкретной операционной системы. Определяет расширения для любой существующей архитектуры IBM-совместимых компьютеров, включая новые BIOS и аппаратные возможности, которые призваны оградить пользователя от проблем с настройкой и конфигурированием системы при подключении новых устройств.

4.4. Интерфейсы в клиент-серверных моделях взаимодействия программ и устройств

При увеличении количества программно-аппаратных средств, которые одновременно используются, как правило, они объединяются соединительными компонентами в сеть.

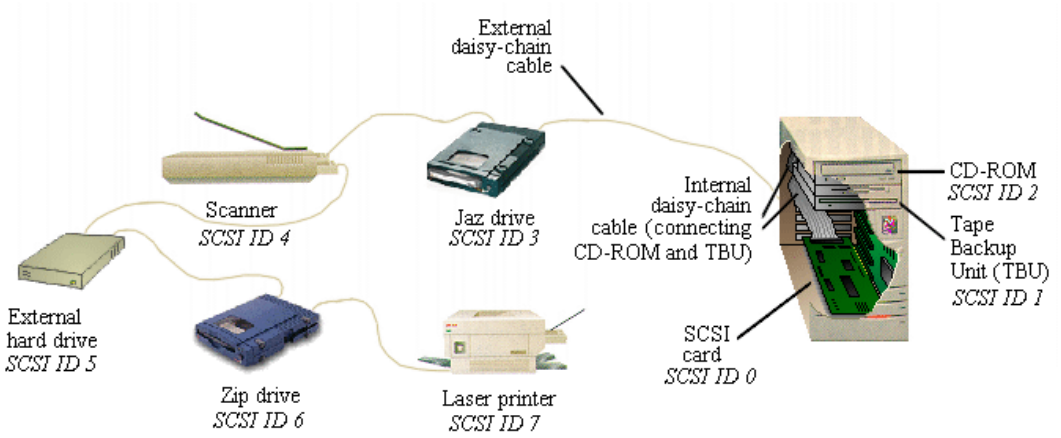


Рис. 4.24. Схема подключения семи устройств через SCSI («скази») порт

Поэтому, компьютеры и программы, которые взаимодействуют и одновременно входят в состав информационной системы или задействованные в сети, не являются равноправными. Некоторые из них владеют ресурсами[17] (к примеру, файловая система, процессор, принтер, база данных и т.д.), другие имеют возможность только обращаться к этим ресурсам (приложения, программы и др.). По такой схеме работают многие компьютерные классы, где большинство компьютеров используют общие ресурсы (программы, файлы, услуги баз данных и др.) одного из них, называемого ***сервером*** (рис. 4.25).

Рис. 4.25 . Компьютерный класс с компьютерною сетью

[17] Ресурс – средство вычислительной системы или компьютера, которое может быть выделено процессу обработки данных (программе пользователя) на определённый момент времени. Основными ресурсами компьютера являются: процессоры, области основной памяти, наборы данных, периферийные (внешние) устройства, программы и т.д.

Компьютер или программу, которая управляет ресурсом, называют ***сервером*** этого ресурса (файл-сервер, сервер бази данных, вычислительный сервер и т.д.). ***Клиент***, то есть компонент, который пользуется услугами ***сервера***, и сам сервер какого нибудь ресурса могут находиться как в рамках одной вычислительной системы (компьютера), так и на разных компьютерах, которые связаны сетью и, таким образом, находятся в одной сети.

В самом общем случае, термин "***клиент-сервер***" означает такую архитектуру программно-аппаратного комплекса, в которой его функциональные части взаимодействуют по схеме "запрос-ответ". Если рассмотреть две взаимодействующие части этого комплекса, то одна из них (клиент) выполняет активную функцию, т. е. инициирует запросы, а другая (сервер) пассивно на них отвечает. По мере развития системы роли могут меняться, например некоторый программный блок будет одновременно выполнять функции сервера по отношению к одному блоку и клиента по отношению к другому.

Наиболее простая форма архитектуры клиент-сервер - это разделение вычислительной нагрузки между двумя отдельными процессами: клиентом и сервером. Компанией Garthner Group, специализирующейся в области исследования информационных технологий, предложена следующая классификация ***двухзвенных моделей*** взаимодействия клиент-сервер (двухзвенными эти модели называются потому, что три компонента приложения различным образом распределяются между двумя звеньями (узлами)) (рис.4.26).

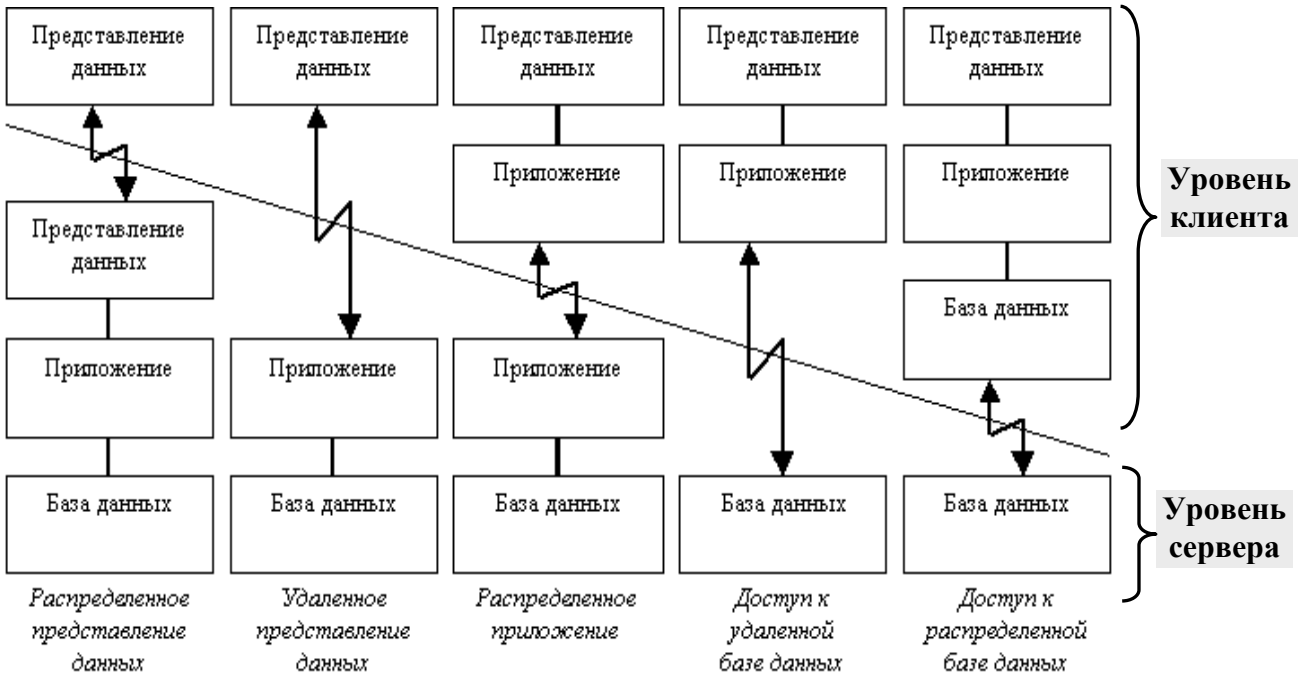


Рис. 4.26. Представление вычислений в сети в двузвенных моделях клиент/сервер

Обычно, звеном-***клиентом*** служит настольный ПК, на котором запускается ***программное обеспечение (ПО) конечного пользователя*** *(front-end software)*, или

любая прикладная программа, приложение либо пакет, способные направлять запросы по сети к серверу и обрабатывать информацию, которая будет получена в ответ на запрос. ***Сервер*** *(back-end* ***software, серверная часть)****,* в свою очередь, получает запросы и предпринимает действия от имени клиента.

К примеру, ПК, который работает под управлением Windows 95 и выполняет клиент-серверную программу, разработанную в среде Delphi, может представить свой запрос на рассмотрение серверу баз данных (скажем, программе Oracle7, версии 8.1, запущенной на сервере Windows NT). Обычно клиент посылает запросы базе данных в виде предложений на языке структурированных запросов (SQL- Structured Query Language), используя понятный серверу базы данных диалект. Результат выборки данных из базы данных сервер вернёт клиентскому приложению, работающему на компьютере-клиенте.

В структуре архитектуры и модели клиент-сервер ***промежуточное обеспечение (программное и аппаратное)*** *(middleware)* предоставляет общий интерфейс для ПО конечного пользователя и сервера. Этот интерфейс проникает сквозь слои графического интерфейса пользователя GUI, операционную систему вычислительной сети и собственных драйверов базы данных с помощью общих вызовов и с применением технологии ODBC[18]. Для завершения операций сервер базы данных выполняет запрос и передает клиенту обратно затребованные данные для обработки их программой клиента.

В связи с постоянным ростом сложности схем сетевого взаимодействия компьютеров, количества языков и систем программирования, программных систем и приложений, в компьютерном мире назрел вопрос: ***"Как обеспечить взаимодействие программных компонентов, которые были написаны в разное время, в разных частях мира, разными компаниями и людьми?"*** Практически для решения этого вопроса необходимо обеспечить следующие условия:

❶ Возможность производителей писать свои собственные компоненты и при этом быть уверенным, что они смогут взаимодействовать между собой на разных платформах и с компонентами других производителей.

❷ Согласованность новых версий компонент со старыми их релизами (версиями).

❸ Независимость компонентов от используемых при их написании типов и диалектов языков программирования.

❹ Свободную связь между компонентами, работающим как в одном, так и в разных процессах (и, возможно, на разных машинах), с использованием единой простой программной модели.

На пути преодоления вышеуказанных проблем, в качестве одной из первых ***клиент/серверных*** моделей программного взаимодействия была разработана и применена фирмой Microsoft технология создания ***динамически***

[18] ODBC (Open Database Connectivity) – представляет собой технологию и спецификацию интерфейса для доступа к базам данных различных форматов, разработанную Microsoft. По сути, это интерфейс API, такой же как и Windows API, который имеет дело с программированием баз данных. Архитектура ODBC включает в себя четыре компонента: приложение (программа пользователя), ODBC менеджер, ODBC драйвера и источник данных (базы данных, например, Interbase, Oracle и др.).

связанных библиотек (DLL–Dynamic Linking Library) для операционной системы Windows. В этой модели клиентами являются:

❶ приложения, которые обращаются к элементам-серверам библиотек DLL;

❷ елементы-клиенты библиотек DLL, которые обращаются к элементам-серверам, которые находятся в своей или чужой библиотеке (рис. 4.27).

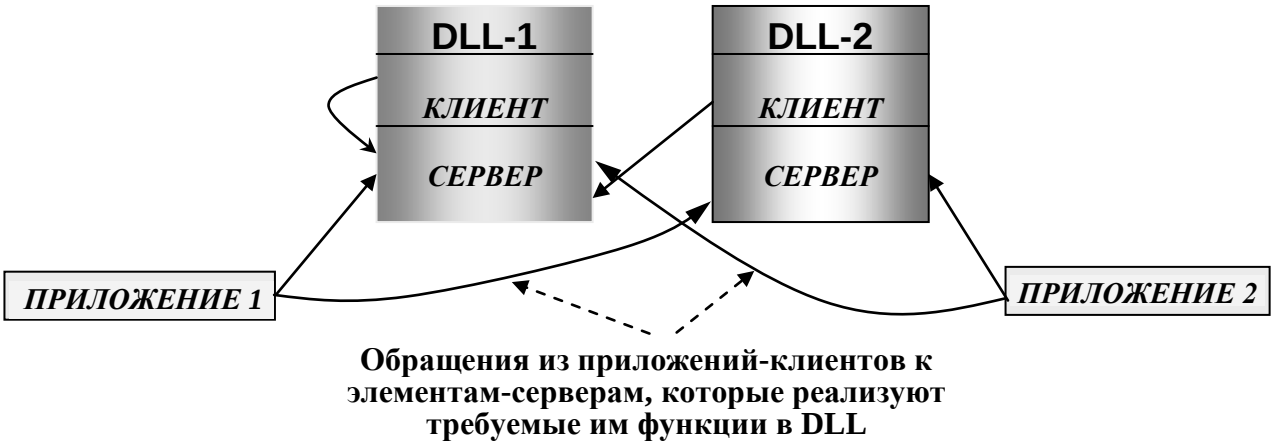


Рис. 4.27. Клиент-серверные связи между приложениями: внешние (от приложения – к DLL) и внутренние (от клиенту-функции DLL – к серверу-функции DLL) в ОС Windows

Развитие этой технологии воплотилось в программной модели, спецификации, архитектуре и технологии, разработанной Microsoft под общим названием COM – Component Object Model (Объектная Модель Компонентов) и основанной также на модели клиент/сервер.

В COM, любая часть программного обеспечения реализует свои сервисы как один или несколько объектов COM (не следует путать объекты COM с объектами в языках программирования типа C++. Несмотря на то, что у них есть общие черты, это разные вещи).

Одним из краеугольных камней технологии COM является принцип взаимодействия программных компонентов исключительно через интерфейсы (рис. 4.28).

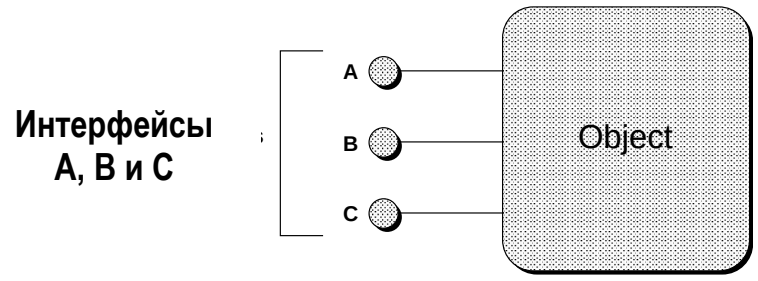


Рис. 4.28. Компонент с группой своих интерфейсов

Каждый объект COM поддерживает один или несколько интерфейсов, реализующих те или иные методы. Клиенты могут получить доступ к сервисам

сервера объекта COM только через вызовы методов интерфейсов объекта — у них нет непосредственного доступа к данным объекта.

Использование интерфейсов COM позволяет:

❶ унифицировать процесс проверки возможностей программного компонента;

❷ обеспечить связь между компонентами, которые находятся в разных нитях, процессах и, возможно, на разных компьютерах в сети;

❸ снизить затраты на разработку программного обеспечения, повысив степень многоразовости использования уже разработанных компонентов.

В основе модели COM лежат несколько важных концепций:

❶ двоичный стандарт вызова функций (для любой платформы);

❷ обеспечение строгой типизации при объединении функций в интерфейсы с уникальными идентификаторами (GUID);

❸ управление временем жизни объекта;

❹ механизм однозначной идентификации компонентов и интерфейсов COM по уникальным идентификаторам (GUID).

Компоненти COM реализуются в виде библиотек DLL, элементов ActiveX, кнопок и команд графических интерфейсов приложений (Word, Fotoshop и др.), элементов форм и так далее. Спецификация на уровне структур данных позволяет связывать компоненты, написанные для разных платформ (Microsoft® Windows®, Microsoft Windows NT™, Apple® Macintosh®, UNIX® и некотрых других). Стандарт COM является открытым, что позволяет создавать собственные компоненты и использовать компоненты, которые созданы другими фирмами.

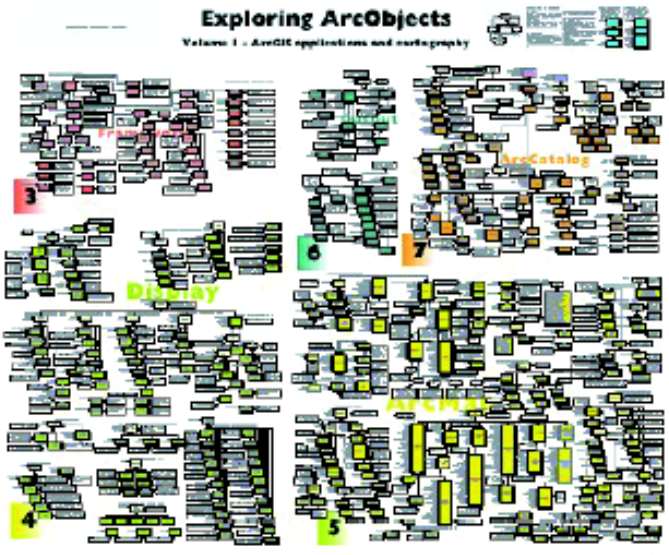


Рис. 4.29. Представление небольшой части компонентов ArcObjects (фирмы ESRI) с их интерфейсами

В настоящее время COM является наиболее широко используемой в мире программной компонентной моделью. По некоторым данным в 2000 г. компоненты COM использовались на более чем 150-и миллионах систем по всему миру. Одной только фирмой ESRI (Environment System Research Institute) (США) на конец 2002 года было разработано более 1200-т компонентов под общим названием ArcObjects для нужд геоинформационных систем (ГИС) (рис.4.29).

Дальнейшим продолжением развития модели COM явилась модель DCOM, то есть

распределённая (Distributed) COM, которая расширяет границы взаимодействия компонентов в сетевом пространстве (рис. 4.30).

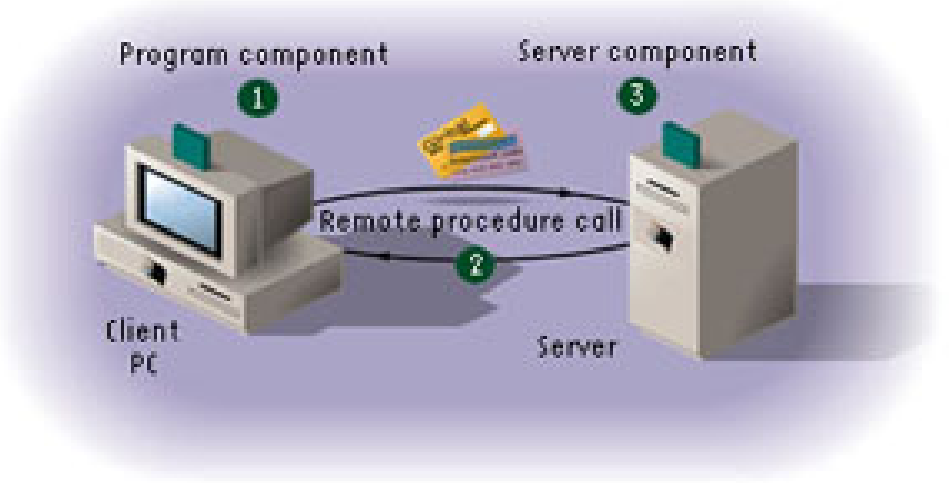


Рис. 4.30. Принцип взаимодействия компонентов с помощью модели DCOM

При вызове одним компонентом другого, в соответствии с принципами модели DCOM (см. рис.4.30), происходит следующее.

1. Компонент программы на клиентском ПК выполняет вызов, который обращается к удалённому серверному компоненту, такому, к примеру, как система авторизации кредитных карточек. При этом он обращается к операционной системе Windows.
2. DCOM установливает соединение через сеть между двумя компонентами, используя механизм удалённого вызова процедур (RPC).
3. Клиентский компонент может взаимодействовать с серверным компонентом так, как если бы они размещались на одной и той же машине.

Такая модель очень хорошо поддерживает модель ***роспределённых вычислений***, которая является парадигмой организации приложений, в соответствии с которой разные части одной и той же программы могут выполняться на разных компьютерах в сети.

Параллельно с DCOM развивается подобная архитектура сетевых вычислений под названием Common Object Request Broker Architecture (CORBA), которую поддерживают такие крупные компании, как IBM, Sun Microsystems і ряд других производителей. Сегодня она применяется многими большими организациями для разворачивания распределённых вычислений у масштабах предприятия на базе Unix-серверов и мэйнфреймов.

Общая архитектура распределённых интероперабельных информационных объектов CORBA (см. подраздел 6.4), базируется на концепции ***промежуточного слоя (middleware)***, который содержит службы и средства поддержки ***глобального пространства объектов***, ***их жизненного цикла*** и ***интероперабельности*** Этот слой находится между операционной системой информационной системы (включая средства управления компьютерными сетями) и прикладными системами, специфичными для конкретных областей применения. Компонентами такого ***глобального пространства*** являются

произвольные информационные ресурсы – программные компоненты, базы данных, базы знаний, файлы данных, которые в том числе содержат и мультимедийную информацию, компоненты существующих информационных систем и т.д., которые представлены коллекциями объектов, независимых от аппаратно/программных платформ, их реализаций и размещения в (сетевом) пространстве. Унифицированные интерфейсы взаимодействия таких компонентов призваны поддерживать создание интероперабельных сред информационных ресурсов, которые обеспечивают динамическую реализацию ***композиций доступных ресурсов***, которые организуются для решения конкретных проблемных задач (к примеру, реализация требуемой информационной системы, реализация проекта, поддержка групповой деятельности и т.д.). Кроме того, они предназначены для развития методологии мегапрограммирования – программирования в среде готовых компонентов для обеспечения технологий ***повторного использования*** информационных ресурсов (например, в Интернете, экстранете, интранете и т.д.).

Главными компонентами архитектуры промежуточного слоя, которая развивается консорциумом OMG (см. подраздел 7.2), являются.

❶ Общая Архитектура Брокера Объектных Запросов – Common Object Request Broker Architecture, то есть – CORBA;

❷ Брокер Объектных Запросов (Object Request Broker - ORB) (рис.4.31), Он играет основную роль "***общей шины***" в глобальном пространстве объектов, с помощью которой объекты этого пространства могут взаимодействовать друг с другом, имея широкую поддержку целым рядом программных систем и продуктов, которые распространяются известными производителями: Digital, IBM, SunSoft, IONA Technologies и некоторыми другими для разных типов платформ.

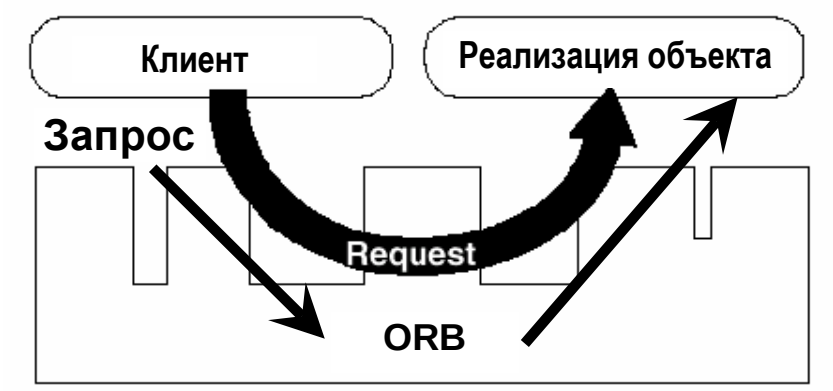


Рис. 4.31 Клиентская программа, взаимодействующая с объектом через ORB

Общая система Объектных Служб и Общих Объектных Средств (CORBA) организует нижний слой в архитектуре промежуточного слоя, который обеспечивает технологическую платформу интероперабельности (т.е. способность к взаимодействию). Один из ключевых принципов архитектуры CORBA, обеспечивающий интероперабельность функционирующих в сетевом информационном пространстве приложений, заключается в независимости спецификаций интерфейсов вызываемых в процессе их работы объектов от их

реализации. Важным компонентом такой концепции является ***язык определения интерфейсов (IDL - Interface Definition Language)***. В версии стандарта CORBA 2.0 этот язык начал называться OMG IDL. Таким образом, OMG IDL является средством, с помощью которого потенциальным клиентам ***объекта-сервера*** сообщается, какие операции доступны в его ***интерфейсе объекту-клиенту*** и как их необходимо вызывать.

IDL – является языком однородного специфицирования интерфейсов разнообразных информационных ресурсов, инкапсулированных средствами технологий CORBA в пространство сетевого взаимодействия. Этот язык является чисто описательным. Определения на языке IDL могут быть отображены стандартными средствами в любом конкретном языке программирования, таком как, C, C++, Object Pascal, Smalltalk и некоторых других. Репозиторий интерфейсов, который содержит определения интерфейсов на языке IDL, позволяет видеть интерфейсы доступных компонентов-серверов в сети и программировать их вызов, а также использовать в програмах-клиентах с учётом возможностей Брокера ORB. Спецификации, которые содержатся в репозитории, могут быть также использованы в процессе выполнения программы-клиента. Роли "клиента" и "сервера" следует рассматривать как относительные: клиент (сервер) в архитектуре CORBA может буть сервером (клиентом) по отношению к другим клиентам (серверам).

Активное распространение и повсеместное применение клиент-серверных компьютерных моделей расширили и углубили понятие и применение концептуальных моделей интерфейса в целом.

Действительно, если в информационных структурах связываются две системы (или больше), то это делается в целях обеспечения совместной работы. В этом случае, по сути, ***интерфейс*** - является спецификацией (описанием) и соответствующей реализацией процедуры или процесса их взаимодействия. Интерфейс определяет свойства, характеристики и параметры границ взаимодействия систем (или является их описанием). Параметры могут быть логическими, физическими, функциональными, электрическими и, возможно, другими, а характеристики – обычно конструктивными. Другими словами, характеристики могут определяться конструкцией устройства, решаемыми им задачами, а также применяемыми технологиями.

Таким образом, за внешней простотой и комфортностью любого существующего ***интерфейса*** используемых человеком устройств, (включающих микропроцессорные элементы – тьюнеров, холодильников, автомобилей и др.) или информационных систем (компьютеров, мобильных систем, программных компонентов и др.) – скрывается многослойная структура многочисленных ***межкомпонентных интерфейсов*** и реализующих их программно-аппаратных средств. Описание взаимодействия разных логических, физических и программных компонентов ***модели клиент-сервер*** в обобщённом виде может быть представлено так, как показано на рисунке 4.32.

Рис. 4.32. Многоуровневая модель структур представлений интерфейса

На рисунке изображены разные "представления" сложного понятия интерфейс в модели клиент-сервер. Действительно, во-первых, в его структуре представлены реализации совместного, реального взаимодействия объектов самой разной природы (человек-компьютер, человек-программа, программа-принтер и т.д.).

Во-вторых, в процессе проектирования и моделирования сложных процессов взаимодействия информационных систем (например, средствами языка UML) интерфейсы рассматриваются на концептуальном, абстрактном, логическом, физическом и архитектурном уровнях (к примеру, введение логических имён устройств, описание взаимодействия компьютеров в сети, взаимодействие программ и распределённых компонентов, которые находятся на нескольких компьютерах одновременно и т.д.).

В-третьих, в связи с расширеннем способов передачи данных между компьютерами, и в том числе и беспроводно, усложняются протоколы и спецификации взаимодействия, а следовательно, и интерфейсы.

Если же рассматривать последовательно принципы взаимодействия сущностей или объектов в процессе их взаимодействия и последующей реализации интерфейсов между ними, то, согласно рисунка 4.32, последовательность изучения с целью дальнейшей реализации такого взаимодействия может выглядеть следующим образом.

1. Принципиальное рассмотрение возможностей реализации взаимодействия двух компонентов или элементов на уровне запросов необходимых сервисов (***клиент***) и обеспечения этими сервисами (***сервер***), проводится на концептуальном и архитектурном уровнях. Здесь изучаются возможности принципиальных подходов для реализации взаимодействия на программном и физическом уровнях, то есть на уровнях поддержки процессов информационного взаимодействия сущностей путём разработки конкретных кодов программ, электрических, электронных и механических компонентов, а также осуществления информационного взаимодействия с помощью создания соответствующих ***интерфейсов***.

2. Далее следует этап анализа и проектирования, то есть представления и моделирования компонентов, которые принимают участие во взаимодействии, реализация которого разрабатывается. На данном уровне применяются абстракции данного этапа, как правило, при поддержке средств языка UML.

3. При выборе средств реализации в рамках моделей интерфейсов и языков реализации применяются логические абстракции.

4. На уровне программной реализации моделей, которые разработаны (классов, объектов и их многоуровневого взаимодействия), применяются средства конкретного языка программирования: C++, Object Pascal, Visual C++, Java, C# и других (программные абстракции).

5. По мере развития технологий всё большее число разнообразных электронных компонентов приобретают возможность использовать сервисы друг друга (технологические абстракциии). К примеру, мобильные телефоны уже имеют возможность обмениваться данными с серверами Internet и другими службами WWW.

6. Физические абстракции позволяют выделять наборы компонентов в логически связанные комплексы, которые представляют собой элементы с более высоким уровнем организации (к примеру, организация интерфейса беспроводного манипулятора мышь с компьютером).

Рассматривая комплексно совокупность множества интерфейсов, существующих между сложными системами (к примеру, человеком и компьютером), можно назвать их системными.

Подводя итоги всему вышесказанному, виды интерфейсов, которые поддерживаются в сложной, многоуровневой архитектуре клиент-сервер, можно условно разбить на следующие группы (таблица 4.1).

Таблица 4.1

Взаимодействующие сущности, объекты, системы, компоненты, элементы и их интерфейсы

№ пп.	Взаимодействующие сущности, объекты и т.д.	Вид (тип) интерфейса	Способы и средства поддержки	Уровни реализации
1	Человек–реальный объект	Системный	Компьютерные, языки программирования, язык UML, CASE-средства, физические устройства (микропроцессоры)	Логический, программный, физический
2	Человек–компьютер (ПК)	Системный	Командные (текстовая строка, сочетания клавиш клавиатуры, щелчки клавишами мыши); графические (графические объекты, кнопки, меню); языки программирования; RAD-средства, физические устройства	Логический, программный, физический
3	Человек–устройство ПК	Системный	Логические имена в ПК, языки программирования, RAD-средства, кнопки устройств	Программный, физический
4	Человек–программа (приложение)	Системный	Командные (текстовая строка, сочетания клавиш клавиатуры, щелчки клавишами мыши); графические (графические объекты, кнопки, меню); языки программирования; RAD-средства	Программный
5	Компьютер–компьютер	Системный	Сетевые, устройства коммуникации, программы, языки программирования	Программный, физический
6	Программа–программа	Программный	Программные, имена, языки программирования	Программный
7	Объект (программы) – объект	Программный	Програмные, имена, GUID, UUID, CLSID, языки программирования	Программный
8	Программа–устройство	Программный	Программные, имена, языки программирования, устройства коммуникации, протоколы	Программный, физический
9	Устройство–програма	Программный	Программные, имена, языки программирования, устройства коммуникации, протоколы	Программный, физический
10	Устройство–устройство	Физический	Устройства коммуникации, протоколы, соглашения	Физический, программный
11	Устройство–устройство	Программный	Программные, языки программирования	Программный

С проникновением компьютеров во всё большое число областей деятельности человека, системный интерфейс продолжает оставаться "горячей"

точкой концентрации усилий разработчиков аппаратных и программных средств. Вместе с тем, и это самое главное, продолжают совершенствоваться информационные модели и технологии их реализации. В последнее время появились визуальные средства программирования (Delphi, Visual C++, Visual Basic и ряд других), в которых реализованы объектно-ориентированные концепции программирования и инструменты реализации соответствующих моделей (COM, DCOM, OLE Automation, ActiveX и т.д.). Быстро развиваются компонентные модели и технологии, базирующиеся на спецификациях и архитектурах .NET и Java, которые реализуют широкие функциональные возможности взаимодействия устройств, программ, компонентов, а также обработки информации в Internet и WWW.

Одновременно с ними совершенствуются новые технологии анализа и проектирования (UML, CASE-технологии, Web-технологии и много других). Не отстают и разработчики аппаратных средств, которые предлагают все новые и новые технические и технологические решения организации доступа ко всё возрастающим объёмам разнообразной информации (DVD, беспроводные Internet-сервисы, радио и телевидение в Internet и т.д.), а гонка всё продолжается...

Вопросы.

1. Почему понятие ″интерфейс″ имеет несколько значений?
2. Приведите примеры интерфейсов разных уровней.
3. Чем отличается аналоговое представление процесса от цифрового?
4. Что такое данные? Приведите примеры.
5. Что такое информация? Приведите примеры.
6. Какие концептуальные составляющие должны присутствовать для обеспечения процесса взаимодействия пользователя с компьютером?
7. Какие экранные объекты, обеспечивающие графический интерфейс пользователя являются пассивными, а какие активными?
8. Какие концепции лягли в основу разработки WIMP-интерфейсов?
9. Приведите примеры метафор, которые используются для создания графического интерфейса пользователя?
10. Какие задачи выполняют в компьютере расширения имён файлов?
11. Чем отличаются аппаратные интерфейсы от программных?
12. Каковы основные элементы модели клиент-сервер и в чём состоит основной принцип их взаимодействия?

В корне неверной является идея, что имеется только один правильный способ или язык, чтобы решить любую проблему для любого пользователя.

Бьёрн Страуструп, автор языка C++.
(из интервью 21.02.2001)

5. ЭВОЛЮЦИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

5.1. Начало развития языков программирования

Создатели первых языков программирования высокого уровня для компьютеров, стремились делать их в меньшей степени похожими на среду общения между человеком и компьютером и в большей – на упорядоченный набор знаков и символов. Изначально держать курс на традиционную и устоявшуюся математическую символику предложил Х. Рутисхаузер (в 1952 г.), ставший родоначальником идеи языков программирования и одним из авторов языка Алгол-60. Широкое распространение и применение его идеи получили лишь в 1957 г., после того, как корпорация IBM опубликовала описание языка Фортран и реализовала для него компилятор, транслировавший программы в машинный код. По сути, с этого момента и началась эпоха языков программирования (рис. 5.1).

```
C ***************************************
C Programming by. CHOI YONG SOK : 199500689
C special type [ using subroutine ]
C Upgrade version !!!
C ***************************************

      INTEGER M, N
      INTEGER TRACE
      PARAMETER(M = 3, N = 3)
      INTEGER MATRIX(M,N)
      INTEGER MATRIX_T(M,N)
      INTEGER ROW, COL
      INTEGER A(9)
      INTEGER I, SIZE

      TRACE = 0
      SIZE = M*N

      PRINT *, 'Please input matrix component (unit $ row)
      DO 10 ROW = 1, M, 1
        READ *, (MATRIX(ROW,COL), COL = 1, N, 1)
   10 CONTINUE

      PRINT *, ' '
      PRINT *, '=== Your input matrix ==='

      DO 20 ROW = 1, M, 1
        PRINT *, (MATRIX(ROW,COL), COL = 1, N, 1)
   20 CONTINUE
```

а)

```
// The main Program
 begin
  integer N;
  Read Int(N);

  begin
    real array Data[1:N];
    real sum, avg;
    integer i;
    sum:=0;

  for i:=1 step 1 until N do
      begin real val;
        Read Real(val);
        Data[i]:=if val<0 then -val
else val
      end;

  for i:=1 step 1 until N do
      sum:=sum + Data[i];
    avg:=sum/N;
    Print Real(avg)
  end
end
```

б)

Рисунок 5.1. Фрагменты текста программ на языках Фортран (а) и Алгол (б)

К настоящему времени программирование, как процесс, состоит в создании компьютерных программ, содержащих конкретные инструкции

(команды) для выполнения их компьютером. Различные части программы могут быть написаны на различных языках программирования (ЯП). Язык программирования представляет собой стандартизированное средство коммуникации для сообщения компьютеру команд на выполнение конкретных задач. Он также предоставляет программисту возможность точно указать, какими видами данных компьютер должен манипулировать, а также последовательность действий в различных обстоятельствах. Таким образом, преследуются две основные цели:

❶ выражать программу на логическом уровне, значительно превосходящем логику низкоуровневых кодов блока центрального процессора (CPU-Central Processing Unit);

❷ обеспечивать совместимость и переносимость разрабатываемых программ между различными компьютерными платформами, которые, как правило, имеют разное программное обеспечение (ПО) для их трансляции в машинный язык конкретной системы.

Если механизм трансляции применяется ко всему тексту программы целиком для перевода её во внутренний формат компьютера, то такой вид обработки называется ***компиляцией***. В результате компиляции фрагмента кода программы мы получаем файл с расширением .EXE. В противоположном случае текст программы транслируется шаг за шагом, и каждый шаг выполняется немедленно и такой механизм называется ***интерпретацией***. Интерпретируемые программы выполняются более медленно, чем компилируемые, но обладают значительно большей гибкостью при взаимодействии с програмно-аппаратными средствами компьютера (рис. 5.2).

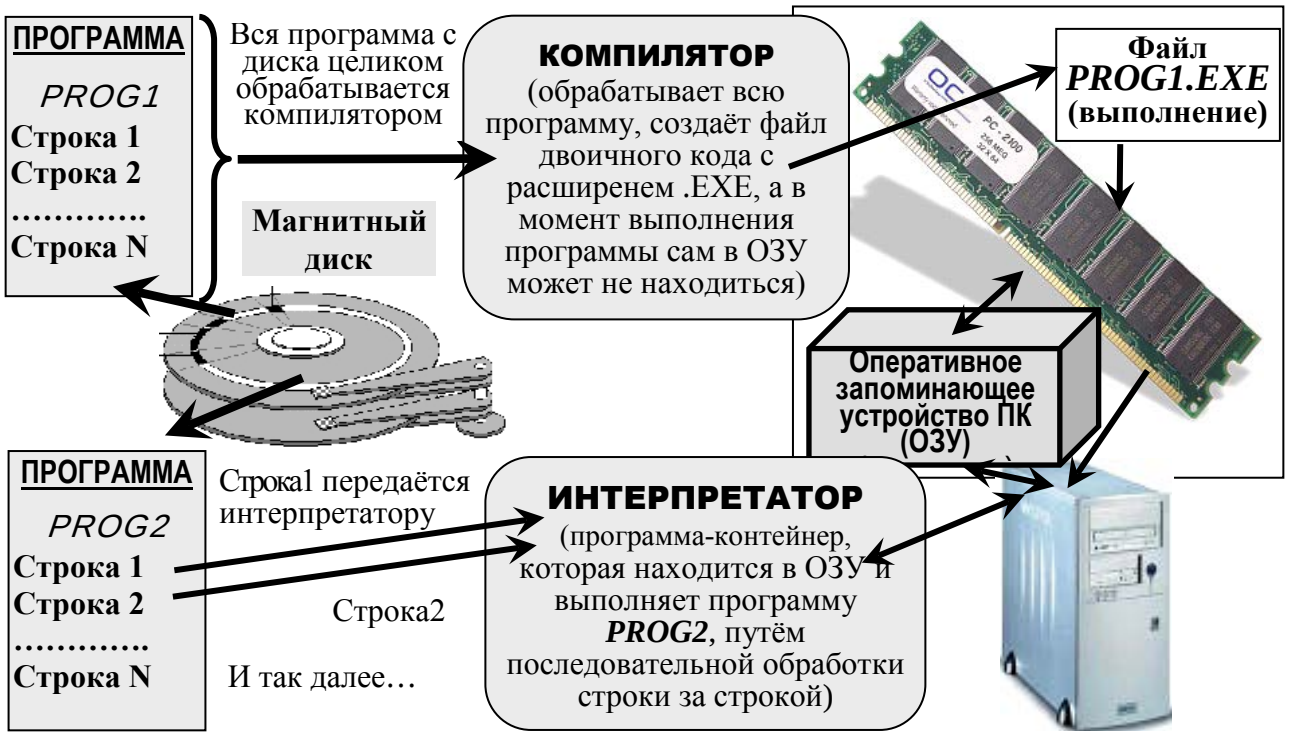


Рис. 5.2. Технология обработки исходных текстов программ компиляторами и интерпретаторами с последующим выполнением

В современных компьютерах интерпретаторы трансформируются в программы (приложения)-контейнеры, обладающие значительно большей функциональностью. К ним можно отнести приложения MS Excel, MS Word, MS Internet Explorer и многие другие (см. главу 6).

5.2. Расширение функциональности языков программирования

В компьютерах 70-80-х годов на начальных этапах их развития большое значение имели языки программирования систем (system programming language) (ЯПС). К ним можно отнести языки ассемблеров и ряд языков высокого (high level) уровня (Паскаль, Си, C++ и Java). Постепенно последние почти полностью вытеснили языки ассемблера при разработке больших приложений. При компиляции текстов программ этих языков получаются достаточно компактные двоичные коды, дающие высокую скорость выполнения программ. Одним из главных достоинств ЯПС всегда считалась так называемая типизация, при которой:

❶ каждая переменная, для хранения данных, должна быть изначально декларирована с тем, чтобы быть приписанной к определенному типу: целое (Integer), вещественное (Real, Double), указатель на строку (String) и т. д., а также должна использоваться только теми способами, которые этому типу соответствуют;

❷ данные и код разделены. При этом трудно, если вообще возможно, создать новый код во время выполнения программмы;

❸ переменные могут быть сгруппированы в объекты с хорошо определенной структурой и процедурами для манипуляции ими. Объект одного типа не может быть использован там, где ожидается использование объекта другого типа (рис. 5.3).

(Число типа *Integer*) нельзя (!!!) разделить не нацело
на (Число типа *Real*)
если результат присваивается переменной **I** типа *Integer*.
То есть, оператор вида: `I := (24 / 5.37e-2);` –
запрещён!

Рис. 5.3. Пример типизации в языке Турбо Паскаль

Типизация обеспечивает целый ряд преимуществ:

❶ большие программы она делает более технологичными благодаря точному определению используемых сущностей и их отличий от других;

❷ компиляторы используют информацию о типах для обнаружения определенных видов ошибок, таких как попытка, к примеру, задействовать величину с плавающей точкой как указатель;

❸ типизация повышает эффективность исполнения, позволяя компилятору генерировать специализированный компактный двоичный код.

В сильно типизированном языке программист обязательно декларирует, как каждая порция информации будет использована, а язык предотвращает ее использование другим способом. В слабо типизированном языке не существует априори заданных ограничений на использование информации; ее смысл определяется только способом, которым она используется.

Вместе с тем, сами по себе современные компьютеры принципиально бестиповые. Любое слово памяти может хранить величину любого типа, будь то целое или вещественное число, указатель или команда. Смысл (значение) величины определяется тем, как она применяется. Одно и то же слово может использоваться в разных случаях по-разному. К примеру, в языке Visual Basic (VB) 6.0 специальный тип данных ***Variant*** позволяет хранить данные всех остальных типов (с возможностью последующей обработки соответствующими обрабатываемому типу операциями) (рис. 5.4).

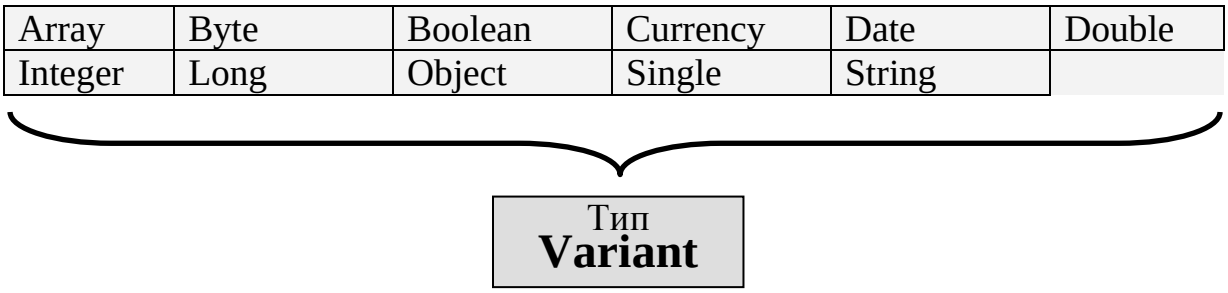


Рис. 5.4. Многофункциональность типа **Variant**
в языке Visual Basic

С середины 90-х годов начали появляться так называемые языки сценариев (scripting languages, скриптовые языки) (ЯС). В первую очередь это было связано с вовлечением в сетевые структуры всё большего количества компьютеров, активизацией процесса распределённых вычислений в сетевых средах и резким ростом объёма информации, обрабатываемой через Internet средствами World Wide Web (WWW). Языки программирования, как правило, разрабатывались для интерактивного использования и имели в своём составе много команд, представляющих собой мини-программы, предназначенные для комбинирования уже существующих компонентов. Из более, чем 30-ти наиболее популярных ЯС (см. табл. 5.1) можно особо отметить языки Visual Basic, Perl, PHP, Rexx, Tcl и оболочки Unix (shell).

Таблица 5.1.

Список основных языков скриптов в алфавитном порядке

ASP (Active Server Pages)	Dylan	JCL	Pike	ScriptBasic
	E	Lua	Pliant	sh (shell)
AppleScript	Euphoria	Miva	Python	Simkin
Awk	Guile	MUMPS	QuakeC	Tcl
bash	ICI	ObjectRexx	REBOL	UnrealScript
Brain	JavaScript (ECMAScript)	Perl	Rexx	Visual Basic (VB)
CobolScript		PHP	Ruby	VBScript

И хотя языки сценариев часто используются для расширения свойств компонентов, они мало пригодны для программирования сложных алгоритмов

и структур данных, обычно как раз и обеспечиваемых компонентами. Вот почему языки сценариев часто называют ***склеивающими языками*** (glue languages) или языками интеграции систем (system integration languages).

″Приближение″ структуры средств ранее разработанных языков программирования к особенностям современных требований приводит к превращению языков в гибридные, т.е. обладающие свойствами разных групп языков (к примеру, C++, Visual Basic, Java и др.). Так как процесс разработки языков программирования продолжается, то в таблице 5.2 приведены характеристики наиболее популярных среди пользователей языков программирования разного назначения. Более полная информация приведена в Приложении 3.

Таблица 5.2.

Наиболее распространённые языки программирования (ЯП) для ПК

Наименование ЯП	**Год создания**	**Тип языка**	**Автор (ы) разработки**
Pascal	1970	ЯПС/–	Никлаус Вирт
C	1972	ЯПС/–	Дэннис Ритчи
C++	1980	ЯПС/ОО	Бьёрн Страуструп
Turbo Pascal	1983	ЯПС/–	Андерс Хейльсберг, Филипп Кан
Perl	1986	ЯС/ОО	Ларри Вол
Python	1990	ЯС/ОО	Гвидо ван Россум
Visual Basic	1991	ЯС/ОО	Microsoft
Delphi/Object Pascal	1995	ЯПС/ОО	Андерс Хейльсберг
Java	1995	ЯПС/ОО	Патрик Нотон и Джеймс Гослинг
Java Script	1995	ЯС/ОО	–
VBScript	1997	ЯС/ОО	Microsoft
XML	1998	–/–	W3C
C#	2000	ЯПС/ОО	Андерс Хейльсберг
ПРИМЕЧАНИЯ: ЯПС–язык программирования систем, ЯС–язык скриптов ОО–объектно-ориентированый язык.			

Одним из важнейших признаков классификации языков программирования является принадлежность их к одному из стилей, основные из которых приведены в таблице 5.3.

Таблица 5.3.

Стили языков программирования

Стиль (тип) языка программирования	**Найменование языка**
Процедурний	FORTRAN, COBOL, BASIC, PL/I, FORTH, Pascal, Turbo Pascal
Декларативний (функциональный)	LISP, Common Lisp
Логический (реляционный)	Prolog
Параллельное программирование	Ada, C++, Concurrent Pascal, Occam
Объектно-ориентированный	Smalltalk, C++, Object Pascal, Python, Java, Simula, Java Script, VBScript, C#
Модульный (компонентный)	C#, C++, Java, Object Pascal, Visual Basic, Ada, Modula-2, Eiffel, Component Pascal
Специализированные языки	HTML, XML, Tcl, Perl, Cold Fusion

Для представления выразительных возможностей языков программирования разных стилей в таблице 5.4 приведены фрагменты программ реализации вывода на экран дисплея текстового сообщения “Hallo World!”, которое стало традиционным в курсах изучения практически всех языков кодирования.

Таблица 5.4.

Программы вивода на экран дисплея стандартной фразы “Hallo, World!”, выполненные на языках програмирования разных стилей.

Процедурный языкTurbo Pascal

```
Program Hello;

Begin
  Writeln ('Hello, World!');
End.
```

Процедурный языкFORTRAN

```
c   Hello, world.
c
      Program Hello
      implicit none
      logical DONE

      DO while (.NOT. DONE)
        write(*,10)
      END DO
   10 format('Hello, World!')
      END
```

Декларативный (функциональный) язык LISP

```
; LISP
(DEFUN HELLO-WORLD ()
         (PRINT (LIST 'HELLO 'WORLD)))
```

Логический (реляционный) язык Prolog

```
% HELLO WORLD.
         Works with Sbp (prolog)

hello :-
printstring("HELLO, WORLD!!!!").

printstring([]).
printstring([H|T]) :- put(H),
printstring(T).
```

Язык параллельного програмирования Ada

```
   with Text_Io; use Text_Io;

   procedure hello is
   begin
      put ("Hello, world!");
   end hello;
```

Модульный (компонентный) язык Visual Basic

```
Private Sub Form_Load()
Static I
for I = 1 to 10
msgbox "Hello, World!"
Next I
end sub
```

Объектно-ориентированный язык С++

```
hallo.cpp
      #include <iostream> //
      ::std::cout
      #include <ostream>  // <<
      int main(){ ::std::cout <<
      "Hallo, world!" << '\n'; }
```

Объектно-ориентированный язык Java

```
Hallo.java
      public class Hallo
      { public static void main(
      String[] args )
        { System.out.println( "Hallo,
      World! " ); }}
```

Специализированный язык HTML

```
<HTML>
<HEAD>
<TITLE>Hello, World!</TITLE>
</HEAD>

<BODY>
Hello, World!
</BODY>
</HTML>
```

Специализированный язык Perl

```
  print "Hello, World!\n" while (1);
```

С точки зрения классификации языков программирования по типам решаемых с их помощью задач, некоторые авторы предлагают следующее разделение (табл. 5.5).

Таблица 5.5.

Классификация языков программирования по типам решаемых с их помощью задач

Тип решаемой задачи	Наименование языков
Задачи искусственного интеллекта	Lisp, Prolog, Multilisp, Common Lisp, Рефал, Planner, QA4, FRL, KRL, QLisp
Параллельные вычисления	Fun, Apl, Alfl, PARAlfl, ML, SML, PPL/1, Hope, Miranda, Occam, PFOR, Glypnir, Actus, параллельный Cobol, ОВС-ЛЯПИС, ОВС-Алгол, ОВС-Фортран, PA(1), PA(G)
Задачи вычислительной математики и физики	Occam, PFOR, Glypnir, Actus, параллельный Cobol, ОВС-ЛЯПИС, ОВС-Мнемокод, ОВС-Алгол, ОВС-Фортран, PA(1), PA(G)
Разработка интерфейса	Forth, c, C++, Ассемблер, Макроассемблер, Simula-67, OAK, Smalltalk, Java, РПГ
Разработка программ-оболочек, разработка систем	Forth, c, C++, Ассемблер, Макроассемблер, Simula-67, OAK, Smalltalk, Java, РПГ
Задачи вычислительного характера	Python, Algol, Fortran, Cobol, Ada, PL/1, Фокал, Basic, Pascal
Оформление документов, обработка больших текстовых файлов, организация виртуальных трехмерных интерфейсов в Интернете, разработка баз данных	HTML, XML, Perl, Tcl/Tk, VRML, SQL, PL/SCL, Informix 4GL, Natural, DDL, DSDL, SEQUEL, QBE, ISBL
Языки сетевой обработки (реализации Web–сервисов)	Java, C++, C#, Visual Basic .NET
Клиент-серверные языки программирования в WWW	Java, JavaScript, VBScript, XML, ASP (Active Server Pages), PHP, Perl, Python, ScriptBasic, HTML, CGI, CSS
Языки программирования геоинформационных систем (языки встроенных систем)	Avenue, Visual Basic for Application, ArcXML, MapBasic, AutoLisp, Visual Basic, VBScript, GML

Приложения и программы пишутся для решения всё более сложных задач с использованием быстро развивающихся информационно-компьютерных технологий, Web– и Интернет технологий. По некоторым данным[1], мир наконец-то осознал фундаментальную важность сетевой обработки данных, и чтобы это оценить, следует вдуматься в следующую ошеломляющую цифру: в 2000 году рыночная стоимость 200 крупнейших сетевых американских компаний, акции которых котируются на фондовой бирже, превысила 5 трлн долл. За этот же год их суммарный оборот увеличился на 100 млрд. долл. (т.е. на 14%) — с 709,7 в 1999 г. до 809,6 млрд долл. в 2000 г.

[1] http://osp.admin.tomsk.ru/nets/2000/07/066.htm

И, как стало ясно из развития событий, ключ к Internet и Web-сервисам — это языки программирования, ориентированные на создание компонентных приложений, программирование беспроводных интерфейсов и мобильных устройств.

Идеолог языков программирования из корпорации Microsoft Андерс Хейльсберг охарактеризовал C# (“си шарп”) как “первый настоящий компонентно-ориентированный язык программирования в семействе Си/Си++”. Он также уверен, что модель программирования, основанная на компонентах с ассоциированными с ними данными (соответствуют свойствам, properties, в некоторых других языках) и вариантами поведения (соответствуют событиям, events), поддерживается в C# более естественно, чем в Java. “Java эмулирует свойства с использованием специальной схемы именования методов доступа, — говорит Хейльсберг, — а обработчики событий — с помощью различных “переходников” и связующего кода”. Он согласен с тем, что и Java, и Си++ также поддерживают компонентно-ориентированный стиль программирования, но считает, что “компоненты в них не обладают гражданством первого класса”. Некоторые отличия этих языков приведены в табл. 5.6.

Таблица 5.6.

Сравнение объектно-компонентно-ориентированных языков

Черта	**Язык C++**	**Язык Java**	**Язык C#**
Управление системными ресурсами	Выделение и высвобождение памяти вручную, с использованием специальных операторов	Автоматическая «сборка мусора»	
Производительность труда программистов	Небольшое число высокоуровневых механизмов	Хорошо структурированные определения структур данных и операторы передачи управления	Расширенный словарь предопределённых типов данных; Повышенная гибкость в передаче управления
Надёжность ПО	Ограниченные возможности контроля типов; неудобные и часто игнорируемые программами механизмы обработки ошибок	Жёсткий контроль типов позволяет избежать ошибок с непредусмотренным переопределением операций; механизмы обработки исключительных ситуаций позволяют обрабатывать ошибки только структурированным образом	
Производительность приложений	Свобода использования указателей на данные позволяет рационализировать многие операции, но в то же время создаёт возможности для многочисленных программистских ошибок; ограниченный набор механизмов, характерных для объектно-ориентированных систем, сводит к минимуму дополнительные вычислительные расходы, связанные с их реализацией	Указатели не используются; процедуры преобразования типа между объектами и примитивными типами бывают порой довольно неуклюжи	Использование указателей возможно, но только в коде, помеченном специальным ярлыком “unsafe”; эффективное преобразование типов объектов и значений с использованием механизма «боксов»
Переносимость приложений	Компиляция в платформенно-независимый код	Компиляция в универсальный байт-код, пригодный для использования на любой платформе, оснащённой виртуальной Java-машиной	Компиляция в код .Net intermediate Language

Следует добавить, что в рамках концепции создания всемирной архитектуры использования Web–сервисов – .Net (дот нет), которую развивает Microsoft, этой корпорацией реализовано целое семейство продуктов разработки с соответствующими языками: Visual Basic .NET 2003, Visual C# .NET 2003, Visual C++ .NET 2003, Visual J# .NET 2003 и планируется продолжение этих работ.

5.3. Некоторые возможные сравнения

Вместе с тем, до сих пор специалистами дебатируются вопросы эффективности того или иного языка программирования с точки зрения глобальности и широты спектра решаемых им проблем,. На этом фоне Международная Комиссия в составе членов комитетов по образованию организаций IEEE-CS[2] и ACM[3] пришли к мнению, что специалистам в области информатики на начальном этапе ***необходимо знание не менее двух языков***.

Понятно, что обычно осваивать программирование и, соответственно, сам язык, начинают с более простого. Чем же может «измеряться» сложность языка? Кроме стилевых и концептуальных особенностей, связанных с областью его применения, в большинстве языков существует т.н. лексемный уровень. Лексемы представляют минимальные единицы языка, имеющие значение в его структуре: идентификаторы, буквенные константы, знаки операций (операторы) и разделители. В языках процедурной направленности большое значение имеют операции работы с данными: арифметические (+, –, *, /), сравнения, логические и ряд других. В некоторых языках их именуют операторами или функциями[4]. В сложных выражениях с их участием необходимо учитывать приоритет каждой из операций. К примеру, в операторе присваивания языка Турбо Паскаль `R := 20–8/4`, сначала будет выполнена операция деления (/), имеющая более высокий приоритет, а затем результат будет вычтен из числа 20.

В таблице 5.7 приведены соотношения количества операций и уровней их приоритета в некоторых популярных языках программирования.

Из таблицы следует, что наиболее «лёгкими», то есть теми, которые имеют относительно небольшое количество операций и уровней их приоритета, а так же с точки зрения простоты освоения возможностей языка, являются Turbo Pascal и Visual Basic. Однако, с точки зрения объёмов средств поддержки, которые включают описание языка и непосредственно саму среду разработки IDE, Turbo Pascal остаётся непревзойдённым до сих пор.

Кроме того, в основе любого языка программирования лежат принципы организации структур данных и, как правило, некоторый набор инструкций

---

[2] Institute of Electrical and Electronics Engineers-Computer Science – Институт инженеров по электротехнике и электронике-компьютерные науки.

[3] Association for Computing Machinery – Ассоциация по вычислительной технике.

[4] Существует различие в названиях элементов языков программирования. Так, управляющие конструкции (типа If-Then-Else, Do, While и др.) к примеру, в языке Turbo Pascal именуются ОПЕРАТОРАМИ, а в Visual Basic – ИНСТРУКЦИЯМИ. Соответственно, операции работы с данными (+, - и т.д.) – в языке Turbo Pascal называются ОПЕРАЦИЯМИ, а в VB – ОПЕРАТОРАМИ.

языка. Усвоив принципы их использования на базе языка Turbo Pascal, можно использовать эти знания для изучения других языков, которые непосредственно относятся к сфере профессиональной деятельности.

Таблица 5.7.

Сравнительная таблица некоторых характеристик языков программирования

Название языка программирования (ЯП)	Количество операций/ операторов (типа +, –, *, /, >, < и т.д.)	Количество уровней приоритета операций/ операторов
Turbo Pascal	20	4
Visual Basic (VB)	14	4
VBA (Visual Basic For Application)	23	4
VBScript	23	9
Java	44	10
Java Script	48	14
C++	52	18
Perl	64	17

Уже давно среди специалистов продолжается полемика по поводу сравнения языков Turbo Pascal и C++ (который по некоторым данным входит в число трёх наиболее популярных на мировом уровне языков программирования вместе с VB и Java). Поэтому представляет интерес отметить некоторые особенности языка C++. Существенная направленность C++ на работу с аппаратурой и компонентами персональных компьютеров обеспечивается 52-ю операторами работы с данными, которые объединены в 18 групп разного приоритета. Сюда, кстати, входят постфиксные и префиксные инкременты (++) и декременты (--), операторы присваивания: с умножением (*=), делением (/=), делением по модулю (%=), суммой (+=), разностью (-=), сдвигом влево (<<=) и сдвигом вправо (>>=). Сюда же входят побитовые операции: И (&), ИЛИ (|), побитовое исключающее ИЛИ (^), логическое И (&&) и ИЛИ (||), тернарная операция (?) и многие другие. В то же время, в языке Turbo Pascal 7.0, используется всего 20 операций работы с данными, которые объединены в 4 группы приоритетностей выполнения, что всего в ***четыре раза меньше*** количества приоритетов в языке C++ (!!!).

Как известно, в алгоритмических языках ***программа*** – это "***алгоритмы + структуры данных***".

Объектно-ориентированные (ОО) языки представляют собой среду, в которой программисты определяют не только типы структур данных, но и типы операторов (функций), которые применяются к этим структурам данных. В таком контексте образуются ***объекты***, включающие и ***данные***, и ***функции*** (т.н. ***методы***). Функционально подобные объекты образуют ***классы***. В рамках создаваемой информационной системы программисты должны спроектировать связи между моделируемыми объектами с учётом наследуемых из классов свойств.

Чрезвычайно важным представляется также тот факт, что язык C++ является существенно ***объектно-ориентированным*** (ОО) и изначально предполагает у изучающего чёткое представление об ОО парадигме,

включающей следующие фундаментальные понятия: ***ОБЪЕКТ, СООБЩЕНИЕ, КЛАСС, НАСЛЕДОВАНИЕ И МЕТОД***. В контексте языка постоянно и широко используются принципы, лежащие в основе объектной модели представления программно моделируемых систем: абстрагирование, инкапсуляция, полиморфизм, модульность, иерархичность, типизация, параллелизм и сохраняемость. Механизмами реализации указанных абстракций являются виртуальные и не виртуальные функции, перегружаемые функции и методы, функции-члены базовых классов и их объекты, потоки, буфера и их классы, шаблоны, объявляющие параметризованные классы массивов, классы и экземпляры шаблонов и т.д. Другими словами, алгоритмическая составляющая в этом языке служит основой реализации методов объектов, при весьма высокой степени абстракции представления элементов иерархической структуры организации взаимодействия абстрактных данных в виде базовых и виртуальных классов реализуемых систем и порождаемых ими объектов. Всё это делает язык C++ достаточно сложным для освоения.

Вместе с тем, расширение горизонтов применения компьютеров и совершенствование возможностей языков программирования требует освоения всё новых и новых не только языковых инструментов, но и их диалектов.

Можно назвать как минимум три десятка языков, которые сыграли заметную роль в развитии программирования, но все же именно три пары — Алгол-60 и Фортран, Паскаль и Си, Java и Си++ — стали самыми яркими, самыми заметными на компьютерном небосклоне.

Говоря об отстраненном и предвзятом отношении людей к «чужим» языкам, Никлаус Вирт, автор языка Паскаль отмечал: «Многие относятся к стилям и языкам программирования, как к религиозным конфессиям: если вы принадлежите к одной из них, то не можете принадлежать к другой. Но это ложная аналогия, и она сознательно поддерживается по причинам коммерческого порядка».

Оппонент Никлауса Вирта — Деннис Ритчи (автор языка C) — недавно отметил: «Паскаль — очень элегантный язык. Он по-прежнему жив. Он породил немало своих последователей и оказал глубокое воздействие на проектирование других языков».

В интервью, данном 21.02.2001 системному аналитику Денни Калеву (Danny Kalev), создатель одного из наиболее широко используемых языков C++, Бьерн Страуструп сказал следующее:

«Многое в программировании лучше сделать методами, которые не помещаются внутри узкой полоски методов, именуемых "объектно-ориентированными". И если Вы не выходите за границу "объектно-ориентированных" методов, чтобы остаться в рамках "хорошего программирования и проектирования", Вы получаете нечто, что является в основном бессмысленным. Будущее за ***мультипарадигматическим программированием***».

Далее Бьерн Страуструп добавил:

«… я думаю, что любой язык, который стремится к господствующему положению во всех областях, должен обеспечить широкую основу для

нескольких методов, включая объектно-ориентированное программирование (на основе иерархии классов) и обобщенное программирование (параметризированные типы и алгоритмы). В частности, необходимо обеспечить хорошие средства для создания программ из отдельных (независимых) частей (возможно, написанных на различных языках). Я также думаю, что для управления сложным процессом обработки ошибок необходимы исключения. Язык, в котором отсутствуют такие средства, вынуждает его пользователей моделировать их (что ведет к дополнительным ошибкам и затратам)».

Не удивительно, что продолжают жить многие старые добрые языки Fortran, COBOL, Ada и другие. В частности, в июне 2001 года фирма COMPAQ выпустила на рынок интегрированную визуальную среду разработки с оптимизирующим компилятором Compaq Visual Fortran 6.6 (рис. 5.5).

Рис. 5.5 CD с компилятором Compaq Visual Fortran 6.6

В проведённом в 2001 году исследовании[5] была сделана оценка комплекса компонентов программного обеспечения, которое относится к так называемым "открытым кодам" (Open Source) и размещено на европейском сайте «The Site for Libre Software Developers [http://libre.act-europe.fr/]. Оказалось, что в комплексе исследуемых программных разработок, занимающих в запакованном виде на диске пространство более чем 1 гигабайт и разрабатываемых с участием 5300 программистов из 41-ой страны, количество разработчиков (кодеров), использовавших те или иные языки программирования, разделились следующим образом (рис. 5.6).

Из полученных данных вытекает, что большинство профессиональных программистов успешно владеют языками C и C++, но продолжают использовать в своей работе не только язык Паскаль, но и многие другие.

Это и неудивительно, ведь язык Паскаль, созданный в 1968 г. Никлаусом Виртом, разрабатывался изначально для обучения написанию надёжных программ с развитой системой данных и является учебным языком высокого уровня. Для него в 1982 году была создана уникальная

[5] Who Is Doing It? A research on Libre Software developers. Fachgebiet für Informatik und Gesellschaft TU-Berlin, August 2001 / Robles G., Scheider H., Tretkowski I, Weber N. WEB-сайт (Электронн. ресурс) / Способ доступа: URL widi.berlios.de/paper/ study.html

интегрированная среда разработки Turbo Pasca, конечным этапом развития которой явилась версия Turbo Pascal 7.0 фирмы Borland International.

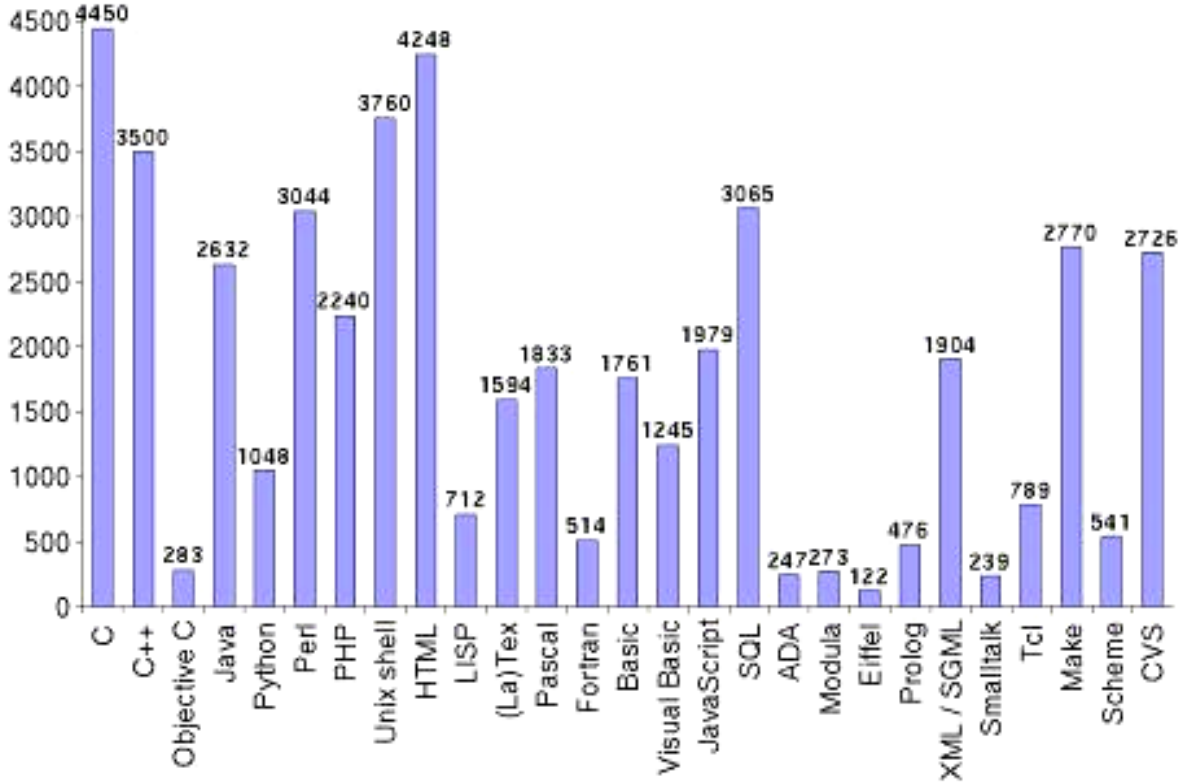


Рис. 5.6. Численное распределение количества программистов по типам используемых ими языков программирования (большинство программистов применяли в своих разработках по несколько разных языков)

Эта интегрированная среда разработки:

❶ Имеет встроенный редактор, компилятор, средства запуска программ на счёт, отладчик, позволяющий не только отлаживать программы, но и ***исследовать*** их работу.

❷ Позволяет систематически и точно выражать концепции и структуры элементов программирования.

❸ Показывает, что использование машинно-независимого языка программирования с гибкими структурами данных и управляющими конструкциями приводит к повышению "удобочитаемости", верифицируемости[6] и, соответственно, надёжности без потери эффективности.

❹ Turbo Pascal 7.0 способствует улучшению понимания методов организации больших программ и методов управления программистскими проектами.

❺ Имеет развитые средства диагностики ошибок, эффективные инструменты отладки и по этим причинам является весьма удобным средством для обучению программированию.

[6] Верификация – процесс подтверждения выполнения программой заложенных в неё функций, т.е. проверка правильности программы путём формального доказательства соответствия программы заданной спецификации.

❻ Включает библиотеку Turbo Vision с мощными средствами объектно-ориентированного программирования.

❼ Имеет компактные размеры интегрированной среды разработки и документации с описанием её работы и собственно конструкций языка Borland Паскаль.

Поскольку язык Паскаль фактически реализует виртуальную машину высокого уровня (Паскаль-машину) с усиленным языковым ограничением, это затрудняет пользователю выход за её пределы, что в процессе обучения и является важным положительным моментом.

На рисунке 5.7 представлена пространственно-временная диаграмма развития языка Паскаль в структуре остальных наиболее значимых языков программирования.

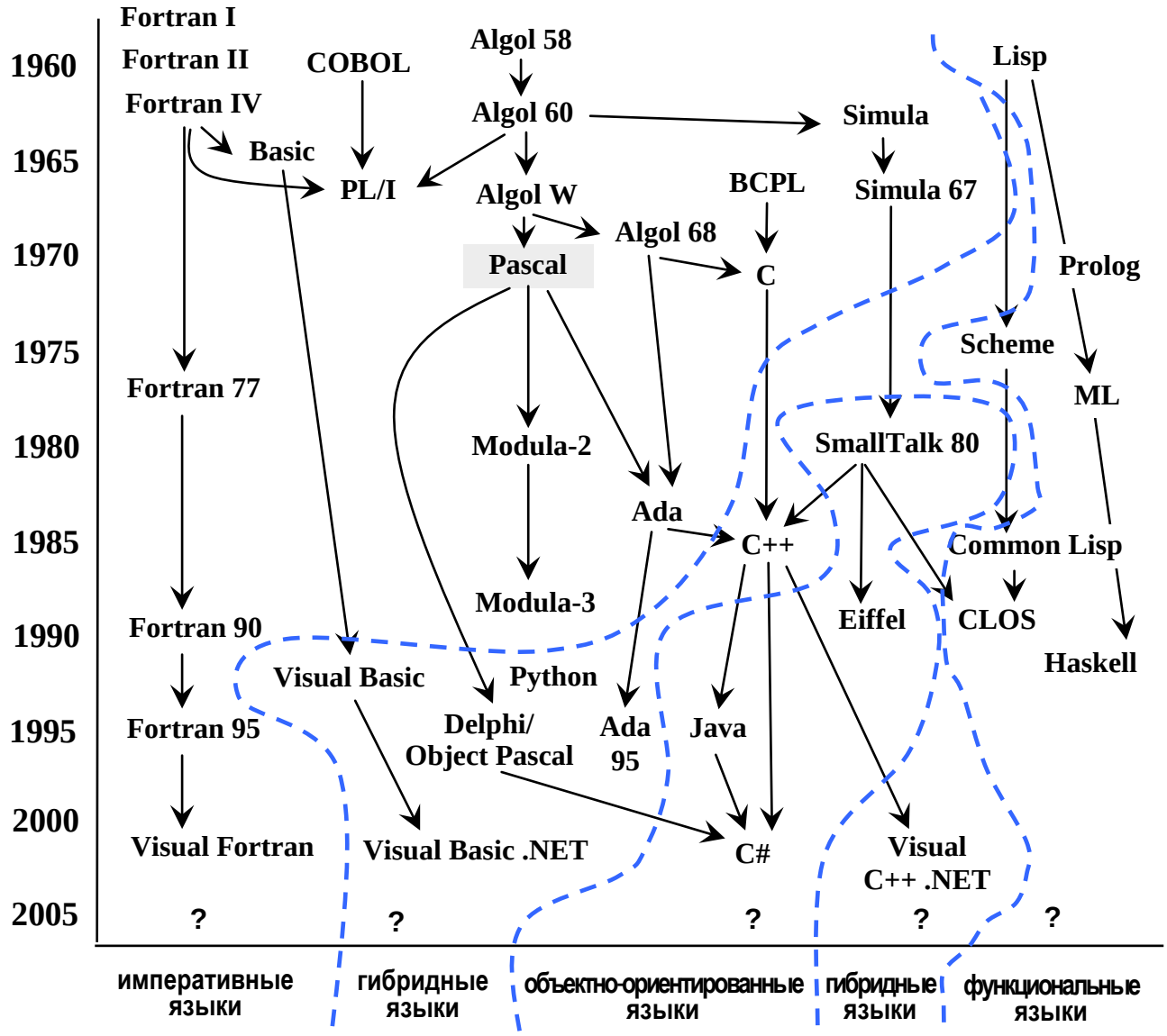


Рис. 5.7. Представление процесса развития наиболее значимых языков программирования

Если смотреть на языки программирования, как на инструмент обработки данных, все они имеют следующие ***основные базовые элементы:***

❶ базовые операторы описания данных и их структур;

❷ базовые инструкции, с помощью которых реализуются процедурные компоненты программ;

❸ базовые операции (операторы): арифметические, логические и др.

❹ уровни приоритета их выполнения.

Поэтому, можно утверждать, что при знании принципов употребления и взаимодействия базовых элементов, хотя бы одного языка, можно значительно облегчить освоение практически любого другого.

Вопросы.

1.	Что вообще называется языком?
2.	Чем отличается язык человека от языка программирования?
3.	Для чего необходимо компилировать или интерпретировать программы?
4.	Чем отличается процесс и результат компиляции от процесса и результата интерпретации?
5.	Чем отличается язык программирования системы от языка сценария?
6.	Какие языки программирования сейчас наиболее распространены?
7.	Какие основные стили программирования применяются при разработке программ?
8.	Как связаны языки программированння с типами задач, которые требуется решить?
9.	Что определяют приоритеты выполнения операций в языках программирования и как они влияют на сложность организации программ?
10.	Какие общие базовые элементы имеют практически все языки программирования?
11	Чем отличается выполнение откомпилированной программы от выполнения интерпретируемой?

Разработка решения бизнес-проблемы — это больше, чем простое написание крутого приложения с применением новейших технологий. Разработка приложения должна вестись, исходя из бизнес-требований. Будучи разработанным, приложение должно быть принято конечными пользователями, которым оно обязано помогать в решении повседневных проблем.

Дональд Р. Брандт. "Архитектура. Экзамен Microsoft – экстерном".

6. ИЗМЕНЕНИЯ В МЕТОДОЛОГИИ СОЗДАНИЯ ПРОГРАММ

6.1. Тенденции развития информационно-компьютерных технологий

Многие изменения, влияющие на информатику, связаны с прогрессом в развитии компьютерных (электронных) технологий (КТ) и неразрывно связанных с ними информационных технологий (ИТ, а в английской транслитерации, IT – Information Technologies), что в комплексе именуется информационно-компьютерными технологиями (ИКТ). Большинство постоянно растущих достижений в этой области представляют собой часть постоянно эволюционного процесса, который длится уже многие годы. Закон Мура (прогноз, сделанный в 1965 году создателем фирмы Intel Гордоном Муром, и гласящий, что плотность транзисторов на кристалле микропроцессора и частота его работы будут удваиваться каждые восемнадцать месяцев) до сих пор остаётся в силе. В результате можно наблюдать экспоненциальный рост вычислительных возможностей, благодаря которым стало возможным решение задач, которые казались неразрешимыми всего лишь ***несколько ближайших лет назад.*** Следует также добавить, что емкость дисковой памяти компьютеров удваивается еще быстрее – каждые 9 месяцев. И совсем фантастическими темпами увеличивается пропускная способность линий связи - в два раза каждые 4-6 месяцев. Только в Китае, в секторе телекоммуникаций совокупный объем инвестиций в эту сферу, согласно статистическому отчету китайского правительства, составил в 2003 году $30,1 млрд. Сейчас Китай занимает второе место после США в мире по размеру своих коммуникационных сетей.

В настоящее время практически для всех организаций характерно критическое нарастание объемов информации. Рост объема данных в информационной системе ставит перед IT-персоналом задачу построения современной системы хранения данных. По сведениям компании Meta Group (Стэмфорд, шт. Коннектикут, США) в 2002 году, объем аккумулируемой в любой организации США информации удваивается каждые 18 месяцев. По данным International Data Corp., плата за поддержку неизбежно возрастающих

объемов данных составляет сегодня приблизительно половину всех расходов на IT-решения. Столь дорогостоящие способы обеспечения постоянного доступа к информации предпринимаются, для того чтобы избежать еще больших расходов, связанных с потерями доступа к хранилищам информации (до нескольких миллионов долларов в час в зависимости от приложения). Эти потери складываются из суммы ущерба от снижения эффективности труда сотрудников из-за простоя системы хранения данных, стоимости работы, которая не может быть выполнена в период простоя, а также стоимости ремонта вышедших из строя элементов системы.

Область информатики, в которую вкладываются огромные средства и которая приносит, в свою очередь, невероятные доходы, постоянно находится под пристальным вниманием ведущих научных и государственных организаций практически всех развитых стран мира (рис. 6.1, здесь и далее данные с сервера: http://www.nsda.ru/home.asp).

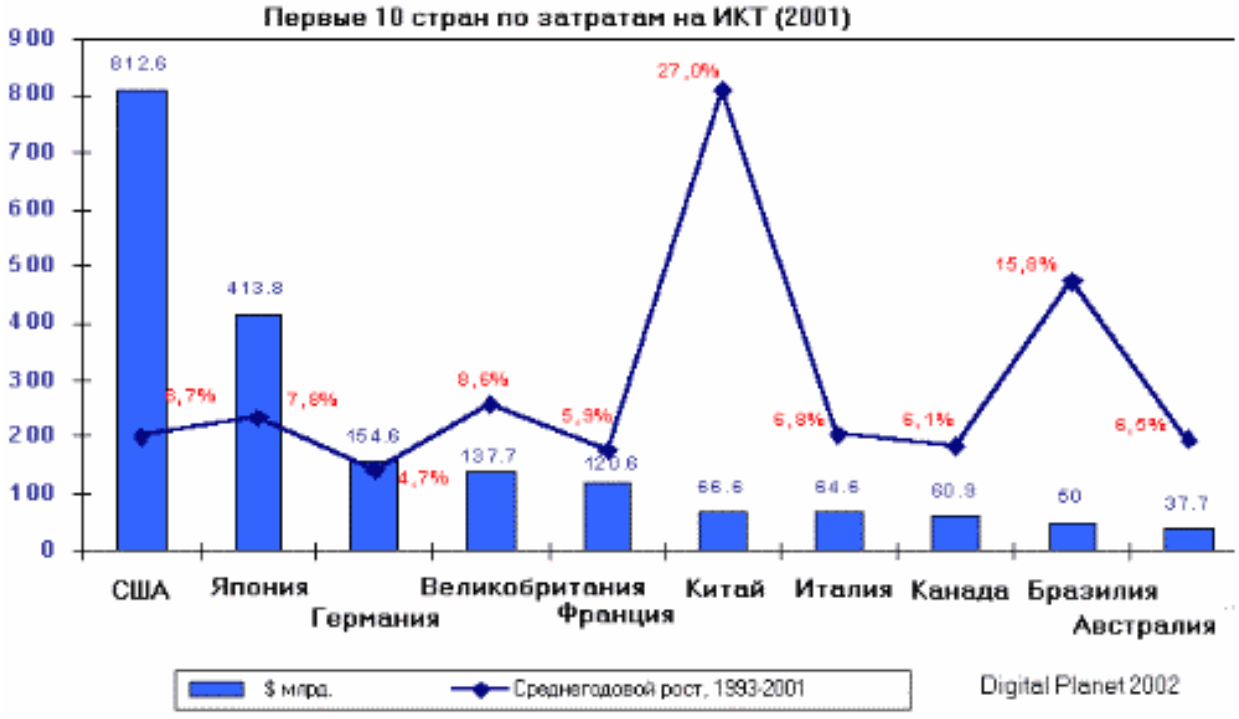


Рис. 6.1. Сравнение вкладываемых передовыми странами мира средств в ИКТ отрасль по данным на 2002 год

К основным факторам, которые определяют развитие отрасли ИКТ, можно отнести следующие:

❶ активное внедрение ИКТ во все структуры и отрасли материального производства в последнее время является стратегичеким направлением, который поддерживается правительствами всех развитых стран мира;

❷ уровень сложности проблем, которые решаются компьютерными средствами постоянно повышается;

❸ количество специалистов, которые владеют необходимым комплексом знаний по информатики значительно меньше необходимого (поэтому только в 2002 г. в ряде развитых государств были дополнительно

выделены десятки и сотни тысяч так называемых "зелёных карт" для въезда в страну " продвинутых" специалистов в области информационных технологий;

❹ постоянnно расширяется спектр новых и "новейших" технологий и, соответственно, новых отраслей применения ИКТ.

Уже давно не секрет, что приоритеты и уровень развития компьютерной индустрии определяет группа государств, возглавляемая США. Красноречивее всего об этом говорят цифры. На конец 2002 г. географическое распределение доходов рынка информационных технологий выглядело так: США –46%, Европа – 29%, Япония – 13%, Азия – 4%; Канада –3%, остальные страны – 5% (рис. 6.2).

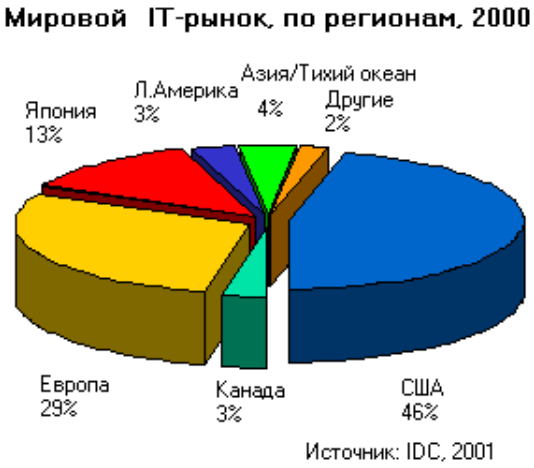


Рис. 6.2. Развитие мирового рынка ИТ по регионам мира в 2001 г.

В непрекращающейся гонке за информационное «господство» выигрывают вкладываемые в отрасль капиталы. Неудивительно, что среди «гегемонов» – производителей базового компонента информационных систем (ИС) – микропроцессоров, из нескольких десятков стартовавших в 70-е годы лидерами остались фактически двое: Intel и AMD (США), которые вывели их рабочие частоты в 2003 году за отметку 3 Ггц. Рынком операционных систем и офисных приложений в 3002 г. монопольно владела Microsoft (США) (по данным 2000 г. соответственно 90% и 86%), а среди разработчиков баз данных – Oracle (США). Это и неудивительно, по экспертным оценкам 2003 года, в российской отрасли информационно-компьютерных технологий (ИКТ) работало около 0,54 млн. человек, а в США – около 10,5 миллионов. Следует отметить, что в это число, естественно, не вошли те специалисты ИТ, которые за деньги США выполняют их заказы на создание программного обеспечения на своей родине: в Индии[5], Ирландии[6], Украине и других странах мира. По

[5] В 2003 г. Индия входила в первую десятку стран мира по ряду показателей производства. По объему ВВП она уже на 4-м месте в мире после США, КНР и Японии. В этом же году в Индии было выпущено около 2,5 млн. ПК, 10,5 млн. телевизоров, 20 млн. радиоприемников и магнитол, 35 млн. электронных часов, 1 млн. видеомагнитофонов. Страна занимает 3-е в мире место по численности научно-технического персонала и второе - по количеству профессионалов компьютерного программирования.

[6] Одной из причин 10-процентного экономического роста в Ирландии в 1999 году явилось то, что она превратилась в настоящую "Силиконовую долину" Европы. Здесь расположилась европейская штаб-квартира Microsoft. Фирма Intel изготавливает здесь свои процессоры Pentium, а Apple, IBM и Dell ведут здесь научные исследования. По производству программного обеспечения Ирландия вышла на 2-е место в мире после США. Исследовательские центры Ирландии участвуют в разработке новых микрочипов и языков программирования.

оценкам международной исследовательской компании International Data Corporation (IDC), объем мирового рынка информационных технологий (IT-рынок) в 2000 г. достиг 395 млрд. дол., а в 2001 г. - 440 млрд. дол. и будет продолжать расти (см. рис. 6.3).

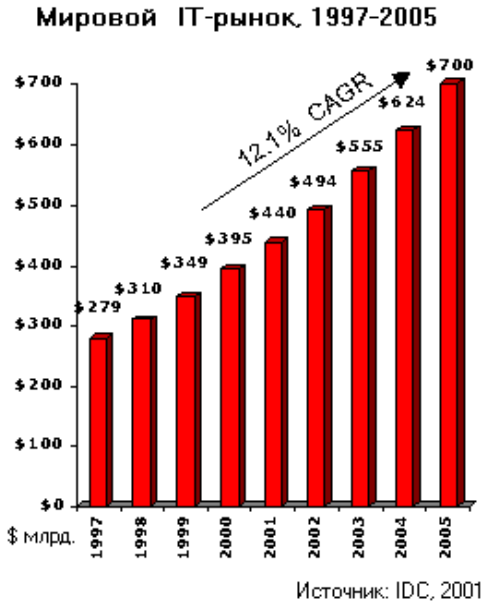


Рис. 6.3. Прогноз развития мирового рынка ИТ во времени (2001 г.)

Рост совокупного дохода ИКТ индустрии ещё более впечатляющ (рис. 6.4). Данные на 2003 г. 2680 млрд. Евро является прогнозом.

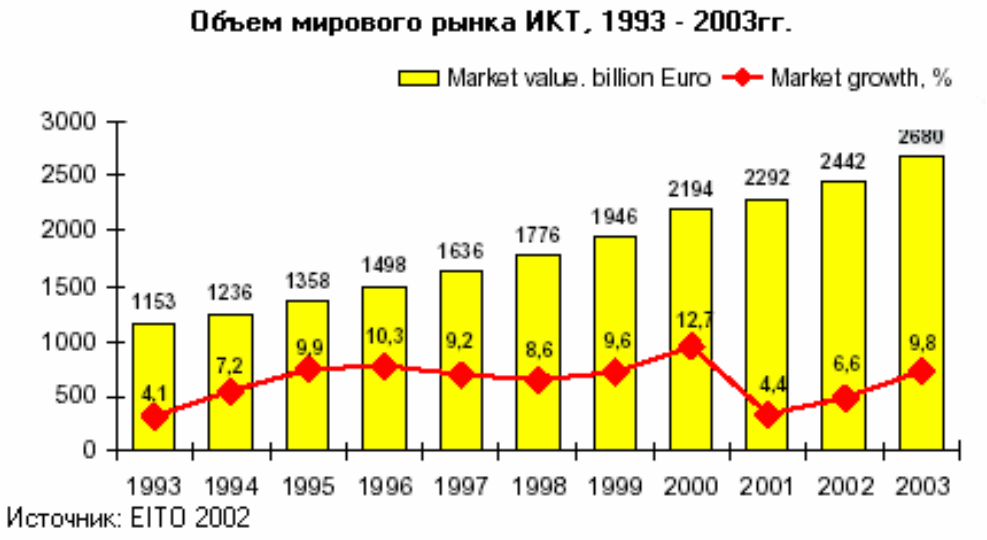


Рис. 6.4. Данные по развитию мирового рынка ИКТ на 2002 г.,(в млрд. Евро)

Международная исследовательская корпорация IDG/IDC для определения уровня развития высокотехнологичной отрасли вычисляет сводный индекс ISI (Information Society Index, сводный индекс развития информационного общества) на основании ряда показателей:

❶ количества персональных компьютеров на душу населения;

❷ уровня использования ПК в промышленности, государственных органах и образовании;

❸ стоимости программного обеспечения на компьютер;

❹ доли ПК, подключенных к Интернету; пропускной способности и стоимости каналов связи.

Согласно этим показателям, сводный индекс развития и ВВП на душу населения некоторых стран в 2003 году выглядел следующим образом (таблица 6.1):

Таблица 6.1.

Сводные индексы ISI и ВВП на душу населения по странам мира в 2003 году.

Страна	ISI	ВВП	Страна	ISI	ВВП
Швеция	5062	20,7	**Польша**	1808	7,2
США	5041	**33,9**	**Чили**	1635	12,4
Япония	4093	**23,4**	**Аргентина**	1651	10
Сингапур	4014	**27,8**	**Малайзия**	1583	10,7
Германия	3558	22,7	**ЮАР**	1537	6,9
Тайвань	3177	16,1	**Венесуэлла**	1491	8
Ирландия	3144	20,3	**Россия**	1444	4,2
Израиль	3140	18,3	**Филиппины**	1012	3,6
Франция	3140	23,3	**Китай**	915	3,8
Ю.Корея	2931	13,3	**Индонезия**	888	2,8
Италия	2703	21,4	**Индия**	871	1,8
Чехия	2130	11,7	**Пакистан**	719	2

Интересно отметить, что, первые три места по ВВП на душу населения занимают США, Сингапур и Япония. Неудивительно, что в недавно опубликованном в США (2003 г.) отчете под названием "The Digital Work Force: Building InfoTech Skills at the Speed of Innovation", сказано, что потребность в США в работниках со знаниями информационных технологий более чем удвоится к 2006 году, в то время как потребность во всех других специальностях возрастет в среднем только на 14%.

Согласно данным многочисленных зарубежных и отечественных источников главную роль во всех компьютерных направлениях (см. Введение) играет у первую очередь ***программирование***.

На смену программированию процедурному, с которого и начиналось программирование как таковое, пришло программирование объектно-ориентированное. Объектно-ориентированный метод имеет столь существенные преимущества, что подавляющее большинство создаваемого программного обеспечения использует преимущества объектных технологий в той или иной форме. В некоторых языках программирования имеется возможность создания объектов разными способами: либо с "нуля", либо используя механизм наследования и объекты заготовки – классы. Языки сценариев применяются для работы с готовыми объектами: объектам можно посылать сообщения и получать от них реакцию в виде выполняемых методов либо данных и т.д

Вместе с тем, продолжает оставаться открытым и наиболее обсуждаемым на всех уровнях специалистов главный вопрос ИКТ отрасли:

Какой же язык наиболее универсален для написания программ и, соответственно, какой же язык в первую очередь следует изучать?

Чтобы попытаться объективно подойти к ответу на данный вопрос, вначале следует ответить на несколько следующих вопросов:

❶ Что собой представляет программа?
❷ В какой среде пишутся программы и приложения?
❸ В какой среде работают программы и приложения?
❹ Как проектируются приложения?
❺ Какие существуют приложения?

6.2. Что собой представляет программа?

Для ответа на этот вопрос реализуем на языке Турбо Паскаль стандартный вывод знакомого всем программистам текста "Hallo, World!" на дисплей компьютера (рис. 6.5).

```
Program Hallo;
  Begin
      Writeln('Hallo, World');
  End.
```

Рис. 6.5. Вывод на дисплей ПК текста "Hallo, World!" в языке ТП

Фрагмент текста, который приведен на рисунке 6.5 и который состоит из четырёх инструкций (операторов или предложений) языка ТП, с точки зрения современных информационных представлений можно ***одновременно*** назвать следующим образом (то есть рассматривать с таких точек зрения) (табл. 6.2).

Таблица 6.2.

Наименования набора инструкций в языках программирования

№ пп	Наименования набора инструкций языка программирования и его сущность
1	***Инструкции***, это ***данные***, предназначенные для управления конкретными компонентами системы обработки данных в целях реализации некоторого алгоритма.
2	***Код*** – описание действий, которые должен выполнять компьютер, записанные на языке низкоуровневого программирования или в машинных кодах.
3	***Программный код*** или просто ***код*** – последовательность команд или операторов любого языка программирования, которая после декодирования её компьютером и транслирующей программой, заставляет последний выполнить некоторую работу.
4	***Фрагмент кода*** – то же самое, что и ***код.***
5	***Программа –*** описание операций, которые необходимо выполнить для решения поставленной задачи. Действия, которые описываются программой, называются ***операторами***. ***Командой*** именуют элементарное предписание, которое предусматривает выполнение какой-нибудь ***операции***.

Продолжение таблицы 6.2.

№ пп.	Наименования набора инструкций языка программирования и его сущность
6	***Последовотельность операторов*** – упорядоченная последовательность команд, которая подлежит обработке.
7	***Модуль*** – программа, которая рассматривается как отдельное целое в процессах хранения, трансляции и объединения с другими программными модулями при их загрузке в оперативную память компьютера для выполения.
8	***Подпрограмма*** – поименованная часть программы, которая вызывается и получает параметры, ввыполнят определённые действий и возвращает результат своей работы и управление в точку вызова. Во многих языках программирования различают два вида подпрограмм: –***процедуры***, действие которых заключается в изменении значений параметров и некотором побочном эффекте. Обычно являются ***операторами*** или ***инструкциями*** языка программирования; –***функции***, которые возвращают зависящий от параметров результат. Являются ***операндами*** в конструкциях языка программирования и описываемых с их помощью выражениях.
9	***Процедура-подпрограмма, процедура-функция (в языке VBA)*** – соответственно подпрограмма и функция. Подпрограмма в VBA может выполнять роль программы.
10	***Метод*** – это ***процедура*** или просто ***набор команд***, сообщающих объекту, что необходимо выполнить некоторую задачу и реализуют алгоритм её выполнения.
11	***Приложение*** – прикладная программа, то есть программа, которая выполняется под управлением операционной системы.
12	***Решение (термин из Microsoft Solution Framework (MSF))*** 1)Программный продукт. 2)Координированная поставка набора элементов (таких, как программно-технические средства, документация, обучение и сопровождение), необходимых для удовлетворения некоторого бизнес–требования конкретного заказчика (MSF). 3)Множество файлов разного типа, в рамках одного ***решения*** *(solution)* составляют ***приложение*** *(application)* среды разработки Visual Studio.Net 7.0.
13	***Операция (operation)***-сервис или услуга, которые могут быть заказаны у объекта в виде результата его поведения. Операция имеет ***сигнатуру***, которая может иметь параметры (UML). ***Сигнатура (signature)*** - имя и параметры поведенческого свойства. Сигнатура может включать опционально возвращаемые параметры. ***Метод (method)***- реализация операции. Она специфицирует алгоритм или процедуру, которые связаны с операцией (UML).

Первый тезис, что программа это данные, подтверждается тем простым фактом, что ***все без исключения оттранслированные программы хранятся на магнитных дисках или в базах данных (тоже на дисках !)*** и начинают выполнять свою работу под управлением операционной системы только на командный стимул и, как правило, после загрузки в оперативную память компьютера (рис. 6.6).

Более того, текстовые строки программного кода являются ***входными данными*** для трансляторов (компиляторов) и интерпретаторов, преобразующих их в конкретные машинные команды, выполняемые микропроцессором.

Кроме того, просмотрев таблицу 6.2. сверху вниз, можно увидеть насколько многогранным является представление ***программной компоненты*** в виде ***кода*** на любом из ***языков программирования***.

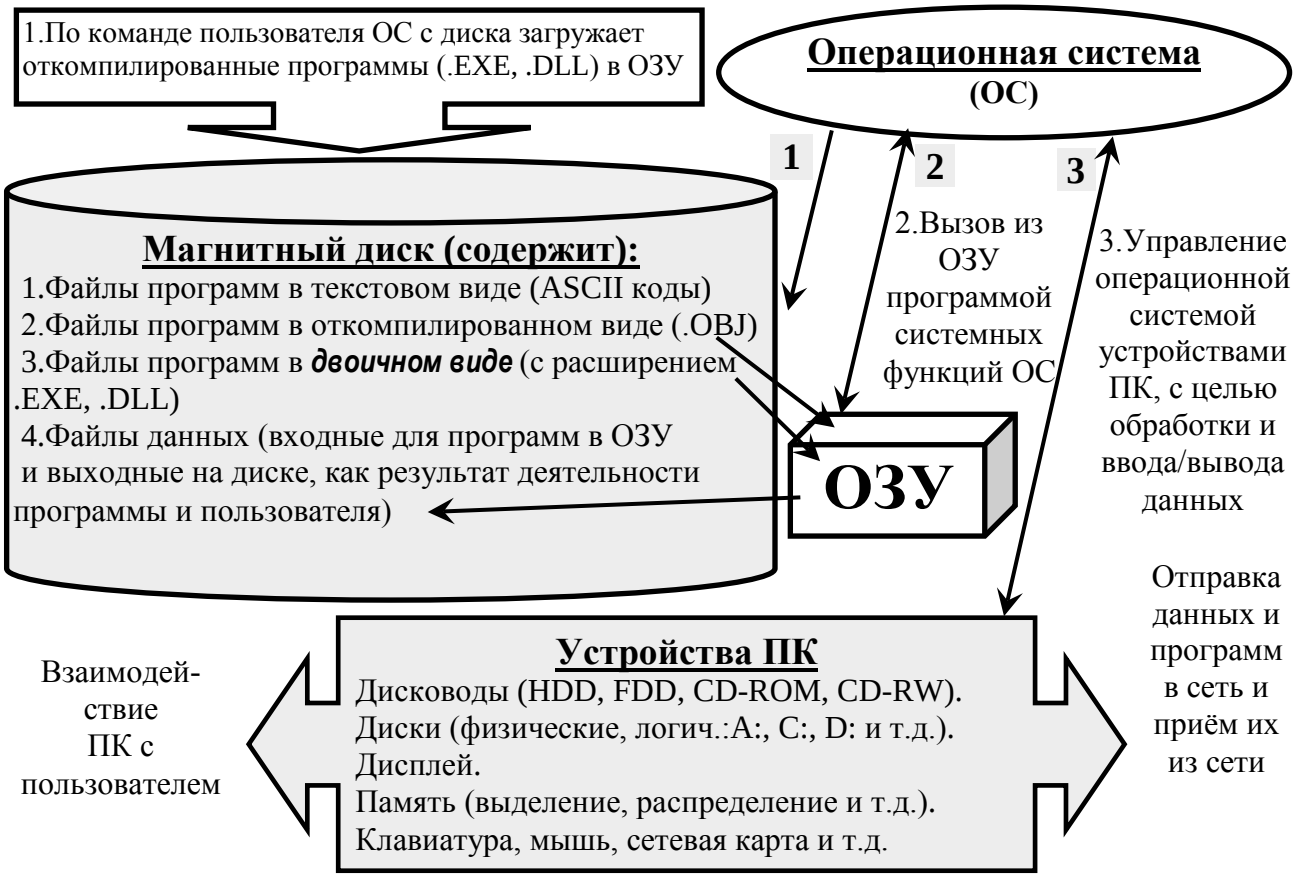


Рис. 6.6. Типовой процесс выполнения программы на ПК

В большинстве современных программных систем инструкции исходных программ записываются в восьмибитных ASCII ("аски") кодах. И поэтому, вообще говоря, коды программ на многих языках программирования можно набирать в любых текстовых редакторах файл-менеджеров (Norton Commander, Far Manager, Total Commander и др.), либо редакторах операционных систем типа Блокнот (Notepad). То же касается и кодов языков разметки: HTML, XML и некоторых других.

6.3. В какой среде пишутся программы и приложения?

Если текст программы уже существует, каким образом можно подать этот текстовой фрагмент кода на вход транслятора?

Наиболее простыми в использовании для этих целей являются Интегрированные среды разработки (ИСР, а на английском языке – IDE – Integrated Development Environment), которые широко представлены продуктами фирмы Borland International (Turbo Pascal, Turbo C, Turbo Prolog), Microsoft Visual Basic, PowerBuilder, Inprise Delphi Object Pascal, Oracle Developer PL/SQL и некоторые другие, которые относятся к языкам 4-го поколения (4GL, Generation Language 4). Эти продукты являются

продолжением развития языков программирования (типа Fortran, Algol, C, C++, COBOL, Ada, Pascal), которые в прошлом не имели многих сервисных возможностей существующих ныне систем и поэтому возлагали на программистов выполнение всех действий по организации ввода текстов программ в компьютер, вызова транслятора, загрузку оттранслированных модулей в память ПК и т.д. Их называют языками 3-го поколения 3GL (Generation Language 3), потому что они не были встроены в интегрированные интерактивные среды разработки программного обеспечения.

Чтобы лучше уяснить принцип их работы, рассмотрим работу с типичным приложением. Обычно, пользователь, прежде всего, должен подготовить текст приложения (программы), подать его на вход транслятора, получить двоичный код и записать его на диск в файл с расширением EXE. Затем, считать двоичный код приложения в память компьютера, выполнить ту часть кода, что отвечает за организацию диалога с пользователем, который должен определить необходимый для работы набор данных. Затем приложение читает набор данных с диска (возможно удаленного). Далее пользователь взаимодействует с приложением, манипулируя представлением данных в основной памяти. Эта фаза продолжается бóльшую части времени работы приложения до тех пор, пока, в конце концов, модифицированный набор данных не запишется на диск.

Для организации такой технологии работы в ИСР включаются все инструменты, обеспечивающие осуществление всего описанного выше фронта работ. Поэтому прохождение решения типовой задачи, к примеру, в рамках ИСР Турбо Паскаль, можно представить следующим образом (рис. 6.7).

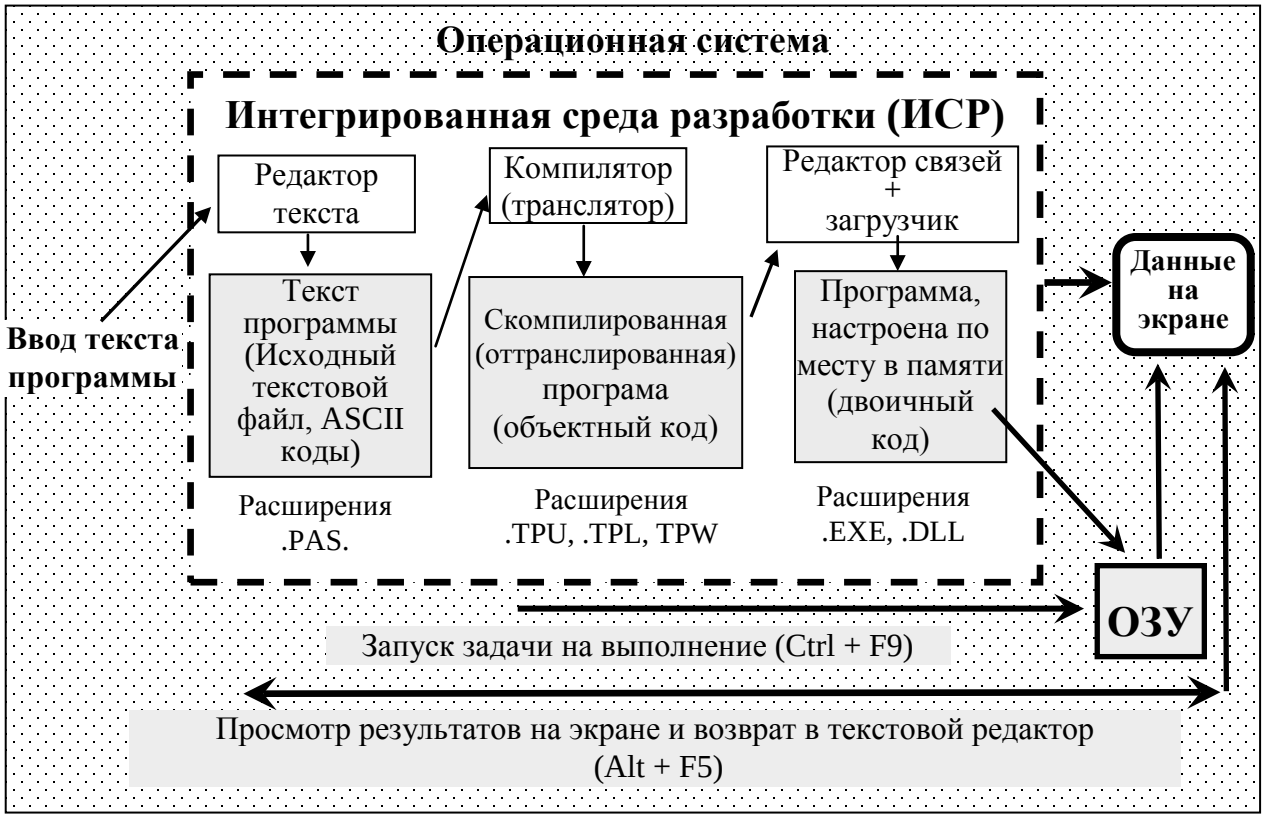


Рис. 6.7. Этапы выполнения программы в ИСР ТП

В ИСР интегрированы текстовый редактор для набора кодов программы, транслятор, редактор связей и загрузчик, поэтому все создаваемые файлы (PAS, TPU, EXE, DLL и другие), как правило, сохраняются на магнитном диске. Кроме того, в среде всегда присутствуют средства отладки программы (дебаггеры) и много других дополнительных сервисных возможностей. Поскольку разработка подобных средств велась до периода расцвета графических возможностей компьютеров, работа с ИСР активно поддерживается командами, представляемыми отдельными клавишами и их сочетаниями (Esc, Ctrl + F9, Alt + F5 и так далее). Однако, несмотря на то, что в режиме MS DOS из Турбо Паскаля можно напрямую работать с портами, видеопамятью, секторами и дорожками дисков и т.д., следует чётко представлять, что все эти действия поддерживаются системными функциями соответствующей операционной системы (MS DOS, Windows и др.).

Появление операционной системы Windows существенно усложнило программирование, так как с позиции программиста ОС Windows представляет для работы набор системных функций, образующих так называемый API–интерфейс прикладных программ (API–Application Programming Interface). Поэтому, чтобы выполнить то или иное действие (например, открыть на экране окно, вычертить линию, создать кнопку и др.), программа должна обратиться к соответствующей функции API.

Функции API операционной системы, которых только в ОС Windows 95 насчитывалось более 800-т, образуют её ядро и хранятся в многочисленных DLL-библиотеках. Среди многих других они реализуют следующие ***системные функции***:

❶ управление памятью, загрузка/удаление программ из памяти и обеспечение непосредственной поддержки выполнения программ;

❷ управление окнами (создание, изменение размеров, перемещение, удаление) и др.;

❸ управление вводом с клавиатуры, работа с мышью;

❹ взаимодействие с внешними устройствами (экраном, принтером и др.).

Усложнение взаимодействия с таким большим количеством системных функций привело к необходимости упрощения для пользователя самого процесса программирования и создание для этого интегрированных сред ***быстрой разработки приложений*** (RAD – Rapid Application Development) для визуального и быстрого программирования (создания Windows-приложений). Пионерами этого процесса выступили корпорация Microsoft, которая создала RAD среды Visual Basic и Visual Studio (для языков C++, Visual Basic и др.), а также Borland International (потом Inprise), которая создала среду Delphi[7] (с языком Object Pascal). Строительными блоками программы стали компоненты – объекты, имеющие визуальное представление на стадии проектирования и во время работы (рис.6.8).

[7] Автором Delphi в 1995 г. стал Андерс Хейльсберг (Anders Hejlsberg), соавтор разработки системы Турбо Паскаль, а затем в 2000 г. в статусе ведущего разработчика (distinguished engineer) корпорации Microsoft, ставший автором языка C# (С-шарп) и соавтором создания платформы .NET (Дот НЕТ) (http://www.microsoft.com/presspass/press/2001/apr01/04-11AndersPR.asp).

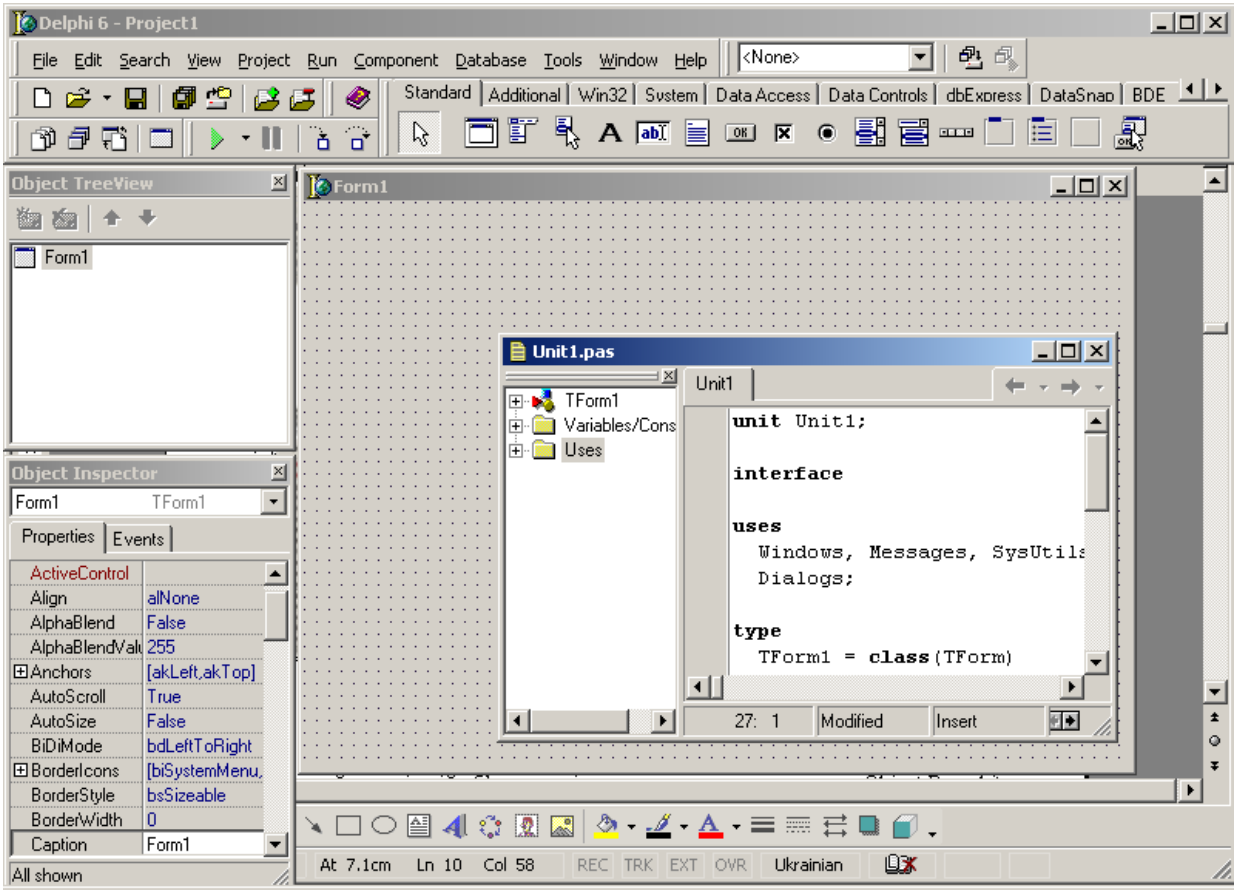


Рис. 6.8. Общий вид визуальной среды программирования Delphi 6. На заднем плане форма, на переднем – окно с кодом.
В левом нижнем углу – список свойств формы.

Популярность среде быстрой разработки приложений Delphi принесли мощный внутренний объектно-ориентированный язык программирования Object Pascal и уникальные по своей простоте и гибкости средства работы с базами данных. В дальнейшем, начиная с версии Delphi 5 и выше, появились возможности создания многофункциональных приложений (Windows, Internet, многоярусных (Multi-Tier) и др.), серверов COM и CORBA, а также отдельных компонентов. Компоненты, участвующие в разработке, представлены в виде иерархии классов, реализованных на языке Object Pascal, и организованы в Библиотеке визуальных компонентов (Visual Component Library, VCL), подключённой к Delphi IDE. Популярность Delphi среди разработчиков объясняется также и тем, что язык Object Pascal по возможностям превосходит язык Visual Basic, мало уступает языку C++, а усваивается в два-три раза быстрее последнего. Выполняемый код здесь получается достаточно эффективным, скорость компиляции очень высока, а отладка, особенно по сравнению с отладкой в C++, значительно проще и протекает быстрее.

Важнейшим элементом всех *RAD*, и Delphi в том числе, является удобная визуальная среда, которая обеспечивает ***двунаправленную разработку приложений из уже разработанных компонентов*** (т.е. изменения, сделанные в визуальной среде, отражаются на исходном коде программы, а изменения исходного кода отражаются в визуальной среде). При этом часть стандартных

фрагментов кода представляет RAD, а требуемые по ситуации коды программист дописывает самостоятельно.

Основным элементом практически любого приложения является ***форма*** (то есть окно произвольной формы), на которой размещаются все остальные визуальные компоненты, а также меню, кнопки и соответствующие команды, функциональность которых обеспечивает сам программист (рис. 6.9).

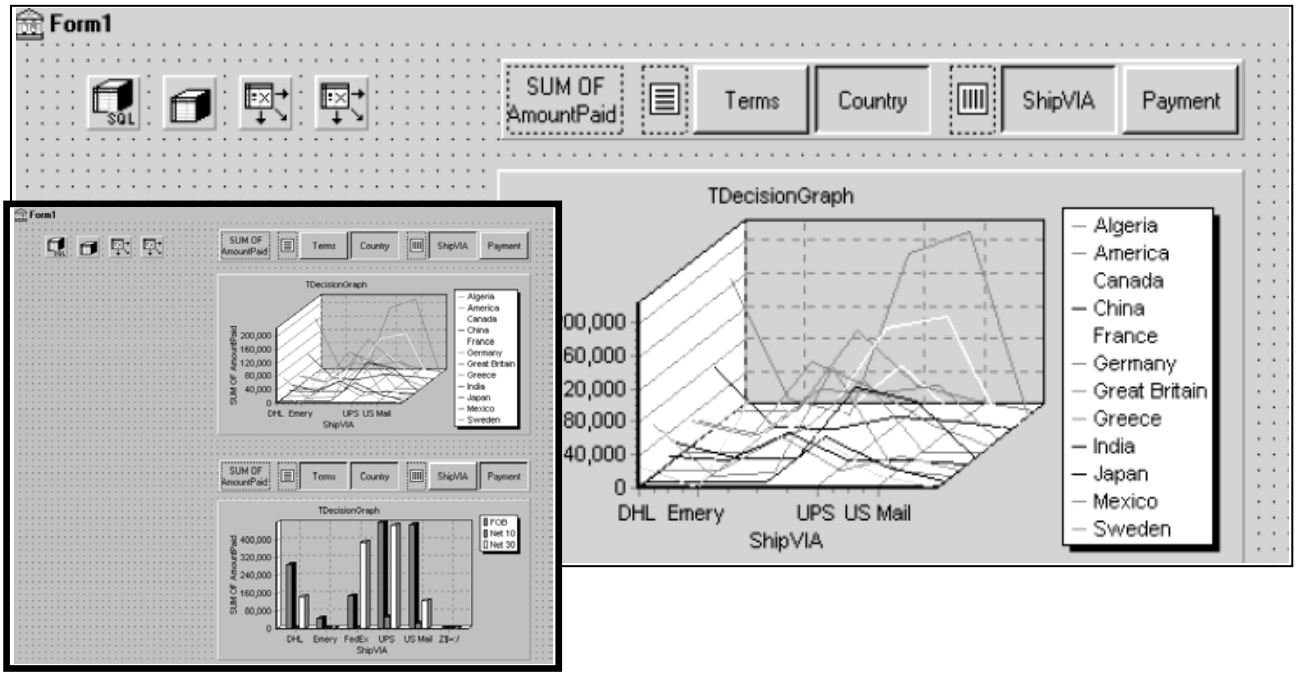


Рис. 6.9. Форма с компонентами (меню, кнопками и командами), которые разработаны в IDE RAD Delphi для решения конкретной задачи (на переднем плане полная форма, на заднем – фрагмент)

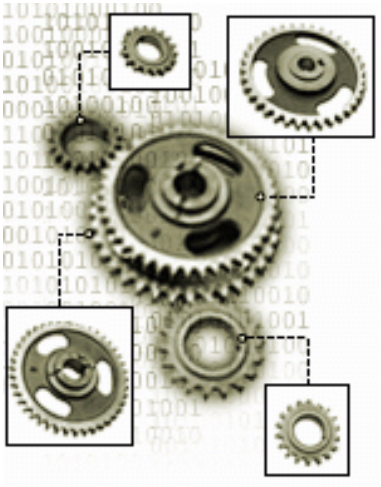


Рис. 6.10. Образное представление взаимодействия компонентов на уровне интерфейсов

Учитывая то, что взаимодействие пользователя с компонентами и компонентов между собой происходят на уровне ***событий*** и ***откликов*** на них, в программный код, кроме алгоритмических составляющих, входят так называемые ***обработчики событий***. В этих фрагментах кода программист указывает, как воспринять ввод тех или иных данных в том или ином месте визуальных компонентов, как расценить щелчок мыши в конкретной точке ***формы*** и какие действия при этом необходимо предпринять. Взаимодействие компонентов между собой происходит на уровне унифицированных COM интерфейсов (рис. 6.10).

Обычно, любая решаемая на компьютере задача реализуется в Delphi в виде приложения. Приложение создаётся из различных частей. Каждая часть размещается в отдельном файле и выполняет строго определённые функции. Набор файлов, необходимых для создания приложения, называется ***проектом***. Поэтому, главной единицей разработки ***приложения*** является ***проект***, хранящийся в

файле, имеющем расширение DPR (сокр. от Delphi PRoject). Он представляет собой главный программный файл на языке Object Pascal, который подключает с помощью операторов `uses` все файлы модулей, входящих в проект. Каждой проектируемой в RAD форме соответствует свой программный модуль (unit), содержащий все относящиеся к форме объявления и методы событий, написанные также на языке Object Pascal. Каждой форме соответствует свой двоичный файл с расширением DFM, содержащий её описание. Программные модули размещаются в отдельных файлах с расширением PAS. Кроме них существует ряд дополнительных файлов.

Файл ресурсов с расширением RES (сокр. от RESource). В нём, кстати, сохраняется пиктограмма приложения, которая будет видна на Панели Задач Windows. В подключённых в виде модулей ***объектных файлах*** (имеют расширение OBJ), размещаются компоненты, написанные на других языках программирования (к примеру, на C++). ***Файл опций*** с расширением DOF (сокр. от Delphi Option Files), содержит задаваемые программистом параметры компиляции и компоновки проекта. Существуют ещё несколько типов файлов выполняющих служебные функции, которые здесь обсуждаться не будут.

В целом, схематически процесс создания приложения в RAD Delphi можно представить как набор последовательных шагов по созданию проекта (см. рис. 6.11).

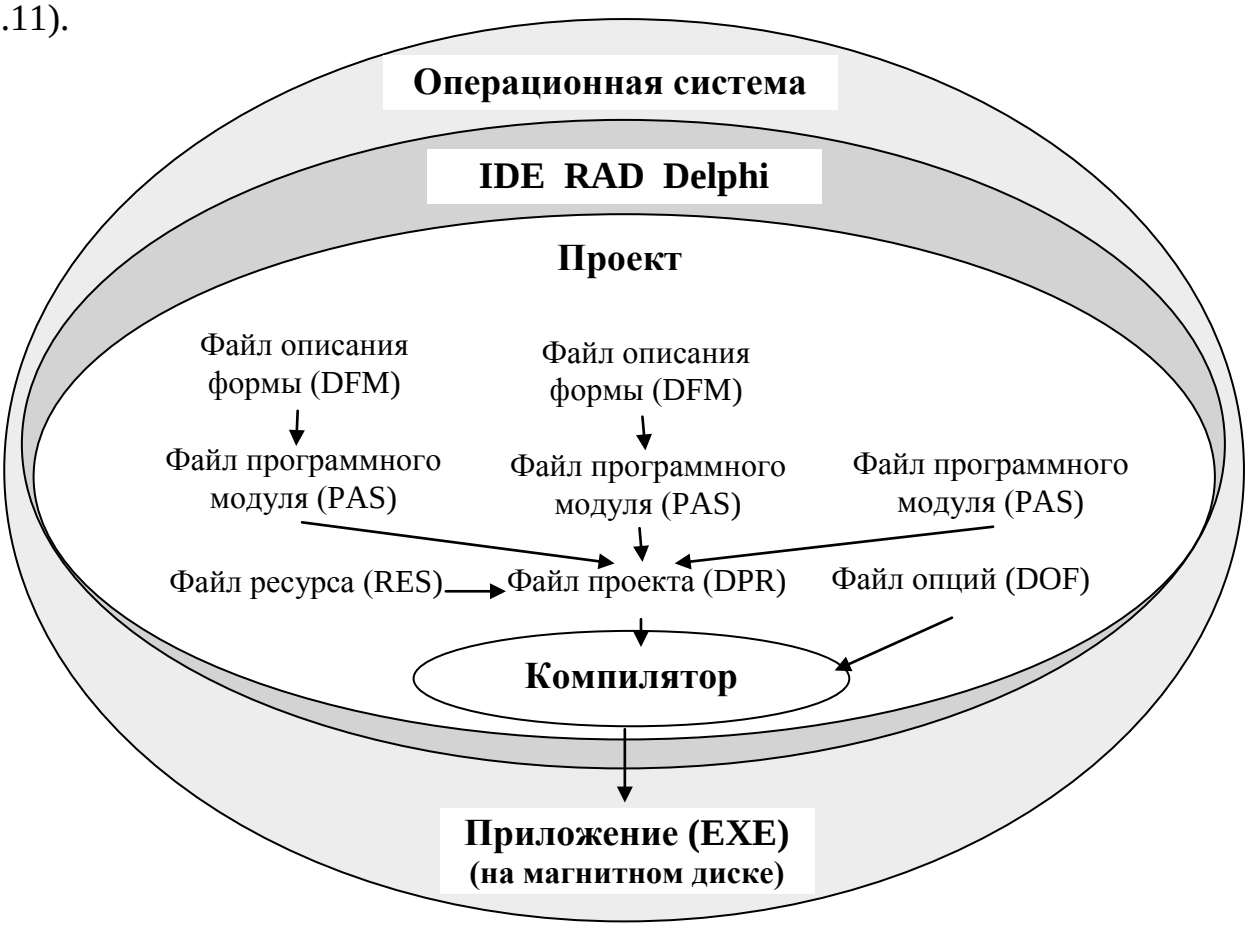


Рис. 6.11. Создание приложения в IDE RAD Delphi

В дальнейшем, загруженное в память компьютера приложение в виде двоичного файла EXE или DLL выполняется как программный модуль.

Идеологически, таким же образом работают многие другие среды визуального программирования для языков C++, Java, C# и ряда других.

Несколько отличается представление взаимодействия программных компонентов в IDE MS Visual Studio.Net, где наибольшей программной единицей является ***решение*** *(solution)*. В ходе работы с IDE этот термин ассоциируется с понятием ***рабочее пространство***, так как буквальный перевод — ***решение*** — не всегда однозначный. Концепция ***решения*** помогает объединить ***проекты*** и другие информационные элементы в одном ***рабочем пространстве***. Множество файлов разного типа, в рамках одного решения составляют ***приложение*** *(application)* Visual Studio.Net 7.0. Рабочеей пространство может содержать несколько ***проектов***, быть пустым или содержать файлы, которые имеют смысл и вне контекста ***решения***. В любом случае, пользователь должен начинать работу в студии с открытия существующего или создания нового ***рабочего пространства***. ***Проект*** как часть ***решения*** состоит из отдельных компонентов, например файлов, которые описывают форму окна или шаблон диалога *(re-**файл**)*, файлов с ***исходными кодами*** программных модулей (.cpp, .cs) и/или файлов, которые представляют собой описание ***запросов*** к ***базе данных*** *(database script)*.

Принципиально отличаются от приложений из визуальных RAD сред те программы (подпрограммы, фрагменты кодов и т.д.), которые реализованы на скриптовых языках и которые представляют собой обычные наборы текстовых строк. К наиболее популярным скриптовым языкам, без сомнения, можно отнести языки Visual Basic и Visual Basic for Applications (VBA), последний из которых встроен во все без исключения офисные приложения Microsoft Office а, в последнее время, и в геоинформационную систему ESRI ArcGIS. При внешней похожести интерфейса VBA (рис. 6.12) на обычные IDE, коды процедур-подпрограмм и процедур-функций ***не компилируются в двоичные файлы с расширением EXE и DLL***, а также ***не загружаются для выполнения в оперативную память компьютера***.

Исполнение инструкций скриптового кода, введённых в тело модуля, а затем вызванных в окно кода, производится путём их интерпретации и немедленного выполнения приложением-контейнером MS Excel, которое в это время находится в оперативной памяти компьютера.

По такому же принципу работают и другие приложения-контейнеры и, в том числе, MS Internet Explorer (IE). Находясь в оперативной памяти, IE, являющийся браузером, т.е. просмотрщиком, принимает на входе строки с тэгами (кодами) языка HTML и отображает закодированные страницы на экране компьютера в соответствии с заданными параметрами разметки. А если он встречает во входном тексте фрагменты программ (кодов) на скриптовых языках, из которых наиболее популярны JavaScript и VBScript, интерпретирует их и выполняет (рис.6.13).

С некоторыми, вообще говоря, отличиями работают и многие другие IDE и RAD для создания программных кодов на разных языках. Не вдаваясь в многообразие тонкостей их работы, приведём мнение одного из авторитетных специалистов по программированию на Visual Basic, Андрея Колесова, автора

многочисленных статей в журналах "КомпьютерПресс", PCWeek/RE и на сервере "Советы" VBA Форума. Его мнение таково[8].

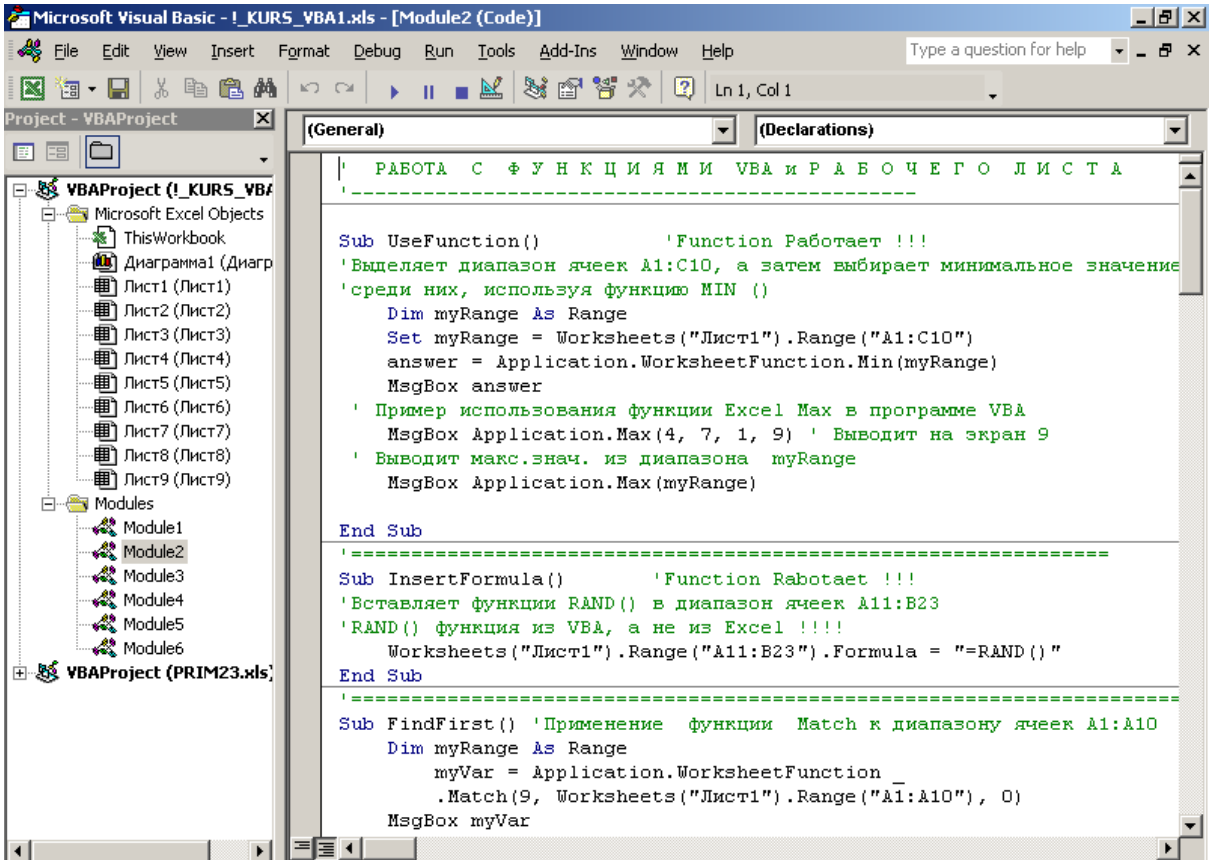


Рис. 6.12. Интерфейс IDE Visual Basic for Applications

```
<SCRIPT LANGUAGE="JavaScript">
   // Присвоение переменной 'a' значения "Привет!".
   a = "Привет!";
</SCRIPT>
```

Рис. 6.13. Пример кода на языке JavaScript, который размещён на HTML-странице

"Еще одной иллюзией является широко распространенное сейчас представления о том, что квалификация программиста определяется тем, каким инструментом он владеет. Более конкретно, C (си)– программист — это сильно, Basic — не очень... Такое мнение не просто неверное: оно формирует неправильное представление о действительности. На самом деле, квалификация разработчика определяется его уровнем ***общеметодической подготовки*** и ***глубиной знания инструмента***. Короче говоря, есть более и менее квалифицированные специалисты, соответственно, есть слабые C-

[8] http://visual.2000.ru/develop/talks/talks1_1.htm

программисты и сильные VB- программисты, и наоборот. При смене инструмента, скорее всего, слабый разработчик останется слабым, а сильный — сильным.

И все же, несмотря на более узкую и глубокую специализацию, "многоязычие" программиста и сегодня является, если не обязательным, то, по крайней мере, весьма желательным. Для Fortran-пользователей (и, наверное, для C тоже) является очень полезными умение создание пользовательского интерфейса с помощью одного из RAD-инструментов. Delphi является в значительной степени самодостаточным инструментом, но вряд ли его пользователь сможет обойтись без применения VB или Java. VB-разработчику знание основ Java также совсем не повредит. И всем им нужно представление о HTML и в ближайшее время — об XML.

В общем, в то время как мы в свое время долго спорили: "Программирование — это наука или искусство?", американцы уже давно пришли к выводу, что это — технология".

6.4. В какой среде работают программы и приложения?

Разработка программ неразрывно связана с ***архитектурой вычислительных сред***, для которых они и создаются. Тут под архитектурой понимается методология объединения и взаимодействия элементов сложной вычислительной структуры на логическом (л.у.), физическом (ф.у.) и программном (п.у.) уровнях. Например, для сети LAN (см. Приложение 4), которая состоит из 5-ти компьютеров, архитектура определяет, какая используется схема соединения ПК (л.у.), какие компоненты аппаратуры соединены (ф.у.) и какой набор программных продуктов обеспечивает работу этой системы: ОС, серверы и т.д. (п.у.). Поэтому разработчику необходимо чётко представлять в каких "***измерениях***" будет функционировать конечный программный код и какую задачу выполнять**:** управлять аппаратурой, реализовывать математический метод или клиент-серверное Web-приложение, COM-сервер, сервлет, апплет и т.д.

Следует особо отметить, что с точки зрения архитектуры по Фон Нейману собственно компьютер состоит из 5-ти следующих ***основных узлов***: (рис.6.14):

❶ ***арифметико-логического устройства (АЛУ);***	❸ ***оперативной памяти (ОП, ОЗУ-оперативного запоминающего устройства***);
❷ ***центрального устройства управления (ЦУУ);***	❹ ***устройств ввода;***
	❺ ***устройств вывода.***

В данном случае, под архитектурой подразумевается описание вычислительной системы на некотором общем уровне, включающем описание пользовательских возможностей программирования, системы команд и средств пользовательского интерфейса, организации памяти и системы адресации, операций ввода/вывода, управления и т.д.

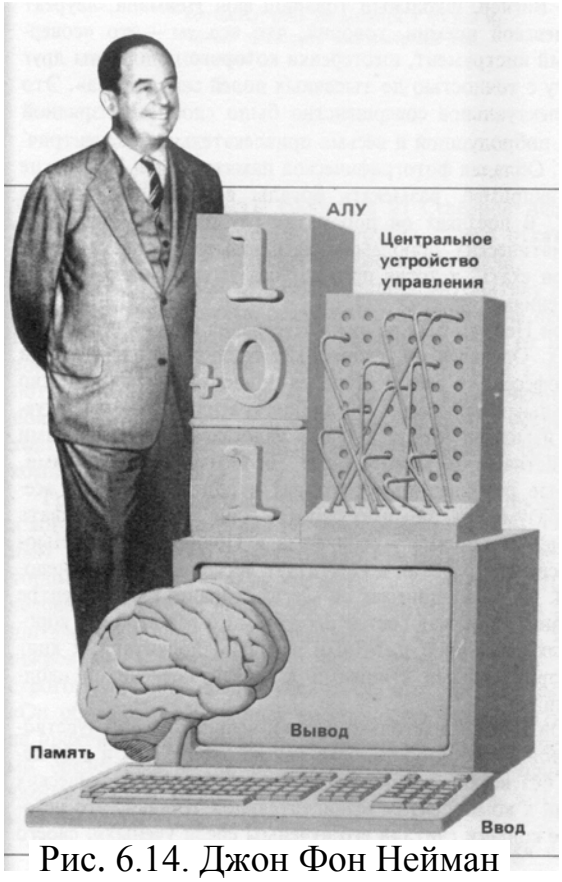


Рис. 6.14. Джон Фон Нейман и его описание архитектуры компютера

И если простые однокомпьютерные (т.е работающие с одним микропроцессором) приложения достаточно просто интегрируются в структуру вычислительной среды, то выход в сеть, где начинают взаимодействовать многопроцессорные и многосетевые образования, как правило, существенно усложняет задачу и приводит к необходимости использования модели клиент/сервер или компонентной модели Web-сервисов.

Термин ***клиент/сервер (или клиент-сервер)*** впервые был применен в 80-х годах, по отношению к процессу использования и взаимодействия персональных компьютеров в сетях LAN (local area network). Современная клиент/серверная модель проявила себя в конце 80-х. Эта программная архитектура является гибкой, основанной на обмене сообщениями в модульной инфраструктуре, предназначенной для повышения практичности (usability), гибкости (flexibility), интероперабельности (interoperability) и масштабируемости (scalability) программных систем и приложений. ***Клиент*** является инициатором запросов на выполнение некоторых услуг (services), а ***сервер*** – предоставляет необходимые запрошенные услуги (services) клиенту.

Итак, рассмотрим последовательно важнейшие вычислительные архитектуры. Архитектуре клиент/сервер предшествовали две архитектуры, существующие и до нашего времени в связи с достаточно большим количеством ***наследуемых систем***, которые продолжают использоваться во многих больших организациях разных стран мира.

Архитектура мэйнфреймов[9] (*mainframe architecture*) (рис. 6.15).

[9] Линию мэйнфреймов продолжают современные суперкомпьютеры. По данным 2002 г., самым быстрым в списке 500-т самых мощных суперкомпьютеров в мире являлся суперкомпьютер Earth Simulator, который был построен японской компанией NEC для моделирования процессов, происходящих в литосфере, атмосфере и гидросфере нашей планеты, подкреплённом соотвествующим ПО. Максимальная производительность суперкомпьютера в тесте Linpack, достигнутая на практике (этот показатель обозначается Rmax), составила 35860 гигафлопс (1 гигафлопс =1 млрд. операций с плавающей точкой в секунду) Этот результат почти вчетверо превосходит достижение ближайших конкурентов - двух идентичных компьютеров Hewlett-Packard ASCI Q, поделивших 2-е и 3-е места в рейтинге с результатом Rmax = 7727 гигафлопс. Оба суперкомпьютера впервые попали в рейтинг TOP 500. Они установлены в Лос-Аламосской национальной лаборатории США и базируются на серверах AlphaServer SC ES45. В каждом суперкомпьютере установлено по 4096 процессоров Alpha с частотой 1,25 ГГц. По суммарной производительности суперкомпьютеров в списке TOP 500 лидирует корпорация IBM (31,6%), за ней идут HP (22,1%) и NEC (14,6%). По количеству произведённых суперкомпьютеров лидирует HP - эта компания построила 137 суперкомпьютеров, представленных в списке TOP 500. На втором месте находится IBM (131 машина), а на третьем Sun Microsystems (88 суперкомпьютеров).

Характеризуется сосредоточением всех встроенных вычислительных средств на центральном хост-компютере. Пользователи взаимодействуют с хостом посредством ввода ключевых командных строк через терминалы. Хосты не поддерживают графический интерфейс пользователя (GUI), но могут взаимодействовать с ПК и рабочими UNIX-станциями (рис.6.15). В последнее время в связи с постоянным ростом нагрузки и объёма информации на серверах WWW, мэйнфреймы начали активно использоваться и в распределённых клиент/серверных архитектурах. Сами же современные мэйнфреймы строятся на кластерах многочисленных процессоров большой мощности.

Рис. 6.15. Архитектура мэйнфреймов и работы с ним

В прошлом, мэйнфреймы назывались универсальными электронно-вычислительными машинами (ЭВМ) и системы программного обеспечения, писавшиеся для них, были, как правило, ***монолитными***. Интерфейс пользователя, деловая (бизнес) логика и функциональные возможности доступа к данным выполнялись в одном большом приложении. Это было связано с тем, что терминалы ввода-вывода, которые использовались для связи с универсальными ЭВМ, не выполняли никаких операций по обработке данных. Все эти операции выполнялись непосредственно мэйнфреймом. Таким образом, создавалась ***монолитная архитектура приложений***, иллюстрируемая на рис. 6.16. Дальнейшее развитие архитектуры приложений было связано с развитием персональных компьютеров, сетевых технологий и технологий WWW и Internet.

Файл-ориентироованняа архитектура (*file sharing architecture*) связана в первую очередь с обычными компютерными сетями LAN, в которых компьютер-сервер загружает файлы из общего дискового пространству в среду, которая используется настольными ПК на рабочих местах. А дальше приложение пользователя запускается (совместно с логикой и данными) в среде

самого настольного ПК. Наиболее активно вычисления с использованием файл-ориентированной архитектуры используется в локальных сетях LAN, которые разворачиваются в компьютерных классах учебных заведений, научных организациях, а также в небольших частных фирмах.

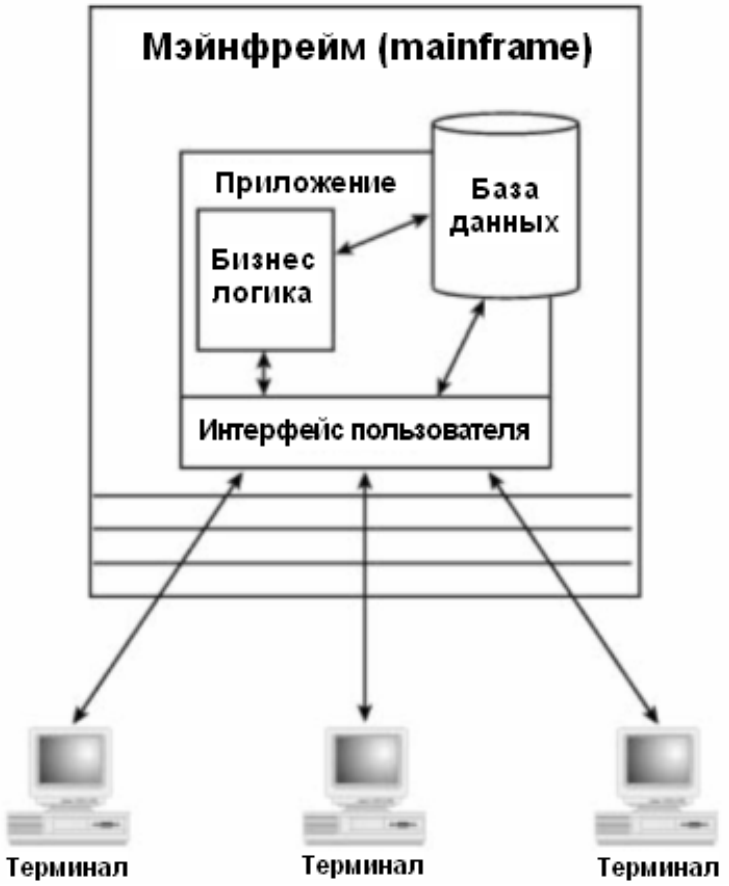


Рис. 6.16. Монолитная архитектура приложений мэйнфреймов

Клиент/серверная архитектура (*client/server architecture*) возникла как результат развития файл-ориентированной архитектуры, которая имеет некоторые ограничения по производительности и масштабируемости. В новой архитектуре мощный компьютер, который используется в качестве сервера баз данных, заменил файл-сервер. Использование систем управления базами данных СУБД (relational DataBase Management System, DBMS) позволило существенно ускорить и упростить связи пользователя с данными. Клиент/серверная архитектура резко снизила сетевой трафик, а также обеспечила многопользовательский доступ к редактированию одних и тех же записей данных в базах данных, которые используются совместно. Основным способом взаимодействия клиентов и серверов стали удалённые вызовы процедур (Remote Procedure Calls, RPCs) или предложения на стандартном языке запросов (Standard Query Language, SQL) (рис.6.17).

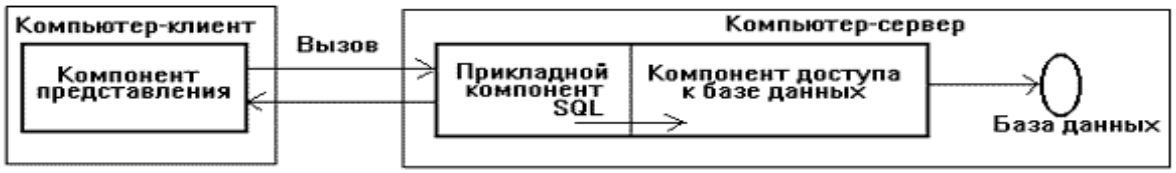
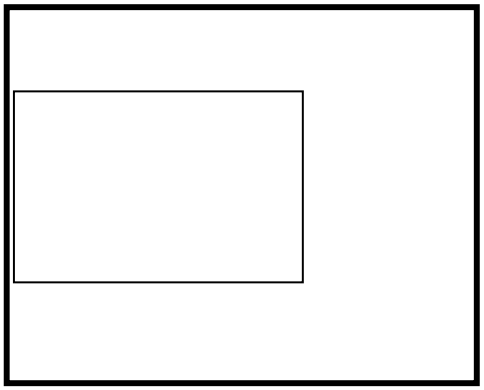


Рис. 6.17. Модель клиент/серверной архитектуры с сервером баз данных

В ***двухярусной архитектуре*** (*two tier architectures*) системный интерфейс пользователя как правило размещён в среде настольного компютера пользователя, а службы управления базами данных – на более мощном компьютере сервере, который обеспечивает сохранение приложений, а также

процедур и триггеров баз данных (рис.6.18). В отличие от очень дорогостоящих мэйнфреймов эта архитектура является хорошим решением для распределённых вычислений в рабочих группах до 100 человек, которые работают в LAN отделов и организаций одновременно.

Компоненты приложений в клиент/серверных вычисленнях, как правило, распределены так, чтобы база данных постоянно находилась на сервере (UNIX или мэйнфрейм), интерфейс пользователя на ПК-клиенте, а ***деловая (бизнес) логика*** находится в обоих компонентах.



Трёхярусная архитектура (*three tier architectures*), известная также как многоярусная (multi-tier architecture), разрабатывалась для снятия ограничений, которые существовали в двухярусной. В ней средний ярус был расположен между системным интерфейсом среды клиента и средой сервера, которая управляет базой данных. Существует достаточно много реализаций этого средного уровня, к к которым можно отнести мониторы обработки транзакций (transaction processing monitors), сервера сообщений (message servers) и сервера приложений (application servers). Трёхъярусная архитектура позволяет одновременно работать группам, насчитывающим несколько тысяч пользователей. Гибкость в настройке требуемой конфигурации здесь достигается простым применением технологии "drag and drop" для фрагментов кодов используемых приложений на разные компьютеры любого яруса из трёх существующих. Ограничения трёхъярусной архитектуры связаны с тем, что использование сред разработки приложений здесь значительно сложнее, чем визуально-ориентированная разработка в двухъярусной архитектуре. В последнее время в трёхъярусной архитектуре в качестве серверов начинают активно применяться мэйнфреймы.

Трёхярусная архитектура с технологиями монитора обработки транзакций (*three tier architecture with transaction processing monitor technology*) является одной из наиболее распространённых и включает на среднем ярусе монитор обработки транзакций (Transaction Processing, TP). В его задачи входит организация очередей запросов, распределение транзакций и приоритетов выполнения задач. Кроме того, он обеспечивает:

❶ возможность обновления многочисленных различных СУБД путём однократно выполняемой транзакции;

❷ соединение и обработку многочисленных источников данных, включая двумерные файлы (flat files), нереляционные БД и мэйнфреймы;

❸ возможность присваивания разных приоритетов транзакциям;;

❹ обеспечение робастной (устойчивой) безопасности.

Трёхярусная архитектура с сервером сообщений (*three tier with message server*) представляет реализацию процесса асинхронного обмена сообщениями (messaging) с разными приоритетами. Сообщения, как правило, содержат заголовки, состоящие из данных об уровне приоритета, а также адреса и идентификационного номера сообщения. Если в данной архитектуре программируемая интеллектуальность сосредоточена в сообщениях, то в среде мониторов транзакций непосредственно в мониторе, обрабатывающем транзакции в виде неинтерпретируемых пакетов данных. ***Трёхъярусная архитектура с сервером сообщений является лучшим решением для беспроводных инфраструктур.***

Трёхярусная архитектура с сервером приложений (*three tier with an application server*) обеспечивает размещение основного кода приложения для выполнения не в среде клиента, а на некотором выбранном для использования хосте, то есть компьютере узла сети. Сервер приложений не имеет и не управляется GUI, а только исполнят бизнес-логику, вычисления и моделирует машину поиска данных. Преимуществом такого подхода является уменьшение компонентов пользовательского ПО на компьютере клиенте и повышение безопасности функционирования решений в целом (рис.6.19).

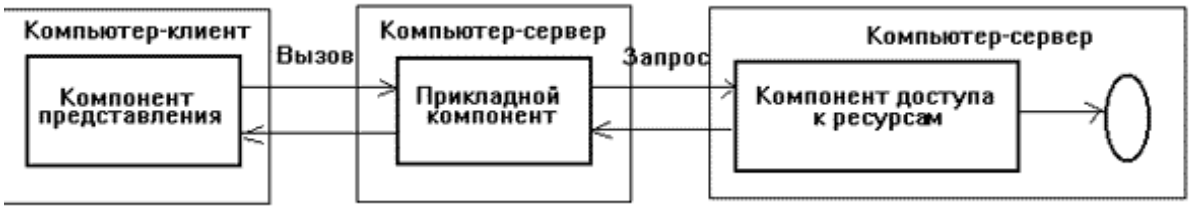


Рис. 6.19. Трёхъярусная архитектура с сервером приложений

Приложения становятся более масштабируемыми, а стоимость развёртывания и поддержки программных продуктов и систем на одном сервере значительно меньше, чем на многочисленных клиентах. Таким образом, данная архитектура наиболее применима при условиях необходимости выполнения требований безопасности, масштабируемости и снижения стоимости работы системы.

Трёхярусная архитектура с ORB (*three tier with an ORB architecture*) для распределённых вычислений активно развивается промышленными консорциумами на уровне стандартов, повышающих интероперабельность и кроссплатформенность приложений, разработанных изначально на разных языках программирования для разных платформ. Основной проблемой для обсуждаемых стандартов является определение механизмов и элементов реализации общих для взаимодействующих систем объектных брокеров запросов (Object Request Broker, ORB). На настоящем этапе разработка клиент/серверных систем с использованием технологий, поддерживающих распределённые, объекты имеет громадные перспективы в условиях повсеместного развития и применения сетевых, Интернет и WWW технологий в существенно гетерогенных средах. Кроме поддержки интероперабельности среди языков и платформ, данная архитектура существенно повышает

эксплуатационную надёжность и удобство сопровождения (maintainability), а также адаптируемость (adaptability) создаваемого ПО. К наиболее развитым и используемым в настоящее время относятся следующие технологии:

❶ Брокер запросов общей объектной архитектуры (Common Object Request Broker Architecture, CORBA);

❷ Компонентная объектная модель/Распределённая КОМ (COM/DCOM, Component Object Model/Distributed Component Object Model).

Промышленные круги активно работают над стандартами, которые обеспечили бы интероперабельность между CORBA and COM/DCOM. Наиболее активным участником данного процесса является консорциум Object Management Group (OMG), который разработал отображения соответствия данных и процессов между CORBA и COM/DCOM на уровне нескольких программных продуктов.

Распределённая/совместная промышленная архитектура (*distributed/collaborative enterprise architecture*) была разработана в 1993 году. Эта программная архитектура базируется на технологии применения объектных брокеров запросов (Object Request Broker, ORB), являющихся продолжением развития архитектуры CORBA, путём применения совместно (share) и повторно (reusable) используемых бизнес-моделей (а не просто объектов!) в масштабе распределённых предприятий (enterprise-wide scale). Преимущества данного подхода заключаются в том, что стандартизированные бизнес-объектные модели (business object models) и распределённые вычислительные объекты (distributed object computing) комбинируются вместе для обеспечения организационной гибкости для достижения операционной и технологической эффективности в масштабах целых предприятий.

***Глобальная** XML **архитектура** Microsoft (Microsoft Global XML Architecture, MS GXA*) является структурой и базовой основой (framework) протокола, спроектированного для обеспечения модели полной совместимости при построении протоколов инфраструктурного уровня для Web-сервисов и Web-приложений. *Web – **сервисы** –* это - прикладные сервисы, включающие программы (приложения), программирование, данные и человеческие ресурсы, которые сделаны доступными с Web-серверов для Web-пользователей или других Web-присоединённых программ. В дополнение к этой основной структуре протокола, GXA определяет семейство подключаемых протоколов инфраструктуры, которые обеспечивают для функционирующих в GXA-среде приложения наиболее необходимыми сервисами, такими, как безопасность, надёжность и поддержка многоуровневых и многосвязных служб согласования их работы. GXA поддерживает принципы, объединяющие отдельные протоколы GXA-инфраструктуры в единую связную платформу для функционирования сервисов и приложений. Основополагающей предпосылкой GXA является ликвидация необходимости для разработчиков приложений перестраивать платформу целиком для каждого нового приложения. Эта предпосылка доминирует в моделях разработки на однокомпьютерных рабочих местах, где 95% программистов всего мира используют разные платформы разработки. Традиционный мир однокомпьютерной разработки ПО

десятилетиями формализовал методы модуляризации программного кода и данных, которые позволили осуществить разделение между приложением и платформой. Мир коммуникационных протоколов также создал свои принципы модуляризации, среди которых наиболее известен 7-и уровневый протокол OSI.

Основной целью GXA является:

❶ создание более общей модели для структурирования протокола;

❷ разработка семейства протоколов платформенного уровня для разработчиков приложений, использующих Web-сервисы и Web-приложения.

Так же, как и операционная система обеспечивает средства ***совместной работы*** программистов, программных систем и приложений, управляет процессами и потоками, обеспечивает контроль доступа к ресурсам и их распределение,управляет памятью так и GXA обеспечивает комплекс общих средств, требуемых в широком спектре разработки и использования Web-сервисов и Web-приложений. Язык XML используется для организации и использования данных. SOAP служит для передачи данных в сетях, WSDL применяется для описания доступных данных, а UDDI используется для перечисления доступных сервисов (рис.6.20.).

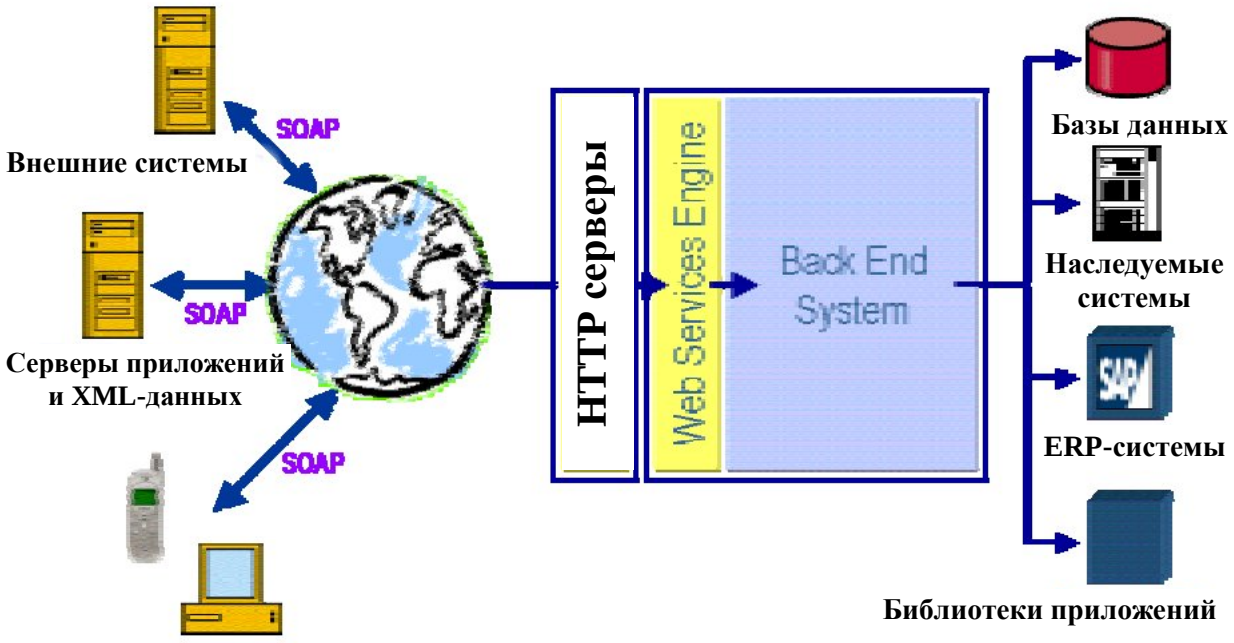


Рис. 6.20. Типичная архитектура Web-сервисов

Сам язык XML служит технологией и средством организации сложно структурированных данных разной природы (программы, текстовые, звуковые, графические, видео файлы и т.д.), которые размещены в разных точках мирового пространства на разных ПК в единое целое (!!!).

Подводя итоги, следует добавить, что вышеуказанные архитектуры не единственные и не последние. В настоящее время активно развивается архитектура ***беспроводных коммуникаций*** *(wireless architecture)* на базе ***адресных сервисов*** *(location services)*. Эта и некоторые другие архитектуры распределённых вычислений поддерживаются двумя мощными архитектурами

и соответствующими технологиями: Jini™ от Sun Microsystems и .NET™ от Microsoft (см. подраздел 6.6).

Поэтому разработчику программного обеспечения необходимо постоянно следить за развитием и взаимосвязью информационных архитектур, а также компонентов и программных средств компьютерной техники. При этом следует своевременно корректировать методологии разработки своих приложений и решений, с соответствующим освоением:

основ программной инженерии, новых технологий программирования, языков программирования и инструментальных средств разработки ПО.

6.5 Как проектируются приложения и решения?

Информационные системы играют основополагающую роль в деле определения успеха практически любого современного бизнеса и решения многих прикладных проблем и задач. С появлением персональных компьютеров такие системы и программные приложения прочно укрепились в большинстве государственных и бизнес-организаций мира. Эти новые системы вместе с существующими унаследованными системами (legacy systems) предоставляют критичные для функционирования организаций средства хранения, организации, получения и обработки информации, в том числе и на правительственном уровне[10]. Потенциально, эти системы могут стать источником детальной и разноплановой информации, которая способствует принятию качественно лучших решений.

В большинстве своем современные информационные системы являются распределенными, то есть состоящими из нескольких частей, обычно взаимодействующих друг с другом по сети. Разбивать систему на части приходится потому, что возможности автономной компьютерной системы не способны покрыть потребности серьезной информационной системы в вычислительных возможностях и хранении данных. К тому же разбиение на части с внесением избыточности в эти части обеспечивает устойчивость механизма восстановления системы после сбоев.

В софтверной индустрии с учётом большого объёма и сложности выполняемых при реализации ***крупных приложений*** дополнительных мероприятий часто последние именуют ***решениями***[11]. Свой большой опыт в разработке бизнес–решений специалисты корпорации Microsoft обобщили в виде комплекса документов по описанию ***Дисциплины разработки решений Microsoft*** (Microsoft Solution Framework, MSF[12]). MSF содержит в себе набор ***моделей*** и четко определенных проектных ***вех (т.е. контрольных точек)***, которые можно рассматривать как рекомендуемые отправные точки, равно как и

[10] B2G, G2C, G2G. Эти аббревиатуры определяют новые сферы бизнеса, в которые, тем или иным образом, вовлечено государство (Government). Они раскрываются как Business-to-Government, Government-to-Citizens, Government-to-Government. Являются следствием включения Государства в процесс электронизации всехх видов деятельности. Концепция *Electronic Government* была объявлена в США на самом высоком государственном уровне первого июля 1997 года.

[11] Брандт Д. Architectures. Экзамен – экстерном. (экзамен 70-100). СПБ.: Питер, 2001. – 432 с.

[12] Материалы Microsoft MSF: (http://www.microsoft.com/msf и http://www.microsoft.com/rus/msf)

руководство по планированию, ведению и управлению проектами в сфере информационных технологий. Эти ***модели*** — результат интеграции в единую систему наиболее успешных и многократно примененных практик, выявленных в процессе анализа опыта по разработке программных продуктов, накопленного не только Microsoft, но и ее заказчиками и партнерами.

Они не являются единственными в своём роде, но в отличие от многих других могут быть получены бесплатно на русском языке.

С точки зрения MSF, разработка решения бизнес–проблемы — это больше, чем простое написание крутого приложения с применением новейших технологий. Разработка приложения должна вестись, исходя из предварительно подготовленных бизнес–требований. Будучи разработанным, приложение должно быть принято конечными пользователями, которым оно обязано помогать в решении повседневных проблем. Другими словами, приложение должно быть написано под конкретный процесс и полностью соответствовать его правилам.

После того, как будут подготовлены и уточнены все ***требования*** и проявятся контуры будущего ***решения***, эти высокоуровневые абстракции смогут предоставить информацию, необходимую для создания детальных разноуровневых проектов, которые лягут в основу реализации ***приложений решения***. Такие шаги, показаны на рис. 6.21.

6.5.1.Подготовительный этап: анализ требований. Разработка решения начинается с анализа требований. Этот высокоуровневый процесс сбора информации должен ответить на вопрос: ***что*** необходимо сделать, ***почему*** это необходимо сделать и какой ожидается ***результат***. После этого можно подвести итоги и зафиксировать требования в ***документации***. Полный ***анализ требований*** даёт информацию, которая описывает:

Рис. 6.21. Процесс разработки прикладной информационной системы

❶ ***границы проекта;***

❷ ***суть проблемы (потребности пользователей);***

❸ ***требуемый уровень безопасности;***

❹ ***производительность приложения;***

❺ ***потребности в сопровождении;***

❻ ***возможности расширяемости;***

❼ ***доступность;***

❽ ***масштабируемость;***

❾ ***человеческий фактор;***

❿ ***интеграцию с существующим окружением;***

❶❶ ***используемые методологии.***

6.5.2.Определение технической архитектуры. После выяснения

требований нужно приступать к оценке возможностей и ограничений технологий, позволяющих реализовать решение, и начинать формулировать техническую архитектуру, подходящую для его реализации и развёртывания.

Здесь следует определиться с количеством обслуживаемых приложением пользователей и соответственно объёмами и расположением источников используемых в будущем данных (т.е. серверов, хранилищ данных и т.д.). Важно оценить имеющиеся стандарты и технологии разработки используемых в будущем решений. И, наконец, техническая архитектура должна в общих чертах давать представление о стратегии развертывания решения после его реализации.

6.5.3.Разработка модели данных. Техническая архитектура дает представление о требованиях к аппаратным и программным компонентам хранения данных, однако она не дает никаких указаний о том, какими должны быть элементы используемых ***решением*** данных и как их следует организовывать. Это определяется ***моделью данных***, которая, в свою очередь, определяется ***метаданными*** (metadata), то есть ***данными о данных***. Модель данных определяет все элементы данных, то есть ***сущности*** (entities), а также характеристики — ***атрибуты*** (attributes) — каждой из них и организацию этих сущностей. Для организации данных существуют специальные ***правила нормализации данных*** (data normalization rules). Помимо этого модель данных определяет ***связи*** (relationships) между сущностями, а также специальные правила, так называемые ***ограничения*** (constraints), используемые для обеспечения ***целостности*** (integrity) накапливаемых в дальнейшем данных.

6.5.4.Разработка проектных частей приложений и решений.

Следующая задача, после того как были разработаны техническая архитектура и модель данных, это разработка ***приложений решения***. Разработка приложения подразумевает три различных вида деятельности:

❶ разработку ***концептуального*** и ***логического проектов приложений***;

❷ проектирование ***интерфейса*** и ***служб пользователя***;

❸ разработку ***физического проекта***.

Большинство людей считает, что основная деятельность при разработке приложения заключается в написании и отладке кода (рис. 6.22). ***Хотя процесс написания и отладки кода и представляет собой неотъемлемую часть процесса разработки, эта деятельность не может начаться до тех пор, пока не будут готовы концептуальный, логический и физический проекты приложений.***

Итак, проектирование приложения начинается с разработки концептуального проекта, который основывается на бизнес-требованиях и обычно содержит ***сценарии***: ***модели*** последовательность действий/процесс (workflow/process model) и ***модели*** задача/последовательность (task/ sequence model).

Логический проект, получаемый далее из концептуального проекта, дает абстрактное высокоуровневое описание приложения. Логическая модель разбивает приложение на функциональные модули, соответствующие бизнес-компонентам приложения. Центральной задачей логического проектирования

является вычленение и специфицирование бизнес-объектов, их свойств и методов. В терминах модели приложения MSF эти бизнес-объекты называются ***службами.***

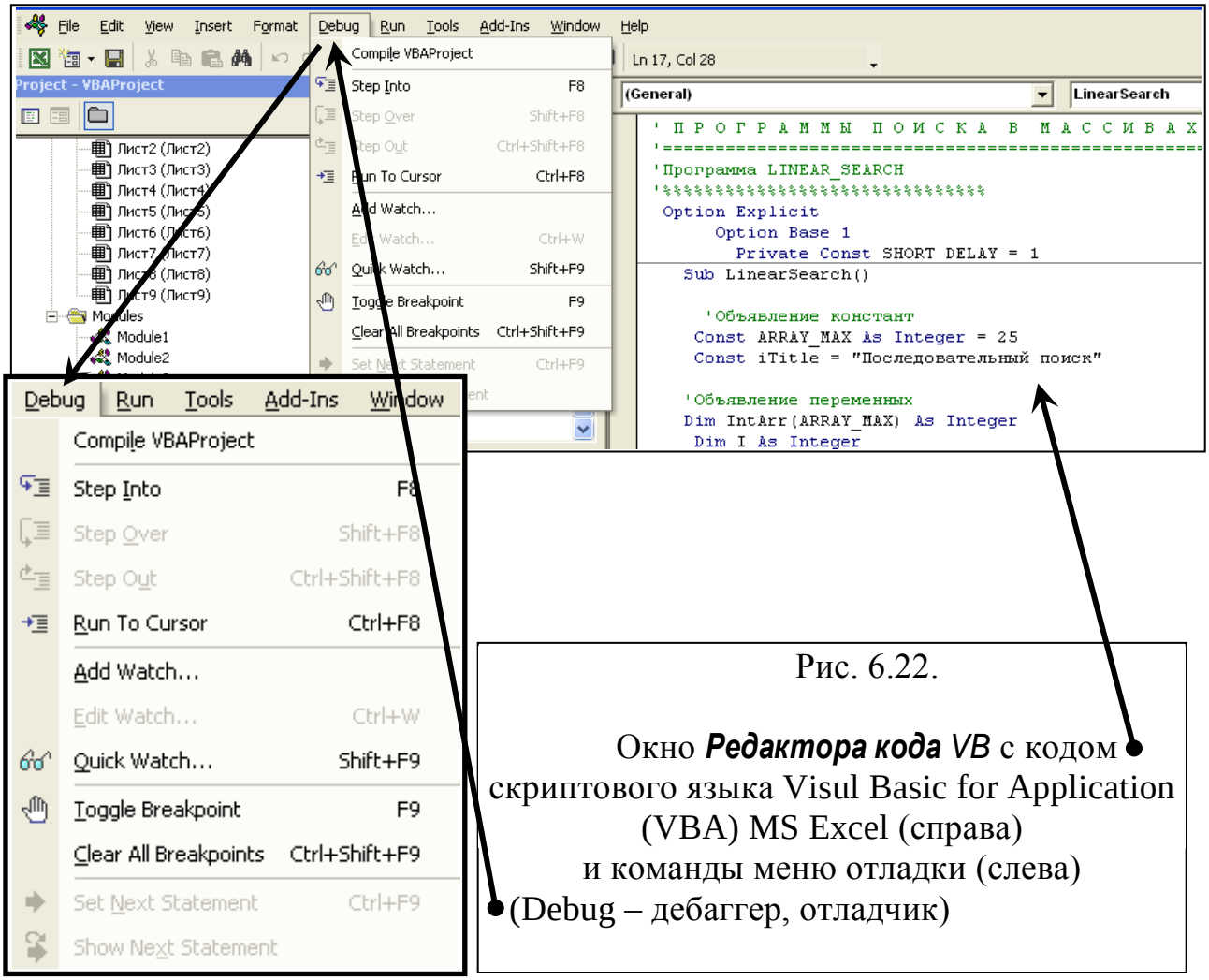


Рис. 6.22.

Окно ***Редактора кода*** *VB* с кодом скриптового языка Visul Basic for Application (VBA) MS Excel (справа) и команды меню отладки (слева) (Debug – дебаггер, отладчик)

Логический проект определяет ***службы приложения*** *(application services)*, предоставляемые данным приложением. Использование моделей, полученных в результате концептуального и логического проектирования, позволяет смотреть на приложение в терминах ***бизнес-компонентов*** и ***программной архитектуры***. Как правило, оценка логического проекта производится по семи ключевым факторам. Более точные критерии оценки определяются конкретным решением, однако для логического проекта наиболее важны ограничения на:

❶ ***производительность (performance),***
❷ ***сопровождаемостъ (maintainability),***
❸ ***расширяемость (extensibility),***
❹ ***доступность (availability),***
❺ ***масштабируемость (scalability)***
❻ ***безопасность (security),***
❼ ***отказоустойчивость*** *(failure-resistance).*

К остальным, также существенно влияющим на логический проект требованиям, относятся:

❶ ***практичность*** *(usability)*
❷ ***гибкость*** *(flexibility)*
❸ ***интероперабельность*** *(interoperability)*
❹ ***кроссплатформенность*** *(crossplatform)*
❺ ***переносимость (мобильность, портабельность) (mobility)***
❻ ***адаптируемость*** *(adaptability)*

6.5.5.Проектирование интерфейса и служб пользователя. ***Интерфейс пользователя*** *(UI)* представляет службы для организации полнофункциональной работы с приложением. Это означает, что он предоставляет средства для выполнения следующих операций:

❶ перемещения компонентов;	❹ вывода данных;
❷ ввода данных;	❺ манипулирования с данными;
❸ представления данных;	❻ помощи пользователю (help).

В зависимости от конкретной реализации, интерфейс пользователя может быть частью приложения, а может быть отдельным клиентским приложением. Кроме того, он может быть и набором сценариев в виде Web-страниц, отображаемых в браузере. Под «пользователем» приложения обычно понимается человек, использующий приложение, поэтому под ***интерфейсом пользователя*** понимаются формы, элементы управления, отчеты и другие объекты, предоставляемые конечному пользователю для взаимодействия с приложением и данными. Однако «пользователь» приложения не обязательно означает человека: другое приложение тоже можно считать пользователем. В этом случае под интерфейсом пользователя подразумевается средство взаимодействия программ, обеспечивающее совместную работу приложений.

6.5.6.Разработка физического проекта. После того, как была спроектирована модель данных, созданы концептуальный и логический проекты, спроектирован интерфейс пользователя и службы приложения, требуется выполнить работы, касающиеся физического проекта решения. Это подразумевает следующую деятельность:

❶ оценку физического проекта;
❷ проектирование компонентов;
❸ разработку стратегии доступа к базе данных.

Физический проект определяет реализацию приложения и используемые для этого технологии. Как правило, есть несколько путей реализации, из которых необходимо выбрать наиболее подходящий. Для оценки физического проекта используются те же самые семь ключевых факторов, что и для оценки логического проекта. Как и для логического проекта, точные критерии оценки физического проекта определяются конкретным решением, но, как правило, для физического проекта наиболее существенными ограничениями являются требования к достижению оптимальных характеристик показателей: ***производительности, сопровождаемости, расширяемости, доступности, масштабируемости и безопасности.*** Особого внимания требуют также и такие области разработки, как ***проектирование компонентов (component design)*** и ***организация доступа к базам данных (database access).*** Проектные решения в этих областях могут сильно повлиять на многие характеристики

производительности функционирования приложения.

6.5.7.Разворачивание и сопровождение решения. После того как приложения были реализованы и оттестированы, начинается разворачивание системы. Когда система развёрнута и верифицирована, ее нужно сопровождать. И развёртывание, и сопровождение чрезвычайно важны для общего успеха приложения не только у настоящего, но и у будущих заказчиков.

6.6.Какие существуют приложения?

Как правило, разработка приложения состоит из проектирования, моделирования, создания прототипа и в конечном итоге его реализации и тестирования. На фазах проектирования и моделирования разрабатывается ***архитектура приложения, которая частично зависит и от архитектуры вычислительной среды***. Почти все приложения содержат код представления, код обработки данных и код обращения к хранилищам данных. Архитектура приложения определяет то, как будет организован этот код. Анализ и проектирование являются неотъемлемой частью процесса разработки приложения. Разрабатывая архитектуру приложения, следует тщательно обдумывать последствия, вытекающие из проектных решений. Перед тем как начать проектирование, имеет смысл определиться с типом будущего приложения. Для описания характеристик приложения или типа приложения используется целый ряд терминов, в том числе (табл.. 6.2):

Таблица 6.2.

Тип приложения в зависимости от архитектуры, в которой оно выполняется

Названия приложения	
❶ настольное; ❷ контейнер; ❸ распределённое; ❹ клиент-серверное; ❺ SDI; ❻ MDI; ❼ консольное; ❽ диалоговое (мастер, Wizard);	❾ одноярусное; ❶❶ двухъярусное; ❶❶ многоярусное (N-уровневое); ❶❷ Web-приложение; ❶❸ совместно работающие приложения; ❶❹ компонент; ❶❺ сетевой сервис или Web-сервис.

По смыслу эти термины перекрывают сферы действия друг друга, что совсем неудивительно. Как правило, реальные приложения представляют собой комбинацию нескольких вышеуказанных типов.

Охарактеризуем кратко каждое из вышеуказанных приложений.

❶ ***Настольные (desktop) приложения*** — это приложения, которые выполняются на одном компьютере и используются одним пользователем. Настольные приложения размещаются на жёстком диске компьютера пользователя и работают только с локальными ресурсами данного ПК. Настольные приложения могут быть основаны на двухярусной архитектуре, которая подразумевает обращение к сетевому серверу за данными.

❷ ***Контейнер (container).*** В разработанной Sun Microsystems компонентной архитектуре *JavaBeans* и в компонентной технологии Microsoft *Component Object Model (COM),* ***контейнер*** является прикладной программой или подпрограммой, в которой выполняется внешний или встроенный блок программы, называемый компонентом (*component*). Например, компоненты типа ***кнопка или элемент выпадающего меню,*** маленький калькулятор или другой элемент графического интерфейса пользователя – все они выполняются с использованием компонентных моделей JavaBeans™ или COM™, которые позволяют выполнять их в контейнере-браузере Netscape или в контейнерах Microsoft, таких как MS Internet Explorer, Visual Basic, Excel или Word. Для трёх последних наполнением контейнера является скриптовый код языка Visual Basic for Application, который они выполняют (интерпретируют), сами при этом находясь в ***оперативном запоминающем устройстве*** *(RAM)*.

❸ ***Распределённые*** *(distributed)* ***приложения*** — это составные приложения, разные части которых выполняются по отдельности, но делают общее дело. Распределенные приложения обычно задействуют для решения проблемы или выполнения одной задачи с использованием нескольких компьютерных систем.

❹ ***Клиент-серверные*** *(client/server)* ***приложения*** — это распределённые приложения, основанные на модели вычислений, в которой клиент запрашивает услуги у другой сущности — сервера. В типичном для бизнес-систем клиент-серверном приложении клиент выполняется на персональном компьютере, а расположенный на удаленной более производительной машине сервер предоставляет ему услуги по доступу к хранящимся на сервере данным. Клиентская часть приложения обычно оптимизируется для взаимодействия с пользователем, в то время как серверная часть предоставляет функциональность, совместно используемую многими пользователями.

В клиент-серверных системах обработка данных производится и на клиенте, и на сервере. Этим они отличаются от систем, данные которых хранятся на файл-сервере: в таких системах никаких вычислений на файл-сервере не производится, и все компоненты приложения выполняются на компьютере-клиенте. Многие распределенные приложения имеют клиент-серверную архитектуру. Это модель, в которой потребитель услуг (клиент) запрашивает их у поставщика (сервера). Клиент-серверная архитектура обычно реализуется в виде нескольких приложений, выполняющихся на различных системах, но это не является обязательным. Эти приложения могут выполняться и на одной компьютерной системе. Дело здесь не в реализации, а во взаимодействии, при котором один запрашивает услуги, предоставляемые другим — именно ***модель взаимодействия*** определяет суть клиент-серверной архитектуры. Ярким примером таких приложений являются библиотеки DLL.

❺ *SDI (****однодокументный интерфейс*** *– Single Document Interface).* В Windows-приложениях встречаются два основных стиля интерфейса пользователя. Например, приложение Блокнот (Notepad), позволяет работать одновременно только с одним документом. Чтобы открыть другой документ, нужно закрыть текущий. Приложение, подобное ***Блокноту*** и использующее одно

главное и несколько дополнительных вторичных окон, называется ***SDI-приложением (Single Document Interface — однодокументный интерфейс).*** Единственный способ работать одновременно с несколькими объектами в SDI-приложении — открыть несколько экземпляров этого приложения. Главные окна SDI-приложения можно свертывать и развертывать независимо друг от друга. Если делается попытка открыть уже открытый объект, активизируется существующее окно. В Windows 95 чаще всего встречаются именно SDI-приложения (рис 6.23.), поскольку в самой ОС Windows и в MS Office акцент в работе с данными делается в рамках ***открываемого приложением документа***.

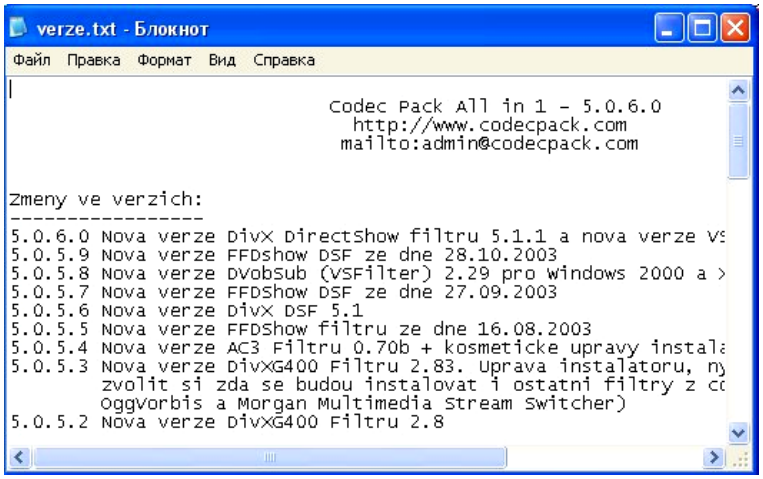


Рис. 6.23 . Приложение Блокнот с SDI–интерфейсом

❻ *MDI (**многодокументный интерфейс** – Multiple Document Interface).* Многие приложения в ОС Windows, например Microsoft Word, Excel, Access и некоторые другие, позволяют работать одновременно с несколькими документами. Каждый документ таких приложений отображается в отдельном окне, переключаться между которыми позволяет пункт меню ***Окно*** (*Window*) (рис. 6.24).

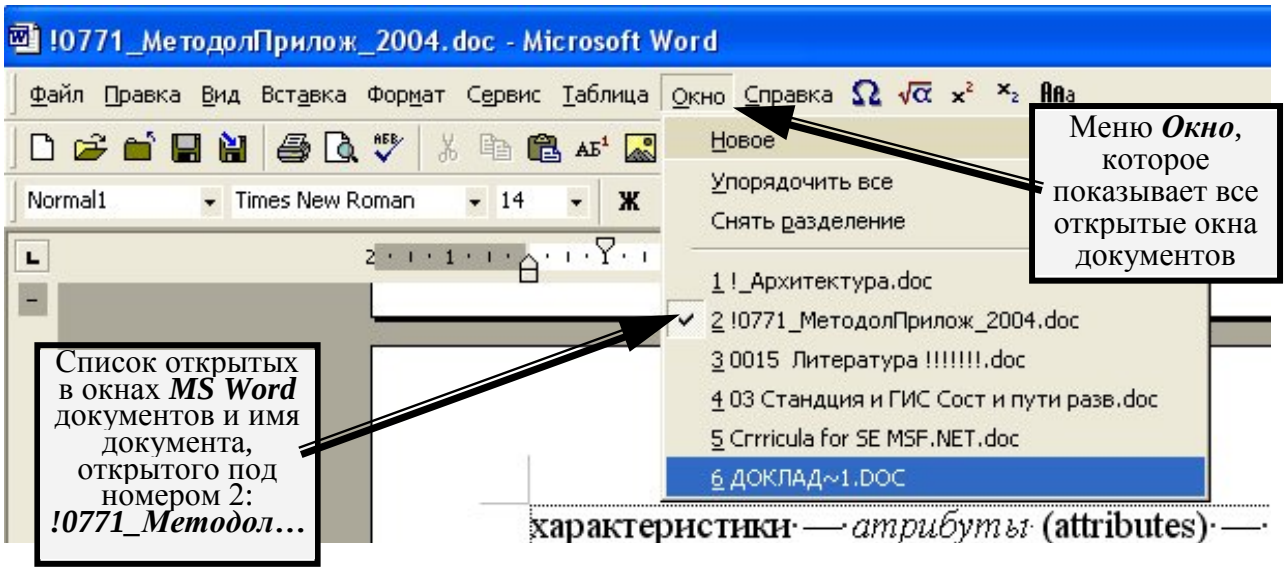


Рис. 6.24. MDI–интерфейс в приложении Word

Каждое дочернее окно по сути своей является главным, но оно всегда содержится внутри родительского окна. Дочерние окна MDI-приложения

свертываются в пределах родительского окна, и, будучи свернутыми, содержат значок документа. Примерами MDI-приложений служат также MS Access и Excel. MS Excel, например, позволяет создавать и отображать дочерние окна нескольких различных типов, такие как диаграммы и электронные таблицы. При активизации таких окон изменяется строка меню приложения. Все индивидуальные окна заключены в пределы родительского окна Excel. При сворачивании Microsoft Excel свертываются все дочерние окна, и на панели задач отображается только значок родительского окна. (рис. 6.25).

Рис. 6.25. Отображение значка родительского
окна приложения MS Excel на панели задач (Task Bar) ОС Windows

❼ ***Консольное приложение.*** У консольных приложений нет графического интерфейса. В его отсутствие взаимодействие пользователя с консольным приложением производится с помощью ***вводимых пользователем команд (т.е. интерфейса командной строки).*** Обычно у каждого консольного приложения существует определённый набор команд, которые можно использовать для доступа к его функциональности. Однако запомнить их имена и каков их синтаксис, порой бывает затруднительно. Большинство приложений для мэйнфреймов и унаследованных приложений относятся к категории ***консольных***. Как правило, эти приложения предназначены для использования с алфавитно-цифровыми терминалами. В среде Windows терминал обычно заменяется программой-приложением эмуляции среды MS DOS – *Command Prompt* (***Командная строка***) (рис. 6.26).

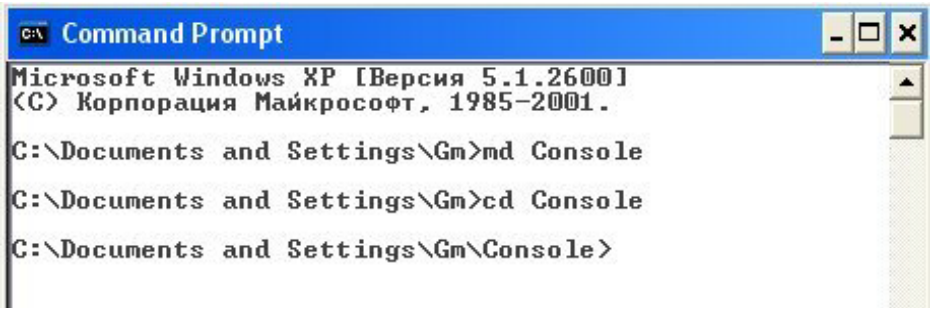


Рис. 6.26. Интерфейс консольного Windows–приложения
Command Prompt (Командная строка)

Следует добавить, что в этом окне функционируют и выполняются практически все команды операционной системы MS DOS.

❽ ***Диалоговое приложение (мастер, Wizard, помощник)*** ведёт пользователя через последовательность шагов к выполнению определенной задачи. Диалоговые приложения обычно активно взаимодействуют с пользователем через набор экранов (или диалоговых окон), посредством которых пользователь делает свой выбор. Хорошим примером диалоговых приложений являются довольно распространенные в среде Windows ***мастера*** MS Office и Windows

мастера (Wizards) (рис. 6.27). Могут являться компонентами GUI и активизироваться щелчком мыши по кнопке-команде, инициализирующей их вызов и выполнение.

❾ *Одноярусным приложением (single-tier application)* называется приложение, реализованное в одном физическом ярусе. Это означает, что такое приложение выполняется как единичный процесс на одной машине. Одноярусное приложение может быть основано на различных логических моделях. Это может быть многоярусная логическая модель, в которой службы различных типов отделены друг от друга. Или же это может быть модель монолитного приложения, в котором нет четкого разделения служб по типам. Для взаимодействия между различными модулями одноярусного приложения используется стандартный вызов процедур. Таким образом, в этих приложениях нет избыточности взаимодействия, присущей распределённым приложениям. Однако производительность такого приложения ограничена возможностями аппаратных средств, на которых оно выполняется. Одно из преимуществ одноярусных приложений заключается в легкости реализации. По этой причине одноярусные приложения достаточно широко распространены.

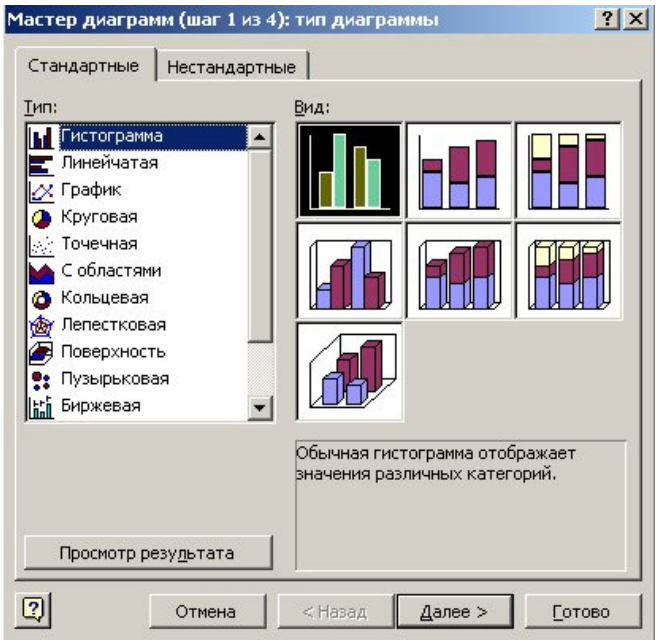


Рис. 6.27.
Мастер диаграмм приложения MS Excel

❶❿ *Двухъярусное приложение (two-tier application)* является хорошим примером распределенного приложения, использующего клиент-серверную модель. Как правило, двухъярусное приложение — это приложение, реализованное в двух физических слоях. Большинство приложений, требующих обращения к базе данных, реализуются как двухъярусные приложения. В двухъярусной модели приложение, выполняющееся на рабочей станции (клиенте), обращается за данными к централизованной базе данных на удаленном компьютере (сервере), поддерживающем множество распределённых клиентов. Такая модель позволяет совместно использовать вычислительные ресурсы и предоставляет централизованную базу данных, однако сопровождение и поддержка клиентских приложений в этой модели все-таки сложнее, чем управление и поддержка централизованных систем прошлых лёт. Однако реализация клиент-серверной модели в трактовке Microsoft COM добавляет ей функциональности и интероперабельности не только во взаимодействии между приложениями разных производителей, но также и между приложениями, которые работают на разных платформах (рис. 6.28).

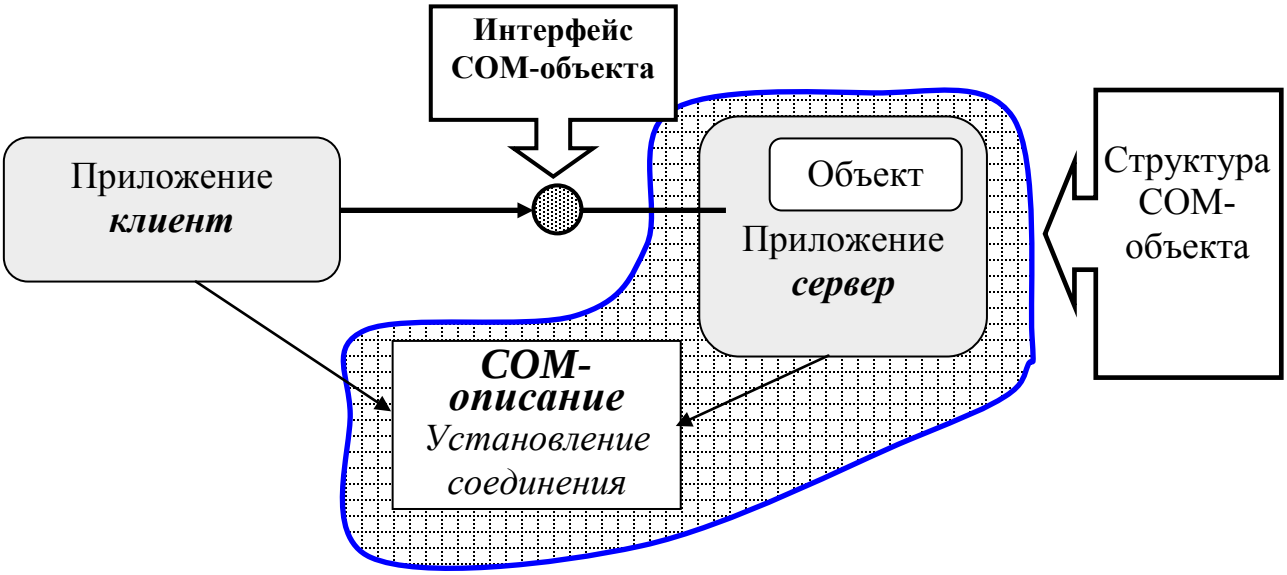


Рис. 6.28. После установления связи с помощью интерфейса COM между клиентом и объектом-сервером, они общаются без дополнительных ресурсов ОС и компьютера

У двухъярусных приложений существенно ограничена масштабируемость, причиной чего, обычно являются ограничения, накладываемые на производительность сервера баз данных. В типичном двухъярусном приложении клиент напрямую соединен с сервером баз данных. При этом от сервера требуется по одному соединению на каждого клиента, каждый из которых, затем держит соединение открытым в течение всего времени жизни приложения. Данный подход, хотя и прост в реализации, не лучшим образом влияет на производительность.

❶❶ В основе архитектуры ***многоярусного приложения*** *(N-tier application*[13]*)* лежит идея разделения приложения на отдельные функциональные компоненты (рис. 6.29). Как правило, приложение проектируется вокруг трёх ярусов служб — пользователя, бизнес-правил и данных — так называемое трёхъярусное приложение. Многоярусное приложение является логическим расширением трёхъярусного. В таком приложении один (или более) из трёх начальных ярусов разбивается на дополнительные ярусы (рис. 6.29). Это добавляет новые уровни абстракции для описания сложных моделей приложений.

❶❷ *Web-**приложения,*** как правило, представляют собой набор Web-страниц, которые отображаются приложением-контейнером – браузером. На начальных этапах своего появления браузеры могли отображать лишь статические Web-страницы, но теперь многие из них поддерживают динамические страницы, что активно используется в Web-приложениях. ***Web-приложения*** объединяют в себе ***Интернет-технологии***, *Web-**технологии*** и ***технологии создания традиционных приложений.*** Web-страницы, отображаемые в браузерах, предоставляют пользователю информацию о результатах работы Web-приложения. Кроме того,

[13] **Application Architecture: An N-Tier Approach - Part 1. (**By Robert Chartier) http://www.15seconds.com/Issue/011023.htm?voteresult=5

допускается локальное размещение Web-приложения в интранете (LAN) или глобальное в Интернете. В свою очередь, Web-страницы предоставляют пользователю информацию о результатах работы Web-приложения.

Рис. 6.29 Современное представление ***многоуровневого*** *(N-tire)* приложения

Большинство Web-приложений являются смесью HTML-страниц (т.е. кодов языка разметки HTML) и встраиваемого в них исполняемого ***на стороне клиента*** программного кода (на языках VBScript, Jscript и др.). То есть это может быть код сценария, содержащегося на странице, коды скриптового языка или же двоичный код приложения или компонента, вызванного со страницы. Комбинировать коды HTML и программный код удается несколькими различными способами. Можно выполнять код только на сервере — такая модель увеличивает число браузеров, работающих с приложением. Можно выполнять код и на клиенте — такая модель уменьшает сетевой трафик и время отклика приложения. Однако не все браузеры поддерживают все возможные модели реализации. Наиболее популярными технологиями для создания Web-приложений, осуществляющих обработку данных ***на сервере***, являются: Active Server Pages (ASP), Internet Server API (ISAPI), и Common Gateway Interface (CGI).

❶❸ ***Совместно работающие приложения*** *(collaborative applications)* — это

распределенные приложения, которые работают вместе. Части совместного приложения обычно сами являются полнофункциональными приложениями, которые можно использовать и независимо друг от друга. Например, совместно работающее приложение может, состоять из приложений, написанных с применением Visual Basic, Microsoft Word и Microsoft Excel, где приложение на Visual Basic использует Automation для доступа к функциональности Word и Excel. В качестве двух других примеров совместных приложений можно привести Lotus Notes и Microsoft Exchange.

❶❹ ***Революционной технологией создания приложений является компоновка их из компонентов. Компонент (component)*** – это фундаментальный строительный блок для разработки распределённых многоярусных приложений. Компоненты достаточно часто базируются на компонентной объектной модели фирмы Microsoft (Component Object Model, COM) или создаются в виде модели независимых программных модулей повторного использования JavaBeans фирмы Sun Microsystems и представляют собой средство формирования функциональности приложений.

Схематически, появление и развитие этой новой концепции построения программ из компонентов, можно представить как переход к взаимозаменяемости, которая свойственна устройствам ПК, из которых он строится, как из кубиков (рис. 6.30)

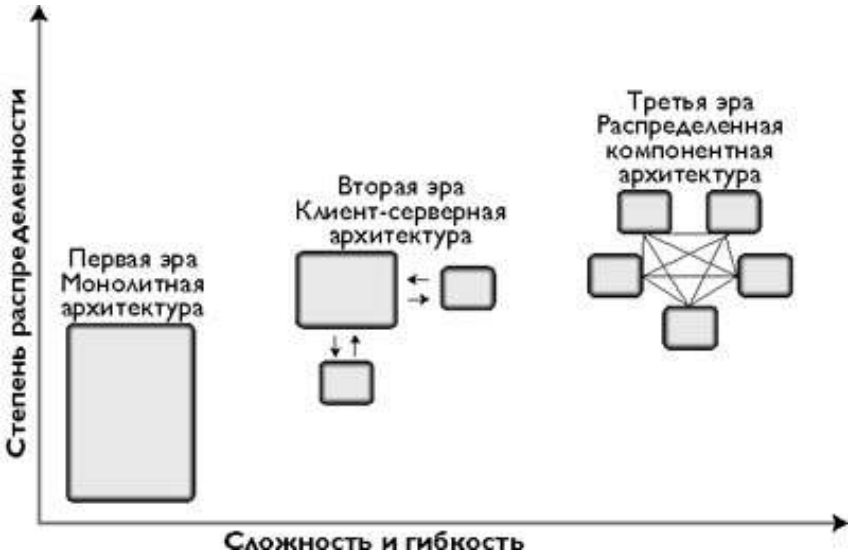


Рис. 6.30. Совершенствование функциональных связей между элементами и компонетами приложений

Так как вся функциональность спрятана внутри компонента, только компонент знает, как именно предоставляется эта функциональность. Приложение же изолировано от всех деталей его реализации. В результате ***любые изменения*** во внутренней организации компонента ***не затрагивают*** остальные части приложения. Услуги и атрибуты компонента, открытые для внешнего мира, называются, соответственно, ***методами*** (methods) и ***свойствами*** (properties) ***компонента*** и могут быть получены ***только через его интерфейсы***.

Проектирование компонента обычно включает следующие шаги:

❶ определение услуг, которые должен предоставлять компонент;

❷ выделение объектов, логическим образом организующих функциональность компонента и специфицирование его внутренних методов;

❸ соотнесение функциональных возможностей компонента его интерфейсам и их именование;

❹ определение типа компонента: внутрипроцессного (in-process) или внепроцессного (out-of-process).

❺ Именование нового компонента с помощью механизма GUID или UUID[14]

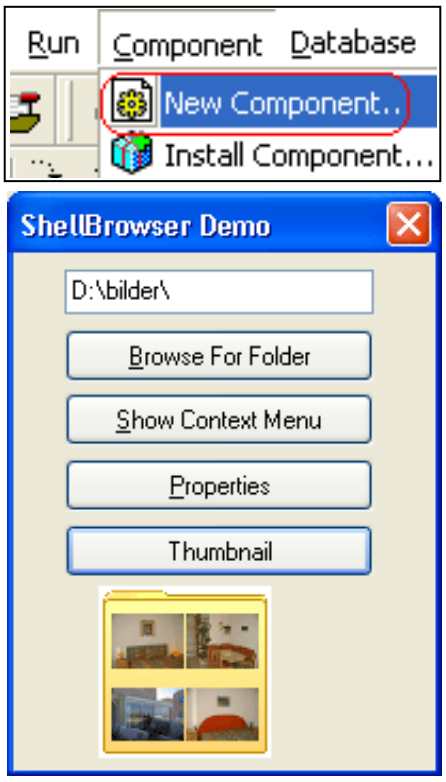


Рис. 6.31. Меню в среде Delphi для создания и подключения компонентов (вверху), а также интерфейс типового компонента для обзора содержимого папок (каталогов) ПК

Компоненты обычно предназначены для повторного использования: грамотно спроектированный компонент можно использовать во многих других приложениях (рис.6.31). Поэтому, имея в своём распоряжении набор подходящих компонентов, можно достаточно быстро собрать приложение с необходимой функциональностью. По технологии Microsoft ***компонент*** структурно представляет собой набор ***классов (classes)*** *COM*, которые организованы в виде отдельных выполняемых программных единиц, а также в виде EXE- или DLL-файлов.

❶❺ ***Компонентами*** нового поколения стали ***Web-сервисы***[15]. Изначально World Wide Web была сетью документов. Web-серверы общались с клиентами с помощью протокола HTTP (Hypertext Transfer Protocol) и пересылали информацию в форме гипертекстовых документов, которые создавались средствами языка HTML (Hypertext Markup Language). Такие документы, как правило, отображаются в браузерах и содержат гиперссылки на другие документы в WWW. Мультимедиа-функции языка HTML также позволяют авторам включать в свои Web-страницы изображения, апплеты (программы на языке Java, которые автоматически загружаются и выполняются на машине пользователя (клиента)), видеоклипы и другие документы в формате HTML.

Совершенствование языков программирования, Internet- и Web-технологий привело к тому, что многие коммерческие сайты начали строиться не на основе

[14] Так как идентификаторы для всего многообразия компонентов (COM-объектов) и их интерфейсов на всех (!!!) ПК во всём мире должны быть глобально уникальными, для них введены имена, называемые GUID (Globally Unique IDentifiers – глобально однозначные (уникальные) идентификаторы). Для создания последних используется программа GUIDGEN.EXE, а для записи самого имени отводится 16-байт (128 бит), дающие возможность записать 2^{128} вариантов имён. Синонимом GUID является UUID (Universally Unique IDentifier – универсально уникальный идентификатор).

[15] Web нового поколения — Web-сервисы. Алексей Федоров. http://www.compress.ru/Temp/1081/index.htm

обычных Web-серверов, а с помощью многозвенной архитектуры, которая подразумевает использование серверов приложений.

Основным отличием обычных Web-серверов от серверов приложений является то, что последние не просто возвращают документ, а ещё и могут обрабатывать запросы пользователей и содержат код, который реализует бизнес-логику. Как правило, серверы приложений генерируют документы динамически, в зависимости от указанных пользователем параметров. Также следует отметить, что применение серверов приложений позволяет создавать масштабированные решения, которые могут одновременно обслуживать большое количество транзакций и, соответственно, и пользователей.

Появление разнообразных мобильных устройств привело к тому, что вместо традиционных браузеров большое количество коммерческих Web-приложений теперь, кроме протокола http, поддерживают и протокол WAP (Wireless Access Protocol). Таким образом, они способны возвращать на запрос информацию не только в стандарте HTML, но и на языке WML (Wireless Markup Language) (рис.6.32).

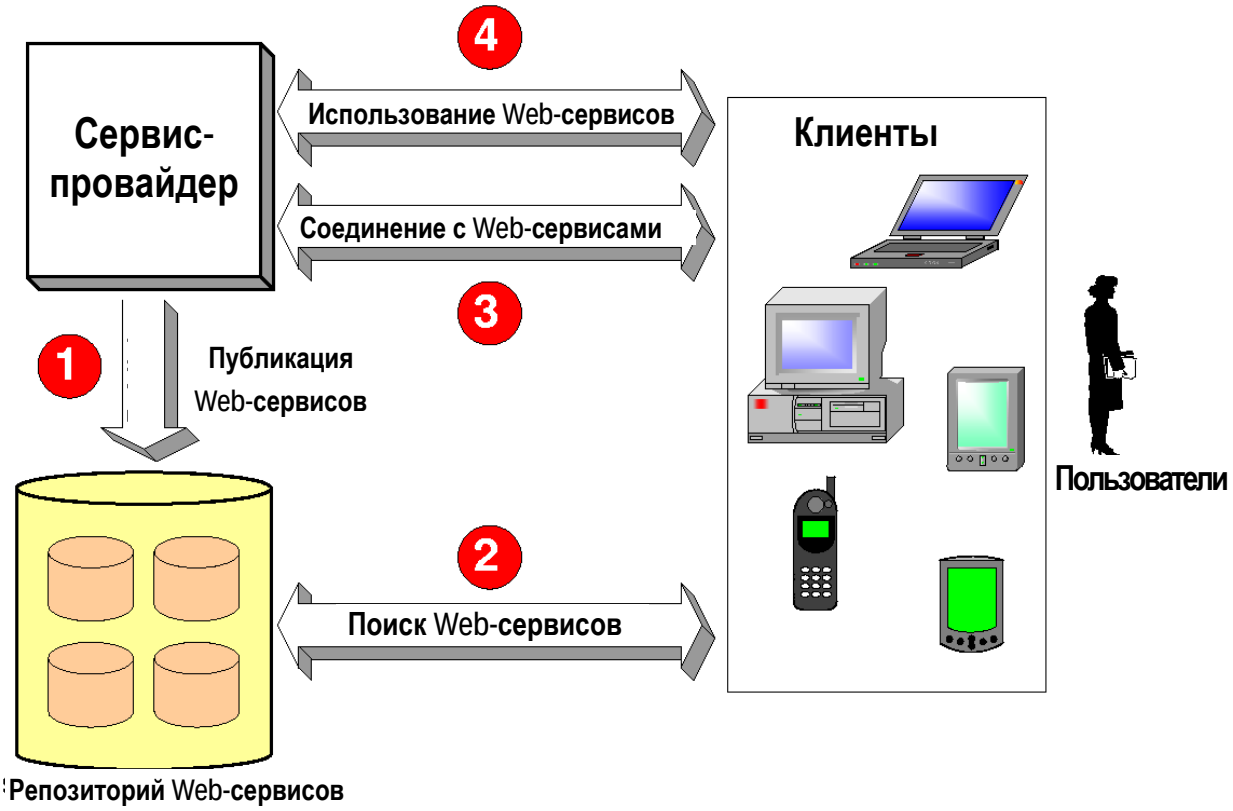


Рис. 6.32.
Процесс работы с Web-сервисами.
1) Публикация Web-сервисов в репозитории на сервере. 2,3,4) Поиск, соединение и использование Web-сервисов мобильными и локальными устройствами.

Естественно, электронная коммерция не может ограничиваться простой обработкой транзакций — следующим логичным шагом в развитии Web (то есть WWW) стала интеграция бизнес-процессов разных компаний. Таким

образом, и появился сервис-ориентированный Web. В его основе лежат две относительно новых технологий — SOAP (Simple Object Access Protocol) и XML (eXtensible Markup Language). Эти технологии определяют сценарий, в соответствии с которым, Web состоит из набора серверов приложений, обменивающихся информацией в формате XML на основе протокола SOAP. Основой сервис-ориентированного Web является ***Web-сервис*** — набор логически связанных функций, которые могут быть программно вызваны через Internet средствами технологии интеграции Web-приложений. Информация о том, какие функции даёт конкретный Web-сервис, содержится в документе WSDL (Web Service Description Language), а для поиска существующих Web-сервисов используются специальные реестры, совместимые со спецификацией UDDI (Universal Description, Discovery and Integration) (рис 6.32).

В общем случае, ***Web-сервисом*** называется активный контент, реализующий некоторую функциональность и содержит данные, расположенные на Web-серверах, которые подключаются в момент использования его внешними приложениями. Web-сервисы полностью независимы от языка и платформы реализации. Внешние приложения взаимодействуют с Web-сервисами с помощью стандартных протоколов и форматов данных.

Естественно, разнообразие устройств, протоколов обмена данными между ними, а также необходимых пользователям функций постоянно расширяют границы применения и технологий создания и использования приложений, решений и компонентов. В таблице 6.3. приведены формы существования и некоторые решения в реализации программных компонентов двух основных платформ: Windows и Sun (J2EE). Из таблицы следует, что сложность взаимодействия между программными компонентами увеличивается и, соответственно, развиваются модели и механизмы конструирования информационных связей между ними.

6.7. Современные технологии создания и использования компонентных приложений, Web-приложений и Web-сервисов

Основным лозунгом фирмы Sun Microsystems, является: "Сеть – это не просто компьютер, это сетевой компьютер". И внутри неё (сети), накоплено множество разных платформ, приложений, решений и программных систем (написанных на разных языках программирования), унаследованных систем и много других компонентов. Это выдвигает на передний план задачу объединения всех этих ресурсов в работоспособный механизм решения полезных задач. Флагманами в области развития моделей использования всех существующих возможностей сетевых структур, являются корпорация Microsoft и фирма Sun Microsystems. Они долгие годы разрабатывают и развивают разнообразные средства обработки информационных ресурсов на всех уровнях архитектур WWW и Internet. В последнее время ими разработаны комплексные технологии и архитектуры Jini™ от Sun Microsystems и .NET™ от

Microsoft с соответствующими средствами программных реализаций компонентов.

Таблица 6.3.

Наименование модульных приложений (компонентов), реализованных средствами компонентных технологий

№ пп	Название компонента	Сущность понятия
1	***Сервис*** (service) или компонент	–1) Компонент, способный выполнять задачу пользователя. –2) WSDL сервис. Набор конечных точек. –3) Смотри Web-сервис.
2	***Портлет*** (стандартный портальный компонент или DDB-компонент)	Реализация некоторого сервиса, запускаемого портальным сервером, которая содержит некоторые данные, набор собственных бизнес-функций, а также стандартное представление на рабочих панелях портала. С точки зрения пользователя, портлет – это небольшое окно на страничке портала в браузере, которое даёт специфические функции или элементы информации, такие как календарь, заголовки новостей и др. С точки зрения разработчика, портлеты являются подключаемыми модулями (фактически – отдельными приложениями), которые разрабатываются для функционирования внутри портлет-контейнера портала.
3	*Assembly* («пакет» или «комплект»)	–1) (в .NET) Новая модель использования приложений, которая упрощает инсталляцию и отслеживание версий. Ключевое понятие этой модели – асембл (assembly). –2) (в .NET) ***асембл (assembly)***, набор ресурсов и типов данных, а также метаданных, которые описывают данные и методы, реализованные в *assembly*. Таким образом, *assembly* – это самоописанный компонент. Основное преимущество таких компонентов в том, что для их использования не требуются никакие другие файлы.
4	***Сервлет*** (servlet)	(В языке Java). Java программа, которая расширяет функции Web-сервера, генерирует динамический контент и взаимодействует с Web-клиентом на основе парадигмы запрос/ответ.
5	***Сервлет-контейнер*** (распределённый)	(В языке Java). Контейнер ***сервлета***, запускающий Web-приложение и который скомпонован (связан) как распределённый. Он также может выполняться в среде множества виртуальных машин Java (Java virtual machine), которые выполняются на одном хосте или на разных хостах.
6	***Апплет*** *(applet)*	(В языке Java). Компонент, который, как правило, выполняется в Web-браузере, но может также выполняться и в других разнообразных приложениях и устройствах, которые поддерживают программную модель взаимодействия апплетов.
7	***Компонент*** (component)	–1) Составная часть распределённого приложения. –2) Компонент, является элементом архитектуры с определёнными (заданными) границами. –3) (В языке Java). Программный модуль уровня приложения, который поддерживается контейнером. Компоненты конфигурируются во время разворачивания. Платформа J2EE определяет четыре типа компонента: ***корпоративные*** (промышленные) ***компоненты «зёрна»*** (enterprise beans), *Web–**компоненты***, ***апплеты*** и ***приложения клиенты***. –4) Компонент является объектом программного обеспечения, который предназначен для взаимодействия с другими компонентами. Он инкапсулирует некоторую функциональность или набор исполняемых функций. Компонент имеет чётко определяемый интерфейс и подчиняется правилам поведения, общим для всех компонентов в данной архитектуре. –5) Компонент является абстрактным набором программных инструкций (команд) и внутреннего состояния, которое обеспечивает преобразование данных через его интерфейс.

Продолжение таблицы 6.3.

№ пп	Название компонента	Сущность понятия
8	***Bean (зерно)***	(В языке Java). Повторно используемый программный компонент (или компонент уровня ***предприятия***). Может служить составной частью при создании приложений.
9	*Java Beans*	(В языке Java) ***Программные компоненты*** - независимые программные модули, которые повторно используются и которые способны взаимодействовать друг с другом.
10	***Контейнер***	(В языке Java) Сущность, которая обеспечивает жизненый цикл управления, безопасности, разворачивания и соответствующие сервисы при выполнении компонента. Каждый тип контейнера (EJB–Enterprise Java Beans, Web, JSP–Java Server Pages, сервлет, апплет и приложение клиент) также обеспечивает компонентно-конкретизованные сервисы.
11	***Приложение***	(В языке Java). Собранное в момент выполнения из отдельных компонентов, соединённых через сеть в отдельной конкретной среде выполнения, как правило, располагаемое на разных платформах. Распределённые приложения поддерживают следующие модели: двухъярусную (клиент/сервер), трёхъярусную (клиент/промежуточное ПО (middleware)/сервер) и многоярусную (клиент/ множественное промежуточное ПО/множество серверов).
12	***Модуль***	(В языке Java). Программный модуль, который состоит из одного или более компонентов J2EE, которые имеют одинаковый тип контейнера и дескриптора (признака) разворачивания. Существует три типа модулей: EJB (Enterprise Java Beans), Web и приложение клиент.
13	*COM, DCOM*	Открытая архитектура для кроссплатформенных разработок клиент/серверных приложений, которая лежит в основе технологий ActiveX, DirectX и OLE 2.0. Спецификация, модель и технология корпорации Microsoft, которые предназначены для построения и розработки компонентов программного обеспечения и их интерфейсов. COM устанавливает абстракции и правила, необходимые для определения объектов и их интерфейсов, которые реализуются. В её состав входит также программное обеспечение, которое реализует все ключевые функции. Такие компоненты легко объединяются в программы или могут быть добавлены к существующим программам, чтобы придать им бóльшую функциональность. Компоненты пишутся на разных языках (больше всего при этом используются языки C++ или C#). COM сервер, как правило, является .DLL или .EXE файлом.
14	***Агент***	–1) Устройство и/или программа, установленные в элементах компьютерной сети для централизованного управления этими элементами и всей сетью. –2) Программа, которая действует от лица другого субъекта, сущности или процесса.
15	*Web-**сервис** (Web service)*	–1) Web-сервисы являются новой Интернет-парадигмой, незаисимой от платформ и языков программирования. Web-сервисы реализуются в виде автономных, модульных приложений, которые могут быть описаны, опубликованы, размещены и быть вызванными через электронную вычислительную сеть для создания новых продуктов и сервисов. –2) Web-сервис является программной системой, которая разработана для поддержки интероперабельности межмашинного взаимодействия в компьютерной сети. Он имеет интерфейс, описанный в машинно-обрабатываемом формате (как правило, WSDL). Другие системы взаимодействуют с Web-сервисом заданным способом, соответствующим описанию, который использует SOAP-сообщение, обычно передаваемое с использованием HTTP с XML сериализацией в привязке к другим Web-ориентированным стандартам.

Технология Jini, основанная на языке программирования Java и компонентах JavaBeans, разрабатывалась с целью создания системы, которая бы не вызывала к себе много внимания при эксплуатации, успевала бы за постоянными изменениями и необходимостью масштабирования, а также обеспечивала постоянную доступность сервисов посредством Internet 24 часа в сутки и 7 дней в неделю (24 х 7). Безопасность системы и конфиденциальность информации, передающейся в сети, достигается за счёт распределённой системы безопасности. Наращивание систем возможно за счёт добавления новых, наследования и замены старых сервисов, также доступных при посредничестве Internet.

Следует отметить, что в отличие от технологий Java и Jini, краеугольным камнем программной модели Microsoft .NET является технология Web-сервисов, базирующаяся на среде .NET Framework, которая в свою очередь основана на следующих базовых концепциях:

❶ независимой от языка программирования средой выполнения (Common Language Runtime);

❷ библиотеке классов .NET (.NET Class Library);

❸ языке-посреднике Microsoft Intermediate Language (MSIL);

❹ языках, которые поддерживают .NET.

Структура .NET Framework приведена на рис. 6.33.

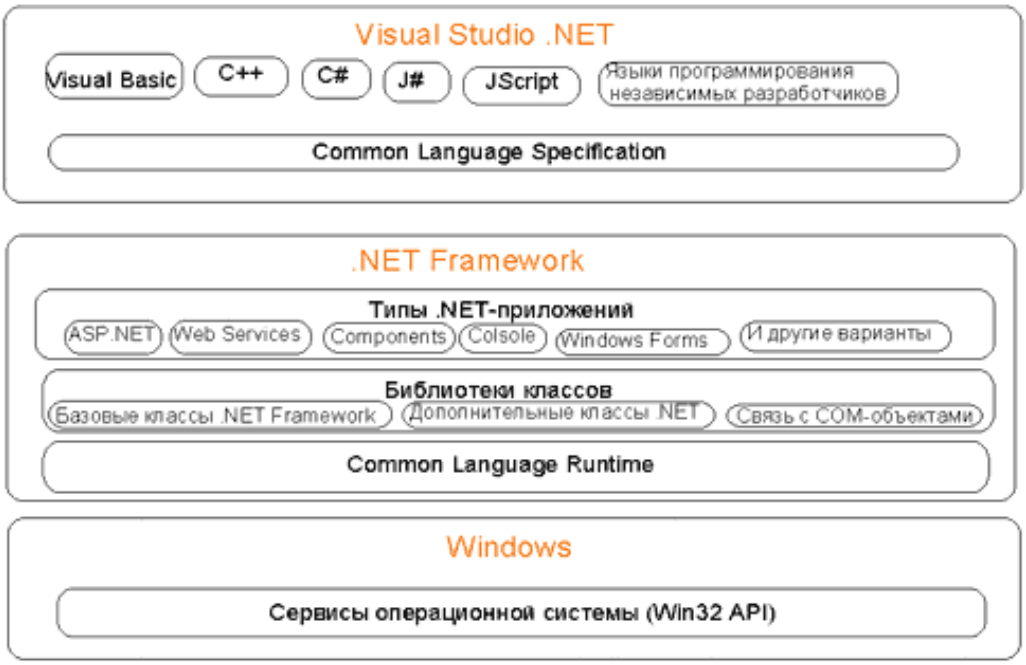


Рис. 6.33.
Структурна схема .NET Framework

Как видно, речь идёт фактически о единой среде выполнения программ и поддержки процессов их разработки. Именно здесь собраны базовые классы для всех языков программирования, которые реализованы в виде библиотеки ядра System, а также большого количества (более 20) специализированных

библиотек с именами System.Data, System.XML и т. д. Над ними располагается набор средств формирования приложений для выполнения модулей различного типа (кстати, общий для разных языков исполняемых модулей).

Интересно рассмотреть некоторые черты вышеуказанных технологий в сравнении (см. табл.. 6.4.)

Таблица 6.4.

Сравнение основ технологий Jini и .Net[16]

Технологии .NET	**Технологии Jini**	**Разница**
C#	Java	C# и Java являются C (си)-подобными языками. При построении некоторых компонентов в C# использовался пакет JavaBeans. Основной разницей является то, что C# запускается исключительно под Windows, а Java работает на любой платформе, где есть JVM (Java Virtual Machine). Оба языка прекомпилируются в just-in-time (JIT) байт-код, который на каждой платформе перекомпилируется во время выполнения в подходящий для данной платформы код и выполняется.
Active Server Pages (ASP+)	Java Server Pages (JSP)	ASP+ будет использовать код, написанный на Visual Basic, C# и на других языках программирования. JSP, в свою очередь, использует код, который написан, исключительно, на Java.
Interface Language (IL) Common Language Runtime	Java Virtual Machine (JVM) и CORBA	Ядро технологии .NET, Common Language Runtime, позволяет запускать код, который написан на любом языке программирования, в среде Windows или других операционных системах, а также использовать совместный набор компонентов и объектов. JVM позволяет запускать, прекомпилированный с Java, байт-код на любой платформе. CORBA позволяет использовать совместный набор объектов, написанных на разних языках программирования, на любой платформе.
ADO+ иі SOAP Web Services	JDBC, EJB, JMS и Java XML Libraries (XML4J, JAXP)	ADO+ использует модель обмена данных, основанную на модели XML и SOAP, которая позволяет создавать надстройки над протоколом HTTP. В Jini эта модель реализована по-другому: технология оставляет возможнность разрабатывать транзакциіі самим программистам с помощью бибилиотек и компонентов языка Java.

С одной стороны, допуская мультиязыковость программного кода, концепция .NET привлекает наилучшее, что есть в таких широко распространенных языках программирования, как Perl, Cobol, Eiffel, Python и другие. С другой стороны, это усложняет работу над одним проектом группе

[16] Jini[tm].NET. *Александр Волоха (alex_frost@ukr.net),* www.ixbt.com/editorial/ jini-vs-net.shtml

програмистов, которые пишут отрезки кода на разных языках программирования. В Jini всё проще. Зжесь можно обойтись исключительно языком программирования Java.

В последнее время появляются сообщения о розработке языков, которые будут наследовать лучшее из Java и дополняться функциями, позаимствованными из других языков. Среди таких можно назвать технологии, возникшие на стыке двух языков. Это JPython, PERCobol, Tcl/Java project, Eiffel-to-JavaVM. Эти языки, возможно, смогуь усилить и без того сильную позицию языка Java среди разработчиков.

Несколько слов следует сказать о среде разработки компонентов и приложений Microsoft Visual Studio.NET. Это набор средств быстрой разработки приложений RAD для создания следующего поколения Web-приложений и Web -сервисов на базе XML (рис.6.34).

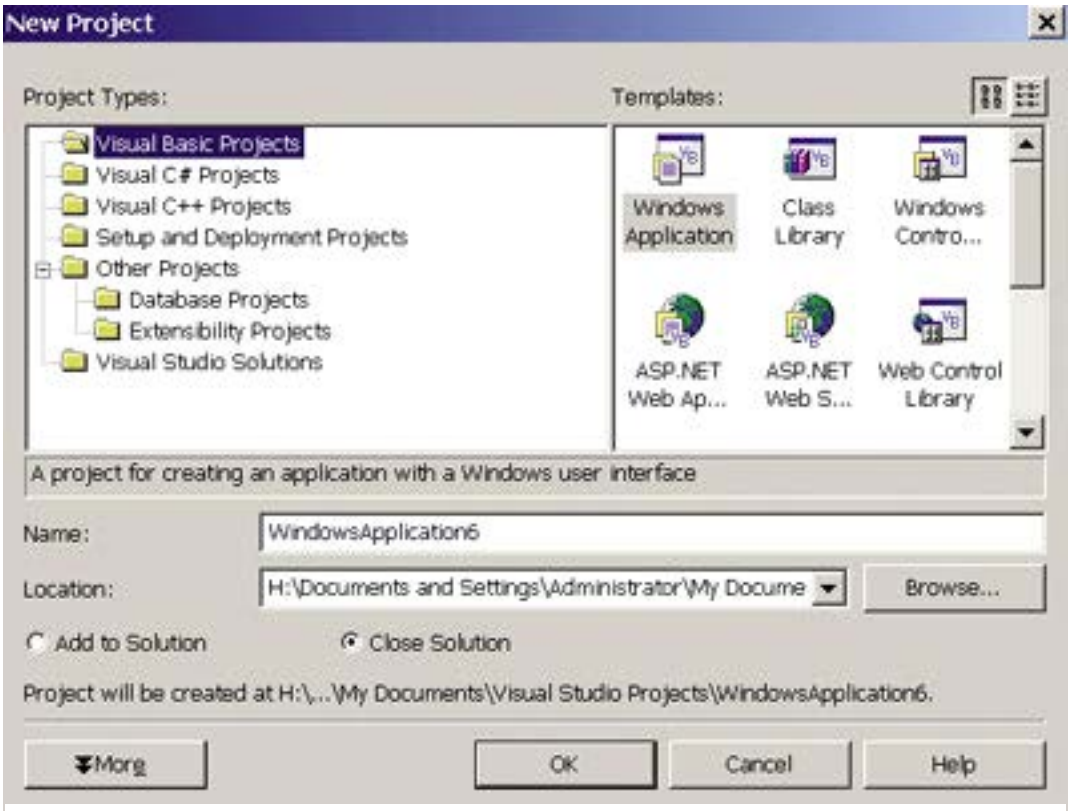


Рис. 6. 34.

Выбор языка и типа проекта в RAD IDE Visual Studio.NET

Visual Studio.NET включает единую IDE с поддержкой функций RAD для построения Web -приложений и компонентов промежуточного уровня, отвечающих за бизнес-логику, а также RAD XML-конструкторы для работы с данными. Microsoft предлагает пять языков "собственного приготовления": VC++, C#, VB.NET, Jscript и J#.

В самую позднюю предыдущую версию VS.NET включены первые три. Независимые поставщики также могут подключать свои средства программирования к среде VS.NET. До настоящего времени известно более

30-ти систем на базе разных языков (Cobol, Fortran, Perl, Python и т. д.), которые поддерживают создание .NET-приложений.

Visual Studio.NET даёт возможность разработчикам быстро проектировать Web-приложения широкого спектра применения для любых устройств и любой платформы. Visual Studio.NET полностью интегрируется с .NET Framework, с обеспечением поддержки множества языков программирования и автоматически решают множество распространённых задач программирования, благодаря чему разработчики освобождаются от рутинной работы и могут быстро создавать Web-приложения на том языке, которому отдаёт предпочтение.

Вопросы.

1.	Что собой представляет программа?
2.	Какие названия имеют последовательности операторов языков программирования в разном контексте?
3.	Сколько Вы можете вспомнить названий программ и их разновидностей ?
4.	Чем отличаются между собой программа, подпрограмма и функция?
5.	В каких средах пишутся программы?
6.	Для чего предназначены API-интерфейсы?
7.	Какие основные системные функции прикладного программного интерфейса (API)?
8.	Какие преимущества даёт использование RAD-систем?
9.	В каких архитектурных средах исполняются программ?
10.	Чем архитектура компьютера отличается от архитектуры сети?
11.	Какие основные составляющие архитектуры Фон-Неймана?
12.	Назовите основные архитектуры, которые существуют в реальных вычислительных системах?
13.	Какие основные этапы разработки приложения или решения?
14.	Какие существуют типы приложений, в зависимости от архитектуры, в которой они работают?
15.	В каких формах существуют компонентные приложения, как они называются и от чего зависит такое их большое количество?
16.	Какие современные вычислительные и программные архитектуры наиболее популярны?

Сообщение – это передача информации от одного объекта к другому, в расчёте на то, что в результате будет выполнена некоторая деятельность. Сообщением может быть сигнал или вызов операции.

7. ЯЗЫК UML И ЕГО ИСПОЛЬЗОВАНИЕ

7.1. Причины появления объектно-ориентированного подхода и языка UML

Программирование во все времена развития компьютеров было непростым делом. Особенно это касалось разработки больших программных систем. Например, Джозеф Фокс, один из руководителей фирмы IBM (в 1969-1977 гг.), приводил пример разработки информационной системы по управлению полётами космических кораблей типа Аполлон/Скайлэб (рис. 7.1). Её общая стоимость составляла 209 млн. долл.(!), а разработкой занимались 700 программистов в продолжение 7-ми лет.

Рис. 7.1. Пункт управления полётами кораблей Аполлон (слева)
и момент стыковки космических кораблей Аполлон и Союз (справа)

При этом количество машинных команд в конце работы над системой достигла 23-х млн., а разработка проекта велась на языках программирования Фортран и Assembler. Те, кто хотя бы раз запрограммировал какой-либо алгоритм на языке Assembler’а поймут что значит отобразить управление компьютером в пошаговом режиме (рис. 7.2). Кстати, в операционной системе Windows ″образца″ 1998 года по оценкам специалистов общее количество команд оценивалось в таких же пределах, как и в проекте Аполлон/Скайлэб!

Поэтому неудивительно, что нареканий на работу подобных больших программных разработок всегда хватало. По оценкам экспертов, от 40 до 60 процентов проектов по созданию крупных программных систем в разных предметных областях проваливаются и не доводяться до свого логического конца. И недаром в новой модели разработки программных решений корпорации Microsoft MSF (см. подраздел 6.5), одной из фаз разработки является оценка рисков процесса их создания[1].

Следует чётко представлять, что на сложную, многофакторную, сложнопрогнозируемую работу больших коллективов специалистов разных направлений (аналитиков, управленцев, системотехников, программистов, системных администраторов, администраторов баз данных, технических писателей, тестеров, сборщиков компонентов и многих, многих других) сильно влияют факторы, которые специалисты выделили в систему следующих 5-и уровней ***сложности программирования*** (рис.7.2):

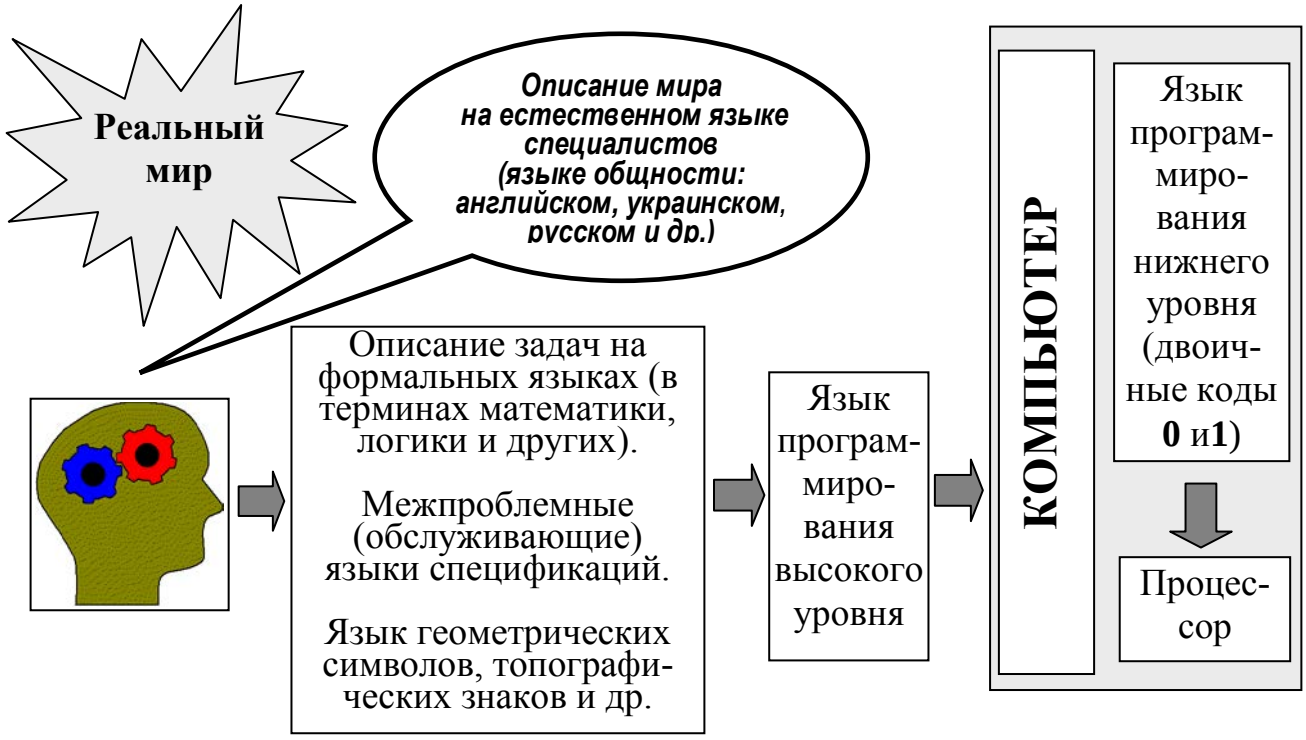


Рис. 7.2. Процесс преобразования (трансформации) данных и объектов окружающей природы в компьютерное представление

❶ сложность самой решаемой задачи и её описания;

❷ сложность языка программирования или комплекса тех языков, которые задействованы в сложном проекте;

[1] Управление рисками (risk management)– это одна из ключевых дисциплин Microsoft Solutions Framework ® (MSF). MSF видит в изменениях и неопределенности, возникающей вследствие этого, неоъемлемую часть жизненого цикла информационных технологий. Дисциплина управления рисками в MSF отстаивает превентивный подход к работе с рисками в условиях такой неопределённости, непрерывное оценивание рисков и использование информации о рисках в рамках процесса принятия решенмй на протяжении всего жизненного цикла проекта. Данный шестишаговый процес включает выявление и анализ рисков; планирование и реализацию стратегий по их профилактике и смягчению возможных последствий; отслеживание состояния рисков и извлечение уроков из приобретённого опыта.

❸ сложность структуры среды выполнения программы (архитектура аппаратных и программных средств), которая в настоящее время является существенно сетевой и распределённой;

❹ организация технологического процесса коллективной розработки и создания программного обеспечения (программных продуктов);

❺ стремление к достижению универсальности и эффективности алгоритмов, программных компонентов и методов организации структур данных (последние всегда распределены и хранятся в больших хранилищах).

От ***свойств сложности*** нельзя избавиться, но можно изменить характеристики их проявлений путём применения соответствующих ***приёмов*** и ***методов управления*** или ***организации***.

Вместе с тем, ещё одним существенным фактором сложности разработки программ на современном этапе, является ***постоянное усложнение взаимодействия*** элементов архитектуры (структуры) сетевых, распределённых информационных систем и ***интефейсов*** взаимодействия их компонентов, в том числе и беспроводных! Поэтому создателям ИС приходится вести их разработку в трёх областях (направлениях):

1.Создавать средства обеспечения всех необходимых пользовательских интерфейсов существующих и вновь разрабатываемых технических и программных компонентов.

2.Разрабатывать архитектуру построения и взаимодействия аппаратных и программных компонентов новой информационной структуры.

3.Проектировать и реализовывать собственно программное обеспечение (новый программный код), объединяющее всё это многообразие сущностей в единый работоспособный информационно-компьютерный комплекс.

По мере развития дисциплины программирования, её методики управления сложностью программных проектов отталкивались от применения широко известного фундаментального принципа управления сложными системами, – ***devide et impera*** (разделяй и властвуй, *лат.*). В соответствии с первой частью этого принципа, при проектировании сложной программной системы проводится ***алгоритмическая декомпозиция*** решаемой задачи. Целью декомпозиции является представление разрабатываемой системы в виде нескольких взаимодействующих подсистем (модулей или блоков), каждый из которых легче детализировать, реализовать и проверить независимо от других. Претворение в жизнь второй части принципа полностью ложится на конкретных разработчиков. При таком подходе исполнителям необходимо держать в голове информацию о значительно меньшем количестве деталей. А при разработке больших программных систем легче разделить фрагменты основной программы между участниками и управлять этим процессом. Для дальнейшего повышения продуктивности работы программистов и программистских коллективов были разработаны методы структуризации элементов кода программ, структуры программ и программных проектов. ***Следует чётко представлять, что все новшества в разработке программ***

немедленно претворялись не только в новых версиях старых языков, но и в абсолютно новых языках программирования! (См. гл. 5,6).

Структурный подход в программировании впервые появился с появлением первых подпрограмм, процедур и функций, которые писались, в так называемом процедурном (процедурно-ориентированном) стиле (рис. 7.3).

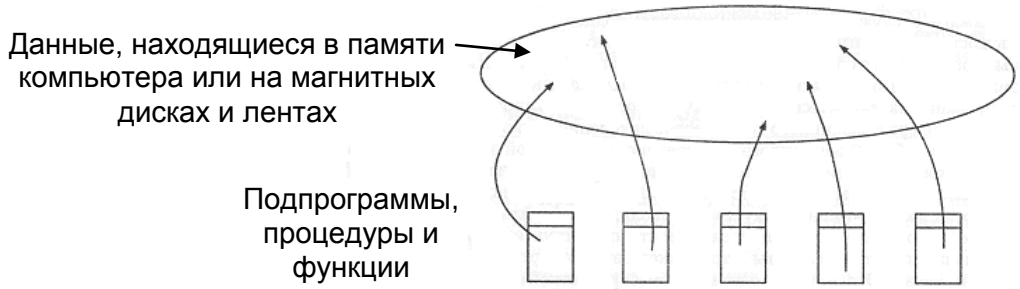


Рис. 7.3. Доступ к данным при работе программ в процедурно-ориентированном стиле

При этом, исполнитель просто определял в программе переменные и константы встроенных (или базовых: ***integer***, ***real***, ***boolean*** и др.) типов, встроенных в язык программирования, сохранял данные в памяти компьютера и описывал или использовал алгоритм их обработки. Теоретическое обоснование структурный подход получил в начале 70-х годов в работах Е. Дейкстры, А.П. Ершова, Е. Йодана, Н. Вирта, Е. Брукса, а также многих других теоретиков и практиков программирования. Дальнейшее развитие структурного подхода (по мере увеличения программных систем и количества разработчиков, одновременно принимающих участие в выполнении одного проекта) привело к началу использования ***модульного программмирования***. Оно предусматривает декомпозицию прикладной задачи в виде иерархии модулей. Специализация модулей по видам обработки данных и наличие у них каких-либо данных определённых типов – являлось теми свойствами, которые отображают генетическую связь модульного и объектно-ориентированного программирования (рис.7.4).

Понятно, что важнейшими инструментами, формирующими структуру представлений и образ мышления производителей программного обеспечения, в которых находят отображение практически все аспекты эволюции технологий программной индустрии, являются ***языки программирования***. Язык программирования с самого начала ориентирован на тип компьютера (компьютерную платформу) и содержит набор типизированных базовых данных, операторов, операций, функций и других операционных единиц языка, которые достаточно просто могут быть переведены в инструкции (команды) управления аппаратным и программным обеспечением компьютера. Язык программирования, как правило, ориентирован на программиста и предоставляет средства для моделирования в виде программы выбранных объектов, их свойств и поведения при решеннии прикладных задач в некоторой предметной области.

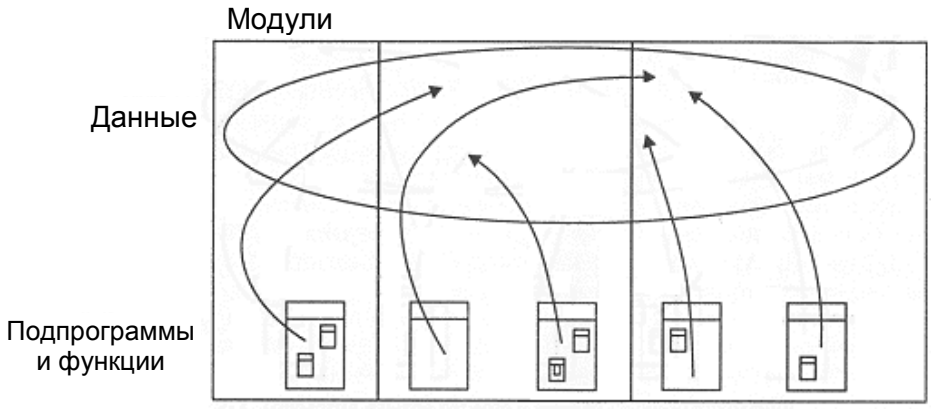


Рис. 7.4. Работа программ с данными в модульно-ориентированом стиле

Развитие языков программирования в направлении повышения ефективности составления прикладных программ привёл к разделению языков на следующие 5 уровней:

❶ Низкий уровень (машинно-ориентированные языки – языки асемблера)

❷ Высокий уровень (процедурно-ориентированные языки: FORTRAN, ALGOL, PL/1, Pascal). Позднее некоторые из них были доработаны до модульно – ориентированного уровня в более поздних версиях. Сюда же относятся языки так называемого третьего поколения (3GL) (см. главу 5).

❸ Уровень задачи, которая решается (проблемно-ориентированные языки – GPS, SQL). Сюда же относятся языки четвертого поколения (4GL) (см. главу 5)

❹ Объектный уровень (объектно-ориентированные (ОО) языки – Smalltalk, C++, Delphi Object Pascal, Java, Python и т.д.).

❺ Компонентный уровень (компонентно-ориентированные языки – C#, C++, Java и т.д.)

Результатом постоянного развития языков, в которых обобщение понятия «тип данных» стали классы объектов (например, в языке C++, Java) или объектные типы (язык Pascal), которые могут содержать в качестве элементов не только данные определённого типа, но и методы их обработки (функции и процедуры) (рис. 7.5).

Таким образом, по мере развития технологий программирования и в программах, и в ***типах данных*** всё адекватнее начала отображаться структура решаемой прикладной задачи и осуществляться соответствующая интеграция данных и программ в модулях и компонентах. По мере повышения уровня абстракции в описании структур данных, при проектировании новых приложений программисты все дальше отходили от фундаментальных (базовых) ***встроенных*** типов данных (то есть символьных, целых, действительных (реальных) и логических). Одновременно с этим, языки программирования пополнились средствами, необходимыми для описания новых объектных структур. Развитие идей абстрагирования и модульности привело к появению в программировании ***объектно-ориентированного подхода***

(ООП). И это понятно, ведь и человек обычно мыслит образами и объектами. Он знает их свойства и манипулирует ими, в соответствии с определёнными событиями.

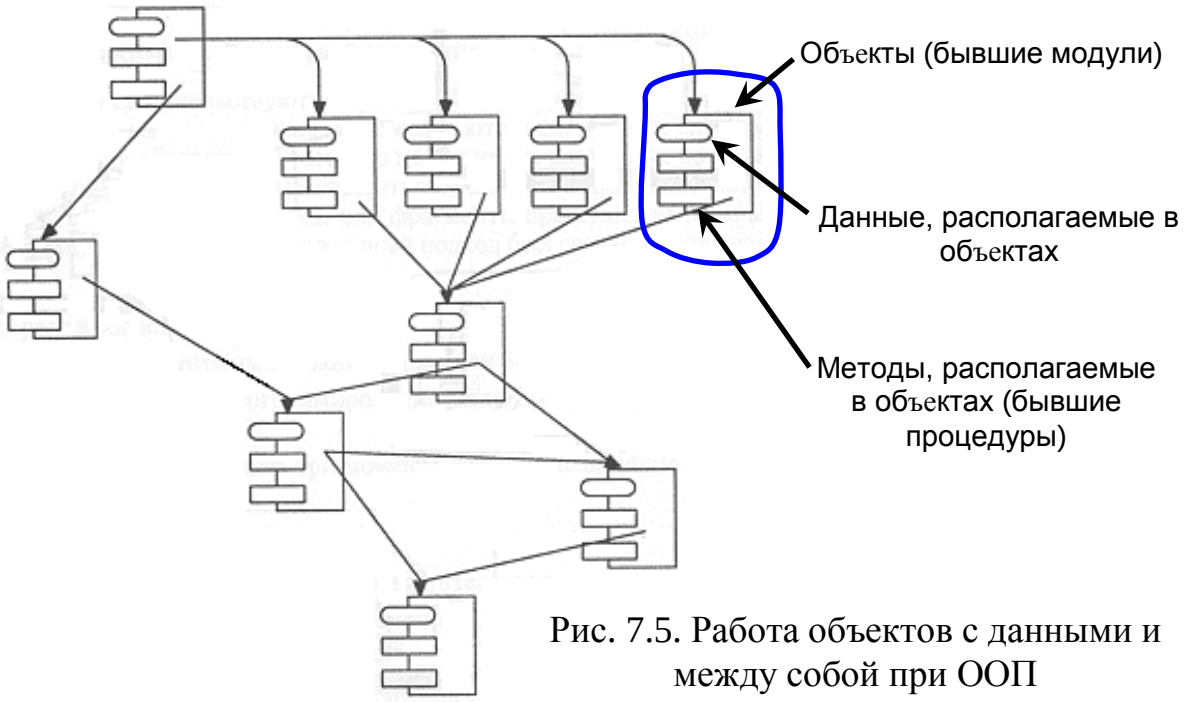


Рис. 7.5. Работа объектов с данными и между собой при ООП

К понятию ООП имеет отношение целый набор следующих концепций:

❶ моделирование событий реального мира;

❷ наличие типов данных, определяемых пользователем;

❸ сокрытие деталей реализации;

❹ возможность многократного использования кода благодаря наследованию;

❺ интерпретация вызовов функций на этапе использования.

Кроме того, в объектно-ориентированном подходе всегда присутствуют ***три типа данных*** и ***пять операционных характеристик***, которые являются ***наиболее важными базовыми элементами и операциями***. Типы данных включают: ***объект***, ***класс*** и ***экземпляр***. Пять характеристик объектно-ориентированной системы содержит: ***абстракцию*** (abstraction), ***наследование*** (inheritance), ***инкапсуляцию*** (encapsulation), ***полиморфизм*** (polymorphism) и ***динамическое связывание*** (dynamic binding). Эти элементы создают мощную среду для разработки приложений.

Согласно ООП, объекты, которые идентичны, за исключением своих состояний, во время выполнения программы, группируются вместе в "***классы объектов***". Так и появилось ключевое слово ***класс***. Создание абстрактных типов данных (классов) – это основополагающая концепция в объектно-ориентированном программировании (рис. 7.6).

Абстрактные типы данных работают почти так же, как и встроенные (базовые) типы данных: вы можете создавать переменные этого типа (которые называются ***объектами*** или ***экземплярами***, если говорить объектно-ориентированным языком) и манипулировать этими переменными (это

называется ***посылкой сообщения*** или ***запросом)***. Вы посылаете сообщение и объект его обрабатывает и пытается выполнить. Поэтому они могут быть представлены как уникальные сущности в компьютерной программе.

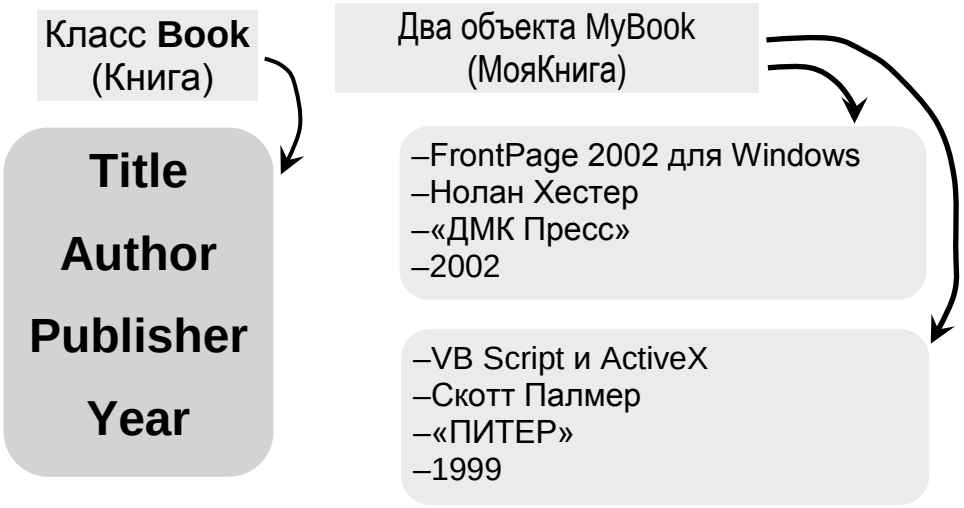


Рис. 7.6. Объекты MyBook, являющиеся реализацией класса Book, каждый из которых имеет своё название (Title), автора (Author), издательство (Publisher) и год издания (Year)

Так как класс описывает набор объектов, имеющих идентичные характеристики (элементы данных) и черты поведения (то есть функциональность), класс реально является типом данных, потому что число с плавающей точкой типа *Real* (например, в Турбо Паскале), также имеет набор характеристик и черт поведения. Разница состоит в том, что программист определяет класс исходя кождый раз из новой реальной проблемы, чтобы представить соответствующий информационный блок для сохранения в компьютере. Следовательно, программист расширяет язык программирования, добавляя спецификации новых типов данных, которые ему необходимы для решения задачи.

Таким образом, программа позволяет адаптировать себя к языку проблемы путём добавления ***любых*** новых типов объектов. Так что, когда Вы читаете программный код, описывающий решение, реализованное на объектно-ориентированном языке, то фактически читаете слова, описывающие проблему. Это более гибкая и мощная абстракция языка, чем была в тех, что разрабатывались ранее. ***Поэтому ООП позволяет Вам описать проблему в терминах проблемы, а не в терминах компьютера, где работает решение (программа).*** Каждый объект в программе выглядит, как маленький компьютер (компонент). Он имеет состояние и может работать так, как Вы распорядитесь (рис. 7.7). Алан Кей, автор и разработчик первого объектно-ориентированного языка программирования SmallTalk (рис.7.8), суммирует пять основных его характеристик.

❶ ***Всё есть объект.*** Следует думать об объектах, как об особых переменных. Они сохраняют данные, но Вы можете "выполнить запрос" к такому объекту, попросив его самого выполнить необходимую операцию. Теоретически Вы можете взять любой умозрительный компонент в проблеме, которую Вы хотите решить (собаку, дом, услугу и т.д.) и представить его как объект в вашей программе.

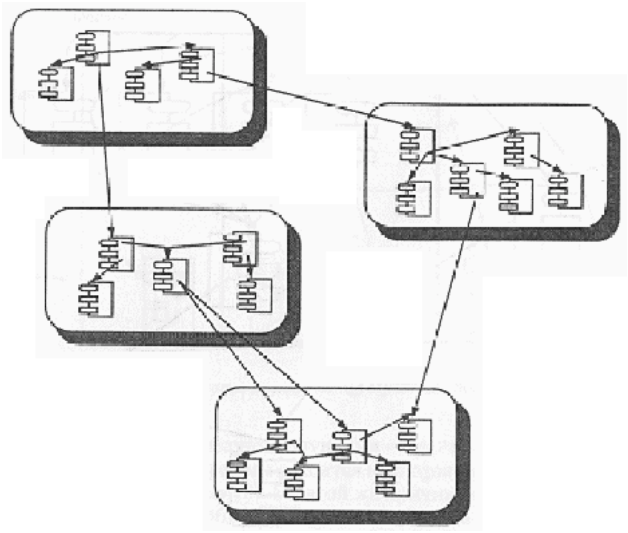


Рис. 7.7. Взаимодействие больших объектов (компонентов), содержащих внутри себя другие объекты

❷ ***Программа – это группа объектов, которые указывают друг другу что делать, обмениваясь сообщениями.*** Чтобы сделать запрос к объекту, Вы "посылаете сообщение" этому объекту. То есть, Вы можете думать про сообщение, как про запрос на вызов функции, которая принадлежит определенному объекту.

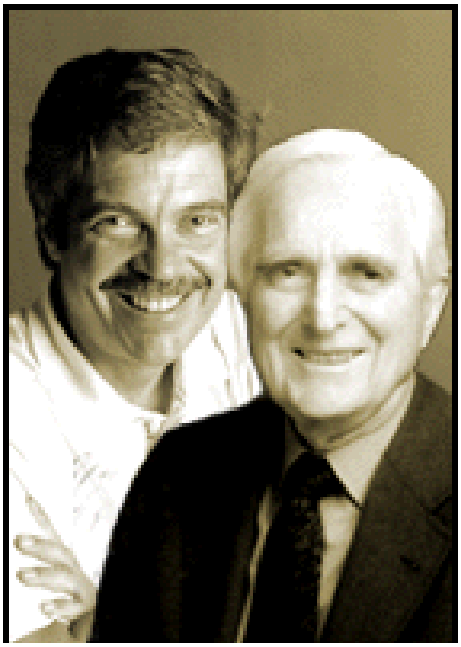


Рисунок 7.8. Автор первого ОО языка программирования Smalltalk Аллан Кэй (позади) и изобретатель первого манипулятора «мышь» Дуг Энгельбарт (Doug Engelbart)

❸ ***Каждый объект имеет свою собственную память, отличную от других объектов.*** Другими словами, Вы создаёте объект нового вида, включая в конструируемый пакет объектов, кроме новых, уже существующие объекты (т.н. компоненты повторного использования). Поэтому, Вы можете построить сложные связи, упрятывая их позади собсьвенных простых объектов.

❹ ***Каждый объект имеет тип.*** Другими словами, каждый объект представляется ***экземпляром класса***, где "***класс***" - это синоним "***типа***". Большинство важных отличий характеристик класса состоит в том, "***Какие сообщения Вы можете ему посылать***?"

❺ ***Все объекты определённного типа (т.е. являющиеся реализацями одного класса) могут принимать одинаковые сообщения.*** Так как объект типа "круг" является объектом типа "форма", круг гарантированно будет принимать сообщения от формы. Это означает, что Вы можете писать код, который общается с формой и

автоматически управляет всеми объектами, отвечающими описанию класса формы. Это представляется одной из наиболее полезных концепций ООП.

Понятно, что программирование было бы очень легким делом, если бы простое объединение объектов решало любую проблему. Сами проблемы становятся всё масштабнее и сложнее. Должна была появиться дисциплина не только ОО-программирования, но и ОО-проектирования. Такой дисциплиной и должен был стать язык UML.

7.2.Моделирование сложных информационных систем

Между тем, современный рынок программного обеспечения диктует жёсткие требования к информационным системам, которые разрабатываются. Системы должны быть надёжными, высокопродуктивными, быть гибкими к внесению изменений, обеспечивать возможность масштабирования и наращивания функциональности, быть интероперабельными в сетевой среде взаимодействующих разнообразных платформ и языков программирования. Кроме того, разработка должна вестись быстро, эффективно и с минимальными затратами.

В связи с этим, одной из важнейших задач разработки становится качественное проведение этапов анализа и проектирования, обеспечивающих создание модели функционирования программной системы, которая может заложить твердый фундамент для дальнейшей ее программной реализации с целью выполнения вышеперечисленных требований. Разработка модели сложной программной системы перед непосредственной ее реализацией является неотъемлемой частью всего проекта, так же как чертеж является основой для постройки большого здания. Построение модели необходимо также и потому, что невозможно охватить с первого взгляда, как всю сложную систему в целом, так и многие отдельные её функциональные части (рис. 7.9).

Таким образом, с развитием объектно-ориентированных (ОО) языков[2] и технологий ОО программирования стали развиваться и общие объектно-ориентированные методы разработки ПО. Действительно, бессмысленной выглядит ситуация, когда ПО спроектировано с помощью структурного метода, а реализовано в объектно-ориентированном стиле. Ведь главной задачей начальных этапов разработки ПО, непосредственно предваряющих программирование, является спецификация предметной области в терминах, удобных для дальнейшего применения в процессе разработки. В этом случае, осуществляется перевод информации из того вида, в котором она существует в сознании специалистов предметной области (инженеров-телефонистов – при разработки ПО для телефонной станции, инженеров-гидрогеологов – при разработке геоинформационных систем расчёта и представления подземных водных ресурсов, инженеров-железнодорожников – при создании систем автоматизации железнодорожных перевозок и т.д.) на язык программистов и программных систем) (рис. 7.10).

[2] В 1972 году была создана первый ОО язык SmallTalk, а в 1980-м – ОО язык C++.

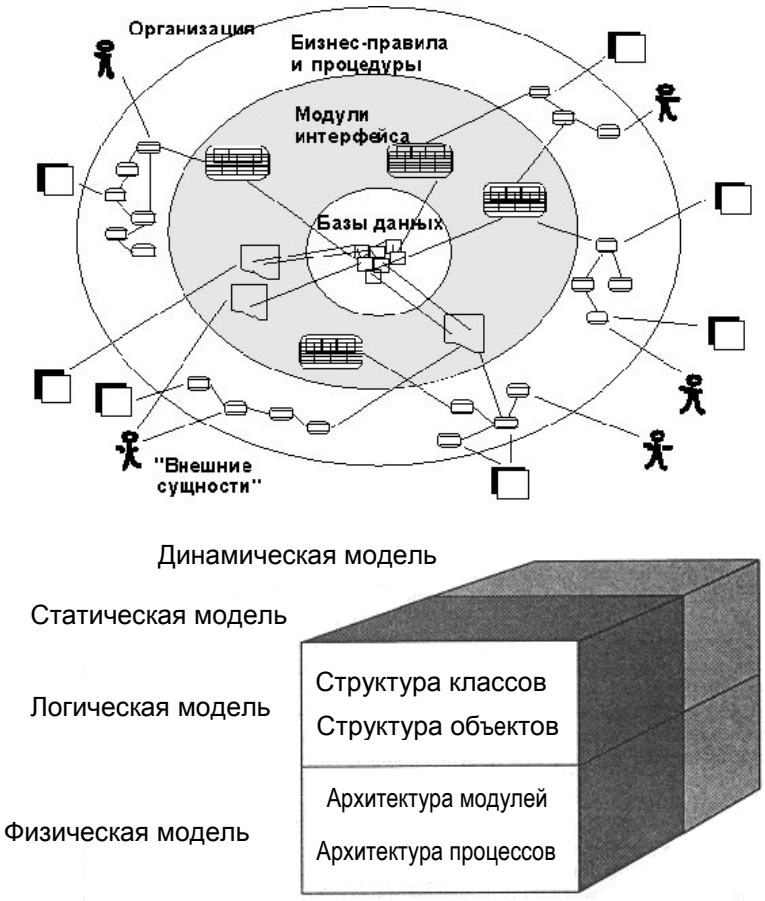


Рис. 7.9. Физическая модель сложной информационной системы (вверху) и комплекс описывающих её объектно-ориентированных моделей во взаимосвязи

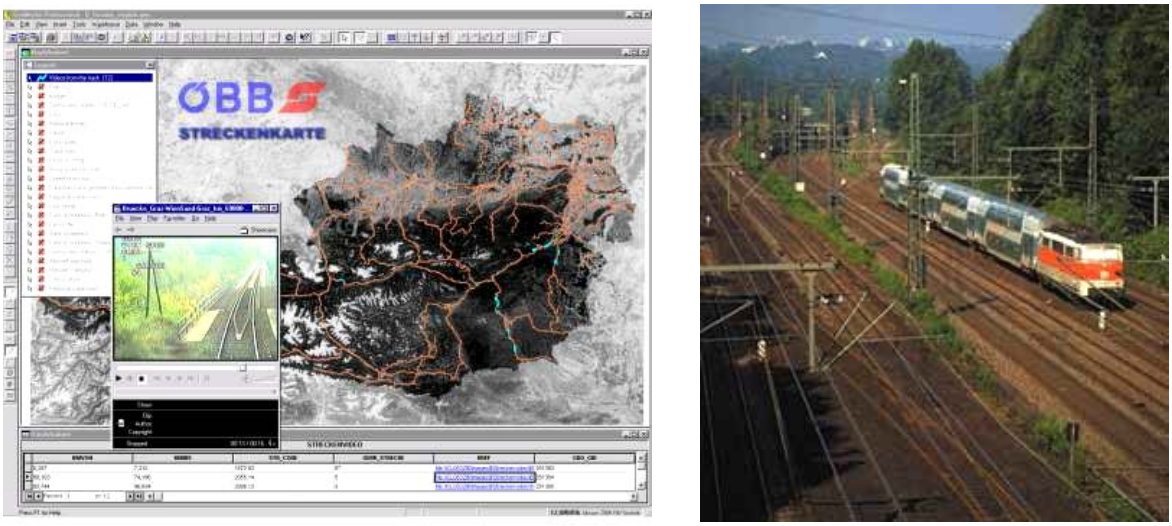


Рис. 7.10. Автоматизация железнодорожных перевозок с помощью геоинформационных систем (ГИС) в Австрии

Важным звеном в развитии ОО технологий анализа, проектирования и разработки программных систем стало образование в 1989 году

некоммерческого консорциума «Группа Управления Объектами» (Object Management Group – OMG[3]) (рис. 7.11), который объединил таких ведущих производителей ПО, как DEC, HP, IBM, Microsoft, Oracle, Rational Software и некоторых других. Это объединение стало мощным катализатором активизации продолжения работ по унификации методов и разработке индустриальных стандартов в области создания интероперабельных[4], неоднородных, распределённых, объектных сред в информационных технологиях.

Принятие OMG, начиная с 1991 года, ряда версий индустриального стандарта CORBA (Common Object Request Broker Architecture), а также активное практическое внедрение самих CORBA-технологий, не могли не вызвать большого интереса разработчиков и пользователей этого стандарта к методам ОО анализа и проектирования, к осознанию необходимости выработки стандарта языка моделирования, который должен был бы стать основой продвижения указанных технологий. В круг задач OMG входила также поддержка создания систем, опирающихся на архитектуру CORBA, обеспечение интероперабельности использующих такие технологии объектно-ориентированных инструментальных средств и интеграция накопленного многочисленными коллективами опыта в этой области.

Рис. 7.11. Логотип организации OMG

Вместе с тем, многообразие возникших в 80-х годах методик и техник ОО моделирования привело к настоящей "войне методов", развёрнутой конкурентами и ставшей причиной того, что идеи объектно-ориентированного подхода при проектировании систем стали терять своих приверженцев. Корпоративный мир ожидал появления концепции и реализующего её языка, способных объединить лучшие методы и представить общую нотацию для ***единого унифицированного представления ОО модели любой проектируемой информационной системы***.

Но так, как к середине 90-х существовало уже более ***50-ти разнообразных объектно-ориентированных языков моделирования информационных систем***, разработчики и заказчики затруднялись при выборе метода проектирования ИС, который, как правило, включал в себя и собственную, не похожую на другие, нотацию. В то же время, в число передовых вырвались улучшенные версии некоторых конкурирующих методологий, среди которых особенно выделялись

[3] Образованная в апреле 1989 года одиннадцатью крупными компаниями, Object Management Group™ (OMG™) уже в 2003 году объединяла более чем 800 организаций-членов. В рамках деятельности корпорации разрабатываются коммерчески перспективные и независимые от производителей спецификации для софтверной индустрии. OMG™ продвигает Архитектуру, Ведомую Моделью (Model Driven Architecture ™) в качестве «Архитектуры Выбора для Связанного (коммуникациями) Мира» ("Architecture of Choice for a Connected World" ™) на базе разрабатываемых ею стандартных всемирно известных спецификаций: CORBA®, CORBA/IIOP™, UML™, XMI™, MOF™, Object Services, Internet Facilities и Domain Interface.

[4] Интероперабельность – взаимная возможность/способность информационных систем или компьютеров обмениваться сообщениями, выполнять программы или пересылать данные между их разными функциональными блоками таким образом, чтобы пользователь при этом практически ничего не должен был бы знать об особенностях этих блоков. Поддерживается средствами развитых многоуровневых интерфейсов.

техника Booch'90 (автор Гради Буч (Grady Booch)) (рис. 7.12), объектно-ориентированная методология OMT (Object Modeling Technique) (автор Джим Рамбо (Jim Rumbaugh)) (рис. 7.13) и ОО подход Fusion.

Рис. 7.12.
Гради Буч

Эти и некоторые другие методы начали смешивать техники друг друга, в результате чего появилось несколько широко известных методологий: *OOSE (Object Oriented System Engineering), OMT-2 и Booch'93*. Но, в целом, каждый из перечисленных методов обладал своей спецификой и оставался обособленным от других, при определенной конструктивности и полезности. Кратко особенности каждого из существующих методов можно выразить следующим образом.

OOSE представлял собой UseCase-ориентированный подход, который обеспечивал прекрасные решения в области бизнес-инжиниринга[5], а также при анализе требований к системам. *OMT-2* хорошо зарекомендовал себя на стадии анализа в системах, где решающую роль играла модель хранения данных. Техника *Booch'93* была полезна на стадии проектирования и играла большую роль при построении систем со сложными алгоритмами функционирования.

Рис. 7.13.
Джим Рамбо

В результате серии переговоров и содействия OMG было решено начать разработку ***Унифицированного Языка Моделирования UML (Unified Modeling Language)*** как унификацию двух методов (Booch'93 и OMT). При поддержке большой софтверной фирмы Rational Software Corporation в октябре 1994 года работа была начата Гради Бучем (Grady Booch) и Джимом Рамбо (Jim Rumbaugh). Первая версия Унифицированного Метода (Unified Method 0.8) появилась в октябре 1995 года, после чего к работе присоединился Айвар Якобсон (Ivar Jacobson) (рис. 7.14), который включил в общий процесс унификации идеи своего метода OOSE. Таким образом, на первом концептуальном этапе UML имел трёх авторов: Буча, Рамбо и Якобсона, каждый из которых, был идеологом свого собственного ОО метода визуального моделирования. В октябре 1996 года была выпущена редакция UML 0.91, в которой нашли отображение многочисленные предложения, которые "три друга (three amigos)" получили в течение 1996 года. Катализатором объединения усилий по унификации UML стал выпуск консорциумом OMG "запроса на предложения"

[5] Бизнес-инжиниринг – деятельность, направленная на проектирование и реализацию бизнес-приложений, т.е. программных средств для решения деловых и экономических задач.

для UML (RFP - Request for Proposal), так как выпуск RFP является первым шагом процедуры принятия OMG того или иного стандарта.

Рис. 7.14.
Айвар Якобсон

После этого Rational Software под своей эгидой объединила специалистов в организацию "Консорциум UML партнёров" ("UML Partners consortium") для подготовки формального определения UML 1.0 в качестве единого мирового стандарта. В работе консорциума приняли участие представители таких известных компаний как: DEC, Hewlett-Packard, Corp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI, Unisys и др.

Каждый из партнеров OMG внёс свой весомый вклад в разработку общего проекта. Hewlett-Packard, например, обеспечивала связь между моделями UML и концепцией ***"повторного использования"***, а IBM разрабатывала язык *OCL* (Object Constraint Language – Язык Объектных Ограничений), описывающий семантику языка UML. Корпорация Microsoft розвивала концепцию ***компонентного программирования*** в рамках модели (COM – Component Object Model) и использования моделей и артефактов UML в ***репозитории***[6] ***моделей, приложений и их взаимосвязей*** (рис. 7.15).

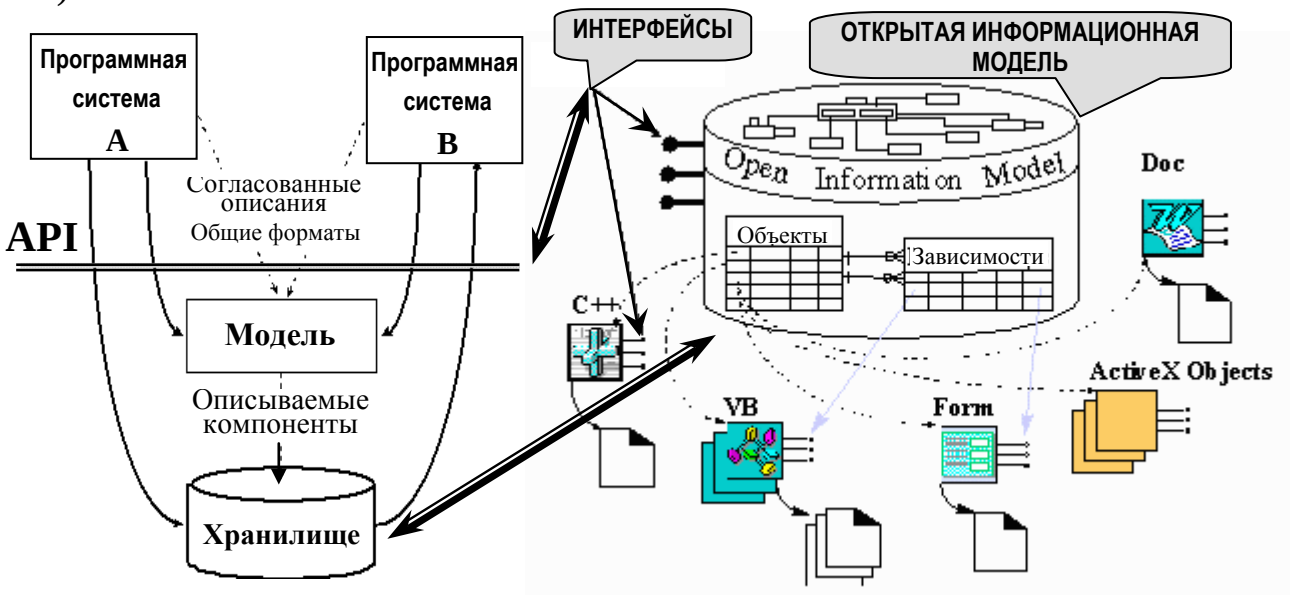


Рис. 7.15. Модель представления репозитория корпорации Microsoft

Следует отметить, что впервые программный продукт Microsoft Repository, основанный на технологиях COM начал распространяться в составе Microsoft Visual Basic 5.0. Только до 2000-го года было продано более 700 000 его копий

[6] Репозиторий – хранилище метаинформации (описательной информации) о разрабатываемых приложениях, компонентах, хранилищах данных, базах данных и т.д. Содержит также, модели процессов, происходящих в системе, модели данных, используемых системой и объектную модель с соответствующим описанием каждой из компонент. Как правило, является специализированным крупным программным продуктом или частью другого ПО.

и более 65-и компаний разработали приложения, которые использовали Microsoft Repository.

Oracle расширила рамки UML для моделирования бизнес-моделей в базах данных. Общий список соисполнителей оказался достаточно обширным, так как каждая компания в работе решала свои конкретные задачи, но при этом согласовывая их с общим направлением работ. С учётом усилий всех официальных участников, в январе 1997года, OMG утвердил вариант UML 1.0, а после детального рассмотрения и учёта всех отзывов представителей общественности – вариант UML 1.1.

Именно этот вариант UML 1.1 в ноябре 1997 года был утверждён как стандарт ИТ отрасли.

Тогда же, OMG принял UML, как главную спецификацию для реализации представления OMA (Object Modeling Architecture – Архитектуры Моделирования Объектов).

4GL, 4th Dimension, ABAP/4, ActiveX, Ada, adb, ALGOL, Apl, ASM, ASN.1, ASP (Active Server Pages), Assembler, ATL (Class Library), awk, AWT Library, bash, Basic, Bourne-Shell, C, C++, CA-EARL, CA-SORT, Centura Team Developer, CFE, CGI, Chill, CL/400, Clarion, Clipper, CLIST, CLOS, CLP, Cobol, CORBA IDL, C-Shell, CSP, DCL, DEC C, Delphi, Delta, Designer/2000, Developer/2000, DeveloperStudio, DHTML, DirectX, Drive (Siemens), Easy for Turbo Databank, Easytrieve Plus, Eiffel, Emacs, ESQL/C, EXEC, FOCUS, Forth, Fortran, Foxpro, GINO-F, GNU-make, GRIT, HASKELL, HP Basic, HP PCL, HP VEE, HPGL, HPSG, HTML, HyperTalk, ILE/400, Imake, IMSL, Informix 4GL, Informix ESQL/C, Java, JavaScript, JCL, JSP (JavaServerPages), KBV, Korn-Shell, Ksh, K-Shell, LabView, Libraries, Lingo, LISP, Lotus Notes Script, M4, Macro Programing, MAGUS, make, Make-Maker, Mantis, Maschinensprachen, Mbed, MFC (Class Library), ML, Modula-2, Motif, MPGS, Mumps, Natural, Nexpert, Oberon, Objective, Object-PAL, Occam, OCL, OjectView, Open-GL, Optima++, OQL, Oracle Forms, Oracle Reports, Oracle SQL Plus, OWL (Class Library), Paisy, Pascal, Perl, PFC, Phigs, PHP, PIC, PL/1, PL/SQL, PL/Z, PLDS, PLM, Power++, PowerBasic, PowerBuilder, PowerHouse Cognos, PROGRESS-4GL, Prolog, Psion OPL, Python, QMF, Qt, RAMIS II (4GL), Rational Rose, rcs, ReXX, RPG, SAL, SAPIENS, SAS, sccs, Scheme, Script Languages, S-Designor, SDL, sed, SESAM, sh, Shell, Simula, Simula67, Siron, Smalltalk, SmartWare, SML, Softedit, S-Plus, SPS, SQL/Windows, STEP-5 (Siemens SPS), STL (Class Library), Superbase unter Windows, Swing, System Builder Plus, TAL, Tcl/Tk, tclsh/wish, tcsh, Tex/LaTeX, Textedit, TK-Library, ToolBook (OpenScript), tools.h++ (Class Library), UIL, UNIFACE, UNIX Shells, UTM, Visual Basic, (VBA) Visual Basic for Applications (Excel, Access, и т.д.), VBScript, vi, Visual Age, Visual C++, Visual Cafe, Visual J++, Visual Objects, Visual SourceSafe, Vocal, VRML, Windows API, WSH (Windows Scripting Host), xedit, XML, yacc/lex, zApp (Class Library), ZINC

Рис. 7.16. Известные современные языки и концепции программирования

"***Предоставляя разработчикам, которые работают с любыми ОО и другими языками программирования*** ((рис.7.16) ***и на любых платформах, общий язык моделирования, для построения распределённых (и бизнес) приложений, в информационных системах, этот подход должен вносить необходимую согласованность в развитие сложной дисциплины реализации программных приложений***" (так было объявлено Ричардом Соли, председателем и исполнительным директором OMG).

Он также заявил, что UML не является чей либо собственностью, он открыт для всех. Хотя UML и является строго определенным во всех отношениях языком, он не ставит барьеров на пути развития средств моделирования. Язык UML может быть расширен без переопределения своего ядра. ***Предусматривается использование UML в качестве основы для создания программных средств визуального моделирования информационных систем с дальнейшей имитацией их работы в этой среде.***

В рамках языка и концепции использования UML, создатели этого ***унифицированного средства моделирования*** и их партнеры объявили следующий комплекс целей, которые были реализованы в данном проекте.

❶ Пользователям предоставляется готовый к употреблению, ***мощный язык моделирования***.

❷ Подготовлены ***необходимые средства моделирования***, а также ***языковые механизмы расширения и специализации*** для разных ***предметных областей***, так как при дальнейшем развитии UML нежелательно переопределять его ядро каждый раз вновь и вновь. Поэтому концепции UML допускают его расширение для нестандартных ситуаций, а также содержат средства учёта специализации задач в конкретных предметных областях (к примеру, в системах реального времни, Web–технологях, Web–сервисах, геоинформационных системах и т.д.).

❸ Обеспечена ***независимость процесса разработки информационных систем от языков программирования и моделей этого процесса***, так как UML поддерживает все разумные языки программирования и модели процессов разработки ПО, которые приняты при разработке программного обеспечения на конкретных предприятиях.

❹ Создание ***формального обоснования языка моделирования***. Авторы UML выдвинули тезис о том, что формализация языка должна присутствовать в разумных границах, для обеспечения лёгкости его восприятия и простоты использования коллективами разработчиков. Для этого, ***правила и ограничения на конструирование валидных (то есть адекватных исходным системам) моделей информационных процессов***, определены в UML ***английской прозой***, а также в ***Языке Объектных Оограничений*** (OCL). Семантика (то есть содержание) моделей разрабатываемых систем, также описана ***английской прозой***. Следует чётко представлять, что ***английская проза*** при изложении спецификации и основ применения UML была использована только для того, чтобы в соответствии с задумкой авторов не отталкивать пользователей от чрезвычайной сложности созданной системы моделирования. Ибо, одновременно с этим, для всех описываемых в UML подходов и моделей разработаны принципиальные математические доказательства и соответствующие математические представления.

❺ С разработкой UML внесён важный вклад в ***расширение рынка производимых объектно-ориентированных инструменто***. Кроме того, UML должен обеспечить интероперабельность для всех существующих и новых объектно-ориентированных средств, которые появляются вновь и вновь. Основным практическим выходом разработанного языка UML является то, что он предназначен для спецификации, визуализации, конструирования и документирования артефактов, которые отчуждаются от разработчиков, а также и других материалов, связанных с элементами разработки программных систем.

В концептуальном плане новый язык характеризуется следующими ***отличительными признаками***. С точки зрения содержания UML наследует принципы и механизмы техники Booch и методов OMT и OOSE. Во-вторых, он превосходит их. В-третьих, следует отметить, что UML - это стандарт языка, а

не стандарт процессов анализа, проектирования и разработки. UML является квинтэссенцией концепций моделирования, в соответствии с которыми был достигнут консенсус в среде ведущих компаний-производителей ОО програмного обеспечения, а конкретно в вопросах:

❶ Соглашений о семантике и нотации для разнообразных аспектов современного моделирования в лаконичной форме.

❷ Определения достаточной глубины семантики для будущих технологий, таких как: компонентное программирование, распределённые вычисления, геоинформационные и беспроводные технологии и т.д.

Необходимо также понимать, что несмотря на широкие возможности UML, это только язык моделирования, то есть средство моделирования, которым необходимо ***уметь*** пользоваться, ***знать***, когда и где использовать ту или другую технику, чётко представлять себе порядок этапов проектирования и структуру модели, которая разрабатывается.

В инфраструктуре UML важное место занимают средства проектирования и реализации. К ним относятся:

❶ ***языки программирования.*** UML - язык визуального моделирования и не предназначен для визуального программирования, хотя инструментальное средство, поддерживающее нотации UML, может реализовывать кодогенерацию в конкретный язык программирования (C++, Delphi, Java и др.) на основе построенных ***моделей.***

❷ ***инструментальные средства.*** UML не является спецификацией для разработки инструментальных сред. В UML определена только модель семантики, но не определены модели интерфейса, хранения и поддержки времени выполнения. В настоящее время имеется несколько CASE– средств, производители которых поддерживают моделирование средствами UML. Среди них можно выделить: Rational Rose, Select Enterprise, Platinum и Visual Modeler (рис. 7.17).

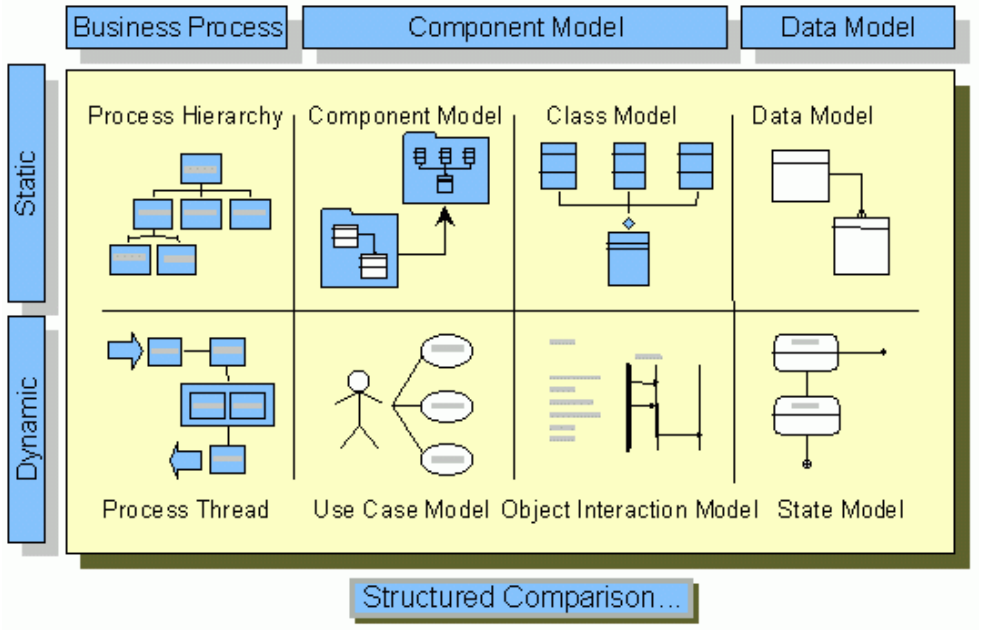


Рисунок 7.17. Применение моделей UML при разработке бизнес приложений с помощью CASE-средств фирмы Select Enterprise

❸ ***модели процесса.*** Предприятия используют UML как единый язык при разработке всех компонентов проекта, хотя кокретные этапы и сам процесс разработки могут быть реализован по-разному. При этом наблюдается сходимость моделей процессов для разных организаций, но общего соглашения по данному вопросу пока не найдено. UML не навязывает свою модель процесса, но при разработке языка авторы имели в виду ***процесс***, обладающий свойствами ***управления прецедентами, итеративный и инкрементный.***

7.3 Структура и состав языка UML

Рассмотрим структуру языка с точки зрения компонент, составляющих его описание. Определение UML, является самодостаточным и включает в себя (в версии 1.3 , выпущенной в 1999 г.) следующие документы.

❶ Семантика UML;
❷ Нотация UML;
❸ Стандартные шаблоны;
❹ Определение средств CORBA-интерфейсов;
❺ Спецификации UML XMI DTD (для обмена моделями с помощью языка XML);
❻ Спецификации языка OCL (Object Constraint Language).

Описание семантики UML производится на уровне метамодели и представляется тремя согласованными способами.

❶ Абстрактный синтаксис включает метамодель UML, разбитую на 9 взаимосвязанных пакетов, каждый из которых описывается диаграммами классов UML (рис. 7.18). Метамодель включает в себя около 50 классов, более 100 ассоциаций и 1000 ограничений.

❷ На каждом етапе моделирования можно использовать правила состоятельности, которые описаны английской прозой с использованием языка ограничений OCL, являющегося объектно-ориентированым языком с ограниченными кванторами.

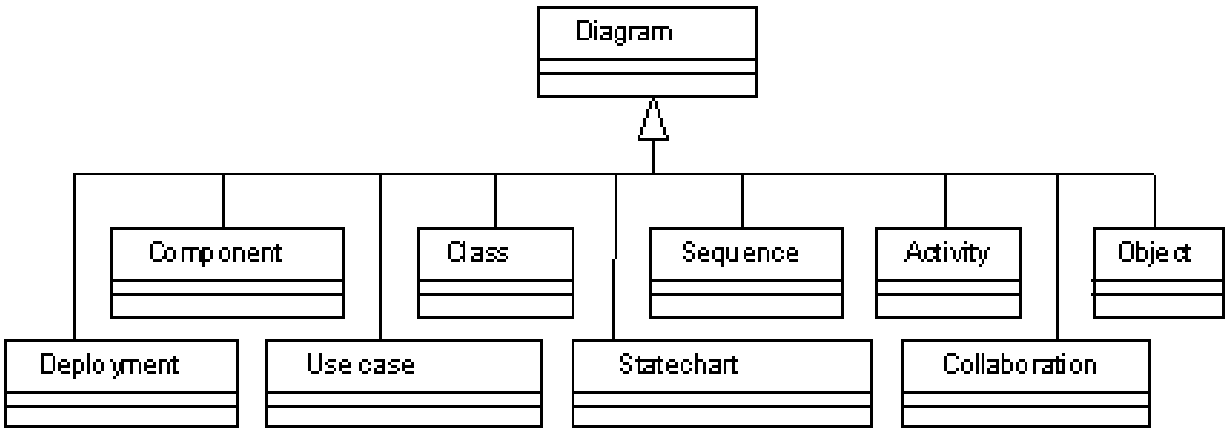


Рис. 7.18. Классы 9-ти типов диаграмм языка UML,
представленные в терминах самого языка

❸ Присутствует описание элементов нотации моделирования (семантики), которое также выполнено в английской прозе.

Рассмотренные три описания в совокупности и образуют формальное определение UML. Более формальное определение потребовало бы применения сложного математического аппарата, что в конечном итоге затруднило бы повсеместное использование языка, хотя и добавило бы строгости определениям.

Вводимые пользователем расширения UML возможны благодаря наличию следующих встроенных в язык механизмов:

❶ Стереотипы позволяют определить подклассы существующих элементов моделирования, сохраняя форму, но вкладывая новое содержание и добавляя новые свойства.

❷ Тэгированные значения позволяют определять новые свойства элементов моделирования, что дает возможность привязать к модели совершенно произвольную информацию.

❸ Ограничения позволяют определить подмножество элементов моделирования, наделенное определенными свойствами (например, упорядоченностью). Указанные условия могут быть описаны с помощью языка OCL.

7.4. Типы диаграмм UML и их использование

Моделирование сложной системы средствами UML сводится к ее описанию в нескольких различных проекциях (рис. 7.19). Каждая проекция описывает определенный аспект разрабатываемой системы, а все вместе они определяют систему с должной степенью полноты, правильности и адекватности. В UML для описания системы при анализе и проектировании предусмотрены следующие графические диаграммы.

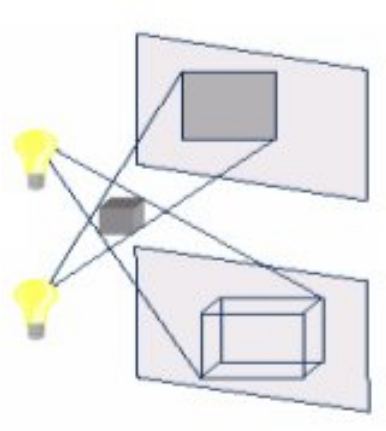


Рис. 7.19. Представление разных проекций (аспектов) системы

Диаграммы, которые описывают статическое состояние системы:

❶ Диаграммы прецедентов (вариантов использования) (Use case diagrams).

❷ Диаграммы классов (Class diagrams).

❸ Диаграммы объектов (Object diagrams).

Диаграммы, описывающие поведение системы:

❹ Диаграммы состояний (State diagrams).

❺ Диаграммы видов деятельностей (Activity diagrams).

❻ Диаграммы последовательностей (Sequence diagrams).

❼ Диаграммы сотрудничества (кооперации, взаимодействия) (Collaboration diagrams).

Диаграммы, описывающие физическую реализацию системы:

❽ Диаграммы компонентов (Component diagrams).

❾ Диаграммы развёртывания (Deployment diagrams) (рис. 7.20).

Таким образом, язык UML позволяет:

❶ формализовать ***функциональные требования к системе*** с помощью аппарата ***use case диаграмм***;

❷ детализировать ***требования к системе*** путём построения диаграмм технологичных сценариев;

❸ разбить ***сложную систему*** на составные части с минимумом взаимных сявзей путём выделения ***пакетов***;

❹ выделить ***классы данных***, построив концептуальную модель данных в виде ***диаграммы классов***;

❺ выделить ***классы***, описывающие ***интерфейс пользователя*** и создать схему навигации экранов для взаимодействия ИС с ним;

❻ описать ***процессы взаимодействия объектов*** при выполнении системных функций;

❼ описать ***поведение объектов*** в виде диаграмм состояний;

❽ показать ***программные компоненты*** и их взаимодействие через интерфейсы;

❾ представить ***физическую архитектуру системы***.

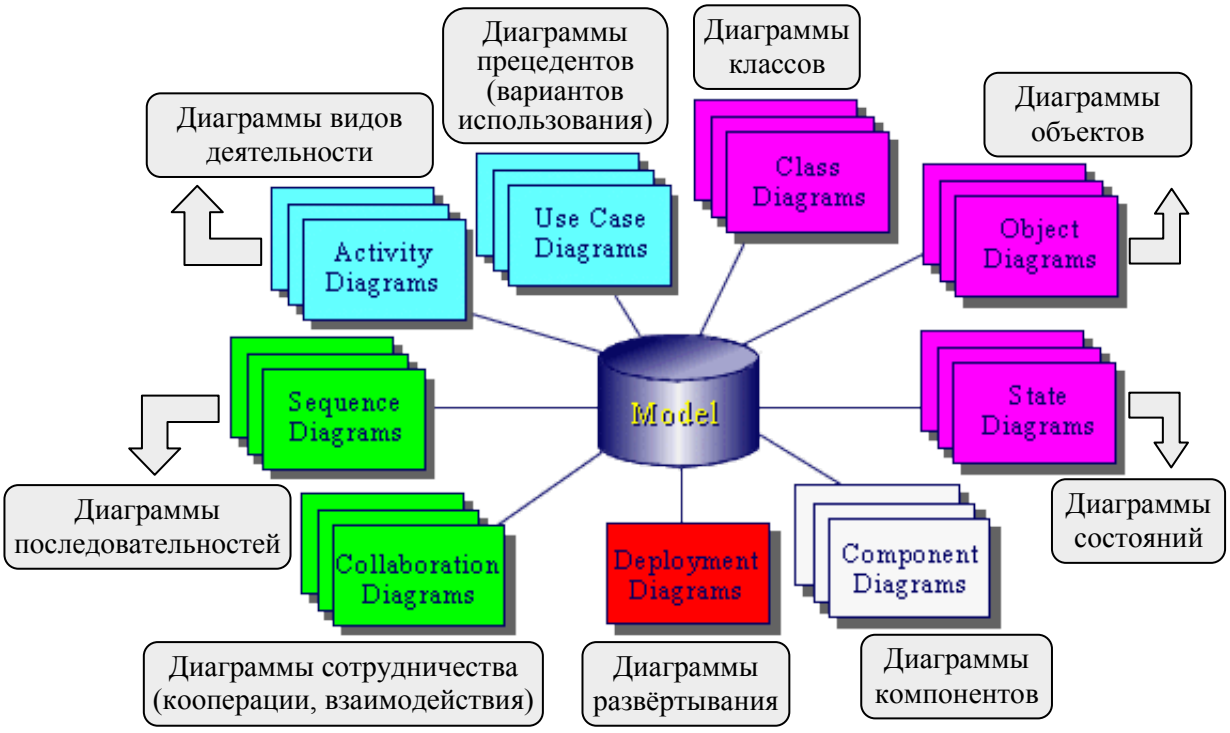


Рис. 7.20. Комплекс диаграмм, описывающих полную модель системы

В комплексе, диаграммы позволяют осветить ***пять важнейших сторон системы, с точки зрения разработчика*** (рис. 7.21).

Рис. 7.21. Пять разных взглядов на систему, получаемые с помощью девяти диаграмм UML.

И так как UML обширный и сложный в плане терминологии и нотации язык, следует кратко описать диаграммы каждого из вышеперечисленных типов.

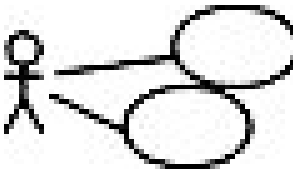


Рис. 7.22. Диаграммы прецедентов (Use case)

Техника Прецедентов (Вариантов Использования – UseCase) - была впервые предложена Айваром Якобсоном в 1992 и быстро завоевала всеобщее признание за счет простоты и легкости восприятия и применения. Суть ее состоит в следующем. Проектируемая система представляется в виде наборов ***актантов (акторов)***, взаимодействующих с системой с помощью так называемых ***прецедентов*** (рис. 7.22). Актантом является любая сущность, взаимодействующая с системой извне.

Им может быть человек, оборудование, другая система, то есть таким образом разработчик определяет, ***что именно*** взаимодействует с системой. В свою очередь прецедент описывает, ***что*** система предоставляет актанту - то есть определяет некоторый набор транзакций, совершаемый актантом при диалоге с системой, при этом ничего не говориться о том, каким образом будет реализовано взаимодействие. Детальное описание всех этих процессов - удел других техник моделирования UML.

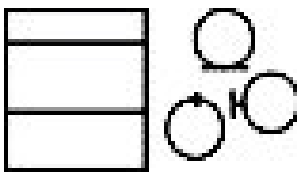


Рис. 7.23. Диаграммы классов (classes)

Диаграмма же прецедентов несет в себе высокий уровень абстракции, что позволяет еще на ранних этапах проекта определить и зафиксировать функциональные требования к системе, её будущие интерфейсы и таким образом обеспечить гибкий и эффективный механизм взаимодействия между разработчиком и заказчиком проекта на этапе анализа.

Диаграмма классов показывает статическую структуру части системы (рис. 7.23). Таким образом, составляющими (элементами) данного типа диаграмм являются классы, объекты и отношения между ними. Нотация классов и объектов проста и интуитивно понятна всем, кто когда-либо имел опыт работы с разного рода CASE-инструментариями (рис. 7.22, 7.23). Класс представлен прямоугольником

с тремя разделами, в которых соответственно помещаются имя класса, атрибуты и операции. Схожая нотация применяется и для объектов - экземпляров класса, с тем различием, что к имени класса добавляется имя объекта и вся надпись подчеркивается.

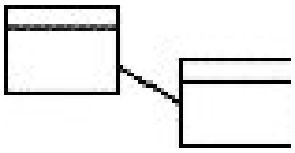


Рис. 7.23. Диаграммы объектов (objects)

Нотация UML предоставляет широкие возможности для отображения дополнительной (и зачастую очень важной) информации, к которой можно отнести абстрактные операции и классы, стереотипы, общие и частные методы, интерфейсы, параметризованные классы и т.д. Важнейшими элементами ***диаграмм классов и объектов*** являются ***ассоциации***, то есть ***статические связии между классами***, изображаемые в виде связанных с элементами диаграмм линий, на которых может указываться мощность ассоциации, её направление, название, и возможное ограничение, реализующее механизм расширения UML (рис. 7.24).

Рис. 7.24. Ассоциации диаграмм языка UML.

Существует возможность отразить специфические свойства ассоциации - например, отношение агрегации, когда составными частями класса могут выступать другие классы.

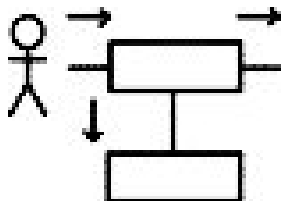


Рис. 7.25. Диаграммы кооперации (collaboration)

Такое отношение изображается в виде ромба, расположенного рядом с агрегирующим классом. Отношение ***обобщения*** также имеет собственную графическую нотацию в виде треугольника и связующей линии, позволяя представить иерархию наследования – от суперкласса к подклассам (рис.7.24).

Диаграммы кооперации (взаимодействия) описывают взаимодействие объектов системы, выполняемого ими для получения некоторого результата (рис.7.25). Под получением результата подразумевается выполнение законченного действия, например, описание в терминах взаимодействующих

объектов смоделированного ранее прецедента ***Использования системы*** или некоторого сервиса системы, объявленного как операция класса на соответствующей диаграмме. ***Диаграммы взаимодействия*** представляются в двух формах - ***Диаграмма следования*** и ***Диаграмма кооперации***. И та, и другая описывают потоки сообщений (вызов методов или сигналы) между объектами, участвующими во взаимодействии. Служа, в целом, одной цели, диаграммы, по сути своей, имеют существенные различия.

Диаграмма следования (рис.7.26), делает упор на временну̂ю последовательность передаваемых сообщений. Когда важен порядок, вид и имя сообщения, на диаграмме изображаются исключительно те объекты, которые непосредственно участвуют во взаимодействии, и не показываются возможные статические ассоциации с другими объектами. Таким образом, для ***Диаграмм следования*** ключевым моментом в моделируемой системе является динамика взаимодействия её элементов.

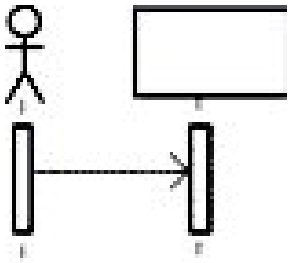


Рис. 7.26.

Диаграммы следования (sequence)

Следует иметь в виду, что ***Диаграмма следования*** (последовательности) представляет (отображает) два измерения. Первое - слева направо, в виде вертикальных линий, изображающих объекты, участвующие во взаимодействии. Верхняя часть линий дополняется прямоугольником, содержащим имя класса объекта или имя экземпляра объекта. Второе измерение - вертикальная временная ось. Сообщения, посылаемые одним объектом другому, изображаются в виде стрелок с именем сообщения и упорядочены по времени возникновения.

Для ***Диаграммы кооперации*** главным является возможность отобразить не столько последовательность взаимодействия, сколько всё окружение объектов, участвующих в нем. То есть, показываются не только посылаемые и принимаемые сообщения, но и косвенные связи между ассоциированными объектами. Говорят, что Диаграммы кооперации описывают полный контекст взаимодействия и представляет собой своеобразный временной "срез" конфигурации сети объектов, взаимодействующих для выполнения определенной бизнес-цели программной системы.

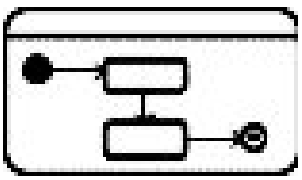


Рис. 7.27. Диаграммы состояний (state)

Диаграмма кооперации изображает объекты, принимающие участие во взаимодействии, в виде прямоугольников, которые содержат имя объекта, его класс и, возможно, значения атрибутов. Ассоциации между объектами, как и на диаграммах классов, изображаются в виде соединительных линий. Возможно указание имени ассоциации и ролей, которые играют объекты в данной ассоциации. Динамические связи - потоки сообщений, представляются также в виде соединительных линий между объектами, сверху которых располагается стрелка с указанием направления и имени сообщения.

Диаграммы состояний описывают поведение объекта во времени, то есть моделирует все возможные изменения в состоянии объекта, вызванные внешними воздействиями со стороны других объектов или извне (рис. 7.27). Диаграммы состояний применяются для описания поведения объектов и для описания операций классов.

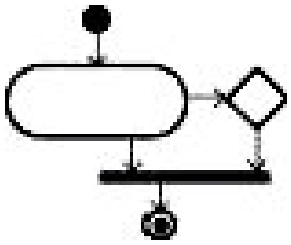


Рис. 7.28. Диаграммы активности (activity)

В отличие от ***диаграмм взаимодействия***, данный тип диаграмм описывает смену состояния только одного класса или объекта. Каждое состояние объекта представляется на ***Диаграмме состояний*** в виде прямоугольника с закруглёнными углами, содержит имя состояния и, возможно, значения атрибутов объекта в данный момент времени. Переход из одного состояния в другое осуществляется при наступлении некоторого события – получения объектом сообщения или приёма сигнала, и изображается в виде стрелки, которая объединяет два соседних состояния. Имя события указывает на переход. Кроме того, на переходе могут указываться действия, вырабатываемые объектом в ответ на внешние события (при переходе из одного состояния в другое или при нахождении в некотором определенном состоянии). Следует отметить, що ***Диаграмма состояний*** описывает, как правило, реакцию объекта на асинхронные внешние события, а для описания реакции на внутренние события предназначены ***Диаграммы активности***. Срабатывание перехода может зависеть не только от наступления некоторого события, но и от выполнения определённого условия, называемого ***пусковым условием***. Объект переходит из одного состояния в другое, только если произошло указанное событие и пусковое условие приняло значение "истина".

Диаграммы активности – частный случай ***Диаграмм состояний*** (7.28). Каждое состояние – это сущность выполнения некоторой операции. А переход в следующее состояние срабатывает только при завершении операции в начальном состоянии. Таким образом, реализуется принцип процедурного, синхронного управления, обусловленный завершением внутренних действий. Описываемое состояние не имеет внутренних переходов и переходов по внешним событиям.

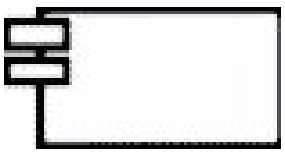


Рис. 7.29. Диаграмма компонентов (component)

Их графическая нотация практически не отличается от нотации ***Диаграмм состояний***, с той лишь разницей, что на переходах отсутствует сигнатура действия и добавляется символ "синхронизации" переходов для реализации параллельных алгоритмов.

Основным направлением использования ***Диаграммы активности*** является описание операций классов, когда необходимо представить алгоритм её реализации, при этом каждый шаг является исполнением операции некоторого класса.

Диаграммы компонентов и ***развёртывания***, в отличие от ранее рассматриваемых диаграмм прецедентов, классов, состояний и следования,

которые являются логическим представлением системы в процессе её разработки, описывают физическое представление системы (рис. 7.29, 7.30).

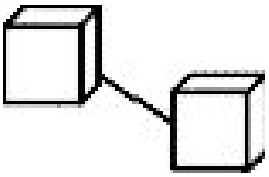


Рис. 7.30. Диаграмма развёртывания (deployment)

Диаграмма компонентов позволяет определить архитектуру системы, которая разрабатывается, установив зависимость между программными компонентами, в роли которых может выступать начальный, бинарный и код, который выполняется. Во многих средах разработки представление о модуле (или компоненте) соответствует (или совпадает) с представлением о файле. Пунктирные стрелки, соединяющие модули, показывают отношення взаимозависимости (аналогичные тем, которые имеют место при компиляции).

Диаграммы развёртывания используется для представления конфигурации компонентов, процессов и объектов, присутствующих в системе на етапе исполнения. Кроме того, ***Диаграммы развёртывания*** показывают (отображают) физическую зависимость аппаратных устройств, спроектированной информационной системы, задействованных в её реализации, а также соединений между ними – маршрутов передачи информации.

Понятно, что язык UML является ещё одним, очередным этапом в совершенствовании инструментов проектирования и разработки продолжающих усложняться информационно-компьютерных систем. При этом повышаются и множатся уровни абстрактных представлений их составных частей и элементов взаимодействия.

Вопросы

1.	Почему все больше усложняется процесс разработки программ и программных систем?
2.	Какие и почему существуют 5 уровней сложности программирования?
3.	Какие существуют 5 уровней в развитии языков программирования?
4.	Чем вызвано появление объектно-ориентированного подхода в программировании?
5.	Какой комплекс целей был воплощён и реализован в языке моделирования процесса разработки программ UML?
6.	Какие существуют средства проектирования и реализаци в инфраструктуре UML?
7.	Что образует структуру языка UML и какие компоненты входят в его состав?
8.	Для чего предназначены диаграммы UML и как они называются?
9.	Какие основные процессы разработки программных систем упрощает язык UML?

Язык – естественная или искусственная знаковая система, которая предназначена для передачи информации. К естественным знаковым системам относятся языки общности: украинский, русский, английский и др., а к искусственным, как правило, языки программирования: C++, Java и т.д.

Энциклопедия.

8. ВВЕДЕНИЕ В ТУРБО ПАСКАЛЬ

8.1. Истоки Турбо Паскаля

Информационные технологии (ИТ), как совокупность средств и методов обработки данных и информации с помощью персональных компьютеров, приобрели статус главных движущих сил современного индустриального общества. Понимание основ ИТ, овладение их главными концепциями и навыками эффективного применения рассматривается сейчас в мире как одна из фундаментальных компонент образования на уровне владения чтением и письменностью. В целом, информатика является областью науки, которая изучает структуры и общие качества информации, а также вопросы, связанные со сбором, хранением, поиском, переработкой, преобразованием, распространением и использованием ее в различных сферах деятельности человека. Поэтому очень важно основной упор в изучении языков программирования делать на модель "с ориентацией на программирование" (programming-first introduction), в которой одним из популярнейших средств обучения продолжает оставаться язык Турбо Паскаль (ТП) (рис.8.1), который является подмножеством объектно-ориентированного языка Object Pascal – основы широко распространенной среды быстрой разработки приложений Delphi.

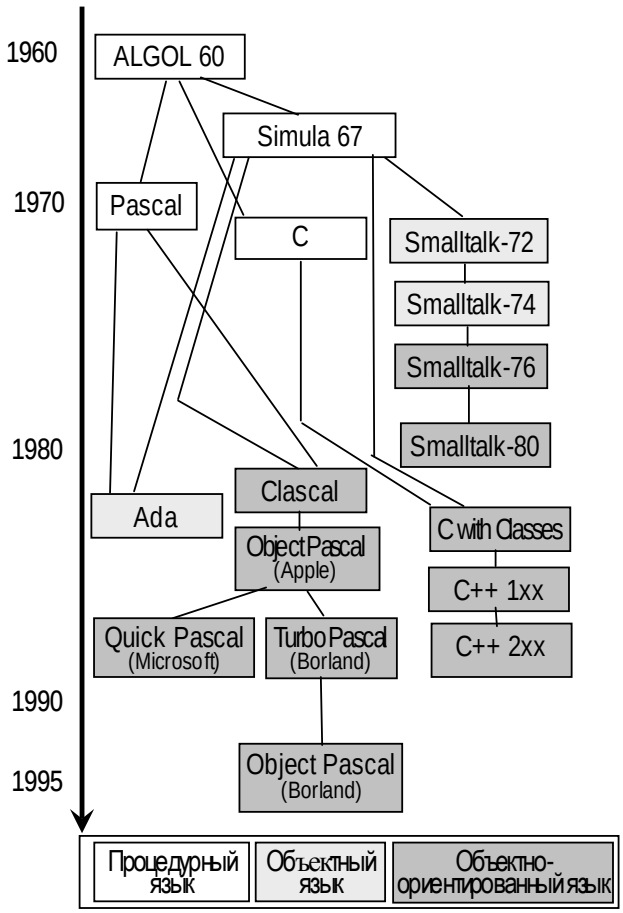


Рис. 8.1. Генеалогия языка Турбо Паскаль (оттенками серого показаны модификации)

Важно также и то, что язык Паскаль и продолжающий его традиции язык ТП, разрабатывались со следующими важными целями:

❶ систематически и точно выражать концепции и структуры главных элементов области программирования;

❷ осуществлять разработку программ систематически;

❸ показать, что язык программирования с богатым набором гибких структур данных (в противоположность языку С) и управляющих конструкций может быть реализован эффективно;

❹ включить в среду программирования развитые средства редактирования текста, диагностики ошибок и отладки программ для облегчения процесса обучения программированию и исследованию программ.

В соответствии с этим подходом, в учебнике сделана попытка, средствами интегрированной среды разработки Турбо Паскаль представить следующие основные концепции программирования:

❶ модели и структуры данных; ❷ управляющие структуры и их исследование; ❸ порядок выполнения управляющих структур;	❹ алгоритмы и вычисления; ❺ концепцию разработки интерфейса программ и программных систем; ❻ средства исследования разработанных программ.

Рис. 8.2.
Блез Паскаль

Для читателя этого издания мы должны отметить некоторые ***очень важные обстоятельства.***

❶ Так, как интегрированная среда разработки (ИСР) Турбо Паскаль разрабатывалась для зарубежных пользователей (американцев, англичан и других англоязычных специалистов), то она в своей работе в операторах программ и элементах интерфейса использует ***только*** текст на английском языке.

❷ На экран можно выводить ***только*** те иноязычные тексты, которые являются строчными константами (то есть наборами символов, которые ИСР ТП не анализируются и выводятся в соответствии с их кодом). Например, русский текст, который ограничен с обеих сторон кавычками, будет однозначно выведен на экран компьютера оператором процедуры `WriteLn('Русский язык')` в таком же виде (при наличии соответствующего русскоязычного драйвера[1]). При этом, Вам следует понимать, что процедура вывода данных на

[1] Драйвер – програма, которая управляет работой внешнего устройства (мышь, клавиатура, принтер и т.д.); драйвер, как правило, является интерфейсом между внешним устройством и программами ввода-вывода

экран компьютера только отображает на экране символы в соответствии с их кодами.

Вообще, даже и сегодня, не всем известно, что интегрированная среда разработки (IDE[2]) Турбо Паскаль (Turbo Pascal) производства фирмы Borland International (США) интегрировала в себе влияние и вклад ряда талантливых европейских специалистов.

Блез Паскаль (Blaise Pascal) (1623 – 1662), француз по происхождению, чьё имя носит язык программирования, был выдающимся математиком, физиком и философом (рис. 8.2). В 1642 году он сконструировал первую в мире машину для суммирования чисел, получившей название "Паскалина", которую можно назвать первым в мире цифровым калькулятором, подобным механическим калькуляторам, используемых значительно позднее в 1940-х годах.

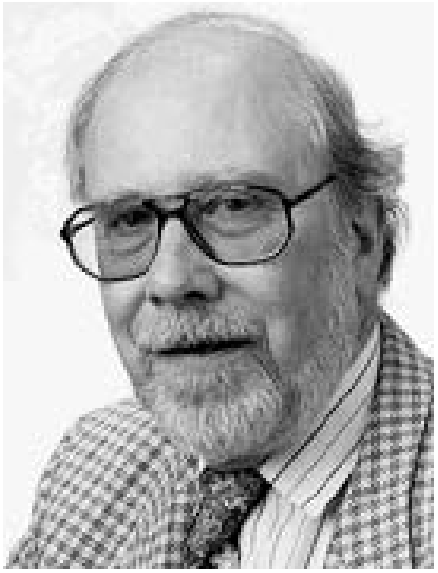


Рис. 8.3.
Никлаус Вирт

Швейцарец ***Никлаус Вирт*** (Niklaus Wirth), известный всему современному миру научный работник, программист и преподаватель Института компьютерных систем в г. Цюрих (Швейцария) (рис. 8.3), разработал новый алгоритмический язык Паскаль в 1968 г. для целей обучения и реализации концепции поддержки выполнения вычислений с четкой типизацией и назвал ее именем Блеза Паскаля. Язык разрабатывался Виртом специально с педагогическими целями и с учетом простоты получения соответствующего машинного кода. Благодаря этому, создание компилятора для Паскаля не превышает по трудоемкости обычную дипломную работу выпускника высшего учебного заведения. Кроме этого, сам язык вышел очень простым – описание стандартного Паскаля занимает около 30-ти страниц печатного текста.

С появлением персональных компьютеров многие программисты начали использовать их для решения своих прикладных задач. Но компиляторы и трансляторы[3] языков программирования того времени ALGOL, FORTRAN и др. работали довольно медленно и не обеспечивали удобства в работе. Желание во что бы то ни стало изменить существующее положение, объединило двух энергичных программистов: француза по имени ***Филип Кан*** (Philippe Kahn), парижанина по происхождению, из семьи немецкого инженера и французского кинорежиссёра, и датчанина ***Андерса Хейльсберга*** (Anders Hejlsberg) (рис. 8.4).

операционной системы. Наиболее характерный пример драйвера – программа KEYRUS.COM, которая кириллизирует клавиатуру и монитор для обеспечения русскоязычного интерфейса пользователя с персональным компьютером.

[2] IDE (Integrated Development Environment – интегрированная среда разработки (ИСР)) – программное средство, объединяющее редактор текста, компилятор, загрузчик и другие компоненты работы с программой.

[3] Компилятор (транслятор) – программа, которая переводит исходный текст программы, написанный программистом на языке высокого уровня в эквивалентную программу на машинном языке компьютера.

Ф. Кан, вдобавок, в свое время учился у "самого" Никлауса Вирта. Вот они и создали в 1982 г. не только самый быстрый в мире компилятор алгоритмического языка Паскаль, а еще и удобную программную среду для работы с ним, которые и назвали Turbo Pascal 1.0 (Турбо, то есть "быстрый" Паскаль) с впечатляющими характеристиками: размер дистрибутива на дискете 131 297 байт, требуемый объём оперативной памяти – 64 Кб. Распространение его было начато в конце 1983 года в США созданной ими же фирмой Borland International по цене US$49.95. Об этом периоде лучше всех сказал А. Хейльсберг: «Не было героев, не было гениев, а только компания полных энтузиазма парней, делающих то, что казалось неплохой идеей».

Следует добавить, что кроме IDE Турбо Паскаль, сотрудники фирмы Borland разработали и одноименный алгоритмический язык, который стал расширением языка Паскаль Никлауса Вирта. Это связано с тем, что, постоянно общаясь с пользователями и следуя за развитием индустрии программирования, они прибавляли к стандарту языка все новые и новые полезные возможности.

Рис. 8.4. Филип Кан
и Андерс Хейльсберг (1982 г.)

Поэтому Турбо Паскаль является одновременно ***интегрированной средой разработки программ*** и ***алгоритмическим языком программирования***. В RAD Delphi дальнейшие увеличение количества управляющих конструкций, типов и структур данных привели к появлению объектно-ориентированного языка Object Pascal/Delphi, который по своим возможностям подобен таким известным языкам, как C++, SmallTalk, Java, Python и др.

Низкая цена и высокое качество сделало Turbo Pascal 1.0 доступным миллионам пользователей и профессиональных программистов. Только за два года было продано 300 000 экземпляров Турбо Паскаля, что превысило количество всех проданных компиляторов других языков программирования для микрокомпьютеров вместе взятых! И сейчас пользуются уважением программистов всего мира программные продукты фирмы Borland, такие, как Delphi последних версий 5, 6 и 7, Paradox, Quattro Pro, Turbo Pascal, Borland C++, Sidekick, ObjectVision и некоторые другие. Более 50-ти миллионов пользователей оставили свои электронные почтовые адреса на Web-сайте этой фирмы для сессионной рассылки новостей и программных апгрейдов. За 2002 год фирмой было продано программных продуктов на сумму $US 226 млн., а 900 ее штатных сотрудников работают в офисах стран США, Австралии, Франции, Англии, Германии, Италии, Швеции, Сингапура и Дании. Удача сопутствовала А. Хейльсбергу и в дальнейшем: в 1996 г. он был приглашён в фирму Microsoft, где был удостоен звания «главный инженер» (distinguished engineer – 1999 г.), которое в этой организации имеют только 16 человек, и в

настоящее время является ведущим разработчиком языка C# (Си шарп). Это не осталось незамеченным в компьютерном сообществе и за большой вклад в развитие программирования, в 2001 году его наградили престижной премией известного в мире издания "Dr. Dobb's Journal".

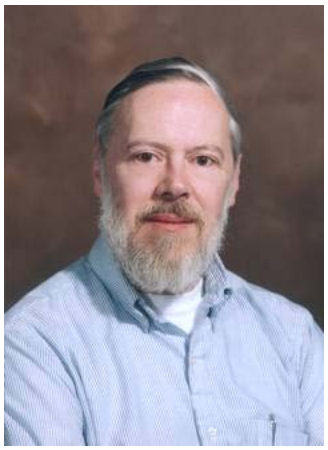


Рис. 8.5.
Деннис Ричи

Интересно, что ИСР языка Турбо Паскаль была написана на языке C (Си). А этот язык, разработал в 1972 году Деннис Ричи (Dennis Ritchie), 31-летний американский специалист по системному программированию (рис. 8.5), который получил степень бакалавра по прикладной математике в Гарвардском университете и начал работать в "Bell Telephone Laboratories" в 1968 году. Ричи надеялся, что его новый язык пригодится для программирования новой операционной системы UNIX, что в дальнейшем и случилось. Кстати, автор UNIX Кен Томсон (Ken Thompson), который был автором прародителя C – языка B, считался среди коллег лучшим программистом в мире!

8.2. Технология работы в среде Турбо Паскаль версии 7.0

Работая с компьютером, Вы всегда выполняете решение своих насущных задач. Если стоящая перед Вами задача не может быть решена уже имеющимися в Вашем распоряжении прикладными программами (Word, Excel, Winamp и т.д.), значит, Вам необходимо творчески создать «нечто», способное заставить компьютер выполнять Ваши желания. Тут и наступает момент, когда Вы вступаете в пространство, именуемое «Программирование». Здесь Вы уже должны уметь выразить свои желания средствами, понятными компьютеру. Для этого служат языки программирования. Владение языками программирования, так же как и разговорными языками, позволяет Вам получать недоступные до сих пор результаты. Конечно, Вам не стоит просить компьютер сделать «мокрую уборку» – он не для этого предназначен. Но массу элементарных действий по обработке Ваших данных с огромной скоростью он будет рад сделать, но только в полном соответствии с алгоритмом, который Вы сами ему и зададите. Естественно – на подходящем языке программирования.

Как и любой другой язык, каждый язык программирования имеет свои синтаксис и грамматику, а также круг решаемых задач. Но если на разговорном языке Вы решаете задачи воздействия на людей, то на языке программирования Вы воздействуете на центральный процессор компьютера с целью получения полезного для себя результата. И здесь отличие состоит в том, что Вы постоянно должны распределять свои усилия между охватом обрабатываемых данных и формированием алгоритма их обработки, а также общаться с компилятором языка, который служит переводчиком между Вами и компьютером при общении.

С этой точки зрения, методически, Ваша работа как программиста при решении конкретной задачи на компьютере с помощью языка программирования Турбо Паскаль состоит из определённого ***набора этапов***, которые постоянно выполняются программистом от программы к программе.

❶ Всё начинается с постановки и решения задачи в ее предметной области: будь-то алгебра, геометрия, электротехника, горное дело и т.д. для получения конкретных конечных результирующих формул (сам Турбо Паскаль за Вас этого не сделает!). К примеру, вычисление полной поверхности прямоугольного параллелепипеда можно выразить следующим образом:

$$S_{полн} = 2 \cdot (a \cdot b + h \cdot (a + b)).$$

❷ Для проведения вычислений, следует осуществить выбор структур входных и выходных данных, которые необходимо описать в разрабатываемой программе и их типы:

```
Spoln, a, b, h : Real.
```

❸ Далее следует уточнение раздела описаний программы ТП, начиная с модулей, которые необходимо подключить (раздел `Uses`). Нужен ли Вам раздел описаний констант и какие из них необходимо в него включить? Нужен ли раздел описаний типов? Нужен ли раздел описаний переменных? и т.д. В результате получаем:

```
Var
     Spoln, a, b, h : Real;
```

❹ Теперь производится разработка интерфейса программы, то есть организация ввода-вывода данных на языке ТП. Здесь Вы должны определить, какие данные необходимо ввести для решения задачи, а какие – вывести после её решения. К примеру, нижеследующий фрагмент программы выведет на экран следующую информацию (рис. 8.6).

```
Write('Введите стороны основания прямоугольного
       параллелепипеда a и b и высоту h : ');
ReadLn(a, b, h);
. . . { Здесь располагаем алгоритм решения задачи }
WriteLn('Полная поверхность прямоугольного параллелепипеда
         = ', Spoln);
```

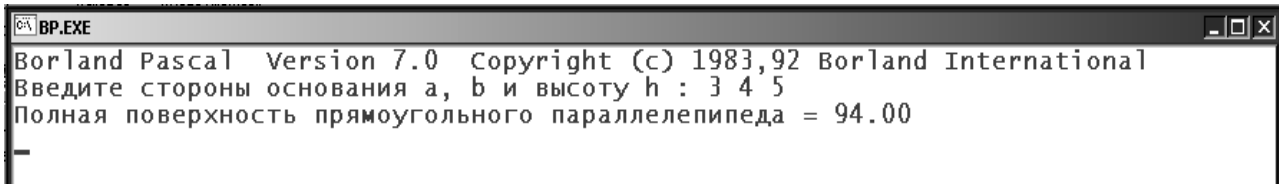


Рис. 8.6 Простой интерфейс программы вычисления полной поверхности прямоугольного параллелепипеда

❺ Далее выбираем необходимые управляющие структуры и операторы языка ТП (`:=, if, case, for, repeat, while`) и его встроенных (стандартных) функций (а может быть нужно разработать дополнительные функции пользователя?) для реализации алгоритма решения задачи.

❻ Производим разработку алгоритма[4] решения задачи на языке Турбо Паскаль. Заметьте как похоже выражение алгоритма на языке ТП на исходную формулу:

```
Spoln := 2*(a*b+h*(a+b));
```

❼ Кодируем алгоритм программы на языке ТП. Результатом этой фазы является готовая программа, в которой программист объединяет алгоритм с избранными структурами данных. Недаром Н. Вирт, в свое время, высказал положение, что "программа = структуры данных + алгоритм".

❽ Вводим текст программы на алгоритмическом языке Турбо Паскаль в окне редактора ИСР ТП с помощью клавиатуры для последующего сохранения его в текстовый файл[5] на диске с именем, указанным пользователем, и расширением **.pas** (рис. 8.7). Вам необходимо помнить, что компилятор ТП ***не чувствителен к регистру*** (Not case sensitive). Использование прописных букв обычно диктуется обеспечением удобочитаемости программ. Например, имя $S_{полн}$ в исходной формуле – в программе записываем в виде `Spoln`, название процедуры чтения строки (Read Line) – `ReadLn` и т.д., поскольку Турбо Паскаль оперирует только английскими лексемами. Русскоязычный текст может быть представлен только строками в кавычках (тип данных ТП `String`, к примеру, `'Площадь'`).

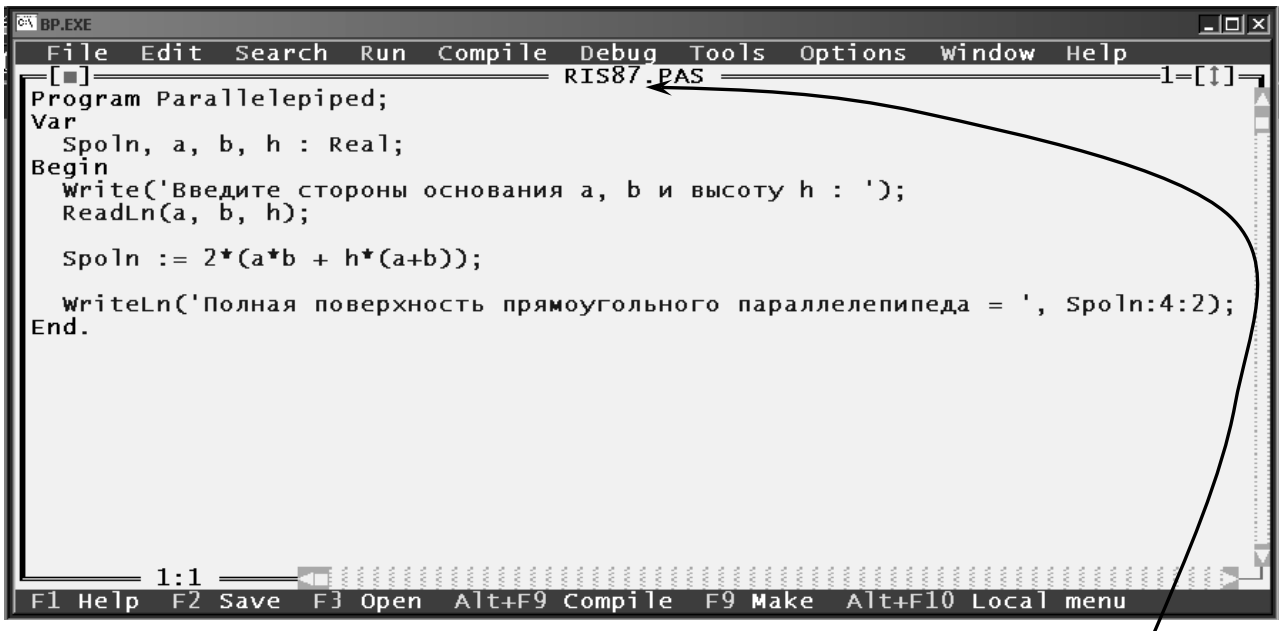


Рис. 8.7 Текст программы, введённый в окне редактора ТП в файл RIS87.PAS

❾ Компилируем текст программы. При этом Вы должны хорошо понимать, что ***для облегчения построения компиляторов*** язык программирования описывается в терминах некоторой ***грамматики,*** обеспечивающей однозначность трактовки записываемых действий. Эта грамматика также определяет ***форму*** (***синтаксис***) допустимых предложений

[4] Алгоритм – последовательность действий или шагов, выполнение которых позволяет решить конкретную задачу.

[5] Файл – поименованная область на диске, которая содержит двоичные данные.

языка. Поэтому компилятор начинает свою работу с определения правильности составленных Вами предложений программы на языке Турбо Паскаль и выдаёт соответствующую диагностику в местах синтаксических ошибок. Для вызова компилятора в ИСР ТП необходимо либо с помощью «горячих клавиш» **Alt+C** вызвать меню **Compile** и в нём выбрать команду **Compile**, либо использовать «горячие клавиши» **Alt+F9**. ИСР ТП определяет по одной ошибке за один запуск компилятора, но при этом «любезно» указывает курсором место нарушения синтаксиса или грамматики языка. Одновременно с этим на экран выводится пояснительное сообщение о характере ошибки на «чистом» английском языке (рис. 8.8):

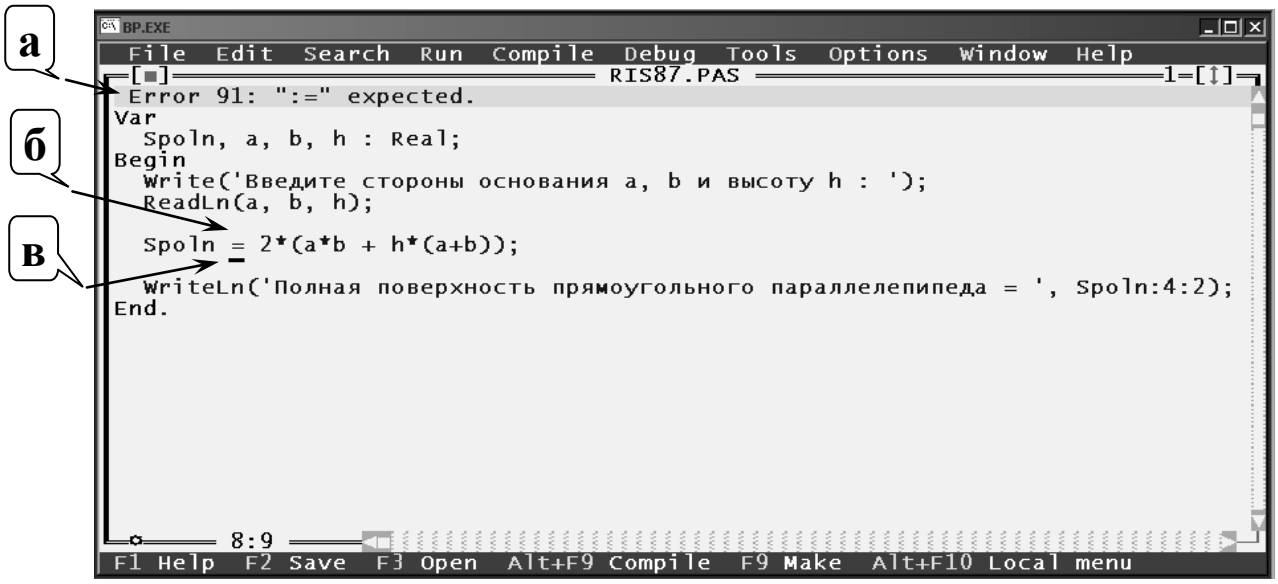


Рис. 8.8 Реакция («диагностика») компилятора ТП (а) на неправильный ввод оператора присваивания («=» вместо «:=») (б). Обратите внимание на положение курсора в месте найденной ошибки (в).

Если в Вашей программе компилятором ошибок не найдено, то появляется окно с сообщением «**Compile successful: Press any key**» («Компиляция успешна: Нажмите любую клавишу») (рис. 8.9). После нажатия любой клавиши это окно исчезает.

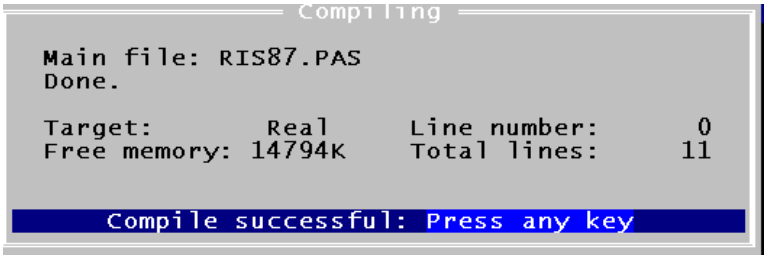


Рис. 8.9 Сообщение ИСР ТП об успешной компиляции

❿ Если программа написана и введена без ошибок (к примеру, в текстовый файл `first.pas`) и не требует дальнейшей отладки[6], то необходимо выбрать команду **Run** из меню **Run** либо использовать «горячие клавиши»

[6] Отладка – поиск ошибок в алгоритме и тексте программы.

Ctrl+F9 для преобразования ***исходного текста*** Вашей программы на языке Турбо Паскаль в ***двоичный загрузочный*** EXE-файл, содержащий исполняемый машинный код (т.е. ***команды, которые «понимают» и могут выполнять процессоры Intel либо AMD***) (рис. 8.7). Этот файл сохраняется на диске, а с диска загружается программой-загрузчиком в оперативную память (ОП) компьютера для ее дальнейшего выполнения. Загруженная в ОП программа во время своей работы интерактивно[7] общается с пользователем, в соответствии с интерфейсом[8], который был Вами запрограммирован.

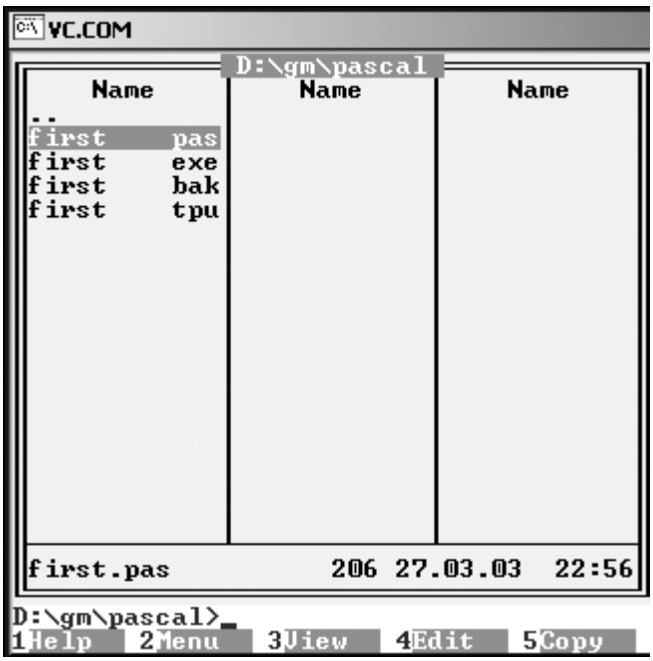


Рис. 8.7. Четыре формы существования программы на диске D:, содержащейся в файлах с именем `first` и разными расширениями в каталоге `pascal`:
❶ first.pas –исходный файл (т.е. текстовый);
❷ first.bak - резервная копия исходного файла;
❸ first.tpu – результат компиляции на диск содержимого first.pas – модуль Турбо Паскаля;
❹ first.exe – загрузочный (исполняемый) файл.

❶❶ В случае некорректной работы отлаживаем программу средствами дебаггера ИСР Турбо Паскаль. Обычно это необходимо в случае, когда Ваша программа работает, но делает не то, что Вы предполагали. Также дебаггер полезен и даже необходим для исследования Ваших программ на этапе обучения для лучшего понимания поведения управляющих конструкций ТП.

Авторы Турбо Паскаля, учитывая такой поэтапный цикл работы программиста, решили в своей интегрированной среде разработки много сложных задач. Среди главных, следует назвать такие.

❶ Процесс подготовки программы к запуску в рамках ИСР Турбо Паскаль, то есть ввод ее текста в окне редактора, компиляция этого текста и загрузка полученного EXE-файла на выполнение ведется комплексно в одном приложении. Последние два этапа можно выполнить либо последовательно командами **Compile** (**Alt+F9**) и **Run** (**Ctrl+F9**), либо вообще одной командой **Run** (**Ctrl+F9**).

❷ Компиляция программы сделана удобной для новичков и профессионалов благодаря точному указанию места ошибки и её описания.

❸ Интерактивный режим работы программы (после ее запуска командой **RUN** или нажатия «горячих клавиш» **Ctrl+F9**) реализован в виде совместного

[7] Интерактивный (диалоговый) – режим, в котором пользователь задаёт программе команды и вводит данные пока она выполняется. Такой режим предусматривает обмен текстовыми командами (запросами) и ответами (приглашениями) программы.

[8] Интерфейс – способ сопряжения двух либо более, различных систем, что обеспечивает их логическое либо физическое взаимодействие.

существования двух окон: «чёрного» окна DOS и «синего» окна редактора ТП (рис. 8.8) (таблицу сочетаний клавиш ТП см. в Приложении 5). После выполнения программы процесс заканчивается возвращением из «чёрного» окна DOS в «синее» окно редактора ТП. Чтобы просмотреть результат работы Вашей программы, Вы должны нажать «горячие клавиши» **Alt+F5**. Возвращение обратно в «синее» окно редактора ТП осуществляется нажатием любой клавиши.

К примеру, программу MyProgram запускаем нажатием клавиш* Ctrl+F9 *в окне редактора Турбо Паскаль, где был набран её текст. Результат работы программы выводится в окне DOS. Переход к окну DOS – нажатие клавиш* Alt+F5, *в окно редактора – нажатие любой клавиши (рис. 8.8)**.**

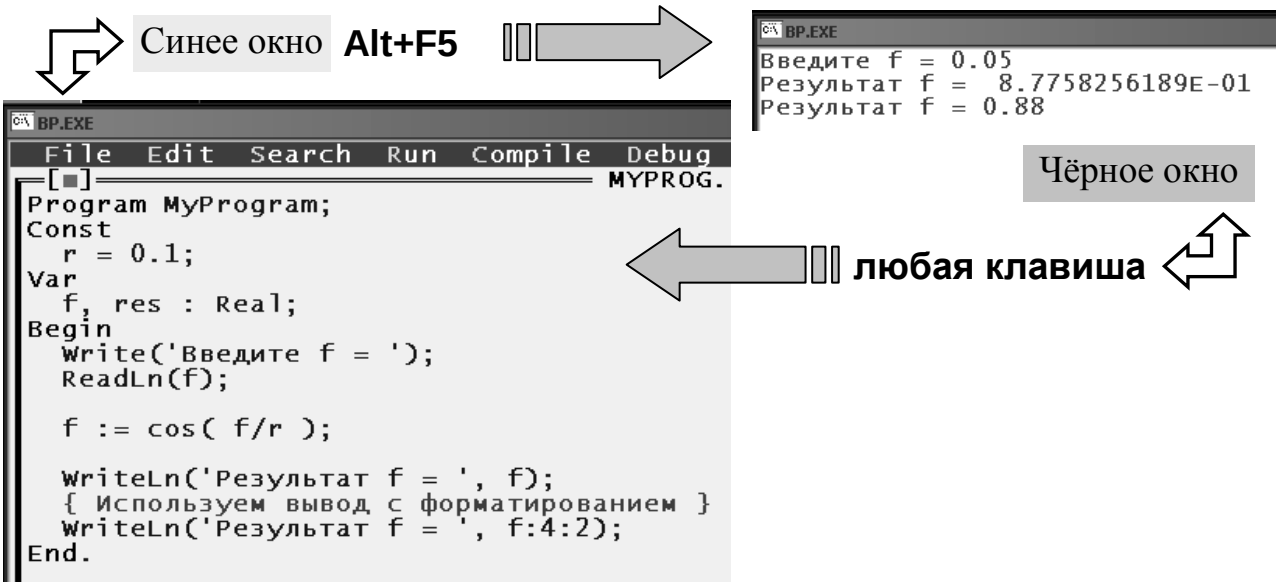


Рис. 8.8 Два окна, с которыми работает пользователь при выполнении своей программы (в данном случае – `MyProgram`)

❹ Процесс отладки и исследования работы программы в ИСР Турбо Паскаль сделан максимально удобным: можно выполнять программу пооператорно, устанавливать контрольные точки, контролировать содержимое переменных (видеть на экране, какие данные в них находятся в текущий момент и как они изменяются в процессе работы) и т.д. Меню ИСР ТП и его команды описаны в Приложении 4.

Первая версия ИСР ТП была создана для использования на персональных компьютерах с операционной системой[9] (ОС) MS DOS, которая была рассчитана на одного пользователя. Поэтому ее вызов выполнялся довольно просто. Нужно было лишь найти на жёстком диске подкаталог `BIN` каталога

[9] Операционная система (ОС) - совокупность программных средств, которые обеспечивают управление аппаратными ресурсами компьютера и взаимодействие программных процессов с аппаратурой, другими процессами и пользователем. ОС выполняет следующие функции: управление памятью, вводом-выводом, файловой системой, взаимодействием процессов, диспетчеризацией процессов, защиту, учёт использованных ресурсов, обработку командного языка.

TP70, указать имя TURBO или TURBO.EXE в командной строке и нажать на клавишу **Enter** (рис. 8.9).

```
C:\>cd tp70
C:\TP70>cd bin
C:\TP70\BIN>turbo
```

Рис. 8.9 Последовательность команд ОС MS DOS
для вызова ИСР ТП

Сейчас же, в компьютерных классах учебных заведений, компьютеры объединяются в сети под управлением ОС Windows NT или версий Windows 2000 и выше. Это накладывает дополнительные условия на процесс работы учащихся с IDE ТП.

Логические[10] диски C: и D: для записи какой-либо информации и доступа пользователю обычно закрыты, поскольку на них расположены системные программы и офисные приложения[11]. Поэтому, для вызова ТП нужно выполнить следующие шаги.

1. Нажать клавиши **Ctrl+Alt+Del**, для того, чтобы войти в операционное пространство на жестком диске, выделенное пользователю на основании установок системного администратора сети. Операционное пространство – это ресурсы системы, в том числе и необходимые для работы программные средства (файл-менеджеры, системы программирования, графические и текстовые редакторы и др).

2. После нажатия совокупности клавиша **Ctrl+Alt+Del** нужно ввести имя пользователя и его пароль в две первые строки диалогового окна операционной системы (рис. 8.10).

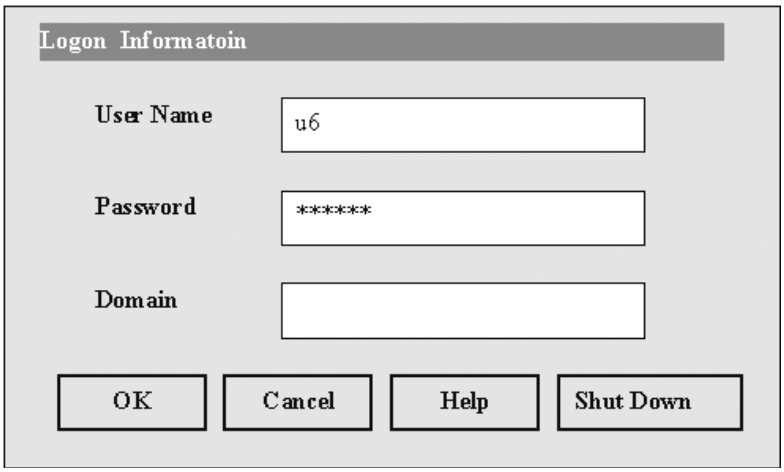


Рис. 8.10. Диалоговое окно для входа в ОС Windows NT

[10] Логический диск – физическое пространство на диске, к которому пользователь обращается по известному операционной системе имени (A: – гибкий диск, дискета, C: , D: , E: , …, Z: – рабочие участки на жёстком диске или логические диски).

[11] Приложение – программа или группа программ, которые разработаны для конечных пользователей и работают под управлением операционной системы. Сюда входят текстовые процессоры, электронные таблицы, программы баз данных и т.д.

Пароль (**Password**) отображается звездочками для того, чтобы посторонний не смог его узнать. Переход из окна **User Name** к окну **Password** можно выполнять или курсором мыши, или клавишей **Tab**.

3. После появления экрана с изображением рабочего стола ОС Windows нужно навести курсор мыши на кнопку **Start** (**Пуск**) и щёлкнуть по ней левой кнопкой мыши. Затем раскрыть меню **Program** (**Программы**), найти строку с надписью **Turbo Pascal** и щелкнуть по ней левой кнопкой мыши.

4. В раскрытом окне ТП в меню **File** (рис. 8.11) щелкнуть по команде **Change dir** (Изменить каталог). В открывшемся диалоговом окне выбрать нужный логический диск (G:, Z: и т.д.) и каталог, в котором находятся программы ТП с расширениями **.pas**, которые принадлежат пользователю (рис. 8.12).

Рис. 8.11 Раскрытое меню **File,** его команды работы с файлами и активизированная команда **Change dir**

Рис. 8.12. Диалоговое окно команды **Change dir,** в котором Вы должны выбрать необходимый каталог (директорий) на конкретном логическом диске

Обратите внимание на то, что переход между различными областями окна **Change Directory** осуществляется клавишей **Tab**. В окне Directory tree перемещения между пунктами осуществляется стрелками ↑ и ↓, а раскрытие подпунктов – клавишей **Enter.**

После этого Вы можете открывать новое окно для набора текста программы (Команда **New** из меню **File**) или редактировать и запускать для выполнения какую-либо из ранее введенных Вами программ (Команда **Open** из меню **File**). Меню **File** содержит следующие команды, обеспечивающие Вашу работу с файлами, располагаемыми на жёстком диске.

❶ **New** – открытие нового окна для ввода редактирования программы с именем **NONAME00.PAS**. Повторное выполнение команды **New** открывает новое окно с именем **NONAME01.PAS** и т.д. Переход между открытыми в окне редактора окнами с текстами программ – **F6**, закрытие текущего окна – **Alt+F3.**

❷ **Open** – выбор и загрузка текстового файла с диска для его последующего редактирования и запуска на выполнение. Синоним команды – клавиша **F3.**

❸ **Save** – сохранение на диске в текущем каталоге редактируемого файла и продолжение редактирования. Синоним команды – клавиша **F2.**

❹ **Save as** – запись текущего редактируемого файла на диск в текущий каталог под новым названием.

❺ **Save all** – сохранение всех изменённых файлов на диске.

❻ **Change dir** – изменение текущего каталога.

❼ **Print** – печать содержимого текущего окна.

❽ **Printer setup** – установка текущих настроек принтера.

❾ **DOS shell** – временный выход в режим **DOS**, например для выполнения команд удаления или переименования файлов. Возврат обратно происходит при выполнении команды **Exit.**

❿ **Exit** – выход из ИСР ТП. Синоним команды – **Alt+X.**

Вопросы и упражнения

1. Кто создал язык программирования Паскаль и с какой целью?
2. Что представляет собой Турбо Паскаль?
3. Какие функции выполняет компилятор языка Турбо Паскаль?
4. Каковы основные этапы при решении конкретной задачи на компьютере с помощью языка программирования Турбо Паскаль?
5. Загрузить Турбо Паскаль и в окне редактирования программ набрать программу Parallelepiped (рис. 8.7). Этот текст записать на диск в файл с именем **PARALL.PAS**. Закрыть окно редактирования.
6. Командой **NEW** меню **FILE** открыть новое окно редактирования для работы с файлом. Набрать текст программы рис. 8.8. Набранному тексту присвоить имя **NAME2.PAS** (переименовать файл). Дале записать файл **NAME2.PAS** на диск.

7. Просмотреть текущий каталог и найти файл с именем **PARALL.PAS.** Загрузить этот файл в окно редактирования. Отредактировать файл (что-то изменить в тексте файла). Затем закрыть текущее окно редактирования и открыть новое для работы с новым файлом.
8. Загрузить в окно редактирования файл **NAME2.PAS**. Затем записать этот файл в другой каталог текущего диска. Переименовать файл **NAME2.PAS** в файл **NAME3.PAS** и записать его на диск.
9. Просмотреть историю работы с файлами. Найти предпоследний файл, с которым Вы работали, и загрузить его в окно редактирования.
10. Временно выйти из Турбо Паскаля в MS-DOS и возвратиться обратно.
11. Загрузить в окно редактирования файл NAME1.PAS, провести редактирование текста файла и выйти из среды Турбо Паскаля (то есть закончить работу в ней).

8.3. Строительные блоки (базовые элементы) программ на языке ТП

Программа на языке ТП, как отображение составленного Вами алгоритма, содержит совокупность объектов (лексем[12]), которые с помощью компилятора языка ТП преобразуются в язык понятный компьютеру – набор машинных команд. Эти команды записываются в EXE-файл, который, либо сразу же загружается в оперативную память компьютера и там выполняется центральным процессором, либо сохраняется на жёстком диске, откуда может затем неоднократно вызываться для выполнения с Вашими новыми данными.

Каждая Ваша программа, написанная на языке ТП, делится на ***лексемы*** и ***разделители***. К разделителям, которые отделяют одну лексему от другой, относятся пробелы, символ "**;**" и комментарии (рис. 8.13). Таким образом, две рядом стоящих лексемы должны отделяться одним или большим количеством разделителей.

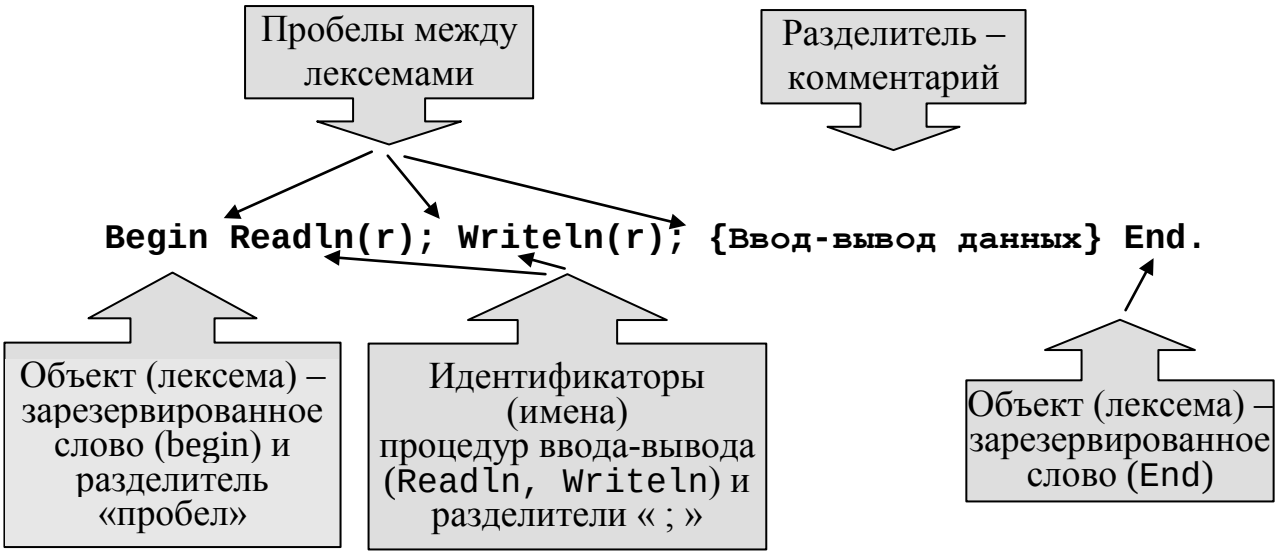


Рис. 8.13. Лексемы языка ТП, записанные в строку. Обычно записываются в столбец

Объекты программы (лексемы) разделяются на группы. К первой группе относятся элементы алфавита языка ТП. Он содержит следующие буквы и цифры в восьмибитном коде ASCII[13] (табл.8.1) (см. Приложение 6).

Таблица 8.1

Символы алфавита языка ТП.

Большие буквы английского языка	От A до Z
Малые буквы английского языка	От a до z
Десятичные (арабские) цифры	От 0 до 9
Шестнадцатеричные цифры	От 0 до 9 и от A до F

[12] Лексема – минимальная единица значений текста в программе.

[13] ASCII (American Standard Code for Interchanging Information) – восьмибитный код, который отображает все необходимые для передачи информации в компьютере символы. Сюда входят буквы, цифры, специальные символы и др.

Ещё раз напомним, что ТП не отличает больших букв от маленьких, поэтому он одинаково правильно поймет слова `BEGIN` и `begin`, поэтому, Вы сами должны использовать большие и маленькие буквы для придания Вашей программе большей информативности.

Вторую группу объектов программ составляют специальные символы. Они могут быть простыми и составными (табл. 8.2, 8.3):

Таблица 8.2

Специальные простые (одиночные) символы языка ТП

Простые одиночные символы	
Название символа	Символ
Добавления	+
Вычитания	-
Умножения	*
Деления	/
Равенство	=
Меньше	<
Больше	>
Открывающая квадратная скобка	[
Закрывающая квадратная скобка	]
Точка	.
Запятая	,

Простые одиночные символы	
Название символа	Символ
Открывающая круглая скобка	(
Закрывающая круглая скобка	)
Двоеточие	:
Точка с запятой	;
Спецсимвол	@
Открывающая фигурная скобка	{
Закрывающая фигурная скобка	}
Знак доллара	$
Спецсимвол «шарп»	#

Таблица 8.3

Специальные составные символы языка ТП

Составные символы	
Название символа	Символ
Меньше либо равняется	<=
Больше либо равняется	>=
Присвоить	:=
Диапазон	..
Открывающая круглая скобка со звездочкой	(*
Закрывающая круглая скобка со звездочкой	*)
Открывающая круглая скобка с точкой	(.
Закрывающая круглая скобка с точкой	.)

В третью группу входят так называемые "зарезервированные" служебные слова. Это название указывает на то, что такие слова следует использовать только в одном значении, которое им предназначено раз и навсегда в языке ТП.

Поэтому Вы не имеете права использовать их по другому назначению. В таблице 8.4 приведены все зарезервированные слова языка ТП.

Таблица 8.4

Зарезервированные (служебные) слова ТП

absolute	end	inline	procedure	type
and	external	interface	program	unit
array	file	interrupt	record	until
begin	for	label	repeat	uses
case	forward	mod	set	var
const	function	nil	shl	while
div	goto	not	shr	with
do	if	of	string	xor
downto	implementation	or	then	
else	in	packed	to	

Следует также заметить, что к зарезервированным, т.е. запрещённым к использованию в других целях и смыслах, относятся имена стандартных (встроенных) функций и процедур языка Турбо Паскаль: `cos, sin, Write, Read` и др. Обратите также внимание на то, что названия стандартных типов отсутствуют в списке зарезервированных слов!

В четвертую группу обязательных элементов программ входят так называемые "идентификаторы[14] ", которые представляют собой имена[15] сущностей (размеров, длин, площадей и других исходных данных) в Вашей программе, которые Вы вводите для обозначения объектов, необходимых для реализации созданного Вами алгоритма.

Идентификаторами Вы обозначаете в своей программе так называемые «пользовательские типы данных» – объекты для хранения Ваших данных различных типов, а также процедуры, функции, модули, программы и поля записей. Идентификатор может иметь длину до 126 символов – это длина программной строки в ТП, но для ТП значение имеют только первые 63 символа. Остальные символы значения не имеют, но сохраняются. Идентификатор должен начинаться с буквы или знака подчеркивания (_) и ***не может*** содержать ***разделителей (пробелов, комментариев и точки с запятой)***. После первого символа идентификатора можно использовать буквы, цифры и символы подчеркивания. В зарезервированных словах и в идентификаторах можно использовать как строчные, так и прописные буквы (компилятор их не различает). Здесь важен смысл, который Вы вкладываете в каждый из идентификаторов – ведь Вы можете вернуться к своей программе через продолжительное время и не вспомнить, что Вы задумывали в ней делать!

[14] Идентификатор – неделимая последовательность символов алфавита, которая образует имя используемого объекта.

[15] Имя является символическим представлением сущности. Сущность может являться объектом либо некоторым выполняемым действием. Сущность может иметь большое количество имён. Каждое имя имеет смысл только в границах некоторого именного контекста. Под контекстом понимается фрагмент документа, в пределах которого можно уяснить значение отдельного слова либо объекта. Только в контексте слово либо объект получают конкретное значение.

Приведем несколько примеров идентификаторов:

```
WriteFile          Dos_Exec          Int22String5
Exit_22            First_Name        NameOfTheGame
```

Заметьте, что в идентификаторах ТП каждый из входящих в них смысловых компонентов, начинается с прописной буквы.

Таким образом, для правильной реализации Ваших алгоритмов на языке ТП важно запомнить следующее.

❶ Имена (идентификаторы) не могут начинаться с цифры и содержать в себе русские или украинские буквы и символ пробела.

❷ В именах допускается использование больших (прописных) и маленьких английских букв, цифр от 0 до 9 и символа подчеркивания (_).

❸ Количество символов в идентификаторах (именах) ограничивается редактором ТП до 126, а сам компилятор ТП отличает один идентификатор от другого по первым 63–м символам.

❹ Для повышения содержательности идентификаторов в языке ТП принято конструировать их по принципу, когда несколько слов пишутся одно за другим, но каждое с большой буквы.

```
VyViditeIdentifikatorDlaHraneniyaPeremennoyTipaReal:=0.265;
VyViditeIdentifikatorDlaHraneniyaPeremennoyTipaReal2:=54.3;
```

Вышеприведенные идентификаторы различаются компилятором ТП, поскольку первый содержит 51 символов, а второй – 52, из которых первые 51 совпадают. Для большей наглядности в идентификаторы можно вводить символ подчёркивания.

```
Vy_Vidite_Identifikator_Dla_Hraneniya_Tipa_Real := 0.005;
```

Упражнения

1. Объясните понятие «Базовые элементы языка Турбо Паскаль». Приведите примеры.

2. Объясните понятие: алфавит языка программирование, ключевое слово, минимальная смысловая единица языка. Каково назначение этих понятий в тексте программы? Приведите примеры.

3. Объясните понятие идентификатора. Сформулируйте правила написания идентификатора. Объясните назначение идентификатора в программе.

4. Что такое разделитель в тексте программы, для чего служат разделители? Приведите примеры.

5. Объясните понятие «комментарий» и его назначение в тексте программы.

6. Чем простые символы отличаются от составных символов?

7. Какими символами может начинаться идентификатор, а какими не может?

8.4. Константы, переменные и их типы

Все данные (числа, значения вычисленных выражений, тексты, результаты сравнений различных объектов и др.) в Вашей программе компилятор ТП интерпретирует как ***константы*** и ***переменные***.

Константы в процессе работы программы никогда не изменяют свое значение. Любое число, которое появляется в программе в своём обычном виде (например: 6.0 или 20) называется ***константой***. Величина, которой Вы определили наименование, может принимать в процессе вычислений разные числовые значения и носит название ***переменной***. В соответствии с тем, какие данные Вы собираетесь размещать в переменных, Вы должны назначить им соответствующие ***типы***, которые определяют допустимые с ними операции (табл. 8.5).

Таблица 8.5

Наиболее полезные типы данных и операции, которые применимы к ним

Наименование типа данных	Способ представления данных этого типа		Допустимые для этих типов данных операции
`Real` (вещественные числа)	1.0 0.0012	1E-6 -2.639E4	+, –, /, *.
`Integer` (целые числа)	1 -56	2892	+, –, *, `DIV`, `MOD`.
`Char` (символы)	A, 'A', 1, '1'		операции сравнения и сцепления (+) т.е. конкатенации
`String` (строки)	'Type mismutch!' 'Украинский язык' 'Не забудьте ввести!'		операции сравнения, сцепления (+) т.е. конкатенации и преобразования
`Boolean` (логические значения)	False, (неправда), (ложь) True (правда) (истина)		операции сравнения, логические операции
`Array` (массивы)	Последовательности значений всех вышеуказанных или других типов		все операции, которые допустимые для соответствующих типов

Примечание: 1) К операциям ***сравнения*** принадлежат следующие: = – равно; <> – не равно; > – больше; < – меньше; >= – больше или равно, <= – меньше или равно.

2) ***Логические*** операции включают: OR – логическое сложение (ИЛИ), AND – логическое умножение (И), NOT – операция отрицания (НЕ), XOR – исключающее ИЛИ.

Целые и вещественные константы различаются по наличию или отсутствию десятичной точки. Так, 3 – целая константа, а 3.0 или 3.000000 –

вещественные константы и способ записи их в памяти ПК и арифметические операции, которые с ними можно выполнять имеют свои особенности.

Чтобы упростить запись очень больших и очень маленьких чисел, используется следующий способ. После вещественной константы располагается буква E и одно– или двузначное целое число с соответствующим знаком (+ или –). Такая запись указывает на то, что начальная константа должна быть умножена на 10 в указанной степени.

Например, константы:

+15.016 (можно записать в виде 1.5016E01 или 1.5016E+01);

5.0E+6 (можно записать в виде 5000000.0);

0.0000173 (можно записать в виде 1.73e–5).

Термин "переменная" используется в языке ТП для обозначения величины, обращение к которой производится по ее наименованию (имени, идентификатору) и которая может принимать различные значения, а не ограничена каким-либо одним. То есть, во время работы программы ***имя*** и ***тип*** каждой переменной постоянны, а свои ***значения*** переменные могут изменять многократно. Типы (целый, вещественный, логический, символьный и др.) и имена переменных Вы должны описать в начале основной (главной) программы с помощью специальных зарезервированных слов (см табл. 8.5).

Тип данных (и, соответственно, идентификатор переменной любого типа!) определяет:

❶ способ представления элементов совокупности (элемент: цифровой, текстовый, логический и др.);

❷ множество допустимых значений переменных данного типа;

❸ множество допустимых операций, которые применяются к данному типу данных (и идентификаторам, имеющим данный тип) (см. табл. 8.5);

❹ размер области памяти, занимаемой переменной данного типа.

В таблице 8.6 приведенные примеры описания констант и переменных при использовании их в программах на языке Турбо Паскаль.

Таблица 8.6

Описание переменных и констант в программах ТП

Описание констант (перед знаком “=” указывается имя константы, см. далее в п.8.5)	Описание переменных (после двоеточия обозначенный тип переменной)	Число байтов, отводимых в памяти ПК под одну переменную
`Const`	`Var`	
`Rand = 5.2E-5;`	`R,t,p : real;`	6
`Norma = - 54;`	`I,j,k : integer`	2
`T = FALSE;`	`a : Boolean;`	1
`Symbol='R';`	`c : char;`	1
`My_Name = 'Alex';`	`S1 : string;`	256

Таким образом, константы в языке ТП могут быть безымянными (10; -0.83), либо представляться в выражениях своими именами (табл. 8.6).

С учётом всего вышесказанного, интересно привести разные представления в компьютере числовой величины 2 (два) (табл. 8.6.а).

Таблица 8.6.а

Представление в компьютере числовой величины 2 (два)

Представление числовой величины	Тип значения в ТП	Число байтов, отводимых в памяти ПК	Представление в памяти ПК
2	Integer	2	00000000 00000010 (Двоичное представление числа 2)
2.0	Real	6	10000000 00000000 00000000 00000000 00000000 00000001
‘2’ (ASCII-код 50)	Char (символ)	1	00101000 (Двоичное представление числа (ASCII кода) 50) (см.п. 8.25)
‘два’ (ASCII-коды: 51, 184, 182, 180)	String (строка)	4	00000011 10111000 10110110 10110100 (см. пп. 8.25.3)

Нелишне напомнить, что ***операции*** с приведенными в табл. 8.6.а ***представлениями разных типов данных***, существенно различаются (см. табл. 8.5). Вы должны это хорошо себе представлять, поскольку в программах, переменные и константы всех типов объединяются в ***выражения***.

Выражения задают порядок выполнения действий над элементами данных и состоят из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков операций. Операции, в свою очередь, определяют действия, которые следует выполнять с операндами (см. таб. 8.7).

Таблица 8.7

Примеры реальных выражений и их записи на языке ТП

Тип выражения и переменных в нем	Математическое (или другое) выражение	Реализация на Турбо Паскале
Вещественный	$6*\sqrt[4]{\frac{a^2\cdot(\frac{b}{c})}{\frac{b+c}{2}}}$	`6*sqrt(sqrt((a*a*(b/c))/((b+c)/2))`
Целый	$\frac{a}{2}\cdot(x^2\cdot\frac{a+b}{c})$	`a div 2 * (sqr(x)*((a+b) div c)`
Логический	$\neg(A\vee B)\wedge(C\vee\neg D)$	`NOT(A OR B) AND (C OR NOT D)`
Символьный	Д + А = ДА	‘Д’ + ‘А’
Строковый	Сегодня вторник	‘Сегодня вторник’
Примечание: `sqrt(x)` – функция извлечения квадратного корня из значения `x`; `sqr(x)` – возведение значения *x* во вторую степень; `div` – математическая операция деления нацело одного целого числа на другое; `NOT, OR, AND` – логические операции (соответственно: отрицания, логического сложения и логического умножения).		

Операции, которые применяются к операндам, могут быть унарными (одноместными) и бинарными (двуместными). Унарные, это такие, как изменение знака числа, применяются к одиночным операндам, а бинарные – к двум или более (см. табл. 8.8).

Таблица 8.8

Одноместные и двуместные операции в языке ТП

Тип выражения и переменных в нем	Тип операции	Математическое (или реальное) выражение	Реализация на Турбо Паскале
Вещественный, целый	Одноместная	-1.35; $\lvert -30 \rvert$ (модуль числа)	`- 1.35; abs(-30)`
	Двухместная	$5.1 \cdot 25$; $7+0.35$	`5.1* 25;  7 + 0.35`
Логический	Одноместная	$\neg D$	`NOT D`
	Двухместная	$A \vee B$	`A OR B`

Термин "выражение" используется в языке ТП в том же значении, что и в обычной математической записи действий, производимых согласно какой-либо формуле. Выражение может конструироваться из констант, переменных и функций (встроенных или созданных пользователем), которые объединяются между собою символами операций, запятыми и скобками.

Вещественные и целые числа можно "смешивать" в вещественных выражениях, но нельзя в целочисленных выражениях, так как арифметика этих двух типов базируется на абсолютно разных принципах. Для вещественных типов `Real` количество цифр, которые могут сохраняться в памяти компьютера, равняется 11-ти или 12- ти. Поэтому арифметические операции, вдобавок, с дробями конечной длины не всегда точно отвечают обычным правилам арифметики. Чтобы это понять рассмотрим следующее выражение:

$$\underbrace{0.40000000000}_{\text{11 цифр}} + \underbrace{123456789778.0}_{\text{11 цифр}} - \underbrace{123456789777.0}_{\text{11 цифр}}$$

Поскольку операции будут выполняться слева направо, то результат добавления с точностью 11 знаков будет равняться 123456789778.0 и 0.4 будет безвозвратно утеряно. После вычитания 123456789777.0 конечный результат будет равняться 1.00000000000. Но, если выражение записать со скобками

$$0.40000000000 + (12345678978.0 - 12345678977.0)$$

то предварительное выполнение операции вычитания в скобках, а потом только добавление первого числа (0.4) даст верный результат 1.4.

Неверный порядок выполнения действий может привести не только к потере точности, но и вообще к полной неудаче. Предположим, Вам необходимо вычислить выражение `A*B/C`, в котором все величины: `A`, `B` и `C` имеют значения 10^{30}. Сначала будет выполнено умножение, которое даст 10^{60}, что превышает значение величины, которая может содержаться в переменной типа `real`. Поэтому компилятор ТП выведет предупреждение:

```
Error 205: Floating point overflow.
```

(Ошибка 205: Переполнение числа с плавающей точкой)

и решение задачи будет прекращено. Но, если те же самые данные описать типом `double`, абсолютная величина которого изменяется в диапазоне

5.0E–324..1.7E+308, то вычисления будут проведены корректно.

При делении целых чисел в ***рамках языка программирования ТП*** также возникают свои особые проблемы. Когда Вы делите целое число, скажем, 7 на целое число 3 на бумаге, то результат у Вас будет дробным числом. И количество знаков после запятой Вы выберете сами. ***Но машинная арифметика – совсем другое дело!*** Если Вы имеете дело с целыми числами в ТП, т.е. описанными как `integer`, то, во-первых, операция деления должна быть записана специальным образом – «`div`», а во-вторых, результат будет получен с отбрасывание дробной части!

`5 div 3 = 1` (5 делится на 3 нацело!);

`6 div 7 = 0` (6 делится на 7 нацело!) и т.д.

Чтобы уверенно чувствовать себя при программировании различных выражений в ТП Вы должны знать о его компиляторе следующее. Первым делом он устанавливает тип переменной слева от знака присваивания, затем проверяет типы операндов, участвующих в выражении справа от знака присваивания, а затем – типы операций, совместимые с типами данных операндов. Зная Ваши привычки при «вычислениях на бумаге» он преобразует целые числа к вещественному типу, но не наоборот! Лучше всего это можно понять из примера, приведенного на рис. 8.14.

```
var x, y : real; i, j : integer;
Begin
  x := 5 / 2; { поскольку слева стоит вещественная величина, то 5 преобразуется в 5.0, а
                2 – в 2.0 и производится операция вещественного деления, которая даёт
                вещественное значение 2.5, а оператор присваивания занесёт его в
                вещественную переменную x }
  y := 5 div 2;
                {операция целочисленного деления при условии наличия слева от оператора
                присваивания вещественной переменной y, выполняется компилятором как
                целочисленное деление, а затем результат преобразуется в вещественное число
                и в переменную y заносится вещественное значение 2.0! }
  i := 5 div 2; {операция целочисленного деления даст целое значение 2,
                 которое и будет занесено в целочисленную переменную i }
  j := 5 / 2;   {операция вещественного деления не будет выполнена и
                 компилятор выдаст следующую диагностику }
 Error 26: Type mismatch.
```

{что переводится, как «***несовпадение типов (данных) справа и слева от оператора присваивания»*** }

```
  i := 5.3 div 2.6; {компилятор откажется выполнять эту операцию целочисленного
                     деления с вещественными числами и выдаст следующую
                     диагностику }
 Error 41: Operand types do not match operator.
```

{что переводится, как «***тип операндов не соответствует оператору»*** }

Рисунок 8.14. Взаимодействие типов операций с типами переменных

Неожиданности могут случаться и в операциях с вещественными числами. Рассмотрим следующую сумму:

1.0 / 3.0 + 1.0 / 3.0 + 1.0 / 3.0.

Результат деления 1.0 / 3.0 равняется с точностью до 11 знаков 0.33333333333. Общая сумма будет равняться 0.99999999999, а никак не 1.00000000000!

Таким образом, Вам нужно очень осторожно конструировать выражения с данными и операциями разных типов данных, участвующих в выражениях.

Упражнения

1.Запишите нижеследующие математические выражения на языке ТП, следуя условию, что все переменные в выражениях вещественные (`real`), а потом, что все переменные целые (`integer`):

а) $X + Y^3$ б) $\frac{A + C}{B + D}$

в) $(X + Y)^2$ г) $\frac{A + D}{C + \frac{B}{F + G}}$

д) $\frac{A + B}{C}$ е) $A + \frac{B}{C}$

ж) $A + \frac{B}{C + D}$ з) $\left(\frac{X}{Y}\right)^3$

2.Выполните действия на совместимость типов и проверьте их на компьютере:

а) X:=45; Y:=578; Z:=X+Y. Какими могут быть типы величин X, Y и Z?

б) X:=6546; Y: =5655.665; Z:=X-Y. Какими могут быть типы величин X, Y и Z?

в) X:=65; Y:=5; Z:=X/Y. Какими могут быть типы величин X, Y и Z?

г) X=567687687; Y=5.89; Z=X+Y. Какими могут быть типы величин X, Y и Z?

8.5. Общая структура программ на Турбо Паскале

Вам уже, наверное, стало интересно, как же создавать свою собственную программу на языке ТП? А этот процесс не сложнее, чем писать кому-нибудь письмо. Ведь для этого, сначала, нужно продумать, чем Вы хотите порадовать своего адресата? А уж после этого, Вы должны вспомнить, что письмо начинается с обращения к Вашему адресату. Затем с абзаца необходимо начинать изложение мыслей с использованием существительных, прилагательных, глаголов и т.д., которые нужно располагать по определённым правилам и следует разделять запятыми, точками и другими знаками препинания для точной передачи смысла Вашего послания. Вспомните: «Казнить нельзя помиловать!».

Так вот, прежде, чем писать программу на Турбо Паскале, Вам необходимо запомнить, что, в целом, она состоит из заголовка блока[16] и заканчивается точкой, поэтому кратчайшая программа в языке ТП, которая ничего полезного не делает, выглядит в форме блока так:

```
Begin
End.
```

С лёгкой руки автора языка Си Денниса Ричи стало хорошим тоном представлять описываемый язык программирования программой, которая приветствует окружающий мир, выводя на экран монитора фразу «**Hello, World!**». Так вот, на ТП такая программа выглядит следующим образом:

```
Begin
  Writeln (‘Hello, World!’);
End.
```

После нажатия клавиш **Ctrl+F9** в среде редактора ТП компьютер выполнит программу и мгновенно возвратится назад в среду редактора. Чтобы увидеть результат её работы нужно нажать клавиши **Alt+F5**. Это позволит Вам увидеть на экране монитора Ваш первый успех в программировании. Нажатие любой клавиши возвратит Вас обратно в окно текстового редактора ТП.

Кстати, можно улучшить программу вывода строки текста `‘Hello, World!’`, прибавив в её конце оператор процедуры `Readln` без параметров, что останавливает работу программы в экране результатов в ожидании нажатия клавиши **Enter** для завершения её работы:

```
Begin
  Writeln (‘Hello, World!’);
  Readln;
End.
```

Общая структура программы на языке ТП приведена на рис. 8.15. Как видите, программа начинается с заголовка `Program` и её имени, а основной блок включает ***раздел описаний*** и ***раздел операторов***. Раздел операторов

[16] Блок – смысловая часть программы, завершающаяся точкой. Вся информация, располагаемая после точки последнего зарезервированного слова `end` средой ТП ***ко вниманию не принимается!!!***

представляет собой, так называемый, составной оператор (операторные скобки `Begin ... End`), включающий разделённые точкой с запятой (;) операторы языка ТП, которые описывают алгоритм разработанной Вами программы.

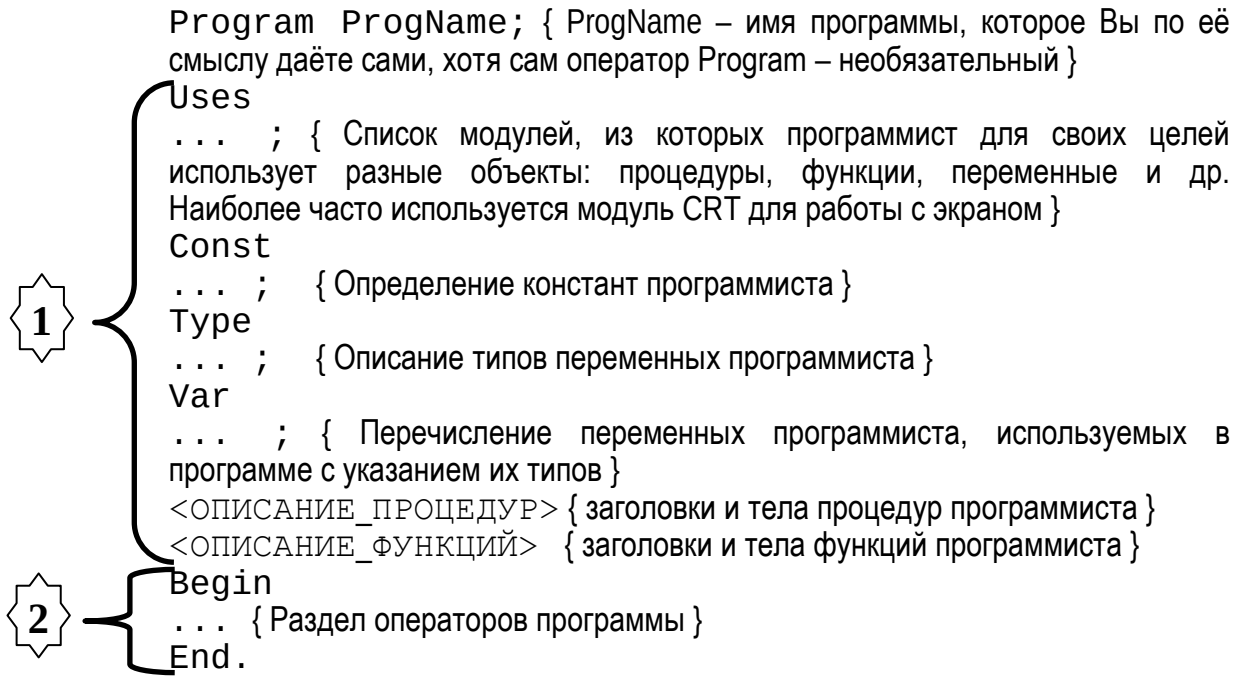


Рисунок 8.15. Общая структура программы на языке ТП
(1 – раздел описаний, 2 – раздел операторов)

Глядя на рис. 8.15 Вы уже немного приуныли от обилия препонов на пути полёта Вашей программистской мысли! Не горюйте! Программирование насчитывает уже немало лет своей истории, которая неумолимо подтверждает тот факт, что ***человек – это существо, которое обязано ошибаться!*** Причём, чем быстрее Вы принимаете решения либо набираете текст, тем больше Вы ошибаетесь и в жизни и в программировании. Поэтому хороший язык программирования и его среда разработки предлагают Вам множество средств для обеспечения, как избежания, так и нахождения Ваших же ошибок при создании программ. И, чем больше подробностей Вам приходится указывать в программе, тем меньше размеры компилятора, тем быстрее он работает, тщательнее проверяет ваши действия на предмет использования недопустимых операций с объектами разных типов, использование разных объектов с одинаковыми именами и так далее (рис. 8.14)… Можно сказать, что чем более Вы внимательны к компилятору, тем более он учтив с Вами!

Возвращаясь к аналогии с естественным языком, можно сказать, что константы и переменные в программе являются существительными языка программирования, типы – прилагательными, а операции и сложные операторы – глаголами. Давайте рассмотрим небольшой пример вычисления формулы:

$$f = 0.5 \bullet x \bullet 0{,}326754892. \qquad (8.1)$$

Составляя программу для вычисления формулы (8.1) согласно рис. 8.15, Вы должны, удерживая в голове общий вид формулы, пройтись по разделу описаний будущей программы, чтобы не пропустить нужного подраздела (рис. 8.16).

```
Program FormulaF;   { Название программы придумываем сами }
Uses Crt;   { Раздел Uses – нужен для подключения модуля Crt, из которого будем }
            { использовать процедуру ClrScr для очистки экрана }
Const
  MyConst = 0.326754892; { Раздел Const – нужен, поскольку сложным констан-}
             { там лучше присваивать имена во избежание ошибок при многократном }
             { наборе с клавиатуры в тексте программы. К тому же, если значение та- }
             { кой константы вдруг изменится, его достаточно изменить только в этом }
             { разделе, а не по всей программе, а 0,5 лучше оставить в покое. }
             { Вещественный тип константы задаёт точка в константе }
{ Раздел Type в этой программе не нужен. Он появится при использовании массивов }
                                                  { с процедурами и функциями }
Var
  f, x : Real; { Раздел Var – нужен, поскольку у нас в формуле (8.1) осталось два }
               { неописанных объекта, которые,  по условию, будут менять }
               { свои значения при каждом вызове программы. По смыслу }
               { задачи их тип – Real («прилагательное»). }
{ Разделы описания процедур и функций отсутствуют }
Begin       { Начался раздел операторов }
  ClrScr;   { Вызов процедуры ClrScr из модуля Crt для очистки экрана при запуске }
  Write('Введите x ='); { Встроенной процедурой Write выдаём на экран }
                        { приглашение ввести значение x с клавиатуры }
  ReadLn(x);  { Встроенной процедурой ReadLn читаем в переменную x, }
              { введённое с клавиатуры требуемое число }
  f := 0.5*x*MyConst; {0.5 – безымянная, неизменяемая константа, }
        { x – переменная, содержащая введённое с клавиатуры требуемое число, }
        { MyConst – именованная неизменяемая константа («существительные»).}
        { «*» – операция умножения, «:=» – операция присваивания («глаголы»). }
  Writeln('Результат f = ', f); { Встроенной процедурой Write выдаём }
                                { на экран поясняющую строку и значение переменной f }
  ReadLn  { Задерживаем экран результатов до нажатия клавиши Enter }
  End.    { точку с запятой после ReadLn перед End можно не ставить }
```

Рис. 8.16. Программа вычисления выражения (8.1)

Упражнения

1. Какова полная структура программы на языке Турбо Паскаль?

2. Какие разделы входят в программу на языке Турбо Паскаль?

3. Можно ли продолжать вводить текст операторной части программы после конца блока (`End.`)?

4. Запишите в виде программ ТП упражнения из предыдущего раздела и выполните их на компьютере.

8.6. Интерфейс программы пользователя. Процедуры ввода-вывода

Когда программа загружена в оперативную память компьютера (в виде EXE-файла) и начинает выполняться, наступает этап взаимодействия трёх сущностей, то есть обмена сообщениями между следующими тремя сторонами:

❶ пользователем программы, которую выполняет компьютер;

❷ разработчиком программы, которой пользуется пользователь;

❸ компьютером, который обеспечивает диалог, запрограммированный разработчиком и собственно работу программы.

При обучении программированию, простейшим случаем является ситуация, когда учащийся сам программирует какой-либо алгоритм, выступая при этом одновременно и программистом, и пользователем. В этом случае, он должен сам прогнозировать ситуации взаимодействия пользователя (то есть самого себя) с компьютером. Последний должен выполнять приглашения на выполнение некоторых действий, а также их и выполнять (рис. 8.17).

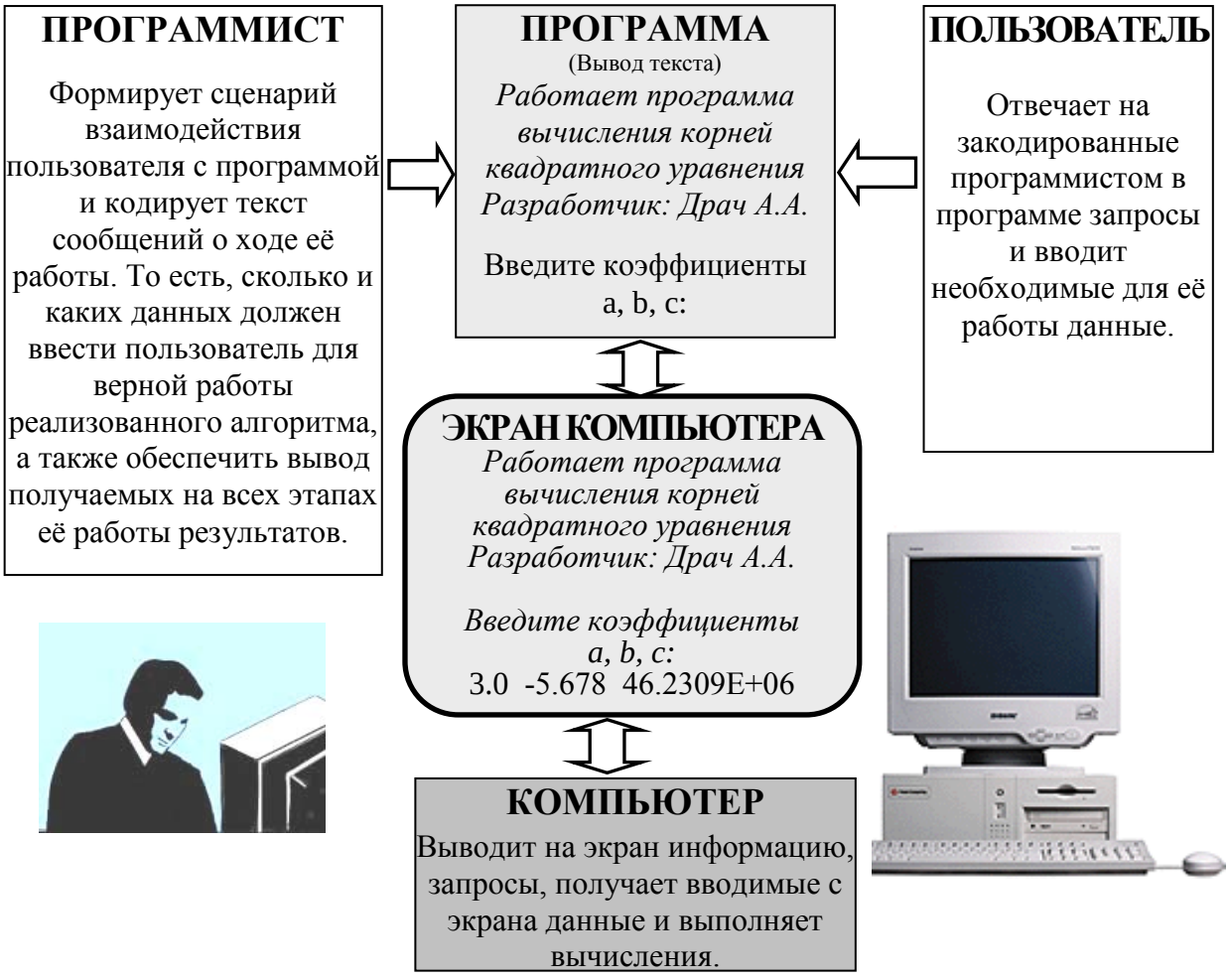


Рис. 8.17. Содержание функций интерфейса пользователя

Простейшая ситуация при выполнении программы состоит в том, что когда пользователю Вашей программы надо выполнить какие-то действия, то ему необходимо знать, что умеет делать Ваша программа и какие данные ей необходимы для работы. Например, по ходу выполнения программы,

необходимо ввести два числа вещественного типа, то есть числа, содержащие точку в своем представлении (к примеру, 2.357 и 42.9). Возникает вопрос: как предупредить пользователя, что компьютер ждет от него именно ввода двух и именно вещественных чисел?

Вот здесь, Вы, как разработчик программы, и должны организовать выдачу сообщения для её будущего пользователя, которое будет ждать в этот момент от него компьютер. В Турбо Паскале для этого существуют операторы вывода с именами `Write` и `WriteLn`, с помощью которых можно выводить на экран строки текста, которые подсказывают пользователю, что именно следует делать далее.

Вообще же, для введения информации в компьютер и вывода ее из него в языке ТП существуют четыре процедуры (все четыре, с переменным количеством входных параметров): `Read, ReadLn, Write` и `WriteLn`.

Процедуры чтения данных с экрана ПК `Read` и `ReadLn` обеспечивают введение данных различных типов: чисел, символов, строк и т.д. Напоминаем, что как только в программе встречается оператор `Read` либо `ReadLn`, компьютер переходит в чёрный экран и в режим ожидания от пользователя ввода требуемого количества данных соответствующего типа. И пока они все не будут введены, работа компьютера не будет продолжена!

Формат (синтаксис) вызова: `Read (x1, x2, x3..., xn);`

Где: `x1, x2... , xn` – переменные тех типов, которые описаны в разделе `Var`, выполняемой программы. При несовпадении между данными, которые описаны в программе и которые в них вводятся, возникает ошибка ввода-вывода! Вызов процедуры `ReadLn` без параметров останавливает действие программы до нажатия клавиши **Enter**. После завершения процесса чтения курсор автоматически переводится на следующую строку экрана. Разница между этими процедурами существует только при чтении из текстового файла.

Примеры программных реализаций ввода данных процедурами *Read, ReadLn:*

```
Var       A, B, C, D, Sum1, Sum2 : real;
  Begin
```

Текст программы (в синем экране)	Ввод данных (на чёрном экране)
`Read(A,B);`	`1.8  3.4  <Enter>`
`Sum1:= A + B;`	
`Read(C,D);`	`2.87  0.7  <Enter>`

{Между числами должно быть не менее одного пробела}

```
  Sum2:= C + D;
  Readkey {встроенная функция ТП, ожидающая нажатия любой клавиши}
```

{Использование процедуры `ReadLn`}

`ReadLn (A,B);`	`1.8  3.4  <Enter>`
`Sum:= A + B;`	
`ReadLn (C,D);`	`2.6  0.7  <Enter>`
`Sum2:= C + D;`	

```
  End.
```

Процедуры вывода данных на экран ПК `Write` и `WriteLn` обеспечивают вывод данных различных типов: чисел, символов, строк и т.д.

Формат (синтаксис) вызова: `Write (x1, x2, x3..., xn);`

Где: `x1, x2... , xn` – переменные тех типов, которые описаны в разделе `Var`, выполняемой программы либо явно заданные строковые константы (`'Результат x = '`) или функции (`sin(0.5*x)`). Различие в их работе состоит в том, что процедура `Write` после вывода оставляет курсор за последним выведенным символом данных, а `WriteLn` – переводит курсор в начало следующей строки. Оператор `WriteLn` без параметров просто переводит курсор на следующую строку экрана. Это следует учитывать при разработке интерфейса программы.

В процедурах `Write` и `WriteLn` имеется возможность ***форматирования*** исходных данных, то есть задания в явном виде выражений возле имен выводимых переменных, которые определяют ширину поля вывода для выводимых значений:

Пример: Вывод целых значений, которые имеют нижеприведенное описание:
```
Var  I : Integer;
```

Значения, содержащиеся в переменной I	Выражение для вывода этого значения	Результат на экране ← левая крайняя позиция поля вывода экрана
134	`Write(I);`	`134`
56789	`Write(I);`	`56789`
287	`Write(I,I,I);`	`287287287`
134	`Write(I:6);`	`  134`
1	`Write(I:10);`	`         1`
312	`Write(I+I:7);`	`  624`

Если для целого числа при выводе все обстоит достаточно просто (то есть на экране необходимо разместить только знак и значения переменной), то для вещественной переменной все значительно сложнее. Это связано с тем, что для вещественных значений существует две разные формы представления:

❶ с фиксированной точкой (к примеру, –34.295);

❷ с плавающей точкой или в экспоненциальной форме (то есть 182600.0 = $1.826*10^5$= +1.826E+05 = 18.26E4 и так далее).

При выводе числа типа `REAL` с фиксированной точкой нужно выводить на экран четыре его элемента:

❶ знак числа (знак “плюс” (+), как правило, не отображается);

❷ целую часть числа;

❸ точку, которая разделяет целую и дробную части числа;

❹ дробную часть числа.

В случае вывода числа типа `REAL` с плавающей точкой для его отображения стандартно отводится 18 позиций, в которые требуется разместить восемь составляющих (рис. 8.18):

❶ пустая позиция (1-й элемент);

❷ знак числа (пустая позиция, если + или -) – занимает одну позицию (2-й элемент);

❸ целое значение мантиссы (3-й элемент);

❹ разделяющая точка (4-й элемент);

❺ дробная часть мантиссы (5-й элемент);

❻ буква E, отмечающая начало значения порядка числа (6-й элемент);

❼ знак порядка: плюс (+) или минус (-) (7-й элемент);

❽ числовое значение порядка (8-й элемент).

Номера позиций в числе →	1	2	3	4	5	6	7	8
Само число →		**–**	**2**	**.**	**815239**	**E**	**+**	**02**

Рис. 8.18. Элементы, представляющие числа с плавающей точкой

Примечание: В компьютерах типа Pentium III и выше для хранения чисел отводится еще больше значащих цифр. А для удобства чтения выходных данных существует возможность управлять количеством знаков, которые выводятся на экран для удобства их восприятия.

Пример: Вывод вещественных значений:

```
Var
      R : real;
```

{В поле шириной 18 символов выводится десятичное значение переменной R в формате с плавающей точкой и с 2-мя пустыми позициями перед числом, из которого вторая – выделяется под знак числа}

Значения, содержащиеся в переменной R	Выражения для вывода этого значения	Результат на экране ← левая крайняя позиция поля вывода экрана
715.432	`Write(R);`	`7.1543200000E+02`
-1.919E+01	`Write(R);`	`-1.9190000000E+01`
567.986	`Write(R/2);`	`2.8399300000E+02`
511.04	`Write(R:15);`	`5.110400000E+02`
-511.04	`Write(R:15);`	`-5.110400000E+02`
46.78	`Write(-R:12);`	`-4.67800E+01`
511.04	`Write(R:8:4);`	`511.0400`
-46.78	`Write(R:7:2);`	`-46.78`
-46.78	`Write(R:9:4);`	`-46.7800`

На рис. 8.19 приведена программа с большим количеством комментариев, которая имеет довольно развитый интерфейс пользователя.

Пример программы на языке Турбо Паскаль, которая реализует вычисления согласно формулам и интерактивно взаимодействует с пользователем.

```
Program Second; {Комментарий: типичная программа на языке Турбо Паскаль}
Uses Crt;              {Подключение модуля Crt}
  const t=2.345;           {числовая константа t }
  const h : integer = 55;{типизированная константа h }

Var                        {оператор описания переменных }
   a,b,c,r,s,Res : Real; {вещественные переменные}
   i,j,k,n : integer; {целые переменные}
   Name : string [20]; {строковая переменная}

Begin                  {Начало раздела операторов программы}
   ClrScr;    {Очищение экрана процедурой ClrScr }
   Write('Введите данные для программы: b,c,r,s = ');
   ReadLn(b,c,r,s); {Вводим с клавиатуры данные для задачи}

{ Реализуем приведенную по правую сторону формулу → }
          {с помощью функций и операций Турбо Паскаля }
                { и заносим результат в переменную a }
                     {оператором присваивания := }
{Возведение в степень реализуем в виде формулы: a^z = exp(z*ln(a)) }
   a:=exp(1/4*ln(b*b*b+c*c*c))/(cos (sqr(r))*
                 sin(exp(2/5*ln(s))));

   WriteLn('Результат a=', a); {Выводим на экран результат}

  {Вычисляем значение t согласно формуле}
   Res:=a*(sqr(b)*b/c);     {и заносим в переменную Res}

   WriteLn('Результат Res = ', Res); {Выводим на экран Res}

   WriteLn('На сегодня хватит?'); Writeln;
   Write('Введите Ваше имя: '); Readln(Name); {Вводим }
                                  {собственное имя в переменную Name }
   WriteLn; {пропуск одной строки на экране}
   WriteLn('Здравствуйте,  ', Name); Writeln;
   WriteLn('  и до свидания  ' , Name);
   ReadLn
End.
```

Раздел описаний

$$a = \frac{\sqrt[4]{(b^3 + c^3)}}{\cos(r)^2 * \sin(s)^{\frac{2}{5}}}$$

$$t = a * \frac{b^3}{c}$$

Конец программы

Рис. 8.19. Пример программы на языке Турбо Паскаль с комментариями

Если Вы запустите программу Second (рис. 8.19) на выполнение нажатием совокупности клавиш **Ctrl+F9**, на экране результатов (после ввода требуемой информации) получим следующие результаты (рис.8.20).

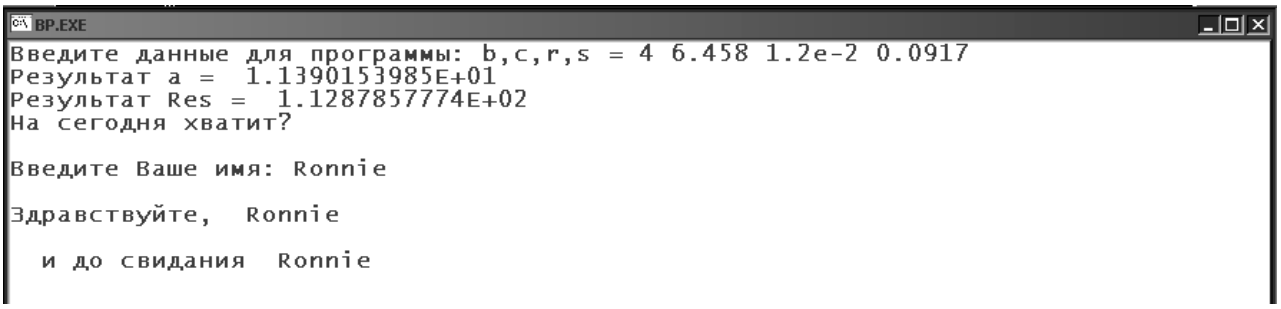


Рис. 8.20. Результат работы программы Second (рис. 8.19)

Обратите внимание на различие между действиями операторов вывода данных Write и WriteLn. Оператор Write, при выводе данных, которые указаны в его списке фактических параметров (Write('Введите данные для программы: b,c,r,s = ')), не переводит курсор на следующую строку. Это означает, что вывод других данных следующим оператором Write будет продолжен в ту же самую строку, но до тех пор, пока не будет исчерпанное место, которое отведено на экране для одной строки. Кстати, длина строки символов, при выводе её на экран, равняется 80-ти символам. Таким образом, как только количество выведенных в одну строку символов превысит 80, все следующие символы начинают выводиться с начала следующей строки. А после выполнения оператора WriteLn('Результат a=', a), курсор после вывода значения a переводится в начало следующей строки.

Это очень хорошо видно на примере вывода 10-ти значений числа с фиксированной точкой 2.5676 с помощью оператора Write(2.5676) и 10-ти значений числа с фиксированной точкой –2.5676 с помощью оператора Write(-2.5676) (рис. 8.21).

Вывод 10-ти значений числа с фиксированной точкой 2.5676 с помощью оператора WriteLn(2.5676) и 10-ти значений числа с фиксированной точкой -2.5676 с помощью оператора WriteLn(-2.5676) приведен на рис. 8.22.

Обратите внимание на то, что при выводе данных операторами Write и WriteLn, Турбо Паскаль выделяет перед каждым вещественным числом пробел и одну позицию под знак числа "–"! Если знака минус ("–") нет, то перед числом выводятся два пробела. Помните, также, что при вводе данных с экрана операторами Read и ReadLn лишние пробелы между вводимыми данными игнорируются. Это означает, что при запросе программы на ввод данных, при наборе с клавиатуры их значения можно разделять не только одним пробелом, но и несколькими.

Теперь, зная особенности работы операторов вывода Write, WriteLn и ввода Read, ReadLn, Вы можете сами создавать удобный интерфейс для работы с Вашими программами.

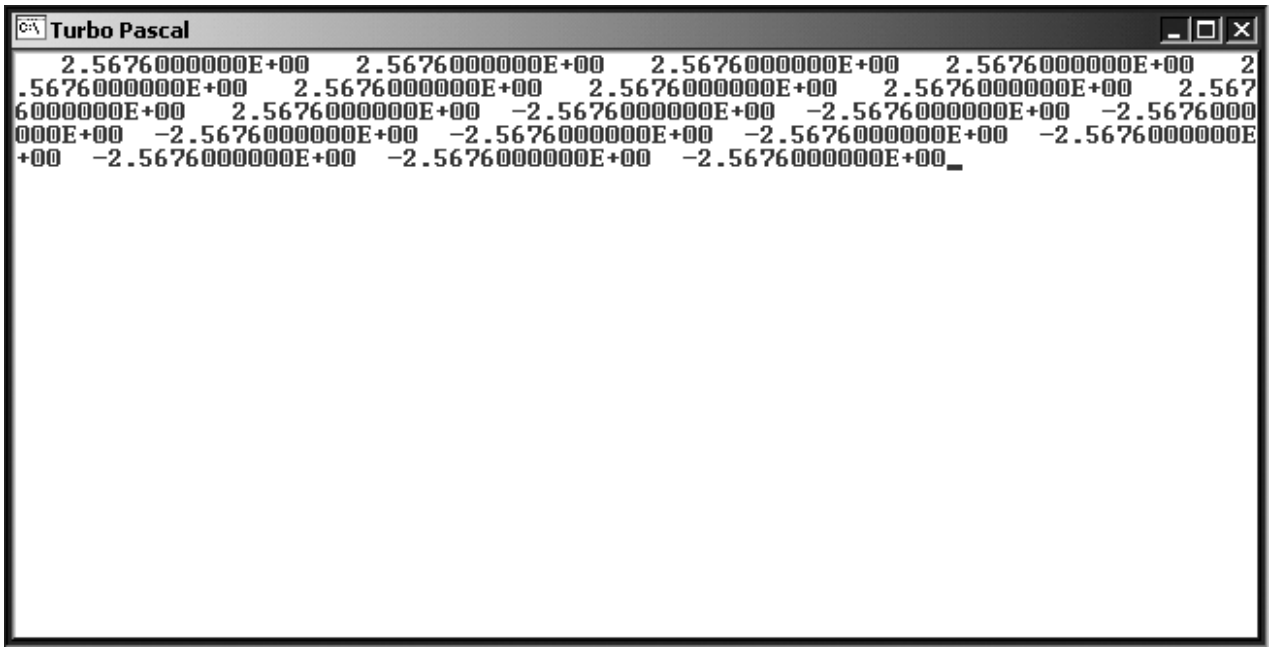


Рис. 8.21. Вывод 20-ти чисел оператором `Write`

Рис. 8.22. Вывод 20-ти чисел оператором `WriteLn`

Упражнения

1. Каким образом можно задавать исходные данные в программе?

2. Как можно вывести информацию на экран из программы?

3. Как можно ввести информацию с экрана в программу?

4. Объясните особенности выполнения операторов `Write` и `Writeln` в программе. В чём их основное отличие?

5. Составьте программу для решения следующей задачи: "Найти площадь прямоугольника со сторонами A и B. Вывести на экран значения площади с точностью до 3-х десятичных знаков в дробной части числа".

6. Составьте программу для вычисления среднего арифметического четырех целых чисел, содержащихся в переменных X, Y, Z и C.

7. Период колебаний маятника длиной L вычисляется по формуле: $t = 2\pi\sqrt{\frac{l}{g}}$, где g – ускорение свободного падения (9.81м/с^2). Найти период колебаний маятника.

8. Сила притяжения F между телами с массами m_1 и m_2, которые находятся на расстоянии r друг от друга, равняется: $F = \gamma \cdot \frac{m_1 \cdot m_2}{r^2}$, где гравитационная постоянная γ=6.673•10^{11}м^3/(кг•с^2). Найти силу притяжения F.

9. Периметр p правильного n-угольника, описанного возле окружности радиусом r, равняется: $p = 2n \cdot r \cdot tg\left(\frac{\pi}{n}\right)$. Найдите периметр r.

10. Энергия E, излучаемая черным телом на волне длины λ при температуре τ, равняется: $E = \frac{2\pi \cdot c \cdot h \cdot \lambda^{-5}}{e^{c \cdot h / \beta \cdot \lambda \cdot \tau^{-1}}}$, где c =2,997924•10 ^{8} – скорость светf; h=6,626•10^{-34} Дж/с – постоянная Планка; β=1,38• 10^{-23} Дж/град – постоянная Больцмана. Найдите энергию E, излучаемую черным телом.

8.7. Выражения, операнды и операции

Выражения в программировании служат для записи действий, которые в математике описываются формулами. В языке Турбо Паскаль, выражения состоят из операций, операндов, встроенных стандартных функций (`cos, sin, exp` и др., см. таблицу 8.20 раздел 8.9) и функций, разработанных самим программистом. Вам следует различать унарные (одноместные) и бинарные (двуместные) операции, которые отличаются количеством операндов, объединяемых операциями (табл. 8.9).

Таблица 8.9.

Примеры выполнения унарных и бинарных операций

Унарные операции		Бинарные операции	
Выражение	Результат	Выражение	Результат
– (+7)	– 7	2 + 6	8
– (–2)	2	(0.5 – 1)*2+10	9
`not False`	`True`	`False or True`	`True`

Последовательность выполнения операций определяется тремя факторами:

❶ приоритетом операций, которые используются;

❷ порядком расположения операций в выражениях;

❸ использованием скобок.

В языке ТП все 22 операции по их приоритетам выполнения делятся на 4 группы (табл.8.10).

Таблица 8.10

Приоритеты операций в Турбо Паскале

Группы приоритетов	Операции	Категория операций
Первый (высший)	+ – not @	Унарные операции
Второй	* / div mod and shl shr	Бинарные операции типа умножения
Третий	+ – or xor	Бинарные операции типа прибавления
Четвертый (низший)	= <> < > <= >= in	Бинарные операции отношений

При вычислении выражений самыми первыми вычисляются выражения в скобках (как самостоятельные операнды), а потом эти результаты уже используется для выполнения операций, которые их соединяют. Операции первого (высшего) приоритета выполняются в первую очередь. Операции

четвертого (низшего) приоритета выполняются в последнюю очередь. Операции равного приоритета (типа *, / либо +, –) выполняются последовательно слева направо. Это означает, что в выражении `A*B/C` значения переменной `A` будет сначала умножено на значение переменной `B`, а потом результат этого действия будет разделен на значение переменной `C`.

По характеру выполняемых операций, их можно разделить на такие группы (табл. 8.11):

Таблица 8.11

Типы операций

№ п/п	Группа операций	Тип операций	
		Унарные	Бинарные
1	Арифметические операции	+, –	+, –, *, /, div, mod
2	Операции отношений	нет	=, <>, <, >, <=, >=
3	Булевские (логические) операции	not	and, or, xor
4	Поразрядные логические и сдвиговые операции	not	and, or, xor, shl, shr
5	Строковые операции	нет	+ (конкатенация)
6	Операции над множествами	+, –	+, –, *, in, <=, >=
7	Операция взятия адреса	@	нет

Приведем примеры выполнения некоторых операций в выражениях с разными типами данных (таблица 8.12).

Таблица 8.12

Примеры вычисления выражений разных типов.

Тип данных в выражениях	Выражение	Последовательность выполнения	
		Выражение	Результат
1	3	4	5
Вещественный	Арифметическое выражение 10.0 – 100.0/(1.0+4.0)*2.0		
		1.0+4.0= 100./5.= 20*2= –(40)= 10–40= Результат	5.0 20.0 40 –40 –30.0 –30.0
Логический	Логическое выражение (4>5) and not (2<=9)		
		4>5= 2<=9= not(true)= false and false = Результат:	False True False False False

Продолжение таблицы 8.12.

1	3	4	5
Целый	Арифметическое выражение 100 div 7 mod 4–(2+6 mod 4)		
		6 mod 4 = 2 + 2 = – (4) = 100 div 7 = 14 mod 4 = 2 – 4 = Результат	2 4 – 4 14 2 –2 –2

При кодировании алгоритма программы Вы как программист должны знать и выполнять некоторые важные правила, чтобы реализованная Вами программа правильно решала поставленную задачу.

1. Два символа операций не должны стоять рядом. Поэтому выражение `A*-B` является бессмысленным, но выражение `A*(-B)` вполне допустимо.

2. Очень важную роль играют в выражениях скобки. От их расположения зависит ход выполнения операций так же, как и в математических записях. Например, выражения `A/B+C` и `A/(B+C)` дадут разные результаты в связи с различием в приоритетах операций, входящих в них.

3. Если последовательность выполнения операций в выражениях не полностью определена скобками, то в первую очередь выполняются возведения в степень, затем все операции умножения и деления, а уж потом операции сложения и вычитания. Поэтому следующие два выражения эквивалентны:

```
A*B+C/B-E*X
(A*B)+(C/B)-(E*X)
```

4. Внутри последовательностей умножений и делений, или сложений и вычитаний, в которых порядок операций не определен скобками, действия выполняются подряд слева направо.

Упражнения

1. Какие операции являются унарными? Приведите примеры.

2. Какие операции являются бинарными? Приведите примеры.

3. Объясните понятие приоритета математических операций при вычислении математических выражений.

4. В каком порядке выполняются математические операции?

5. Каково назначение скобок в математических выражениях?

6. Как записываются математические выражения в Турбо Паскале?

7. Как можно изменить естественный порядок выполнения математических операций в математических выражениях?

8. Объём цилиндра с радиусом основания R и высотой H равняется: $V = \pi \cdot R^2 \cdot h$. Площадь его боковой и полной поверхностей соответственно равны: $S_{б\ ок} = 2 \cdot \pi \cdot R \cdot h, S_{п} = 2 \cdot \pi \cdot R \cdot h + 2 \cdot \pi \cdot R^2$. Найти *V*, $S_б$ и $S_п$.

9. Найти длину окружности, площадь круга и объем шара одного и того же радиуса R. При вычислении использовать формулы: $l = 2 \cdot \pi \cdot R$, $S = \pi \cdot R^2$, $V = \frac{4}{3} \pi \cdot R^3$.

10. Определить скорость резания круга шлифовального станка: $V = d_1 \cdot z \cdot \pi \cdot \frac{d_3}{d_2}$, где d_1 - диаметр шкива двигателя, d_2 - диаметр рабочего вала, d_3 - диаметр инструмента, который режет, *z* - частота обращения.

11. Вычислите общую поверхность и объем конуса, который имеет радиус основания *R* и длину образующей *L*. При вычислении использовать формулы: $S = \pi \cdot R^2 + \pi \cdot R \cdot L$, $V = \frac{1}{3} \pi \cdot R^2 \cdot H$, где H - высота конуса, вычисляемая по формуле: $H = \sqrt{L^2 - R^2}$.

12. Дана окружность радиуса *r*. Найдите площади сектора и сегмента. При вычислении использовать формулы: $S_{сек} = \frac{\pi \cdot r^2 \cdot \alpha}{360^\circ}$, $S_{сег} = \frac{r^2}{2}\left(\frac{\pi \cdot \alpha}{180^\circ} - \sin\alpha\right)$, где α - центральный угол в градусах.

13. Дана гипотенуза и катет прямоугольного треугольника. Найти второй катет и радиусы описанной и вписанной окружностей *R* и *r*. При вычислении использовать формулы: $r = \frac{2 \cdot S}{a + b + c}$, $R = \frac{a \cdot b \cdot c}{4 \cdot S}$, где *a, b, c* - стороны треугольника, *S* - площадь.

14. Вычислить расстояние между двумя точками с координатами (x_1,y_1) и (x_2,y_2). Для вычислений воспользуйтесь формулой: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

15. Найдите периметр и площадь прямоугольного треугольника, если известные длины двух катетов.

16. Дана сторона равностороннего треугольника. Найти его периметр и площадь.

17. Найти площадь кольца и площадь части кольца с центральным углом φ (в градусах). Для вычислений воспользоваться формулами: $S = \pi\left(R^2 - r^2\right)$, $S = \frac{\varphi \cdot \pi}{360}\left(R^2 - r^2\right)$.

8.8. Главные задачи компьютерных вычислений. Простые типы данных. Инициализация данных перед вычислением выражений

Главной задачей, в решении которой компьютеру нет равных, является выполнение всевозможных вычислений по заданным алгоритмам. При решении математических задач, такие вычисления выполняются в соответствии с формулами. Формулы могут быть простыми или сложными (рис. 8.23), но подход к их программной реализации всегда единый.

$$z = \sqrt{\frac{S-R}{D+Y}}, \quad t = (y+6)^5 + 2{,}705 \cdot \log_5\left[\frac{1}{B} + \frac{1}{B^2} - \frac{1}{B^{\frac{3}{4}}} + B^3\right] - \sin^6 y + \left|\sqrt{B} - 3\right|$$

$$y(x) = \iiint \frac{(x^2+1)\,dx}{\sqrt{x^3+7x^2-23}}$$

Рис. 8.23. Пример математических формул различной сложности

В соответствии с правилами языка Турбо Паскаль формируется ***выражение***, корректное с точки зрения его синтаксиса и семантики. Полученный в результате вычисления выражения результат, как правило, нужен для дальнейшего использования, например, чтобы:

❶ записать его на диск;
❷ записать его в базу данных;
❸ вывести на экран компьютера;
❹ вывести на принтер для печати в виде твёрдой копии,

и тому подобное.

Хранение целых и вещественных чисел, символов, строк текста и других типов данных во время работы программы производится в ячейках памяти компьютера. Каждый из этих объектов в терминах языка ТП в Вашей программе выступает либо как ***константа*** либо как ***переменная,*** и имеет следующие характеристики (табл. 8.13):

❶ имя (так называемый идентификатор);
❷ присвоенный Вами тип, который определяет доступные операции;
❸ размер (в байтах), который зависит от присвоенного типа.

Таблица 8.13.
Свойства переменных в выражениях

Имя переменной (идентификатор)	Тип переменной	Значения	Количество байтов, отводимых в памяти
`Flag`	`Boolean` (логический)	False	1
`Disc`	`Integer` (целый)	234	2
`TriTri`	`Real` (вещественный)	3.33333333333	6

Имена констант и переменных в программе позволяют Вам фактически использовать те значения, которые в них расположены. Тогда как для компьютера эти ***имена*** являются ***адресами*** этих объектов в оперативной памяти для использования их при вычислениях.

В языке ТП типы данных разделяются на ***простые*** (базовые) и ***структурные*** (см. Приложение 7). Простые типы являются базовыми, так как на их основе строятся более сложные структурные типы. Данные простого типа, к которым относятся константы, переменные, выражения и функции имеют только одно значение, поэтому они еще носят название ***скалярных***. Скалярный тип определяет множество значений переменных, констант, выражений и функций, которые принадлежат к данному типу, форму представления в компьютере значений этих величин и операции, которые к ним могут быть применены.

Скалярный тип может быть стандартным (базовым) или задаваться программистом с помощью служебного (зарезервированного) слова `type`.

К базовым (стандартным) типам принадлежат:

❶ вещественный (`real`);

❷ целый (целочисленный) (`integer`);

❸ логический (`boolean`);

❹ символьный (`char`).

Поэтому, чтобы Вы могли вычислить результат по какой-либо формуле, нужно в первую очередь сконструировать адекватное ей выражение с объектами соответствующего типа языка ТП. А для сохранения результата нужно заготовить, то есть описать в разделе описаний, соответствующие константы и переменные.

Чаще всего полученный результат помещают в переменную с помощью оператора присваивания (см. диаграмму ниже, рис. 8.24).

:=

Если в программе определено имя `A` для некоторой переменной вещественного типа, то действие данного оператора можно графически представить следующим образом:

A ⬅ **<значение>**

A ⬅ **H/ 0.5**

Рис. 8.24. Замещение новым, вычисленным значением предыдущего в переменной `A`, где <значение> – величина БАЗОВОГО типа `real` либо `integer`. Если <значение> имеет тип `integer`, то компилятор преобразует его к типу переменной `A`.

Тогда, если в переменной H находилась величина 2.0, то деление её на 0.5 во время выполнения программы даст значение 4.0, которое занесётся в переменную A по адресу, соответствующему в памяти месту, связанному с именем A (рис. 8.24).

Вам очень важно понять, что при данной операции новое значение (4.0) ***ЗАМЕЩАЕТ***, то есть ***СТИРАЕТ*** в переменной A её ***ПРЕДЫДУЩЕЕ ЗНАЧЕНИЕ***. Следующим важным моментом является то, что действия с числами и другими значениями выражений (символами, логическими значениями и др.), выполняются на сумматоре[17] . Для описанного выше примера процесс перемещения числовых значений между устройствами ПК можно схематично представить так (рис. 8.25).

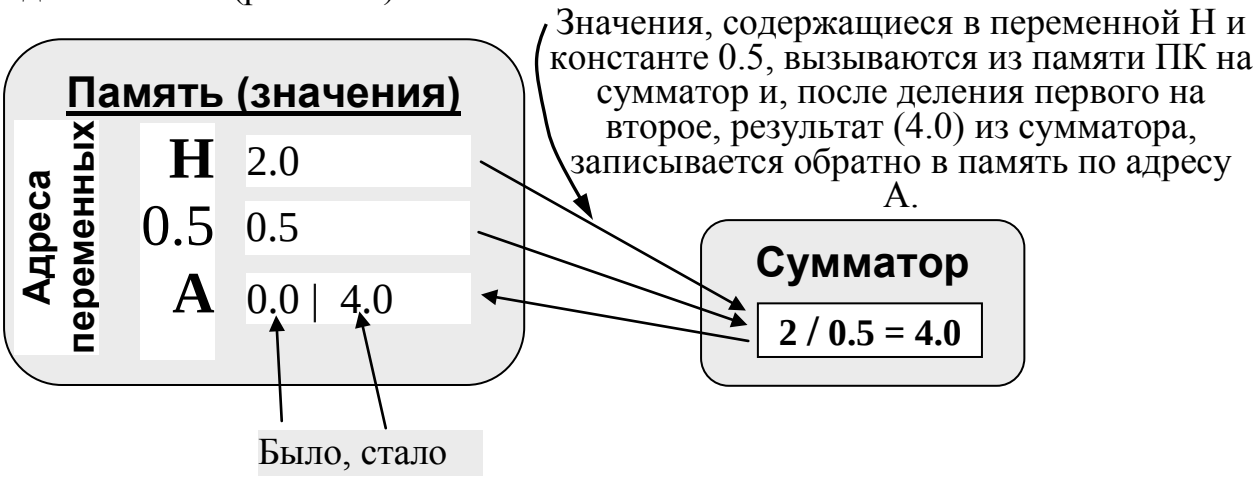


Рис. 8.25. Взаимодействие данных в процессе вычислений

Следует иметь в виду также и то, что в математическом выражении X = 1, знак "=" имеет значения ***"равняется"***, а в выражениях на алгоритмическом языке ТП в выражении X **:=** 1, знак "**:=**" имеет смысл – ***"засылается"***. То есть значение 1 засылается в переменную X и замещает там предыдущее значение, которое сохранялось до текущего момента.

Формулы и более сложные последовательности алгоритмических действий конструируются с помощью трех главных инструментов любых языков программирования:

❶ выражений;

❷ функций;

❸ процедур.

Кстати, программные механизмы функций и процедур, широко применяются в объектно-ориентированных языках программирования при конструировании так называемых методов объектов, а также обработчиков событий.

[17] Сумматор – электронное устройство компьютера, которое является компонентом микропроцессора и предназначено для выполнения арифметических и логических действий с данными (числами). Другое название – арифметико-логическое устройство.

Итак, рассмотрим, к примеру, процесс вычисления объёма конуса, математическая запись которого (8.2), может быть программно реализована следующим образом (табл. 8.14)

$$V = \frac{1}{3}\,\pi\, R^2\, H \qquad (8.2)$$

Таблица 8.14

Программная реализация формулы (8.2) тремя разными способами

№	**Тип реализации**	Форма реализации:	Занесение значения вычисленного объема в соответствующие переменные: *Vkon1, Vkon2, Vkon3*
1	Выражение	`1/3*PI*(R*R)*H`	*Vkon1* := 1/3*PI*(R*R)*H;

2	Функция (описание в программе)	`Function V(PI,R,H :Real):Real;`
		`Begin`
		`  V:= 1/3*PI*(R*R)*H ;`
		`End;`
			Vkon2 := V (PI,R,H);

3	Процедура (описание в программе)	`Procedure V(PI,R,H: Real;`
		`  var W:Real);`
		`Begin`
		`W:= 1/3*PI*(R*R)*H; End;`
			V (PI,R,H, *Vkon3*);

После выполнения всех трех действий, которые представлены в столбце 3 таблицы 8.14 (двух операторов присвоения и одного оператора процедуры) в трех переменных *Vkon1, Vkon2, Vkon3* получим одинаковые результаты. Из этих примеров хорошо видно, что в алгоритмических языках, ***функция является числом (или механизмом получения числа)***, а ***процедура – оператором для получения числа (или групп численных значений).***

Если внимательнее посмотреть на конструкции в таблице (8.14), то очень хорошо видно, что и выражение, и функция, и процедура – это прямое отображение последовательности вычислений, проводимых в соответствии с разработанным программистом алгоритмом. Но компьютер сам без Вас никогда не сможет вычислить эти выражения, так как не знает, какие же именно данные следует подставить в случае очередного запуска программы на место переменных, которые характеризуют высоту конуса `H` и радиус его основания `R`. Число π, к счастью, является встроенной константой языка ТП.

Процесс присваивания начальных значений переменным, принимающим участие в вычислениях, называется ***инициализацией переменных.***

Инициализироваться могут константы и переменные. Константы приобретают значения в виде простых и типизированных констант (табл. 8.15). Типизированная константа отличается от обычной тем, что имеет определяемый в её описании тип (`Real, Integer` и др.) и значения которой может изменяться пользователем в процессе дальнейшей работы программы. Таким образом, к примеру, для вычисления значения объёма конуса по формуле (8.2) в программе высоту и радиус можно задать в виде констант (а не переменных!) и значения инициализировать так, как показано в таблице 8.15.

Переменные, описываемые предварительно в операторе Var (который отводит соответствующим объектам место в памяти), могут инициализироваться или с помощью оператора присваивания (:=), или с помощью стандартных (встроенных) процедур с переменным количеством параметров Read и ReadLn. Эти процедуры позволяют вводить данные в переменные пользователем с экрана компьютера (табл.. 8.16).

Таблица 8.15

Примеры инициализации объектов в программах в виде констант

Название (вид) констант	Реализация в программе
Простые константы	`Const H = 1.34;` `R = 0.56; I = 2;`

Типизированные константы	`Const H : Real = 1.34;`
	`R : Real = 0.56;`
	`I : Integer = 2;`

Таблица 8.16

Инициализация переменных в программах

Операторы инициализации	Реализация инициализации

Присваивания (:=)	`Var`
	`R,H : Real;`
	`I : Integer;`
	`Begin`
	`R:=0.56;`
	`H := 1.37;`
	`I := 2;`
	`End.`

Процедуры Read и ReadLn	`Var`
	`R,H : Real;`
	`I : Integer;`
	`Begin`
	`ReadLn(R, H, I);`
	`End.`

При выполнении инициализации с помощью оператора присваивания компьютер сам занесет указанные Вами данные в соответствующие переменные R,H и I. А при использовании для инициализации в программе процедур Read и Readln, любая из них вызывает остановку процесса выполнения программы в окне результатов (чёрный экран DOS) и ждет от Вас набора на клавиатуре нужных данных (они тут же отображаются на экране) и нажатия клавиши Enter для их ввода (рис. 8.26).

На первом экране DOS (рис. 8.26, б) представлено состояние, когда был произведён первый переход системы ИСР ТП из окна редактора в окно DOS и компьютер ждет ввода пользователем очередного значения , которое должно вводиться в переменную R. Второй экран DOS (рис. 8.26, в, г) демонстрирует состояние, когда значения, введённые в соответствующие переменные оператором процедуры Readln выведены на экран операторами процедур Write и Writeln. В строке (рис. 8.26, в) экрана DOS данные выведены без форматирования, а в строке (рис. 8.26, г) – с использованием спецификаций формата.

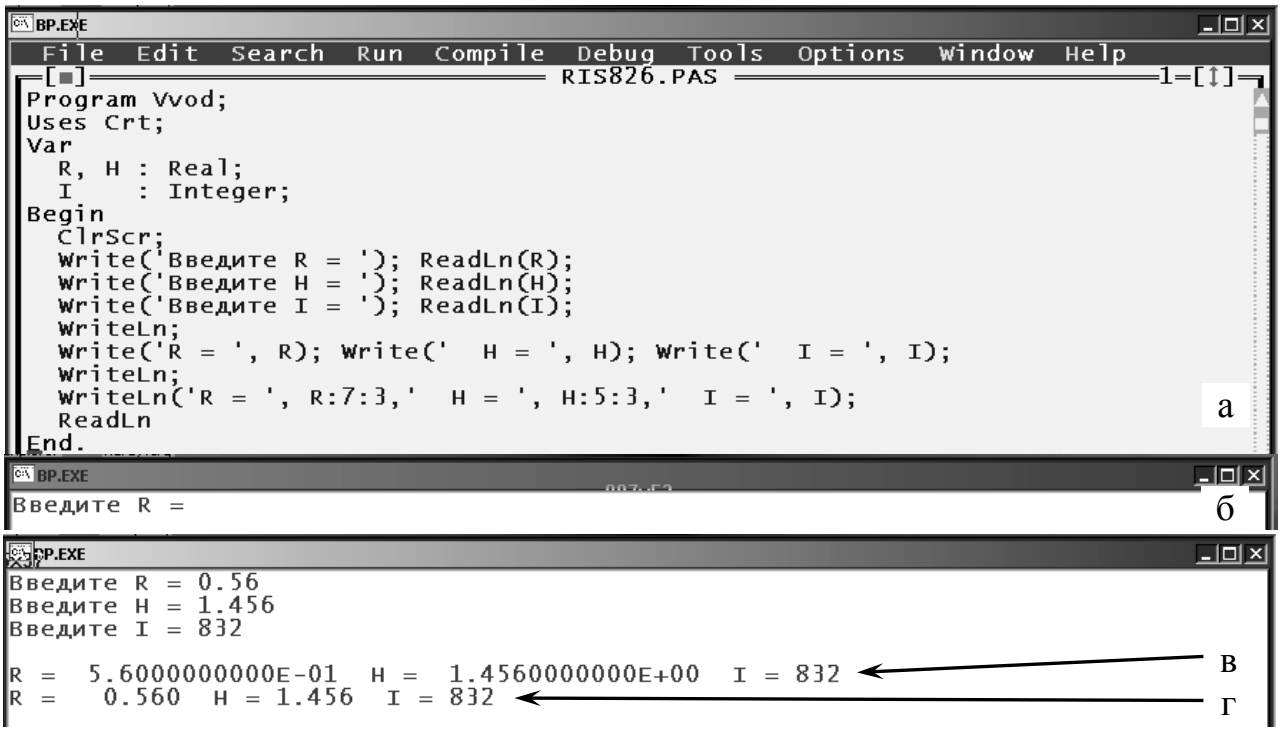


Рис. 8.26. Экран среды ТП (а) и два экрана MS-DOS: в процессе ввода данных в переменные программы процедурой `Readln` (б), вывода процедурой `Write` без форматирования (в) и `Writeln` с форматированием (г).

Упражнения

1. Для чего нужна инициализация данных в программах?

2. Как выполняется в программе операция присваивания?

3. Какие типы данных в ТП являются базовыми? Почему они носят такое название?

4. Какие свойства данных определяет стандартный тип?

5. Какие средства применяются для конструирования выражений в языке Турбо Паскаль?

8.9. Вещественные типы данных (Real). Операции и встроенные функции работы с ними.

Одним из наиболее распространенных типов данных для решения задач с рациональными (дробными) числами является вещественный тип, имеющий в ТП название `Real`. Число, которое описывается таким типом, представляется в памяти компьютера максимально 11-12-ю цифрами его мантиссы[18]. Минимальное (самое малое по модулю) число имеет порядок 10^{-39} (рис.8.266, С), а максимальное – 10^{+38} (рис.2.66, В). А, минимальное число типа `Real` равняется –9.9999999999E+38 (то есть $-9.9999999999*10^{+38}$) (рис8.266, А).

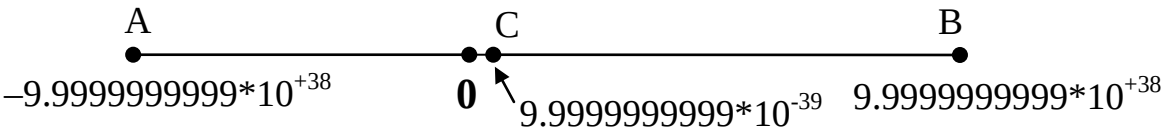


Рис. 8.266 Расположение чисел на числовой оси

При вводе данных в программу для решения Ваших задач такая большая точность практически не нужна. Но она имеет большое значение при проведении вычислений по сложным алгоритмам, где могут возникать ситуации, рассмотренные в разделе 8.4.

Вещественные числа характеризуются тем, что имеют точку в их записи, например, `23.76`. Точка отделяет целую часть (`23`) от дробной части (`76`). Конструируя выражения с числами типа `Real`, Вы можете использовать в них различные объекты (см. табл. 8.17):

❶ константы;
❷ переменные;
❸ значения соответствующих стандартных и пользовательских функций.

Таблица. 8.17

Формы представления и записи вещественных чисел

Вид представления данных	Какими объектами представляются данные	Примеры записи данных (чисел)		
Константы	Числами	2.36	–40.256	1e-7
	Именами	`Beta`	`Gamma`	`Teta`
	Текстом (стрингами)	'61.912E-04'	'0.0000001'	'1.0e+5'
Переменные	Именами	`Frez2`	`NextFunk`	`IOrect`
Функции	Зарезервированными именами функций с нужными типами аргументов	`cos(0.5)`	`exp(Irez2)`	`sqrt(2.0)`
		`ln(0.239)`	`abs(–5.1/0.3)`	`sqr(3E–2)`

[18] Число с плавающей точкой является представлением в компьютере вещественных чисел, обеспечивающим одинаково эффективные средства записи как очень малых, так и очень больших чисел. Число в форме с плавающей точкой в общем случае записывается как $\pm m \times R^{e}$, где m – *мантисса*, R – основание систем счисления, e – *порядок*.

Константы, переменные и функции вещественных типов объединяются в выражения знаками арифметических операций (+, –, *, /). В выражениях вещественного типа могут присутствовать и объекты целого типа, что не влияет на вещественный тип результата.

Вещественные константы отображаются в двух формах: с фиксированной точкой и в экспоненциальной форме (т.е. с плавающей точкой) (табл. 8.18). Экспоненциальная форма представляет число в виде целой и дробной его частей, которые умножаются на основание 10 в целой степени.

Таблица 8.18

Разные формы представления вещественных чисел

Число с фиксированной точкой		Оно же в экспоненциальной форме (другие названия: логарифмическая форма или с плавающей точкой)
1.0	=	1.00000000000E+00
0.0000001	=	1e–7 = 1.0E–7
456.437	=	4.56437e+02 = 456.43E0
–0.0256	=	–2.56E–2

В языке Турбо Паскаль существует ряд ограничений, которые следует учитывать при записи вещественных чисел.

<u>Не допускается</u> (!), чтобы запись числа вещественного типа:

– начиналась с точки (.15 ← неверная запись!);

– заканчивалась точкой (29. ← неверная запись!);

– не имела *значения порядка* после указателя порядка буквы E и знака порядка (6.11E - ← неверная запись!);

– имела бы унарный знак перед указателем *порядка* буквы E:

(-54.99+E -2 ← неверная запись!).

Понятно, что в записи чисел не допускаются также и любые сочетания из вышеуказанных четырёх некорректных элементов записи.

Итак, запись вещественного числа (константы) с фиксированной точкой содержит следующие компоненты (рис. 8.27)

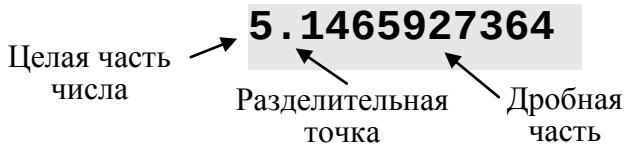


Рис. 8.27 Компоненты представления числа с фиксированной точкой

Для числа в экспоненциальной форме, то есть с плавающей точкой, составные части представления числа таковы (рис. 8.28):

Название "число с плавающей точкой" возникло потому, что эту точку можно передвигать между цифрами в числе, изменяя соответственно показатель степени основания числа, то есть порядок. Таким образом, равнозначны между собою такие формы записи числа:

5.1465927664 E–2 = 514.65927664 E–4 = 0.051465927664

Точность отображения числа вещественного типа зависит от размера области в памяти ПК, которое отводится для его записи (т.е. для его цифр, которые имеют значение) (табл. 8.19).

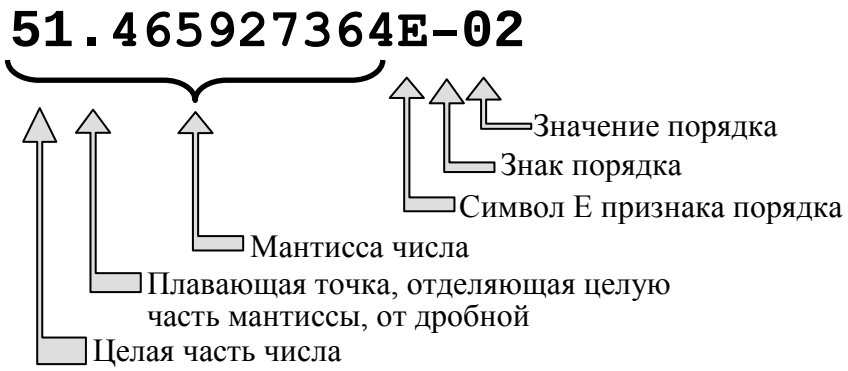


Рис. 8.28. Составные части числа вещественного типа с плавающей точкой (так называемая экспоненциальная или логарифмическая форма записи)

Таблица 8.19

Границы изменений значений вещественных чисел для разных описаний

Длина, байт	Название типа	Количество значащих цифр мантиссы	Диапазон десятичного порядка
4	`Single`	7..8	$10^{-45} \dots 10^{+38}$
6	`Real`	11..12	$10^{-39} \dots 10^{+38}$
8	`Double`	15..16	$10^{-324} \dots 10^{+308}$
10	`Extended`	19..20	$10^{-4932} \dots 10^{+4932}$
8	`Comp`	19..20	$-2^{-63}+1 \dots 2^{+63}-1$

В выражениях вещественного типа можно использовать следующие стандартные арифметические функции, которые встроены в язык ТП (см. таблицу 8.20).

Следует обратить внимание на то, что в таблице 8.20, в скобках у имён функций, указаны типы входных, как правило, вещественных аргументов (`X:real`), а за скобками – тип результата работы этих функций, то есть тип возвращаемых ими значений. Кроме того – во всех тригонометрических функциях ТП, аргументы задаются ***только в радианах***! Кстати, такое же самое правило существует и в приложении MS Excel. Обычно, для перевода из градусной меры в радианы необходимо значение в градусах умножить на выражение $\pi/180°$. Тогда, к примеру, запись вычисления $\sin 60°$ в программе следует записать в виде выражения `sin(60*PI/180)`, где `PI` – встроенная константа ТП, содержащая значение числа π.

Так как список встроенных функций в Турбо Паскале несколько ограничен, то Вы можете самостоятельно конструировать необходимые в работе программ функции. Например, поскольку в ТП нет функций *tg x* и *ctg x*, их можно выразить с помощью функций *sin x* и *cos x*, например так:

$$tg\ x = \frac{\sin x}{\cos x}, \quad ctg\ x = \frac{\cos x}{\sin x} = \frac{1}{tg\ x} \qquad (8.3)$$

А когда в программе Вам потребуется вычислить, допустим, значения этих функций от аргумента 0.6, на языке ТП эти выражения запишутся так:

```
  ..........................
  zntg := sin(0.6)/cos(0.6);
    znctg := cos(0.6)/sin(0/6);
  ..........................
```

То есть в переменные `zntg` и `znctg` будут занесены значения соответствующих тангенса и котангенса от значения константы 0.6.

Таблица 8.20

Встроенные (стандартные) арифметические функции языка ТП

Наименования функций	Возвращаемый результат
`Abs(X:real):real;`	абсолютное значение аргумента ($\|x\|$)
`ArcTan(X:real):real;`	арктангенс аргумента (*arctg x*)
`Cos(X:real):real;`	косинус аргумента (*cos x*)
`Exp(X:real):real;`	показательная функция с основанием *e*, то есть e^X.
`Frac(X:real):real;`	дробная часть аргумента
`Int(X:real):real;`	целая часть аргумента
`Ln(X:real):real;`	натуральный логарифм (*ln x*)
`Pi` (`встроенная константа`)	значение π =3.141592653……..
`Sin(X:real):real;`	синус аргумента (*sin x*)
`Sqr(X:real):real;`	квадрат аргумента (x^2)
`Sqrt(X:real):real;`	корень квадратный от аргумента ($\sqrt{x}$)
`Round(X:real):Integer;`	округление числа *X* до ближайшего целого
`Trunc(X:real): Integer;`	округление числа *X* до целого с отбрасыванием дробной части
`Randomize`	инициализирует случайное значение встроенного генератора случайных чисел
`Random(Range:Word);`	– если параметр `Range` отсутствует функция возвращает вещественное число в диапазоне [0,1]; – если параметр `Range` указан – возвращается целое число в диапазоне от 0 до значения `(Range-1)`

Ниже приведены возможные реализации функций, которые могут понадобиться Вам в работе (табл. 8.21). Понятно, что в программах их должны конструировать Вы сами.

Таблица 8.21

Дополнительные функции, которые Вы можете использовать в программе

Алгоритм реализации функции	Возвращаемое значение
Tan(X:real)=sin(x)/cos(x)	*tg x*
Ctg(X:real)=cos(x)/sin(x)	*ctg x*
Arcsin(X:real)=Arctan(x/sqrt(1-x*x))	арксинус *x* (для ($\|x\|$<1)

Продолжение таблицы 8.21

Алгоритм реализации функции	Возвращаемое значение
Arccos(X:real)=Pi/2-Arctan(x/sqrt(1-x*x))	арккосинус *x*
Power(X:real)=exp(x*Ln(a))	a^x (вещественная степень числа a)
Log(X:real)=Ln(x)/Ln(a)	логарифм значения *x* с любым основанием a, т.е. $ln_a x$ ($a>0$, $x>0$)
ArcCot(X:real)=Pi/2-Arctan(x)	арккотангенс *x*
ArcCsc(X:real)=Arctan(1/sqrt(x*x-1))+(Sgn(x)-1)*(Pi/2)	арккосеканс *x* (Abs(*x*)>=1)
Sgn(X:real)={ 1, если x>0; 0, если x=0; -1, если x<0}	функция знака
ArcSec(X:real)=ArcTan(Sqrt(x*x-1)+(Sgn(x)-1)*(Pi/2)	арксеканс *x* (Abs(*x*)>=1)
Cosh(X:real)=(exp(x)+exp(-x))/2	гиперболический косинус *x*
Csc(X:real)=1/sin(x)	косеканс *x* (*x* не равняется 0)
Sec(X:real)=1/cos(x)	секанс *x* (*x* не равняется π/2)

В общем виде, взаимодействие переменных, констант и функций в выражении вещественного типа может быть представлено так (рис. 8. 29).

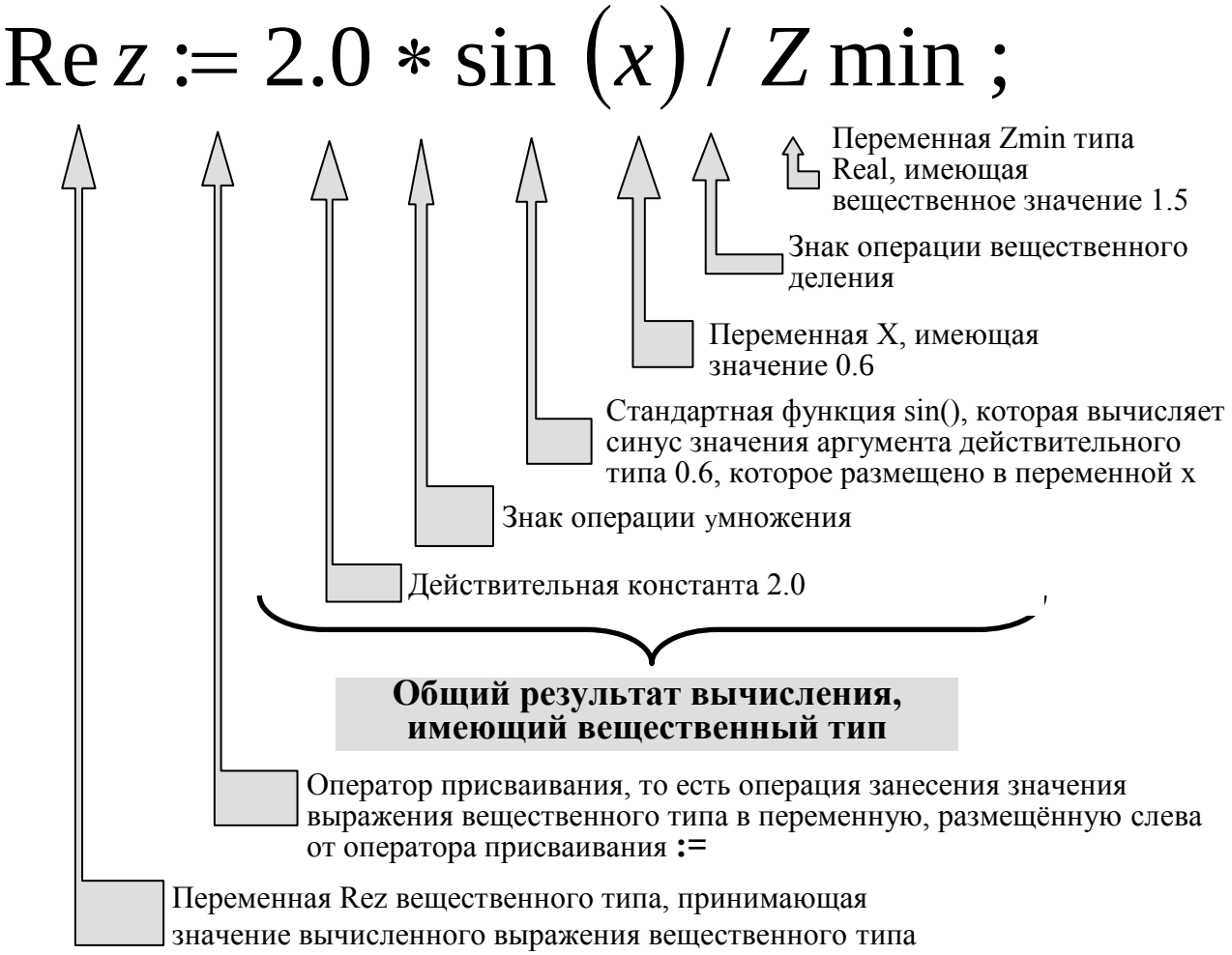


Рис. 8.29. Соответствие типов данных, констант, функций и операций в выражениях обработки данных вещественного типа (`Real`)

Отметим ещё несколько особенностей работы встроенных функций, приведенных в таблице 8.20.

Так как функция `Frac()` возвращает дробную часть вещественного аргумента, а `Int()` – целую его часть, то имеют место следующие преобразования вещественных чисел (знак → обозначает возвращаемый функциями результат).

`Frac(3.141)`→ `0.141; Int(3.141)` → `3.0;`

`Frac(3.141) + Int(3.141)`→ `3.141`

Вычисление показательной функции с основанием e производится с помощью функции `Exp()`. При этом, к примеру:

$e^{1.7}$ → `Exp(1.7)` → `5.4739…;` $e^{-1.7}$ → `Exp(–1.7)` → `0.1827…`

При сложных взаимодействиях разных типов данных часто возникает необходимость преобразовывать вещественные числа в их целые эквиваленты или же производить их округление. Для этих целей существуют функции `Trunc()` и `Round()`. Первая из них `Trunc()`, (*truncate–обрезать*), «отсекает» дробную часть вещественного числа от целой и возвращает целочисленное (т.е. без точки!) значение:

`Trunc(5.67)` → `5;  Trunc(8.0)` → `8;  Trunc(–0.23)` → `0`

Функция `Round()` производит округление вещественного числа до ближайшего целого:

`Round(3.2)` → `3;  Round(3.5)` → `4;  Round(5.0)` → `5`

Достаточно часто в арифметических выражениях требуется вычислять значения функций вида: a^x, $log_a x$ и $\sqrt[a]{x}$. Их математические реализации, запись на языке ТП и последующее вычисление можно представить следующим образом.

$a^x = e^{x \cdot \ln a}$ → `exp(x*ln(a));` $2.8^{3.5}$ → `exp(3.5*ln(2.8)).`

$log_a x = \frac{\ln x}{\ln a}$ → `ln(x)/ln(a);` $\log_5 17.5$ → `ln(17.5)/ln(5).`

$\sqrt[a]{x} = x^{\frac{1}{a}}$ → `exp(1/a*ln(x));` $\sqrt[3]{5.4}$ → `exp(1/3*ln(5.4)).`

Таким образом, реализуя разнообразные алгоритмы, можно записать математические выражения практически любой сложности.

Упражнения

1. Найти внутренний угол α и сумму внутренних углов правильного выпуклого n-угольника. При вычислении использовать формулы: $\alpha = \pi\frac{n-2}{n}$, $s = \pi(n-2)$.

2. Найти объем и площадь поверхности прямого параллелепипеда с сторонами a, b и c.

3. Найти среднюю линию и площадь трапеции, если известные ее основание и высота.

4. Даны координаты трех вершин треугольника A(x_1,y_1), B(x_2,y_2) и C(x_3,y_3). Найти середины его сторон. При вычислении использовать формулы: $x = \frac{(x_1 + x_2)}{2}$, $y = \frac{(y_1 + y_2)}{2}$, где M(x,y) - середина отрезка AB, заданного точками A(x_1,y_1) и B(x_2,y_2).

5. Даны координаты трех вершин треугольника A(x_1,y_1), B(x_2,y_2) и C(x_3,y_3). Вычислить периметр треугольника. Для вычислений воспользуйтесь формулой расстояния между двумя точками A(x_1,y_1) и B(x_2,y_2): $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

6. На плоскости задана прямая уравнением Ax+By+C=0 и точка M с координатами (x_1,y_1). Найти расстояние d от точки до прямой: $d = \frac{A \cdot x_1 + B \cdot x_2 + C}{\sqrt{A^2 + B^2}}$.

7. Даны два вектора $\vec{a}$ (x_1,y_1)и $\vec{b}$ (x_2,y_2) и угол φ между ними (в градусах). Найти скалярное произведение векторов по формуле: $\left(\vec{a}, \vec{b}\right) = \left|\vec{a}\right| \cdot \left|\vec{b}\right| \cos\varphi$.

8. Даны два вектора $\vec{a}$ (x_1,y_1) и $\vec{b}$ (x_2,y_2). Найти угол φ между ними. При вычислении использовать формулы: $\left(\vec{a}, \vec{b}\right) = \left|\vec{a}\right| \cdot \left|\vec{b}\right| \cos\varphi$, $\left(\vec{a}, \vec{b}\right) = x_1 \cdot x_2 + y_1 \cdot y_2$, $|\vec{a}| = \sqrt{x_1^2 + y_1^2}$.

9. На плоскости заданы две прямые линии: $y=k_1x+b_1$ и $y=k_2x+b_2$. Найти угол φ между прямыми, воспользовавшись формулой: $tg\varphi = \frac{k_2 - k_1}{1 + k_1 k_2}$.

10. Вычислить углы треугольника, стороны которого заданы уравнениями прямых: $y=k_1x+b_1$, $y=k_2x+b_2$ и $y=k_3x+b_3$. Для вычислений воспользоваться формулой: $tg\varphi = \frac{k_2 - k_1}{1 + k_1 k_2}$, где k_1 и k_2 - коэффициенты прямых, заданных уравнениями $y=k_1x+b_1$ и $y=k_2x+b_2$, а φ - угол между ними.

11. Написать программу для вычисления площади боковой поверхности $S_{\text{бок}} = 2\pi rH$ и объема $V = \pi r^2 H$ цилиндра по заданным радиусу основания r и высоте H. Ответ вывести дважды: в стандартном виде и с заданной шириной поля вывода.

8.10. Целочисленные типы данных (Integer). Операции и встроенные функции работы с ними.

Наравне с вещественными типами данных в программах ТП используются целочисленные типы данных, которые в ТП объявляются служебным словом `Integer`. Целые числа, в отличие от вещественных, записываются без точки. Например, целое число тридцать четыре запишется так: 34.

Целые числа отнесены в языке Турбо Паскаль к ***порядковым типам*** данных (см. Приложение 7). К порядковым типам также относятся и некоторые другие типы (которые рассматриваются далее):

❶ логический;
❷ символьный;
❸ перечислимый;
❹ тип-диапазон.

К значениям каждого из этих типов может применяться функция `ord (x)`, в результате чего могут быть получены следующие значения:

❶ для целого (значения целого, то есть `ord (9)` → 9);

❷ для логического (0 или 1, то есть `ord(false)` → 0, а значения `ord(true)` → 1);

❸ для символьного (0–255, то есть, к примеру, `ord ('M')` → 77, см. таблицу ASCII кодов в Приложении 6);

❹ для перечислимого (значения перечислимого типа);

❺ для типа-диапазона (значения базового типа).

В арифметических выражениях целые числа, также как и вещественные, используются в виде:

❶ констант;
❷ переменных;
❸ значений соответствующих стандартных функций (см. табл. 8.22).

Следует иметь в виду, что некоторые функции (`abs(x)`, `sqr(x)`, `random(x)`) при использовании с целыми аргументами дают целочисленные результаты, а с вещественными – вещественные. Например:

```
Var
r, t : real;
i, j : integer;
begin
 ..............
  i := sqr (3); { i приобретает целое значение 9}
  r := sqr (3.0); { r приобретает (вещественное!!!) значение 9.0 }
  j := i + r; {Компилятор не пропустит это выражение и выдаст
                диагностику «несовпадение типов»}
  j := i + Round(r); {А если с помощью функции Round округлить значение r
до ближайшего целого, то компилятор такую конструкцию пропустит}
    t := i + r; {Так тоже можно !!!}
...............
```

Вот почему необходимо внимательно следить за сооответствием типов в выражениях ТП (табл. 8.22).

Таблица. 8.22

Представления целочисленных данных

Вид представления данных	Представляются	Примеры записи		
Константы	Числами	`1`	`–256`	`10000`
	Именами	`One`	`DoThat`	`Next`
	Текстом (стрингами)	'64'	'–127'	'10000'
Переменные	Именами	`Irez2`	`NextFunk`	`IOrect`
Функции	Зарезервированными именами функций с нужными типами аргументов	`pred` (5)	`succ` (Irez2)	`sqr`(4)
		`trunc` (0.239)	`abs`(–75)	`ord` (32)

Константы, переменные и функции целых типов объединяются в выражения знаками арифметических операций (+, –, *, div (целочисленное деление), mod (остаток от целочисленного деления)). В выражениях, в первую очередь выполняются операции в скобках, потом операции типа умножения (*, div, mod), а потом сложения (+, –). Операции (*, div, mod) выполняются последовательно (табл. 8.23).

Таблица. 8.23

Выполнение вычислений в целочисленных выражениях

Тип данных в выражения	Выражение	Последовательность выполнения	
		Частичное выражение	Результат
Целый	–(3+6 mod 8)+125 div 5 mod 4*2		
		6 mod 8 =	0
		3 + 0 =	3
		– (3) =	– 3
		125 div 5 =	25
		25 mod 4 =	1
		1 * 2 =	2
		–3 + 2 =	–1
		Результат	–1

Количество цифр в отображении числа целого типа зависит от места в памяти ПК, которое отводится для его представления. Так, если Вы желаете отобразить целое число с количеством цифр, равным шести, то Вам нужно избрать тип `Longint`, так как другие типы не вместят такое количество значащих цифр (см. табл. 8.24).

Таблица. 8.24

Типы целочисленных данных и диапазоны их значений

Длина, байт	Название типа	Диапазон значений чисел
1	`Byte`	0 ... 255
1	`ShortInt`	–128 ... +127
2	`Word`	0 ... 65535
2	`Integer`	–32768 … +32767
4	`Longint`	–2 147 483 648 … +2 147 483 648

Заметьте, что в переменной типа `Integer` Вы уже не сможете уместить величину 40 000 – компилятор выдаст диагностику «`Constant out of range`" («Константа за пределами допустимых значений»). Зато оператор `ReadLn` позволит Вам ввести число 40 000 и Вы будете несказанно удивлены, выведя содержимое переменной с этим числом и получив -25536!

В выражениях целого типа Вы можете использовать следующие встроенные арифметические процедуры и функции (табл. 8.25). Обратите внимание на то, что в скобках указанные типы аргументов, а за скобками – тип результата.

Таблица 8.25

Встроенные функции для работы с целочисленными данными

Наименования функции или процедуры	Возвращаемый результат
`Abs(X:integer): integer;`	абсолютное значение аргумента *X*
`Sqr(X:integer): integer;`	вычисляет квадрат аргумента *X*
`Round(X:real):integer;`	округление вещественного числа *X* до ближайшего целого
`Trunc(X:real):integer;`	округление вещественного числа *X* до целого с отбрасыванием дробной части
`Randomize`	инициализирует начальное значение встроенного генератора случайных чисел
`Random(Range:Word);`	если указан параметр `Range` – возвращает (вычисляет) целое число в диапазоне от 0 до значения (`Range` -1)
`Ord (`значения перечислимого типа и в, том числе, целого`):integer;`	возвращает номер элемента в последовательности значений аргумента
`Pred(`значения перечислимого типа и, в том числе, целого`):integer;`	возвращает номер предыдущего элемента в последовательности
`Succ(`значения перечислимого типа и, в том числе, целого`):integer;`	возвращает номер следующего элемента в последовательности

`Inc(X [,k]); – процедура`
`Inc(X);`
`Inc(X,k);`
	если присутствует только аргумент *X*, то отнимает от него единицу. Если присутствуют *X* и *k*, отнимает от *X* значения *k*.

`Dec(X [,k]); -процедура`
`Dec(X);`
`Dec(X,k);`
	если присутствует только аргумент *X*, то прибавляет к нему единицу. Если присутствуют *X* и *k*, прибавляет к *X* значения *k*.

Используя функции Round и Trunc Вы можете «помогать» компилятору, обеспечивая необходимый тип операндов в выражениях (см. рис. 8.14):

вместо `j := 5 / 2;` записать `j := Round(5/2);` { компилятор пропустит }

вместо `i := 5.3 div 2.6;` записать `i := Round(5.3) div Round(2.6);` { компилятор пропустит также}

Для целых чисел и других порядковых типов справедливы такие соотношения:

`Ord(x)–1 = Ord(Pred(x)), Ord(x)+1 = Ord(Succ(x)).`

Примеры:

```
Var
j, i, k, p, n, m : integer;
Begin
j := 23; i := 5;
.........................................
k := pred(23); {k приобретает значения 22}
 p := succ(k) + ord(и); {p приобретает значения 23+5=28}
  dec(j,и); {j приобретает значения 23-5=18}
   ink(k); {k приобретает значения 22+1=23}
    m := sqr(2)div pred(и); {m приобретает значения 4/4=1}
.........................................
end.
```

Когда Вы начинаете решать задачу, то должны четко представлять, какие типы данных будете использовать, так как от этого зависят:

❶ значения переменных и их типы, которые выбираются и принимают участие в вычислениях (integer, byte, word, real, extended, boolean и др.);

❷ допустимые операции над переменными в выражениях с данными этих типов (+, –, *, /, div, mod, and, or и др.);

❸ типы конечных значений выражений с выбранными данными;

❹ результаты вычислений, которые возвращают встроенные функции. Так как некоторые из них возвращают результаты, которые зависят от типа аргументов.

Вам следует запомнить:

❶ Недопустимо выбирать тип integer для переменных, используемых для задания аргументов тригонометрических функций *sin x* и *cos x*.

❷ Недопустимо пересылать с помощью оператора присваивания данные одного типа в переменную другого типа. Например, невозможно заслать логическое значение True в целую или вещественную переменную Rez.

❸ Недопустимо в выражениях с переменными одного типа, применять операции и функции, которые относятся к другим типам.

Например, в целую переменную Irez невозможно переслать результат вещественной операции (/):

```
Irez := 2 / cos(0.5) ;
```

❹ Недопустимо смешивать в выражениях данные (константы и переменные) разных типов.

Например, невозможно делить целочисленную константу на вещественную константу с помощью оператора целочисленного деления:

```
Irez  := 40 div 4.0 ;
```

Но возможно переслать в логическую переменную результат операции сравнения, так как эта величина является логическим значением.

```
Brez  :=  (5 >= 2) ;
```

Упражнения

1. Какие числа носят название целых?

2. Какие существуют целочисленные встроенные функции в языке ТП?

3. Почему нельзя заносить (присваивать) значения типа `real` в переменные типа `integer`?

4. Выполните следующие действия с целочисленными данными. Проверьте свои вычисления на компьютере. Значения входных данных введите сами.

а) `A div Y;`

б) `A mod A;`

в) `(A+ Y) div (A - Y);`

г) `(A + Y) mod (Y);`

д) `A div 10;`

е) `A mod 10;`

же) `A div 1000;`

з) `A mod 1000;`

и) `A div 100`

к) `A mod 100;`

л) `A mod 2;`

м) `(A mod 10)*((A mod 10) mod 2).`

8.11. Логические типы данных (Boolean). Операции и встроенные функции работы с ними. Конструирование логических выражений для формирования логики работы программ на основе пяти уровней абстракции

При конструировании логики выполнения различных ветвей программы в зависимости от требуемых условий, на первый план выходит умение программиста использовать логический тип данных, имеющий в языке ТП название `Boolean`. Он характеризуется только двумя конкретными значениями:

0 (`False` – ложь);
1 (`True` – истина).

Каждое из них занимает в памяти ПК только один байт. Логические величины рассматриваются компилятором в программах в таком контексте:

❶ в логических выражениях, как значения `True` или `False`;

❷ в целочисленных выражениях и в результате выполнения встроенных функций работы с порядковыми типами (`ord, succ, pred`), как значения 0 или 1.

В выражениях языка ТП логические данные используются в виде:

❶ констант;
❷ переменных;
❸ значений соответствующих встроенных функций (см. табл. 8.26).

Таблица 8.26

Встроенные функции ТП для работы с логическими переменными и значениями

Наименование функции, тип	Выполняемые действия
`Ord` (значение порядкового типа и, в том числе, целого) : `integer;`	возвращает номер элемента в последовательности значений аргумента `Ord(False) = 0, Ord(True) = 1`
`Pred` (значение порядкового типа и, в том числе, логического) : `integer;`	возвращает номер предыдущего элемента в последовательности: `Pred (True ) = False`
`Succ` (значение порядкового типа и, в том числе, логического) : `integer;`	возвращает номер следующего элемента в последовательности: `Succ (False) = True`

`Odd` (значение целого типа) : `boolean;`	возвращает значение `True`, если значение целого нечётное, и значение `False`, если оно чётное:
	`Odd (-2) = False, Odd (-1) = True,`
	`Odd (0) = False,`
	`Odd (1) = True, Odd (2) = False,`
	`Odd (3) = True, Odd (4) = False` и так далее.

Обратите внимание на то, что ***логические значения*** в программе Вы можете ***получать*** в результате сравнения значений ***разных*** объектов.

Например, число 10 меньше 15-ти или нет, небо синее или нет; дождь идёт или нет; дождь сильный или нет и так далее. В математике такие заключения именуются логическими высказываниями. Основным свойством логического высказывания является то, что о нём всегда можно сказать истинно оно или ложно.

Если работа с числами для компьютера является обыденной, то такие общие характеристики, как «синий», «сильный», «вкусный» и т.д. ему неизвестны. В этих случаях для ПК, Вы должны, в соответствии с контекстом решаемой задачи, выбрать необходимые типы данных, а затем уже из них формировать логические выражения.

Начнём с простейшей задачи. Например, Вы знаете, что не существует логарифма отрицательного числа. Поэтому, перед вычислением логарифма в Вашей программе, при необходимости сделать её универсальной, следует проверить, в каком диапазоне находится введённая пользователем величина. Для этого следует сравнить входное числовое значение с нулевым (значением). И если это значение меньше нуля, вычисление логарифма в программе не производится, о чём выдаётся соответствующее сообщение. Это не единственно возможная ситуация, которая может возникнуть при решении Вами прикладных задач. Поэтому в языке ТП существуют специальные операции сравнения (отношения) (табл. 8.27).

Таблица 8.27

Операции сравнения (отношений) языка ТП

Математическое обозначение операций сравнения	=	≠	<	>	≤	≥
Обозначения этих операций в языке Турбо Паскаль	=	<>	<	>	<=	>=

Операции сравнения объединяют сравниваемые объекты (константы, переменные или выражения) в ***простые или сложные логические выражения. Но при этом одна переменная (константа, выражение) может объединяться только одной операцией сравнения и только с одной переменной (константой, выражением)! То есть, в программе нельзя записать такое выражение:*** **`0<x<9!`**. Однако, верными простыми логическими выражениями с точки зрения конструкций языка ТП могут быть такие:

`5 < 1` {Всегда имеет значение `False`}

`x >= 4` {При значениях переменной x ≥ 4 это выражение имеет значение True}

`a <= 'f'` {При описании переменной *a* символьным типом (`var a : Char;`) и присваивании переменной *a* значения символа d (`a:='d';`), выражение `a` ≤ `'f'` будет иметь значение `True`}

`(x*x – b) <> 0` {Имеет значение `True`, если значение выражения (x^2-b) ≠ 0}

Сложное логическое выражение (вида **`0<x<9`**) Вы можете получить, объединяя простые логические выражения с помощью ***логических операций*** (табл. 8.28).

Таблица 8.28

Обозначения логических операций

№ п/п	Название операции	Математическое обозначение	Обозначение в языке ТП
1.	Отрицание	¬	**NOT**
2.	Логическое умножение, И	∧	**AND**
3.	Логическое сложение, ИЛИ	∨	**OR**
4.	Исключающее ИЛИ	⊕	**XOR**

Логическими операциями (`NOT, AND, OR` и `XOR`) называются операции, которые могут выполняться с логическими константами, переменными, выражениями и значениями (`True` или `False`).

Результаты выполнения логических операций с двумя операндами `A` и `B`, имеющих значения `True` и `False` в разных сочетаниях, приведены в таблице 8.29.

Таблица 8.29

Результаты выполнения логических операций NOT, AND, OR и XOR

Логические аргументы, принимающие разные значения True и False		Результат выполнения соответствующих логических операций с разными комбинациями значений аргументов A и B			
A	**B**	**NOT A**	**A AND B**	**A OR B**	**A XOR B**
`False`	`False`	`True`	`False`	`False`	`False`
`False`	`True`	`True`	`False`	`True`	`True`
`True`	`False`	`False`	`False`	`True`	`True`
`True`	`True`	`False`	`True`	`True`	`False`

В побитовой нотации (то есть при взаимодействии отдельных битов), таблица 8.29 принимает следующий вид (табл. 8.30).

Таблица 8.30

Побитовое выполнение логических операций NOT, AND, OR и XOR

Значения аргументов A и B		Результаты выполнения логических операций с аргументами A и B			
A	**B**	**NOT A**	**A AND B**	**A OR B**	**A XOR B**
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

При выполнении логических операций наивысший приоритет имеет унарная операция отрицания (`NOT`). Затем идёт операция логического умножения (`AND`), а самый низкий приоритет имеют операции `OR` и `XOR`.

При вычислении сложных логических выражений, общая последовательность выполнения различных операций, с учётом их приоритета, принята следующей:

❶ первыми выполняются арифметические операции (*, /, div, mod, +, –) и, возможно, некоторые другие;

❷ далее выполняются операции сравнения (=, <>, >, <, >=, <=), дающие логические значения `True` и `False`;

❸ и затем выполняются логические операции: `NOT, AND, OR` и `XOR`.

Первое, что Вы должны усвоить для конструирования верных логических выражений, это то, что ***выполнение допустимых операций для каждого типа данных даёт результат того же типа***. А выполнение операций сравнения для значений (переменных, выражений) этих же типов даёт уже ***логические значения*** (табл. 8.31).

Таблица 8.31

Типы результатов выполнения разных операций с некоторыми типами данных

Тип взаимодействующих данных	Выражение и результат	Тип результата	Сравнение тех же данных	Тип результата логический!
Вещественный	3.1 + 6.91 = 10.1	Вещественный	3.1 < 6.91	`True`
Целый	8 div 4 = 2	Целый	8 <= 4	`False`
Стринговый (строковый)	'con' + 'tent' ='content'	Стринговый	'con' > 'tent'	`False`

Примечание: Недопустимо сравнивать между собой данные разных типов!!!

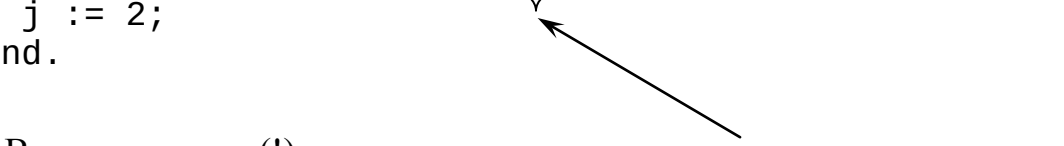
Итак, если в программе, за основу абстрактного представления взаимодействия различных сущностей, при выполнении операций их сравнения, принять ***логическое значение*** (которое существует только в компьютере!), то первый уровень абстракции языковых средств Турбо Паскаля содержится в сравнении двух объектов (значений констант, переменных или функций). То есть, появление логических значений начинается с выполнения атомарных ***логических действий***. Они дают ***логические значения*** (`True` и `False`), получаемые в результате выполнения операций сравнения (>, < и т.д.). С другой стороны, эти логические значения могут объединяться в выражения, которые, в свою очередь, могут содержать достаточно много таких малых элементарных сравнений. Кроме того, для проверки в программах результатов выполнения отдельных блоков, логические значения могут изменяться и сохраняться в логических переменных. Эти объекты также могут принимать участие в разной степени сложности логических выражениях. И так далее.

Приведём пример логического оператора со сложным логическим выражением для демонстрации пяти уровней логических абстракций (см. фрагмент программы):

```
Var
  A, B : Boolean;
  i, j : integer;

Begin
  READ (i);
   j := 10;
    B :=True;
     A := i > j;
If A=TRUE AND (5*j-1 > i) OR NOT ODD(j) AND (-5 <> 4) Then

  j := 2;
End.
```



В сложном (!) логическом выражении, являющемся элементом логического оператора `If … Then`

```
A=TRUE AND (5*j-1 > i) OR NOT ODD(j) AND (-5 <> 4)
```

можно выделить следующие пять уровней абстракции представления различных сущностей (т.е. элементов-участников сравнения), результатов произведённых сравнений в программах, а также результата их взаимодействия (табл. 8.31,а).

Таблица 8.31,а

Пять уровней абстракции взаимодействия сущностей в логических выражениях

Первый уровень	***Операции сравнения*** **(=,<>, >,** и др.), дающие логические значения
Второй уровень	Логические ***значения (константы)*** **(***True, False***)**
Третий уровень	Логические ***переменные (которые описываются: Var A, B : boolean;)*** и сохраняют логические значения, а также значения функций, возвращающих логические значения
Четвёртый уровень	Логические ***операции (AND, OR, NOT и др.),*** объединяющие логические значения, переменные и функции в сложные логические выражения
Пятый уровень	Логические ***выражения (высказывания)*** **((**`5 > i)`, `(A=true AND (5 > i))` и др.), объединяющие все эти элементы вместе

Приведём пример хода выполнения вычислений в сложном логическом выражении с учётом приоритета выполняемых в нём операций разного типа (табл. 8.31,б). В вычислениях используется соглашение, что логическое значение `FALSE` в ТП эквивалентно значению `0`, а TRUE – `1`.

Таблица. 8.31,б

Выполнение вычислений в логическом выражениии
при `i=4, j=10` и `A=FALSE (0)`

Тип данных в выраже-ниях	Выражение	Последовательность выполнения	
		Частичное выражение	Ре-зуль-тат
Логи-ческий, целый	`A=TRUE AND (5*j-1 > i) OR NOT ODD(j) AND (-5 <> 4)`		
		5*j–1=	49
		ODD(j) =	0
		NOT ODD(j) =	1
		A=TRUE =	0
		5*j–1 > i =	1
		–5 <> 4 =	1
		A=TRUE AND (5*j-1 > i) =	0
		NOT ODD(j) AND (-5 <> 4)	1
		(FALSE OR TRUE) 0 OR 1 =	1
		Результат	**1 т.е. TRUE**

Таким образом, так как конечное значение сложного логического выражения имеет значение TRUE, будет выполнен оператор присваивания `j := 2,` являющийся элементом условного оператора `If … Then`.

Упражнения

1. Какие выражения являются простыми логическими выражениями?
2. Какие выражения являются сложными логическими выражениями?
3. Какие операции отношения (сравнения) между величинами Вы знаете?
4. Какие логические операции применяются при конструировании логических выражений?
5. Каков порядок вычисления сложных логических выражений?
6. Вычислите значения логических выражений и реализовать решение в программе:

   а) Sqr(X) + Sqr(Y)<=4 при X=0.3, Y= -1,6;

   б) k mod 7 = k div 5-1; при k = 15;

   в) Odd(Trunc(10*p)); при p = 0.182.

7. Поясните ошибки в следующих записях:

а) 1 And 0 б) True + False в) True < 0

г) NOT 2 = 5 д) X>0 or Y=4 е) NOT NOT У OR OR D.

8. Укажите порядок выполнения операций при вычислении выражений:

а) A and b or not c and d;

б) (X >= 0) or T and Odd(X) or (Y*Y <> 4).

9. Вычислите значения выражений и проверьте свои вычисления в программах:

а) NOT (Odd(n)); при n=0;

б) T and (P mod 3 = 0); при T=true, P=101010;

в) (X*Y<>0) and (Y>X); при X=2, Y=l;

г) (X*Y<>0) or (Y>X); при X=2, Y=l;

д) A or (NOTB); при A= false, B=true.

10. Запишите в виде логического выражения следующие условия:

а) Четырёхугольник ABCD является квадратом.

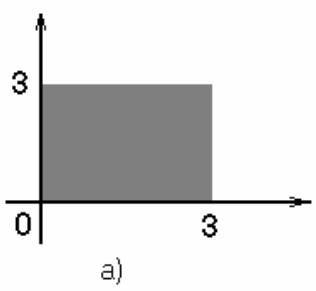
б) Четырёхугольник ABCD является ромбом.

в) Четырёхугольник ABCD является трапецией.

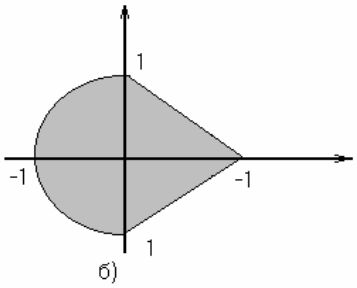
г) Треугольник ABC равнобедренный.

д) Треугольник ABC равносторонний.

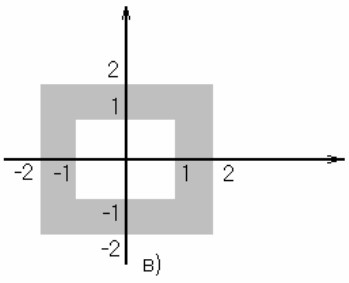
11. Составьте программу, которая определяет принадлежит точка A(x,y) заштрихованной области или там её нет. Если принадлежит, выведите на экран значение TRUE, иначе - FALSE.



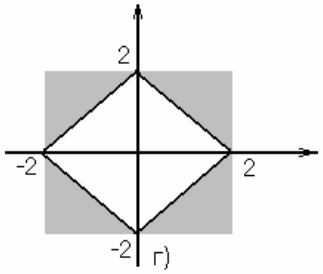
а)



б)



в)



г)

8.12. Использование логических операций и операций отношения для записи сложных условных выражений

Довольно часто, при вычислении математических выражений в программах встречаются сложные условия, требующие учёта расположения значений аргументов функциональных зависимостей на числовой оси. Тогда в логических операторах Вам нужно соответствующим образом конструировать логические выражения, которые отвечают каждому конкретному случаю.

Рассмотрим несколько таких логических выражений для наиболее характерных примеров и их реализаций на языке ТП.

1. Пусть аргумент ***x*** должен принадлежать отрезку [0,4], что математически записывается так

$$x \in [0,4]$$

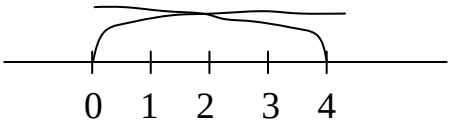


Рис. 8.30. Интервал типа "отрезок" $0 \le x \le 4$

То есть, если в переменной ***x*** будет содержаться значения –3 или +10, Ваша программа должна «понять», что эти значения не принадлежат к указанному выше отрезку числовой оси. И наоборот, если ***x*** примет значение 2, то Ваша программа должна «понять», что мы попали в заданный интервал, т.е. на заданный отрезок числовой оси.

Вероятно, Вы помните, что язык ТП не позволяет целиком записать условную конструкцию `0<= x <=4`, которая бы могла решить нашу проблему. Поэтому, необходимо разложить ее на атомарные сравнения, а потом уже объединить в единое выражение. И вот здесь нам на помощь приходят логические операции. Чтобы верно выбрать нужную логическую операцию, рассмотрим вероятные вариации переменной ***x*** на интервале $(-\infty,+\infty)$, с учетом заданного условия $0 \le x \le 4$ поодиночке и во взаимодействии логических выражений (табл. 8.32).

Сравнивая полученные результаты из таблицы 8.32, можно увидеть, что на языке ТП сложное логическое выражение $0 \le x \le 4$ запишется в таком виде:

```
(x >= 0) AND (x <= 4)                    (8.4)
```

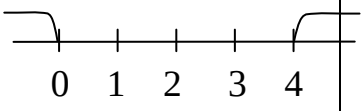
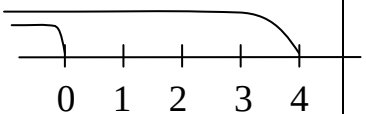
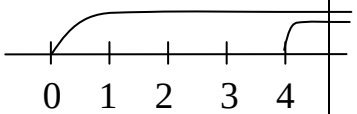
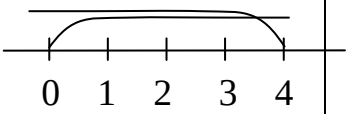
Например, для разных значений, которые могут содержаться в переменной ***x***, значения логического выражения (8.4) принимают значения (при $x = -3; 2; 6$):

```
(-3 >= 0) AND (-3 <= 4) равняется 0 (False);
(2 >= 0)  AND  (2 <= 4) равняется 1 (True);
(6 >= 0)  AND  (6 <= 4) равняется 0 (False).
```

Таким образом, компьютер понимает, что когда логическое выражение равняется 0 (`False`), значение ***x*** не принадлежит отрезку, а когда равняется 1 (`True`) – значение ***x*** принадлежит отрезку.

Таблица 8.32

Значения простых логических выражений

№ п/п	Атомарные (простые) логические выражения	Их логические значения для аргумента ***x***, принадлежащего отрезку [0,4], (при ***x*** = 2)	Взаимодействие полученных логических значений	Графическое изображение
1.	***x*** < 0	(***x*** < 0) = (2 < 0) = 0	(0 AND 0) = 0	
	x > 4	(***x*** > 4) = (2 > 4) = 0		
2.	***x*** <0	(***x*** < 0) = (2 < 0) = 0	(0 AND 1) = 0	
	x ≤ 4	(***x*** ≤ 4) = (2 ≤ 4) = 1		
3.	0 ≤ ***x***	(0 ≤ ***x***) = (0 ≤ 2) = 1	(1 AND 0) = 0	
	x > 4	(***x*** > 4) = (2 > 4) = 0		
4.	0 ≤ ***x***	(0 ≤ ***x***) = (0 ≤ 2) = 1	(1 AND 1) = 1	
	x ≤ 4	(***x*** ≤ 4) = (2 ≤ 4) = 1		

Примечание: ❶ Напоминаем, что взаимодействие логических значений, полученных в результате сравнений, которые объединены логической операцией OR, дает совсем другие результаты:

(0 OR 0) = 0, (0 OR 1) = 1, (1 OR 0) = 1, (1 OR 1) = 1.

❷ В таблице значение 0 эквивалентно `False`, а 1 – `True`.

2. Пусть Вам нужно программно учитывать требования, чтобы значения аргумента ***x*** находились за пределами отрезка [0, 4], то есть выполнялось условие 0>***x***>4. Тогда, используя вспомогательную таблицу (см. табл.8.33), получим:

```
(x < 0) OR (x > 4)
```
(8.4)

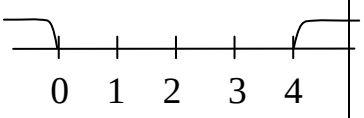
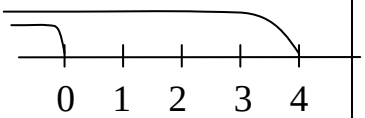
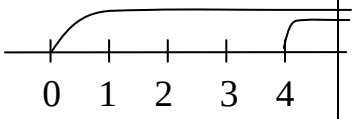
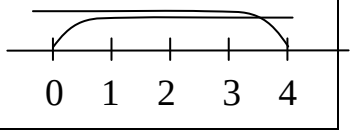
Для разных значений, которые могут содержаться в переменной ***x***, значения логического выражения (8.4) равняются (при ***x*** = –3; 2; 6):

```
(-3 < 0) OR (-3 > 4) равняется 1;
( 2 < 0) OR ( 2 > 4) равняется 0;
( 6 < 0) OR ( 6 > 4) равняется 1.
```

Таким образом, компьютер понимает, что когда значение логического выражения равняется 0, значение аргумента ***x*** принадлежит отрезку, а когда равняется 1 – значения ***x*** находятся за пределами отрезка и условие 0 > ***x*** > 4 выполняется.

Таблица 8.33

Значения простых логических выражений

№ п/п	Атомарные (простые) логические выражения	Их логические значения для аргумента ***x***, не принадлежащего отрезку [0,4], (***x*** = 6, или –6)	Взаимодействие полученных логических значений	Графическое изображение
1.	***x*** < 0	(***x*** < 0) = (6 < 0) = 0	(0 OR 1) = 1	
	x > 4	(***x*** > 4) = (6 > 4) = 1		
2.	***x*** <0	(***x*** < 0) = (6 < 0) = 0	(0 OR 0) = 0	
	x < 4	(***x*** ≤ 4) = (6 ≤ 4) = 0		
3.	***x*** > 0	(***x*** > 0) = (6 > 0) = 1	(1 OR 1) = 1	
	x > 4	(***x*** > 4) = (6 > 4) = 1		
4.	***x*** > 0	(***x*** > 0) = (6 > 0) = 1	(1 OR 0) = 1	
	x < 4	(***x*** < 4) = (6 < 4) = 0		

3. Рассмотрим случай, когда аргумент ***x*** должен принадлежать одновременно двум интервалам: [2, 5] и [10, 14] (рис. 8.31):

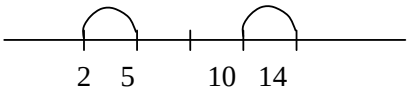


Рис. 8.31. Принадлежность значений переменной одновременно двум отрезкам

Для этого случая выражение на языке ТП примет вид:

```
((x >= 2) AND (x <= 5)) OR ((x >= 10) AND (x <= 14))
```

4. Для случая, когда аргумент ***x*** должен находиться за пределами интервалов [2, 5] и [10, 14] можно записать такое сложное логическое выражение:

```
((x < 2) OR (x > 5)) AND ((x < 10) OR (x > 14))
```

Понятно, что, используя логические операции и операции сравнения (отношения), можно записать логические условия произвольной степени сложности.

Пример: Требуется записать логическое выражение на языке ТП для вычисления функции:

$$y(x) = \begin{cases} \sqrt{x^2+1}, & 1< \boldsymbol{x} <6 \\ \dfrac{1}{x^3}, & 1\geq \boldsymbol{x} \geq 6 \end{cases}$$

Такое выражение можно записать с помощью двух разных форм, которые дадут одинаковый результат (см. первый и второй логические операторы на языке ТП):

```
Program VariousIf;
Var
  y, x : real;
Begin
  x := 2.0;
  if (x > 1.0) AND (x < 6.0) then {первый оператор}
    y := sqrt(x*x+1)
  else
      y := 1/(x*x*x);
  if (x <= 1) or (x >= 6) then    {второй оператор}
    y := 1/(x*x*x)
  else
    y := sqrt(x*x+1);
End.
```

Упражнения

1. Даны координаты вершин четырехугольника $A(x_1,y_1)$, $B(x_2,y_2)$, $C(x_3,y_3)$ и $D(x_4,y_4)$. Определить, является ли данный четырехугольник ромбом. Для вычислений воспользуйтесь формулой расстояния между двумя точками $A(x_1,y_1)$ и $B(x_2,y_2)$: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

2. Даны координаты трех вершин треугольника: $A(x_1,y_1)$, $B(x_2,y_2)$ и $C(x_3,y_3)$. Определить, является ли данный треугольник равносторонним. Для вычислений воспользуйтесь формулой расстояния между двумя точками $A(x_1,y_1)$ и $B(x_2,y_2)$: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

3. Даны три числа. Требуется найти наименьшее из них и вывести его значение на экран.

4. Даны три числа. Найти большее из них и вывести его значение на экран.

5. Составить программу определения номера наименьшего элемента из трех x_1, x_2, x_3.

6. Расположите три числа *a, b, c* в порядке увеличения их значений и выведите на экран.

7.Составьте программу нахождения произведения двух наибольших из трех чисел *x, y, z* .

8.13. Управляющие структуры (операторы) языка ТП. Простые операторы.

Операторы языка ТП предназначены для описания действий, которые будут выполняться при реализации разработанного заранее алгоритма. Структурно, они являются элементами языка, служащего для организации проведения операций над данными (фактически они явялются глаголами языка ТП и отвечают на вопрос "что сделать? ": присвоить, повторить и так далее). Однако следует постоянно помнить, что разработка алгоритма всегда проводится на основе знания и умелого использования необходимых типов данных и операторов, которые могут помочь эти данные обработать.

Операторы (в соответствии с синтаксисом) делятся на две основные группы:

❶ простые операторы;

❷ сложные операторы.

Операторы отделяются друг от друга "разделителем" – "**;**", то есть символом "точка с запятой". Например:

```
... x:=0.324; r:=cos(x); t :=sqrt(sqr(x*x+r)+1; ...
```

Поэтому в составном операторе (`begin end;`) перед ключевым словом `end` знак "**;**" можно не ставить. Но если он поставлен, то считается что после разделителя "**;**" располагается ***пустой оператор***. К примеру,

```
... begin ... x:=0324; r:=cos(x);      end;
```

Пустой оператор

Простые операторы – это операторы, которые не содержат в себе других операторов (табл. 8.32).

Таблица 8.32

Простые операторы языка ТП

Название оператора	Форма записи оператора
Оператор присваивания	`A := B;`
Операторы вызова процедуры	`writeln('Уведить w: ',w); readln;`
Оператор перехода	`goto` (Подробно не рассматривается далее, потому что не отвечает требованиям структурного программирования!)

Примечание. Обратите внимание на то, что оператор ***процедуры***, в отличие от других простых операторов (`:=`), которые записываются всегда одинаково (!), имеет только общую смысловую ***форму*** записи:

<имя процедуры и (возможно!) (список фактических параметров)> ;

А ***содержание*** записи каждого нового оператора – разное, потому что все процедуры, выполняющие разные действия, имеют:

1. Разные имена;
2. Разное количество фактических параметров (или могут совсем их не иметь!).

Причастность группы символов, которые отображают имя и список фактических параметров в скобках, к операторам процедуры свидетельствует символ (**;**) в конце их последовательности.

Оператор присваивания "**:=**" служит для присваивания значений переменным. Другими словами, он позволяет заносить результаты, вычисленные с помощью выражений (арифметических, логических и др.), в конкретные места памяти компьютера, поименованные для удобства дальнейшего использования помещённых в них значений.

Смысл действия оператора присваивания можно представить так: ***следует заменить (заместить) предыдущее значение переменной, имя которой стоит в левой части оператора, на значение выражения, которое стоит в правой части оператора присваивания.***

Таким образом, оператор

```
A := B + C;
```

означает то, что необходимо вычислить сумму значений, содержащихся в переменных `B` и `C` и заменить этой суммой предыдущее значение переменной `A`. Специфические особенности оператора присваивания проявляются очень ярко в такой форме:

```
N := N + 1;
```

Смысл этого выражения заключается в том, что переменной `N` необходимо присвоить её предудущее значение, увеличенное на 1.

Оператор процедуры. Выполнение оператора процедуры приводит к активизации действий, которые описаны в её теле. Наиболее часто в программах используются встроенные в систему ТП процедуры ввода данных в переменные программы с экрана (`read, readln`) и вывода данных на экран компьютера (`write, writeln`). Их часто называют операторами ввода–вывода:

```
... writeln('Введите a,e: '); readln(a,e); ...
```

После выполнения этой группы операторов на чёрном экране DOS появится строчка текста `'Введите a,e: '` и компьютер буде ждать от пользователя ввода двух значений данных, отвечающих описанию их в программе. Например, если они описаны как действительные переменные, то необходимо будет ввести подряд, ***но разделённые пробелом***, два действительных значения, допустим такие: `2.0 0.291`.

Следует добавить, что в состав программы могут быть включены комментарии – которые представляют собой строчки текста, поясняющие содержание действий программиста, но не влияют на ход выполнения самой программы. Хорошиим стилем программирования в софтверных[23] фирмах (то есть тех, которые производят программное обеспечение) считается такой, когда 45% текста программы составляют только комментарии!

Комментарии размещаются в специальных скобках, первая из которых является открывающей для начала комментария, а последняя – закрывающей (табл. 8.33).

[23] Софтверный (от английского слова “software”) – тот, который имеет отношение к производству программного обеспечения.

Таблица 8.33

Примеры комментариев

	Открывающая скобка	Комментарий	Закрывающая скобка
Три разных вида скобок для записи комментариев	**{**	Первый комментарий	**}**
	/*	Второй комментарий	***/**
	(*	Третий комментарий	***)**

Упражнения

1. Какие операторы ТП зовутся простыми?
2. Как отделяются операторы один от другого?
3. Какой оператор зовётся “пустым”?
4. Какой смысл имеет оператор присваивания?
5. Чем оператор процедуры отличается от других простых операторов?
6. Что такое комментарий и для чего он применяется?

8.14. Сложные (структурные) операторы управления выполнением алгоритмов. Составной оператор `begin ... end`

Сложные (структурные) операторы включают в себя другие операторы и управляют последовательностью их выполнения (таблица 8.34).

Таблица 8.34

Сложные операторы языка ТП

Название оператора	Форма записи оператора
Составной оператор или структурный оператор, который ещё называют операторными скобками	`begin ... end;`
Условный оператор	`if ... then ... else`
Оператор выбора	`case ... of ...else ... end`
Оператор цикла с параметром	`for ... to ... do`
Оператор цикла с предусловием	`while ... do`
Оператор цикла с послеусловием	`repeat ... until`

Иногда синтаксис языка ТП требует, чтобы в некотором месте программы находилось не более одного оператора, а для решения задачи их требуется несколько. ***Составной оператор*** включает в себя их всех, но компьютером рассматривается как один (рис.8.32).

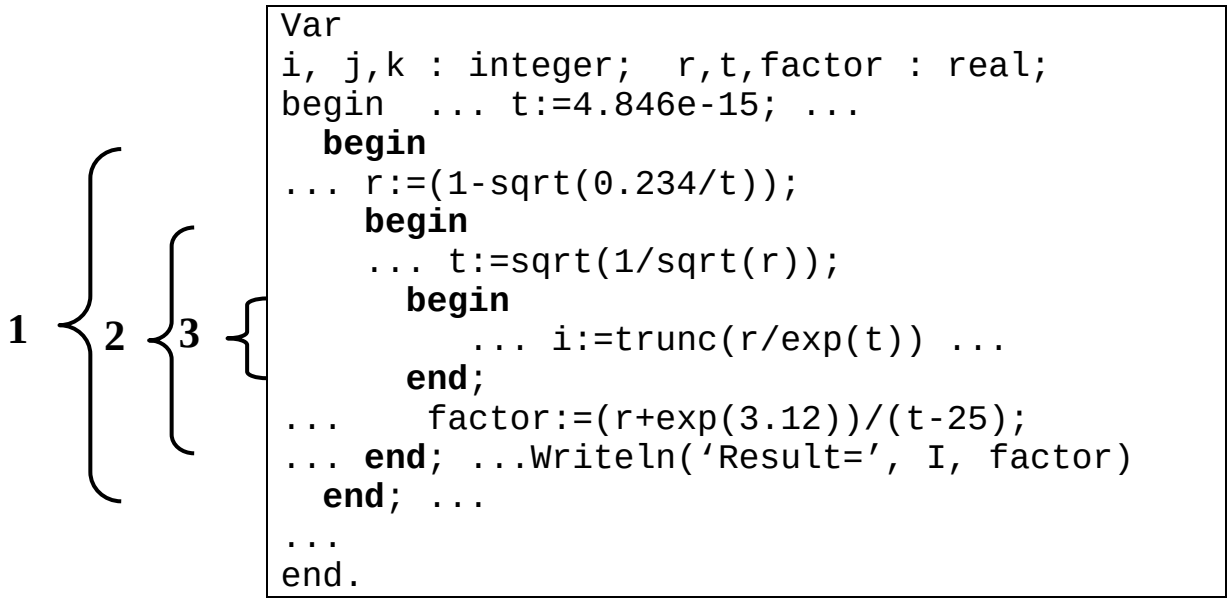


Рис. 8.32. Уровни (1, 2 и 3) включения операторов в операторные скобки `begin` и `end`

Следовательно, составной оператор объединяет группу операторов в единое целое, после чего их можно рассматривать как один оператор. Таким образом, составной оператор состоит из последовательности операторов,

которые объединяются и расположены между ключевыми словами (операторными скобками) `begin` и `end`:

❶ скобки, которые открываются, соответствуют оператору – `begin`;

❷ скобки, которые закрываются, соответствуют оператору – `end`.

Не следует путать операторные скобки с аналогичными зарезервированными словами, которыми начинается и заканчивается раздел операторов программы. Следует также внимательно следить, чтобы количество операторных скобок, которые открываются соответствовало количеству скобок, которые закрываются за соответствующими группами операторов (рис. 8.32).

Приведём список операторов которым необходим составной оператор для объединения простых операторов в группы, рассматриваемые компьютером как единое целое (табл. 8.34).

Таблица 8.34

Операторы, которым необходим составной оператор

Название оператора	Форма записи оператора

Условный оператор	`if ... then`
	`  begin <операторы> end`
	`else`
	`  begin <операторы> end;`

Оператор выбора	`case ... of`
	`<МЕТКА> : begin <операторы> end`
	`else`
	`begin <операторы> end`
	`end;`

Оператор цикла с параметром `for`	`for ... to ... do` `begin <операторы> end;`
Оператор цикла с предусловием `while`	`while ... do` `begin <операторы> end;`

Упражнения

1. Какие операторы называются составными?

2. Для чего используются составные операторы?

3. Каким сложным операторам необходимы составные операторы, а каким нет?

4. Перечислите все сложные операторы языка ТП.

8.15. Операторы разветвления алгоритмов. Условный оператор `if`. Оператор выбора `case`

Вычислительный процесс называется разветвляющимся, если в зависимости от выполнения определённых условий в программе следует выполнять те или другие группы операторов. Количество групп операторов и условия, на основании котрых они выполняются, конструирует сам программист. Каждая такая группа операторов называется ветвью алгоритма вычисления. Выбор той или иной ветви осуществляется уже при выполнении программы в результате проверки запрограммиованных условий, которые зависят от значений входных данных и некоторых промежуточных результатов.

В Турбо Паскаль включены фактически два условных оператора `if` и `case`. Они имеют полную или неполную форму (см. далее по тексту), но первый называется условным оператором, а второй – оператором вибора.

Условный оператор `if` предназначен для выполнения или невыполнения разных групп операторов в зависимости от выполнения или невыполнения условий, задаваемых пользователем. Он имеет так называемую неполную и полную формы. Их синтаксис:

Неполная форма:

`if` <логическое выражение условия> `then` <оператор P1>;

Полная форма: `if` < логическое выражение условия > `then` <оператор P1> `else` <оператор P2>;

Логические выражения конструируются с помощью следующих элементов:

❶ круглых скобок ( );

❷ логических значений (`false, true`);

❸ логических переменных, представляемых их именами (`Switch, tt1` и др.);

❹ операций сравнения (=, <, >, <>, <=, >=);

❺ логических операций (`not, or, and`).

Операторы P1 и P2 могут быть:

❶ простыми операторами;

❷ простыми или сложными операторами, объединёнными операторными скобками `begin … end;`

❸ сложными операторами и, в том числе, другими условными операторами.

Та как операторы в языке ТП разделяются символом "`;`" (точка с запятой), внутри сложного логического оператора этот символ использовать НЕЛЬЗЯ!

ошибка!

```
if A > B then C := A ; else C := B;
```

В этом примере символ точка с запятой, стоящий перед служебным словом `else` заканчивает текст оператора `if`. А это приводит к синтаксической ошибке, поскольку оператора, начинающегося со служебного слова `else` – не существует.

Для рассмотрения примера использованияя логического оператора составим программу вычисления следующего выражения (14.1):

$$y = \begin{cases} x^2 & 0 < x < 1 \\ x^{\frac{1}{2}} & 1 < x < 3 \\ |x|^{-\frac{1}{2}} & 3 < x < 0 \end{cases} \qquad (8.5)$$

Алгоритм вычисления выражения (8.5) на языке ТП может выглядеть так:

```
Program Sign_X;

Var
  x, y : real;
Begin
    writeln('Введите x');
       readln(x);

  if (x > 0) and (x < 1) then  y := sqr(x);
  if (x > 1) and (x < 3) then  y := sqrt(x);
  if (x > 3) and (x < 0) then  y := 1/sqrt(abs(x));

   writeln('Значение Y=', y:8:3);

end.
```

После зарезервированных (служебных) слов `then` и `else` должен стоять только один (!) оператор. Поэтому, если после них необходимо выполнить несколько операторов, следует их объединить в один операторными скобками `begin` и `end.` Существуют, так называемые, вложенные операторы `if` (вложенные ифы). Это значит, что за зарезервированным словом `else` помещается следующий оператор `if.`

Например, пусть необхдодимо вычислить значения функции `y(x) = sign(x)` в соответствии с выражением (8.6).

$$y\,(x) = sign\,(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \qquad (8.6)$$

Алгоритм последовательности вычислений по этой формуле с использованием вложенного оператора `if` может выглядеть так.

```
  Label 10;
Var
   x, y : real;
Begin
   writeln('Введите x');
   readln(x);
  if (x < 0) then
                  y := -1
  else
       if (x = 0) then
                        y := 0
    else  y := 1;
  writeln('Значение Y=', y:8:3);
end.
```

При необходимости конструирования сложных логичеких условий возникают синтаксические скложности корректного использования оператора `if`, когда необходимо вкладывать один такой оператор `if` в другой. Если его отделять операторными скобками `begin ... end`, больших проблем не возникает.

Например:

```
if условие
     then   begin
                  оператор;
                   if условие
                    then оператор
                     else оператор;
                      оператор
            end
     else   begin
                  оператор;
                   if условие
                    then оператор
            end;
```

Однако, если вложенный оператор `if` является единственным оператором в ветви альтернативы, то может возникнуть неоднозначность: какому `if` соответствует ветвь `else`.

Например:

```
if условие then
if условие  then
    оператор
    else  оператор;
```

В таких случаях должно выполняться следующее правило.

Ключевое слово `else` связывается с ближайшим, стоящим перед ним ключевым словом `if`, которое ещё не было связано с другим ключевым словом `else`.

Если в предыдущем примере более конкретно представить структуру вложенности и записать `else` на одном уровне с `then`, которому оно соответствует, то можем получить такой фрагмент программы.

```
if условие
    then
        if условие then оператор
                                    else оператор;
```

Поэтому следует запомнить следующее ограничение. Оператор, который располагается непосредственно после служебного слова **`then`** не ***должен*** быть ***условным***, в то время как оператор, располагаемый после **`else`** ***может быть любым***, и, в том числе, условным. Это ограничение возникает, потому что компилятору непонятно к какому из условий ***относится*** `else` (проверьте на компьютере) (табл.8.35):

Таблица 8.35

Примеры конструирования вложенных операторов `if`

НЕВЕРНО	ВЕРНО (второй оператор `if` входит в составной оператор)	ВЕРНО (оператор `if` используется по правую сторону от оператора `else`)

`if a > 0 then`
`  if a < 2 then`
`  a := 1`
`  else`
`  a := 3;`
	`if a > 0 then`
	`  begin`
	`  if a < 2 then`
	`  a := 1 end`
	`  else  a := 3;`
		`if not a > 0 then`
		`  a := 3`
		`  else`
		`  if a < 2 then`
		`  a := 1;`

При использовании условного оператора `if` после `else` никаких двусмысленностей не возникает.

Примером, когда логическое выражение в операторе `if` имеет более сложную структуру, может быть решение задачи определения, можно ли построить треугольник из длин отрезков, задаваемых в переменных: x, y, z (x>0, y>0, z>0).

Такой условный оператор имеет вид.

```
if (X+Y>Z) AND (X+Z>Y) AND (X+Z>Y)
then Writeln('ТРЕУГОЛЬНИК ПОСТРОИТЬ ВОЗМОЖНО')
else Writeln('ТРЕУГОЛЬНИК ПОСТРОИТЬ НЕВОЗМОЖНО');
```

Оператор выбора **`case`** служит для переключения на выполнение групп операторов в зависимости от значения некоторого целочисленного переключателя или выражения. Синтаксис полной его формы такой:

```
case <переключатель> of
     <метки M1> : <операторы P1>;
     <метки M2> : <операторы P2>;
…..................................……
     <метки MN> : <операторы PN>;
else < операторы PM>  {альтернатива предыдущим операторам}
end;
```

Где:

❶ переключатель – может быть целочисленной переменной или целочисленным выражением (`i, Iswitch, 2*i, abs(j)` и др.).

❷ метки M1 ... MN – могут отображаться:

→❶ Одним целочисленным значением (3, 15 и т.д.).

→❷ Диапазоном целочисленных значений (1..27).

→❸ Перечислением целочисленных констант (2,4,10,75).

→❹ Набором комбинаций элементов первых трёх пунктов (1,3..15,45,54..71).

❸ операторы P1…PN, PM – это операторы любой степени сложности, но взятые в операторные скобки `begin ... end`.

Для проведения аналогии между двумя вышеописанными операторами значение функции, вычисленное с помощью условного оператора `if` по формуле (8.5) можно реализовать с помощью оператора `case` следующим образом.

```
Var
   x, y : real; i : integer;
Begin
   writeln('Введите x');
                      readln(x);
  if (x > 0) and (x < 1) then
                              i := 1
  else
  if (x > 1) and (x < 3) then
                              i := 2;

case i of
1 : y := sqr(x);
2 : y := sqrt(x);
else
y := 1/sqrt(abs(x));
end;

writeln('Значение Y=', y:8:3);
end.
```

Упражнения

1. Даны два числа. Заменить второе число нулём, если оно больше первого, и оставить его без изменения, если это не так.

2. Найти наименьшее из трёх данных чисел.

3. Найти наибольшее из трёх данных чисел.

4. Дано три числа. Возвести в квадрат те из них, значения которых не отрицательны. Отрицательные числа оставить без изменения.

5. Даны три числа *a*, *b* и *c*. Выяснить, верно ли, что *a*<*b*<*c*. Ответ вывести на экран в текстовой форме: «Верно» или «Неверно».

6.Написать программу для вычисления разных значений функции *y*:

$$y=\begin{cases}\sin x, & при \quad x\le 0\\ \operatorname{arctg} x, & при \quad 0<x\le \pi/4\\ \log_2 x, & при \quad \pi/4<x\le 32\\ 1/x, & в\ остальных\ случаях\end{cases}$$

Протестировать программу при разных значениях аргументов.

7.Написать программу для вычисления разных значений функции *z*:

$$z=\begin{cases}\textbf{ln}\ (x), & при \quad x<-\pi\\ \sin x+\cos 2x, & при \quad -\pi\le x<\pi\\ x^3+1, & при \quad \pi\le x<10\\ \dfrac{x+1}{x^2+8}, & при \quad 10\le x<100\\ \ln x, & в\ остальных\ случаях\end{cases}$$

Протестировать программу при разных значениях аргументов.

8.Написать программу для вычисления разных значений функции *z*:

$$z=\begin{cases}\operatorname{arctg}\dfrac{x}{y}, & при \quad y\ne 0\\ \arcsin\dfrac{x}{y}, & при \quad y\ne 0\\ 0, & в\ остальных\ случаях\end{cases}$$

Протестировать программу при разных значениях аргументов.

9. Алгоритмы программ 6, 7, 8 реализовать с помощью оператора ***выбора***.

8.16. Циклические вычислительные процессы и операторы циклов. Циклы с параметром. Оператор цикла с параметром `for`

Когда вычислительный процесс содержит многоразовые (неоднократные) одинаковые действия, выполняемые по одним и тем же математическим зависимостям, но для разных значений переменных, принадлежащих им, его называют циклическим. Например, если необходимо вычислить сумму:

$$\sum_{n=1}^{10} a^{n} \quad (8.7),$$

то она включает следующие составляющие (табл. 8.36).

Таблица 8.36

Составляющие вычисления суммы по формуле (8.7)

Изменения значений степени *n*	Элементы суммы	Другая запись элементов суммы и их компонентов
1	a^1	a
2	a^2	a·a
3	a^3	a·a·a
4	a^4	a·a·a·a
5	a^5	a·a·a·a·a
6	a^6	a·a·a·a·a·a
7	a^7	a·a·a·a·a·a·a
8	a^8	a·a·a·a·a·a·a·a
9	a^9	a·a·a·a·a·a·a·a·a
10	a^{10}	a·a·a·a·a·a·a·a·a·a
Элементы суммирования	$a^1 + a^2 + a^3 + a^4 + a^5 + a^6 + a^7 + a^8 + a^9 + a^{10}$	a + a·a + a·a·a + a·a·a·a + a·a·a·a·a + a·a·a·a·a·a + a·a·a·a·a·a·a + a·a·a·a·a·a·a·a + + a·a·a·a·a·a·a·a·a + a·a·a·a·a·a·a·a·a·a
Результат выполнения сложений для 10-ти эл-тов	$\sum_{n=1}^{10} a^{n}$	$\sum_{n=1}^{10} a^{n}$

Понятно, что если мы знаем конкретное количество этих элементов и как они конструируются, процесс можно запрограммировать достаточно легко. То есть, необходимо циклически, десять раз подряд, произвести определённые операции умножения и сложения. И для выполнения таких многоразовых действий предназначены несколько специальных операторов языка Турбо Паскаль.

Таким образом, группы операторов, которые неоднократно повторяются, называются ***циклами,*** а переменные, меняющие свои значения в цикле – ***переменными*** цикла.

Как правило, алгоритм циклической структуры в наиболее общем виде должен содержать такие фрагменты действий и соответствующих им операторов:

❶ подготовку цикла: инициализация (задание начальных значений) переменным цикла;

❷ тело цикла: действия, которые повторяются в цикле для разных значений переменных цикла;

❸ модификацию (изменения) значений переменных цикла перед каждым новым его повторением;

❹ управление циклом: проверку требований продолжения (или завершения) цикла и переход на начало тела цикла, или выход из него.

Одним из примеров простого циклического процесса является задача табулирования значений функции одной переменной $y = f(x)$.

Для её вычисления задаётся начальное X_0 и конечное X_N значения аргумента x этой функции, а также количество K значений функции, которые необходимо вычислить на всём отрезке между X_0 и X_N при дополнительно вычисленных на участке остальных аргументов X_i. Вышеприведенные данные дают возможность вычислить шаг h, с которым изменяются значения аргумента X на выбранном участке от X_0 до X_N.

Следует помнить, что количество K значений аргументов на единицу больше количества отрезков N, на которые, как правило, делят участок между X_0 и X_N. То есть $N = K - 1$.

Например, в таблице 8.37 зададим следующие виражения и значения для вычисления функции, которая вычисляет возведение аргумента X в степень 1,2 (то есть одну целую и две десятых).

Таблица 8.37

Составляющие процесса вычисления значений аргументов и функций на отрезках участка $[X_0,X_N]$

Значения	Величина	Виражения/Значения
Функция	$Y = f(X)$	$X^{1,2}$
Начальное значение аргумента	X_0	1
Конечное значение аргумента	X_N	5
Количество отрезков на участке	N	4
Количество значений функции	K	5
Шаг изменения аргумента функции	h	$(X_N - X_0)/N = 1$

В таблице 8.38 приведены вычисленные значения функции $Y(X)=X^{1,2}$ (X в степени 1.2), вычисленные на участке [1,5], которая разделена на четыре части с шагом $h= (5–1)/4 = 1$. В таблице 8.38 также показано, что каждое следующее значение аргумента X можно вычислять двумя разными методами. Либо добавляя к каждому предыдущему значению аргумента X величину шага h:

$X_{i+1} = X_i + h$, либо увеличивая начальное значение аргумента X_0 на количество шагов, кратних количеству отрезков, предшествующих значению X_i:

$X_i = X_0 + i*h$.

Таблица 8.38

Значения аргументов X и значений функции $Y=f(X)=X^{1,2}$

Значения аргументов X	Значения функции $X^{1,2}$
$X_0 = 1$	1.0
$X_1 = X_0 + h = 1 + 1 = 2$	2.3
$X_2 = X_1 + h = 2 + 1 = 3$, или $X_2 = X_0 + 2h = 1 + 2 = 3$	3.7
$X_3 = X_2 + h = 3 + 1 = 4$	5.3
$X_4 = X_3 + h = 4 + 1 = 5$, или $X_4 = X_0 + 4h = 1 + 4 = 5$	6.9

Выходя из этих значений аргументов можно вычислить практически любые значения функции одной переменной (*sin x, cos x,* e^x, *(a+bx),* x^2, $\sqrt{x}$ и др.).

На рисунке 8.33 приведена схема размещения вышеперечисленных объектов в декартовых осях координат X и Y.

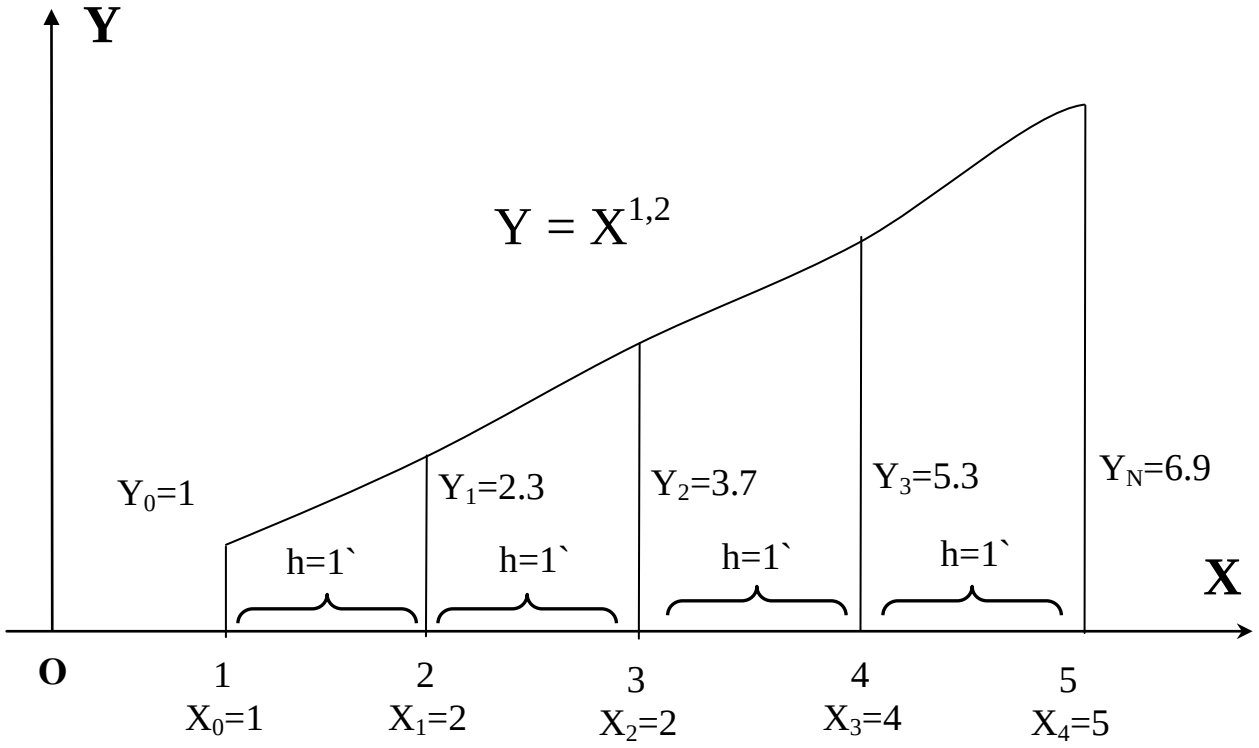


Рис. 8.33. Размещение и значения абсцисс X и значений ординат Y для заданных значений и выражений

Из данного примера легко увидеть, что циклы можно конструировать, если определены:

❶ начальное X_0 и конечное X_N значения аргумента;

❷ шаг h изменения аргумента;

❸ количество N отрезков на участке между X_0 и X_N;

❹ количество $K=N+1$ значений аргументов X.

Условиями выхода из цикла могут быть:

❶ исчерпание количества N вычисленных значений функции $Y=f(X)$ для всех N значений аргументов $X_0, X_1, \ldots, X_N$.

❷ превышение текущим значением аргумента X_i, который моделируется математическим выражением (к примеру, $X_{i+1} = X_i + h$), конечного значения аргумента X_N ,.

В общем случае, программно цикл можно реализовать с помощью следующих управляющих структур языка Турбо Паскаль:

❶ операторов присваивания (`:=`);

❷ условного оператора (`if … then … else`);

❸ оператора безусловного перехода (`goto`).

Вместе с тем, в языке ТП для этой цели существуют специальные операторы цикла, которые исключают необходимость конструирования программ с помощью других операторов.

Кроме того, операторы цикла обеспечивают более компактную, прозрачную и существенно более простую форму записи алгоритмов, а также дают возможность создавать более эффективные программы.

Таким образом, следует очень чётко понимать, что существуют ***два*** типа циклов:

❶ с ***наперёд известным*** количеством повторений операций тела цикла;

❷ с ***неизвестным наперёд*** количеством повторений операций.

Циклы первого типа называют также циклами со счётчиком или с параметром, имея в виду, что этот параметр должен изменяться в ходе вычислений. Число повторений тела цикла в этом случае подсчитывается с помощью специальной переменной (счётчика), для которой известны начальное и конечное (пороговое) значение и значение шага её изменения. Управление таким циклом осуществляется с помощью сравнения текущего значения счётчика с заданным пороговым (максимальным либо минимальным) значением. Переменную-счётчик часто называют параметром цикла, а сам цикл – циклом с параметром.

Оператор цикла с параметром в языке Турбо Паскаль является управляющей структурой, которая задаёт повторения процесса выполнения одного оператора или группы операторов в границах изменения значения некоторого параметра между двух целых чисел, из которых одно является начальным, а другое – конечным. Параметр, пробегающий в своих значениях от начального до конечного, называется управляющим параметром цикла (УПЦ). Тип УПЦ всегда ***целый*** (!), а изменение его значений производится с шагом только 1 (либо –1).

Общий вид оператора цикла с параметром следующий:

`for … to … do` (или `for … downto … do`).

Синтаксис написания этих двух форм данного оператора следующий:

for <имя УПЦ := начальное значение> **to** <конечное значение> **do** <оператор>;

for <имя УПЦ := конечное значение> **downto** <начальное значение> **do** <оператор>;

При использовании цикла `for … to … do` нужно иметь в виду следующее:

❶ в теле цикла может стоять только один оператор, а при необходимости использования большего их количества, последние объединяются в составной – операторными скобками `begin` и `end;`

❷ в средине тела цикла нельзя принудительно изменять значения УПЦ;

❸ в цикл можно входить только через его заголовок. То есть, нельзя передавать управление из за его границ нутрь него (рис. 8.34).

❹ переменная цикла может иметь только ***перечислимый тип***.

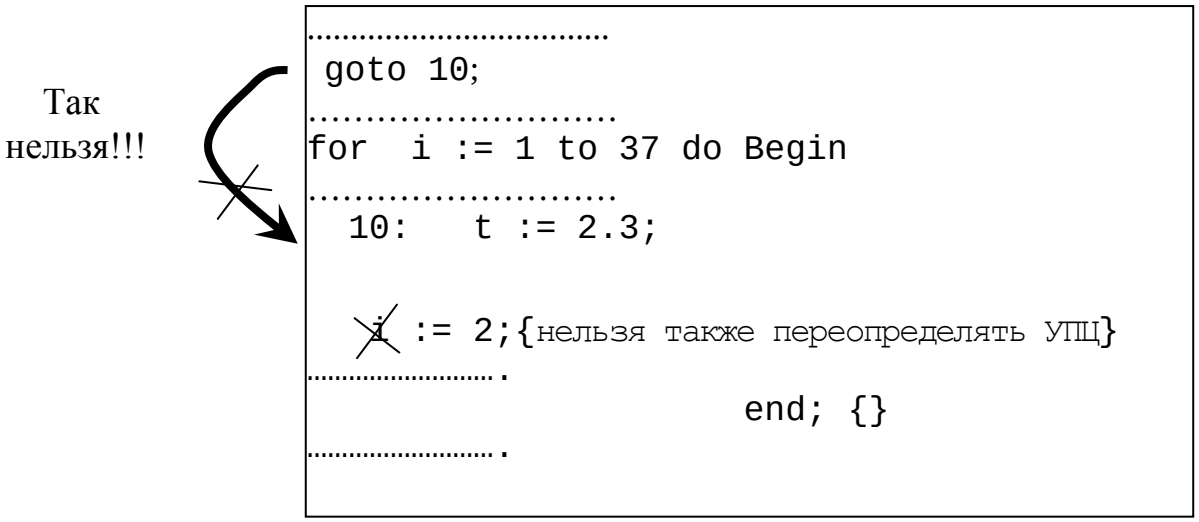


Рис. 8.34 Невозможность передачи управления внутрь цикла с параметром

Последовательность действий, происходящих, например, при вычислении факториала числа 10 (т.е. 10!) и выполнении для этого следующей последовательности операторов языка ТП с применением цикла `for`:

```
… p:=1; for i:=1 to 10 do p:=p*I; ...
```

можно описать следующим образом.

Первым делом, переменной `p` присваивается значение 1 (т.е. в поле памяти с именем `p` заносится значение 1).

На первом шаге начала работы цикла `for` управляющей переменной цикла (УПЦ) `i` присваивается значение 1 (`i:=1`). Создаётся константа 10 (`to 10`) для сравнения её с изменяющимися текущими значениями УПЦ. Извлечённое из переменной `p` значение 1 умножается на значение УПЦ, равное 1 (`p*I`). Результат (значение 1) заносится обратно в переменную `p`

(p:=p*I). Значение УПЦ i увеличивается на значение шага изменения УПЦ равное 1 и i принимает значение 2. Производится сравнение текущего значения i с константой 10. Так как при i=2 выполняется условие (i <= 10) = TRUE, и работа цикла должна быть продолжена. На втором шаге, предыдущее значение переменной p, равное 1, умножается на новое текущее значение УПЦ i=2 и в переменную p заносится значение 2. Результат сравнения значения i, увеличенного на 1 (т.е. i=3) со значением 10 продолжает иметь значение TRUE. На третьем шаге, после аналогичным описанным выше действиям, значение p станет равным p =1*2*3, т.е. 3!. На десятом шаге, после присваивания переменной p значения 1*2*3*4*5*6*7*8*9*10, т.е. 10! и очередного увеличения i на 1-цу, её значение станет равным 11. Результат сравнения i<=10 получит значение FALSE и управление будет передано на оператор, следующий непосредственно за оператором цикла. Таким образом работа цикла будет завершена.

Приведём пример вычисления с помощью ***цикла с параметром*** значения действительного числа, возведённого в целую степень $y = a^n$.

Для этого воспользуемся формулой $a^n = \underbrace{a \cdot a \cdot a \cdot \ldots a}_{\text{(умножить } n \text{ раз)}}$.

Дополнительно вычислим сумму $\sum_{i=1}^{n} a^i = \underbrace{a^i + a^i + \ldots + a^i}_{\text{(сложить } n \text{ раз при } i=1,2, \ldots n.\text{)}}$

```
{Вычисление заданной степени вещественного числа a}
                              { и суммы этих степеней}
Program Step;
  Var
    A, Y, S : real;
    I, N : integer;
Begin
    Writeln('Введём основание a и степень n');
    Readln(A, N);
      Y:=1;
       S:=0;
      For I:=1 to N do
        Begin
          Y:=Y*A;
           S:=S+Y;
        End;
     Writeln(N, 'Степень числа', A);
     Writeln('Равняется= ',Y, 'a сумма степенем=',S);
End.
```

Как правило, циклы с параметром используются со структурами данных типа массивов. Это связано в первую очередь с тем, что этот тип является статическим и с начала выполнения программы его границы должны быть описаны и принять конкретные числовые значения. А параметр цикла при

своём изменении проходя от начального значения индекса массива до конечного его значения, которое совпадает с пороговой величиной этого параметра, позволяет легко обрабатывать его (массива) элементы.

Упражнения

1. Поясните, что такое цикл?

2. Для чего нужен параметр цикла?

3. Для чего служит пороговое значение параметра цикла?

4. Поясните логику выполнения циклов `For ... To … Do` и `For ... DownTo ..Do`.

5. Какие переменные могут использоваться в качестве параметра цикла For? Какие переменные могут использоваться в качестве границ диапазона для параметра цикла For?

6. Какие фрагменты действий должен содержать алгоритм циклической структуры в наиболее общем виде?

7. Построить таблицу значений абсцисс и ординат для функции $f(x) = x - \sin(x)$ на отрезке $[0,\pi/2]$ с числом разбиений отрезка m=10 и вывести полученные данные на экран.

8. Написать программу для вывода на экран таблицы значений функции $y = tg\ x - sin\ 2x$ на промежутке $[-\pi/4; \pi/4]$ с шагом $\pi/16$.

9. Написать программу для вывода на экран таблицы значений функции $y = x^2 + 7x - 14$ на промежутке [-3; 8] с шагом 0,55.

10. Написать программу для вывода на экран таблицы значений функции $y = \dfrac{\sin x + \sin 2x}{\cos x + \cos 2x}$ на промежутке $[0; \pi]$ с шагом $\pi/12$.

11. Вычислить значения переменной S_n (суму N первых значений) выражений S для разных N, если:

$$S = 10 - 3\left(1 + \frac{2}{3}\right) + 7\left(2 - \frac{3}{9}\right)^2 - 11\left(3 + \frac{4}{27}\right)^3 + \ldots.$$

12. Вычислить значения конечной суммы S для выражения:

$$S = \sum_{n=1}^{5} n^2 + \sum_{n=1}^{12} n^3 .$$

13. Вычислить значения суммы S для конечного ряда: $\dfrac{1}{3} + \dfrac{1}{8} + \cdots + \dfrac{1}{n^2 - 1}$.

14. Вычислить значения конечного результата для выражения суммы S

$$\left(1 + \frac{1}{1(1+2)}\right) \cdot \left(1 + \frac{1}{2(2+2)}\right) \cdots \left(1 + \frac{1}{n(n+2)}\right).$$

В пп. 13 и 14 значения n вводить с экрана.

8.17. Оператор цикла с предусловием `while`. Оператор цикла с послеусловием `repeat`

Выше были рассмотрены алгоритмы с конструкцией циклической структуры типа:

```
for <УПЦ:=1> to <пороговое значение N> do <оператор>,
```

которая работает с заранее известным количеством повторений цикла N.
Вместе с тем, существуют такие случаи, когда количество повторений тела цикла вообще неизвестна, но заданы некоторые условия его завершения или продолжения. Для програмной реализации таких вычислительных процессов в языке ТП существуют два сложных оператора:

❶ оператор цикла с предусловием (`while ... do`);

❷ оператор цикла с послеусловием (`repeat ... until`).

Оператор цикла с предусловием имеет следующую общую форму записи:

```
while <УСЛОВИЕ> do <ОПЕРАТОР>;
```

Здесь:

❶ слова `while` (пока) и `do` (выполнять) являются служебными;

❷ `УСЛОВИЕ` – логическое выражение любой степени сложности;

❸ `ОПЕРАТОР` – любой оператор языка ТП и в том числе составной оператор. Обратите внимание на то, что в этом цикле может использоваться только ***один(!)*** оператор (присваивания, условный, выбора, и т.д.), либо же совокупность операторов, взятая в операторные скобки `begin … end`.

Важно иметь в виду, что цикл с предусловием выполняется тогда и только тогда, когда значение логического выражения <УСЛОВИЕ> равно логическому значению TRUE. Как только оно принимает значение FALSE – цикл завершается и управление передаётся оператору, который следует за этим циклом.

Если <УСЛОВИЕ> имеет значение FALSE до начала выполнения цикла, то нам не удастся войти в его тело и, следовательно, не выполнится ни один из операторов тела цикла.

Допустим, что Вам необходимо вычислить с помощью оператора цикла с предусловием ряд значений функции X^2 с шагом 1 для трёх разных случаев, указанных в таблице 8.39. Сложность конструирования цикла в этом случае состоит в необходимости учёта трёх логических условий:

❶ переменная X должна принадлежать отрезку [-200, 200];

❷ значения функции X^2 не должны превышать значения 10^3;

❸ в двух последних вариантах задания необходимо учесть вычисление чётных и нечётных значени функции X^2.

Сложность решения этих задач заключается у том, что Вы должны учесть начальное значение аргумента X, его участие в трёх вышеприведенных логических условиях и участие логического выражения в операторе цикла `while ... do`. Напоминаем, что для объединения нескольких простых

логических выражений в одно сложное необходимо использовать логические операции (`and, or, not`).

Таблица 8.39

Варианты решения комплексных заданий

Условия трёх заданий		
1	2	3
Вычислить значения X^2 для переменной X (–200≤X≥200), чтобы максимальное значение степени X не превышало 10^3	Вычислить значения X^2 для чётных значений переменной X на отрезке (–200≤X≥200), чтобы максимальное значение степени X не превышало 10^3	Вычислить значения X^2 для нечётных значений переменной X на отрезке (–200≤X≥200), чтобы максимальное значение степени X не превышало 10^3
Варианты решения задания 1,2 и 3 с циклом `while ... do`		

const tens = 1e3;
var
x , st : Longint;
begin
writeln(' Введите X=');
readln(x);
while ((x>=-200) and (x<=200)) and (x<=tens) do begin
st := sqr(x);
x := x +1; end;
end.
	const tens = 1e3;
	var
	x , st : Longint;
	begin
	writeln('Введите X=');
	readln(x);
	while ((x>=-200) and (x<=200)) and (x<=tens) do begin
	If not (odd(x)) then
	st := sqr(x);
	x := x +1; end;
	end.
		const tens = 1e3;
		var
		x , st : Longint;
		begin
		writeln(' Введите X=');
		readln(x);
		while ((x>=-200) and (x<=200)) and (x<=tens) do begin
		If (odd(x)) then
		st := sqr(x);
		x := x +1; end;
		end.

Обратите внимание на то, что если будут введены значения X < –200 или X > 200 цикл во всех трёх программах не будет выполнен ни разу.

Оператор цикла с послеусловием имеет следующий вид.

```
repeat
        <ОПЕРАТОР_1>;
        <ОПЕРАТОР_2>;
……..................
        <ОПЕРАТОР_N>
until   <УСЛОВИЕ>;
```

Где:

❶ `repeat` (повторять), `until` (до тех пор, пока...) – служебные слова;

❷ `ОПЕРАТОР_i` (i=1,2, ..., N) – любой оператор языка ТП;

❸ `УСЛОВИЕ` – логическое выражение, построенное по правилам языка ТП.

Принцип действия оператора repeat ... until подобен работе оператора while ... do, но в его работе имеется несколько важных отличий.

❶ Проверка значения логического выражения <УСЛОВИЕ> реализуется ***после однократного выполнения тела цикла***. То есть все операторы ОПЕРАТОР_1, ОПЕРАТОР_2 ... _N> будут выполнены хотя бы один раз при любых условиях.

❷ Операторы repeat ... until подобны операторным скобкам begin ... end, поэтому между ними можно размещать группы операторов, отделяя их между собой точкой с запятой.

❸ Цикл repeat ... until выполняется, в отличие от цикла while ... do, до тех пор, пока логическое выражение <УСЛОВИЕ> имеет значение FALSE и завершает свою работу когда оно принимает значение TRUE.

❹ После последнего оператора <ОПЕРАТОР_N> перед служебным словом until разделительный символ "**;**" точка с запятой ***не ставится***.

Приведём пример использования этого цикла (см. табл.8.40) для решения ранее рассмотренной задачи (см. табл. 8.39) вычисления значений функции X^2.

Таблица 8.40

Варианты решения комплексных заданий

Задания для решения		
1	2	3
Вычислить значения X^2 для переменной X (–200≤X≥200), чтобы максимальное значение степени X не превышало 10^3	Вычислить значения X^2 для чётных значений переменной X на отрезке (–200≤X≥200), чтобы максимальное значение степени X не превышало 10^3	Вычислить значения X^2 для нечётных значений переменной X на отрезке (–200≤X≥200), чтобы максимальное значение степени X не превышало 10^3
Варианты решения задания 1,2 и 3 с циклом repeat ... until		

const tens = 1e3;
var
x , st : Longint;
begin
writeln('Введите X=');
readln(x);
repeat
st := sqr(x);
x := x +1
until
not (((x>=-200) and (x<=200)) and (x<=tens))
end.
	const tens = 1e3;
	var
	x , st : Longint;
	begin
	writeln(' Введите X=');
	readln(x);
	repeat
	If not (odd(x)) then
	st := sqr(x);
	x := x +1
	until
	((x<=-200) or (x>=200)) and (x>=tens))
	end.
		const tens = 1e3;
		var
		x , st : Longint;
		begin
		writeln(' Введите X=');
		readln(x);
		repeat
		If (odd(x)) then
		st := sqr(x);
		x := x +1
		until
		not (((x>=-200) and (x<=200)) and (x<=tens))
		end.

Интересно проследить как работают все три операторы циклов (“с параметром”, “с предусловием” и “с послеусловием”) при вычислении одного и того же выражения.

К примеру, пусть необходимо вычислить следующую суму $\sum_{n=1}^{10} a^{\frac{1}{n}}$. Решение задачи будет состоять из следующих действий.

❶ На первом этапе необходимо ввести (т.е. инициализировать) значение постоянной величины *a*.

❷ Далее необходимо алгоритмически смоделировать изменение значения выражения $\frac{1}{n}$ на каждом новом шаге.

❸ На следующем этапе требуется смоделировать вычисление функции одной переменной $a^{\frac{1}{n}}$. Это, кстати, можно сделать с помощью тождества $a^k = e^{k \cdot \ln a}$, которое в языке ТП реализуется с помощью встроенных функций `exp`(x) и `ln`(x) в виде оператора $a^k = \exp(k \cdot \ln(a))$. В выражении $e^{k \cdot \ln a}$ значение k будет моделировать частное $\frac{1}{n}$. Значение элемента суммы для каждого значения переменной *n* будем присваивать переменной `elm`.

❹ Далее необходимо организовать циклический процесс вычисления вышеприведенной суммы из сконструированных компонентов.

Приведём алгоритм и его реализацию на языке ТП, который вычисляет сумму $\sum_{n=1}^{10} a^{\frac{1}{n}}$ в трёх разных циклах, с занесением результатов в переменные с именами SFOR, SWHILE, SREPEAT.

```
var
   n : integer;    a, elm, SFOR, SWHILE, SREPEAT, k : real;
Begin
     Writeln('Введём a='); readln(a);
{Cycle FOR ---------------------------------}
    SFOR := 0;
  for n := 1 to 10 do begin
    k := 1/n; elm := exp(k*ln(a)); SFOR := SFOR + elm; end;
{Cycle WHILE---------------------------------}
    n := 1; SWHILE := 0;
  while  n<= 10 do begin
      k := 1/n; elm := exp(k*ln(a)); n := n + 1;
    SWHILE := SWHILE + elm;
{Cycle REPEAT-------------------------------}
    n := 1; SREPEAT := 0;
  repeat k := 1/n; elm := exp(k*ln(a));
    SREPEAT := SREPEAT + elm; n := n + 1
until n > 10 ; {Можна так : not(n<=10)}
Writeln('Три Суммы =',SFOR:10:3,SWHILE:10:3,SREPEAT:10:3);
end.
```

Сформулируем вышеприведенную задачу в другом виде. Допустим, необходимо вычислить сумму $\sum_{n=1}^{\infty} a^{\frac{1}{n}}$ с точностью $\varepsilon = 0.01$.

Тогда цикл с параметром (`for … to … do`) для этого случая использовать уже в принципе невозможно, поскольку заранее неизвестно сколько раз понадобиться прибавлять очередные элементы к общей сумме, чтобы выполнить поставленное условие. Не лишне напомнить, что решение этой задачи невозможно непосредственно и в приложении MS Excel. Ведь и в таком мощном программном продукте необходимо будет написать специальную процедуру-подпрограмму на языке Visual Basic for Application с использованием подобных операторов цикла из арсенала данного языка.

Таким образом, на языке ТП решение новой задачи можно записать с использованием одного из двух операторов цикла с условиями:

`(while … do)` или `(repeat … until).`

Для демонстрации их возможностей используем в решении задачи и тот и другой. Точность эпсилон (ε) вычисления значения суммы в программе моделируем константой с именем `eps`, начальное значение которой задаём на уровне 0.01. Обратитет внимание, что основным требованием завершения циклов должно быть выполнение условия, чтобы значение очередного элемента суммы `elm` было меньше значения ε=0.01:

`elm < eps,`

Однако, Вы помните, что условием работы цикла `while ... do` является значение TRUE условного выражения его выполнения. Понятно, что значение первого элемента суммы не может быть меньше `eps`. Поэтому мы можем записать это условие с использованием логической операции инвертирования `(NOT)` логического значения.

```
Program Next_Cycles;
   const eps = 0.01;
var
   n : integer;
   k, a, SWHILE, SREPEAT : real;
Begin
     Writeln('Введите a='); readln(a);
{Cycle WHILE---------------------------------}
    n := 1; SWHILE := 0; elm := 1;
  while  not (elm < eps) do begin
    k := 1/n; elm := exp(k*ln(a)); n := n + 1;
    SWHILE := SWHILE + elm;
{Cycle REPEAT--------------------------------}
    n := 1; SREPEAT := 0;
  repeat k := 1/n; elm := exp(k*ln(a));
    SREPEAT := SREPEAT + elm; n := n + 1
  until  elm < eps ;
{--------------------------------------------}
    Writeln(Две суммы=',SWHILE:10:3,SREPEAT:10:3);
end.
```

Так как оператор цикла `repeat ... until` выполняет роль операторных скобок `Begin ... end`, после последнего оператора тела цикла перед служебным словом `until` ***разделительный знак*** `";"` ставить не нужно.

Оператор `while` используется чаще оператора `repeat`. Это связано с тем, что во многи практических случаях необходимо выполнять проверку на окончание цикла до его виыполнения и иметь возможность при необходимости вообще пропустить этот цикл.

Упражнения

1. Чем отличаются циклы с условиями от циклов с параметром?

2. Что собой представляет условие у этих обоих циклах?

3. Какое значенне должно иметь условие в цикле `while`, чтобы он выполнялся?

4. Какое значение должно иметь условие в цикле `repeat`, чтобы он выполнялся?

5. Чем вообще отличаетсяся цикл `while` от цикла `repeat`?

6. С использованием оператора цикла с предусловием `while` напишите и выполните программу табулирования функций *y* с действительным аргументом *x*. Границы изменения значений аргумента *x* взять на отрезке [–1, 1].

$$a)\ y = \frac{\sin 3\pi - x}{x + \sqrt{\left|\frac{3\pi}{2} + x\right|}};$$

$$b)\ y = \frac{x + ctg\ (x+1)}{\sin^2 (x+1)};$$

$$c)\ y = 10^{\log_{10}\left|x + \frac{a}{4}\right|};$$

$$d)\ y = \frac{\left|\sin^2 x^2 + \cos^2 x^2\right|}{1 + \dfrac{x^2 \cdot \sin^2 b \cdot x + 1}{x}}.$$

7. Выполните упражнения п.6. с использованием цикла `repeat`.

8. Найдите значение суммы в соответствии с формулой: $S = \sum_{n=1}^{\infty} \frac{1}{n^2}$.

8.18. Средства исследования выполнения действий программы с помощью дебаггера[24]

Если алгоритм программы достаточно сложный и Вы получаете непонятные или непредсказуемые результаты, или некоторые операции в программе останавливают работу компьютера (с диагностикой, например: деление на ноль, переполнение разрядной сетки ПК и т.д.), то Вы должны провести исследования программы средствами ИСР ТП. Для этого в Турбо Паскале есть достаточно развитые средства проверки состояния переменных, результатов операций с этими переменными и некоторых других операций, выполняемые с помощью встроенного дебаггера (отладчика). ***Но помните: дебаггер поможет Вам только в том случае, если в каждой строке программы будет находиться только по одному оператору ТП !***

Для построения простой исследовательской программы выполним в ИСР ТП команду **File / New**, которая откроет пустое окно редактора ТП с именем `NONAME00.PAS`. В этот чистый лист введём текст следующей программы для её исследования под названием (именем) `MyFirst`:

```
program MyFirst;
var
  A,B: Integer;
  Ratio: Real;
Begin
  repeat
    Write('Введите два числа A и B : ');
    Readln(A,B);
    Ratio := A/B;
    Writeln('Результат деления = ',Ratio:8:2);
    Write('Нажмите <Enter>...');
    Readln;
  until B = 0;
End.
```

Для сохранения текста Вашей программы, то есть записи её на диск с соответствующим именем, войдите в меню **File**, выберите **Save As** и наберите в строке с названием **Save file as** соответствующего меню имя Вашего файла `MyFirst`, а потом нажмите **Enter**. Расширение имени файла `.PAS` добавляется при записи текста программы на диск автоматически (рис. 8.35).

Компилируйте и запускайте свою программу, используя **Ctrl-F9**. Турбо Паскаль автоматически откомпилирует Вашу исследовательскую программу перед запуском, а затем и выполнит её.

Поскольку Ваши операторы были вложены в цикл `repeat ... until`, это приведёт к тому, что все они будут выполняться до тех пор, пока условное выражение, влияющее на количество выполнений циклов, не станет иметь значение `True` (истина). Это условное выражение вычисляется для проверки,

[24] Дебаггер (орладчик) – программа, помогающая находить и локализовывать ошибки программирования.

равно значение B нулю или нет. Если перемнная B примет значение 0, цикл должен завершиться.

Рис. 8.35. Вид программы MyFirst в синем окне редактора ТП

Вообще говоря, эта программа должна работать без остановки, так как условием выхода из цикла является фактически ошибочное значение переменной *B* = 0, что останавливает программу и вызывает диагностику:

```
Error 200: Division by zero
(ошибка 200: деление на 0)
```

Код нашей программы спроектирован так специально, чтобы показать Вам, как использовать средство под названием дебаггер (то есть искатель ошибок –"багов"), встроенный в интегрриованную среду Турбо Паскаль и который позволяет Вам перемещаться по своему коду по строкам, изучая работу программы. В то же время, Вы можете просматривать значения всех необходимых Вам переменных и исследовать как они меняются в ходе выполнения программы.

Чтобы начать сеанс исследования, выберите команду **Run/Trace Into** (или просто нажмите **F7**). Если Ваша программа имеет необходимость в перекомпиляции, Турбо Паскаль выполнит это. Первый оператор в разделе операторов Вашей программы (в данном случае – begin) будет высветлен специальной полосой; с этого момента мы будем называть эту светлую полосу – ***полосой запуска*** (рис. 8.36). Последующие нажатия клавиши **F7** приводят к перемещению курсора по строкам Вашей программы с последовательным выполнением находящихся в них операторов.

Вторым этапом исследования, является Ваше указание дебаггеру на имена переменных, изменения значений которых Вы хотите увидеть в этом сеансе работы данной программы. Для ввода опредённых программистом указаний в

ИСР ТП используется специальное диалоговое окно **Add Watch** (то есть добавление контрольных точек для наблюдения).

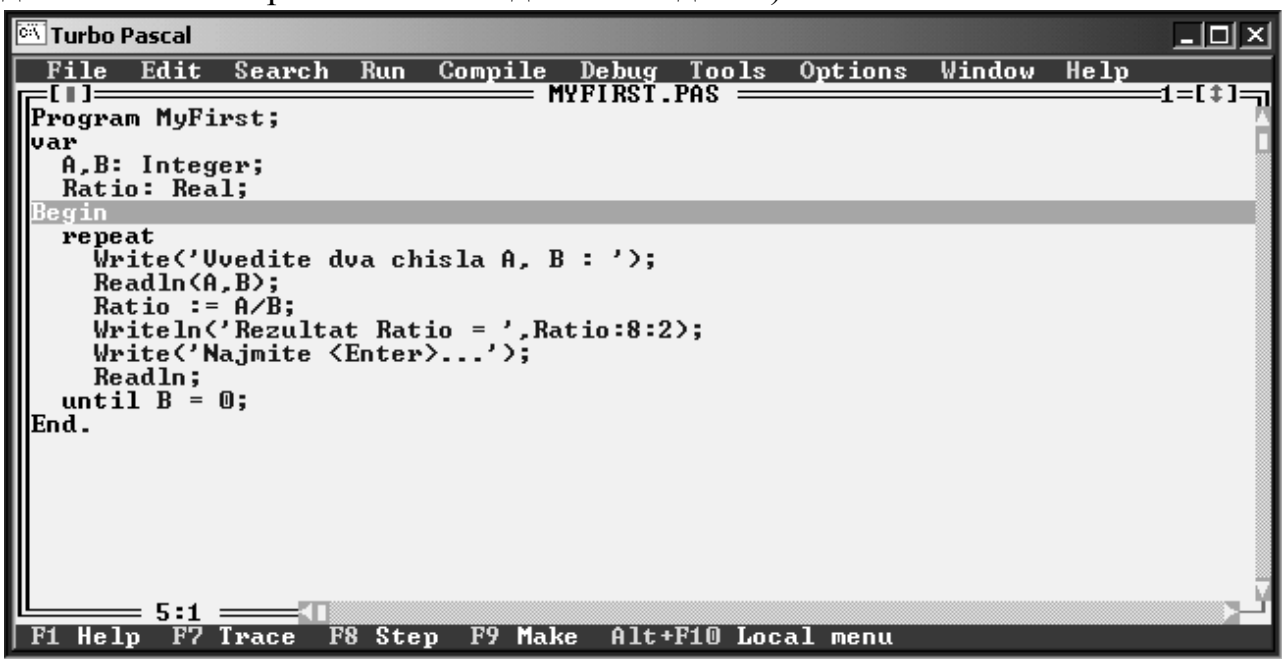


Рис. 8.36. Начало работы программы в режиме сеанса исследования

Для задания необходимых точек выполните команду **Debug/Add Watch** (рис. 8.37) и когда появится диалоговое окно с названием **Add Watch**, наберите имя первой необходимой для исследования в программе переменной, то есть **A**, и нажмите **Enter** (рис. 8.38). Напоминаем, что выбор команд можно выполнять или с помощью курсора мыши, или нажатием управляющих клавиш на клавиатуре ПК: в данном случае **Alt+D**, а потом **A**.

Рис. 8.37. Среда ТП перед выполнением команды **Debug/Add Watch**

Выполнение вышеуказанных действий приведёт к появлению добавочного окна с именем **Watches** в нижней части главного экрана, в котором появится имя указанной переменной и её текущее значение (рис. 8.39). ***Примите к сведению, что это дополнительное окно закрывается командой*** **Window / Close** (или – Alt+F3)***, а полоса запуска удаляется командой*** **Run / Program reset** (или – Ctrl+F2)**.**

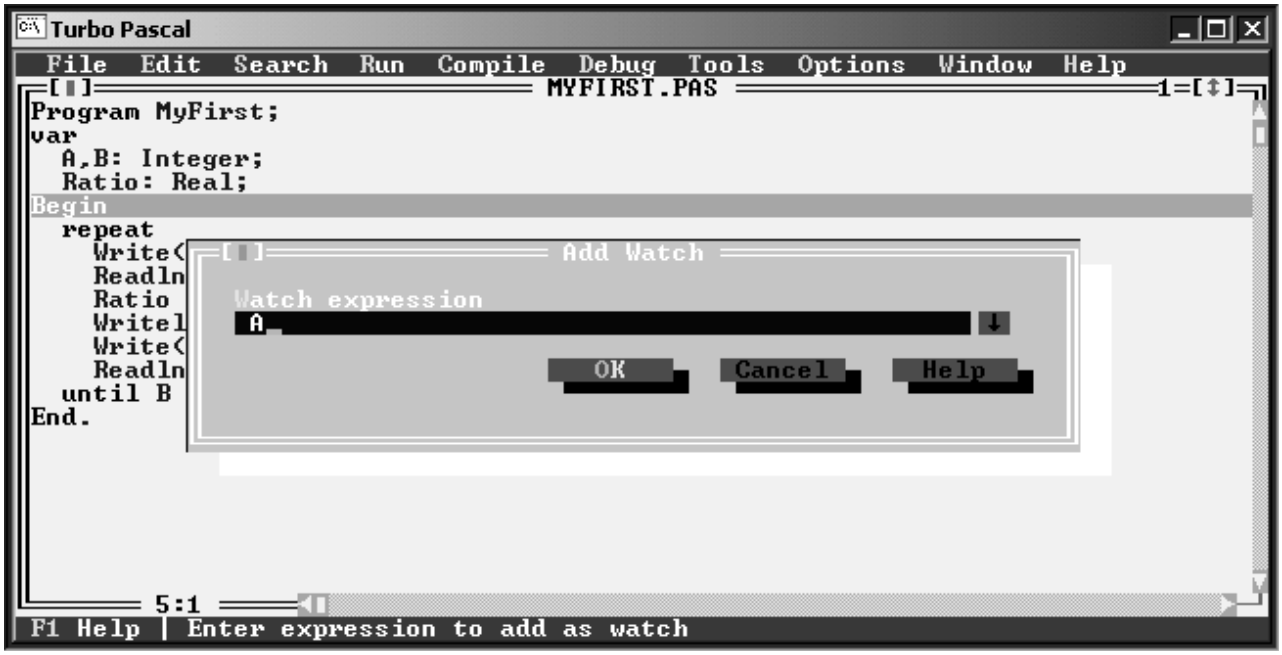


Рис. 8.38. Ввод имени переменной A, значение которой исследуется

Важно также помнить, что для добавления следующих имён переменных (B и Ratio) в окно **Watches** достаточно перед вводом их имён предварительно нажимать клавишу **Insert**, расположенную справа над клавишей **Delete**.

Рис. 8.39. Отображение значений переменных в окне **Watches** (Наблюдение)

После ввода очередного имени переменной в строку меню **Add Watch** и нажатия **Enter**, в окне **Watches** внизу экрана добавляется это имя с его текущим значением (рис. 8.39).

Теперь, когда все значения переменных нами выведены в окне, после очередного нажатия **F7,** программа будет переведена в чёрный экран DOS (т.е. экран результатов), потому что оператор `Readln(A,B)` ожидает ввода двух новых чисел. Наберите два целых числа, разделённые пробелом; и убедитесь, что второе число – не нуль (рис. 8.40).

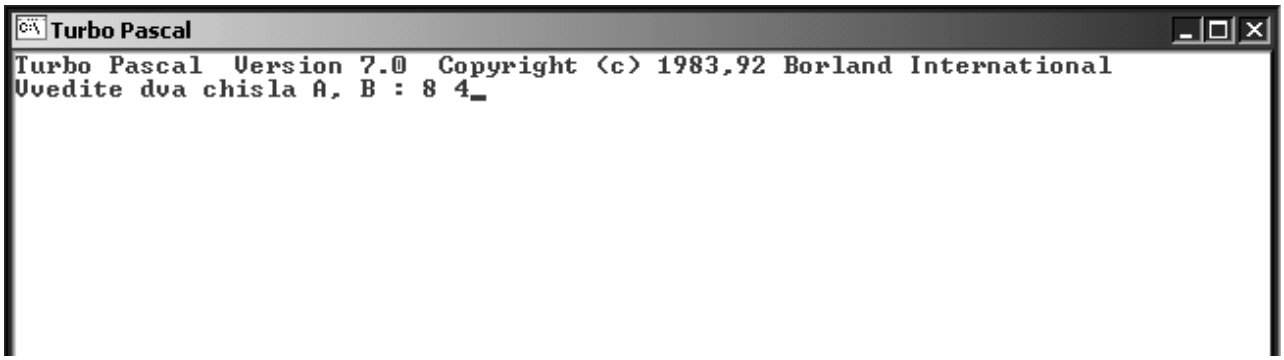


Рис. 8.40. Экран DOS при введении данных в переменные `A` и `B`

Теперь нажмите **Enter** и Вы возвратитесь назад в окно редактора с ***полосой запуска*** на операторе присваивания в строке 9 (`Ratio := A/B;`) программы `MyFirst` (см. рис. 8.36). Нажмите **F7,** программа выполнит этот оператор присваивания и полоса запуска переместится на оператор `Writeln` в строке 10. Одновременно значение переменной `Ratio` получит новое значение 8/4 = 2, инициированное введёными данными, объединёнными в выражение оператором деления (/) (рис. 8.41).

Рисунок 8.41. Значения переменной `Ratio` при введённых даных `A` и `B`

Нажмите **F7** дважды. Теперь выполнится оператор вызова процедуры `Readln` в строке 12. Ещё раз нажмите **F7**, посмотрите результат работы программы и нажмите **Enter**. Полоса запуска теперь находится на операторе `until`. Нажмите **F7** несколько раз и программа вернётся к началу цикла `repeat`, то есть к оператору `Write`, и т.д.

Таким образом, мощный инструмент – дебаггер – позволяет выполнять перемещения по коду программы "строка за строкой" и одновременно наблюдать за поведением Ваших переменных и выражений (табл. 8.41).

Таблица 8.41

Команды дебаггера и его действия

Название действия	**Команда ТП**	**Горячие клавиши**
Переход к режиму дебаггера	**Run/Trace Into**	**F7**
Перемещение по операторам программы	**–**	**F7**
Вызов окна **Watches** для просмотра значений переменных и выражений в пошаговом режиме	**Debug/Add Watch**	**Alt+D**, а потом **A**
Добавление в окно **Watches** имён переменных и выражений	**Debug/Add Watch**	**Insert (клавиша)**
Закрытие окна **Watches**	**Window/Close**	**Alt+F3**
Удаление полосы запуска	**Run/Program reset**	**Ctrl+F2**

Кроме значений отдельных переменних, в окне **Watch** можно также наблюдать значения выражений. Чтобы ввести необходимое для исследования выражение нажмите **Alt-D** для появления меню **Debug** (это альтернативная возможность вызова). Выберите команду **Add Watch** из меню Watches (или нажмите **Ctrl-F7**). Теперь можно набрать в окне ввода **Watch Expression** не только имена переменных `A, B` и `Ratio,` которые появятся в окне **Watch** вместе со своими текущими значениями, а также и выражение `A/B`.

Далее, выберите **Run/Trace Into** (или нажмите **F7**) для того, чтобы сделать шаг в своей программе. В этот раз, когда Вы должны ввести очередных два числа, введите `0` для второго из них. Когда Вы нажмёте **Enter** и вернётесь в IDE Турбо Паскаль, посмотрите на выражение `A/B` в окне **Watch** (для этого нажмите Alt и # (номер) окна или **Alt-W W**). Вместо этого значения будет стоять фраза `"Invalid floating-point operation"` (неправильная операция над числами с плавающей точкой). Это случилось потому, что операция деления на нуль неопределена. Хотя заметим, что наличие этого выражения в окне **Watch** не приводит к остановке программы с ошибкой. Вместо этого выводится сообщение об ошибке, а отладчик не выполняет операцию деления в окне **Watch**.

Тепер нажмите **F7** опять, присваивая значение выражения `A/B` переменной `Ratio`. В этом месте произойдёт аварийное завершение программы, и вверху окна редактора снова появится сообщение про ошибку `"Division by zero"`.

Тепер возможно Вы задумались про то, что же неправильно работает в вашей программе, если Вы вводите значенння 0 для второго числа (`B`) и программа завершается с ошибкой выполнения.

Как зафиксировать её? Очевидно, что если переменная `B` имеет значение `0`, не следует делить `A` на `B`. Отредактируйте Вашу программу так, чтобы она выглядела следующим образом:

```
   program MySecond;
     var
        A,B: Integer;  Ratio: Real;
   Begin  repeat
             Write('Enter two numbers: ');  Readln(A,B);
          if B = 0 then   Write('The ratio is undefined')
          else
             begin
            Ratio := A/B;  Writeln('The ratio is ',Ratio:8:2);
             end;
             Write('Press <Enter>...');  Readln;
       until B = 0;
End.
```

Теперь запустите свою программу (или используя дебаггер или без него). Если Вы используете дебаггер, обратите внимание, как меняются значения в окне **Watch** по мере выполнения шагов в программе. Когда Вы готовы завершить вычисления, введите 0 в перменную `B`. Программа остановится после вывода сообщения `"The ratio is undefined. Press <Enter>..."` ("Отношение не определено. Нажмите <**Enter**>...").

Теперь Вы можете оценить, каким мощным средством является дебаггер для исследования работы Вашей программы. С его помощью можно перемещаться в программе строка за строкой; можно рассматривать в окне **Watch** значения переменных и выражений Вашей программы, а также наблюдать изменения их значений по ходу выполнения программы.

Упражнения

1. Что такое дебаггер и для чего он предназначен?

2. Какие команды и горячие клавиши служат для вызова дебаггера и перехода в режим отладки программы пользователя?

3. Какие команды и горячие клавиши служат для перемещения по операторам программы в режиме её отладки?

4. Какие команды и горячие клавиши служат для вызова окна **Watches** для просмотра значений переменных и выражений в пошаговом режиме?

5. Какие команды и горячие клавиши служат для добавления в окно **Watches** имён переменных и выражений?

6. Какие команды и горячие клавиши служат для закрытия окна **Watches**?

7. Какие команды и горячие клавиши служат для удаления полосы?

8. Вычисление с помощью цикла с параметром `for` значения конечной суммы S для выражения:

$$S = \sum_{n=1}^{5} n^2 + \sum_{n=1}^{12} n^3 .$$

Проведите исследование работы программы с помощью дебаггера.

9. С использованием оператора цикла с предусловием `while` напишите и выполните программу табулирования значений функции *y* с действительным аргументом на отрезке [–2, 0]. Проведите исследование работы программы с помощью дебаггера.

$$y = \frac{\sin 3\pi - x}{x^{\frac{1}{3}} + \sqrt{\left|\frac{7\pi}{2} + x^2\right|}} .$$

10. С использованием оператора цикла с послеусловием `while` напишите и выполните программу табулирования значений функции *y* с действительным аргументом на отрезке [–1, 1]. Проведите исследование работы программы с помощью дебаггера.

$$y = \frac{\sqrt{\left|x^3\right|}}{4 - x} .$$

8.19. Моделирование в циклических вычислениях некоторых типовых выражений

В циклических процессах очень часто используются некоторые выражения, которые имеют стандартные алгоритмические реализации. К ним можно отнести следующие:

$$(-1)^n \qquad (8.8)$$

$$n! \qquad (8.9)$$

$$(2n)! = 2n! \qquad (8.10)$$

Все они зависят от элементов натурального ряда чисел:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, ... , *n*

Как правило, сокращённо, изменение значений натурального ряда на отрезке [*1*,*n*] записывают так:

1, 2, ... , *n*

В зависимости от изменения значений элементов ряда *n*, вышеуказанные выражения принимают соответствующие значения (табл. 8.42). При этом и ряд можно рассматривать как выражение, значение которого изменяется в соответствии с изменением номера элемента *n*.

Таблица 8.42

Изменения значений выражений в соответствии с изменением значений, которые подставляются в них

Выражение	Изменение значений выражения				
n	1	2	3	4	…
2n	2	4	6	8	…
$(-1)^n$	$(-1)^1 = -1$	$(-1)^2 = +1$	$(-1)^3 = -1$	$(-1)^4 = -1$	…
n !	1!=1	2!=1·2=2	3!=1·2·3=6	4!=1·2·3·4=24	…
$(2n)! = 2n!$	2!=2	4!=24	6!= 1·2·3·4·5·6= =720	8!=1·2·3·4·5·6·7·8= =40320	…

Значение выражения $(-1)^n$ можно моделировать двумя разными способами:

❶ на каждом шагу приформировывать унарный знак "–" к предыдущему значению переменной, сохраняющей значения (–1) или (+1);

❷ на каждом шагу умножать предыдущее значение (–1) или (+1) на (–1).

Вместе с тем, следует помнить, что для выполнения приформировывания знака или умножения на (–1) необходимо предварительно извлекать число из перменной на сумматор, проводить требуемые действия, а потом засылать их назад в переменную.

Первые пять строчек значений этих функций, которые будут выведены на экран программой и текст самой программы приведены на следующей странице.

n	$(-1)^n$	n!	2n	2n!
1	-1.0	1.0	2	2.0
2	1.0	2.0	4	24.0
3	-1.0	6.0	6	720.0
4	1.0	24.0	8	40320.0
5	-1.0	120.0	10	3628800.0

```
{Программа моделирования значений функций (8.8), (8.9), (8.10)}
  Var
       med1, med2, Factor1, Factor2 : Real;
           n, n2, I : Word;
  Begin  {Инициализируем начальные значения переменных}
    med1 := -1; {Имя переменной, принимающей значения (-1)}
      med2 := -1; {Имя переменной, принимающей значения (-1)}
  Factor1n := 1;  {Имя переменной, принимающей значения n!}
    Factor2n := 2;  {Имя переменной, принимающей значения 2n!}

  n := 1;  { Имя переменной, принимающей значения n}
    n2:= 2;  { Имя переменной, принимающей значения 2n}
  { Фактически присваиваем переменным значения выражений}
  { при n = 1, которое моделирует значения переменной n.}
  { Переменную 2n моделирует значение переменной n2.}
  {Ограничиваем действие цикла с предусловием значением n=40,}
  {что эквивалентно максимальному значению n1=40}.
  {}
              Writeln ('Значения переменных=', med1:8:1,
med2:8:1, n:3,

Factor1n:8:1, n2:3, Factor2n:8:1);
  while  n <= 40 do   Begin
    med1 := -med1;
       med2 := (-1)*med2;
          n:= n + 1;
             Factor1n := Factor1n * n;
                n2 := n2 + 2;
                   Factor2n := Factor2n * (n2-1) * (n2);
                       Writeln  ('Значения  переменных=',
med1:8:1, med2:8:1, n:3,

Factor1n:8:1, n2:5, Factor2n:11:1);
      end;
  end.
```

Очень важным моментом является процесс отображения циклического (то есть последовательного) добавления некоторых данных к значению определённой переменной. Допустим, необходимо вычислить выражение Y = A+5, при значении A=2.

На языке ТП это выглядит так:

```
A := 2;
Y := A + 5; {Стало Y = 7}
```

Однако, если нужно добавить к значению, которое находится в переменной Y ещё какие-либо значения (допустим 8 и 10), это будет выглядеть так:

```
Y := Y + 8;  { Становится Y = 15}
Y := Y + 10; { Становится Y = 25}
.......... {и так далее}.
```

Выражение `Y := Y + 10` для компьютера указывает, что необходимо:

❶ взять предыдущее значение, находившееся в переменной Y;

❷ сложить его с постоянной 10;

❸ заслать новое значение на старое место в переменную Y.

Другой пример. Допустим необходимо вычислить выражение:

$$f = \cos x + \cos y + \cos z.$$

На языке ТП оно может быть записано следующим образом:

```
f := cos (x) + cos (y) + cos (z).
```

Тогда встроенной функцией `cos(x)` будут вычислены три значения косинусов от соответствующих аргументов *x, y, z,* а потом, после суммирования этих трёх значений, сумма будет заслана в переменную *f* и заменит там предыдущее значение. Однако, если это же выражение вычислять в цикле, то необходимо предварительно очистить переменную *f*, а потом проводить прибавления на каждом шагу. Очистка содержимого переменной *f* выполняется засылкой в неё нулевого значения. Соответствующий код может выглядеть так:

```
.....................
f := 0;
f := f + cos (x);
f := f + cos (y);
f := f + cos (z);
.............   ....
```

В результате выполнения этой последовательности команд получим результат, соответствующий вычисленному в вышеприведенном выражении.

Упражнения

1. Вычислить с помощью цикла с параметром `for` значения конечной суммы S для выражения:

$$S = \sum_{n=1}^{5} (-1)^n n^2 + \sum_{n=1}^{12} (-1)^{3n} n^3 .$$

Проведите исследование работы программы с помощью дебаггера.

2. С использованием оператора цикла с предусловием `while` напишите и выполните программу табулирования значений функции *y* с действительным аргументом x = 0.25 на отрезке измененя *n* [1, 5]. Проведите исследование работы программы с помощью дебаггера.

$$y = \frac{\sin 3\pi - x}{n!}.$$

3. С использованием оператора цикла с послеусловием `repeat` напишите и выполните программу табулирования значений нижеприведенного выражения функции *y* с действительным аргументом на отрезке изменения *n* [1, 7] при *x*= –9. Проведите исследования работы программы с помощью дебаггера.

$$y = \frac{\sqrt{\left|x^3\right|}}{2n!}.$$

4. С использованием оператора цикла с предусловием `while` напишите и выполните программу табулирования значений нижеприведенных функций:

$$\operatorname{csc} h(x) = \frac{2}{e^x - e^{-x}} = \frac{1}{x} + \sum_{n=1}^{\infty} \frac{2(-1)^n (2^{2n} - 1)}{(2n)!} n^{\frac{6}{7}} x^{2n-1}$$

$$y = \sum_{n=1}^{\infty} \frac{x^{2n-n!}}{n!}$$

$$y = \sum_{n=1}^{\infty} (-1)^{2n} \frac{x^{2+n}}{n!+1}$$

$$\frac{1}{2} - \frac{\pi}{8} = \sum_{n=1}^{\infty} \frac{1}{(4n-1)(4n+1)}$$

8.20. Особенности вычисления бесконечных сумм. Организация итерационных процессов с помощью циклов `while` и `repeat`

Одним из наиболее распространённых алгоритмических процессов является математическое действие прибавления. В простых случаях сумма выглядит обычно так:

$$C = A + B.$$

Однако в более сложных случаях сумма выглядит совсем иначе, когда количество прибавляемых элементов заранее неизвестно, а результат стремится к некоторому пределу по какой либо формуле. Типовым примером итерационного циклического процесса может быть задача вычисления суммы бесконечного ряда. Понятие суммы неразрывно связано с понятием сходимости такого бесконечного ряда. Обычно, ряд значений $t_0, t_1, \ldots, t_n, \ldots$ называется сходящимся, если сумма $s_n = t_0 + t_1 + \ldots t_n$ его первых *(n+1)* элементов, которые суммируются при бесконечном возрастании *n,* стремится к некоторой границе *S*, которая и называется суммой ряда, то есть:

$$\lim_{n\to\infty} \sum_{i=0}^{n} t_i = S$$

Общий член t_n ряда, который сходится, при этом стремится к нулю, то есть:

$$\lim_{n\to\infty} t_n = 0; \quad \lim_{n\to\infty} (s_n - s_{n-1}) = 0.$$

Таким образом, последовательность $s_1, s_2, \ldots, s_n, \ldots$ является последовательностью значений, которые мы ищем и определяет следующие условия завершения суммирования:

$$\left| s_n - s_{n-1} \right| \le \varepsilon \quad \textit{или} \qquad \left| t_n \right| \le \varepsilon.$$

Как правило, суммой нескольких элементов является результат прибавления очередных значений к сумме всех предшествующих элементов из этой последовательности.

Математически это записывается с помощью знака суммирования ∑:

$$\sum_{n=1}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1\cdot 2} + \frac{1}{1\cdot 2\cdot 3} + \ldots + \frac{1}{1\cdot 2\cdot 3\cdot \ldots \cdot n} \qquad (8.11)$$

$$\sum_{n=1}^{\infty} \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} \qquad (8.12)$$

Несмотря на простоту записи этого процесса, имеются некоторые важные обстоятельства для более внимательного рассмотрения того, что происходит при суммировании элементов бесконечных рядов.

Так, как операции прибавления значений в таких суммах в математическом понимании нужно выполнять до бесконечности (∞), следует понимать, что в компьютере такие большие значения, конечно же, обрабатываться не могут. С другой стороны, как правило, необходимо вычислять конечную сумму с некоторой точностью. Рассмотрим, что в этом случае имеется в виду.

Когда выполнена определённая часть вычислений, на некотором *k*-ом шаге уже имеется накопленная сумма (*k-1*) елементов и очередной *k*-й элемент суммирования (рис. 8.42), который должен быть прибавлен к общей сумме.

$$\left\{ \sum_{n=1}^{k-1} = (a_1 + a_2 + \ldots + a_{k-1}) \right\} + a_k$$

Рис. 8.42. Схема суммирования при вычислениях бесконечных сумм

Очевидно, что очередной элемент ряда сделает тем меньший вклад в общую сумму, чем меньше его значение. При этом, если его величина окажется меньше некоторой достаточно малой постоянной величины (константы) (например, наперёд заданой величины $\varepsilon = 0.0000001$ или какой либо другой), он должен перестать вносить соответствующий вклад в общую сумму и, соответственно, оказывать влияние на те цифры суммы, которые определяют её точное значение. Однако, как свидетельствует опыт многочисленных вычислений, существует ещё ряд факторов, которые влияют на процесс разработки подобных алгоритмов. Посмотрим внимательнее на результаты вычисления сумм, представленных формулами (8.11) и (8.12). Оказывается, что точность (т.е. близость к точному значению) текущей суммы зависит ещё и от того, насколько быстро увеличивается знаменатель дроби, которая вычисляется. К примеру, для формулы (8.11) значения компонентов будут изменяться так (табл.8.43).

Таблица 8.43

Изменение значений компонентов формулы (8.11)

Значение *n*	Значения элемента $\frac{1}{n!}$	Значения суммы *K* элементов $\sum_{n=1}^{K}\frac{1}{n!}$
1	1.000000000	1.0
2	0.500000000	1.5
3	0.166666666	1.6(6)
. . . . .		
10	0.000000275	1.718281801
11	0.000000025	1.718281826
12	0.000000002	1.718281828
Примечание: *K* – количество просуммированых элементов *n*		

Видим, что знаменатель дроби, принимающей участие в выполнении необходимых действий растёт очень быстро и процесс суммирования сходится за несколько десятков шагов. Другая картина наблюдается при вычислении формулы (8.12) (см. табл. 8.44):

Таблиця 8.44

Значения элементов ряда (8.12)

Значения n	Значения $\frac{1}{n}$	Значения $\sum_{n=1}^{K}\frac{1}{n}$
1	1	1.0000000
2	0.5	1.5000000
3	0.3(3)	1.8333(3)
.....		
10	0.1000000	2.9289683
.....		
1350	0.0007407	7.7854450
.....		
10224	0.0000978	9.8097577
.....		
20000	0.0000500	10.4807282
.....		
100000	0.0000100	13.0901461
.....		
1000000	0.0000010	15.3927267
.....		
5000000	0.0000002	17.0021642
.....		
10 000 000	0.0000001	17.6953113

Из таблицы 8.44 хорошо видно, что когда элемент, который прибавляется, уменьшается очень медленно, предусмотреть его вклад в общую сумму достаточно трудно. Поэтому Вы должны иметь ввиду несколько вариантов алгоритмов, которые могут дать возможность получить необходимый результат в разных случаях или использовать дебаггер для контроля точности результатов вычислений.

Посмотрим внимательнее на таблицы 8.43 и 8.44. Очевидно, что для успешного завершения суммирования элементов первой суммы, достаточно проверить условие, чтобы очередной элемент ряда был меньше некоторого малого числа ε (епсилон), которое задаёт сам программист (пользователь). Величина ε и будет контролировать точность конечной суммы. Другими словами, (в соответствии с данными таблицы 8.45), если мы сравним значения очередного элемента $\frac{1}{n!}$ с ε = 1e-6 (так записывается в ТП число 0.000001), то получим значения суммы с точностью приблизительно 6 знаков после точки. При сравнении значения этих же елементов с ε= 1e-7, точность увеличивается, и так далее (табл. 8.45).

Таблица 8.45

Зависимость значения конечной суммы от ε

Значения ε, задаваемые пользователем	Конечное значение параметра n	Значения последнего элемента $\frac{1}{n!}$ в сумме	Значения конечной суммы $\sum_{n=1}^{K}\frac{1}{n!}$
.....			
1e-6	10	0.000000275	1.718281801
1e-7	11	0.000000025	1.718281826
1e-8	12	0.000000002	1.718281828
.....			

Реализация алгоритма итерационного вычисления ряда $\sum_{n=1}^{\infty}\frac{1}{n!}$ с таким подходом может выглядеть так:

```
Uses Crt;
Const Eps = 1.e-7;  {Задаём точность вычисления ε=0.0000001}
Var
Sum, Elm, Fact  : Real; {Объявляем вещественные переменные}
         n : Integer; { Объявляем переменную для вычисления n}
Begin  ClrScr; {Очищаем экран процедурой ClrScr}
   n := 1;      {Инициализируем n}
    Sum := 0; {Обнуляем переменную под значение суммы}
       Elm := 1; {Инициализируем Elm  для входа в цикл}
         Fact := 1; {Инициализируем начальное значение факториала}
While  Elm > Eps Do
   Begin
Fact := Fact * n; {Моделируем значение факториала n!}
    Elm := 1/ Fact; { Моделируем значения элемента 1/n!}
        Sum := Sum + Elm; { Моделируем сумму сумма 1/n!}
           n := n + 1; { Моделируем переменное значение индекса n}
   End;
Writeln('Результат= ', Sum); {Выводим на экран результат}
End.
```

Упражнения

1. Вычислить приближённое значение бесконечной суммы S с точностью до ε=0.05. $S = \frac{1}{1\cdot 4}+\frac{1}{4\cdot 7}+\cdots+\frac{1}{(3n-2)(3n+1)}+\cdots$

2. Вычислить приближённое значение бесконечной суммы S с точностью ε=0.005. $S = x-\frac{x^3}{3!}+\frac{x^5}{5!}-\cdots+(-1)^n\frac{x^{2n+1}}{(2n+1)!}\pm\cdots$

3. Вычислить приближённое значение бесконечной суммы S с точностью ε=0.00001. $S = x - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots + (-1)^n \frac{x^{2n}}{(2n)!} \pm \cdots$

4. Вычислить приближённое значение бесконечной суммы S с точностью ε=0.00000001. $S = \frac{x-1}{x} + \frac{(x-1)^2}{2x^2} + \cdots + \frac{(x-1)^n}{nx^n} + \cdots$

5. Вычислить сумму с точностью до ε=0.00000001. $y = \sum_{n=1}^{\infty} \frac{(x+n)(x-n)}{2n!}$

6. Вычислить следующие суммы с определённой заранее точностью:

$$\frac{1}{2} - \frac{\pi}{8} = \sum_{n=1}^{\infty} \frac{1}{(4n-1)(4n+1)}$$

$$\frac{1}{2} = \sum_{n=1}^{\infty} \frac{1}{(2n-1)(2n+1)}$$

$$\sin x = \sum_{n=1}^{\infty} (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!}$$

$$(1 \pm x)^{-\frac{5}{2}} = 1 \mp \frac{5}{2}x + \frac{5 \cdot 7}{2 \cdot 4}x^2 \mp \frac{5 \cdot 7 \cdot 9}{2 \cdot 4 \cdot 6}x^3 + \frac{5 \cdot 7 \cdot 9 \cdot 11}{2 \cdot 4 \cdot 6 \cdot 8}x^4 \mp \ldots$$

7. Вычислить следующие суммы с определённой точностью (при x=m, где m–номер в журнале).

$$(1-x)^{-m} = 1 + \sum_{n=1}^{\infty} \frac{m(m+1)\ldots(m+n-1)}{n!} x^n$$

$$(1+x)^{-m} = 1 + \sum_{n=1}^{\infty} (-1)^n \frac{m(m+1)\ldots(m+n-1)}{n!} x^n$$

8. Вычислить ряд.

$$y(x) = \frac{1}{x} + \sum_{n=1}^{\infty} \frac{2 \cdot (-1)^n \left(2^{2n-1} - 1\right)}{(2n)!} n^{\frac{6}{7}} x^{2n-1}$$

8.21. Бесконечные произведения и их вычисления

Одним из достаточно рспространённых, является также процесс умножения одного числа на другое. К примеру:

$$C := A \cdot B;$$

Однако, когда количество елементов, принимающих участие в процессе умножения, начинает возростать, вычисления такого результата усложняется.

Циклическим умножением называется процесс умножения последовательности некоторых величин на число, отображающее результат умножения всех ранее перемноженных чисел.

В математических выражениях оно имеет обозначение **П**, а его действие можно отобразить на графичекой схеме следующим образом (рис. 8.43).

$$\left\{ \prod_{n=1}^{k-1} = (a_1 \times a_2 \times \ldots \times a_{k-1}) \right\} \times a_k$$

Рис. 8.42. Схема вычисления бесконечных произведений

Рассмотрим одну из формул (8.13), описывающих процесс умножения. Если начать подставлять в неё значения n = 1, 2, 3, 4, 5, … , ∞, то элементы этой последовательности сомножителей будут выглядеть так:

$$\prod_{n=1}^{\infty} \frac{1}{\left(\frac{1}{n}+1\right)} = \frac{1}{2} \cdot \frac{1}{\left(\frac{1}{2}+1\right)} \cdot \frac{1}{\left(\frac{1}{3}+1\right)} \cdot \ldots \frac{1}{\left(\frac{1}{n}+1\right)} \ldots \qquad (8.13)$$

А результаты вычисления значений этой формулы при изменениях *n*, моделирующих *K*, приведены в таблице 8.46.

Процесс вычисления результата перемножения *n* елементов, приведенный в данной таблице, ограничен величиной ε= 1e–7. Поэтому он закончился на значении *K* = 3162. Следует особо отметить, что подход к выбору проверки условия завершения вышеприведенного итерационного процесса существенно отличается от того, который рассматривался при вычислении формул суммирования. Это связано с тем, что значения очередного элемента произведения влияют на конечный результат не пропорционально какой либо постоянной величине. Поэтому в алгоритме программы вычисления произведения был использован подход, завершающий процесс вычисления в том случае, ***когда разница между предыдущим и текущим значениями результата умножения становится меньше ε = 1e-7*** (рис. 8.44). Тогда можно считать, что дальнейшие действия не приведут к изменению цифр значения произведения в первых 6-ти позициях (в данном случае, при ε = 1e–7) .

Таблица 8.46

Вычисление результатов умножения по формуле (8.13)

Значения n	Значения $\frac{1}{\left(\frac{1}{n}+1\right)}$	Значения формулы $\prod_{n=1}^{K}\frac{1}{\left(\frac{1}{n}+1\right)}$
1	2	3
1	0.5	0.5
2	0.6(6)	0.33333333
3	0.75	0.25
4	0.8	0.2
5	0.83(3)	0.16666666
6	0.85714285	0.14285714
7	0.875	0.125
8	0.88888889	0.11111111
9	0.9	0.1
10	0.90909090	0.09090909
11	0.91666666	0.08333333
,,,		
51	0.98076923	0.01923007
,,,		
100	0.99009900	0.00990099
,,,		
160	0.99378882	0.00621118
,,,		
400	0.99750623	0.00249376
,,,		
1400	0.99928622	0.00071377
,,,		
3161	0.99968374	0.00031625
3162	0.99968384	0.00031615

Следует отметить также, что значения конечного произведения в формуле умножения (столбец 2, табл.8.46), не уменьшаются, как в процессе суммирования, а увеличивается.

$$\underbrace{(\mathbf{a_1 \cdot a_2 \cdot \ldots \cdot a_{k-1}})}_{\prod_{n=1}^{k-1}} \; - \; \underbrace{(\mathbf{a_1 \cdot a_2 \cdot \ldots \cdot a_{k-1} \cdot a_k})}_{\prod_{n=1}^{k}} \; < \; \varepsilon$$

Рис. 8.44. Схема сравнения элементов процесса умножения для его своевременного завершения

Алгоритм вычисления произведений значений по формуле $\prod_{n=1}^{\infty}\frac{1}{\left(\frac{1}{n}+1\right)}$ можно записать так:

```
Uses Crt;
Const Eps = 1.e-7;  {Задаём точность вычисления: ε=0.0000001}
Var
  Mun, Mun_1, Mun_2, Elm, Chis, Znam : Real;{Объявляем
                                                переменные}
       n : Integer; { Объявляем переменную для вычисления n}
Begin      ClrScr; {Очищаем экран процедурой ClrScr}
   n := 1;      {Инициализируем n}
    Mun := 1; {Заносим единицу в переменную Mun для умножения на
                                                неё}
     Mun_1 := 1;{Заносим константу 1 в Mun_1  для выполнения
                        требований выполнения цикла (т.е. входа в него}
       Mun_2 := 0.5;{Значение первого элемента умножения,
                                        вычисленное пользователем}
While  abs(Mun_1 – Mun_2) > Eps Do
  Begin
    Mun_1 := Mun_2; {Запоминание предыдущего значения произведения,
                                    для последующего сравнения с текущим}
      n := n + 1; { Моделируем изменение значений индекса n}
       Chis := 1; { Моделируем значение числителя дроби}
        Znam := 1+1/n; { Моделируем значение знаменателя дроби}
       Elm := Chis/Znam; {Вычисляем значение очередного
                                элемента по формуле 1/(1/n+1)}
   Mun := Mun * Elm; {Вычисляем произведение}
     Mun_2 := Mun; {Запоминаем значение текущего произведения}
   End;
Writeln('Результат= ', Mun); {Выводим на экран результат}
End.
```

Упражнения

1. Вычислить приближённое значение бесконечного произведения с точностью до ε=0.0001 для следующих выражений.

$$\prod_{n=2}^{\infty}\left(1-\frac{1}{n^2}\right) \qquad \prod_{n=1}^{\infty}\frac{4n^2}{4n^2-1}$$

8.22. Подпрограммы: процедуры и функции. Формальные и фактические параметры. Передача параметров “по значению” и “по адресу” (по ссылке)

Очень часто в программах пользователя необходимо неоднократно вызывать некоторые заранее определённые последовательности операторов для выполнения обработки сложных последовательностей данных. Это можно выполнять с помощью трёх типов циклов, описанных в предыдущих разделах книги.

Однако в программировании, чрезвычайно широко распространена другая форма повторения последовательностей действий, которые часто используются. В алгоритмах такого рода в разных местах программы встречаются одинаковые фрагменты последовательностей операторов, которые отличаются только по значениями исходных данных (к примеру, вычисление тригонометрических функций *sin x, cos x,* численное решение систем линейных алгебраических уравнений и т.д.). При разработке программы по таким алгоритмам необходимо задавать одну и ту же группу операторов, отвечающих каждому из фрагментов, которые повторяются. Для повышения эффективности программирования в языке ТП введено понятие ***подпрограммы***. Группа операторов, которая будет повторно использоваться, оформляется в виде самостоятельной программной единицы – подпрограммы. Она записывается однократо в разделе описаний основной программы, а в соответствующих местах просто ***вызывается***. При вызове указывается ***имя подпрограммы и*** сопутствующие этому вызову, так называемуе, ***фактические параметры.***

Таким образом, ***подпрограммой*** называется логически завершённая последовательность (группа) операторов, выполняющих определённую последовательность действий, которую можно вызвать из разных мест программы, называемой ***главной*** или ***вызывающей***. Механизм подпрограмм допускает достаточно большое количество уровней вложенности. Это означает, что подпрограмма, которую вызывает главная программа, может в свою очередь вызывать другие подпрограммы, которые также вызывают какие либо необходимые подпрограммы и так далее (рис. 8.45). Существуют два вида подпрограмм.

❶ ***Подпрограмма–функция***, результатом работы которой является одно значение любого типа (действительного, целого, логического или др.).

❷ ***Подпрограмма–процедура***, результатом работы которой может быть несколько значений разных типов или последовательности наборов данных разных типов (действительных, целых, массивов, записей и др.).

Поэтому отличаются и формы их вызовов. Имя функции используется в выражениях в виде ***операнда, то есть элемента выражения*** (к примеру, F:=5*cos(x). Здесь функция– cos(x)). Наоборот, вызов процедуры всегда является ***оператором*** (к примеру, …readln(f,d,t)**;**…). Обратите внимание на то, что вызов процедуры readln завершается так же, как и всякий оператор ТП разделителем "**;**"точкой с запятой. Однако, так как общие механизмы их работы

практически одинаковы, здесь и далее под подпрограммами будем понимать и процедуры и функции.

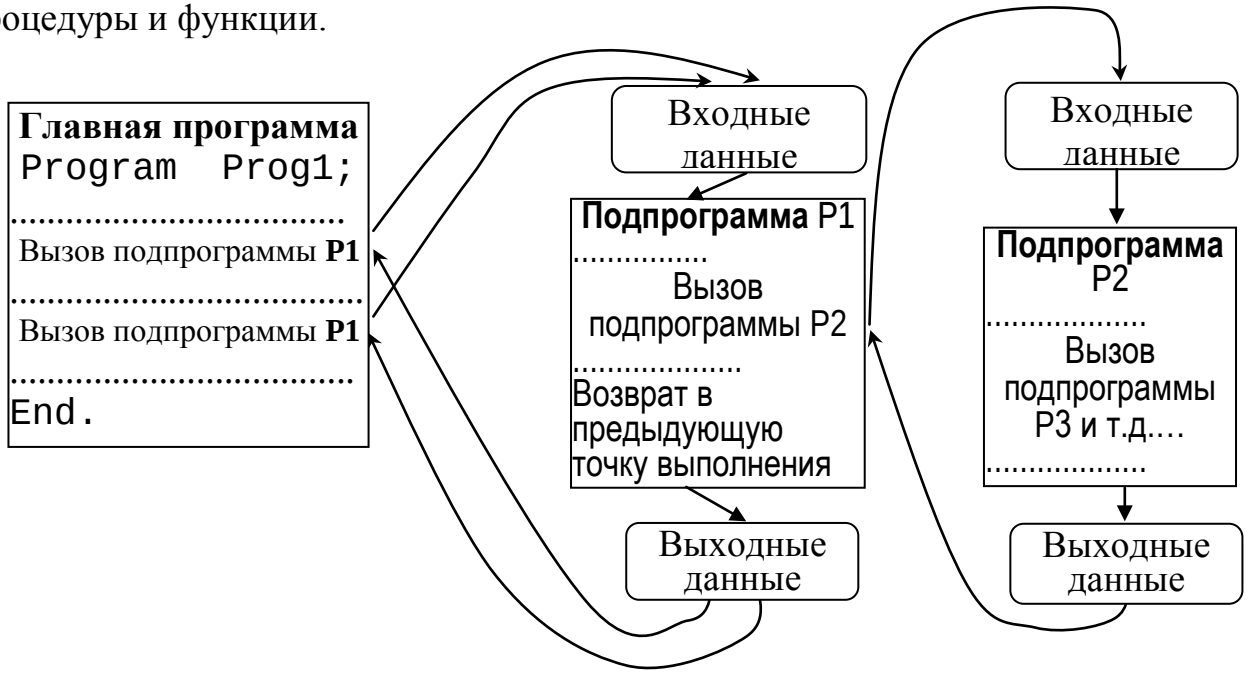


Рис. 8.45. Взаимодействие главной программы с вызываемыми подпрограммами

Процедуры и функции описываются в специальном разделе описательной части главной программы следом за описанием её констант и переменных. Описание каждой из этих подпрограмм начинается с заголовка, который содержит название соответствующего блока операторов (`Procedure` или `Function`), ***имя*** этого блока, ***список формальных параметров***, а функция ещё дополнительно объявление ***типа результата***, который она возвращает в вызывающую программу (рис. 8.46).

1

```
Procedure S1(a, b, c, …: real, i, j, k, … : integer;
                                  var S, T, …:real);
begin
    <Операторы процедуры> ……..
    S := 3.141592653 * a * b;   T := 2 * S;
{Процедура возвращает значения выражений через параметры S и T}
    <Операторы>
end;
```

2

```
Function S2(a, b, c, …:real, i, j, k, … : integer):
real;
Begin
    <Операторы>
 S2 := 3.141592653 * R * R;
{Функция возвращает результат через имя S2, посредством
использования механизма оператора присваивания,
оставляя при этом результат на сумматоре ПК}
end;
```

Рис. 8.46. Схематическое представление описаний процедуры (1) и функции (2) в вызывающей программе ТП

Заголовки подпрограмм состоят из:

❶ Определённых зарезервованных слов (`Procedure` или `Function`).

❷ Имён этих подпрограмм, задаваемых в соответствии с общими правилами построения идентификаторов.

❸ Параметров, представляющих собой перечисление имён объектов для обозначения начальных (входных) данных и результатов их работы с обязательным описанием типов этих данных. Эти параметры называются ***формальными.***

Заголовок подпрограммы-функции завершается описанием типа результата, который она возвращает в вызывающую программу.

В языке ТП можно описывать подпрограммы, которые не содержат перечисление формальных параметров (рис. 8.47).

```
Procedure S1;
 begin
.....................................
          <Операторы>
 end;
```

```
Function S2: real;
 begin
.....................................
 <Операторы> S2:= <число>
 end;
```

Рис. 8.47. Описание процедур и функций,
не содержащих формальные параметры

Функция отличается от процедуры также и тем, что в её конце обязательно должен находится оператор присваивания вычисленного результата имени этой функции (рис.8.48, строки 4 и 6).

```
Program  Sfunc;
Var
     x, y, z : real;
Function Sqrt_11(elm :Real):Real; {Ф-ция вычисляет корень второй степени}
  Begin
     Sqrt_11 := sqrt(elm);  {Тут elm – формальный параметр}
  End;
Begin
   X := 24.68;
  Y := Sqrt_11(X); {Тут X – фактический параметр, то есть тот, который
 имеет численное значениет и передается из программы в подпрограмму-
 функцию}
   Writeln('Значення Y=', Y);
End.
```

Рис. 8.48. Присваивание значения действия функции (**`Sqrt_11`**) её имени

Смысловая часть подпрограмм представляет собой ***блок*** и состоит из:

❶ Раздела описаний (меток, констант, типов, переменных, процедур, функций).

❷ Раздела операторов, представляющих собои составной оператор (`begin … end`).

❸ Блок подпрограмм заканчивается точкой с запятой.

Практически всегда относительно простые выражения можно реализовать и с помощью процедур, и с помощью функций. Компоновка использования переменных может быть разнообразной. Это можна увидеть на примере демонстрационной программы GLAVN, в которой показано взаимодействие формальных, фактических, глобальных и локальных параметров (рис. 8.49).

```
Program GLAVN;
Const GLOB1 = 5.0;
Var
      GLOB2, FACT1, FACT2, V,S : Real;
Procedure Demo1(FORM1 : Real; Var FORM2 : real);{Первая процедура}
  Const Lock1 = 3.141592653; {В локальной константе Lock1– значение π}
  Var
      Lock2, Lock3 : real;
Begin
   {Процедура Demo1 вычисляет значение площади круга S = π·R², а потом объём}
   {цилиндра по формуле V = H * S = H * π * R². Значения (аргумента)  }
   { R передаются через формальный параметр FORM1. Вычисленное значение }
   { V возвращается через фактический параметр FACT, который подставляется на}
   {место формального параметра FORM2, а значения площади S передаём}
    {через глобальную переменную GLOB2. Высота цилиндра h находится в }
     {глобальной константе GLOB1.  Lock2, Lock3 – локальные переменные.}
LOCK2:=LOCK1 * SQR(FORM1); {S засылаем в локальную переменную LOCK2}
   GLOB2 := LOCK2;
     FORM2 := GLOB1 * LOCK2;
end; {Proc Demo1}
{-------------------------------------------------------------------------}
Procedure Demo2;{Вторая процедура, работающая без формальных параметров}
Begin S := PI * SQR(FACT1);
      v := GLOB1 * S; end; {Proc Demo2}
Begin {Операторна частина програми}
  Writeln('Введите радиус основания цилиндра в переменную FACT1:');
   Readln(FACT1); {Значение радиуса R введено в переменную FACT1}
    DEMO1(FACT1, FACT2);
  Writeln('Знач.R',R,FACT1,'Знач.S=',GLOB2, 'Знач.V=', V);
      DEMO2;
  Writeln('Знач.R',R,FACT1,'Знач.S=',GLOB2,'Знач.V=', V);
end.
```

Рис. 8.49. Пример использования процедур с параметрами и без параметров, а также разных типов параметров и переменных

Следует различать переменные и константы, описанные в главной программе и в подпрограммах. Те переменные и константы, которые описаны в главной программе называются ***глобальными*** и их так называемая ***"область видимости"***, то есть пространство, где их можно использовать, распространяется на все процедуры и функции, описанные в этой программе.

Кроме этого, их значения можно использовать, передавая через фактические параметры, или употребляя в выражениях подпрограмм непосредственно.

Переменные и константы, описанные в подпрограммах можно употреблять только в них самих и поэтому они называются ***локальными***.

Следует знать и понимать, что имена глобальных переменных и констант могут совпадать с именами:

❶ Формальных параметров подпрограмм.

❷ Локальных параметров в подпрограммах.

Однако механизмы языка ТП и компилятор следят за этими ситуациями и делают так, чтобы в программах все эти объекты рассматривались и обрабатывались как абсолютно разные. То есть их значения никогда не пересекаются. Вместе с тем, глобальные параметры с именами, совпадающими с такими же именами в подпрограммах употреблять вместе с ними ***нельзя*** (рис. 8.50).

```
Program GLAVN;
Const GLOB1 = 5.0;
Var
         GLOB2, FACT1, FACT2, V,S : Real;
     Procedure Demo1(FORM1 : Real; Var FORM2);
               Const GLOB1 = 3.141592653;
                  Var
                     FACT1, Lock3 : real;
     Begin {Это разные объекты с разным содержанием (значениями)}
                    FACT1 := GLOB1 * SQR(FORM1);
                    GLOB2 := FACT1 * GLOB1;
    end;
Begin
……..
End.
```

Рис. 8.50. Пересечение имён глобальных и локальных переменных

Данный фрагмент программы демонстрирует, что переменные **GLOB1** и **FACT1,** описанные в качестве локальных параметров процедуры Demo1 имеют значения, отличные от глобальных переменных с такими же именами **GLOB1** и **FACT1** в главной программе!

А при передаче значений в подпрограммы через фактические параметры существует два основных механизма их дальнейшего использования: ***“по значению”*** и ***“по адресу”.***

Передача параметров "по значению" используется в тех случаях, когда нет необходимости с их помощью возвращать в главную программу какие-либо данные.

Вообще, при написании подпрограмм, необходимо иметь ввиду, что для них, формальные параметры (например **FORM1** или **FORM2** на рис. 8.51) являются просто "пустышками", замещаемыми при вызове конкретными значениями, которые можно потом использовать в необходимых действиях в операторах главной программы (рис. 8.51).

```
Var
            FACT1, FACT2  : real;
 Procedure Demo1(FORM1 : Real; Var FORM2 : Real);
   Const Lock1 = 3.141592653;
     Var
         Lock2, Lock3 : real;
 Begin
    LOCK2 := LOCK1 * SQR(FORM1);
      GLOB2 := LOCK2;
         FORM2 := GLOB1 * LOCK2;
 end;
Begin
……..
     FACT1 := 2.3;
         Demo1 (FACT1, FACT2);
             FACT1 := FACT1 / FACT2;
……..
End.
```

Рис. 8.51. Два типа передачи формальных параметров:
–"по значению" – формальный параметр **FORM1**
–"по адресу" – формальный параметр **FORM2**

Таким образом объект или так называемая “пустышка” FORM1 зовётся формальным параметром подпрограммы Demo1. А переменная FACT1, передающая конкретное значение (2.3) из тела главной программы и подставляемая при вызове процедуры Demo1 на место формального параметра FORM1 зовётся фактическим параметром.

FORM1 обеспечивает при вызовах процедуры передачу фактических параметров "по значению". Это означает, что значения аргумента (то есть фактического параметра) просто назначаются соответствующему формальному параметру. Другими словами, перед началом работы процедуры вычисляется

конкретное значение фактического параметра и он может передаваться в подпрограмму следующим образом:

❶ В виде значения глобальной переменной (например – GLOB1), которая передаётся в процедуру PP2 при её вызове через формальный параметр FR (рис. 8.52).

```
Var
      GLOB1, GLOB2 : Real;


 Procedure PP2 (FR : Real);
 Begin
     GLOB2 := FR * 5.62e-4;
 end;
Begin
     GLOB1 := 3.14 *2.73;
       PP2 (GLOB1);
End.
```

Рис. 8.52. Фактический параметр – глобальная переменная

❷ В виде некоторого значения (числовой константы) (рис. 8.53):

```
 Var
    GLOB1, GLOB2 : Real;

    Procedure PP3 (FR : Real);
      Var
        Lock1 : Real;
  Begin
               GLOB2 := 25 *FR; {GLOB2    буде мати
значення 25 * 23.2}
  end;
 Begin
         PP3 (23.2);
 End.
```

Рис. 8.53. Фактический параметр – константа

❸ В виде выражения, вычисляемого до передачи в подпрограмму (рис. 8.54):

```
Var
     GLOB1, GLOB2 : Real;
 Procedure PP4 (FR : Real);
         Var
            Lock1 : Real;
  Begin
              GLOB2 := 29.31 * FR;
  end;
Begin
     PP4(cos(0.52)*3.14/exp(3.0));
End.
```

Рис. 8.54. Фактический параметр – выражение

Во всех трёх случаях после обработки вызова процедуры полученное значение копируется в соответствующий формальный параметр, принадлежащий процедуре. Как только начинается выполнение процедуры, уже никакие изменения значения формального параметра не оказывают влияния на значение соответствующего аргумента. Это значит, что по окончанию работы процедуры аргумент будет иметь точно такое же значение, каким оно было до начала работы процедуры, независимо от того, что происходило с этим формальным параметром (рис. 8.55)

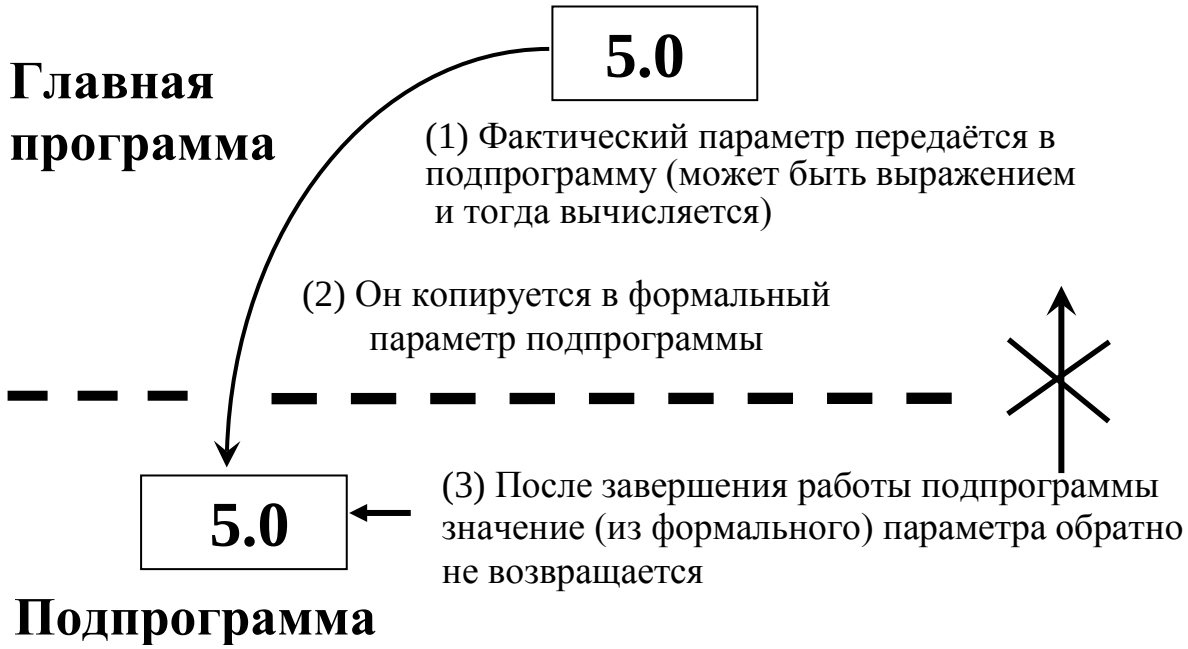


Рис. 8.55. Механизм передачи параметров “по значению”

Совершенно другим образом выполняется ***передача параметров "по адресу", которая ещё называется "передачей по ссылке", передачей по var–параметру или параметром–переменной.***

При таком способе передачи параметра в подпрограмму (процедуру или функцию) пересылается уже не ***значение аргумента***, а ***адрес его местонахождения*** (адрес в памяти компьютера). Если формальный параметр имеет атрибут `var` в его описании в заголовке подпрограммы, а соответствующий фактический аргумент является переменной, то любые изменения значения формального параметра в теле подпрограммы будут заменены его значением по адресу подставленного аргумента. Это связано с тем, что в этом случае формальный и фактический параметры занимают одно и то же место в памяти (рис. 8.56)

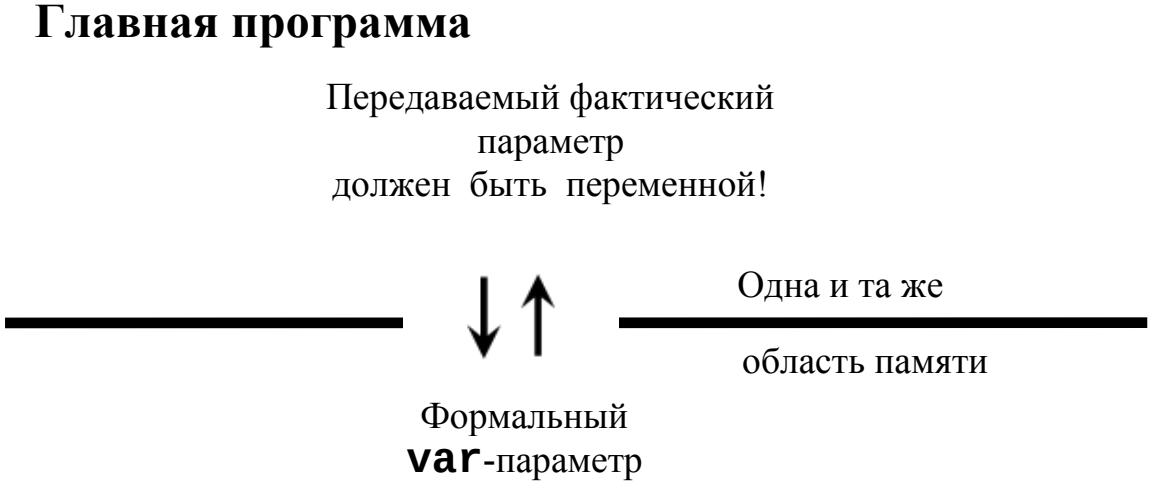


Рис. 8.56. Механизм передачи параметров “по ссылке” (или по адресу)

Поэтому, чтобы предупредить компилятор о намерении пользователя передавать какой-либо параметр по адресу (по ссылке), в заголовке подпрограммы необходимо указать перед его именем служебное слово "`var`".

Таким образом, всё что будет происходить с содержимым этого параметра в подпрограмме, буде отображаться по адресу его расположения в главной программе. Приведём пример взаимодействия значений формальных параметров (рис. 8.57).

Как уже стало ясно, в теле подпрограммы использование глобальных переменных и `var`–параметров практически не отличаются. Важно другое.

Во-первых, если глобальная переменная фигурирует в выражениях программы и подпрограммы, её имя заменить другим объектом уже никак невозможно. Однако можно изменить содержащееся в ней значение до вызова соответствующей подпрограммы. Тогда, если её имя фигурирует в правой

части оператора присваивания в подпрограмме, изменение её значения в главной программе отобразится на значении всего выражения (рис. 8.58).

```
Var
    GLOB1, GLOB2 : Real;
Procedure PP5 (FORM1 : Real; var FORM2 : real);
   Var
                Lock1 : Real;
 Begin
   FORM2 := FORM1+ 2.5; {Значение GLOB2 в главной программе }
    { изменится на (5.0 + 2.5) = 7.5, т.к FORM2 var-параметр!}
    FORM1 := FORM1 – 3.32; {Так вообще писать нельзя  (!!!),}
        {т.к. FORM1 – значение, которому ничего присвоить нельзя!}
 end;
……..
Begin
  GLOB1 := 5.0;
   PP5 (GLOB1,GLOB2); {Вызов процедуры PP5}
  Writeln('GLOB1=', GLOB1:8:2, 'GLOB2=', GLOB2:8:2);
End.
```

Рис. 8.57. Пример формирования вызова процедуры

При завершении работы программы (рис.8.57) на экран будет выведено:

GLOB1= 5.00 GLOB2= 7.50,

То есть значение GLOB1 (которое подставлялось на место параметра-значения) – не изменилось, а поменялось только значение в переменной GLOB2, которое подставлялось на место параметра-переменной.

```
Var
    GLOB1, GLOB2 : Real;
Procedure PP6 (FORM1 : Real);
 Begin
      GLOB2:= GLOB1 + FORM1;
 end;
Begin
   GLOB1 := 5.0;
     PP6 (5.0);     {Результат – GLOB2 =10 }
      GLOB1 := 10;
       PP6 (10);  {Результат – GLOB2 =20 }
End.
```

Рис. 8.58. Пример влияния изменения значений глобальной переменной GLOB1 на результат вычисления процедурой PP6

Фундаментальным элементом языкового механизма представляется, обеспечение возврата результатов действия процедуры в главную программу только через ***глобальный параметр*** (например – GLOB2) или через var–***параметр***.

И тут следует понимать, что в общем случае необходимо разрабатывать подпрограмму таким образом, чтобы результат её действия возвращался ***только*** через var–параметр. Это связано с тем, что в языке ТП есть возможность компилировать подпрограммы отдельно от главной программы в виде так-называемых ***модулей***[25]. В этом случае связь с подпрограммами должен обеспечиваться только через формальные (простые и var) параметры (рис. 8.59).

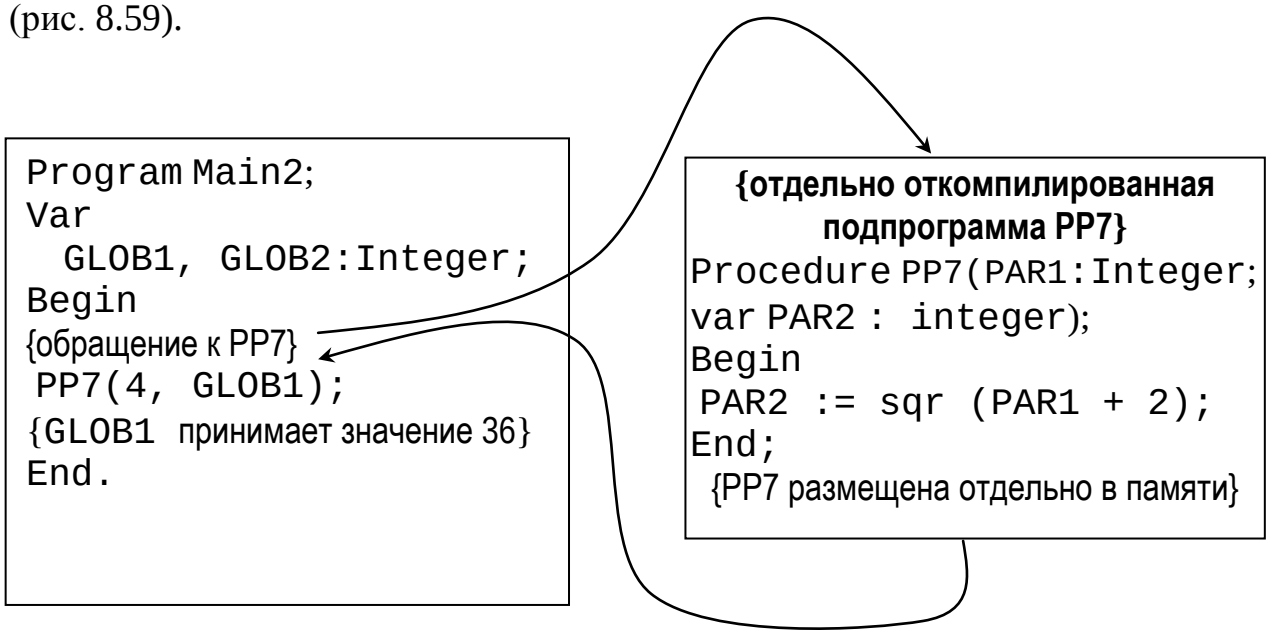


Рис. 8.59. Вызов процедуры, которая откомпилирована отдельно и размещена в модуле

Рассмотрим пример взаимодействия объектов в программах, процедурах и функциях при разных комбинациях передачи в подпрограммы данных в виде глобальных и фактических параметров. Для этого рассмотрим такую задачу.

Реализовать в виде процедуры и функции вычисление следующей формулы: $Z = f\ (x, y) = \dfrac{x + y^2}{a + b}$ (8.14).

Понятно, что значения постоянных величин a и b будут определены в программе в виде констант. А от значения переменных x и y необходимо вводить в процессе решения, то есть на этапе выполнения программы. Проектирование и кодирование описания процедур и функций допускает достаточно много вариантов разнообразных действий, так как данные в

[25] Модуль – программа, которая рассматривается как отдельное целое в процессах сохранения, трансляции и объединения с другими программными модулями при их загрузкеі в оперативную память компьютера

подпрограммы можем передавать и через глобальные переменные, и через фактические параметры. Поэтому рассмотрим некоторые возможные примеры, которые вытекают из того, что данные в подпрограммы могут передаваться двумя основными способами: по значению (параметр-значение) либо по адресу. (параметр-переменная). В следующем примере вычисленные с помощью разных подпрограмм значения вышеприведенной формулы заносятся в переменные Z1, Z2, ... Z6. Обратите внимание на процедуру P2 и функцию F5, которые сконструированы вообще без формальных параметров (рис. 8.60).

```
Program FXY;
    const a=3.4, b=-7.302;
 var
    x, y,
     Z1, Z2, Z3, Z4, Z5, Z6 : real;

{ Процедура P1 спроектирована с параметрами. Все данные вводятся}
{как параметры-значения, а результат выводится через параметр-переменную yy}

Procedure P1(x1, y1, a1, b1 : real; var yy :real);
   begin yy:=(x1+sqr(y1))/(a+b) end;

{ Процедура P2 без параметров. В ней используются глобальные переменные программы FXY –
x, y, a, b, а результат возвращается через глобальную переменную Z2}

Procedure P2; begin Z2:=(x+sqr(y)/(a+b)); end;

{Процедура P3 с одним параметром-переменной zz, через которую
в главную программу FXY возвращается вычисленное по формуле значение }

Procedure P3 (var zz : real);
   begin zz := (x+y*y)/(a+b); end;

{Функция F4 с четырьмя параметрами-значениями. Результат вычисления
 по формуле возвращается через имя функции F4 }

Function F4(x2,y2,a2,b2 : real):real;
   begin F4 := (x2+sqr(y2))/(a2 +b2); end;

{Функция F5 без параметров. Использует глобальные переменные программы.
 А через какие непосредственно? }

Function F5:real; begin F5:=(x+sqr(y))/(a+b); end;

{Функция F6 с двумя параметрами-значениями, через которые к ней передаются значения
переменных x и y }

Function F6(xx, yy :real):real;
   begin F6:=(xx+yy*yy)/(a+b); end;
```

Рис. 8.60. Программа описания подпрограмм разными способами

{ продолжение программы рис. 8.60 на следующей странице }

```
        {Операторная часть программы: вызовы подпрограмм  (процедур и функций)}
Begin
      P1(4.1,2.56,-3.1,4.6,Z1);  {А}
        Z3:=F4(3,7,2,61);        {Б}

      Write ('Введите x и y:');
        readln(x,y);
  P2;
    P3(Z4);
      Z5:=F5;
        Z6:=F6(Z4-2.5*Z5, ln(Z1/Z2));
  Writeln('Результаты=', Z1, Z2, Z3, Z4, Z5, Z6);
    Readln;
End.
```

Рисунок 8.60. (продолжение)
Программа описания подпрограмм разными способами

Программа на рисунке 8.60 позволяет ответить на очень интересный вопрос: Можно ли в начале выполнения программы **FXY** вызвать любую процедуру или функцию из вишеописанных, если переменные a, b, x, y, определяющие результат вычисления по формуле (8.14) ещё не инициализированы (то есть в них не введены начальные значения)?

Строки **А** и **Б** на второй части рисунка 8.60 дают положительный ответ на этот вопрос. Действительно, это возможно сделать при условии, если обращаться к некоторым процедурам и функциям фактическими параметрами по значениям, которые являются константами (см. обращения к процедуре P1 и функции F4 в строках А и Б).

Упражнения

1. Написать на языке ТП функцию для вычисления знака числа по формуле: $sign(x) = \begin{cases} 1, x > 0 \\ 0, x = 0 \\ -1, x < 0 \end{cases}$.

2. Вычислить следующие выражения: sign(a*b), sign(a/b), sign(-b), sign(-a), воспользовавшись функцией, написанной для предыдущего примера.

3. Написать функцию для вычисления суммы цифр трёхзначного натурального числа. Вычислить сумму цифр для чисел от 100 до 120.

4. Написать функцию для вычисления факториала n!=1•2•3•...•n. Вычислить факториалы чисел от 1 до 7.

5. Написать функцию для возведения числа x в степень m (m -натуральное число). Вычислить x^3, x^4, x^5.

6. Написать функцию для вычисления арксинуса. Вычислить arcsin(0.9), arcsin(0.1), arcsin(--0.9), arcsin(0.99). Для вычислений воспользоваться формулой: $\arcsin(x) = arctg\left(\frac{x}{\sqrt{1-x^2}}\right)$.

7. Для пунктов 1-6 реализовать на языке ТП те же алгоритмы, только в виде процедур.

8. Написать программу вычисления определённых интегралов по формуле трапеции:

$$\int_a^b y\,dx = \frac{b-a}{n}\cdot\left(\frac{y_0+y_n}{2} + y_1 + y_2 + \ldots y_{n-1}\right),$$

для следующих выражений:

a)	$\frac{1}{1+x}$	*b*)	$\frac{\ln(1+x)}{1+x^2}$
c)	$\frac{\sin x}{x}$	*d*)	e^{-x^2}

9. Написать подпрограммы вычисления определённых интегралов с помощью формулы трапеции для функций предыдущих пунктов *a-d* с точностью около $\varepsilon = 0.00001$.

8.23. Работа с массивами. Примеры многомерных массивов

В линейной алгебре и других разделах математики существует множество задач, связанных с использованием ***матриц***. Как Вы очевидно знаете, матрицей называется совокупность чисел a_{ij}, размещённых в виде прямоугольной таблицы[26]:

$$\begin{pmatrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & a_{1m} \\ a_{21} & a_{22} & \cdot & \cdot & \cdot & a_{2m} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & \cdot & \cdot & \cdot & a_{nm} \end{pmatrix}$$

где a_{ij} – элементы матрицы.

Здесь индексы i и j означают, что элемент a_{ij} расположен на пересечении i-й строки и j-го столбца матрицы. Если матрица имеет n строк и m столбцов, то она называется матрицей ***размера*** $n \times m$. Матрица называется ***квадратной*** порядка n , если $n = m$. При $n \neq m$ матрица называется ***прямоугольной***. Прямоугольная матрица размера $n \times 1$, которая состоит из одного столбца называється ***вектор-столбцом***.

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_n \end{pmatrix}$$

А прямоугольная матрица размера $1 \times m$, состоящая из одной строки называется ***вектор-строкой***.

$$\begin{pmatrix} \beta_1 & \beta_2 & \cdots \beta_m \end{pmatrix}$$

Для работы с матрицами в Турбо Паскале существует структура данных под названием ***массив***.

8.23.1. Описание массивов в программе

Массив – это регулярная структура (последовательность) ***однотипных*** данных, объявляемых специальной конструкцией языка ТП:

```
Array [<ДиапазоныИндексов>] of <ТипКомпонентов>;
```

[26] Воеводин В.В., Кузнецов Ю.А. Матрицы и вычисления. – М.: Наука. Главная редакция физико-математической литературы, 1984. – 320 с.

Под однотипностью данных, содержащихся в массивах, понимается принадлежность каждому конкретному массиву только описанных в его заголовке элементов (Real, Integer и др.). То есть, в массиве не могут, к примеру, одновременно находиться числа вещественного и целого типов.

Доступ к любому компоненту массива обеспечивается простым указанием его порядкового номера. В двумерных массивах компоненты (элементы) размещаются в памяти ПК последовательно, но так, что скорее изменяется самый правый индекс. То есть можно сказать, что массивы хранятся ***построчно***.

Наиболее часто массивы используют для хранения вектор-столбцов и вектор-строк (одномерные массивы), а также для конструирования двумерных массивов (то есть матриц), трёхмерных массивов и т.д.:

```
Var
   Vector : Array [1..3] of Real; {одномерный массив Vector,
                                   аналог вектор-столбцов и
                                   вектор-строк, на 3 элемента}
   Matrix : Array [1..3,1..3] of Real; {двумерный массив
                                        Matrix, аналог
                                        матрицы}
```

Таким образом, мы объявляем две структуры: `Vector` – из трёх значений типа Real, проиндексированных заданным диапазоном целых чисел:

```
Vector[1], Vector[2], Vector[3]
```

и `Matrix` – из девяти значений типа Real, проиндексированных заданным диапазоном целых чисел (так они и хранятся в памяти компютера):

```
Matrix[1,1], Matrix[1,2], Matrix[1,3],
Matrix[2,1], Matrix[2,2], Matrix[2,3],
Matrix[3,1], Matrix[3,2], Matrix[3,3],
```

что соотвествует матрице вида:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Кроме того, что Вам теперь понятно, как получить доступ к каждому из элементов массивов, Вы можете также заметить схожесть индексирования элементов массива `Matrix` с обозначениями элементов матриц a_{ij}, которые представляют совокупности вещественных чисел или элементов других типов

данных (к примеру, Boolean, String и т.д.), расположенных в виде прямоугольной таблицы. Этот факт позволяет использовать массивы для проведения вычислений с использованием матриц и не только. Вместе с тем, Вы должны знать некоторые тонкости работы с массивами в ТП.

Если индексация элементов задаётся числовым диапазоном, то необходимо выполнять только два требования:

– группы индексов <ДиапазоныИндексов> в описании массивов не должны иметь тип LongInt. То есть их тип должен "вмещаться" максимум в типе Word и таким образом принимать максимальные значения не более 65535;

– произведение количества компонентов массива на размер компонентов, выраженный в байтах не может превышать 65520 байт ≈ 64Кбайт.

В рамках возможностей описания структур данных ТП можно использовать, например, такие конструкции:

```
Var
  A : Array [9..99] of Char; {Массив A из 91-го символьного элемента}
  B : Array [1..100] of Boolean; {Массив B из 100-а элементов
                                  булевского типа}
  C : Array [-10..10] of LongInt; {Массив C из 21-го числового
                                   элемента, к тому же не требует
                                   приведения индексов к диапазону
                                   1..N (как в языке Фортран) или 0..N (как в
                                   языке C)}
```

Более того, любой из ***структурированных типов*** (к которым в ТП относятся: массивы, записи, множества и файлы) может быть описан с помощью служебного слова **Type**. Например, приведенное выше описание трёх массивов с его помощью может быть записано так.

```
Type
R : Array [9..99] of Char;
S : Array [1..100] of Boolean;
T : Array [-10..10] of LongInt;
U : Array [0..20] of LongInt;

Var
  A : R; {Массив A из 91-го символьного элемента}
  B : S; {Массив B из 100-а элементов булевского типа}
  C : T; {Массив C из 21-го числового элемента}
  D : T of S; {Двумерный массив D из 21 x 21 числових элементов}
  E : Array [-10..10] of Array [1..100] of Array [-
  10..10] of Real; {Трёхмерный массив E (попробуйте сами представить его
                                         структуру!)
```

В данном описании, в переменных (R, S, T и U) описываются ″формы″ задания расположения в памяти тех или других структур данных (то есть наборы последовательностей компонентов разных типов). А потом в операторе `Var` они используются для определения идентификаторов массивов для обработки их в программе. Следует чётко представлять разницу между этими двумя служебными словами (`Type` и `Var`). Если `Type` задаёт ***только форму*** представления последовательностей элементов для занесения данных соответствующих типов в будущие структуры массивов, то служебное слово `Var` обеспечивает ***размещение*** описанных в нём массивов данных в ***памяти компьютера.***

Это значит, что после обработки средой ТП строки с описанием (`Var A : R;`), в памяти ПК будет поименовано (именем `A`) и выделено ровно 90 байтов для размещения 90 компонентов (переменных) типа `Char` (то есть символов).

8.23.2. Ввод-вывод массивов

Теперь, когда Вы уже знаете, как описывать массивы и умеете работать с операторами циклов с параметрами, Вы в состоянии организовать в своей программе ввод данных в массивы и вывод данных из них. Имейте в виду, что в Турбо Паскале, массивы в оперативной памяти располагаются статически, что не позволяет без специальных средств (ссылок или динамических переменных) ***изменять их границы в процессе работы программы***. Обычным выходом из этой неудобной ситуации является описание массивов на максимальную длинну, а необходимые размеры векторов и матриц для решения каждой конкретной задачи вводятся в программе. Также обратите внимание на то, что ввод и вывод массивов выполняется поэлементно (рис. 8.61).

```
Program ArrayInputOutput;
Uses Crt;
Type
  VectorType = Array[1..50] of Real;
  MatrixType = Array[1..50, 1..50] of Real;

Var
  A: VectorType; {Массив A векторного типа из 50 элементов}
  B: MatrixType; {Массив B матричного типа из 50x50 элементов}
  i, j : Integer; {Параметры цикла для обробки векторов и матриц}
  n, m : Integer; {К-во элементов массивов в строке и в столбце}

Begin
       ClrScr;
```

Рис. 8.61. Ввод и вывод массивов на языке Турбо Паскаль

```
 { == Ввод-вектора A ==}
 WriteLn('Введите (n) к-во элементов вектора A, которые
будут обрабатываться (не более 50-и)');
 ReadLn(n);
 WriteLn('Введите n элементов вектора A:');
 for i:=1 to n do {Организуем цикл по количеству n элементов}
   begin
     Write (' A[',i,'] = '); {Удобно видеть номера элементов,
которые вводятся}

     ReadLn (A[i])
   end;
 WriteLn; {Устанавливаем курсор на новую строку}

 { == Вывод-вектора A ==}
 WriteLn ('Введённные значения вектора A: ');
 for i:= 1 to n do {Используем оператор Write с форматированием}
   Write (' A[',i,'] = ',A[i] : 9);{чтобы экономить пр-во экрана}
 WriteLn; {Устанавливаем курсор на новую строку}

 {== Ввод матрицы B ==}
 WriteLn ('Введите количество элементов матрицы B, которые
обрабатываются - (n,m) (не более 50x50)');
 ReadLn(n, m);
 WriteLn('Введите n × m элементов матрицы B:');
 for i:= 1 to n do { Ввод матрицы B требует }
   for j:= 1 to m do { двух циклов с параметром}
     begin
       Write (' B[',i,',j,'] = ');
       Read (B[i,j]); {Чтение элемента матрицы}
     end;

 WriteLn; {Установка курсора после чтения последнего элемента массива}

 WriteLn('Матрица B с', n,' строк и',m,' столбцов:');
 for i := 1 to n do { Вывод матрицы B требует }
   for j:= 1 to m do { двух циклов с параметром}
     Write(' B[',i,',',j,'] = ',B[i,j] : 9);
      WriteLn; {Устанавливаем курсор на новую строку}
  ReadLn

 End.
```

Рис. 8.61. Ввод и вывод массивов на языке Турбо Паскаль (продолжение)

8.23.3. Стандартные приёмы работы с векторами и матрицами. Суммирование элементов массивов

При работе с матрицами, как правило, приходится иметь дело с набором стандартных приёмов и их композицией. Поэтому Вам необходимо знать компоненты этого набора, чтобы в будущем конструировать из него фрагменты алгоритмов программ, которые могут возникнуть из постановки конкретных задач.

Самой простой из них, является суммирование элементов векторов и матриц. Для этого в программе необходимо описать переменную Sum такого же типа, что и элементы массива, и в цикле, по очереди, добавлять к ней элементы массива:

```
... {Суммирование элементов одномерного массива – вектора}
. . .
Type
  VectorType = Array[1..20] of Real;
Var
  A: VectorType;
  i, n : Integer; {Параметр цикла и размер вектора}
  Sum  : Real; {Переменная Sum для суммирования}
. . .
  Sum := 0; {Подготовка переменной Sum для суммирования!!!}
  for i: =1 to n do
    Sum := Sum + A[i];
. . .
```

Для суммирования элементов двухмерного массива к предыдущему алгоритму следует добавить ещё один цикл с параметром для обеспечения перебора всех его индексов:

```
.   .   .
    { Суммирование элементов двухмерного массива}
Type
  MatrixType = Array[1..20, 1..25] of Real;
Var
  B: MatrixType;
  i, j, n, m : Integer; {Параметры циклов и границы массива}
  Sum : Real; {Переменная Sum для суммирования элементов}
.   .   .
  Sum := 0; {Подготовка переменной Sum для суммирования}
  for i:= 1 to n do {Двигаемся вдоль строки}
    for j:= 1 to m do {Двигаемся по столбцу}
      Sum := Sum + B[i, j];{Суммируем в Sum каждый элемент
                            матрицы}
{Для вычисления среднего арифметического добавляем оператор}
  Sum := Sum / (n*m);}
.   .   .
```

8.23.4. Использование счётчика

Ещё одной распространённой задачей, является подсчёт элементов матриц с определёнными характеристиками, к примеру, положительных или отрицательных элементов. Это требует введение в тело цикла условного оператора:

```
. . .
{Определим количество положительных элементов Count типа Integer}
  Count := 0; {Подготовка переменной Count для суммирования!!!}
  for i:= 1 to n do
    if A[i] > 0 then{Определение полодительного элемента}
       Count := Count + 1;{Увеличение количества положительных элементов}
. . .
```

Теперь у Вас не возникнет трудностей при составлении алгоритма для определения положительных элементов двухмерного массива, то есть матрицы. Кроме этого, теперь Вы можете использовать этот алгоритм для определения:

1. числа отрицательных или нулевых элементов;
2. числа элементов, равных заданному числу;
3. числа элементов массива, содержащих числа превышающие или не превышающие заранее заданное число;
4. числа элементов, содержащих значения, находящиеся в заданном интервале и т.д.

Теперь давайте используем наши знания для конструирования алгоритма нахождения среднего арифметического положительных элементов двухмерного массива. Для этого необходимо вычислить сумму положительных элементов матрицы и их количество, то есть применить одновременно уже известные нам алгоритмы суммирования и «счётчика»:

```
. . .
{Композиция: «суммирование» + «счётчик»}
{Находим середнее арифметическое положительных элементов }
Type
  MatrixType = Array[1..20, 1..25] of Real;
Var
  B: MatrixType;
  i, j, n, m : Integer; {Параметры циклов и границы массива}
  Sum : Real; {Переменная Sum для суммирования}
. . .
  Sum := 0; {Обнуление переменной Sum для суммирования}
   Count := 0;
  for i:= 1 to n do
    for j:= 1 to m do
      if B[i,j] > 0
        then begin
               Sum := Sum + B[i,j];
               Count := Count +1;
             end;
    Sum := Sum/ Count; {Среднее арифм. положительных элементов }
. . .
```

8.23.5. Определение max/min элемента массива

Очень часто программирование алгоритмов работы с матрицами требует определения максимального или минимального элемента. Для этого существует очень простой алгоритм, который Вы должны знать:

```
. . .
Max := A[1]; {Считаем, что первый элемент массива – максимальный }
for i:= 2 to n do {Поэтому начинаем обработку со второго элемента}
  if A[i] > Max {Сравниваем каждый следующий элемент с текущим максимальным значением }
     then Max:= A[i];{Заменяем старое значение Max новым бо́льшим }
{После выполнения цикла в Max находится максимальное значение из A }
. . .
```

На базе этого алгоритма можем сконструировать алгоритм определения минимального элемента двухмерного массива и его индексов:

```
. . .
Var
  IndI, IndJ : Integer; {Переменные для сохранения индексов}
. . .
Min := B[1,1];
  for i:= 1 to n do
    for j:= 1 to m do
      if B[i,j] < Min
        then begin
               Min := B[i,j];
               IndI := i;
               Indj := j
             end;
. . .
```

8.23.6 Работа с чётными и нечётными элементами массива

Ещё одним достаточно полезным алгоритмом, который может Вам понадобиться при работе с массивами, является поиск чётных/нечётных элементов целых массивов и их суммирование. Для этого в Турбо Паскале существует специальная встроенная логическая функция `Odd(x)`, которая при целом аргументе `x` возвращает значение `True`, если `x` нечётное число, и `False` в случае чётного числа:

```
. . .
{Находим сумму нечётных элементов с использованием функции Odd(i)}
  Sum := 0;
  for i:=1 to n do
    if Odd(i) then {для чётных – условие – not Odd(i)}
       Sum := Sum + A[i];
  . . .
```

8.23.7. Решение практической задачи с массивами

Тепер, когда Вы уже владеете основными приёмами работы с матрицами, попробуем решить конкретную задачу в соответствии со сформулированной в п.8.1 технологии работы в среде ТП. Например, рассмотрим такую задачу.

Даны два трёхмерных вектора $\overline{X} = \left(x_1,\ x_2,\ x_3 \right)$ и $\overline{Y} = \left(y_1,\ y_2,\ y_3 \right)$. Требуется найти угол между ними.

1. На первом шаге, Вам необходимо решить задачу в её предметной области, то есть привести её к виду, когда Вы можете представить себе все действия по созданию алгоритма решения задачи. Как следует из справочника по математике[27], эта задача относится к области векторного счисления, из раздела "Геометрические приложения векторной алгебры", где приведена формула вычисления угла между трёхмерными векторами $\overline{X}$ и $\overline{Y}$:

$$\cos\varphi = \frac{\overline{X}\ \ \overline{Y}}{\left|\overline{X}\right|\ \ \left|\overline{Y}\right|} \qquad (8.15)$$

–где $\overline{X}\ \ \overline{Y} = x_x y_x + x_y y_y + x_z y_z$ – скалярное произведение векторов $\overline{X}$ и $\overline{Y}$;

$\left|\overline{X}\right|$ – длина вектора $\overline{X}$:

$$\left|\overline{X}\right| = \sqrt{X^2} = \sqrt{x_x^2 + x_y^2 + x_z^2}\ ; \qquad (8.16)$$

$\left|\overline{Y}\right|$ – длина вектора $\overline{Y}$.

С учётом этих зависимостей, конечная формула примет вид (8.17):

$$\cos\varphi = \frac{x_x y_x + x_y y_y + x_z y_z}{\sqrt{x_x^2 + x_y^2 + x_z^2}\ \ \sqrt{y_x^2 + y_y^2 + y_z^2}} \qquad (8.17)$$

Таким образом, мы получили конечное, формульное выражение искомой величины, вполне пригодное для программирования на языке Турбо Паскаль. Вычисление самого значения φ рассмотрено далее в п.8.24.

2. Далее, на втором шаге, Вы должны выбрать входные и выходные данные для программного компонента. Для нашей задачи входными данными будут значения компонентов, составляющих векторы $\overline{X}$ *и* $\overline{Y}$: x_1, x_2, x_3 и y_1, y_2, y_3. Выходными данными будет значение $\cos\varphi$ (т.е. косинуса угла между ними).

3. Проходим по разделам описаний программы. Из стандартных библиотек для работы программы, очевидно, понадобится модуль `Crt` для использования из него процедуры очистки экрана `ClrScr`.

4. На четвёртом шаге выбираем структуры данных. Для нашей задачи они могут выглядеть следующим образом:

---

[27] Бронштейн И.Н., Семендяев К.А. Справочник по математике для инженеров и учащихся втузов. – М.: Наука, Гл. ред. физ.-мат. лит., 1968. – 544 с.

```
Const
  n = 3;
Type
  VectorType = Array[1..n] of Real;
Var
 X, Y : VectorType; {Массивы для векторов}
  XY,                {Переменная для значения скалярного произведения}
  SumX, {Переменная для суммирования квадратов компонентов X }
  SumY, {Переменная для суммирования квадратов компонентов Y }
  CosFi : Real; {Переменная для вычисления результата}
  i : Integer; {Переменная-параметр цикла}
```

Так, как Вы уже имеете некоторый опыт в программировании, то можете предвидеть, что в алгоритме понадобится отдельно суммировать скалярное произведение в числителе формулы (8.17), и суммы квадратов компонентов векторов в знаменателе, для чего понадобятся отдельные переменные.

5. Разработка интерфейса программы не представляется сложной задачей. Ввод элементов векторов был уже рассмотрен в разделе 8.23.2, а вывод результата и вовсе простой.

6. Для работы с массивами нам понадобится управляющая структура `for`, в теле которой мы и разместим все вычисления с векторами.

7. На данном этапе Вы должны собрать все разработанные элементы в единое целое, что и будет в конечном итоге программой вычисления (рис. 8.62):

```
Program VectorAngle;
Uses
  Crt;
Const
      n = 3;
Type
     VectorType = Array[1..n] of Real;
Var
    X, Y : VectorType;  XY, SumX, SumY, CosFi : Real;
  i : Integer;
Begin
  ClrScr;
 {Здесь должен располагаться алгоритм заполнения данными массивов X и Y}
  XY := 0;
  SumX := 0;
  SumY := 0;
  for i:=1 to n do
    begin
      XY := XY + X[i]*Y[i];
      SumX := SumX + X[i];
      SumY := SumY + Y[i];
    end;
  CosFi := XY / (sqrt(SumX)*sqrt(SumY));
  WriteLn('CosFi = ', CosFi);   ReadLn
End.
```

Рис. 8.62. Вычисление угла между двумя трёхмерными векторами

Упражнения

1. Даны два трёхмерных вектора $\overline{X} = (x_1, \ x_2, \ldots x_n)$ и $Y = (y_1, \ y_2, \ldots y_n)$. Найти угол между ними при произвольных числовых значениях компонентов векторов.

2. В массиве $D=(d_1, d_2, ..., d_{12})$ определить среднее арифметическое S отрицательных элементов и их количества K.

3. В вещественном массиве $D=(d_1, d_2, ..., d_{12})$ определить произведение P положительных элементов, количество K нулевых элементов и сумму S отрицательных элементов.

4. В целом массиве $B=(b_1, b_2, ..., b_{10})$ определить максимальный элемент M и значение его индекса. Максимальный элемент заменить на значение “0”.

5. Дана вещественная матрица $Q(3,6)$. Определить значения минимального и максимального элементов матрицы, значения их индексов в массиве и их произведение.

6. Дана матрица $A(4,6)$. Определить сумму S элементов, которые превышают по модулю единицу в каждом чётном столбце и количество таких элементов.

7. Дана матрица $X(6,6)$. Определить максимальный элемент на главной диагонали и сумму елементов главной диагонали.

8. Дана матрица $C(7,7)$. Определить произведение положительных элементов матрицы, расположенных на пересечении чётных строк и нечётных столбцов.

8.24. Модули и работа с ними

При составлении предыдущих алгоритмов мы уже неоднократно использовали в своих программах так называемые модули (вспомните подключение библиотечного модуля Турбо Паскаля `Crt` для вызова из него процедуры очистки экрана `ClrScr`). Все системные библиотеки Турбо Паскаля реализованы в виде модулей, и чтобы использовать их мы должны поместить в программе строку описания:

```
Uses Crt, Graph; {Или имена других необходимых модулей}
```

Вообще ***модули*** предназначены для поддержки принципов модульного программирования при разработке программ. В соответствии с этим принципом взаимное влияние логически независимых фрагментов программы должно быть сведено к минимуму. Однако важнейшими полезными свойствами модулей является возможность сохранять в них оформленные в виде процедур и функций алгоритмы, которые Вы разрабатываете в течение Вашей работы и желаете ***повторно использовать***. Кроме этого, Ваша программа, в которой описаны модули, "видит" все ***данные*** модулей (а также описания типов констант и переменных). Поэтому, при необходимости их использования, нет необходимости описывать все эти характеристики в основной программе.

Как правило, ***модуль*** разделяется на четыре части:

1. Заголовок модуля (`Unit Name;`)
2. Раздел объявлений или интерфейс (`Interface`)
3. Раздел реализации (`Implementation`)
4. Раздел инициализации (между `Begin` и `End`)

Все блоки, составляющие эти разделы, являются необязательными и могут или отсутствовать, или появляться неоднократно. Наименьший по своим размерам, бесплодный, но правильный с точки зрения языка ТП модуль имеет вид:

```
Unit Empty;
Interface
Implementation
End.
```

Обратите внимание, что символы точка с запятой после ключевых слов отсутствуют. Если отсутствует раздел инициализации, то служебное слово `Begin` не ставится.

В принципе, структура полезного ***модуля*** подобна структуре паскаль-программы:

```
Unit UnitName; {Заголовок модуля. Рекомендуется длина имени
                UnitName <= 8 символов}
Interface  {Раздел объявлений, которые будут использоваться в этом
            модуле и в программах, которые его будут использовать!}
Uses ...; {Список модулей, вызываемых из данного модуля}
Const ...; {Объявления библиотечных констант}
```

```
Type ...; {Блок объявлений библиотечных типов}
Var ...; {Блок объявлений библиотечных переменных}
Procedure ProcName(FormalParametersList); {В этом разделе записываются только заголовки процедур с формальными параметрами!}
Function FuncName(FormalParametersList):Type; {В этом разделе записываются только заголовки функций с формальными параметрами!}

Implementation {Раздел реализации}
Uses ...; {Используются при реализации модулей}
Const ...; { Описания в разделе Implementation }
Type ...; {  "видят" только этот  }
Var ...;  {  модуль  }
Procedure ProcName; {В этом разделе записываются только заголовки процедур без формальных параметров и их тела!}
Function FuncName; {В этом разделе записываются только заголовки функций без формальных параметров и их тела!}
                    Begin
  {Блок инициализации модуля}
End.
```

Обратите внимание на то, что точки с запятой после ключевых слов отсутствуют. Если нет раздела инициализации, то `Begin` тоже не ставится.

В принципе, структура полезного ***модуля*** похожа на паскаль-программу:

```
Unit UnitName; {Заголовок модуля. Рекомендуемая длина имени
                UnitName <= 8 символов}
Interface {Раздел объявлений, которые будут использоваться в этом
           модуле и в программах, вызывающих его!}
Uses ...; {Список модулей, используемых в этом модуле}
Const ...; {Объявления библиотечных констант}
Type ...; {Блок объявлений библиотечных типов}
Var ...; {Блок объявлений библиотечных переменных}
Procedure ProcName(FormalParametersList);{В этом разделе записываются только заголовки процедур с формальными параметрами!}
Function FuncName(FormalParametersList):Type; {В этом разделе записываются только заголовки функций с формальными параметрами!}

Implementation {Раздел реализации}
Uses ...; {Используемые при реализации модули}
Const ...; { Описания в розделе Implementation }
```

```
Type ...;  {     Описания, которые "видит"          }
Var ...;   {       только этот модуль               }
Procedure ProcName; {В этом разделе записываются только
                     заголовки процедур без формальных параметров и их тела!}
Function FuncName; {В этом разделе записываются только
                    заголовки функций без формальных параметров и их тела!}
Begin
  {Блок инициализации модуля}
End.
```

Тепер мы можем выполнить упражнение 1 из раздела 8.23.7 с использованием модуля. Приведём его условие.

***Упражнение** 8.23.7.1:* Даны два вектора $\overline{X} = (x_1, \ x_2, \ldots x_n)$ и $Y = (y_1, \ y_2, \ldots y_n)$. Найти угол между ними.

Для вычисления углов между произвольными векторами, расположенными в трёхмерном пространстве, можно реализовать стандартный, т.е. общеупотребительный модуль, который будет содержаться в библиотеке модулей и вызываться по мере необходимости. Данный модуль может содержать описания необходимых структур данных, процедуру ввода компонентов векторов с наперёд заданным их количеством и функцию вычисления угла между двумя векторами с *n* компонентами. В п. 8.23.7 приведена формула вычисления *cos φ*. Учитывая тот факт, что все встроенные тригонометрические функции ТП работают с аргументами в радианах, значение *φ* может вычисляться в зависимости от конкретной задачи следующим образом:

$y = \cos \varphi_{рад}$; $\varphi_{рад} = \arccos(y) = \pi/2 - arctg(y/(1-y^2)^{0.5})$; $\varphi_{град} = \varphi_{рад} \times 180^o/\pi$.

К примеру, $y = \cos \pi/3 = \cos(1.047198) = \cos 60^o = 0.5$;

$\varphi_{рад} = \arccos(0.5) = \pi/2 - arctg(y/(1-y^2)^{0.5}) = 1.047198$;

$\varphi_{град} = 1.047198 \times 180^o/\pi = 60^o$ (где $\pi = 3.141593$).

Текст модуля `MyUnit` приведен на рисунке 8.63.

```
Unit MyUnit;
Interface
Uses
  Crt; {В этом модуле используем библиотечный модуль Crt}
Type
  VectorType = Array[1..50] of Real;
Var
  XX, YY : VectorType; {Массивы, которые используются в
модуле,}
                       {а также в программе, где он описан}
  j : Integer;
```

Рис. 8.63. Структура модуля, для вычисления углов между векторами

```
Procedure VectInput(Var Vect : VectorType;
                                n : Integer);
{ VvodVect – процедура ввода вектора из n елементов}

Function CosFi (Var X, Y : VectorType; n : Integer) : Real;
{ CosFi – функция для вычисления косинуса угла между векторами}
{ Z1, Z2 – одномерные массивы с компонентами векторов}
{ n – количество компонентов каждого из векторов}
Implementation
Procedure VectInput; {Списка параметров здесь нет!}
Var
  i : Integer;
Begin
  for  i:=1 to n do
    begin
      Write('Vect[', i, '] = ');
      ReadLn(Vect[i]);
    end
End; {Proc VvodVect}
Function CosFi; {Списка параметров здесь нет!}
Var
  XY, SumX, SumY : Real;
  i : Integer;
Begin
  XY := 0;
  SumX := 0;
  SumY := 0;
  for i:=1 to n do
    begin
      XY := XY + X[i]*Y[i];
      SumX := SumX + X[i];
      SumY := SumY + Y[i];
    end;
  CosFi := XY / (sqrt(SumX)*sqrt(SumY));
End; {Func CosFi}
Begin {Массивы XX и YY инициализируем для использования в программе}
  ClrScr;
  WriteLn('Введите по 3 компонента массивов  XX  и  YY:');
  for  j:=1 to 3 do
    begin
      Write('XX[', j, '] = ');
      ReadLn(XX[j]);
      Write('YY[', j, '] = ');
      ReadLn(YY[j]);
    end
End. {Unit MyUnit}
```

Рис. 8.63. Структура модуля, для вычисления углов между векторами (продолжение)

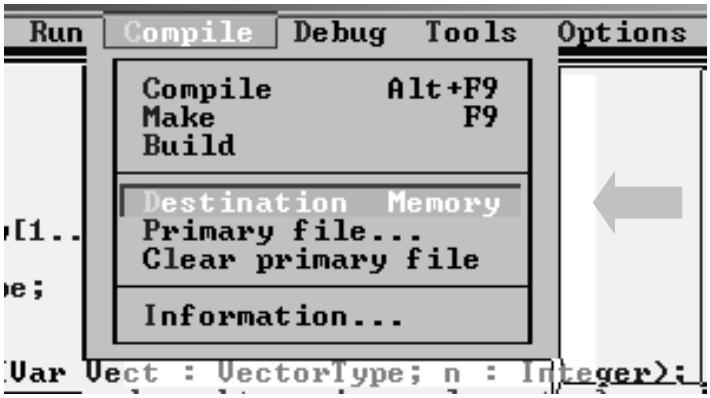


Рис. 8.64. Стандартная установка опции **Compile → Destination**

Рис. 8.65. Установка опции **Compile → Destination → Disk** для сохранения модуля на жёстком диске

Теперь, когда мы имеем свой личный модуль, нужно сделать так, чтобы этим модулем можно было пользоваться в Вашей программе. Скорее всего до этого момента Вы не обращали внимание на то, что Ваши программы при нажатии клавиш **Alt+F9** компилируются в памяти ПК (рис. 8.64). Для модулей удобнее, чтобы они сохранялись где то на жёстком диске. Чтобы установить такой режим компилирования, нужно выбрать опцию **Compile → Destination** и нажать клавишу **Enter**. Если после этого Вы откроете меню опции **Compile,** то увидите следующие изменения (рис. 8.65).

Однако, не спешите компилировать свой модуль, так как существует ещё несколько тонкостей в этом процессе... Дело в том, что в Турбо Паскале необходимо ***обязательно*** указывать каталоги (директории), куда при компиляции будут сохранены **EXE** и **TPU** файлы (**EXE & TPU Directories**), а также откуда потом брать модули, которые подключаются к Вашей программе (**Unit Directories**).

Рис. 8.66. Меню опции **Options**

Для начала, нажатием клавиш **Alt+O,** откройте меню опции **Options,** а в ней выберите команду **Directories** (рис. 8.66). При этом появится меню **Directories** (рис. 8.67). В строке **EXE & TPU Directories** укажите адрес, куда Вы сохраняете откомпилированный модуль. Помните, что исходный файл ***должен иметь то же самое имя, что и название модуля*** с расширением **PAS** (**MYUNIT.PAS**). После компиляции в указанный Вами каталог, полученный файл будет иметь точно такое же имя, но с расширением **TPU** – Turbo Pascal Unit (модуль Турбо Паскаля).

Следует также сообщить компилятору ТП, из какого каталого нужно взять откомпилированный модуль. Для библиотечных модулей этот маршрут известен – каталог **C:\BP\Unit**, куда при инсталляции Турбо Паскаля устанавливаются все библиотечные модули. Откомпилированные лично Вами модули лучше сохранять в отдельном каталоге, который следует указать после точки с запятой в строке **Unit Directories** меню **Directories** (рис. 8.67).

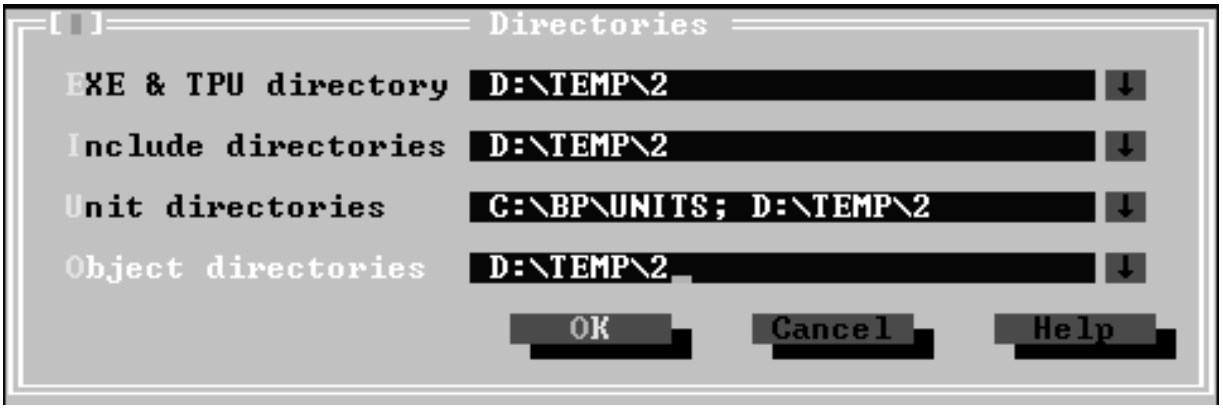


Рис. 8.67. Меню опции **Directories**

Сформулируем этапы Ваших действий по созданию личного ***модуля*** (**Unit**):

1. Наберите текст Вашего модуля (**Unit MyUnit;**) и сохраните его в соответствующем каталоге (**D:\TEMP\2)** ***под названием, совпадающем с именем модуля*** и с расширением .**PAS** (**MYUNIT.PAS**).

2. Нажатием клавиш **Alt+C** откройте меню опции **Compile,** выберите опцию **Compile → Destination** и нажмите клавишу **Enter**, для компиляции модуля на диск (рис. 8.64).

3. Нажатием клавиш **Alt+O** откройте меню опции **Options**, а в ней выберите команду **Directories** (рис. 8.66). При этом, появится меню **Directories** (рис. 8.56). В строке **EXE & TPU Directories** укажите, куда Вы желаете сохранить откомпилированный модуль.

4. В строке **Unit Directories** меню **Directories** (рис. 8.67) укажите через точку с запятой свой каталог, где расположен Ваш модуль (**; D:\TEMP\2).**

5. Подключите свой модуль к программе в строке оператора `Uses:`

```
Uses Crt, MyUnit; {Только после библ. модулей ТП!}
```

Теперь пришло время заняться самой программой. Надеемся, что для Вас это не будет очень сложным делом (рис. 8.68).

```
Program MyUnitProg;
Uses Crt, MyUnit; {Личный модуль описываем после имени модуля ТП}
Var
  X1, Y1 : VectorType; {Тип VectorType используем из модуля
                                MyUnit}
  m      : Integer;
```

Рис. 8.68. Програма, использующая модуль `MyUnit`

```
Begin
  ClrScr;
  WriteLn('Угол между векторами X и Y',
          ' с тремя компонентами = ',
          CosFi (XX, YY, 3)); { Используем инициализированные}
                              { в модуле MyUnit массивы XX и YY,}
                              { а также функцию CosFi}

  WriteLn('Введите 4 компонента массива X1:');
  VectInput(X1, 4); { Используем процедуру VectInput,}
                    { которая описана в модуле MyUnit}

  WriteLn('Введите 4 компонента массива Y1:');
  VectInput(Y1, 4);

  WriteLn('Угол между векторами X1 i Y1',
          'с четырьмя компонентами = ',
          CosFi (X1, Y1, 4)); { Используем массивы X1 i Y1,}
                              { которые описаны в программе}
  ReadLn
End.
```

Рис. 8.68. Программа, использующая модуль `MyUnit` (продолжение)

Обязательно выполните самостоятельно эту программу с модулем `MyUnit`, который необходимо подготовить заранее.

Упражнение

1. Разработать программный модуль ТП, который должен включать:
   – процедуру ввода вектора из *n* элементов;
   – процедуру вычисления вектора, который равен сумме двух векторов;
   – функцию, которая находит наибольший компонент вектора;
   – процедуру вывода векторов на экран ПК.
2. Разработать программный модуль ТП, который должен включать:
   – процедуру ввода вектора из *n* элементов;
   – процедуру вычисления вектора, который равен произведению вектора на скаляр;
   – функцию, которая находит наименьший компонент вектора;
   – процедуру вывода векторов на экран ПК.
3. Разработать программный модуль ТП, который должен включать операции с комплексными числами, и программу, которая его использует.

8.25. Обработка символов и строк

В семействе персональных компьютеров IBM PC используется 256 разноообразных символов. Они имеют свои числовые коды, значения которых лежат в диапазоне от 0_{10} до 255_{10}, то есть общее количество символов равно 256-ти. Когда Вы нажимаете клавишу на клавиатуре, это приводит к тому, что в компьютер посылается сигнал в виде двоичного числа, которое ставится в соответствие ***кодовой таблице***, то есть внутреннего представления символов в компьютере. Во всём мире в качестве стандарта прийнята ***таблица ASCII*** *(American Standard Code for Information Interchange –* ***Американский стандартный код обмена информацией)***. Она указывает на соответствие между изображениями или условными обозначениями символов и их внутренними числовыми кодами. Для сохранения двоичного кода одного символа в этом стандарте выделен 1 байт, который содержит 8 бит. Исходя из того, что каждый бит принимает значения “0” или “1”, количество возможных их сочетаний в байте равно 256.

Кроме того, в компьютерах IBM PC существуют особые комбинации клавиш (табл. 8.47).

Таблица 8.47

Особые комбинации клавиш

Комбинации клавиш	Действия, которые производятся
Ctrl+Alt+Del	Запрос на перезапуск компьютера
Ctrl+NumLock	Остановка работы программы
Ctrl+Break	Прерывания (завершение) работы программы
Shift+PrtSc	Копирование изображения с экрана на принтер

Кодовая таблица допускает представление следующих групп символов:

❶ управляющие символы;
❷ знаки арифметических операций, служебные символы и цифры;
❸ буквы латинского алфавита;
❹ буквы национальных алфавитов;
❺ символы псевдографики;
❻ математические символы.

Управляющие символы используются для специальных целей управления печатными устройствами. К ним относятся маркеры и ограничители при записи, чтении и передаче информации. Наибольший интерес представляют символы *Line Feed* (LF – код 10) и *Carriage Return* (CR – код 13). Они встретятся нам при работе с текстовыми файлами в п. 8.26.7.

Язык Турбо Паскаль поддерживает стандартный символьный тип Char, каждый элемент данных которого занимает в памяти 1 байт. Его значения Вы можете задавать в виде символов, ограничиваемых кавычками:

`‘A’, ‘a’, ‘1’` и так далее.

Обратите внимание и на то, что ‘1’ – это символ единицы с кодом 49_{10}, а представить его можно (также, как и другие ***ASCII-символы),*** с помощью его ***ASCII-кода***, функции `Char()` и специального префикса # (“шарп”) (см. Приложение 6). Таким образом:

```
    #49  ⇨ Char(49) ⇨  ‘1’ {Равноценные представления ‘1’}
Write( #49 ); ⇨ Write( Char(49) ); ⇨ Write( ‘1’ ); {Вывод на
                                                экран символа ‘1’}
```

Для работы с ***ASCII-символами*** используются полезные встроенные функции языка Турбо Паскаль (табл. 8.48).

Таблица 8.48

Функции работы с ***ASCII-символами***

Функция : Тип	Назначение
`Chr(X : Byte) : Char`	Возвращает символ из таблицы *ASCII* по целочисленному значению `X`
`Ord(C : Char) : Byte`	Возвращает порядковый номер символа `C` в таблице ***ASCII-кодов***

То есть:

```
Ord(Char(49)) ⇒ 49.
```

Для лучшего знакомства с символами, попробуйте самостоятельно ознакомиться с тем, как выглядит таблица ***ASCII-символов*** на Вашем компьютере. Для этого можно реализовать и запустить на выполнение следующую программу (рис. 8.69).

```
Program ASCIIChar;
Uses Crt;
Var
  i : Integer; {Переменная цикла}
  StartCode, EndCode, {Начальный и конечный коды символов}
  XStart, YStart : Byte; {Позиции вывода на экран}

Procedure ShowChar(FirstCharCode, {Начальный код символа}
                   LastCharCode, {Конечный код символов}
                   X, Y : Byte); {Позиции вывода символа на экран}
Begin
  while FirstCharCode<=LastCharCode do
    begin
       GotoXY(X,Y); {Установим курсор в позицию X,Y}
       Write((FirstCharCode):4); {Выведем код символа}
       GotoXY(X,Y+1);{Установим курсор в позицию X,Y+1}
       Write('  ', Chr(FirstCharCode), ' '); {Выводим символ}
       Inc(FirstCharCode); {Увеличим код символа на 1}
       Inc(X,4); {Увеличим координату столбца на 4}
       Delay(3000) {Задержим выполнение программы на 3 сек.}
     end;
End; {Proc ShowChar}
```

Рис. 8.69. Вывод ***ASCII-символов*** на экран компьютера

```
Begin {Основная программа}
  ClrScr; {Очистим экран}
  StartCode:=33; {Пропускаем управляющие символы}
  EndCode:=52; {Выводим по 19 символов в строке}
  XStart:=1; {Начинаем с первой позиции экрана X}
  YStart:=1; {Начинаем с первой позиции экрана Y}
  for i:=1 to 10 do
    begin
      ShowChar(StartCode,EndCode,XStart,YStart);
      Inc(StartCode,20); {Переходим к следующей порции символов}
      Inc(EndCode,20); {Переходим к следующей порции символов}
      Inc(YStart,2); {Переводим курсор на две строки ниже}
    end;
  ReadLn {Задерживаем екран до нажатия клавиши Enter}
End.
```

Рис. 8.69. Вывод *ASCII-**символов*** на экран компьютера (продолжение)

После выполнения программы на экране компьютера можно увидеть порядковые номера (коды) и соответствующие им символы из таблицы ASCII (рис. 8.70):

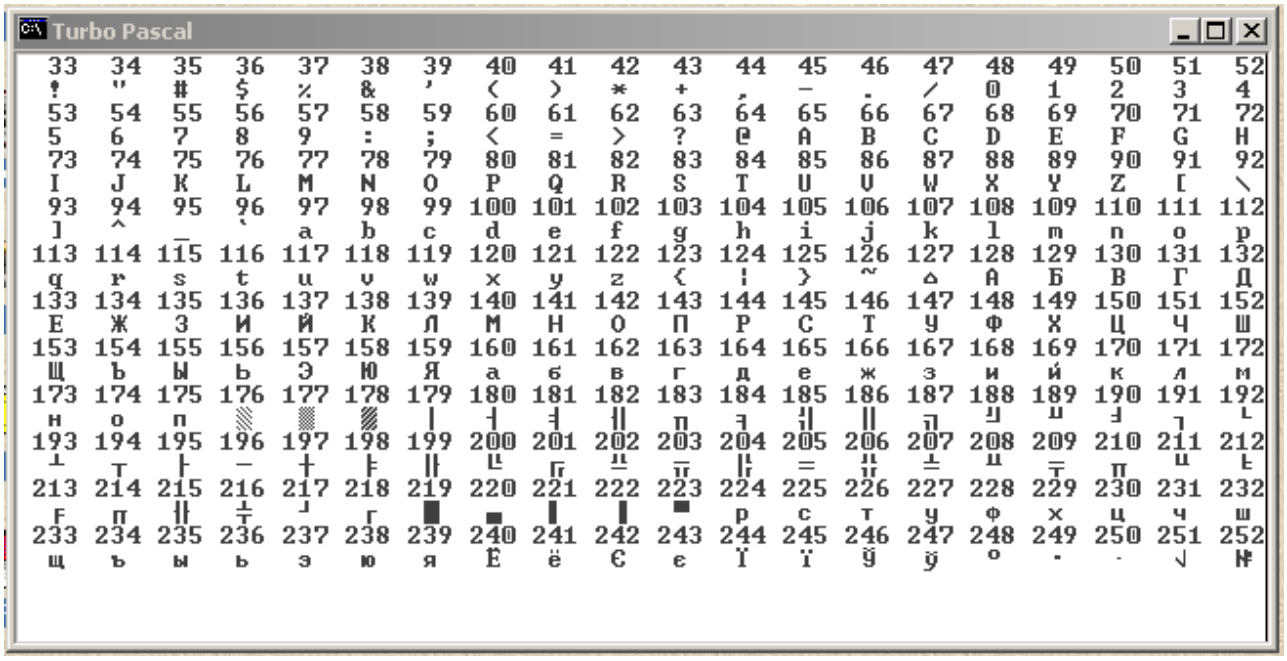


Рис. 8.70. Вывод на экран ПК *ASCII-**символов*** и их кодов (номеров)

Среди многих других, в таблице присутствуют символы русского и украинского алфавита. Эти символы заносятся в кодовую таблицу специальными программами, называемыми драйверами клавиатуры. Среди них наиболее широко распространённой является программа KeyRus.com, которую разработал Д. Гуртяк из г. Донецка, Украина.

Однако, вернёмся к вышеприведенной программе (рис. 8.69). При её разработке использовались некоторые встроенные процедуры Турбо Паскаля, о назначении которых следует сказать подробнее (табл. 8.49).

Таблица 8.49

Стандартные (встроенные) процедуры ТП

Процедура	Назначение
`Inc(Var X:Integer)`	Увеличивает значение `X` на 1
`Dec(Var X:Integer)`	Уменьшает значение `X` на 1
`Inc(Var X:Integer; N:Integer)`	Увеличивает значение `X` на `N`
`Dec(Var X:Integer; N:Integer)`	Уменьшает значение `X` на `N`
`GotoXY(X,Y : Byte)`	Устанавливает курсор в столбец `X` и строку `Y` (находится в модуле `Crt`)
`Delay(ms : Word)`	Задерживает процесс (держит паузу) на `ms` миллисекунд (находится в модуле `Crt`)

Обратите внимание на то, что процедуры `Inc` и `Dec` имеют, так же, как и процедуры `Writeln` и `Readln`, переменное количество параметров. А после того, как процедура `GotoXY` устанавливает курсор в столбец `X` и строку `Y`, следующая операция вывода текста на дисплей размещает первый символ в позицию строки, соответствующей позиции (`X`, `Y`). При этом, процедура `GotoXY` использует систему координат текущего текстового окна, где координаты (1,1) соответствуют его левому верхнему углу (рис. 8.71).

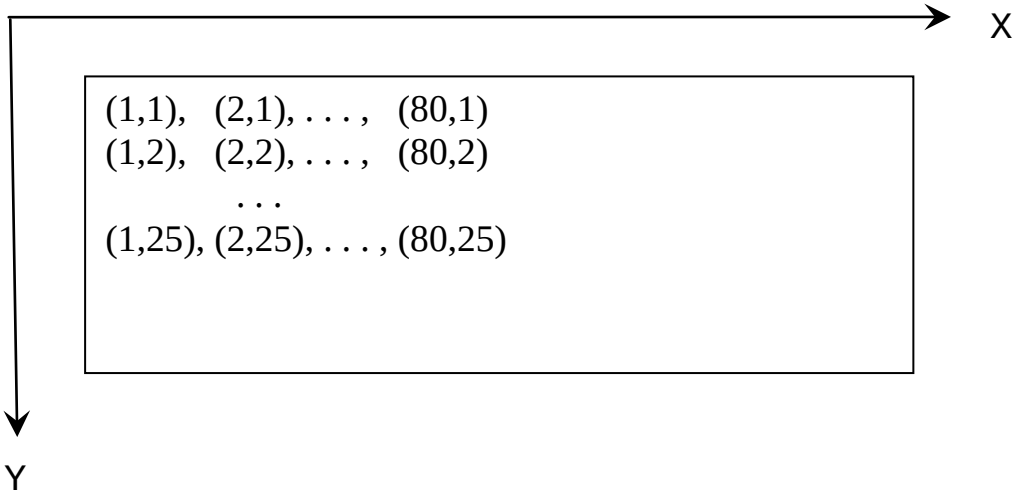


Рис. 8.71. Координаты позиций окна при использовании процедуры `GotoXY`

Если аргументы процедуры `GotoXY` оказываются вне текущего окна, то её вызов не оказывает никакого действия. Минимальные разрешённые значения для `X` и `Y` всегда равны 1, а максимальные (по умолчанию) соответственно `X=80` и `Y=25.`

С точки зрения прорисовки на экране разнообразных прямоугольных рамок с помощью одинарных и двойных линий, наиболее полезными являются символы с кодами 179-218 (рис. 8.72). Увидеть их можно следующим образом:

❶ нажмите клавишу Alt;

❷ не отпуская её, наберите соответствующий код символа псевдографики на дополнительной (серой) цифровой клавиатуре;

❸ отпустите клавишу Alt, и тогда соответствующий символ появится на экране.

Так как комбинирование символов псевдографики между собой достаточно сложное дело, то Вам будет полезна информация, приведенная на рис. 8.61.

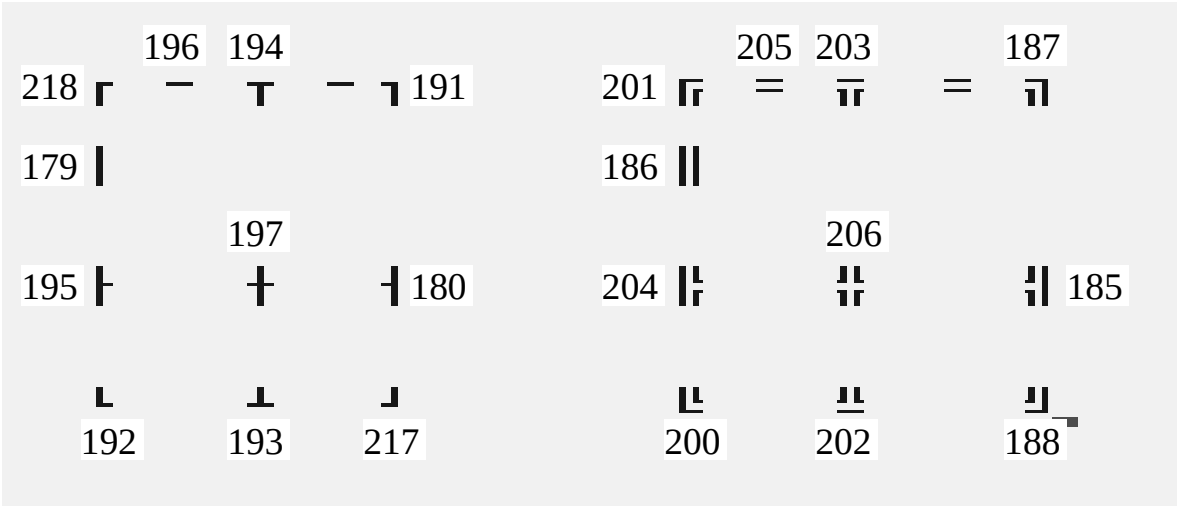


Рис. 8.72. Символы для рисования рамок

Следует отметить, что и интерфейс самого Турбо Паскаля построен с использованием вышеуказанных символов.

8.25.1. Операции над символами

Символы можно только сравнивать друг с другом и присваивать переменным. При этом они считаются равными, если равны их ***ASCII-коды,*** а бо́льший из них тот, который имеет бо́льший ***ASCII-код*** (рис. 8.73).

```
Program Comparison;
Var
  Char1, Char2 : Char;
Begin
  Char1:='A'; {Код символа 'A' равен 65 (рис. 8.70)}
  Char2:='a'; {Код символа 'a' равен 117 (рис. 8.70)}
  if Char1 < Char2 then {Сравниваем символы}
    WriteLn('Первый символ меньше второго');
  ReadLn
End.
```

Рис. 8.73. Сравнение двух символов

8.25.2. Опрос клавиатуры

Чтобы организовать программно опрос клавиатуры при вводе символов нам понадобятся две процедуры. Первая из них “очищает” буфер клавиатуры (рис.8.74).

```
{Файл ClrKey.inc}
Procedure ClrKeyBuf;
Var
  ch : Char;

Begin
  while KeyPressed do
    ch:=ReadKey
End; {Proc ClrKeyBuf}
```

Рис. 8.74. Процедура “очистки” содержимого буфера клавиатуры

В этой процедуре встроенная функция `KeyPressed` возвращает логическое значение `True`, если в буфере ввода с клавиатуры находится хотя бы один символ, и `False`, если буфер пуст.

Когда программа стартует, буфер естественно пустой. Однако, любое нажатие символьной клавиши (кроме клавиш регистров **Ctrl, Shift, Alt** и переключателей типа **NumLock, CapsLock** и т.д.) занесёт её код в буфер. Коды в буфере будут сохраняться до тех пор, пока они не будут считаны или буфер не будет очищен самой программой.

Функция `ReadKey` как бы “вынимает” последовательно введенные в буфер символы по одному за каждое обращение. Эта функция всегда возвращает только один символ, то есть одно значение типа `Char`. Однако у неё есть две важные особенности:

❶ полученные символы никогда не отображаются на дисплее, то есть ввод символа всегда осуществляется “ вслепую ”;

❷ режим работы `ReadKey` зависит от состояния буфера ввода: содержит ли он символы или пуст. Если в буфере что то есть, то `ReadKey` вернёт первый символ в буфере (тот, который был введён ранее всех) и удалит этот символ из буфера. Но если символы в буфере отсутствуют, то функция `ReadKey` приостанавливает работу программы и ожидает, пока не будет нажата какая либо клавиша, генерирующая символьный код.

Эти её особенности можно использовать для построения нескольких полезных конструкций (здесь переменная `ch` должна иметь тип `Char`):

```
while KeyPressed do
    ch:=ReadKey; {Очистка буфера ввода}
repeat until
  KeyPressed; {Ожидание нажатия любой клавиши}
```

Последний цикл завершится, когда в буфер попадёт какой-либо символ. Запомните, что Ваша программа должна в конце своей работы всегда очищать буфер, иначе всё, что было накоплено в буфере, будет выведено в строку **MS-DOS** или в редактор среды программирования.

С учётом всего вышесказанного, реализуем вторую процедуру, которая ожидает нажатия клавиши (рис. 8.75).

```
{Файл Wait.inc}
Procedure Wait;
Begin
  repeat until
    KeyPressed
End; {Proc Wait}
```

Рис. 8.75. Процедура, ожидающая нажатия клавиши

После этого можно создать программу, которая принимает символ нажатой клавиши (рис. 8.76).

```
Program UntilKeyPressed;
Uses Crt;
     {$I clrkey.inc} {Подключаем файл clrkey.inc}
     {$I wait.inc}   {Подключаем файл wait.inc}
Var
  c : Char;
Begin
  ClrScr;
  WriteLn('Нажмите любую символьную клавишу');
  ClrKeyBuf;          {Очищаем буфер клавиатуры}
  c:=ReadKey;         {Ждём нажатия клавиши}
  WriteLn('Была нажата клавиша с символом ', c);
  WriteLn;
  WriteLn('Программа держит паузу до первого ',
          'нажатия любой символьной  клавиши... ');
  Wait;               {Ждём нажатия символьной клавиши}
  ClrKeyBuf           {Убираем "мусор" из буфера}
End.
```

Рис. 8.76. Программа, котрая принимает символ нажатой клавиши

8.25.3. Строковые типы данных (String)

Строковый (или стринговый) тип данных определяет множество символьных цепочек произвольной длины (от 1 до 255 символов). Его можно представить себе в виде массива символов:

```
Var
     CharArray : Array[1..255] of Char;
```

Для описания строкового типа используется служебное слово String и идентификатор переменной, либо он же с квадратными скобками, в которых

указана максимальная длина строки, которая не должна превышать значения 255. Если размер строки в операторе описания не указан, её длина по умолчанию принимается равной 255-ти символам.

```
Type
   Line = String; {Строка длиной 255 символов}
   Line30 = String[30]; {Строка длиной 30 символов}
Var
   STR1 : String; {Строка длиной 255 символов}
    MyLine : Line;
     MyLine30 : Line30; . . .
```

Для описанной строковой переменной длиной N символов в ТП отводится N+1 байтов памяти, из которых первый байт с номером “0” содержит значение текущей длины этой строки (т.е. количество символов в ней), а следующие N байтов предназначены для хранения символов строки, например, ‘I know Turbo Pascal’ (рис. 8.77). Проверьте это на ПК, имея в виду, что код символа “пробел” равен 32.

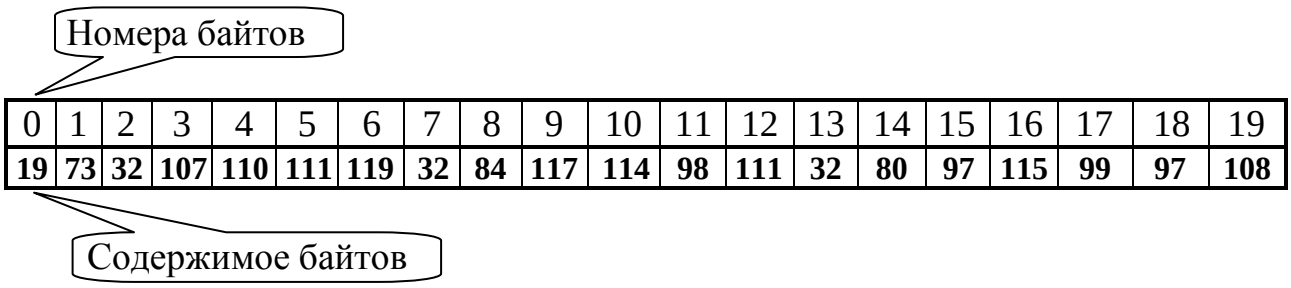


Рис. 8.77. Сохранение символов строки в переменной типа `String`

При хранении данных в переменных типа `String` , существует одна тонкость, которую Вы должны себе очень чётко представлять. Содержимое их нулевого байта, хранящее количество занесённых в переменную символов, может трактоваться по разному (рис. 8.78).

```
Program StrigNumber;
Uses Crt;
Var
  St : String;
  Ch : Char;
  By : Byte;

Begin
  ClrScr;
  St:='Let Me Tell You That I Love You Very Much';
  WriteLn('Zero Byte Char : ', St[0]);
  WriteLn('Zero Byte Ord: ', Ord(St[0]));
  ReadLn
End.
```

Рис. 8.78. Проверка содержимого нулевого байта строки

При запуске программы рис.8.78 получаем следующий результат (рис. 8.79):

Рис. 8.79. Содержание нулевого байта переменной типа `String`

Отсюда следует, что содержимое нулевого байта трактуется, как ***код символа!*** Можно отметить также, что ТП будет видеть столько символов строки, какое значение Вы занесёте в нулевой байт.

Итак, элементы строки нумеруются целыми числами от 1 до 255. Доступ к каждому i-му элементу строки подобен обращению к i-му элементу массива (к примеру, S[i]).

```
MyLine := 'ABCD' {Var MyLine : String;}
WriteLn (Ord(MyLine[0])); { На экран будет выведено "4"}
MyLine[0] := #2;
WriteLn(MyLine); {Будет выведено "AB"}.
```

Строки можно присваивать строчным переменным, сливать содержимое двух и более строк в одну (т.н. операция конкатенации, имеющая знак "+"), а также сравнивать их между собой. Значения символов вводятся в строки операциями Read и ReadLn:

```
Var
  S1, S2, S3 : String;
Begin
  S1 := 'Вам ';
  S2 := 'привет';
  S3 := S1 + S2; {S3 = 'Вам привет'}
  S3 := S3 + '!'; {S3 = 'Вам привет!'}
End.
```

Если при присваивании символов стринговой переменной количество символов в правой части оператора присваивания превышает описанный её размер, то лишние символы отбрасываются.

Сравнение строк выполняется по следующим правилам:

❶ Более короткая строка всегда меньше, чем более длинная;

❷ При равных длинах производится поэлементное сравнение символов этих строк с учётом лексикографической упорядоченности значений стандартного символьного типа Char:

```
'abcd' = 'abcd'   ⇒ True
'abcd' <> 'abcde' ⇒ True
'abcd' <> 'abcd'  ⇒ True
'abcd' > 'abcD'   ⇒ True  {'d' < 'D'}
'abcd' > 'abc'    ⇒ True
'aBcd' < 'ab'     ⇒ True   {'B' < 'b'}
```

Содержимое строк можно вводить с клавиатуры. При этом Вы должны отслеживать соответствие количества введенных символов длинне описанных строк (рис. 8.80).

```
Program StringInput;
Var
  S1 : String[5]; {Длина строки S1 – пять символов}
Begin
  ReadLn(S1);   {Вводим 6 символов: 'Привет'}
  WriteLn(S1); {Избыток символов отбрасывается. Выводится: 'Приве'}
  ReadLn
End.
```

Рис. 8.80. Соответствие введённых символов длине строки

Для работы со строками существует большое количество процедур и функций, которые приведены в таблице 8.50.

Таблица 8.50.

Процедуры и функции работы со строками

Процедуры и функции	Назначение
Редактирование строк	
`Length(S: String) : Byte`	Выдаёт текущую длину строки `S`
`Concat(S1, S2, . . . ,Sn ): String`	Возвращает конкатенацию, то есть слияние указанных строк S1…Sn
`Copy (S: String; Start, Len: Integer) : String`	Возвращает подстроку длины `Len`, начинающуюся с позиции `Start` строки `S`
`Delete (Var S: String; Start, Len: Integer)`	Удаляет из строки `S` подстроку длиной `Len`, начинающуюся в позиции `Start` строки `S`
`Insert (Var S: String; SubS: String; Start: Integer)`	Вставляет в строку `S` подстроку `SubS`, начиная с позиции `Start`
`Pos (SubS, S: String;): Byte`	Ищет вхождение подстроки `SubS` в строке S и возвращает номер первого символа `SubS` в `S` или 0, если `S` не содержит `SubS`
Процедуры преобразования	
`Str (x[:F[:n]]; Var S: String)`	Преобразует числовое значение `X` в строковое `S`. Можно задавать формат для представления `X` в строке (`[:F[:n]`)
`Val (S: String; Var x; Var ErrCode: Integer)`	Преобразует цифры из строки `S` в значение числовой переменной `X`

Упражнения

1. Назовите, какие особые комбинации клавиш существуют в компьютерах IBM PC?

2. Какие полезные функции для работы с ***ASCII-символами*** существуют в Турбо Паскале?

3. С помощью программы, показанной на рис. 8.69, выведите на экран компьютера символы, ***ASCII-коды*** которых превышают значение 240.

4. Создайте на экране компьютера рамки с одинарной и двойной обводкой, используя символы рис. 8.72.

5. С использованием программы рис. 8.75 разработайте свою собственную программу, которая опрашивает клавиатуру и ожидает нажатия нескольких клавиш, каждой из которых соответствует определённое действие.

6. По каким правилам размещается информация в строковых переменных?

7. Разработайте программу, которая считывает символьную информацию с клавиатуры и добавляет её к строковой переменной.

8. Разработайте программу табулирования десяти значений функции F при изменении двух любых других.

$$F = \frac{1}{1 - \frac{1 + (\frac{r}{g})^2}{(\frac{E^2}{S^2}) \cdot \left(1 + \frac{R}{P}\right)^3}}$$

Начальные значения для переменных задайте следующие:
r = 1.56; g = 0.34; E = 2.47; S = 1.004; R = 7.492; P = 3.015.

Результаты вычисления вывести на экран в виде таблицы, построенной с помощью символов псевдографики рис. 8.72.

8.26. Рекурсия, множества и текстовые файлы

8.26.1. Рекурсия

Рекурсия – это одно из средств программирования, в котором язык Паскаль традиционно превосходит другие языки программирования. Турбо Паскаль в полной мере позволяет строить рекурсивные алгоритмы. Под рекурсией обычно понимается вызов функции (или процедуры) из тела этой же самой функции (процедуры).

Рекурсивность часто используется в математике, поскольку многие определения математических формул рекурсивны. Как пример можно привести формулу вычисления факториала:

$$n! = \begin{cases} 1, & \text{если } n = 0; \\ n! = n*(n-1)!, & \text{если } n > 0. \end{cases}$$

а также – целой степени числа:

$$x^n = \begin{cases} 1, & \text{если } n = 0; \\ x * x^{n-1}, & \text{если } n > 0. \end{cases}$$

В приведенных формулах для вычисления каждого следующего значения необходимо знать предыдущее. В Турбо Паскале рекурсия записывается так же, как и в формулах. К примеру, функция вычисления факториала, запишется так:

```
Function Fact(n : Word) : LongInt;
Begin
  If n=0 Then
    Fact:=1
  Else
    Fact := n*Fact(n -1)
End;
```

Целая степень числа *x* запишется в ТП аналогично:

```
Function IntPower(x : Real; n : Word) : Real;
Begin   {x - число, возводимое в степень n}
  If n=0 Then
    IntPower:=1
  Else
    IntPower :=x*IntPower(x, n-1)
End;
```

Если в функцию передаётся *n>0*, то происходит следующее. Сначала запоминаются известные значения членов выражения в ветви `Else` (для факториала это *n*, а для степени – *x)*, а для вычисления неизвестных значений вызываются те же функции, но с «предыдущими» аргументами. При этом снова запоминаются (но уже в другом месте памяти – стеке!) известные значения

членов и осуществляются очередные вызовы. Так происходит до тех пор, пока выражение не станет полностью определённым (в наших примерах – это присваивание ветви Then), посля чего алгоритм начинает «раскручиваться» в обратную сторону, извлекая из памяти «отложенные», вычисленные заранее значения. Поскольку при этом, на каждом очередном шаге, все члены выражений уже будут известны, через *n* таких «обратных» шагов мы получим конечный результат.

Несмотря на наглядность рекурсии, во многих случаях эти же задачи более эффективно решаются итерационными методами (при сравнимой скорости вычислений, но с экономией памяти). Однако в задачах грамматического разбора символьных конструкций рекурсия достаточно эффективна.

8.26.2. Множества

Тип “множество” задаётся либо перечислением значений (перечислимый тип), либо отрезком типа, либо именем скалярного типа. Например:

```
Type
  Letter  = Set Of ‘A’..’Z’;
  All     = Set Of 11..88;
  SubColr = Set Of (Red, Green);
  AllLet  = Set Of Char;
```

При определении множеств необходимо учитывать следующие ограничения:

– базисный тип – должен быть любым скалярным типом (помните, что тип Real не является скалярным!);

– максимальное число элементов – не более 256-ти;

– значения элементов из базисного типа должно принадлежать множеству 0..255.

Множества могут вычисляться с применений выражений над множествами. Эти выражения состоят из констант, переменных и операций.

Константы из множеств – это список подмножеств, составляющих множество:

```
['A'..'Z', 'a'..'z', '0'..'9']
```

Переменные типа множество при использовании подчиняются специальному синтаксису – вкладываются в квадратные скобки:

```
Sbyte := [1, 2, 3, 4, 10, 20, 30];
Schar := ['a', 'б', 'в'];
Sdiap := [1..4]; {то ж, что и [1, 2, 3, 4]}
Schar := ['a..п','p..9'];
Empty := []; {Так записывается пустое множество}
```

8.26.3. Операции, применяющиеся к множествам

Над множествами определены следующие операции:

`S1 = S2` {Равенство множеств. Имеет значение `True` если `S1` и `S2` состоят из тех же самых значений независимо от их порядка}

`S1 <> S2` {Неравенство множеств. Равно `True` если хотя бы один из елементов отличается}

`S1 <= S2` {Вхождение `S1` в множество `S2`. Проверка на подмножество. Имеет значение `True`, если все элементы `S1` содержатся в `S2`, независимо от порядка расположения}

`S1 >= S2` {Включение `S2` в множество `S1`. Проверка на надмножество. Имеет значение `True`, если все элементы `S2` содержатся в `S1`, независимо от порядка расположения}

`E in S1` или `E in [..]` {Проверка вхождения (принадлежности) элемента `E` во множество `S1` или принадлежность базовому типу `[..]`}

`S1 + S2` {Объединение множеств. Новое множество содержит все элементы из `S1`, `S2` кроме тех, которые дублируются}

`S1 – S2` {Разность множеств. Новое множество содержит элементы `S1` без елементов `S2`}

`S1 * S2` {Пересечение множеств. Новое множество содержит только элементы, содержащиеся одновременно в `S1` и `S2`}

Например, рассмотрим операции над множествами:

```
[1, 2, 3] = [1, 3, 2]          {Даст значение True }
[1, 2] <> [1], 5 in [0..5]     {Даст значение True}
[1, 2] + [3, 4]                {Даст значение [1, 2, 3, 4]}
[1, 2, 3, 4] – [3, 4]          {Даст значение [1, 2]}
[1, 2, 3, 4] * [3, 4, 5, 6]    {Даст значение [3, 4]}
```

Приведём пример программы, использующей множества при проверке символов, вводимых при нажатии клавиш клавиатуры (рис. 8.81).

8.26.4. Ввод-вывод данных и файловая система MS-DOS

Любой обмен данными требует наличия 3-х компонентов: ***источника информации***, её ***приёмника*** и ***канала связи*** (рис. 8.82).

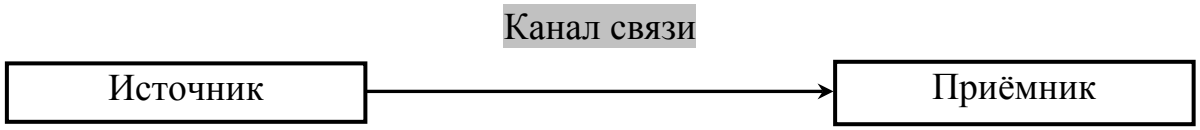


Рис. 8.82. Общий вид процесса обмена данными

```
Program ChekSymbol;{Программа опроса клавиатуры}
Uses Crt;              {с применением множеств}
Type
  Let = Set Of 'A'..#255;
Var
  Ch    : Char;
  Yes   : Let;
  No    : Let;
  YesNo : Let;
Begin
  Yes:=['Y', 'y', 'Д', 'д'];
  No :=['N', 'n', 'Н', 'н'];
  YesNo:=Yes+No; {Объединение множеств}
  Write('Продолжить?(Д/Н)');
  Repeat
    Ch:=ReadKey; {Функция ReadKey возвращает код нажатого
                  символа}
  Until Ch In YesNo; {Проверка на принадлежность Ch
                      множеству допустимых значений
                      ответов пользователя}
  If Ch In No Then
    Halt;
  WriteLn(Ch); {Выводим символ, введённый пользователем}
  WriteLn('Для окончания работы нажмите любую клавишу ');
  Repeat Until KeyPressed;
End.
```

Рис. 8.81. Использование множеств при опросе клавиатуры

В случае обмена данными между программой и периферийными устройствами ПК (дисками, клавиатурой, дисплеем и др.), начальным или конечным пунктом доставки информации каналом обмена данными всегда служит оперативная память ПК (ОЗУ). Соответственно, приёмником или источником данных из/для этого канала в Турбо Паскале определён ***файл*** (рис. 8.83).

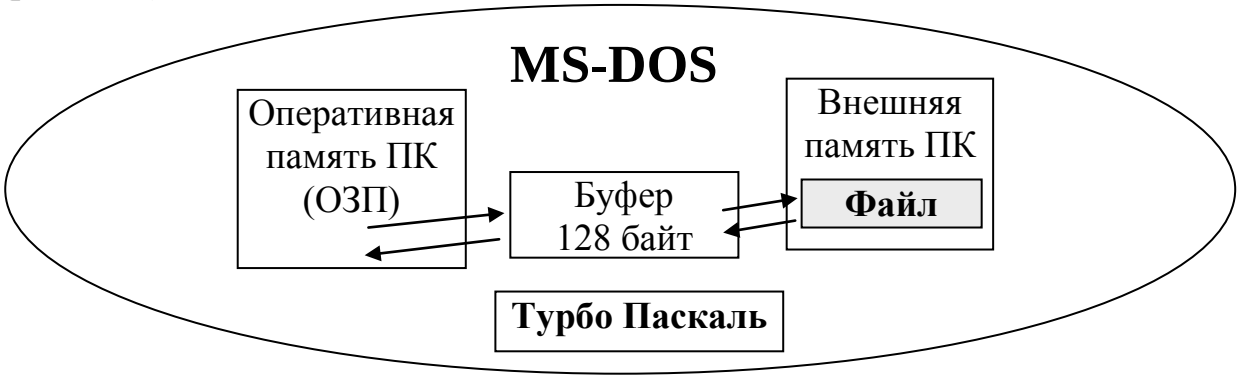


Рис. 8.83. Каналы ввода-вывода в ТП

Все операции обмена производятся определёнными порциями и поддерживаются операционной системой, которая в системной области памяти ПК выделяет специальный буфер для накапливания данных. Это делается для уменьшения количества обращений к внешним устройствам. Размер буфера равен 128 байт, однако может изменяться пользователем в сторону увеличения. При записи в файл вся информация сначала направляется в буфер и там накапливается до тех пор, пока весь объём буфера не будет заполнен. Только после этого или после специальной команды сброса буфера происходит передача данных.

Аналогично, при чтении из файла считывается не столько данных, сколько запрашивается, а сколько поместится в буфер. ***Например, если из файла на диске считывается 4 числа из общего количества 64, то следующие 60 читаются уже из буфера (!).***

Операция вывода данных означает пересылку данных из рабочей памяти (ОЗУ) в файл, а операция ввода – заполнение ячеек памяти данными, полученными из файла.

Файловая система в ТП базируется на двух уровнях представлений: ***логических файлахв*** и ***физических файлах***(см. п.3.2).

8.26.5. Понятие логического файла

Логический файл описывается в программе ТП как переменная файлового типа. После этого она связывается с ***физическим файлом*** MS-DOS и может использоваться для операций ввода-вывода. Так, если мы хотим работать с текстовым файлом `'A:\TEXT.DAT'` , то в программе должны присутствовать следующие строки кода:

```
Var
 f : Text; {Объявляем текстовую переменную f в качестве логического
           файла, а по сути – это признак создания буферной области в
           памяти ПК для размещения фрагментов физического файла}
Begin
  Assign( f, 'A:\TEXT.DAT'); {Связываем физический файл
           'A:\TEXT.DAT' на диске A: с логическим файлом f}
.  .  .
End.
```

После выполнения этих действия, обращения к файлу на диске будут выполняться через файловую переменную `f`. Введение понятия ***логического файла*** позволяет освободить программиста от решения сложных программных проблем самостоятельной организации обмена данными между программой и каждым конкретным внешним устройством, имеющим, естественно, свою собственную систему команд доступа и структуру организации массивов данных разных типов.

8.26.6. Физические файлы в MS-DOS

В операционной системе MS-DOS существуют два вида ***физических файлов*** (пп.2.3, 2.4):

❶ файлы на магнитных носителях (т.е. на флоппи-дисках, винчестерах (жёстких дисках), виртуальных (логических) дисках, оптических дисках и на многих других внешних физических запоминающих устройствах);

❷ файловые модели внешних устройства MS-DOS.

В ТП имена файлов первого типа могут быть строчными константами либо сохраняться в строчных переменных. Например:

```
'C:\PAS\TESTFILE.PAS'
'A:\TEST.TXT'
'..\PRIMER.DAT'
```

Имена устройств MS-DOS имеют фиксированные имена и используются как обычные файлы (табл.8. 51).

Физические файлы-устройства в MS-DOS организуются как ***текстовые файлы*** и для их нормальной работы нужно связывать их имена с текстовыми логическими файлами. Имена физических файлов-устройств записываются также как и у простых текстовых файлов: `'CON'`, `'PRN'` и т.д.

Таблица 8.51

Стандартные имена файлов-устройств MS-DOS

Имя	Файл-устройство	Примечания
`CON`	Консоль (клавиатура и экран)	Ввод с `CON` – это чтение с клавиатуры, а вывод в `CON` – это вывод на экран

`LPT1`
`LPT2`
`LPT3`
	Параллельные порты вывода	Через эти имена файлов производится вывод данных на принтер или другие устройства

`PRN`	Принтер. Синоним имени `LPT1`	Имя для обращения к принтеру, подключённому к порту `LPT1`
`COM1` `COM2`	Последовательные порты	Имена файлов-устройств для ввода-вывода данных через серийные порты коммуникации
`NUL`	Фиктивное устройство	Это бездонный файл, который принимает любые данные, но всегда пустой

Не определена только такая структура данных, как файл в памяти ПК! Любой объявленный ***логический файл*** имеет смысл только после связывания с внешним ***физическим файлом*** или ***файлом-устройством***.

8.26.7. Файловые типы ТП

Турбо Паскаль поддерживает три файловых типа (то есть способов организации хранения разных типов данных на диске):

❶ ***текстовые файлы*** (типа `Text`);

❷ ***компонентные файлы*** (типа `File Of Real, File of Integer`; и др.)

❸ ***бестиповые файлы*** (типа `File`).

Текстовые файлы – это файлы, которые состоят из ASCII-кодов, включая расширенные и управляющие коды. Текстовые файлы организуются построчно. Концом строки является специальный символ **EOL** (**End Of Line** – конец строки), состоящий из двух ASCII-кодов:

–CR = #13 **(Carriage Return** – возврат каретки);

–LF = #10 **(Line Feed** – перевод строки).

Эти символы ***не печатаются на экране при вводе!***. Любую информацию (числовую, символьную или строчную) текстовый файл сохраняет в виде последовательностей текстовых (тип `Char` языка ТП) символов, представляющих её.

Компонентные файлы в отличие от текстовых состоят из машинных представлений чисел и построенных из этих представлений символов и других структур данных. Такие файлы хранят все записываемые данные в таком же виде, как и память ПК. Поэтому с помощью компонентных файлов можно осуществлять обмен данными только между дисками и робочей памятью программы, но нельзя, например, прямо вывести с диска данные непосредственно на экран.

Бестиповые файлы также содержат машинные представления данных. Их отличие от компонентных файлов состоит в том, что они содержат произвольные наборы байтов. Все без исключения файлы обязательно содержат в своём конце специальный код **End Of File** – **EOF**, называемый концом файла.

Для всех типов файлов минимальной единицей хранения информации в них является байт. Принципы роботы со всеми типами файлов едины, хотя и имеют некоторые отличия.

8.26.8. Текстовые файлы

Текстовый файл можно рассматривать как последовательность символов, разбитую на строки специальным маркером. На практике такой маркер представляет собой последовательность из двух символов ASCII: возврат каретки *`chr(код 13)`* (**CR** – Carriage Return) и перевод строки *`chr(код 10)`* (**LF** – Line Feed) (см. Приложение 6). Эти два символа задают стандартные действия по управлению текстовыми файлами. По сути, редактор Турбо Паскаля выводит в своём окне строки текста на основе анализа появления пары кодов (**CR**/**LF**). То есть, в конце каждой строки всегда присутствует пара этих управляющих символов. Они вставляются в текстовом редакторе ТП в конце каждой строки, ***ввод которой завершается нажатием клавиши*** **Enter**. При

нажатии в конце любой текстовой строки на клавишу **Del** текущая строка и следующая объединяются (так как эти два символа вместе удаляются).

Для работы с текстовыми файлами используются специальные процедуры:

`Assign(f; Name:String)` – связывает файловую переменную `f` с полным внешним именем файла на диске, включая и маршрут к нему.

`AssignCRT(Var f:Text)` – связывает файловую переменную с экраном монитора. Определена в модуле `Unit Crt`. Эта процедура необходима в том случае, когда приходится выводить информацию на экран через файловую переменную и необходимо использовать иекущий установленный цвет символов. Если просто использовать процедуру `Assign`, то текст будет виводиться белым цветом на чёрном фоне.

`Append(Var f:Text)` – открывает существующий файл для добавления строк текста. ***Если файл отсутствует на диске, то возникает ошибка ввода-вывода.***

`Rewrite(f)` – создаёт новый текстовый файл, к которому можно лишь добавлять строки. ***Если файл с таким именем уже существует на диске, то он удаляется и создаётся новый.***

`Reset(f)` – используется только к существующему файлу, после чего из этого файла можно только последовательно читать. Когда новый текстовый файл закрывается, к нему автоматически добавляется маркер конца файла `EOF` (**Ctrl+Z**).

`Read(Var f:text, W1[,W2, ...Wn])` – расширение стандартной процедуры чтения `Read`, что позволяет работать со значениями символьного типа, читая информацию из файла (если он задан, иначе производится чтение с клавиатуры) в заданные переменные. Осуществляет чтение из файла `f`, где `W1[,W2, ...Wn]` – переменные стандартного паскалевского типа, в которые и помещаются или символы, или числа, полученные путём интерпретации символов цифр из файла `f`. То есть, строка из двух символов «4» и «6» интерпретируется как число 46 и т.д. Разделителями в строке в этом случае является символ пробела, а между строками – маркер конца строки (**CR**/**LF**).

`ReadLn(Var f:text, W1[,W2, ...Wn])` – действует аналогично процедуре `Read`. Отличие заключается в том, что после чтения данных в переменные, пропускаются все символы, которые остались в данной строке и маркер конца строки. Если в процедуре отсутствует список переменных, то осуществляется переход к следующей строке.***

`Write(f, W1[,W2, ...Wn])` – переводит числа `W1[,W2, ...Wn]` из заданных переменных в последовательность соответствующих символов и записывает их в файл `f`, не разделяя пробелами. ***Поэтому, при необходимости, пробелы и маркеры конца строки необходимо вставлять самостоятельно***. Для символьных, арифметических и строковых переменных в файл выводятся их значения, а для `Boolean` выводится строка `True` или `False` в зависимости от

его значения. Если поле задаёт больше позиций, чем необходимо для вывода, то оно слева дополняется пробелами, иначе выводятся самые правые символы из представления переменной (рис. 8.84).

```
Program WriteFile; {Программа записи/вывода переменных на экран}
Var
  Ch     : Char;
  Bool   : Boolean;
  S      : String;
  I      : Integer;
  R1, R2 : Real;
Begin
  Ch:='X';
  Bool:=1>2;
  S:='abcd';
  I:=12;
  R1:=1.25;
  R2:=R1;
  Write('Результат:')
  Write('!',Ch:3,Bool:7,S:8,I:4,R1:18,R2:12:4,'!');
  ReadLn
End.
```

```
Результат:
!  X  False    abcd  12  1.2500000000E+00      1.2500!
```

Рис. 8.84. Вывод информации оператором `Write`

В следующем примере (рис. 8.85) на диске **D** в корневом каталоге формируется (записывается) текстовый файл. Данные из этого файла, в свою очередь, могут быть в будущем считаны для использования в программе пользователя.

8.26.9. Функции для работы с файлами

Для обработки текстовых файлов имеется ещё ряд полезных функций:

`EoF[(f:Text)]:Boolean` – возвращает `True`, если следующий за последним прочитанным символом имеет маркер конца файла. Если файловая переменная не определена, то предполагается стандартный файл ввода.

`Eoln[(f:Text)]:Boolean` – возвращает `True`, если следующий за последним прочитанным символом является маркер конца строки. Если файловая переменная не определена, то предполагается стандартный файл ввода. Как правило, если `EoF(f)` – истина, то и `Eoln(f)` – истина (т.е. `True`).

`SeekEoln[(f:Text)]:Boolean` – аналогична `Eoln`, однако до проверки на маркер конца строки удаляет все следующие пробелы и символы горизонтальной табуляции.

`SeekEoF[(f:Text)]:Boolean` – аналогична `Eoln`, однако до проверки на маркер конца файла удаляет все последующие пробелы, символы горизонтальной табуляции и **CR**/**LF**.

```
Program FileWorks; {Программа формирования тектового файла с данными}
Uses Crt;
Const
  Bar : Char = ' '; {Пробел – #32}
  lf  : Char = #10; {Line Feed – перевод строки}
  cr  : Char = #13; {Carriage Return – возврат каретки}
Var
  f    : Text; {Объявляем логическую переменную f текстового типа}
  n, i : integer; {Целые переменные для записи в файл}
 Begin
  ClrScr;
  Assign (f, 'd:\ee.dat'); {Связываем логическую переменую f с
                            физическим файлом на диске для ввода
                            (записи) текста}
  ReWrite(f); {Создаём новый текстовый файл, к которому можно только
               прибавлять строки}
  i:=1; {Присваиваем значение 1 целой переменной i}
  n:=2; {Присваиваем значение 2 целой переменной n}
  Write(f, i, Bar); {Записываем в буфер 1-цу и пробел}
  Write(f, n, cr, lf); {Добавляем 2 й маркер конца строки CR/LF}
  Write(f, n); {Добавляем 2 в следующую строку}
  Close(f); {Закриваем файл. При этом информация из буфера записывается в
             файл на диске, автоматически добавляется маркер конца файла
             EOF, для файла записываются следующие данные: время записи,
             длина в байтах, тип файла(архивный, скрытый и т.д.). Теперь им
             можно пользоваться}
End.
```

Результат: у кореневому каталозі диску `d:\` з'явився файл `ee.dat`, який містить два рядки:

```
1 2
2
```

Рис. 8.85. Програма формування текстового файлу з даними

Пример. Разработаем программу, которая запрашивает в диалоге выходное устройство (это может быть консоль, принтер или дисковый файл) и выводить на него файл, имя котого тоже задаёт пользователь. Программа заканчивает работу, когда вместо буквы, задающей устройство (`C`, `P` или `D`), пользователь нажмёт клавишу **Esc**. При задании имени входного файла программа проверяет его присутствие на диске. При выводе файла можно приостановить работу программы, нажатием клавиш **Ctrl+S**. Для продолжения вывода – нажмите любую клавишу (рис. 8.86).

```
Program SelectFile; {Программа запроса устройства и вывода файла}
Uses Crt;
Var
  F : Text;
  S : String;
Function SelectDevice(Var F : Text) : Boolean;
Var
  Ch    : Char;
  Fname : String;
Begin
  HighVideo; {Включение яркости цвета символов}
  Write(‘ C’);
  LowVideo; {Выключение яркости цвета символов}
  WriteLn(‘. Вывод файла на экран’);
  HighVideo;
  Write(‘ P’);
  LowVideo;
  WriteLn(‘. Вивод файла на принтер’);
  HighVideo;
  Write(‘ D’);
  LowVideo;
  WriteLn(‘.Вивод файла на диск’);
  HighVideo;
  Write(‘ Esc’);
  LowVideo;
  WriteLn(‘ - закончить работу.’);
  WriteLn;
  Write(‘ Ваш выбор : ’);
  Repeat
    Ch:=UpCase(ReadKey);
  Until Ch In [‘C’, ‘P’, ‘D’, #$18];
  SelectDevice:=Ch=#$18;
  HighVideo;
  If Ch=#27 Then
    WriteLn(‘ Esc’)
  Else
    WriteLn(Ch);
  LowVideo;

  Case Ch Of
    ‘C’ : Assign(F, ‘Con’);
    ‘P’ : Assign(F, ‘Prn’);
    ‘D’ : Begin
            Write(‘Имя выходного файла (с путём!) :’);
            ReadLn(FName);
            Assign(F,FName);
          End;
  End; {Case}
```

Рис. 8.86. Вывод файла на разнообразные устройства

(продолжение программы)

```
  If Ch In ['C', 'P', 'D'] Then
    ReWrite(F);
End;  {Func SelectDevice}

Procedure PrintFile(Var F1 : Text);
Var
  FIn : Text;
  S   : String;
  Ch  : Char;

Procedure GetInputName(Var F : Text);
Var
  Fname  : String;
  IOCode : Word;

Begin

  Repeat
    Write('Задайте имя входного файла');
    HighVideo;
    ReadLn(FName);
    LowVideo;
    Assign(F, FName);
    {$I-} {Откл. аварийное завершение при возникновении ошибки I/O}
    Reset(F);
    {$I+} {Возобновляем режим аварийного отключения}
    IOCode:=IOResult;
    If IOCode>0 Then
      WriteLn(^G^G' Файл ',Fname,' не найден!');
  Until IOCode=0;
End; {Proc GetInputName}
Begin {Начало Proc PrintFile}
  GetInputName(FIn);
  While Not Eof(FIn) Do
    Begin
      If KeyPressed Then {Клавиша нажата}
         Begin
           Ch:=ReadKey;
           If Ch=^S Then {Необходима пауза}
              Ch:=ReadKey; {Продолжим работу по любой клавише}
         End;
      ReadLn(Fin,S);
      WriteLn(F1,S);
    End;
    Close(FIn);
    Close(F1);
End; {Proc PrintFile}
Begin {Початок основної програми}
  While Not SelectDevice(F) Do
    PrintFile(F);
End.
```

Рис. 8.86. Вывод файла на разнообразные устройства (окончание)

8.26.10. Решение задачи с использованием рекурсии, множеств и текстовых файлов

Теперь, когда мы с Вами разобрались с такими элементами языка Турбо Паскаль как, рекурсии, множества и текстовые файлы, решим комплексную задачу на использование этих понятий и разработаем соответствующую программу.

Итак, имеется следующая задача[1]. В компьютерной сети используется некоторое множество разных протоколов. Каждый узел (компьютер) поддерживает несколько протоколов. Все узлы между собой связаны. Пакет данных может быть передан из узла в узел только в случае, если у этих двух узлов имеется общий протокол. Сначала пакет был послан первому узлу. В задаче требуется найти все узлы, к которым дойдёт этот пакет, и вывести их количество. (Всего узлов может быть не более 100, а протоколов не более 50). Пример сети приведен на рис. 8.87.

Входные данные. Файл `e.dat`, содержит текстовые данные следующего формата:

– первая строка – количество узлов ***n***;

– каждый из следующих ***n*** строк – количество поддерживаемых протоколов и номера этих протоколов.

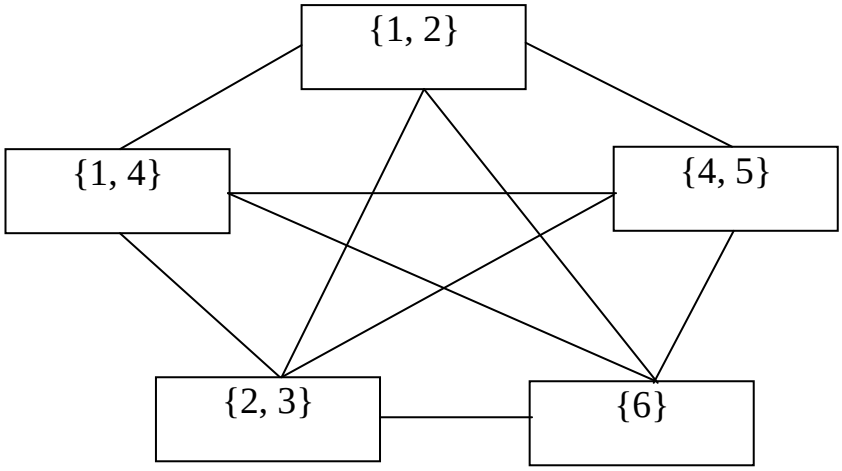


Рис. 8.87. Схема сети

Пример данных для ввода:

```
5
2 1 2
2 1 4
2 2 3
1 6
2 4 5
```

[1] **Бобак И. Думай рекурсивно // Мой компьютер. – 2001. – № 23. – С. 36-39. (ibobak@torba.com)**

Выходные данные. Количесво узлов, которые могут получить пакет, посланный сначала первому узлу. Пример (для приведенных выше входных данных):

4

Ниже приведен алгоритм и программа решения поставленной задачи.

```
Program Recursion; {Программа решения сетевой задачи}
Uses Crt;
Type
  TProtocolSet = Set Of 1..50; {TProtocolSet – тип-множество}
Var
  f : Text; {Описываем текстовый файл}
  A : Array[1..100] Of TProtocolSet;{Массив, элементами которого
                        является множество TprotocolSet, содержащее
                        номера протоколов для каждого узла}
  was : array[1..100] of Boolean; {Статус обработки узла. Все
                        элементы его должны содержать значения False.
                        Узлы обходим, начиная с первого. Сначала войдём в
                        первый узел. Далее из него войдём во все узлы,
                        имеющие с ним общий протокол. Потом для каждого
                        из них сделаем то же самое. Те узлы, где мы уже
                        побывали, будем отмечать в булевом массиве
                        значениями True.}
  n   : integer; {Количество узлов}
  cnt : integer; {Результат – количество узлов, к которым дойдёт пакет}

Procedure Rec(const p : Integer);{=* Рекурсивная процедура,
                        определяющая количество узлов cnt (глобальная
                        переменная), до которых дойдёт пакет *=}
{Суть процедуры Rec заключаетсяв распространении пакета. Сделав вызов
Rec(1), мы вызовем её рекурсивно для узлов, к которым пакет может прийти с
выходного узла. Далее, для последних будет сделан рекурсивный вызов процедуры
Rec с номерами узлов, в которые пакет может попасть во вторую очередь и т.д.
Для нашего случая будет: 1, 2→1, 4; 1, 2→2, 3; 1, 4→4, 5}

var
  i : Integer; {Параметр цикла for}
Begin
  if was[p] then
    exit;          {Процедура exit завершает работу программного блока}
  was[p] := True; {Отмечаем узел, который посещён}
  inc(cnt);  {Встроенная процедура ТП inc увеличивает на 1 значение
аргумента}
  for i:=1 to n do {Цикл обхода узлов (n – глобальная переменная)}
if A[i]*A[p] <> [] then { A[i]*A[p] – пересечение множеств.
                        Результатом является множество, состоящее только
                        из тех элементов двух множеств, которые
                        содержатся одновремено в каждой из них. [] –
                        пустое множество. В нашем случае – это отсутствие
                        общего протокола передачи данных у двух узлов}
    Rec(i); {Если узлы p и i имеют общий протокол, значит данные дойдут
                        до узла i. Тогда рекурсивно вызываем процедуру
                        Rec}
End; {Proc Rec}
```

```
Procedure Init; {=*= Процедура считывания данных =*=}

Var
  i, j, sc, p : Integer; {i, j – параметры циклов; sc – количество
                          протоколов в узле; p – номер протокола}
Begin
  FillChar(A, SizeOf(A), 0); {Встроенная процедура ТП, для
                          заполнения массива (A) числом (SizeOf(A), где
                          SizeOf – встроенная функция, возвращающая
                          размер типа с именем A ) и значением (0)}
  cnt := 0; {Инициализация cnt}
  for i:=1 to n do     {Цикл по числу n узлов сети, прочитанных в
                          основной программе из файла e.dat}
    begin
      Read(f, sc);      {Читаем из файла e.dat количество протоколов
                          sc на узле i}
      for j:=1 to sc do {Цикл по числу sc протоколов}
       begin
        read(f, p); {Читаем из файла e.dat номера протоколов}
         include(A[i], p);     {Включаем встроенной процедурой
                          include номера протоколов в i-тое множество
                          TProtocolSet}
        end;
    end
End; {Proc Init}

Begin  {=========================Начало основной программы}
   ClrScr; {Очищаем экран}
   Assign (f, 'e.dat'); {Устанавливаем связь между логическим файлом
                          f и физическим файлом e.dat}
   Reset(f); {Открываем логический файл f для чтения}
   Read(f, N); {Читаем из файла e.dat количество узлов сети N}
   Init; {Вызываем процедуру считывания данных о протоколах из файла
                          e.dat}
   Rec(1); {Передаём пакет в первый узел. Рекурсивно он распространяется во
                          все остальные, если это возможно}
   WriteLn(cnt); {Печатаем результат – количество узлов}
   Close(f); {Закриваем физический файл e.dat}
End.        {КОНЕЦ ОСНОВНОЙ ПРОГРАММЫ}
```

При использовании рекурсии необходимо заботиться о том, чтобы вовремя из неё выйти. Рекурсия в этом примере не бесконечна, выход из неё предусмотрен: мы проверяем факт посещения (обработки) узла оператором:

```
if was[p] then exit;
```

и если узел уже был посещён, программа выходит из рекурсии.

Упражнения

1. Создайте рекурсивную программу «Ханойская башня». Условие задачи следующее: имеется доска с тремя штырьками (рис. 8.88). На первом из них нанизано несколько дисков увеличивающегося к низу диаметра. Необходимо

расположить эти диски в таком же порядке на третьем штырьке. Диски можно перекладывать только по одному, а класть больший на меньший нельзя.

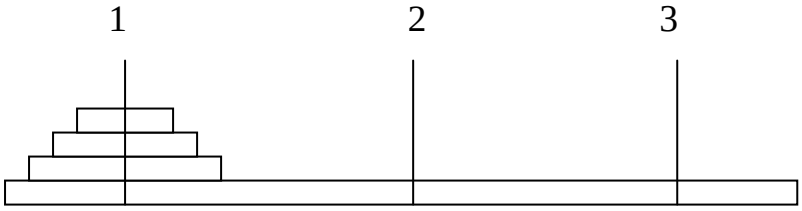


Рис. 8.88. Вид игры «Ханойская башня»

Алгоритм решения головоломки следующий:

1.1. Переместить верхние n-1 дисков на 2-ой штырёк.

1.2. Нижний диск первого штырька переместить на третий.

1.3. Переместить n-1 дисков на третий штырёк, используя первый в качестве вспомагательного.

1.4. Повторять, пока на третьем штырьке не будет сформирована новая пирамида.

Конечная задача сводится к двум основным:

–нескольких о переносе (n-1)-го диска;

–одной задачи о переносе одного диска.

Естественно, что для n=1 необходимо всего одно перемещение.

2. Разработать программу с испоьзованием множеств для решения следующей задачи: пользователю предлагается ввести одну из букв латинского алфавита: A, L, P или T, причём всё равно на каком регистре – прописном или строчном. Если то, что введёт пользователь, будет отлично от каждой из введённых букв, программа должна напечатать сообщение, что приглашает к повторному введению.

3. В текстовом файле содержатся длины сторон 20 прямоугольных параллелепипедов. Вычислить объёмы этих параллелепипедов. Результаты расчётов записать в файл в виде таблицы, содержащей следующие столбцы:

–длина первой стороны параллелепипеда;

–длина второй стороны параллелепипеда;

–длина третьей стороны параллелепипеда;

–объём параллелепипеда.

Позаботьтесь о соответствующих заголовках для столбцов.

Файл с выходными данными сформируйте с помощью текстового редактора.

4. Трансформируйте текст программы о доставке пакетов в сети (рис. 8.87) так, чтобы в процедуре глобальные параметры передавались через формальные параметры.

8.27. Записи, ссылки, динамические переменные и структуры

8.27.1. Тип “запись” (record) и оператор присоединения with

Для компактного представления комбинаций разнотипных данных в Турбо Паскале их можно объединять в структуры-записи. Каждая запись состоит из объявленного числа полей и определяется конструкцией:

```
Var
  InfRec : Record;
              Поле1 : Тип_поля1;
              Поле2 : Тип_поля2;
                . . .
              ПолеN : Тип_поляN
           End;
```

Где `Тип_поляN`, это некоторый тип данных ТП. Обратите внимание на то, что после служебного слова `Var` перед описанием типа данных `Record` ставится знак (:), а после `Type` – знак (=) (смотри следующий пример).

Если типы нескольких полей совпадают, то имена полей могут быть просто перечислены, например – `x,y`:

```
Type
PointRecType = record x,y : Integer end;
Var
Point : PointRecType;
Px, Py : Integer;
. . .
{Обращение к полям записи:}
Px := Point.x;
Py := Point.y;
. . .
```

Так как имена полей «спрятаны» в средине типа, то они могут дублировать «внешние» переменные и поля в других описаниях записей:

```
Type
  PointRecType = record x, y : integer end;
  ColorPointType = record
                     x, y : integer;
                     color : word
                   end;
Var
  X, Y : integer;
  Point : pointRecType;
  ColorPoint : ColorPointRecType;
. . .
```

Обратите внимание также на то, что в этом примере `x`, `Point.x` и `ColorPoint.x` – совсем разные значения.

Например, Вы можете разработать свои пользовательские типы данных (рис. 8.89). Связь значения в операторе вариантной части записи с соответствующими значениями компилятором никак не отслеживаются. Запись с вариантами может использоваться, например, для чтения полей разных типов из файлов базы данных в одну и ту же область памяти.

```
Type
  OS       = (MS_DOS, CPM, MPM, Unix);
  CPU      = (I8088, I8086, I80186, I80286, I80386);
  Computer = Record
               OperatingSystem : Array[1..4] of OS;
               Processor       : CPU;
               Price           : Real;
               MadeIn          : String[80];
             End;
Var
  Users : Array[1..50] of Computer;
Type
  List = record
           Name     : String[40];
           BirthDay : Word;
           Case Citizen : Boolean Of
             True : (BirthPlace : String[40]);
             False: (Country  : String[20];
                     Port     : String[18];
                     ExitDate : Word;
                     EntryDate: Word);
         end;
```

Рис. 8.89. Пример конструирования элементов типа “запись”

Переменная типа "запись" может участвовать только в операциях присваивания. Но поле записи может участвовать во всех операциях, которые относятся к типу этого поля. Для доступа к полям записи применяется квалификационное имя (рис. 8.89):

```
Users[3].Processor
```

Для облегчения работы с полями записей вводится оператор присоединения – `With`:

`With` <Имя_Переменной_Записи> `do` <Оператор>;

Внутри оператора `With` (он может быть составным) обращения к полям записи производится прямо (рис. 8.90). Внутри области действия оператора присоединения `With` могут указываться и переменные, не имеющие отношения к записи, однако необходимо придерживаться следующих правил (рис. 8.91).
В случае, если одно из полей записи само является записью и снова содержит поля записи, оператор присоединения можно распространить на несколько полей внутрь, перечислив их через запятую. Однако внутри тела оператора можно обращаться только к последним полям:

```
  With ИмяЗаписи.Поле_Запись do
  Begin
     {Обращения к полям Поле_Запись, то есть к тем, которым предшествовала
       конструкция "ИмяЗаписи.Поле_Запись"}
  End; {With}
```

```
{Два эквивалентных способа обращения к полям записи рис. 8.89}
{Перший:}
Users.OperatingSystem:=MS_DOS;
Users.Processor      :=I80286;
Users.Price          :=2250;
Users.MadeIn         :='Taiwan';
{Другий:}
With Users Do
  Begin
    OperatingSystem:=MS_DOS;
    Processor      :=I80286;
    Price          :=2250;
    MadeIn         :='Taiwan';
  End;
```

Рис. 8.90. Два эквивалентных способа обращения к полям записи

```
Program Main;
Var
  X, y, z : integer;
  RecXY : record x, y : Integer end;
Begin
  X := 10;  Y := 20;
  With RecXY do
    Begin
      X := 3.14 * Main.X / Z; {«Main» - квалификатор}
      Y := 3.14 * Main.Y / Z; {для «развязки» совпдающих
                               идентификаторов. Для Z  не требуется}
    End; {With}
  . . .
End.
```

Рис. 8.91. Переменные, не имеющие отношения к записи в операторе `With`

8.27.2. Система адресации MS-DOS

Адресуемое пространство памяти в MS-DOS организовано сегментами, представляющими собой последовательные блоки памяти по 64Кб каждый. Если Вам известен сегмент, то дальнейшее уточнение адреса происходит по его смещению, то есть номеру байта от начала сегмента. Таким образом, любая ячейка адресуемого пространства памяти определяется парой чисел:

<СЕГМЕНТ> : <СМЕЩЕНИЕ>,

который занимает четыре байта: 2 байта <СЕГМЕНТ> и 2 байта <СМЕЩЕНИЕ>. Такой способ позволяет адресовать большее пространство оперативной памяти меньшими числами. Чтобы Вам было проще это представить, приведём такой пример. В некотором районе построенно 20 домов по 99 квартир. Необходимо найти способ адресации, чтобы не выходить за границы двузначных чисел, ибо сквозная нумерация всех квартир требует возможности представлять максимальное число:

$$20 \times 99 = 1980,$$

которое занимает четыре разряда. Выход из такой ситуации состоит в двойной адресации:

<ДОМ> : <КВАРТИРА>

Тогда любую квартиру Вы сможете найти по номеру дома и номеру квартиры в этом доме. Например, дом 10, квартира 67:

<10> : <67>,

в то время, как при сквозной нумерации для хранения адреса объекта понадобилось бы три разряда: $10 \times 67 = 670$.

8.27.3. Тип Pointer

Основным механизмом для организации динамических данных в ТП является выделение в специальной области памяти, называемой «куча», участка (блока) необходимого размера и сохранения адреса начала этого участка в специальной переменной в формате <СЕГМЕНТ>**:**<СМЕЩЕНИЕ>.

Условимся называть далее ***указателем*** переменные, которые имеют обобщённый тип `Pointer` – ***указатель***. То есть можно объявлять переменные, значениями которых будут адреса ячеек памяти:

```
Var
  P : Pointer; {Переменная-указатель}
```

Значение этого типа занимают 4 байти памяти и содержат адрес начала любого объекта ТП. Адрес сохраняется как два слова, то есть две переменные типа `Word` (каждая из них занимает в памяти 2 байта): одно из них задаёт ***сегмент***, а другое – ***смещение***. Значение ***указателя*** не может быть выведенным на экран или на печать. Его нужно предварительно расшифровувать.

Имейте ввиду, что компилятор ТП всегда должен знать, какой объект адресуется, чтобы корректно его обрабатывать. Вспомните, что переменная типа `Integer` занимает в памяти 2 байта, переменная типа `LongInt` – 4 байта, переменная типа `Real` – 6 байтов, переменная типа `Byte` занимает в памяти 1 байт и т.д (см. Приложение 7).

Ещё одна из тонкостей работы в ТП состоит в том, что Вы должны соблюдать все правила «правописания» его конструкций. К примеру, когда Вы пишете письмо к близкому человеку, то отступаетете все необходимые абзацы,

грамотно формулируете фрази, где необходимо ставите запятые, тире, точки и т.д. Так вот, программа, которую Вы пишете для транслятора, тоже должна быть грамотно написана, чтобы он смог её правильно понять...

8.27.4. Средства работы с адресами

Для начала рассмотрим функции для работы с адресами разных объектов ТП (табл.. 52).

Таблица 52

Функции для работы с адресами разных объектов ТП

Функция : Тип	Возвращаемое значение
`Addr(X) : Pointer`	Ссылка на начало объекта `X` в памяти
`Seg(X) : Word`	Сегмент, в котором сохраняется объект `X`
`Ofs(X) : Word`	Смещение в сегменте для объекта `X`
`Ptr(S, 0 : Word) : Pointer`	Ссылка на место в памяти, которое задано значениями смещения 0 и сегмента `S`
`SizeOf(X) : Word`	Размер объекта `X` в байтах
Операция `@X : Pointer`	Возвращает ссылку на начало объекта `X` в памяти (аналог функции `Addr`)

Функции `Addr(X)`, `Seg(X)` и `Ofs(X)`, а также оператор `@` возвращают адрес объекта `X` или компоненты этого адреса. Под переменной `X` можно понимать любой объект: переенные встроенных типов ТП, пользовательских типов, объекты, процедуры и функции (но не константы).

Функция `Addr(X)` и оператор `@` возвращают значения типа `Pointer` – адрес объекта `X`. Их действие одинаково:

```
Var
  X : String;
p, q : Pointer;
 . . .
     p:=Addr(X);
     q:=@X;  {Теперь p=q , то есть их адреса равны: они указывают на один
                 и тот же объект}
```

Как Вы уже знаете, значения типа `Pointer` не может быть выведено на экран. Но поскольку этот тип состоит из двух слов (`Word`), хранящих сегмент и смещение, их можно вывести поодиночке, используя функции `Seg` и `Ofs` (обе типа `Word`):

```
WriteLn('Сегмент ', Seg(p), ' смещение ', Ofs(p));
```

Функция `Ptr(Seg, Ofs : Word)` выполняет противоположную функции `Addr(X)` работу: она организовывает ссылку на место в памяти которое определено заданным сегментом и смещением. Необходимость в такой функции может возникать, когда требуется наложить динамическую структуру на системную область памяти. Так, например, известно, что образ текстового

экрана начинается с адреса $B000:$0000 и занимает 4 000 байт (цветной и чорно-белый режимы, 80 столбцов на 25 строк), но можно “наложить” на него некоторую структуру, например, массив, используя ссылку на такой массив и функцию `Ptr` (рис. 8.92):

```
Type
  VideoArray = Array[0..3999] of Byte;
Var
  V : ^VideoArray; {Ссылка на структуру}
Begin
  V:=Ptr($B000,  0); {Далее V^[i] обращается непосредственно к
                      ячейкам видеопамяти в текстовом режиме}
 . . .
End.
```

Рис. 8.92. Использование функции `Ptr(Seg,Ofs : Word)`

Функция `SizeOf(X) : Word` возвращает объём в байтах, который занимает объект `X`. Причём `X` может быть не только переменной, но и идентификатором типа (рис. 8.93):

```
Program ObjectSizeOf;
Type
  XType = Array [1..10,1..10] of Byte;
Const
  L : LongInt = 123456;
Var
  X : String;
Begin
  WriteLn('Xtype =', SizeOf(XType):5,
          '  L = ', SizeOf(L):5,
           '  X = ',SizeOf(X));
  ReadLn
End.
```

Результат работ программы:

```
Xtype  =   100   L =    4    X =   256
```

Рис. 8.93. Використання функції `SizeOf(X) : Word`

Значение `SizeOf(`строка`)` всегда даёт максимальное значение длины строки. Реальное значение даёт функция Legth.

8.27.5. Ссылочные переменные

Все ссылочные переменные имеют однаковый размер, равный 4-м байтам, и содержат адреса начала участка оперативной памяти в котором размещена конкретная динамическая структура данных (Integer, Array, Record и т.д.), например:

```
 . . .
Var
  J : ^Integer; {Ссылочная переменная, указывающая на целое значение}
  I : Integer; {Целая переменная}
 . . .
   J := 200;
   I := 200;
 . . .
```

Разница между целой переменной `I` (тип Integer) (рис. 8.94, а) и ссылочной переменной `J` (^Integer) на целую переменную заключается в следующем (см. рис. 8.94, б):

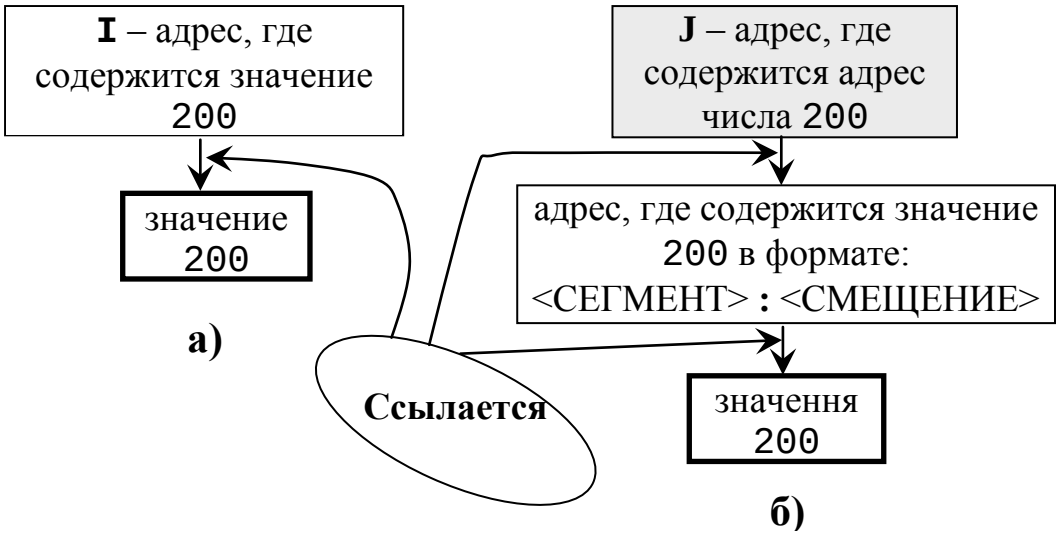


Рис. 8.94. Содержимое данных разного типа I и J

Чтобы ссылка ни на что не указывала, ей присваивается значение **`Nil`**:

```
J := Nil; {или nil}
```

Это определённая константа типа `Pointer`, соответствующая адресу `0000:0000`.

Подытоживая всё вишесказанное можем сделать вывод, что для определения ссылочной переменной в ТП необходимо описать её как ссылочный тип:

```
Type
   ИмяСсылочногоТипа = ^ИмяБазовогоТипа;
    {Где ИмяБазовогоТипа - любой идентификатор типа (Real,
     Integer и т.д.), это необходимо компилятору ТП для
     организации работы программы}
```

В результате этого определения создаваемые затем ссылочные переменные будут указывать на объекты базового типа, определяя тем самым динамические переменные базового типа:

```
Type {БАЗОВЫЕ ТИПЫ ЯЗЫКА ТУРБО ПАСКАЛЬ}
  DimType = Array [1..10000] of Real; {Массив}
  RecType = record ... end; {Запись}
   { ССЫЛОЧНЫЕ ТИПЫ}
  IntPtr = ^Integer; {Ссылка на целое значение}
  Dim Ptr = ^DimType; { Ссылка на массив данных}
  RecPtr = ^RecType; { Ссылка на запись}
  XXXPtr = Pointer; { Ссылка «вообще» - указатель}
```

8.27.6. Операция разименования

Суть этой операции состоит в переходе от ссылочной переменной через адрес, на который она ссылается к значению, на которое она указывает. При этом следом за ссылочной переменной указывается символ "^":

```
 . . .
Var
  I, J : ^Integer;
 . . .
I:=2;
J:=4;
J^:=I^; {Копируем содержимое I в J. Сейчас J указывает на значение 2
                    (рис. 8.95, а)}
J:=I; {Или можно так: идентификаторы I и J теперь тоже указывают на
                    значение 2 (рис. 8.95, б)}
```

Однако эти операции выполняются компилятором языка ТП по разному (рис. 8.95, а, б).

Ячейка со значением 4 превращается в так называемый «мусор», поскольку к ней теперь нет доступа.

Ссылочные переменные и указатели совместны между собой по типу, то есть нет ошибки в присваивании:

```
DimPtr := RecPtr;
```

Однако после разименования начинается контроль типов объектов по определённым адресам:

```
DimPtr^ := RecPtr^; {Даёт ошибку!!!}
```

Разименованные ссылки на структуры (массивы) индексируются или разделяются на поля записи обычным образом:

DimPtr^[i] – доступ к элементу I динамического массива
RecPtr^.Поле – доступ к полю динамической записи

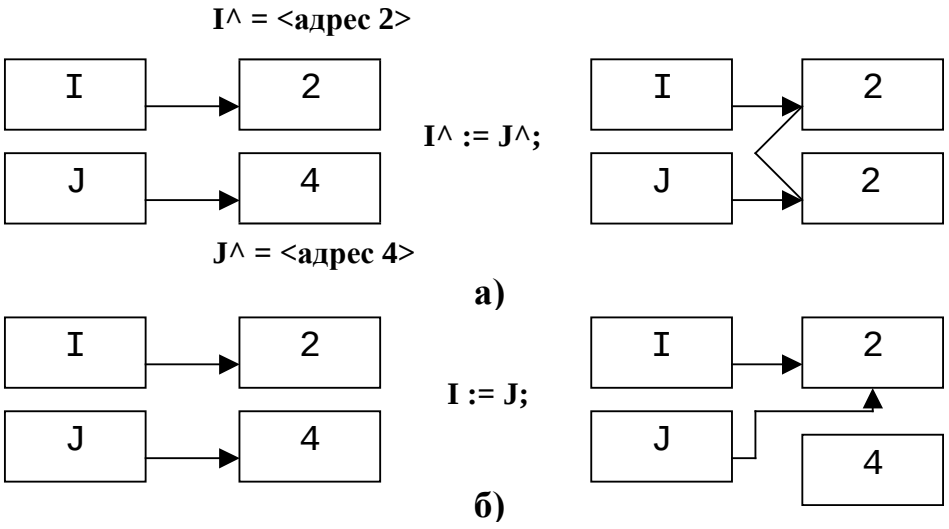


Рис. 8.95. Примеры операций разименования

8.27.7. Размещение динамических переменных. Процедуры New и GetMem

Размещение динамических переменных в ТП выполняется процедурами `New(Var P: Pointer)` и `GetMem(Var P: Pointer)` (табл. 52).

Таблица 52.

Процедуры размещения динамических переменных

Процедуры и функции	Назначение
`New(Var P : Pointer)` или `New(ТипСсылка) : Pointer`	Отводит место для хранения динамической переменной `P^` и присваивает её адрес ссылке `P`
`GetMem(Var P: Pointer; Size : Word)`	Отводит место в `Size` (байт в куче) (Heap), присваивая адресу его начала указатель (ссылку) P

Как Вы уже знаете, в ТП имеется специальный не типизированный указатель `Pointer`. Он объявляется стандартным образом:

```
 . . .
Var
      X : Pointer;
 . . .
```

Не типизированному указателю может быть назначено значение любого типизированного указателя или наоборот. Он выполняет роль своеобразного буфера для сохранения адреса динамической переменной любого типа. Пусть имеем описание:

```
Type
   IntPtr = ^Integer; {Ссылка на целое значение}
Var
  P : IntPtr; {Ссылочная переменная типа IntPtr, определяющая ссылку
                на целое число}
```

Тогда при вызове в следующем виде:

```
New(P);
```

Или:

```
P:=New(IntPtr);
```

В куче выделится блок памяти размером `SizeOf (Integer)` то есть 2 байта и адрес первого байта этого блока запишется в `P`. После выполнения `New` можно уже ссылаться на динамическую переменную `P^`.

Кроме того, вызов `New (P)` можно заменить на вызов `GetMem (P, SizeOf (ИмяБазовогоТипа_P))`, где `SiZeOf (X)` – стандартная функция ТП, возвращающая размер в байтах базового типа.

Процедура `GetMem` предназначена для выделения памяти указателям:

```
Var
  P : Pointer; {Объявляем указатель}
Begin
  GetMem (P, 4*1024); {Теперь P указывает на блок памяти в куче
                       размером 4 Кб.
                       P^ не имеет типа, однако содержит 4096
                       байтов}
 . . .
End.
```

8.27.8. Освобождение динамических переменных. Процедуры `Dispose` и `FreeMem`

Для освобождения памяти, занимаемой динамической переменной `P`, используется процедура `Dispose (Var P: Pointer)`. Эта процедура работает только с типизированными ссылочными переменными (табл.. 53). Для корректной работы программы Вы всегда должны выполнять вызов `Dispose` парными вызовам `New`. После вызова процедуры значение ссылки `P` не определено, как и значение разименованной переменной `P^`.

Таблица 53.

Процедуры освобождения динамических переменных в ТП

Процедуры и функции	Назначение
`Dispose (Var P: Pointer)`	Уничтожает связь, созданную ранее `New`, между ссылкой P и значением, на которое она ссылается
`FreeMem (Var P: Pointer; Size : Word)`	Освобождает `Size` байт в куче, начиная с адреса, записанного в указателе (ссылке `P`)

Для освобождения непрерывных участков памяти необходимо использовать процедуру

```
FreeMem (Var P : Pointer; Size : Word)
```

Вызовы `FreeMem`, так же как и `Dispose`, должны быть парными вызовам `GetMem`. Значения ссылочной переменной `P` после вызова `FreeMem` считаются неопределёнными.

8.27.9. Объединяем вместе понятия Record и динамических переменных: решение задачи по созданию динамических структур типу "очередь"

Наиболее важным аспектом программирования с использованием динамических структур данных является возможность программной реализации стековых структур, односвязных и двусвязных списков, очередей, двоичных деревьев и т.д. Для программной поддержки всех этих структур можно использовать массивы, но при ближайшем рассмотрении становится очевидным, что при этом необходимо описывать достаточно большие массивы с определённой избыточностью, либо, в случае описания заведомо малых массивов, область прикладного применения программы будет существенно ограничена. Альтернативой массивам служат указатели в сочетании со структурой `Record`, в которую включают информационную часть и указатель на следующую структуру `Record`. Их преимущество заключается как в том, что они позволяют создавать динамические структуры необходимой размерности, так и в том, что мы получаем возможность оперировать большими объектами с помощью всего лишь 4-байтных указателей.

8.27.10. Логическая структура очереди FIFO

Очередью *FIFO* (First In – First Out – "первым пришёл – первым исключается") называется такой последовательный список с переменной длиной, в котором включение элементов выполняется только с одной стороны списка (эту сторону часто называют ***концом*** или ***хвостом очереди***), а

исключение – с другой стороны (называемой ***началом*** или ***головой очереди***). Те самые очереди, которые возникают у прилавков и кассс, являются типовым бытовым примером очереди FIFO.

Основные операции над ***очередью*** – это включение и исключение элементов, определение размера очереди, очистка, не разрушающее чтение.

Очереди строятся на базе ***линейных списков***. Представление ***очереди*** с помощью ***линейного списка*** позволяет достаточно просто обеспечить любые необходимые операции обслуживания очереди. Особенно это удобно, когда число элементов в очереди сложно предвидеть. Линейные списки также находят широкое применение в приложениях, где непредсказуемы требования к размеру памяти, необходимой для сохранения данных; имеется большое число сложных операций над данными, особенно включений и исключений.

8.27.11. Связанные линейные списки

Списком называется упорядоченное множество, состоящее из переменного числа элементов, к которым применяемы операции ***включениня*** и ***исключения***. ***Список***, отражающий отношения соседства между элементами, называется ***линейным***. Если ограничения на длину списка не допускаются, то список представляется в памяти в виде динамичной связанной структуры. ***Линейные связанные списки*** являются наипростешими динамическими структурами данных. Графически связи в списках удобно изображать с помощью стрелок. Если компонент не связан ни с каким другим, то в поле указателя записывают значение которое не указывает ни на один из элементов. Такая ссылка обозначается специальным именем – **nil.**

На рис. 8.96 приведена упрощённая структура ***односвязного списка***. На нём поле **INF** – это информационное поле, которое может включать разнообразные данные, как встроенных, так и пользовательских типов данных языка Турбо Паскаль, **NEXT** – указатель на следующий элемент списка. Каждый список должен иметь особый элемент, называемый ***указателем начала списка*** или ***головой списка,*** который естественно по формату отличен от других элементов. В поле указателя последнего элемента списка находится специальный признак **nil**, свидетельствующий о конце списка.

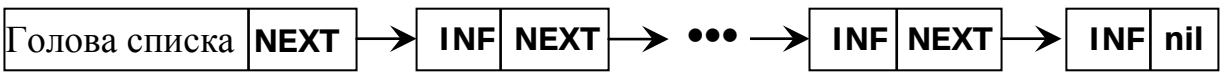


Рис 8.96. Логическая структура односвязного списка

8.27.12. Реализация операций над связными линейными списками

Для рассмотрения программных примеров определим следующие типы данных:

```
Type
  data = ...; { Любая структура информационной части списка }

Type  { Элемент односвязного списка (sll – single linked list):}
  sllptr = ^slltype;  { указатель в односвязном списке }
  slltype = record { элемент односвязного списка }
              inf : data;   { информационная часть }
              next : sllptr; { указатель на следующий элемент }
            end;
```

Для графической формализации описываемых операциями действий, далее приводятся рисунки, демонстрирующие их применение к связанным линейным спискам. На них, сплошными линиями показаны связи, имевшиеся до и сохранившиеся после выполнения операции. Пунктиром показаны связи, установленные при выполнении операции. Значком "↯" отмечены связи, разорванные при выполнении операции. Для всех описываемых операций следует отметить чрезвычайную важность задания последовательности изменения указателей, для обеспечения корректного выполнения операций со списком, которые не затрагивают другие его элементы. При неправильном порядке смены указателей легко потерять какую-либо часть из обрабатываемого списка. Поэтому на рисунках рядом с устанавливаемыми связями в скобках показаны шаги, на котрых эти связи устанавливаются.

Словесные описания алгоритмов даны в виде комментариев к реализованным программно примерам.

8.27.13. Перебор элементов списка.

Эту операцию, Вы будете выполнять над линейными списками чаще других. При её выполнении осуществляется последовательный доступ ко всем элементам до конца списка, либо находится некоторый определённый элемент.

```
Procedure LookSll (head : sllptr); { Перебор 1-носвязного списка }
  { head - указатель на начало списка }
Var
  cur : sllptr; { адрес текущего элемента }
Begin
  cur:=head; { 1-й элемент списка назначается текущим }
   while cur <> nil do
     begin
       < обработка cur^.inf>
```

{обрабатывается информационная часть (ИЧ) элемента, на который указывает `cur`. Обработка может включать: печать содержимого ИЧ; модификацию полей ИЧ; сравнение полей ИЧ с эталоном при поиске по ключу; подсчёт итераций цикла при поиске по номеру и т.д.}

```
    cur:=cur^.next;
```

{из текущего элемента выбирается указатель на следующий элемент и для последующей итерации следующий элемент становится текущим. Если текущий элемент был последним, то его поле **NEXT** содержит пустой указатель и поэтому в `cur` запишется **nil**, что приведёт к выходу из цикла при проверке условия `while`}

```
      end; {while}
End; { Proc LookSll}
```

8.27.14. Вставка элемента в список.

Прграммная реализация процесса вставки элемента во внутрь односвязного списка показана в следующем примере кода и на рис.8.97.

```
                                  { Вставка элемента в середину односвязного списка }
Procedure InsertSll(prev : sllptr; inf : data);
 {prev – адрес предыдущего элемента; inf – данные нового элемента}
Var
  cur : sllptr; {Адрес нового элемента}
Begin
  New(cur); {Выделение памяти для нового элемента}
  cur^.inf:=inf; {Запись в информационную часть элемента}
  cur^.next := prev^.next; {Элемент, который шёл за предыдущим
                                теперь будет идти за новым}
  prev^.next := cur; { Новый элемент идёт за предыдущим}
End; {Proc InsertSll}
```

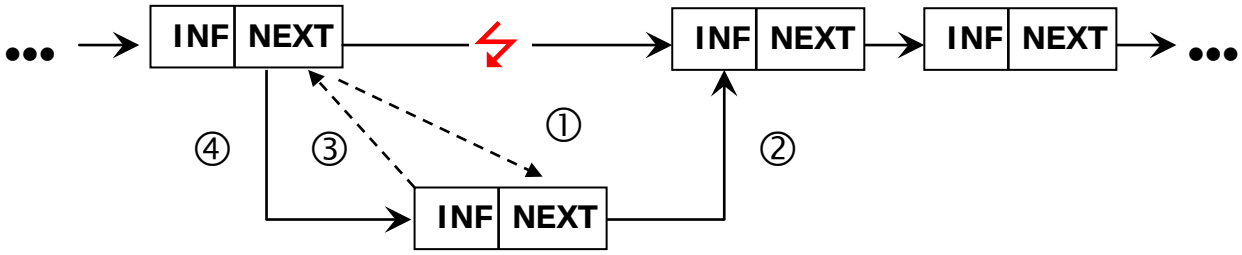


Рис. 8.97. Вставка елемента в середину (вовнутрь) списку

Следует чётко представлять, что после того, как Вы с помощью процедуры `New` получили адрес нового элемента списка `cur` (current – текущий) и заполнили его информационную часть `inf`, необходимо на первом же шаге ① (рис. 8.97) перенести в его поле **NEXT** адрес из предыдущего элемента. Тогда он будет указывать на следующий элемент (шаг ②). После этого Вы должны занести адрес `cur` нового элемента в поле **NEXT** предыдущего элемента (шаг ③). Тогда, после этого, предыдущий элемент будет указывать на новый элемент (шаг ④).

Таким способом Вы можете сделать вставку в средину списка, но не сможете сделать вставку в его начало. При этом должен модифицироваться указатель на начало списка, как показано на рис. 8.98.

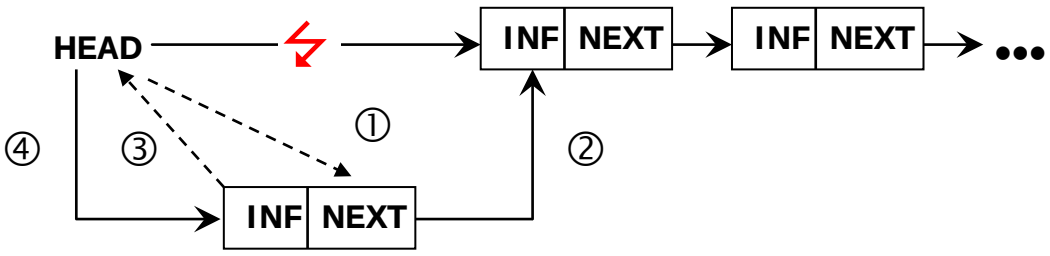


Рис. 8.98. Вставка элемента в голову списка

Как Вы видите, данная операция очень похожа на предыдущую (рис. 8.97), однако тонкости заключаются в средстве сохранения указателя **HEAD**. Поэтому, необходимо разработать процедуру, которая будет выполнять вставку элемента в любое место односвязного списка:

```
                                { Вставка элемента в любое место односвязного списка }
Procedure InsertSll (Var head : sllptr; { Указатель на начало списка, который может измениться в процедуре. Если head=nil – список пуст }
                     prev : sllptr; {Указатель на элемент, после которого осуществляется вставка. Если prev=nil – вставка выполняется перед первым элементом }
                     inf : data; {Данные нового элемента }
                     Var cur : sllptr) {Адреса нового элемента}
Begin
  New(cur);  {Выделение памяти для нового элемента и запись в его информационную часть}
  cur^.inf:=inf;
  if prev <> nil then
    begin {Если есть предыдущий элемент – вставка во внутрь списка, рис. рис.8.97}
      cur^.next:=prev^.next;
      prev^.next:=cur;
    end
  else
    begin {Вставка в начало списка }
      cur^.next:=head; {Новый элемент указывает на бывший 1-й елемент списка. Если head=nil, то новый элемент будет одновременно и последним элементом списка}
      head:=cur; { Новый элемент становится 1-вым в списке, указатель на начало теперь указывает на него}
   end {if}
End; {Proc InsertSll}
```

8.27.15. Удаление элемента из списка.

Порядок удаления элемента из разных частей односвязного списка, то есть из середины списка или из его начала несколько различен. Поэтому, давайте начнём с удаления элемента из середины списка (рис. 8.99).

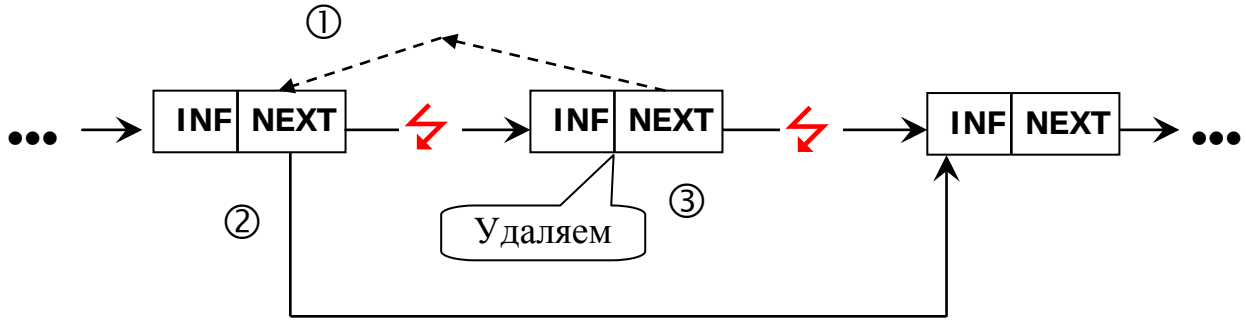


Рис. 8.99. Удаление элемента из середины списка

На первом этапе данной операции мы должны указатель на следующий элемент с элемента, который удаляется, перенести в предыдущий (шаг ①), тогда на шаге ② указатель обходит элемент, который удаляется. И на третьем шаге ③ Вы должны удалить сам элемент, чтобы освободить от него память.

В случае удаления элемента с начала списка Вы должны указатель **HEAD** переадресовать на следующий элемент (рис. 8.100).

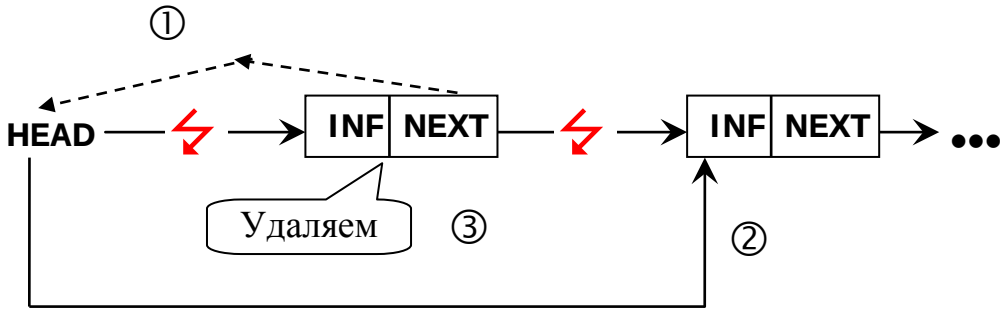


Рисунок 8.100. Удаление элемента из головы списка

Здесь, Вы сначала должны взять указатель на следующий элемент и с элемента, который удаляется, перенести его в указатель **HEAD** (шаг ①). Тогда на шаге ② указатель **HEAD** обходит элемент, который удаляется. А на третьем шаге ③ Вы должны удалить сам элемент, чтобы освободить от него память.

Очевидно, что процедуру удаления легко разработать, если известен адрес элемента, предшествующего удаляемому. Однако, более универсальной будет процедура для случая, когда удаляемый элемент, задаётся своим адресом **del**. Процедура обеспечивает удаление как из середины, так и с начала списка.

```
                {Удаление элемента из любого места односвзного списка}
Procedure DeleteSll(var head : sllptr; {Указатель на начало списка,
                                        может измениться в процедуре}
                    del : sllptr {Указатель на элемент, который удаляется});
var     prev : sllptr; {Адрес предыдущего элемента}
Begin
   if head=nil then
     begin {Попытка удаления из пустого списка расценивается как ошибка (в
            следующих примерах этот случай учитываться на будет)}
       Writeln('Ошибка!');
       Halt;
     end;
  if  del=head  then {Если элемент, который удаляется, 1-й в списке, то
                      следующий за ним становится первым }
    head:=del^.next
  else
    begin { При удалении из середины списка приходится искать элемент,
            который предваряет тот, что удаляется; поиск выполняется
            перебором списка из самого его начала, пока не будет найден
```

```
            элемент, поле next которого хранится с адресом удаляемого
            элемента}
      prev:=head^.next;
      while (prev^.next<>del) and (prev^.next<>nil) do
        prev:=prev^.next;
        if prev^.next=nil then
          begin  {Данная ветвь соотвествует случаю, когда перебран весь
                  список, а элемент не найден. Он отсутствует в списке и это
                  роасценивается как ошибка (в следующих примерах этот
                  случай учитываться на будет!)}
            Writeln('Ошибка!');
            Halt;
          end;
      prev^.next:=del^.next; {Предыдущий элемент теперь указывает на
                              следующий за тем, который удаляется}
    end;
  Dispose(del);  {Элемент исключён из списка – теперь можно освободить
                  занимаемую им память}
End; {Proc DeleteSll}
```

8.27.16. Перестановка элементов списка.

Изменчивость динамических структур данных допускает не только изменения размеров структуры, но и изменения связей между элементами. Для связанных структур изменение связей не требует пересылки данных в памяти, а только требует изменения указателей в элементах связной структуры.

Давайте попробуем сделать перестановку двух соседних элементов списка. Однако, сначала попробуем разобраться, как проходит этот процесс (рис. 8.101).

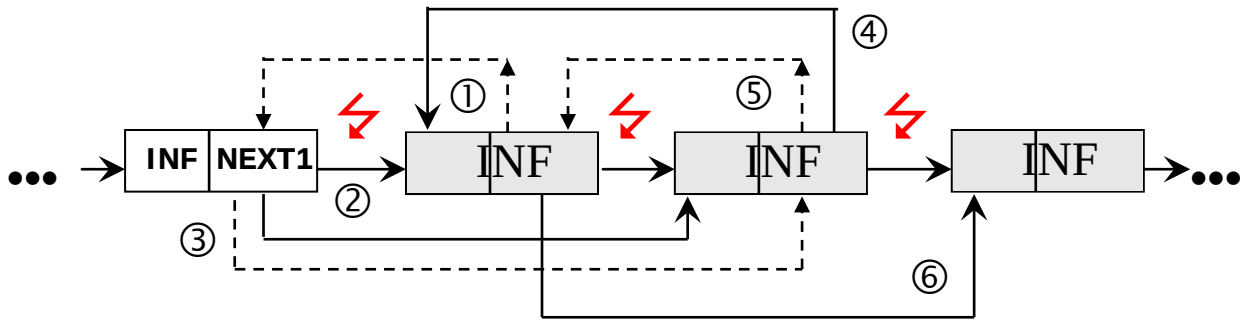


Рис. 8.101. Перестановка двух соседних элементов в середине списка

При конструировании и реализации алгоритма перестановки в односвязном списке (рис. 8.101) будем исходить из предположения, что известен адрес элемента, предшествующего паре, в которой осуществляется перестановка. В нижеприведенном коде также не учитывается случай перестановки первого и второго элементов.

```
{ Перестановка двух соседних элементов в односвязном списке }

Procedure ExchangeSll(prev : sllptr {Указатель на элемент,
                                     предшествующий переставляемой паре});
Var
  p1, p2 : sllptr; {Указатели на элементы пары}
Begin
  p1:=prev^.next;   {Указатель на 1-й елемент пары}
  p2:=p1^.next;     {Указатель на 2-й елемент пары}
  p1^.next:=p2^.next; {1-й элемент пары теперь указывает на следующий за
                       парой}
  p2^.next:=p1;     {1-й элемент пары теперь располагается за 2-ым }
  prev^.next:=p2;   {2-й элемент парытеперь становится 1-ым }
End; {Proc ExchangeSll}
```

8.27.17. Решение задачи по созданию очереди и реализация операций с нею

Теперь Вы хорошо себе представляете, что линейные списки можно применить в приложениях, где существуют непредусмотренные требования к размерам памяти, необходимой для сохранения данных, а также велико число сложных операций над даннми и, особенно, включений и исключений.

Стек – это такой последовательный список с переменной длиной, включение и исключение елементов в котором выполняются только ***с одной стороны списка***, называемого ***вершиной стека***. Существуют и другие названия стека – магазин, а также очередь, функционирующая по принципу LIFO (Last-In-First-Out – "последним пришёл – первым исключается"). Примером стека может служить обойма в пистолете (рис. 8.102).

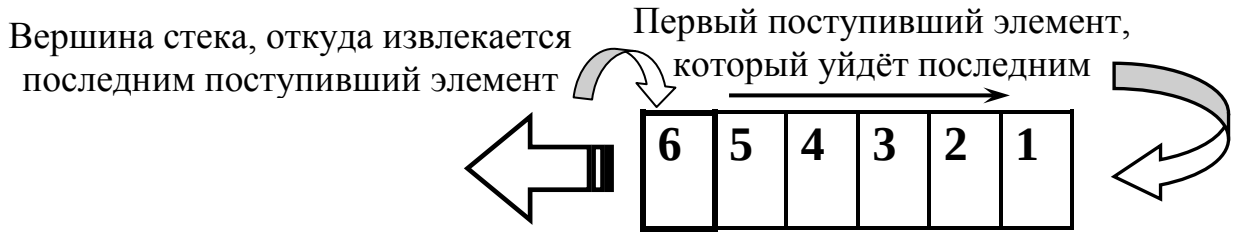


Рис.8.102 Схема функционирования стека

В программных приложениях стек широко применяется в операциях сохранения, передачи и дальнейшого использования фактических параметров при обращениях к процедурам, функциям, модулям, удалённым объектам и другим автономно функционирующим программным компонентам (см. пп. 6.2, 6.6, 6.7)

Основные операции над стеком – это включение нового элемента (английское название push – заталкивать) и исключение элемента из стека (англ. pop – выскакивать).

Известно, что стек может быть представлен как линейный список, в котором включение элементов всегда проводится с начала списка, а

исключения – также с начала. Для представления его Вам достаточно иметь один показатель – top, который всегда указывает на последний записанный в стек элемент. В выходном состоянии (при порожнем стеке) показатель top – пуст. Главные процедуры StackPush и StackPop реализуют соответственно операции включения элемента в начало списка (стека) и, соответственно, исключения элемента из начала этого списка.

Обратите внимание на то, что при включении элемента для него выделяется память, а при исключении – освобождается. Перед включением элемента проверяется доступное программе пространство в памяти, и если оно меньше требуемого для включения нового елемента, стек считается заполненным. При очистке стека последовательно просматривается весь список и исключаются все его элементы. При списковом представлении стека, определение его размера представляется непростой задачей, так как при этом приходится предварительно производить перебор всего списка для подсчёта числа всех элементов. Чтобы избежать выполнения таких списка Вы должны ввести дополнительную переменную `stsize`, содержащую текущее число элементов в стеке и корректируется при каждом включении/исключении (см. программный код).

{Реализация стека на 1-носвязном линейном списке}

```
unit Stack;
Interface
type data = ...; {Элементы могут иметь любой тип}
Procedure StackInit;
Procedure StackClr;
Function StackPush(a : data) : boolean;
Function StackPop(Var a : data) : boolean;
Function StackSize : integer;

Implementation
type stptr = ^stit; {Указатель на элемент списка}
     stit = record {Элемент списка}
       inf : data; {Данные}
       next: stptr; {Указатель на следующий элемент}
       end;
Var top : stptr; {Указатель на вершину стека}
    stsize : longint; {Размер стека}
{** Инициализация – список пустой }

Procedure StackInit;
  begin   top:=nil; stsize:=0;  end; {Процедура StackInit}
{**Очищение – освобождение всей памяти}

Procedure StackClr;
 var x : stptr;
  begin {Перебор элементов до конца списка и их удаление}
  while top<>nil do
    begin  x:=top; top:=top^.next; Dispose(x);  end;
  stsize:=0;
end; {Proc StackClr}

Function StackPush(a: data) : boolean; {Ф-ция занесения в стек}
```

```
 var x : stptr;
  begin   {Если нет более свободной памяти – отказ от операции выделения}
  if MaxAvail < SizeOf(stit) then StackPush:=false
  else  {Выделение памяти для элемента и заполнения инф. части}
    begin  New(x); x^.inf:=a;
                {Новый элемент содержится в голове списка}
      x^.next:=top; top:=x;
      stsize:=stsize+1; {Коррекция размера}
      StackPush:=true;
    end;
end; {Func StackPush}

Function StackPop(var a: data) : boolean; {Выборка из стека}
 var x : stptr;
  begin
{Список пустой – стек пустой}
  if top=nil then StackPop:=false
  else begin
    a:=top^.inf; {Выборка информации из 1-го элемента списка}
    {1-й элемент исключается из списка, освобождается память}
    x:=top; top:=top^.next; Dispose(top);
    stsize:=stsize-1; {Коррекция размера}
    StackPop:=true;
end;  end; { StackPop }

Function StackSize : integer; {Определение размера стека}
  begin   StackSize:=stsize;   end; {Func StackSize}
END.
```

Программный код примера для организациия на односвязном линейном списке очереди **FIFO** разработайте самостоятельно. Для линейного списка, представляющего очередь, Вам будет необходимо сохранять:

`top` – указатель на первый элемент списка;

`bottom` указатель на последний элемент списка.

Упражнения

1. Для каких целей используется оператор `With`?

2. Разработайте и реализуйте коды программы для решения следующей задачи.

В часовой мастерской все вновь поступающие заказы, регистрируются в компьютере. О каждой заявке сохранчетсяся следующая информация:

– номер заказа;
– фамилия зказчика;
– тип часов (механические, электронно-механические, электронные);
– марка часов;
– срок выполнения и стоимость работы.

Часы разных типов ремонтируются разными мастерами. Необходимо организовать выдачу задания каждому мастеру в порядке срочности заказов, а также указать общую сумму заказов.

3. Написать программу для обработки информации о товарах, которые хранятся на складе. Информация о них содержит следующие данные:

– наименование товара.
– стоимость товара.
– страну-производителя товара.
– конечный срок реализации товара.
– количество товара, имеющегося в наличии.

Программа должна позволять пользователю считывать данные с клавиатуры и заносить их в требуемую запись, а также считывать данные из заданной записи и выводить на экран список товаров, с заканчивающимся сроком реализации (меньше месяца).

4. Создать модуль для работы с комплексными числами и вызывающую его программу, проверяющую правильность работы модуля.

5. Добавить в конец не пустой очереди все ёё элементы, расположенные в обратном порядке.

6. Информационное поле элемента очереди – строка. Подсчитать количество слов очереди, расположенных за предпоследней строкой.

7. Информационное поле элемента очереди – целочисленное. Подсчитать и вывести на экран элементы очереди, не равные нулю.

Приложение 1

Преподавание информатики: потерянная дорога

Никлаус Вирт*

Приветствие на открытии Международной конференции по преподаванию информатики ITiCSE
г. Аархус (Дания), 24 июня 2002 г.

Всего через пару дней после получения приглашения выступить на открытии данной Конференции по преподаванию информатики, я прочел доклад коллеги из США. Доклад достигает кульминации в следующем абзаце о преподавании:

> ***Поучительно сравнить учебники для средней школы по математике и по информатике. Я имел несчастье проделать это довольно внимательно, и вот мое заключение: мы ни на что не годимся <we suck>. Похоже, мы заставляем студентов сделать вывод, что серьезно думать о карьере в информатике могут только мазохисты.***

И я с этим согласен. Посвятив существенную часть своей карьеры доведению искусства создания программ до такого уровня, чтобы его можно было преподавать методично и систематически, я разочарован в доминирующих разрушительных тенденциях. Хоть я и устал от непопулярной роли вечного критика, процитированная статья вновь всколыхнула эмоции, и вот я здесь, поскольку упомянутый доклад продолжается так:

> ***Как профессионалы в информатике, мы обязаны поднять свои голоса против традиции, приравнявшей компьютерную грамотность к знанию темных деталей языка программирования, используемого в индустрии.***

Мне вспоминается рассказ Э.Дейкстры о его ночном кошмаре после чтения спецификаций нового языка программирования PL/1 в 1965 г. Ему представилось, что в будущем ***программирование*** приравняют к ***выучиванию PL/1,*** а ***информатику*** – к ***овладению JCL к OS/360*** (речь идет о языке программирования и языке управления заданиями для компьютеров фирмы IBM, печально известных своим крайне неудачным дизайном; российские программисты старшего поколения помнят, что это такое, по опыту работ на ЕС ЭВМ – *прим. перев.*). Достаточно заменить PL/1 на C++ или Java, а JCL – на Windows или Linux, и вы чудесным образом перенесетесь в настоящее.

Тогда я написал своему коллеге о полном согласии с ним. Он ответил следующим разъяснением:

> ***Мои резкие замечания о преподавании – результат полного провала попыток помочь сыну, ученику старших классов, освоить C++. Дизайн языка чудовищен, а учебник написан отвратительно. Мой сын не мог понять, почему x = y должно отличаться от y = x. Дейкстра также жаловался мне, что важная книга по языку Java не содержала формальной грамматики.***

* http://www.inr.ac.ru/~info21/greetings/wirth_doklad_rus.htm

Действительно, формальные правила синтаксиса были заданы лишь в четвертой версии языка! Но позвольте мне продолжить цитату:

Я был разочарован не только таким положением вещей, но и тем, что серьезные специалисты по информатике воспринимают его как совершенно нормальное. Мне еще ни разу не попадался учебник по UNIX/C++/Java, который я мог бы освоить за неделю. Их учебники невозможно читать, они предполагают, что читатель принадлежит какой-то секте, чьи заклинания должны оставаться тайной для публики, и читателю не следует ожидать многого в плане надежности, связности или общей элегантности. Мое отчаяние достигло апогея, когда я пытался научить своего сына программировать на C++ – факультативный курс в средней школе! После полугода агонии – как для отца, так и для сына – я посоветовал сыну бросить этот курс.

Чего я не понимаю, так это отсутствия возмущения среди ученых, специалистов по информатике. Когда управляющий совет колледжа решил включить в программу C++ в середине 90-х гг., я письменно выразил им свое возмущение. Но я не в счет.

Преподавание в средней школе оказывается отличной проверкой не только Способности к преподаванию, но и ясности учебников. Время от времени я получаю жалобы от учителей, сообщающих о трудностях, которые они испытывают с современными средствами и языками программирования (и нередко они просят версию старого Паскаля для новых машин). Вот выдержка из переписки с физиком-теоретиком из России, который предпринял эксперимент по преподаванию программирования Способным ученикам старших классов лицея. Он пишет:

Интересно, что курс в лицее помогает лучше понять как само программирование, так и то, как его нужно преподавать. Примерно половина улучшений в моем университетском курсе будет сделана благодаря опыту, полученному в лицее. Кроме того, я почувствовал, в каком ужасном состоянии находится преподавание программирования – как на уровне средней школы, так и университета.

Слова резкие, но они не преувеличение. В чем же ошибка? Что можно сделать? Вспоминается смиренное: “Я не в счет” – из процитированного выше письма. Мы чувствуем себя ***беспомощными***. Кое-кто чувствует себя обреченным на жалобы, а большинство решают примириться с очевидными фактами и кое-как приспособиться к ним. Но вряд ли подобную позицию интеллектуальных лидеров можно оправдать. Посмотрим правде в глаза: разве большинство учреждений образования не оказалось заложниками горстки компаний, чья профессиональная цель состоит в повышении доходов – идет ли речь о производителях оборудования, программного обеспечения или об издательствах? Университеты, которые могли бы оказывать хоть какое-то корректирующее влияние, и чьи преподаватели числятся среди авторов популярных учебников, на самом деле больше заинтересованы в том, чтобы демонстрировать свое участие в модных исследованиях, оставляя преподавание ассистентам.

Конечно, в такой постмодернистской академической среде профессор давно перестал быть мудрецом, углубляющимся все дальше в свой излюбленный предмет в тиши кабинета. Современный профессор – это менеджер большой команды исследователей, хваткий добытчик грантов, поддерживающий тесные связи с ключевыми огранизациями-источниками финансирования, и неутомимый автор волнующих проектных заявок и впечатляющих отчетов о достигнутых успехах. В этом высоко конкурентном бизнесе было бы самоубийством растрачивать время на размышления о том, как лучше рассказать о простых вещах массе начинающих. Когда речь заходит о материалах для курса, программном обеспечении и т.п., очевидный выбор – взять то, что лежит на полке и заведомо принято всеми остальными. В этой борьбе за успех и выживание лучше всего примкнуть к толпе. Достижения измеряются размером команды, количеством публикаций, цитирований и докладов на конференциях, и использованными ресурсами – но не преданностью делу преподавания, которую все равно невозможно измерить. Разумеется, такой стиль академической жизни нередко противоречит внутренним убеждениям индивидуума, но навязывается давлением извне превратить храмы учености в хорошо разрекламированные источники доходов, и этот стиль граничит с проституцией.

Конечной целью образования должно быть искусство конструктивного мышления. Именно в таком контексте становится важным наличие хорошо спроектированного языка программирования.

Слова резкие. Что же можно сделать? Ведь и в самом деле индивидууму очень трудно противостоять глобальной тенденции. Я имею в виду тенденцию перехода от долгосрочного планирования к немедленному извлечению доходов, от учения с целью понимания к применению навыков непосредственного действия. Что касается нашего предмета – информатики и программирования для компьютеров, – то конечная цель учреждения образования должна быть гораздо шире, чем овладение каким-либо языком программирования. Это должно быть никак не менее, чем искусство проектирования артефактов для решения сложных задач. Иногда это называют искусством конструктивного мышления. Именно в таком контексте становится важным наличие подходящего инструмента, хорошо спроектированного языка программирования. Он играет роль теории, на которой основываются наши методы. Как можно научиться хорошему и эффективному проектированию, если базовый формализм, само основание представляет собой ошеломляющую, непостижимую путаницу? Как можно освоить такое искусство без образцовых примеров, достойных изучения и подражания? Конечно, не все равно одарены в том, что касается хорошего проектирования, и все же надлежащее обучение, инструменты и примеры играют важнейшую роль.

Люди по ошибке принимают сложность за изощренность.

Я говорил о проектировании сложных артефактов. Очевиден факт, что наши артефакты становятся все сложнее. Например, автомобиль это уже не просто двигатель и

четыре колеса. В нем еще есть компьютер для определения оптимального количества впрыскиваемого топлива в самые подходящие моменты времени. Это нужно, чтобы снизить затраты топлива, чтобы сберечь энергию. Но современный автомобиль еще содержит с десяток (или больше) хорошо спрятанных электромоторов для управления окнами и антеннами, радар для определения расстояния до опасных объектов, системную шину для связи всех этих устройств, а также компьютер для управления ими. Эти вещи не слишком способствуют истинному назначению автомобиля, но они добавляют сложность, затрудняют обслуживание и повышают стоимость. Это мнение было выражено в недавней статье о “совершеннейшем автомобиле” в газете Нью-Йорк Таймс (озаглавленной ***За рулем, BMW 745i, технический нокаут***):

“Люди по ошибке принимают сложность за изощренность," сказал однажды Никлаус Вирт, швейцарский ученый, специалист по информатике. В роскошных автомобилях, напичканных новейшими технологиями, прибамбасами и завлекалками, уже трудно увидеть эту разницу – так же как трудно определить ту точку, за которой техника, целью которой была помощь водителю, становится помехой вождению ... Достаточно сказать, что семерка, несомненно, самый «продвинутый» седан в мире, но отнюдь не самый легкий в управлении.

Ненужная, самодельная сложность порождает множество проблем.

Этот пример легко переносится в область компьютерных и программных систем. Они стали безмерно сложными не столько потому, что все их чудесные средства и возможности были на самом деле нужны, сколько просто потому, что они были возможны. Поэтому производители надеются, что они дадут преимущество перед конкурентами. «Преимущество», однако, приводит к избыточной громоздкости, трудности использования и снижению надежности. Но обманутый клиент понимает это только после покупки.

Требуется гораздо больше таланта, проницательности и времени, чтобы спроектировать экономную, простую и эффективную систему, нежели сложную и громоздкую.

Вывод состоит в том, что ненужная, самодельная сложность порождает множество проблем. Главное, она размывает различием между тем, что действительно важно (оптимальное впрыскивание топлива), и тем, что эфемерно (моторчики для закрытия окон). Беда в том, что требуется гораздо больше таланта, проницательности и времени, чтобы спроектировать экономную, простую и эффективную систему, нежели сложную и громоздкую.

Итак, хороший дизайн должен быть в центре нашего преподавания. Но как нам учить образцовому дизайну с помощью инструментов и языков, которые делают нас посмешищем? К нашему сожалению, индустрия программирования сделала немного, чтобы помочь нам, преподавателям, преодолеть наши трудности.

Давайте поэтому взглянем на индустрию программирования, которая оказывается неспособной обеспечить нас идеальными инструментами. Мы

увидим, что она сама страдает от чрезмерной, немотивированной сложности, а также от отсутствия регулярности и надежности в своих продуктах. На самом деле индустрия была бы гораздо производительней, если бы строила свои системы на прочной основе вместо того, чтобы прививать новые ростки на гниющие стебли. Мне известно о случае, когда в большой компании был запущен проект, чтобы с нуля спроектировать замену широко используемому программному приложению, ставшему слишком громоздким и трудным в сопровождении. Однако через некоторое время проект был закрыт по совету отдела маркетинга. Решение было принято из-за всеобщего сопротивления клиентов любому изменению, а следовательно и усовершенствованию. Среди клиентов было много и учреждений образования. Им нужно было сохранить свои инвестиции в создание курсов и курсового материала.

Кроме того, вузы становятся все более похожими на коммерческие предприятия, предлагая то, что требуют и за что платят их клиенты, вместо того, что более способствовало бы развитию в долгосрочной перспективе. Но студенты сосредоточивают свое внимание на том, что даст им лучшие шансы в поиске работы, т.е. на овладении навыками, необходимыми в данный момент для работы на предприятиях индустрии.

Очевидно, перед нами в высшей степени устойчивый порочный круг: учителя не могут изменить свои курсы, т.к. они должны привлечь и доставить удовольствие студентам; студенты требуют то, что практикуется в промышленности; а индустрия применяет и воспроизводит то, чему обучены ее работники. Этот замкнутый круг напоминает ситуацию, описанную мной во введении к сообщению о Паскале в 1970 г.: вузы стремятся учить тому, что требует индустрия, а в индустрии практикуется то, чему ее работники выучились в вузах.

Программирование является, возможно, самой важной новой дисциплиной постиндустриальной эры.

Порочные круги существуют, чтобы их разрывать. Делать это должны те, кто обнаруживает их порочность. Беда сегодня в том, что эта долгосрочная порочность недостаточно признана. Однако программирование как искусство конструктивного дизайна слишком важно, чтобы пожертвовать им в пользу краткосрочных коммерческих выгод и привычек. Программирование является, возможно, самой важной новой дисциплиной постиндустриальной эры.

Часто обвиняют могучую индустрию программирования в навязывании своих продуктов беспомощному сообществу клиентов, включая пренебрежимое меньшинство преподавателей и студентов. Однако ситуация еще хуже: преподаватели большей частью добровольно поддались доминирующим коммерческим тенденциям, зачастую ощущая при этом угрызения совести и открыто жалуясь на свои горести, имеющие причиной их собственную позицию. Многие в индустрии сожалеют об этом отречении от лидерства и ответственности.

Только университетские преподаватели в состоянии сломать этот порочный круг ... Они просто обязаны подняться до роли лидеров.

Только университетские преподаватели в состоянии сломать этот порочный круг. Это сделать нельзя ни быстро, ни легко. Но если это окажется невозможным, то что-то, видимо, глубоко неправильно с преподавателями и их академической свободой. Они просто обязаны подняться до роли лидеров.

Порочный круг был однажды разорван, когда распространился Паскаль. При поддержке коллег-единомышленников и в упорном противостоянии рутинерам, Паскаль распространился в учебных заведениях и проник в индустрию. Это произошло, несмотря на могучую конкуренцию со стороны самой индустрии и других больших организаций, в соперничестве с языками PL/1, Алгол 68 и Ада. Однако наследники Паскаля, существенно его превосходившие, Модула-2 и Оберон, не получили должного внимания среди преподавателей, и сами пали перед лицом самого недостойного из соперников – C.

Самого недостойного, т.к. в этом языке были нарушены все открытые к тому времени принципы серьезного программирования. Он запутывает студентов, допуская разный смысл для x = y и y = x и принуждая всех писать x = = y вместо обычного x = y. Только за одни эти пороки он заслуживает изгнания из учреждений образования. Однако, сей уродливый синтаксис был целиком воспроизведен в языке Java, принятие которого академическим сообществом произошло, по меньшей мере, отчасти благодаря этой преемственности.

В ряде университетов на общепризнанный разрыв между желательным, с точки зрения образования, и практикой “реального мира”, был дан ответ посредством выбора функционального языка для первоначального обучения программированию. Однако эта увертка только увеличила разрыв, т.к. теперь обучение стало вестись в рамках другой парадигмы программирования и мышления при создании программ и, как следствие, требуется известное усилие при переходе от обучения к профессиональной деятельности. В результате научная дисциплина и инженерная практика программирования стали восприниматься как разные сущности, почти не имеющие видимой связи. Первая осталась своего рода искусством для искусства, вторая эволюционирует как комбинация эвристик и интуиции, все более изощренного ***hacking’a***. Но все это не может служить фундаментом научной дисциплины, каковая должна, в конце концов, лечь в основу любых инженерных конструкций.

И все же борьба с порочным кругом не совсем безнадежна. Мы указали на тех, кто должен возглавить атаку. Но как?

Позвольте мне закончить выступление смелым предложением для этой просвещенной аудитории профессионалов преподавания. Я вижу в своем воображении образцовый учебник в качестве подходящего исходного пункта. Он должен удовлетворять следующим критериям:

1. Начинаться сжатым введением в основные понятия программного проектирования.

2. Использовать лаконичную формальную нотацию, строго определенную не более чем на примерно 20 страницах.

3. Основываясь на этой нотации, вводятся основные понятия итерации, рекурсии, логического утверждения <assertion> и инварианта.

4. Центральная тема – структурирование утверждений и типизация данных.

5. За этим следуют концепции упрятывания информации, модульности и проектирование интерфейсов, продемонстрированные образцовыми примерами.

6. Книга устанавливает терминологию, которая столь же интуитивна, сколь и точна.

7. Книга имеет умеренный размер.

Позвольте мне заключить еще двумя замечаниями. Мой коллега, чьи слова приведены в начале доклада, закончил так:

> ***Руководящим для моей карьеры в преподавании и исследованиях был тот принцип, что хорошо подготовленные профессионалы должны быть гораздо эффективнее, чем вдохновенные любители. В их производительности должно быть различие, и притом существенное. Думаю, что нашей общей целью должно быть увеличение этого различия.***

Несколько месяцев назад я получил просьбу дать список задач, которые я считаю первостепенными для информатики на ближайшие десятилетия. Возможно, создание подобного учебника следовало бы включить в этот список, и может быть даже первым номером. Во всяком случае, эта задача пока не решена.

Приветствие на открытии Конференции ITiCSE, Аархус (Дания), 24. 6. 2002

Перевод на русский язык: Ф.В.Ткачев, 4 июля 2002 г.

Преподавание информатики: потерянная дорога (Никлаус Вирт)

Приветствие на открытии Международной конференции по преподаванию информатики ITiCSE

г. Аархус (Дания), 24 июня 2002 г. http://www.inr.ac.ru/~info21/greetings/wirth_doklad_rus.htm

Введение в позиционные системы счисленя

Д.2.1. Позиционные системы счисления

Мы с Вами хорошо знакомы и привыкли работать с десятичной системой счисления. Однако при реализации вычислений на компьютере специалисты с толкнулись с проблемой невозможности её технической реализации. Ведь для отображения десяти устойчивых состояний 0..9 требуются соответсвующие технические решения. Их пока в природе не существует! Поэтому в компьютере повсеместно используется только двоичная система счисления. Это и понятно, решения лежат на поверхности: две цифры 0 и 1 отобразить значительно легче. Реле включено или выключено, конденсатор заряжен или не заряжен, магнитный домен поляризован или нет, триггер находится в состоянии 0 или 1 и т.д. Кстати, триггер–основа конструкции всех компьютеров.

Триггер – это последовательностная электронная схема с двумя состояниями, каждое из которых при определённых условиях на входах поддерживается постоянным (т.е. стабильным). Каждому из этих состояний ставится всоответствие логическое значение, которое «хранит» триггер. Таким образом, совокупное состояние последовательностной схемы, запоминающие свойства которой реализованиы на триггерах, представляет собой просто комбинацию состояний этих триггеров.

Поэтому при вводе данных в виде десятичных чисел они автоматически преобразуются в их двоичный эквивалент, чтобы их было удобно представлять с помощью триггеров. Вместе с тем пользователям, которым приходится работать с языками программирования, а в особенности с языком Ассеблера, необходимо хорошо представлять процессы преобразования чисел из одних систем счисления в другую. Хотя двоичная система и обеспечивает точное представление чисел в памяти, тем не менее, с последовательностью из одних нулей и единиц трудно работать. Кроме того, возрастает вероятность совершить ошибку, поскольку при наборе числа вида 10110101 чрезвычайно легко сделать опечатку при вводе.

Много лет тому назад программисты убедились, что обычно им приходилось работать не с отдельными битами, а с группами битов. Первые микропроцессоры были 4-битовыми устройствами (они обрабатывали по 4 бита за один приём). Поэтому логической альтернативой двоичной системе оказалась система, которая оперировала четвёрками битов.

Как Вам известно, четырьмя битами можно представить двоичные значения, от 0000 до 1111 (что эквивалентно десятичным значениям от 0 до 15), т.е. всего 16 возможных комбинаций. Если в системе счисления должны быть обозначены все эти комбинации, то она должна иметь 16 цифр. Такая система счисления называется шестнадцатиричной. Из шестнадцати цифр этой системы счисления первые 10 получили обозначения от 0 до 9 (десятичные значения от 0 до 9), а остальные шесть – от A до F (десятичные значения от 10 до 15).

Необходимо представлять, что обычная десятичная система — это лишь одна из многих ***позиционных систем счисления.*** В этих системах используется конечный набор различных символов. Каждый символ называется ***цифрой*** и обозначает определённое количество единиц. Число различных символов в наборе называется ***основанием*** системы счисления. Если выражаемое значение больше значения максимальной цифры в системе счисления, то для его выражения используется последовательная запись цифр, которая и образует ***число.*** В зависимости от той позиции, которую каждая цифра занимает в числе, ей ставится в соответствие весовой множитель равный основанию системы.

Рассмотрим привычную десятичную систему. В ней только 10 различных цифровых символов: 0, 1, ... 9. Тогда, например, число 536.4 с учётом позиционности расположения цифр, выражается следующим полиномом:

$5\times10^{2}+3\times10^{1}+6\times10^{0}+4\times10^{-1}$

Здесь цифра 5 входит с весом 100, цифра 3 – с весом 10, цифра 6 – с весом 1**, а** цифра 4 – с весом 0,1.

Тогда в обобщённом виде, для любой позиционной системы счисления по некоторому основанию число может быть записано:

$$N = d_{n-1}\, d_{n-2}\ldots d_1\, d_0\ d_{-1}\, d_{-2}\ldots d_{-m}\ . \qquad (2.1)$$

Выражению (1) будет отвечать полином:

$$\boldsymbol{N = d_{n-1}b^{\,n-1} + d_{n-2}b^{\,n-2} + \ldots + d_{-m}b^{\,-m}\ .} \qquad (2.2)$$

В этой общей форме d_i – цифры, лежащие в диапазоне $0\le d_i<b$; n – число цифр левее ***разделительной,*** или позиционной, точки (в некоторых случаях вместо точки используется запятая); ***m*** – число цифр правее точки, а b – основание системы счисления. В табл. 2.1 перечислены наиболее употребимые системы счисления. Как правило, в системах с основанием, меньшим 10, в качестве цифровых символов используются соответствующие первые цифры десятичной системы; для систем же с основанием, большим 10, используются десятичные цифры с добавлением первых букв латинского алфавита. В таблице указана также относительная упорядоченность цифр в системах.

Таблица 2.1

Системы счисления

Основание	Система счисления	Цифровые символы (цифры), представляющие числа
2	двоичная	0,1
3	троичная	0,1,2
4	четверичная	0,1,2,3
5	пятеричная	0,1,2,3,4
8	восьмеричная	0,1,2,3,4,5,6,7
10	десятичная	0,1,2,3,4,5,6,7,8,9
12	двенадцатеричная	0,1,2,3,4,5,6,7,8,9,A,B
16	шестнадцатеричная	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Из упомянутых систем три системы представляют особый интерес при изучении вычислительной техники — это ***двоичная, восьмеричная*** и ***шестнадцатеричная***. Записи некоторых чисел в этих системах выглядят следующим образом:

$1011{,}101_2$
$372{,}46_8$
$C65F{,}B3_{16}$

По соглашению десятичный индекс, сопровождающий число, указывает основание системы счисления. Индекс опускается, когда значение основания ясно из контекста.

Как и в десятичной системе, число представлено совокупностью выписанных рядом цифр. Дробная и целая части располагаются соответственно справа и слева от разделительной точки. В случае использования двоичной системы цифры 0 и 1 называют ***битами***, как сокращение от *BInary digiTs* (англ.) (т.е. двоичные цифры).

Таблица 2.2

Первые 32 числа в двоичной, восьмеричной и шестнадцатеричной системах и их десятичные эквиваленты

Десятичные	Двоичные	Восьмеричные	Шестнадцатиричные
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	1011	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F

Д.2.2. Преобразование чисел из одной системы счисления в другую

Любое число может быть интерпретировано соответствующим полиномом по основанию выбранной системы счисления (двоичной, восьмеричной, шестнадцатеричной и других систем). При работе с компьютерами очень часто приходится переводить числа из одной системы счисления в другую. Тогда исходное и переведённое в новую систему счисления числа можно считать эквивалентными представлениями одной и той же величины в разных системах счисления.

Д.2.2.1. Перевод в десятичную систему чисел из не десятичной системы

Рассмотрим этапы преобразования числа из недесятичной позиционной системы в эквивалентную десятичную форму. Такое преобразование легко получается простым вычислением значения полинома, соответствующего числу. Это вычисление можно, например, выполнить следующим образом:

1. Записываем число в виде полинома

$$d_{n-1}\, d_{n-2}\ldots d_0 d_{-1}\ldots d_{-m}=d_{n-1}b^{n-1}+d_{n-2}b^{n-2}+\ldots+d_0b^0+d_{-1}b^{-1}+\ldots+d_{-m}b^{-m},$$

где b – основание системы, выраженное в десятичной форме, а d – цифры исходной системы счисления. Для тех систем, где цифры представляются буквами, последние при вычислении заменяются на десятичные эквиваленты, например A=10, B=11, C=12 и т.д.

2. Вычисляем значение полинома, пользуясь десятичной системой счисления.

Для иллюстрации перевода из двоичной системы в десятичную систему рассмотрим двоичное число 1110.1_2. Записывая его в виде полинома по степеням основания 2, получим

$$\begin{aligned}1110{,}1_2&=1\times2^3+1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}\\&=1\times8+1\times4+1\times2+0\times1+1\times0{,}5\\&=8+4+2+0+0{,}5\\&=14{,}5_{10}\end{aligned}$$

Таким образом, 14.5 есть десятичный эквивалент двоичного числа 1110.1.

В качестве второго примера преобразуем в десятичную систему шестнадцатеричное число $D3F.4_{16}$:

$$\begin{aligned}D3F{,}4_{16}&=D\times16^2+3\times16^1+F\times16^0+4\times16^{-1}\\&=13\times16^2+3\times16^1+15\times16^0+4\times16^{-1}\\&=13\times256+3\times16+15\times1+4\times0{,}0625\\&=3328+48+15+0{,}25\\&=3391{,}25_{10}\end{aligned}$$

Д.2.2.2. Перевод из десятичной системы в любую позиционную систему

Перевод десятичного числа в эквивалентную форму в другой системе счисления более сложен. В процессе преобразования приходится порознь трансформировать целую и дробную части числа.

Рассмотрим сначала преобразование целого десятичного N_I в систему счисления с основанием b (b — целое положительное число). Поскольку число в системе с основанием b можно записать в виде полинома по степеням Несоответствующими цифрами в качестве коэффициентов, мы получаем

$$\begin{aligned} N_I &= d_{n-1}b^{n-1} + d_{n-2}b^{n-2} + \ldots + d_1 b^1 + d_0 b^0 \\ &= d_{n-1}b^{n-1} + d_{n-2}b^{n-2} + \ldots + d_1 b^1 + d_0 \end{aligned} \qquad (2.3)$$

Теперь нужно найти цифры $d_{n-1}, \ldots, d_1, d_0$ удовлетворяющие выписанному уравнению (2.3). Для этого разделим обе части выражения на b. Получим целое частное:

$$N' = d_{n-1}b^{n-2} + \ldots d_2 b^1 + d_1 b^0 \qquad (2.4)$$

и остаток:

$$\text{Остаток}\left(\frac{N_I}{b}\right) = d_0 \qquad (2.5)$$

Таким образом, остаток равен младшей цифре числа в системе счисления с основанием b, т.е. d_0 . В результате деления в остатке может оказаться более одной десятичной цифры, если b больше 10. Однако поскольку остаток всегда меньше b, то его значение будет соответствовать цифре d_0.

Если процесс деления повторить для целого частного N'_I, мы получим снова целое частное:

$$N'' = d_{n-1}b^{n-3} + \ldots + d_2 b^0 \qquad (2.6)$$

и остаток:

$$\text{Остаток}\left(\frac{N'_I}{b}\right) = d_1 \qquad (2.7)$$

В этом случае остаток соответствует следующей справа цифре числа с основанием системы b. Легко видеть, что, повторяя описанный процесс, вплоть до нулевого частного, мы получим все цифры d_i уравнения для N_I. Обратите внимание на то, что остаток следует каждый раз представлять цифрой в системе счисления с основанием b. Очевидно, что процесс завершится после конечного числа шагов.

Разберем описанную процедуру на примере перевода десятичного числа 52 в эквивалентную двоичную форму. Вычисления проводятся многократным делением на 2:

Остаток:

52 |2 $0=d_0$
26 |2 $0=d_1$
13|2 $1=d_2$
6 |2 $0=d_3$
3 |2 $1=d_4$
1 |2 $1=d_5$
0

Следовательно, $52_{10}= d_5\ d_4\ d_3\ d_2\ d_1\ d_0 =110100_2$.

В качестве второго примера рассмотрим перевод десятичного числа 58 506 в ***шестнадцатеричную*** систему. Последовательные деления на 16 дают:

Деление	Остаток	Значения остатка в шестнадцатиричной системе счисления
58 506 |16	10	$A=d_0$
3 656 |16	8	$8=d_1$
228 |16	4	$4=d_2$
14 |16	14	$E=d_3$
0		

Следовательно, $58\ 506_{10}= d_3\ d_2\ d_1\ d_0 = E48A_{16}$.

Процедура перевода правильной десятичной дроби в систему счисления с основанием b должна быть несколько иной. Обозначим через N_F десятичную дробь, соответствующую полиному (2.8):

$$d_{-1}b^{-1} +d_{-2}b^{-2}+\ldots+d_{-m}b^{-m} \qquad (2.8)$$

–где d_{-1} , d_{-2} , …, d_{-m} – цифры, которые нужно определить. Поскольку полином и N_F обозначают одну и ту же величину, то имеет место равенство: (2.9):

$$N_F = d_{-1}b^{-1} +d_{-2}b^{-2}+\ldots+d_{-m}b^{-m} \qquad (2.9)$$

Умножая обе части равенства (2.9) на b, получим, (2.10):

$$\begin{aligned} b\,N_F &= d_{-1}b^{0} +d_{-2}b^{-1}+\ldots+d_{-m}b^{-m+1} \\ &= d_{-1} +d_{-2}b^{-1}+\ldots+d_{-m}b^{-m+1} \\ &= d_{-1}+N'_F \end{aligned} \qquad (2.10)$$

Произведение состоит из целой части d_{-1} и дробной части N'_F. Целая часть эквивалентна старшей цифре исходной дроби в системе счисления с основанием b. Как и ранее, легко видеть, что целая часть, соответствующая d_{-1i}, лежит в диапазоне от 0 до b-1, и, следовательно, для систем с основанием, большим 10, мы должны в соответствующих случаях в качестве цифр брать буквы.

Если провести те же действия над дробной частью результата $b\ N_F$, т. е. умножить его на b, то можно будет определить следующую цифру разложения дроби N_F .(2.11). А именно, поскольку

$$
\begin{aligned}
b\,N'_F &= d_{-2}b^0 + d_{-3}b^{-1} + \ldots + d_{-m}b^{-m+2} \\
&= d_{-2} + d_{-3}b^{-1} + \ldots + d_{-m}b^{-m+2} \\
&= d_{-2} + N''_F
\end{aligned}
\qquad (2.11)
$$

то целая часть произведения соответствует d_{-2}. Очевидно, повторяя описанный процесс, мы сможем определить последующие цифры числа в системе по основанию b. Процесс прекращается, если получается нулевая дробная часть. Однако в отличие от процесса преобразования целых чисел, всегда заканчивающегося через конечное число шагов, процесс преобразования десятичной дроби может быть бесконечным. Другими словами, представление десятичной дроби с конечным числом цифр может иметь бесконечное число цифр в системе счисления с другим основанием. Поэтому в любом случае процесс преобразования останавливают при достижении требуемой точности.

Приведем два примера преобразования десятичных дробей в разные системы счисления. Сначала, рассмотрим перевод числа 0.6875 в двоичную форму:

Произведения	Значения разрядов
$2 \times 0{,}6875 = 1{,}3750$	$d_{-1} = 1$
$2 \times 0{,}375 = 0{,}750$	$d_{-2} = 0$
$2 \times 0{,}75 = 1{,}50$	$d_{-3} = 1$
$2 \times 0{,}5 = 1{,}0$	$d_{-4} = 1$

В результате мы получаем $0{,}6875_{10} = 0.d_{-1}\,d_{-2}\,d_{-3}\,d_{-4} = 0{,}1011_2$.

Тепер переведём десятичную дробь 0,8435 в шестнадцатиричную систему. Необходимоо ввыполнить следующую цепочку умножений:

Произведения	Значения разрядов
$16 \times 0{,}8435 = 13{,}496$	$d_{-1} = D$
$16 \times 0{,}496 = 7{,}936$	$d_{-2} = 7$
$16 \times 0{,}936 = 14{,}976$	$d_{-3} = E$
$16 \times 0{,}976 = 15{,}616$	$d_{-4} = F$

Остановив процесс на этом шаге, мы получим $0.8435_{10} = 0.d_{-1}\,d_{-2}\,d_{-3}\,d_{-4} = 0.\text{D7EF}\ldots_{16}$.

На этом примере видно, что процесс преобразования бесконечен, поскольку третья цифра дробной части на всех шагах равна 6.

Для смешанных десятичных чисел целая и дробная части обрабатываются порознь. Целая часть преобразуется последовательными делениями, а дробная – последовательными умножениями. Получающееся в результате смешанное число записывается в виде этих двух частей разделенных точкой.

Д.2.3. Выполнение операций в двоичной системе счисления

Двоичная система счисления является основной системой представления информации в памяти компьютера. В этой системе счисления используются цифры: 0, 1. Над числами в двоичной системе счисления можно выполнять все арифметические действия.

При этом используются следующие соглашения: (табл.. 2.3):

Таблица 2.3

Операции в двоичной системе счисления

Сложение	Вычитание	Умножение
0+0=0	0-0=0	0*0=0
1+0=1	1-0=1	1*0=0
0+1=1	1-1=0	0*1=0
1+1=10	10-1=1	1*1=1

Двоичная система счисления обладает такими же свойствами, что и десятичная, только для представления чисел используется не 10 цифр, а всего две. Для кодирования числа, участвующего в вычислениях, используется специальная система правил перевода из десятичной системы счисления в двоичную. В результате число будет записано двоичным кодом, т.е. представлено различным сочетанием всего двух цифр - 0 и 1.

Для сравнения рассмотрим число 45 для двух вариантов кодирования. При использовании в тексте это число потребует для своего представления 2 байта, т.к. каждая цифра будет представлена своим кодом в соответствии с таблицей ASCII. В шестнадцатеричной системе код для цифры 4 будет 43, в двоичной системе - 01000011, соответственно для цифры 5 - 53 и 01010011.

Таблица 2.4

Представление чисел в позиционных системах

Десятичная	Шестнадцатиричная	Двоичная
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Д.2.4. Способы кодирования информации

На уроках физики при рассмотрении какого-либо физического явления вы используете формулы. Формула - это своего рода математический код.

Рассмотренные примеры говорят о том, что для представления информации могут использоваться разные ***коды*** и, соответственно, надо знать определенные правила - законы записи этих кодов, т.е. уметь кодировать.

Код - набор условных обозначений для представления информации. Кодирование - процесс представления информации в виде кода.

В качестве источников информации может выступать человек, техническое устройство, предметы, объекты неживой и живой природы. Получателей сообщений может быть несколько или один.

Д.2.4.1. Использование двоичной системы для кодирования текстовой информации в ПК

Нажатие клавиши на клавиатуре приводит к тому, что сигнал посылается в компьютер в виде ***двоичного числа***, которое хранится в ***кодовой таблице***.

Кодовая таблица - это внутреннее представление символов в компьютере. Во всем мире в качестве стандарта принята ***таблица ASCII*** (American Standard Code for Information Interchange - Американский стандартный код для обмена информацией).

Для хранения двоичного кода одного символа выделен 1 байт = 8 бит. Учитывая, что каждый бит принимает значение 0 или 1, количество их возможных сочетаний в байте равно $2^8 = 256$.

Значит, с помощью 1 байта можно получить 256 разных двоичных кодовых комбинаций и отобразить с их помощью 256 различных символов. Эти комбинации и составляют таблицу ASCII (см. Приложение 6).

Например, вы нажимаете на клавиатуре латинскую букву S. В этом случае в память компьютера записывается код 01010011. Для вывода буквы S на экран в компьютере происходит декодирование – то есть, по этому двоичному коду строится (т.е. отображается в виде пикселов) изображение этого символа.

Д.2.4.2. Кодирование графической информации

Создавать и хранить графические объекты в компьютере можно двумя способами - как растровое изображение или как векторное изображение. Для каждого типа изображения используется свой способ кодирования.

Векторное изображение представляет собой графический объект, состоящий из элементарных отрезков и дуг. Положение этих элементарных объектов определяется координатами точек и длиной радиуса.

Для каждой линии указывается ее тип (сплошная, пунктирная, штрих-пунктирная), толщина и цвет. Информация о векторном изображении кодируется как обычная буквенно-цифровая и обрабатывается специальными программами.

Растровое изображение представляет собой совокупность точек, используемых для его отображения на экране монитора. Объем растрового изображения определяется умножением количества точек на информационный объем одной точки, который зависит от количества возможных цветов. Для черно-белого изображения информационный объем одной точки равен 1 биту, т.к. она может быть либо черной, либо белой, что можно закодировать двумя цифрами - 0 или 1. Рассмотрим, сколько потребуется бит для отображения цветной точки:

–для 8 цветов - 3 бита; для 16 цветов - 4 бита; для 256 цветов - 8 битов (1 байт). В таблице 2.5 показано кодирование цветовой палитры из 16 цветов. Разные цвета и их оттенки получаются за счет наличия или отсутствия трех основных цветов (красного, синего, зеленого) и их яркости. Каждая точка на экране кодируется с помощью 4 битов.

Таблица 2.5

Кодирование 16-цветной палитры.

Цвет	Яркость	Красный	Зелёный	Синий	Значения в системе счисления	
					10-ная	16-ная
Чёрный	0	0	0	0	0	0
Синий	0	0	0	1	1	1
Зелёный	0	0	1	0	2	2
Голубой	0	0	1	1	3	3
Красный	0	1	0	0	4	4
Фиолетовый	0	1	0	1	5	5
Коричневый	0	1	1	0	6	6
Белый	0	1	1	1	7	7
Серый	1	0	0	0	8	8
Светло-синий	1	0	0	1	9	9
Светло-зелёный	1	0	1	0	10	A
Светло-голубой	1	0	1	1	11	B
Светло-красный	1	1	0	0	12	C
Светло-фиолетовый	1	1	0	1	13	D
Жёлтый	1	1	1	0	14	E
Ярко-белый	1	1	1	1	15	F

Приложение 3

Характеристики языков программирования

Название языка программирования (ЯП)	Год созд.	Тип языка	Автор (ы) разработки	География	Органи-зация	Стандарт
Фортран (FORTRAN)	1954	A	Джон Бекус	Америка	IBM	ISO 1539:1997
Лисп (LISP)	1958	F	Джон Маккарти	Америка	MIT	–
Алгол-60 (Algol 60)	1960	A	Питер Наур+	Между-народн.	IFIP	–
Кобол (COBOL)	1960	A	+	Между-народн.	CODASYL Committee	ISO 1989:1985
Симула (Simula)	1962	B	Кристен Нигаард+	Европа	–	–
Бейсик (BASIC)	1963	A	Томас Курт и Джон Кемени+	Америка	Dartmouth College	ISO 10279:1991
ПЛ/1 (PL/I)	1964	A	Джордж Радин	Америка	IBM	ISO 6160:1979
Алгол-68 (Algol 68)	1968	A	Аад ван Вайнгартен+	Между-народн.	IFIP	–
Паскаль (Pascal)	**1970**	**C**	**Никлаус Вирт**	**Европа**	**ETH**	**ISO 7185:1990**
Форт (FORTH)	1970	A*	Чарльз Мур	Америка	Mohasco Industries	ISO 15145:1997
Си (C)	1972	C*	Деннис Ритчи	Америка	AT&T Bell Labs	ISO 9899:1999
Smalltalk	1972	B/OO	Аллан Кэй	Америка	Xerox PARC	–
Пролог (Prolog)	1973	E	Аллан Кольмеро+	Европа	Univers. of Aix-Marseille	ISO 13211:1995
Ада (Ada)	1980	H*	Джин Ишбиа+	Америка	CII Honeywell	ISO 8652:1995
Си++	1980/ 1984	H* –/OO	Бьёрн Страуструп	Америка	AT&T Bell Labs	ISO 14882:1998
Turbo Pascal	**1983**	**–/–**	**Филипп Кан, Андерс Хейльсберг (Anders Hejlsberg)**	**Америка**	**Borland International**	**–**
Perl	1986	ЯС/OO	Ларри Вол (Larry Wall)	Между-народн.	–	–
HTML	1989	I	Тим Барнерс-Ли	Европа	CERN	–
Python	1990 (99v2.0)	ЯС/OO	Гвидо ван Россум (Guido van Rossum)	Голландия/ Америка	–	–
Visual Basic	1991	ЯС/OO	–	Америка	Microsoft	–
Delphi/Object Pascal	**1995**	**–/OO**	**Андерс Хейльсберг**	Америка	**Borland International**	**–**
Java	1995	H ЯС/OO	Патрик Нотон и Джеймс Гослинг	Америка	Sun Labs	–
Java Script	1995	ЯС/OO	–	Америка	–	–
VBScript	1997	ЯС/OO	–	Америка	Microsoft	–
Cold Fusion	1997	–/OO	–	Америка		–
XML	1998	I	–	Америка	W3C, World Wide Web Consortium	–
C# (Си Шарп)	**2000**	**H* /OO/D**	**Андерс Хейльсберг+**	**Америка**	**Microsoft**	**–**

ЭКСПЕРИМЕНТАЛЬНЫЕ И КОРПОРАТИВНЫЕ ЯЗЫКИ

Название языка программирования (ЯП)	Год	Тип языка	Автор (ы) розработки	География	Организация	Стандарт
АПЛ (APL)	1957	I	Кеннэт Айверсон	Америка	Harvard Univ.	ISO 8485:1989
Снобол (Snobol)	1962	I	Ральф Грисуолд	Америка	AT&T Bell Labs	–
Сетл (SETL)	1969	I	Джек Шварц	Америка	IBM	–
Параллельний Паскаль (Concurrent Pascal)	**1974**	**G**	**Пер Бринч Хансен**	**Америка**	**CIT**	**–**
CLU	1974	D	Барбара Лисков	Америка	MIT	–
Scheme	1975	F	Гай Стил+	Америка	MIT	–
Mesa	1976	D*	Дж. Мичел+	Америка	Xerox PARC	–
Icon	1977	I	Ральф Грисуолд	Америка	AT&T Bell Labs	–
Модула-2 (Modula-2)	**1979**	**D***	**Никлаус Вирт**	**Европа**	**ETH**	ISO 10514:1996
Оккам (Occam)	1982	G*	Дэвид Мэй+	Европа	Inmos	–
Cedar	1983	H*	Батлер Лэмпсон+	Америка	Xerox PARC	–
Common Lisp	1984	F	Гай Стил+	Америка	MIT	–
Objective C	1986	H*	Брэд Кокс	Америка	Productivity Products	–
Эйфель (Eiffel)	1986	D*	Бертран Мейер	Европа	ISE	–
Оберон (Oberon)	**1988**	**D***	**Никлаус Вирт**	**Европа**	**ETH**	**–**
Модула-3 (Modula-3)	1988	H*	Билл Калсов+	Америка	DEC SRC	–
Оберон-2 (Oberon-2)	**1991**	**D***	**Ханспетер Мёссенбёк+**	**Европа**	**ETH**	**–**
Limbo	1996	D*	Деннис Ритчи	Америка	Bell Labs (Lucent)	–
Component Pascal	**1997**	**D***	**Куно Пфистер+**	**Европа**	**Oberon microsystems**	**–**

УСЛОВНЫЕ ОБОЗНАЧЕНИЯ

Серым цветом выделены языки семейства Паскаль-

Виды языков (парадигмы)

A - процедурное программирование;
B - объектно-ориентированное программирование;
C - структурное программирование;
D - модульное (компонентное) программирование;
E - логическое (реляционное) программирование;
F - функциональное программирование;
G - параллельное программирование;
H - гибрид (смесь парадигм; B+C+D+G);
I - специализированные языки;
ЯС - язык скриптов;
ОО - объектно-ориентированный язык.

Сокращения

MIT - Massachusetts Institute of Technology
PARC - Palo Alto Research Center
ETH - Swiss Federal Institute of Technology
SRC - Systems Research Center
ISE - Interactive Software Engineering
ISO - International Standard Organization
CIT - California Institute of Technology
* – Поддержка системного программирования.

\+ Несколько авторов языка.

Приложение 4

Уровни развития сетей в информационно-компьютерных технологиях

Сеть

Прежде, чем приступить к рассмотрению межсетевого взаимодействия, уточним, что понимается под термином "***сеть***". Этот термин может употребляться в широком смысле (сеть - это совокупность связанных между собой компьютеров) и в узком смысле (сеть - это совокупность компьютеров, соединенных между собой в соответствии с одной из стандартных типовых топологий - шина, звезда, кольцо, и использующих для передачи пакетов один из известных протоколов канального уровня, определенный для этой топологии).

Каждая сеть имеет свой номер, который используется на сетевом уровне при выполнении маршрутизации. Когда две или более сетей организуют совместную транспортную службу, то такой режим взаимодействия обычно называют ***межсетевым взаимодействием (internetworking)***. Для обозначения составной сети в англоязычной литературе часто также используются термины **интерсеть** (internetwork или *internet)*. Интерсеть обеспечивает только передачу пакетов, не занимаясь их содержанием. *Internetwork* является объединением отдельных сетей, соединённых промежуточными сетевыми устройствами и работающая как единая большая сеть. (рис. П.4.1).

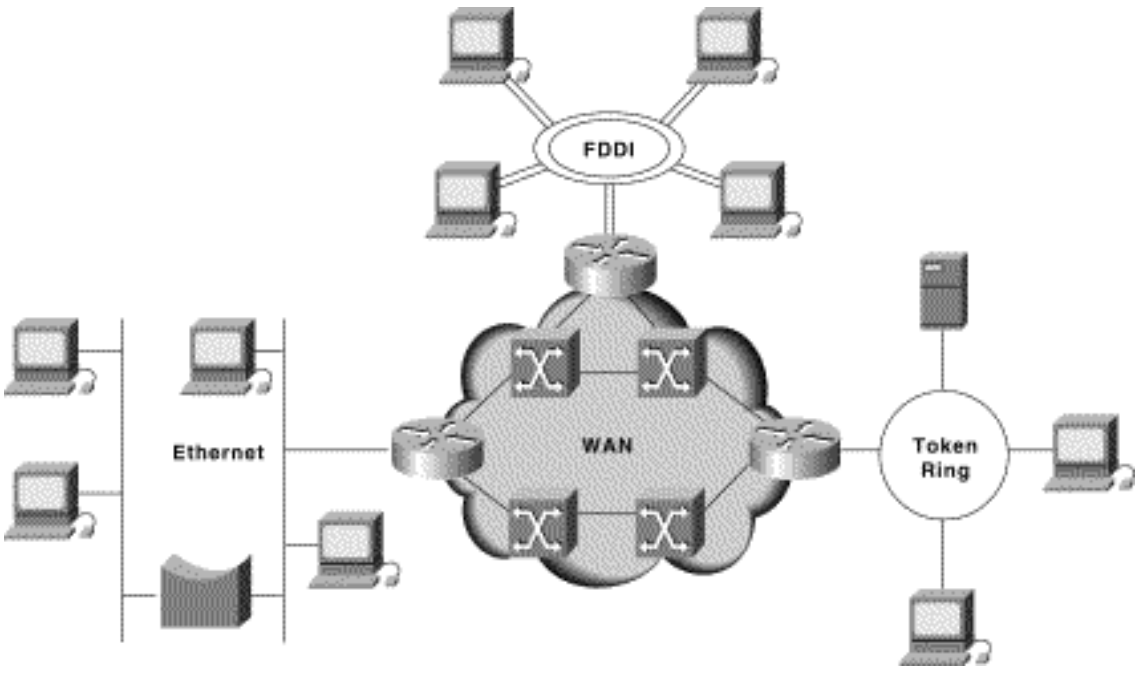


Рис. Д.4.1. Пример взаимодействия разных по составу сетей в едином комплексе интерсети WAN

Объединение сетей (*Internetworking*) относится к отраслям, продуктам и процедурам, которые отвечают требованиям создания и администрирования сетевых комплексов.

Возникновение интерсетей происходило в основном случайным образом. При этом, приобретённые компьютеры и ОС, отвечали индивидуальным потребностям групп пользователей. Сети отделов строились для решения конкретных задач групп сотрудников. Например, инженерный отдел мог выбрать рабочие станции SPARC фирмы Sun Microsystems, соединённые сетью Ethernet, потому что им были необходимы приложения, которые бы работали только в среде UNIX. Распределение файлов при этом реализовывалось с помощью протоколов TCP/IP и NFS. В отделе продаж этой же самой организации уже могли быть к этому времени приобретены компьютеры PS/2, установлена сеть Token Ring и операционная система NetWare для решения их собственных задач: ведения базы данных о клиентах, подготовки писем, разработки коммерческих предложений. Потом в рекламном отделе были выбраны компьютеры Macintosh, поскольку они лучшим образом подходят для создания презентационных материалов. Macintosh'ы соединялись с помощью LocalTalk, а файлы и принтеры распределялись с использованием сетевых средств AppleTalk. Отдел, отвечающий за автоматизацию предприятия, должен был ***интегрировать*** все эти в целом несовместимые системы в единый прозрачный сетевой организм.

Гетерогенность.

Таким образом, только небольшое количество сетей имеет ***однородность (гомогенность)*** программного и аппаратного обеспечения. Однородными чаще всего являются сети, состоящие из небольшого количества компонентов от одного производителя.

Нормой сегоднешнего дня являются сети ***неоднородные (гетерогенные)***, состоящие из разных рабочих станций (Intel, Sun, IBM, Hewlett-Packard, Dell, COMPAQ), операционных систем (Windows, Unix, Linux, Sun, Macintosh) и всевозможных приложений многочисленных производителей ПО, а для реализации взаимодействия между компьютерами используются разные протоколы. Разнообразие всех компонентов, из которых строится сеть, порождает ещё большее разнообразие структур сетей, вытекающих из наличия этих компонентов. Структурные и организационные объединения компьютеров в единый комплекс имеют следующие названия.

Local Area Network (LAN) – локальная сеть, соединённых между собой рабочих станций совместно использующих ресурсы процессора или сервера в пределах относительно небольшого географического пространства. Предназначена для совместного использования ресурсов файл-серверов для обмена файлами и сообщениями, а также сетевых принтеров. Может обслуживать от нескольких до нескольких тысяч пользователей.

Wide Area Network (WAN) – физическая коммуникационная сеть, связывающая географически удаленные друг от друга компьютеры и сетевые

сегменты. Характеризует более широкую телекоммуникационную структуру, чем LAN. Может состоять из сетей частных компаний, а также включать государственные сети. Обязательно включает все средства передачи.

Интранет (Intranet) – является сетью, которая расположена в пределах предприятия. Он может состоять из многих связанных между собой локальных сетей (LAN), а также использовать арендованные линии в WAN. Он также может включать или не включать соединения через один или несколько шлюзов с внешним Интернетом. Основным назначением Интранета является объединение информации и вычислительных мощностей (средств) предприятия и обеспечения ими его работников. Интранет может также использоваться для обеспечения групповой работы и проведения телеконференций.

Экстранет (Extranet) – объединённая сеть, которая использует Интернет-технологии для соединения фирм и предприятий с их поставщиками, клиентами или другими фирмами, связанными общими целями. Экстранет можно представить в виде части Интранет′а компании, которая сделана доступной для других компаний или уже является собственность нескольких компаний. Общая для них информация доступна только для участников комплекса или может открываться для доступа по особым соглашениям.

Интернет (Internet) – глобальная сеть, соединяющая многие миллионы компьютеров. Более 100 стран объединяются ею для обмена данными, новостями и другой информацией. Является децентрализованной сетью (не имеющей одного главного компьютера). Каждый Интернет компьютер называется ***хостом (host)*** и абсолютно независим. Интернет включает все материальные составляющие, включая коммуникации. Следует отметить, что Интернет не является синонимом World Wide Web (WWW, мировая паутина). WWW – это только один из его сервисов.

World Wide Web (WWW ***– мировая паутина)*** – ***сервис Интерне***т, поддерживающий функции сохранения и доставки по всей сети специальным образом форматированных документов. Документы форматируются средствами скриптового языка *HTML (HyperText Markup Language),* поддерживающего ссылки на другие документы (текст, графику, аудио-, видео и т.д.), расположенные в любой части мира на любом компьютере-сервере – мощном компьютере с программой-сервером (этот компьютер должен быть включён в данный момент времени для обеспечения доступа, то есть функционировать). Сервера, на которых располагаются документы, могут также содержать другие разнообразные и многочисленные данные, как правило в заархивированном виде. Эти данные можно "скачивать" на компьютер пользователя для дальнейшей работы. Документы просматриваются на компьютерах пользователей с помощью приложений, которые называются браузерами. ***Браузер*** – это программа-клиент, позволяющая пользователям читать HTML-документы с локального диска и Internet с помощью сервиса WWW, а также осуществлять навигацию между ними. Примерами программ-браузеров являются следующие: MS Internet Explorer, Netscape Navigator, Opera и другие.

К другим сервисам Интернет можно отнести: ***E-mail (электронная почта)***, ***FTP (передача файлов)***, *Telnet* ***(телеконференции)*** и др.

Приложение 5

Команды интегрированной среды разработки Turbo Pascal 7.0

Управляющие команды (“горячие клавиши”)	
F1	-получить справку / помощь
F2	-сохранить текущий файл, котрый редактируется, на диск
F3	-читать файл с диска
F4	-исполнить программу до строки, где расположен курсор
F5	-(вкл/откл) совмещение окон редактора и отладчика
F6	-циклическая смена окон (переход из окна в окно)
F7	-трассировка подпрограммы
F8	-пооператорное выполнение программы
F9	-компилировать программу
F10	-переход в верхнее (главное меню)
Alt–X	-выход в DOS (конец работы)
Alt–0	-показать список активных окон
Alt–F1	-показать последний экран подсказки
Alt–F3	закрыть текущее окно
Alt–F5	-показать результаты выполнения программы
Alt–F9	-компилировать текущий файл
Alt–[первая буква названия меню]–вызывает соответствующее меню: (**F**ile, **R**un, …)	
Ctrl–F1	-подсказка по слову
Ctrl–F2	-закончить процесс отладки
Ctrl–F3	-показать стек
Ctrl–F4	-вычислить выражение или изменить значение переменной
Ctrl–F5	-перемещение окна или изменение его размеров
Ctrl–F9	-запуск задачи на выполнение (на счёт)
Esc	-закрытие диалогового окна
PrintScre	-печать копии экрана на принтере
Shift–F6	-переключение активных окон
Ctrl–Break	-прерывание выполняемой программы
Alt–R–Enter	-запуск задачи на выполнение
Shift– "клавиши стрелки"	-виделить фрагмент программы
Alt–BackSpace	-отмена последнего редактирования

Команды работы с блоком	
Ctrl–K B	-пометить начало блока
Ctrl–K K	-пометить конец блока
Ctrl–K T	-пометить слово слева от курсора
Ctrl–K P	-напечатать блок
Ctrl–K C	-копировать блок в позицию курсора
Ctrl–K V	-переместить блок
Ctrl–K H	-(убрать/вновь выделить) пометку блока
Ctrl–K Y	-удалить блок
Ctrl–K R	-читать блок из дискового файла
Ctrl–K W	-записать блок на диск
Ctrl–K I	-сместить блок вправо
Ctrl–K U	-сместить блок влево

Команды удаления/вставки	
Shift–Del	-переместить выделенный блок в буфер обмена (Clipboard)
Ctrl–Ins	-скопировать выделенный блок в буфер обмена (Clipboard)
Ctrl–Del	-удалить выделенный блок
Shift–Ins	-вставить блок из буфера Clipboard в позицию курсора
Ctrl–V	-вкл/выкл. режим вставки
Ctrl–N	-вставить строку
Ctrl–Y	-удалить строку
Ctrl–H	-стереть символ слева от курсора
Ctrl–G	- стереть символ над курсором
Ctrl–T	- стереть слово справа от курсора

Приложение 6

Коды ASCII (0-127)
(American Standard Code for Information Interchange)

DEC	CHAR	Name	DEC	CHAR	DEC	CHAR	DEC	CHAR
0	Ctrl-@	NUL	32	SPC	64	@	96	'
1	Ctrl-A	BOH	33	!	65	A	97	a
2	Ctrl-B	STX	34	"	66	B	98	b
3	Ctrl-C	ETX	35	#	67	C	99	c
4	Ctrl-D	EOT	36	$	68	D	100	d
5	Ctrl-E	ENQ	37	%	69	E	101	e
6	Ctrl-F	ACK	38	&	70	F	102	f
7	Ctrl-G	BEL	39	'	71	G	103	g
8	Ctrl-H	BS	40	(	72	H	104	h
9	Ctrl-I	HT	41	)	73	I	105	i
10	Ctrl-J	LF	42	*	74	J	106	j
11	Ctrl-K	VT	43	+	75	K	107	k
12	Ctrl-L	PF	44	,	76	L	108	l
13	Ctrl-M	CR	45	-	77	M	109	m
14	Ctrl-N	SO	46	.	78	N	110	n
15	Ctrl-O	SI	47	/	79	O	111	o
16	Ctrl-P	DLE	48	0	80	P	112	p
17	Ctrl-Q	DC1	49	1	81	Q	113	q
18	Ctrl-R	DC2	50	2	82	R	114	r
19	Ctrl-S	DC3	51	3	83	S	115	s
20	Ctrl-T	DC4	52	4	84	T	116	t
21	Ctrl-U	NAK	53	5	85	U	117	u
22	Ctrl-V	SYN	54	6	86	V	118	v
23	Ctrl-W	ETB	55	7	87	W	119	w
24	Ctrl-X	CAN	56	8	88	X	120	x
25	Ctrl-Y	EM	57	9	89	Y	121	y
26	Ctrl-Z	SUB	58	:	90	Z	122	z
27	Ctrl-[	ESC	59	;	91	[	123	(
28	Ctrl-\	FS	60	<	92	\	124	|
29	Ctrl-]	GS	61	=	93	]	125	)
30	Ctrl-^	RS	62	>	94	^	126	~
31	Ctrl-_	US	63	?	95	_	127	DEL

Примечание: В таблице: DEC – десятичный номер символа; CHAR – символы, которые кодируются. Например, символ ″4″ в таблице имеет десятичный номер 52, что в двоичном эквиваленте (см. Приложение 2) равно: 110100, а в восьмибитном эквиваленте – *00*110100 (слева добавлено два нуля до 8-ми битов).

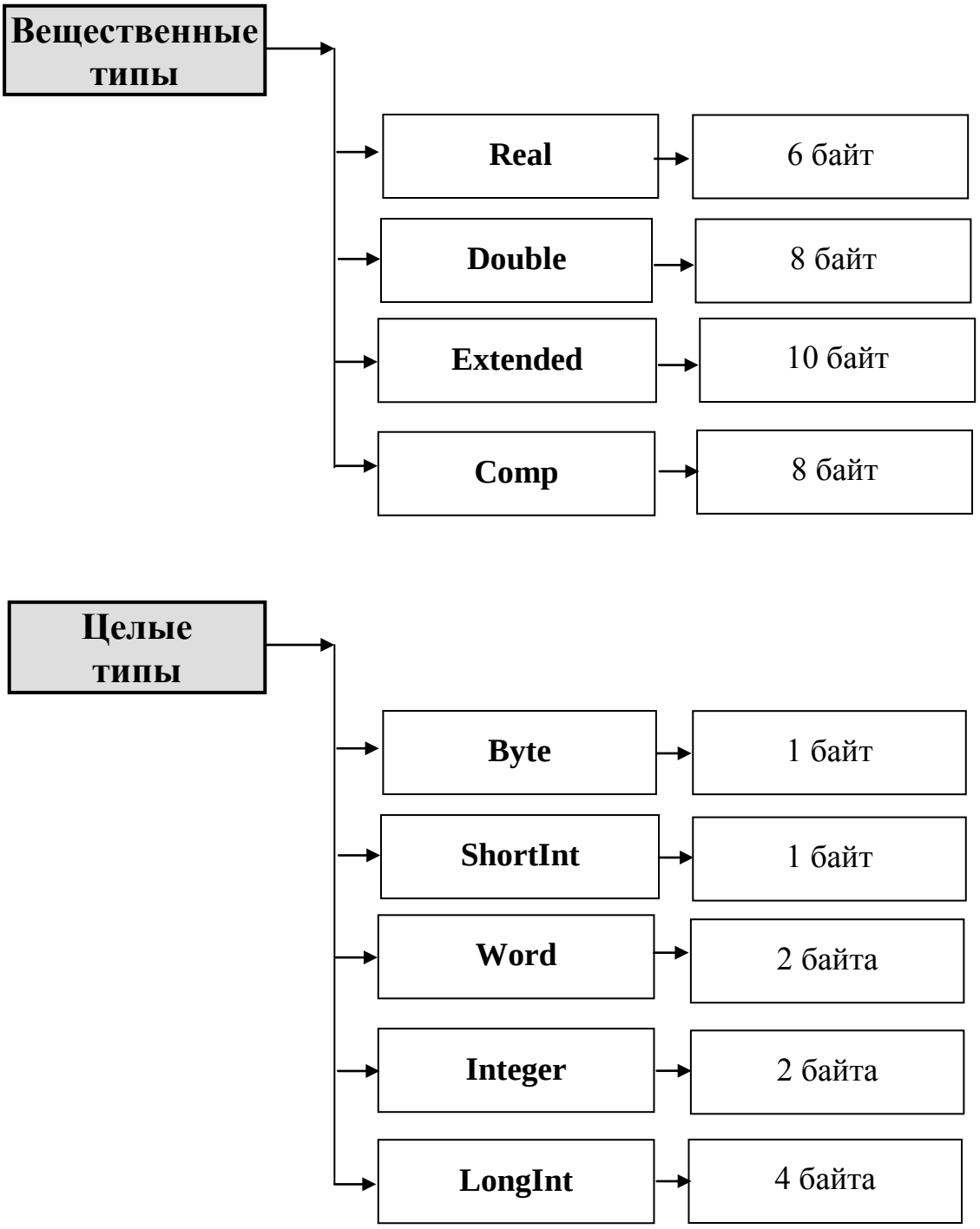
Приложение 7

Базовые (встроенные) типы данных Турбо Паскаля

Основные типы данных языка Турбо Паскаль. К основным, то есть к базовым типам, относятся четыре: Действительные, Целые, Символьный и Логический. Первые два содержат подтипы, отличающиеся величиной переменных и местом (в байтах), которое каждый из типов занимает в памяти (см. Диаграммы ниже). Три из них (Целые, Символьный и Логический) являются порядковыми, то есть объединяют данные, каждое из которых всегда имеет фиксированное место в последовательности (Например: 1,2, ... и т.д.). Из

этих трёх типов строятся два пользовательских типа: Тип-диапазон и Перечислимый тип.

Разновидности Действительного (Real) и Целого (Integer) типов и количество байтов, занимаемых ими в памяти компьютера.



Приложение 8

ТИПОВЫЕ ЛАБОРАТОРНЫЕ РАБОТЫ

1. Технология работы в среде Турбо Паскаль (ТП) версии 7.0. Выполнение программы с базовыми типами данных (констант и переменных).

2. Изучение общей структуры программы. Разработка простого интерфейса пользователя программы.

3. Решение задач, базирующихся на использовании выражений, операндов и операций языка ТП.

4. Инициализация данных перед вычислением выражений. Действительные (`Real`) и целые (`Integer`) типы данных. Операции и встроенные функции работы с этими типами данных.

5. Логические типы данных (`Boolean`). Операции и встроенные функции работы с ними. Конструирование логических выражений для формирования логики работы программ. Использование логических операций и операций отношения для записи сложных условных выражений.

6. Управляющие структуры (операторы) языка ТП. Простые операторы. Сложные (структурные) операторы управления выполнением алгоритмов. Составной оператор. Операторы ветвления алгоритмов. Условный оператор `if`. Оператор выбора `case`.

7. Циклические вычислительные процессы и операторы циклов. Циклы с параметром. Оператор цикла с параметром `for`. Оператор цикла с предусловием `while`. Оператор цикла с послеусловием `repeat`. Средства исследования выполнения действий программы с помощью дебаггера.

8. Моделирование в циклических вычислениях некоторых типовых выражений. Особенности вычисления бесконечных сумм.

9. Организация итерационных процессов с помощью циклов `while` и `repeat`. Бесконечные умножения и их вычисление.

10. Подпрограммы: процедуры и функции. Формальные и фактические параметры. Передача параметров “по значению” и “по адресу `(val)`”.

11. Программирование задач по алгоритмам численных методов, путём реализации в основной программе соответствующих процедур и функций:

– вычисление определённых интегралов методом трапеций и Симпсона;

– вычисление трансцендентных уравнений методом дихотомии;

– решение задач интерполяции и экстраполяции и т. д.

Литература

К главе 1. Долгий путь к персональному компьютеру

1. Fortune 500. Rank Company Revenues. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.fortune.com
2. Glossary of Terms. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.pcwebopaedia.com
3. Гуриев В. История компьютеров. Краткий курс. Журнал Компьютерра, № 12, 2000, – С. 56-63.
4. Интернет-версия издания: Шауцукова Л.З. Информатика 10 - 11. — М.: Просвещение, 2000. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.kbsu.ru/%7Ebook/theory/chapter1/1_1_3.html
5. История создания Windows. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://coop.chuvashia.ru/SanyaSoft/KAFEDRA/iags.3x/Lections/ Chap1_2.htm
6. Знакомьтесь: компьютер. Пер. с англ. Под ред. В.М. Курочкина. – М.: Мир, 1989. – 240 с.
7. Микро-ЭВМ / Пер. с англ. Под ред. А. Дирксена. –М.: Энергоиздат, 1982. – 328 с.
8. Модернизация и ремонт ПК. 6-е изд. –К.: Диалектика, 1997. – 679 с.
9. Новейший самоучитель работы на компьютере / Под ред. Симоновича С.В. –СПб.: Изд-во DECC-КОМ, 2001. – 654 с.
10. Работа и история персонального компьютера**.** WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://mii-corp.newmail.ru/ W_E_S_PC/ microsoft.htm
11. Работа и история персонального компьютера. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://mii-corp.newmail.ru/W_E_S_PC/apple.htm
12. Работа и история персонального компьютера. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://mii-corp.newmail.ru/W_E_S_PC/intel.htm
13. Работа и история персонального компьютера. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://mii-corp.newmail.ru/W_E_S_PC/ibm.htm

К главе 2. Изящество процесса включения персонального компьютера

1. MS-DOS 6.0. Справочное руководство для пользователей компьютеров IBM PC. - М.: Продукция фирмы "ВА Принт", 1994. – 319 с.
2. Айден К., Фибельман Х., Крамер М. Аппаратные средства PC: Пер.с нем. – СПб.: BHV-Санкт-Петербург, 1996. – 544 с.
3. Богумирский Б. Norton Commander 5.0. Новые возможности для пользователя. – СПб.: Питер, 1995. – 288 с.
4. Богумирский Б.. Эффективная работа на IBM PC. –СПб.:Питер, 1995. - 688 с.
5. Брябрин В.М. Программное обеспечение персональных ЭВМ. – М.: Наука. Гл. ред. физ.-мат. лит., 1990. – 272 с.

6. Букчин Л.В., Безрукий Ю.Л. Дисковая подсистема IBM совместимых персональных компьютеров. –М.: МП «БИНОМ», 1993. – 284 с.

7. Войников Н.А. Системное программирование для Правец-16. – София: Техника, 1990. – 256 с.

8. Данкан Р. Профессиональная работа в MS-DOS /Пер. с англ. –М.: Мир, 1993. – 509 с.

9. Дейтел Г. Введение в операционные системы. В 2-х т. Т. 1. / Пер. с англ. – М.: Мир, 1987. – 359 с.

10. Дейтел Г. Введение в операционные системы. В 2-х т. Т. 2. / Пер. с англ. – М.: Мир, 1987. – 398 с.

11. Краковяк С. Основы организации и функционирования ОС ЭВМ: Пер. с франц. –М.: Мир, 1988. – 480 с.

12. Мюллер, Скотт. Модернизация и ремонт ПК. –М: Издательство Que, 2001. – 1128 с.

13. Нортон П. Программно-аппаратная организация IBM PC / Пер. с англ. / Под ред. В.Г. Абрамова. – М.: Радио и связь, 1991. – 328 с.

14. Нортон П. Персональный компьютер фирмы IBM и операционная система MS-DOS / Пер. с англ. – М.: Радио и связь, 1992. – 416 с.

15. Нортон П., Джорден Р. Работа с жестким диском IBM PC. – М.: Мир, 1992. – 560 с.

16. Операционные системы: Учеб. пособие / В.П. Грибанов, С.В. Дробин, В.Д. Медведев. –М.: Финансы и статистика, 1990. – 239 с.

17. Питер Нортон Руководство по DOS Питера Нортона / Пер. с англ. – М.: БИНОМ, 1995. – 480 с. (Версии 6.2 и 6.22).

18. Рош У.Л. Библия по техническому обеспечению Уинна Роша / Пер. с англ. –Мн.: МХКК «Динамо», 1992, – 416 с.

19. Рош Л.У. Библия по модернизации персонального компьютера / Пер. с англ. –Мн.: ИПП «Тивали-Стиль», 1995, – 208 с.

20. Скэнлон Л. Персональные ЭВМ IBM PC и XT. Программирование на языке ассемблера / Пер с англ. –М.: Радио и связь. 1989. – 336 с.

21. Фигурнов В.Э. IBM PC для пользователя, 2-е изд., перераб. и доп. –М.: Финансы и статистика, 1991. – 288 с.

22. Фигурнов В.Э. IBM PC для пользователя. Изд. 7-е. - М.: ИНФРА-М, 1997. – 640 с., ил.

23. Фигурнов В.Э. IBM PC для пользователя. Краткий курс. - М.: ИНФРА-М, 1999. – 480 с.

24. Финогенов К.Г. Самоучитель по системным функциям MS-DOS. –М.: МП «МАЛИТ», 1993. – 262 с.

25. Франкен Г. MS-DOS 5.0 для пользователя : Пер. с нем. –2-е изд. перераб. –К.: Торгово-издательское бюро BHV, 1992. – 513 с.

26. Фролов А.В., Фролов Г.В. Операционная система MS-DOS: В 3 кн. Кн. 1-2. – М.: "ДИАЛОГ-МИФИ", 1991. – 240 с.

27. Фролов А.В., Фролов Г.В. Операционная система MS-DOS: В 3 кн. Кн. 3. – М.: "ДИАЛОГ-МИФИ", 1991. – 224 с.

28. Эско Валтанен. Дисковые операционные системы для ПЭВМ. –К.: Региональный центр перевода и информационных услуг. 1992. – 744 с.

К главе 3. Командная основа работы компьютера

1. Component Object Model, Part II: Programming Interface Version 0.9 (Draft) October 24, 1995 Microsoft Corporation and Digital Equipment Corporation. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: www.graphcomp.com/info/specs/com/comch02.htm

2. Dave Anderson. WELCOME TO THE PC TECHNOLOGY GUIDE! WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.pctechguide.com/

3. Levy, H. 1984. Capability-Based Computer Systems. Bedford, MA: Digital Press, – p. 13.

4. Yonezawa, A. and Tokoro, M. 1987. Object-Oriented Concurrent Programming. Cambridge, MA: The MIT Press. – p. 2.

5. Берзтисс А.Т. Структуры данных / Пер. с англ. –М.: Статистика, 1974. – 403 с.

6. Брукс Фредерик. Мифический человеко-месяц или как создаются программные комплексы. – Пер. с англ. – СПб.: Символ-Плюс, 1999, – 304 с.

7. Буч Гради. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. – М.: БИНОМ, СПб.: Невский диалект, 1998. – 560 с.

8. Борн Гюнтер. Форматы данных/ Пер. с нем. –К.: Торгово–издательское бюро BHV, 1995. – 472 с.

9. Войников Н.А. Системное программирование для Правец-16. – София: Техника, 1990. –256 с.

10. Гивоне Д., Россер Р. Микропроцессоры и микрокомпьютеры: Вводный курс: Пер. с англ. – М.: Мир, 1983. – 464 с.

11. Головач Влад. В. Дизайн пользовательского интерфейса. –147 с. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.journals.ru/users/webdev (http://progs-maker.narod.ru/other.html)

12. Громов Г.Р. Национальные информационные ресурсы: проблемы промышленной эксплуатации. – М.: Наука, 1984. – 240 с.

13. Данкан Р. Профессиональная работа в MS-DOS /Пер. с англ. –М.: Мир, 1993. – 509 с.

14. Информатика. Базовый курс/ Симонович С,В. и др. –СПб.: Издательство «Питер», 1999. – 640 с.

15. Информатика: Учебник / Под редакцией Н.В. Макаровой. – М.: Финансы и статистика, 1997. – 768 с.

16. Новейший самоучитель работы на компьютере / Под ред. Симоновича С.В. –СПб.: Изд-во DECC-КОМ, 2001. – 654 с.

17. Основы современных компьютерных технологий: Учебное пособие / Под ред. проф. Хомоненко А.Д.. Авторы: Артамонов Б.Н., Брякалов Г.А., Гофман В.Э., Кадигроб Я.Е., Компаниец Р.И., Липецких А.Г., Мальцев М.Г.,

Рыжиков Ю.И., Хомоненко А.Д., Цыганков В.М. – СПб.: КОРОНА принт, 1998. – 480 с.

18. Программирование микропроцессорных систем: Учеб. пособие для вузов по спец. «Ав томатиз. сист. обр.информ. и упр.» / В.Ф. Шаньгин, А.Е. Костин, В.М. Илюшечкин, П.А. Тимофеев; Под ред. В.Ф. Шаньгина. –М.: Высш. шк., 1990. – 330 с.

19. Рамбо Джеймс, Якобсон Айвар, Буч Грэди. Язык UML: Руководство пользователя : Пер. с англ. –М.: ДМК, 2000. – 432 с.

20. Справочник по математике для инженеров и учащихся втузов. Бронштейн И.Н., Семендяев К.А. –М.: Наука, Главная редакция физико-математической литературы. 1981. –718 с.

21. Фрир Дж. Построение вычислительных систем на базе перспективных микропроцессоров: Пер. с англ. –М.: Мир, 1990. – 413 с.

22. Хильер Скот. Создание приложений COM+ в среде Visual Basic. Руководство разработчика : Пер. с англ. : – М.: Издательский дом «Вильямс», 2001. – 416 с.

23. Цикритзис Д., Лоховский Ф. Модели данных. М.: Финансы и статистика, 1985. – 344 с.

24. Шагурин И.И.. Микропроцессоры и микроконтроллеры фирмы "Motorola": Справочное пособие. - М.; Радио и связь, 1998. – 560 с.

25. Шафрин Ю. Основы компьютерной технологии. Учеб. пособ. –М.: АВФ, 1997. – 655 с.

К главе 4. Концепции интерфейса

1. Dan R. Olsen Jr., Morgan Kaufmann. Developing User Interfaces. –NY: Sunsoft Press, Prentice Hall 1998. –637 p.

2. David Frost, "Psychology and Program Design", Datamation, May 1975, - P. 137. Reprinted with the permission of DATAMATION® Copyright 1975 by Technical Publishing Company, Greenwich, Connecticut 06830.

3. Kevin Mullet & Darrell Sano, Designing Visual Interfaces - communication oriented techniques. –NY: Sunsoft Press, Prentice Hall, 1995. – 457 p.

4. Microsoft Corporation. Компьютерные сети. Учебный курс./ Пер. с англ. - М.: Изд. отдел "Русская редакция" ТОО "Channel Trading Ltd.", 1997. – 696 с.

5. Norman D.. The Design of Everyday Things. –New York, Doubleday, 1998. – 187 p.

6. Бусигин Б.С., Коротенко Г.М., Коротенко Л.М. Модель образного восприятия в информатике как составляющая повышения качества инженерного образования // Науковий вісник НГА України, №6, 2000, с. 3-9.

7. Васкевич Д. Стратегии клиент/сервер. Руководство по выживанию для специалистов по реорганизации бизнеса. – К.: "Диалектика", 1996. – 384 с.

8. Головач Влад. В. Дизайн пользовательского интерфейса. –147 с. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.journals.ru/users/webdev (http://progs-maker.narod.ru/other.html)

9. Головач Влад. В. 5 ПРАВИЛ хорошего интерфейса. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.osp.ru/publish/2000/06/073.htm

10. Калиниченко Л.А., Когаловский М.Р. Стандарты OMG: Язык определения интерфейсов IDL в архитектуре CORBA // Системы Управления Базами Данных · № 2/96. – С. 115-129.

11. Коротенко Г.М., Коротенко Л.М. Графический интерфейс для визуализации результатов решения задач динамики сложных механических систем // Препринт ИТМ АН УССР, г. Днепропетровск: ИТМ АН УССР, 1989. – 39 с.

12. Коутс Р., Влейминк И. Интерфейс «человек-компьютер» / Пер. с англ. – М.: Мир, 1990. – 512 с.

13. Основы информационных систем. Часть 1. Интерфейсы. –105 с. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.citforum.primorye.ru/operating_systems/ois/introd.shtml

14. Формирование языка взаимодействия объектов в пользовательском интерфейсе, основанном на метафоре модели мира. WEB-сайт (Электрон. ресурс) / Способ доступа URL: http://www.ostu.ru/nit/staff/ag/public1.htm

15. Хэйес Фрэнк. Distributed Component Object Model. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.citforum.ru/programming/application/dcobjmod.shtml

16. Эндрю вэн Дам. Пользовательские интерфейсы нового поколения. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://javaworld.osp.ru/os/1997/06/34.htm

К главе 5. Эволюция языков программирования

1. Albahari Ben. A Comparative Overview of C#. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://genamics.com/developer/csharp_comparative.htm

2. Anders Hejlsberg. From Wikipedia, the free encyclopedia. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.wikipedia.org/wiki/Anders_Hejlsberg

3. Kalev Danny. Интервью с Бьерном Страуструпом. Будущее за мультипарадигматическим программированием. WEB-сайт (Электрон. ресурс) / Способ доступа URL:http://www.softcraft.ru/paradigm/common/siw.shtml

4. Mauny M.. Functional Programmig using Caml Light. January 1995. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://cristal.inria.fr/tutorial/index.html

5. MS Office 2000 для разработчиков. WEB-сайт (Электрон. ресурс) / Способ доступа: URL:http://visual.2000.ru/develop/vb/source/mod2000.htm

6. Report on the Programming Language Haskell 98. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.haskell.org/definition/

7. Visual Basic 6.0: Пер.с англ. –СПб.:БХВ-Петербург, 2001. – 992 с.

8. XML для профессионалов / Пер. с англ. / Мартин Дидье, Бирбек Марк, Кей Майкл и др. –М.: Изд-во «Лори», 2001. – 864 с.

9. Абрамов С.А., Антипов И.Н. Основы программирования на Алголе. – 2-е изд., перераб. –М.: Наука, 1982. –172 с.

10. Алгоритмический язык Алгол 60. Модифицированное сообщение. –М.: Мир, 1982. – 72 с.

11. Алексеев М.А., Коротенко Г.М., Коротенко Л.М. Обучение программированию в контексте особенностей языков программирования // Науковий вісник НГА України, №2, 2003. – С. 6-8.

12. Бабенко Л.П., Лавріщева К.М. Основи програмної інженерії: Навч. посібник. – К.: Т-во "Знання", КОО, 2001. – 296 с.

13. Бен-Ари М.. Языки программирования. Практический сравнительный анализ. М.: Мир, 2000. –366 с.

14. Березин Б.И., Березин С.Б. Начальный курс С и С++. - М.:ДИАЛОГ-МИФИ, 1996.- 288 с.

15. Бизли Дэвид М. Язык программирования Python. Справочник. Пер. с англ. –К.: Изд-во «ДиаСофт», 2000. – 336 с.

16. Богатырев Руслан. Рождение Паскаля. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://pascal.sources.ru/articles/058.htm

17. Браух В. Программирование на Фортране 77 для инженеров: Пер. с нем. –М.: Мир, 1987. – 200 с.

18. Брусенцов Н.П. Миникомпьютеры. –М.: Наука, 1979. (глава 9: "Многорегистровые миникомпьютеры DEC PDP-11")

19. Бусыгин Б.С., Коротенко Г.М., Коротенко Л.М. Современные тенденции применения языков программирования в геоинформационных системах // Геоінформатика, – №3. Київ. –2003. –С. 24-30.

20. Буч Гради. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. – М.: БИНОМ, СПб.: Невский диалект, 1998. – 560 с.

21. Буч Гради. Объектно-ориентированный анализ и проектирование с примерами приложений на С++ / Пер. с англ. – Спб.–М.: "Невский Диалект" – "Издательство Бином", 1999. – 560 с.

22. Введение в XML / Курт Кэгл, Дэйв Гиббонс, Дэвид Хантер, Никола Озу, Джон Пинок, Пол Спенсер. –М.: Издательство «Лори», 2000. – 638 с.

23. Вебер Д. Технология Java™ в подлиннике / Пер. с англ. –СПб.: БХВ-Петербург, 2001. – 1104 с.

24. Вигдорчик Г. В.и др. Основы программирования на ассемблере для СМ ЭВМ. –М.: Финансы и статистика, 1987. – 240 с.

25. Вирт Н. Язык программирования Паскаль. –Л.: Изд-во ЛГУ, 1974. – 49 с.

26. Гончаров А. Самоучитель HTML. –СПб.: Питер, 2000. – 240 с.

27. Дейкало Г. Ф., Новиков Б. А., Рухлин А. П., Терехов А. Н.. Новые средства программирования для ЕС ЭВМ: Транслятор с языка Алгол 68 и диалоговая система JEC. –М.: Финансы и статистика, 1984. – 207 с. (часть I: "Введение в Алгол 68").

28. Дейтел Харви, Дейтел Пол. Как программировать на C++: – 3-е изд.: Пер. с англ. – М.: ЗАО "Изд.-во БИНОМ", 2001. – 1152 с.

29. Джехани Н. Язык Ада: Пер. с англ. –М.: Мир, 1988. – 552 с.

30. Джоунз Г. Программирование на языке Оккам. –М.: Мир, 1989. – 208 с.

31. Додж М., Кината К., Стинсон К. Эффективная работа с Excel 7.0 для Windows 95 / Перев. с англ. –СПб.:Питер, 1997. – 1040 с.

32. Жарков В.А. Visual C# .NET в науке и технике. – М.: Жарков Пресс, 2002. – 638 с.

33. Жарков В.А. Самоучитель Жаркова по Visual Basic .NET. – М.: Жарков Пресс, 2002. – 336 с.

34. Жарков В.А. Самоучитель Жаркова по Visual Studio .NET: Visual Basic .NET, Visual C# .NET, Visual C++ .NET, Visual J# .NET. – М.: Жарков Пресс, 2002. – 319 с.

35. Жарков В.А. Самоучитель Жаркова по анимации и мультипликации в Visual Basic .NET 2003. – М.: Жарков Пресс, 2003. – 416 с.

36. Жарков В.А. Самоучитель Жаркова по анимации и мультипликации в Visual C# .NET 2003. – М.: Жарков Пресс, 2003. – 432 с.

37. Жарков В.А. Самоучитель Жаркова по анимации и мультипликации в Visual C++ .NET 2003. – М.: Жарков Пресс, 2003. – 448 с.

38. Жарков В.А. Самоучитель Жаркова по интеграции Visual Basic .NET 2003 с другими платформами. – М.: Жарков Пресс, 2003. – 592 с.

39. Использование HTML4: Пер. с англ. -2-е изд. / Луиза Паттерсон, Сью Шарльворс, Джоди Корнелиус и др. : Уч. пос. –М.: Издательский дом «Вильямс», 2000. – 400 с.

40. Йенсен К., Вирт Н.. Паскаль. Руководство для пользователя. –М.: Финансы и статистика, 1989. –255 с.

41. Как программировать на XML / Дейтел Х.М., Дейтел П.Дж., Нието Т.Р., Лин Т.М., Содху П. Пер. с англ. –М.: ЗАО «Издательство БИНОМ», 2001. – 944 с.

42. Катцан Г. Язык Фортран 77. –М.: Мир, 1982. – 208 с.

43. Кергаль М. Методы программирования на Бейсике (с упражнениями). – М.: Мир, 1991. – 288 с.

44. Керниган Б., Ритчи Д. Язык программирования Си: Пер. с англ. / Под ред. и с предисл. В.С. Штаркмана.- 2-е изд., перераб. и доп.-М.: Финансы и статистика, 1992. – 272 с.

45. Керниган Б.В., Пайк Р. UNIX - универсальная среда программирования: Пер. с англ.; Предисл. М.И. Белякова.-М.: Финансы и статистика, 1992. – 304 с.

46. Классификация языков программирования по типам задач. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://schools.keldysh.ru/sch444/MUSEUM/LANR/index.htm

47. Клоксин У., Меллиш К.. Программирование на языке Пролог. –М.: Мир, 1987. – 336 с.

48. Колесов Андрей, Павлова Ольга. MS Office 2000 для разработчиков Взгляд первый //"КомпьютерПресс", № 12, 1999. –С.169-175.

49. Конопка Р. Создание оригинальных компонент в среде Delphi. – СПб.: ООО «ДиаСофтЮП», 2002. – 512 с.

50. Культин Н. Delphi 3. Программирование на Object Pascal. –СПб.: BHV-Санкт-Петербург, 1998. – 304 с.

51. Кэнту М. Delphi 6 для профессионалов. СПб.: Питер, 2002. – 1088 с.

52 Лавров С. С., Силагадзе Г. С.. Автоматическая обработка данных. Язык ЛИСП и его реализация. –М.: Наука, 1978. – 239 с.

53. Лесса Андре. Python. Руководство разработчика. Пер. с англ. –СПБ.: ООО «ДиаСофтЮП», 2001. – 688 с.

54. Либерти Джесс. Освой самостоятельно C++ за 21 день. (третье издание) / Пер. с англ. : Уч. пос. –М.: Издательский дом "Вильямс", 2001. – 816 с.

55. Липаев В.В. Надёжность программных средств. –М.: Изд-во “СИНТЕГ”, 1998. – 354 с.

56. Лутц М. Программирование на Python. –Пер. с англ. –СПб.: Символ – Плюс, 2002. – 1136 с.

57. Мюррей У. Паппас К. Visual C++. Руководство для профессионалов. – СПб.: BHV-Санкт-Петербург, 1996. – 912 с.

58. Ноутон П., Шилдт Г. Java™2 : Пер. с англ. –СПб.: БХВ–Петербург, 2001. – 1072 с.

59. Палмер Скотт. VBScript и ActiveX: библиотека программиста. – СПб: ЗАО «Издательство «Питер», 1999. – 368 с.

60. Пратт Т. Языки программирования: разработка и реализация. –М.: Мир, 1979. – 574с.

61. Программирование на Фортране 77: Пер. с англ. / Дж. Ашкрофт, Р. Элдридж, Р. Полсон, Г. Уилсон. –М.: Радио и связь, 1990. – 272 с.

62. Ричерд Вайнер, Льюис Пинсон. C++ изнутри / Пер. с англ. – Киев: «Диасофт», 1993. – 304 с.

63. Сайлер Брайан, Споттс Джефф. Использование Visual Basic .NET. Специальное издание.: Пер. с англ. –М.; Издательский дом «Вильямс», 2002. – 752 с.

64. Сафонов В.О. Языки и методы программирования в системе «Эльбрус». / Под ред.С.С.Лаврова. –М.: Наука. Гл.ред.физ.-мат.лит., 1989. – 392 с.

65. Скотт Р., Сондак Н. ПЛ/1 для программистов. Пер. с англ. Э.А. Трахтенгерца. –М.: Статистика, 1977. – 223 с.

66. Соммервилл Иан. Инженерия программного обеспечения, 6-е издание.: Пер. с англ. –М.: Издательский дом «Вильямс», 2002. – 624 с.

67. Стобо Д.Ж.. Язык программирования Пролог. –М.: Радио и связь, 1993. – 368 с.

68. Страуструп Б.. Язык программирования C++. (3-е изд.), –СПб, М.: Невский диалект, Бином, 1991. – 991 с.

69. Страуструп Б. Язык программирования C++, спец. изд. / Пер с англ. – М.; СПб.: «Издательство БИНОМ»-«Невский диалект», 2002. – 1099 с.

70. Седжвик Р. Фундаментальные алгоритмы на C++. Анализ/Структуры данных. –СПБ.: ООО «ДиаСофтЮП», 2001. – 688 с.

71. Уайттеккер Джеймс, Воас Джеффри. 50 лет программирования: основные принципы качества / Открытые системы, #03/2003. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.citforum.ru/programming/theory/50letprogr.shtml

72. Филд А., Харрисон П.. Функциональное программирование. –М.: Мир, 1993. – 447 с.

73. Форт С., Хоуи Т. Программирование в среде Access 2000. Энциклопедия пользователя. –СПБ.: ООО «ДиаСофтЮП», 2001. – 544 с.

74. Фридман А., Кландер Л., Михаэлис М., Шильдт Х. C/C++. Архив программ. –М.: ЗАО «Издательство БИНОМ», 2001. – 604 с.

75. Фролов А.В., Фролов Г.В. Программирование для Windows NT: ч. 2. – М.: «ДИАЛОГ-МИФИ», 1997. – 271 с. (Библиотека системного программиста т. 27).

76. Хефлин Д., Ней Т. Разработка Web-скриптов. Библиотека программиста. – СПб.: Питер, 2001. – 496 с.

77. Хильер Скот. Создание приложений COM+ в среде Visual Basic. Руководство разработчика. : Пер. с англ. : –М.: Издательский дом «Вильямс», 2001. – 416 с.

78. Хоар Ч.. Взаимодействующие последовательные процессы. –М.: Мир, 1989. –264 с.

79. Хольцшлаг Молли Э. Использование HTML4, 6-е изд. Специальное издание: Пер. с англ. : Уч. пособие. –М.: Издательский дом "Вильямс", 2001. – 1008 с.

80. Цейтин Г.С.. От логицизма к процедурализму. На автобиографическом материале // В сб.: Алгоритмы в современной математике и ее приложениях. ВЦ СОАН, Новосибирск, 1982, ч.2. – С. 181–193.

81. Языки программирования Ада, Си, Паскаль. Сравнение и оценка / Под ред. Фьюэра А.Р., Джехани Н. Пер. с англ. Под ред. В.В. Леонаса. –М.:Радио и связь, 1989. – 368 с.

К главе 6. Изменения в методологии создания программ

1. Active Server Pages 3/0 на примерах / Чейз Николас : Пер. с англ. : Уч. пос. –М.: Издательский дом «Вильямс», 2001. – 352 с.

2. ASP XML для профессионалов / Бартси Марк, Блэр Ричард, Болоньи Лука и др. Пер. с англ. –М.: Издательство «Лори», 2002. – 704 с.

3. Chartier Robert. Application Architecture: An N-Tier Approach - Part 1. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.15seconds.com/ Issue/011023.htm?voteresult=5

4. Szyperski Clemens, "Component Software: Beyond Object-Oriented Programming," 2nd edition, Addison Wesley Professional, 2002, – 624 pp.

5. Understanding GXA. Microsoft Global XML Architecture (GXA). WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http:/microsoft.com/

6. Vawter Chad, Roman Ed. J2EE vs. Microsoft.NET. A comparison of building XML-based web services. June 2001. Prepared for Sun Microsystems, Inc. WEB-

сайт (Электрон. ресурс) / Способ доступа: URL: http://www.theserverside.com/resources/article.jsp?l=J2EE-vs-DOTNET

7. Web Services Development Tools – J2EE & .NET. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.wcca.ifas.ufl.edu/web_services_development_tools.htm

8. XML для профессионалов / Пер. с англ. / Мартин Дидье, Бирбек Марк, Кей Майкл и др. –М.: Изд-во «Лори», 2001. – 864 с.

9. Аншина Марина. Сервер приложений – не пуп Земли? WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://osp.admin.tomsk.ru/os/2000/05-06/064.htm

10. Бадд Тиммоти. Объектно-ориентированное программирование в действии. –СПб.: Питер, 1997. – 464 с.

11. Байцер Б. Архитектура вычислительных комплексов / Пер. с англ. – М.: Мир, 1974, Т. 1. – 498 с.

12. Бек. Е. Экстремальное программирование. – Спб.: Питер, 2002. – 224 с.

13. Берлинер Э.М., Глазырин Б.Э, Глазырина И.Б. Microsoft WINDOWS 95. Русская версия. Дополнения 96-го года. –М.: ABF, 1997. – 522 с.

14. Богумирский Б. Энциклопедия Windows 98 (второе издание). –СПб.: Питер, 2001. – 896 с.

15. Бойко Сергей (Banzai). Пашарпанный Си. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.mycomp.com.ua/article.php?id=5282

16. Боэм Б.У. Инженерное проектирование программного обеспечения: Пер. с англ./Под. Ред. А.А. Красилова. –М.: Радио и связь. 1985. – 512 с.

17. Брандт Д. Architectures. Экзамен – экстерном. (экзамен 70-100). СПБ.: Питер, 2001. – 432 с.

18. Брауде Эрик Дж. Технология разработки программного обеспечения. – СПб.: Петербург, 2004. – 655 с.

19. Буч Гради. Объектно-ориентированное проектирование с примерами применения. – Киев: НПФ Диалектика, 1992. – 519 с.

20. Введение в XML / Курт Кэгл, Дэйв Гиббонс, Дэвид Хантер, Никола Озу, Джон Пинок, Пол Спенсер. –М.: Издательство «Лори», 2000. – 638 с.

21. Вильям Дж. Орвис. Visual Basic For Application на примерах: Пер. С англ. –М.: БИНОМ, 1997. – 512 с.

22. Волоха Александр. Jini[tm].NET. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http:// www.ixbt.com/editorial/ jini-vs-net.shtml

23. Гуннерсон Э. Введение в C#. Библиотека программиста. –СПб: "Питер", 2001. – 304 с.

24. Дантеманн Джефф, Мишел Джим, Тэйлор Дон. Программирование в среде Delphi: Пер. с англ. –К.: НИПФ «ДиаСофт Лтд.», 1995. – 608 с.

25. Дейтел Х.М., Дейтел П.Дж., Сантри С.И. Технологии программирования на Java 2. Кн. 3: Корпоративные системы, сервлеты, JSP, Web-сервисы: Пер. с англ. –М: Бином, 2003. – 672 с.

26. Джамса К., Коуп К. Программирование для Internet в среде Windows / Перевод с англ. –СПб.: Питер, 1996. – 688 с..

27. Джеффри Рихтер. Windows для професионалов (программирование в WIN32 API для Windows NT 3.5 и Windows 95)/ Пер. с англ. –М.: Издательский отдел «Русская редакция» ТОО «Channel Trading Ltd», 1995. – 720 с.

28. Дональд Ф. Шафер, Линда И. Шафер, Роберт Т. Фатрелл Управление программными проектами: достижение оптимального качества при минимуме затрат. –М.: Вильямс 2003, – 1136 с.

29. Дэйтел Х.М., Дэйтел П.Дж., Нието Т.Р. Как программировать для Internet и WWW. Пер. с англ. – М.: ЗАО «Издательство БИНОМ», 2002. – 1184 с.

30. Как программировать на XML /Дейтел Х.М., Дейтел П.Дж., Нието Т.Р., Лин Т.М., Содху П. Пер. с англ. –М.: ЗАО «Издательство БИНОМ», 2001. – 944 с.

31. Колесов Андрей. Архитектурные решения Microsoft .NET Framework. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.pcweek.ru/?ID=184308

32. Колесов Андрей. Visial Basic.NET — страсти накаляются // PC Week/RE, № 18. 2001. –С. 36. (WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://visual.2000.ru/kolesov/pcweek/2001/10424vb7.htm

33. Колесов Андрей. Преобразование объектов COM в Web-сервисы. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://bytemag.ru/Article.asp?ID=1239

34. Кэнту М. Delphi 6 для профессионалов. СПб.: Питер, 2002. – 1088 с.

35. Либерти Джесс, Крейли Майк. Создание документов XML для Web. Пер.с англ.: Уч. Пос. – М.: Издательский дом «Вильямс», 2000. – 256 с.

36. Ливингстон Б., Штрауб Д. Секреты Windows 95. –К.: ”КОМИЗДАТ”,”Диалектика”, 1996. – 560 с.

37. Майкл Янг. XML.Шаг за шагом: Практ. Пособ. / Пер. с англ. –М.: Издательство ЭКОМ, 2000. – 384 с.

38. МакКрэри Кен. Динамическая графика в Java сервлетах. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.citforum.ru/internet/ javascript/javaservlet.shtml

39. Марти Холл, Лэрри Браун. Программирование для WEB. Библиотека профессионала. : Пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 1264 с. (программирование на Java).

40. Мэтчо Дж. И др. Delphi 2 для профессионалов. Пер. с англ. –СПб.: BHV-Санкт-Петербург, 1997. – 784 с.

41. Нидерст Дж. Web–мастеринг для профессионалов. – СПб.: Питер, 2001. – 576 с.

42. Ньюкомер Э. Веб-сервисы. XML, WSDL, SOAP и UDDI: Пер. с англ. – СПб.: Питер, 2003. – 256 с.

43. Одинцов Игорь. Профессиональное программирование. Системный подход. – СПб.: БХВ-Петербург, 2002. – 512 с.

44. Петзолд Ч. Программирование для Windows 95: в двух томах./ Пер. с англ. –СПб.: BHV-Санкт-Петербург,1997. (Т.1. – 752 с.), (Т.2. – 368 с.)

45. Разработка приложений в среде Visual Basic 6.0. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://old.sgu.ru/users/matmodel/ tutorial/tutor2/tutor2.htm

46. Рассохин Д.Н. World Wide Web - Всемирная Информационная Паутина в сети Internet. –М.: "МГУ", 1997. – 208 с.

47. Сервисы Интернет: практическое рассмотрение (Часть 1) WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://planetfree.hop.ru/webart/internetservices.htm

48. Смирнов А.Д. Архитектура вычислительных систем: Учеб. пособие для вузов. –М.: Наука. Гл. ред. физ.–мат. лит., 1990. – 320 с.

49. Соммервилл Иан. Инженерия программного обеспечения, 6-е издание.: Пер. с англ. –М.: Издательский дом «Вильямс», 2002. – 624 с.

50. Спенсер Пол. XML. Проектирование и реализация. –М.: Изд-во «Лори», 2001. – 509 с.

51. Сурков К.А., Сурков Д.А., Вальвачёв А.Н. Программирование в среде DELPHI 2.0 – Мн.: ООО «Попурри», 1997. – 640 с.

52. Темерев Александр. Net, и точка! WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.iworld.ru/magazine/index.phtml?do=show_article&p=99781627

53. Туротт Пол, Брент Гарри, Ричард Багдазиан, Тэндон Стив. Супербиблия Delphi 3. –К.: Издательство «ДиаСофт», 1997. – 848 с.

54. Федоров Алексей. Знакомство с Microsoft .NET Framework. Часть 1. Common Language Runtime. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.interface.ru/microsoft/Net/7.gif

55. Федоров Алексей. Web нового поколения — Web-сервисы. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://www.compress.ru/Temp/1081/index.htm

56. Харрис Мэтью. Освой самостоятельно программирование для Microsoft Excel 2000 за 21 день: Пер. с англ.: Уч. пособие. – М.: Изд. Дом «Вильямс», 2000. – 880 с.

57. Чакраборти А., Кранти Ю., Сандху Р. Дж. Microsoft® .NET Framework: разработка профессиональных проектов: Пер. с англ. – СПб.: БХВ-Петербург, 2003. – 896 с.

58. Шапошников И.В. Web-сервисы Microsoft .NET. –СПб.: BHV-Санкт-Петербург, 2002. – 453 с.

59. Шеперд Деван. Освой самостоятельно XML за 21 день, 2-е издание. : Пер. с англ. –М.: Издательский дом «Вильямс», 2002. – 432 с.

К главе 7. Язык UML и его использование

1. Object-Oriented Programming. WEB-сайт (Электрон. ресурс) / Способ доступа: URL: http://people.cs.vt/edu/~kafura/cs2704/basic.concepts.html

2. Бадд Тиммоти. Объектно-ориентированное программирование в действии. –СПб.: Питер, 1997. – 464 с.

3. Буч Гради. Объектно-ориентированное проектирование с примерами применения. – Киев: НПФ Диалектика, 1992. – 519 с.

4. Буч Г., Рамбо Д., Джекобсон А. Язык UML: Руководство пользователя : Пер. с англ. –М.: ДМК, 2000. –432с.

5. Коналлен Джим. Разработка Web-приложений с использованием UML : Пер. с англ. –М.: Издательский дом «Вильямс», 2001. – 288 с.

6. Коротенко Г.М., Гаркуша И.Н. Технологические аспекты использования объектно-ориентированного подхода при создании прикладных геоинформационных систем // Сб. научн. трудов Национальной горной академии Украины, № 9,т.1,2000. –С. 98-103.

7. Ларман Крэг. Применене UML и шаблонов проектирования. Введение в объектно-ориентированный анализ и проектирование : Уч. пос. –М.: Издательский дом «Вильямс», 2001. – 496 с.

8. Леоненков А.В. Самоучитель UML. –СПб.: «BHV-Петербург», 2001. – 304 с.

9. Рамбо Джеймс, Якобсон Айвар, Буч Грэди. UML: Специальный справочник. СПб.: Питер, 2002. – 656 с.

10. Трофимов С.А. Case-технологии. Практическая работа с Rational Rose. –М.: Бином, 2001. – 272 с.

11. Шмуллер Джозеф. Освой самостоятельно UML за 24 часа, 2-е издание. : пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 352 с.

К главе 8. Введение в Турбо Паскаль

1. Turbo Pascal 6.0 Programmer’s Guide. Borland International. Inc. –1990. – 372 p.

2. Turbo Pascal 6.0 User’s Guide. Borland International. Inc. –1990. – 257 p.

3. Turbo Pascal 6.0 Turbo Vision Guide. Borland International. Inc. –1990. – 409p.

4. Алкок Д. Язык Паскаль в иллюстрациях. –М.: Мир, 1991. – 192 с.

5. Белецкий Я. Турбо Паскаль с графикой для персональных компьютеров/Пер. с польск. –М.: Машиностроение, 1991. – 320 с.

6. Боон К. ПАСКАЛЬ для всех/ Пер. с гол. –М.: Энергоатомиздат, 1988. – 190 с.

7. Бородич Ю.С., Вальвачев А.Н., Кузьмич А.И. Паскаль для персональных компьютеров. –Мн.: Выш.шк., 1991. – 364 с.

8. Вальвачев А.Н., Крисевич В.С. Программирование на языке Паскаль для персональных ЭВМ ЕС. –Мн.: Выш.шк.,1989. – 223 с.

9. Введение в язык Паскаль/ Абрамов В.Г., Трифонов Н.П., Трифонова Г.Н./ Учеб. пособие. –М.: Наука.Гл.ред.физ.-мат.лит., 1988. – 320 с.

10. Вирт Н. Алгоритмы+структуры данных=программы. –М.: Мир, 1985. – 406 с.

11. Вирт Н. Алгоритмы и структуры данных/ Пер. с англ. –М.: Мир, 1989. – 360 с.

12. Далека В.Д., Деревянко А.С., Кравец О.Г., Тимановская Л.Е. Модели и структуры данных. Учебное пособие. – Харьков: ХГПУ, 2000. – 241 с.

13. Дель В.Д. Основы алгоритмизации и программирования: Учеб. пособие. –Винница: ВПИ, 1988. – 168 с.

14. Джонс Ж., Харроу К. Решение задач в системе Турбо Паскаль/ Пер с англ.. –М.: Финансы и статистика, 1991. – 720 с.

15. Дэвис Дж. Статистика и анализ геологических данных. –М.: Мир, 1977. – 572 с.

16. Дэвис Дж. О.. Статистический анализ данных в геологии. –М.: Недра, 1999. В 2-х кн. Кн.1. – 319 с.

17. Збірник завдань з програмування для студентів ii курсу спеціальностей: 7.050107, 7.050201, 7.050102, 7.080401, 7.091503 денної і заочної форм навчання. / Укл.: Огданский Н.Ф., Нейковская Л.С., Виноградов К.Г., Дніпропетровськ: УГХТУ, 2000. – 72 с.

18. Зуев Е.А. Язык программирования Turbo Pascal 6.0. –М.: Унитех, 1992. – 292 с.

19. Йенсен К., Вирт Н. Паскаль: руководство для пользователя. –М.: Финансы и статистика, 1989. – 255 с.

20. Кнут Д. Искусство программирования для ЭВМ. Т.1. Основные алгоритмы/ Пер. с англ. –М.: Мир, 1976. – 735 с.

21. Кнут Д. Искусство программирования для ЭВМ. Т.2. Получисленные алгоритмы/ Пер. с англ. –М.: Мир, 1977. – 724 с.

22. Кнут Д. Искусство программирования для ЭВМ. Т.3. Алгоритмы сортировки и поиска/ Пер. с англ. –М.: Мир, 1978. – 844 с.

23. Мак-Кракен Д., Дорн У. Численные методы и программирование на ФОТРАНЕ. –М.: Мир, 1988. – 584 с.

24. Марченко А.И., Марченко Л.А. Программирование в среде Turbo Pascal 7.0. -М.: Бином Универсал, к. ЮНИОР, 1997. – 496 с.

25. Мейер Б., Бодуэн К. Методы программирования: В 2-х томах. –М.: Мир, 1982. Т.1. – 356 с. (Т.2. – 368 с.).

26. Методичні вказівки і завдання до виконання лабораторних робіт (Турбо Паскаль) для студентів спеціальностей 7.070908 "Геоінформаційні системи і технології", 7.080404 "Інтеллектуальні системи прийняття рішень", 7.080407 "Комп'ютерний еколого-економічний моніторинг" / Упорядники: Г.М. Коротенко, Л.М. Коротенко, В.М. Куваєв, В.В. Ішков, М.В. Гусенко. – Дніпропетровськ: Національний гірничий університет, 2003. – 124 с.

27. Мизрохи С.В. TURBO PASCAL и объектно-ориентированное программирование. –М.: Финансы и статистика, 1992. – 192 с.

28. Огенстайн Н., Тененбаум А. Структуры данных для ЭВМ. –М.: Мир, 1985. – 568 с.

29. Паскаль: Учеб. пособие для сред.спец.учеб.заведений/ В.С. Новичков, Н.И. Парфилова, А.Н. Пылькин. –М.: Высш.шк., 1990. – 223 с.

30. Поляков Д.Б., Круглов И.Ю. Программирование в среде Турбо Паскаль. Версия 5.5. –М.: Изд-во МАИ, 1992. – 570 с.

31. Прайс Д. Программирование на языке Паскаль: Практическое руководство/ Пер. с англ. _М.: Мир, 1987. – 232 с.

32. Программирование в среде Turbo Pascal 6.0: Справ. Пособие/Ю.С. Климов, А.И. Касаткин, С.М. Мороз. –Мн.: Выш.шк., 1992. – 158 с.

33. Программирование в среде Turbo Pascal 7.0. / Марченко А.И., Марченко Л.А. / Под ред. В.П. Тарасенко: – 5-е изд. перераб. и доп. – К.: ВЕК+, 1999. – 464 с.

34. Светозарова Г.И. Алгоритмизация и основы программирования. –М.: Высш. шк., 1987. – 127 с.

35. Техніка обчислень і алгоритмізація/ І.Ф. Следзиньскій та ін. –Київ: Вища шк., 1990. – 199 с.

36. Трамбле Ж., Соренсон П. Введение в структуры данных. М.: Машиностроение, 1982. – 784 с.

37. Турбо Паскаль 7.0.–К.:Торгово-издательское бюро BHV,1995. – 448 с.

38. Турбо Паскаль 7.0. – К.: Издательская группа BHV, 2002. – 496 с.

39. Фаронов В.В. Турбо Паскаль 7.0. Начальный курс.Учебное пособие. – М.: «Нолидж», 1997. – 616 с.

40. Фаронов В.В. Турбо Паскаль 7.0. Практика программирования. Учебное пособие. –М.: «Нолидж», 1997. – 482 с.

41. Фигурнов В.Э. IBM PC для пользователя, 2-е изд., перераб. и доп. –М.: Финансы и статистика, 1991. – 288 с.

42. Эрбс Х.-Э., Штольц О. Введение в программирование на языке Паскаль. –М.: Мир, 1989. – 299 с.

Общеметодическая литература

1. Алгоритми: представлення, конструювання та відлагодження. :Навч. посібник/ С.Л. Нікулін, І.В. Гриценко, С.В. Яковлєв. – Дніпропетровськ: Національна гірнича академія України, 2002. – 83 с.

2. Борланд Р. Эффективная работа с Word 7.0 для Windows 95 / Пер. с англ. СПб: Питер, 1997. – 1104 с.

3. Васкевич Д. Стратегии клиент/сервер. Руководство по выживанию для специалистов по реорганизации бизнеса. – К.: "Диалектика", 1996. – 384 с.

4. Вейскас Д. Эффективная работа с Microsoft Access 2/ Перев. с англ. – СПб.: Питер, 1995. – 864 с.

5. Додж М., Кината К., Стинсон К. Эффективная работа с Excel 7.0 для Windows 95 / Перев. с англ. –СПб.:Питер, 1996. – 1040 с.

6. Калянов Г.И. CASE: структурный системный анализ (автоматизация и применение). –М.: Лори, 1996. – 242 с.

7. Калянов Г.Н. CASE–технологии. Консалтинг в автоматизации бизнес-процессов. 2-е изд. перераб. и доп. –М.: Горячая линия–Телком, 2000. – 320 с.

8. Перкинс Ч., Стиб М., при участии Джеймса Чаллиса. NT Workstation. Учебное пособие для специалистов MSCE. - М.: "Лори", 1997. – 436 с.

9. Рекомендации по преподаванию информатики в университетах: Пер. с англ. –СПб.: 2002. – 372 с.

Беседы учёных мужей часто оканчиваются разногласиями по поводу смысла слов.

Фрэнсис Бэкон

Когда вам приходится встретить слово или символ, которые вы не понимаете, первое, что надо сделать – это взять словарь и быстро просмотреть его определение, чтобы выбрать именно то, которое подходит к контексту, где употреблено это слово. Прочтите найденное определение и составляйте предложения с этим словом, пока у вас не сложится четкое представление о его значении. При этом может потребоваться десять или более предложений.

Рон Хаббард.

ГЛОССАРИЙ

.NET – (читается ″дот нет″)

Концепция, архитектура и платформа, разрабатываемая Microsoft для создания и использования Web-сервисов с применением компонентных технологий. Инфраструктура .NET, опирающаяся на стандарт XML, обеспечивает более легкое программирование, лучшую производительность и масштабирование, меньшую зависимость от особенностей источников данных и большую способность взаимодействовать с другими платформами в сетях любого уровня, в том числе и беспроводно. Архитектура .NET основана на следующих базовых концепциях:

–независимой от языка среде исполнения (CLR, Common Language Runtime);

–библиотеке классов .NET (.NET Class Library);

–языке-посреднике Microsoft Intermediate Language (MSIL);

–группе языков программирования, поддерживающих .NET (C#, Visul Basic .NET, Visual C++ .NET и др.).

32-разрядная операционная система

Операционная система, способная непосредственно обрабатывать 32-разрядные коды чисел и оперировать 32-разрядными адресами. Естественным образом обеспечивает выполнение 16-и и 32-разрядных приложений.

32-разрядное приложение

Приложение, способное непосредственно обрабатывать 32-разрядные коды чисел и оперировать 32-разрядными адресами. И то, и другое, в конечном счёте, даёт выигрыш в быстродействии. 32-разрядные приложения могут исполняться только на микропроцессорах Intel 80386 и выше. Для выполнения таких приложений необходима 32-разрядная операционная система.

3D-графика.

До недавнего времени 3D-графика в реальном времени была для компьютера фантастически сложной задачей, т.к. при её реализации необходимо выполнять миллионы геометрических операций в секунду, успевая при этом выводить их результат на экран. Для создания одного кадра компьютеру необходимо построить изображение, разбить его на элементы (в современной 3D индустрии это либо полигоны, либо воксели), которых может быть до 30000 на кадр, просчитать отражения и тени, скорректировать перспективу, и размыть изображения по краям, дабы избежать "лестниц" на переходах между пикселами.

Все эти операции надо успевать выполнять за 1/20--1/30 секунды, чтобы получить достаточное число кадров в секунду. При этом необходимо учесть, что компьютер не только строит изображения, он должен реагировать на действия игрока или пользователя. Неигровые 3D-программы появились впервые на рынке в 1996 г. Они позволяли создавать 3D изображения с расчетом теней и наложением текстур, что и называется рендерингом.

3GL (Generation Language 3)

Языки 3-го поколения, то есть языки, не встроенные в интегрированные интерактивные среды разработки прогаммного обеспечения такие, как C, C++, COBOL, Ada, Pascal.

4GL (Generation Language 4)

Языки 4-го поколения, то есть языки, встроенные в интегрированные интерактивные среды разработки прогаммного обеспечения или RAD-средства. К ним относятся Microsoft VisualBasic, PowerBuilder, Inprise Delphi Object Pascal, Oracle Developer PL/SQL и др.

64-разрядные приложения

Приложения, подобные 32-разрядным, но оперирующие с объектами, имеющими в два раза большую разрядность. Это повышает производительность компютерных систем и увеличивает скорость вычислений.

- A -

Account (бюджет)

Объём ресурсов вычислительной системы (дисковое пространство, дисковые устройства: флоппи, CD-ROM и др., принтеры, сканеры и т.д.), который данный пользователь или группа пользователей может использовать в течение определённого времени. Обычно выделяется для пользователей системным администратором и защищается индивидуальным именем и паролём.

ACM (Association for Computing Machinery)

Ассоциация по Вычислительной Технике. Международная организация со штаб-квартирой в США, занимающаяся прогнозированием и исследованием процессов развития информационных технологий.

ACPI (Advanced Configuration and Power Interface)

Спецификация управления режимами пониженного энергопотребления, пришедшая на смену APM. Впервые была использована компанией Intel в чипсете 430TX. Является частью спецификации PC97.

ActiveX

Набор технологий, позволяющих программным компонентам, написанным на разных языках программирования, совместно работать в рамках сетевого окружения. Основными технологическими составляющими ActiveX являются компонентная объектная модель (Component Object Model, COM) и распределённая компонентная объектная модель (Distributed Component Object Model, DCOM).

ActiveX объект.

❶ Вариант (разновидность) технологии Microsoft OLE, специально предназначенной для использования в Internet, где невозможно добиться высокой скорости передачи данных между узлами. Обеспечивает взаимодействие программных компонентов (написанных на разных языках программирования) в сетевой среде. В основе ActiveX лежит спецификация COM.

❷ Небольшая программа, соответствующая стандарту Microsoft, встроенная в Web-страницу. Эта программа предназначена для расширения возможностей браузера.

Ada

Универсальный язык программирования высокого уровня, созданный, в первую очередь, для разработи программного обеспечения встроенных и управляющих

компьютерных систем. Язык Ада основан на идеях структурного программирования и обеспечивает поддержку разработки сложных многомодульных программ, высокую степень платформо-независимости и переносимости. Назван в честь Августы Ады Лавлейс, первого в истории программиста.

ADO (ActiveX Data Objects)

Технология Microsoft, которая является надстройкой OLE Automation над OLE DB, открывающей доступ к объектам OLE DB через любой язык программирования или инструментальное средство, поддерживающее COM. Вытекает из поддержки ADO дуальных (dual) интерфейсов

AGP (Accelerated Graphic Port, скоростной графический порт)

❶ Стандарт, предложенный фирмой Intel в виде архитектуры графических ускорителей видеосистем. Появился в 1997 году как замена шины PCI для видеокарт. Главным отличием AGP от PCI была его однопортовость, то есть возможность подключить одну видеокарту и больше ничего. Именно поэтому AGP часто называется портом, а не шиной. Основными целями, преследовавшимися разработчиками AGP, были:

1) использование части системной памяти компьютера для хранения текстур и больших трехмерных сцен, не вмещающихся в ограниченную память видеокарты;

2) прямая передача информации между видеокартой и оперативной памятью, минуя процессор;

3) увеличение скорости передачи данных между видеокартой и системной шиной. Частота работы порта AGP составляла в начале его появления 66,6 МГц.

❷ 32-разрядная шина с частотой передачи данных 66 МГц, используемая для подключения графического адаптера и позволяющая ускорить обмен данными между графическим адаптером и основной памятью. Режим 2x подразумевает использование каждого такта работы шины дважды, для передачи вдвое большего количества информации.

❸ Технология, позволяющая графическому процессору получить доступ к оперативной памяти, минуя основной процессор. У видеоподсистем с AGP в качестве буфера кадров по-прежнему используется высокоскоростная видеопамять.

ANSI-кодировка (ANSI – American National Standards Institute, Американский национальный институт стандартов).

Кодировка символов 8-разрядными двоичными числами, используемая в ОС Windows. Обеспечивает представление 256-и символов. Отличается от ASCII-кодировки, поэтому русскоязычные текстовые файлы, подготовленные в среде MS-DOS, без предварительной конвертации оказываются нечитаемыми в среде Windows и наоборот.

API (Application Programming Interface или Win API – интерфейс прикладного программирования)

❶ Набор функций, предоставляемых операционной системой Windows каждой программе. Все эти функции находятся в стандартных динамически компонуемых библиотеках DLL, таких как kernel32.dll, user32.dll, gdi32.dll. Файлы находятся в директории Window\System. С другой стороны API есть интерфейс доступ к системным ресурсам операционной системы Windows. Совокупность таких функций называется прикладным программным интерфейсом или API. Для взаимодействия с Windows приложение запрашивает функции API, с помощью которых реализуются все необходимые системные действия, такие как выделение памяти, вывод на экран, создание окон и т.п. Библиотека MFC Visual C++ инкапсулирует многие функции API. Хотя программам и разрешено обращаться к ним напрямую, все же чаще это выполняется через соответствующие функции-члены языка C++.

❷ (в программировании) Набор соглашений по процедурам вызова, определяющих способ запроса приложениями различных услуг. Часто под API понимается набор процедур операционной системы, которые приложение может вызывать для осуществления различных низкоуровневых операций.

❸ Спецификация для программиста, как необходимо писать приложение, с целью управления поведением и состоянием классов и объектов.

❹ Набор функций, реализуемых операционной системой с целью оказания услуг исполняемым приложениям, взаимодействия между ними и управления устройствами компьютера из приложений. Поддержка таких функций на уровне операционной системы делает совершенно ненужным их программирование в каждом приложении. По сути, API обеспечивает взаимодействие приложений с операционной системой. Называется также *программным интерфейсом*.

Application (Приложение. Синоним-*program* (программа); см. также *приложение*)

Application model (модель приложения)

Одна из моделей дисциплины *Microsoft Solution Framework*. Предлагает методику создания модульных приложений, обеспечивающих достаточную гибкость для достижения желаемой масштабируемости, производительности, расширяемости и распределённости приложений.

ASP (Application Service Provider – Провайдер услуг доступа к приложениям.). *(См. Outsourcing)*

❶ Агрегирование, продвижение и посредничество в распространении информационных (IT) сервисов для доставки IT ориентированных решений в сетях по ценам, согласуемым с подписчиками на заказанные услуги. (Gartner Group, 1999).

❷ Сторонние организации, которые управляют услугами и распределяют услуги, основанные на эксплуатации программных средств и программных *решений* для выполнения задач покупателей (клиентов) в распределённой сетевой среде из центрального узла управления. По существу, ASP обеспечивают компаниям возможность перераспределить часть или большинство своих информационных потребностей под ответственность выполнения их третьими фирмами.

По классификации ASPnews.com, ASP могут быть разбиты на следующие категории:

–ASP масштаба предприятия (Enterprise ASPs) – обеспечивают клиентов приложениями уровня конечного пользователя (high-end business applications).

–Локальные или региональные ASP (Local/Regional ASP) – обеспечивают клиентов широким спектром услуг сервисов для мелкого бизнеса в локальных сетях.

–Специализированные ASP – обеспечивающие клиентов в сфере специализированных решений, таких как сервисы Web-сайтов и человеческие ресурсы.

–ASP Вертикального рынка– обеспечивающие поддержку в специфических отраслях, таких как здравоохранение.

–ASP Массового рынка – обеспечивающих большинство бизнесменов малого и среднего уровня услугами конечных приложений.

❸ ***Поставщики услуг (сервисов) приложений*** представляют собой компании, которые предлагают частным лицам или предприятиям доступ через Интернет к приложениям (applications) и сопутствующим сервисам, которые, в противном случае, должны были бы находиться на собственных компьютерах указанных частных лиц или предприятий. Часто именуются "приложения под рукой" ("apps-on-tap") или аутсорсингом. Пионерами аутсорсинга являлись компании Hewlett-Packard, Xerox, SAP (со своим популярным, но дорогим для фирм, программным продуктом R/3) и Microsoft со своими продуктами: BackOffice, SQL Server, Exchange Server и Windows NT Server.

❹ Провайдер услуг доступа к приложениям - компания, занимающаяся сдачей в аренду, обслуживанием и продажей прикладных программ на своей технологической базе. Обычно услуги такой компании нацелены на:

–хостинг сайтов и почтовых служб;

–эксплуатацию ERP-систем, Интернет-магазинов и торговых площадок;

–доступ к сводным каталогам Интернет-продавцов;

–предоставление защищенного доступа в сеть и др.

Готовые ASP-решения позволяют минимизировать риск и финансовые затраты при вхождении в Интернет-бизнес.

Active Server Page (ASP – активные серверные страницы).

Среда, позволяющая выполнять на сервере приложения (скрипты), написанные на языках VBScript, Jscript и некоторых других. ASP представляется HTML страницей, которая включает один или более скриптов (script), являющихся небольшими встроенными программами, обрабатываемыми на сервере *Microsoft Web server*, перед тем, как страница будет отослана пользователю для отображения в браузере. ASP напоминает технологию *общего сетевого интерфейса* (common gateway interface, CGI), используемого на стороне Web-сервера и обычно применяется для формирования результирующей страницы "на лету" после обработки запроса к базе данных перед посылкой её обратно клиенту. Технология ASP предоставляется в рамках использования продукта Microsoft Internet Information Server (IIS). Файл ASP может формироваться путём встраивания кодов VBScript или JScript в код HTML или путём использования в файле HTML программных инструкций ActiveX Data Objects (ADO).

Application-to-Application (Из приложения в приложение)

Передача данных из одного программного продукта напрямую в другой программный продукт. Согласно EDI, данные программы должны располагаться в системах торговых партнеров.

ASCII (American Standard Code for Information Interchange)

Американский, стандартный код для обмена информацией. Код ASCII является наиболее популярным в локальных компьютерных сетях способом представления символьной информации. Стандартные символы ASCII кодируются семью битами, то есть значениями от 0 до 127. Остальные (оставшиеся до 256) 128 символов составляют расширенный набор ASCII, состав которого может меняться в зависимости от используемого национального языка.

ASP.NET

ASP.NET (ранее технология называлась ASP+) – это больше, чем следующая версия ASP. Это унифицированная среда разработки Web-приложений, обеспечивающая новую модель разработки и инфраструктуру, позволяющие создавать фунциональные приложения уровня предприятия и включающие все необходимые для разработчиков сервисы построения таких приложений. Пользователи могут постепенно расширять функциональность ASP-приложения, добавляя в него функциональность ASP.NET. ASP.NET - это также среда, основанная на .NET Framework, поэтому приложения можно создавать на любом языке программирования, совместимым с .NET Framework, включая Visual Basic, C# и JScript. Кроме того, для любого ASP.NET приложения доступны все возможности платформы .NET Framework, включая полностью управляемую, защищенную и многофункциональную среду выполнения программы, упрощенную разработку и внедрение, а также "бесшовную" интеграцию с большим количеством других языков программирования.

Assembly (асембл, пакет, комплект)

В архитектуре .NET, набор ресурсов и типов, а также метаданные, описывающие типы и методы, реализованные в структуре *assembly*. Таким образом, *assembly* – это самоописанный компонент. Основное преимущество таких компонентов в том, что для их использования не нужны никакие другие файлы.

ATA (AcceleraTed hub Architecture)

Архитектура, применяемая в чипсетах Intel i810 и i815. Предназначена для увеличения пропускной способности канала обмена данными и поэтому шина между южным и северным мостами, называемыми теперь хабами, имеет пропускную способность 266 Мбайт/с.

- B -

B2B (business-to-business – бизнес-бизнесу)

❶ Интеграция межкорпоративных систем экономической направленности. Вид маркетинговых коммуникаций, которые ориентированы на работу между компаниями в процессе производства и продажи продукции, товаров и услуг.

❷ Системы безбумажного платежного и другого документооборота между сложившимися кооперациями промышленности и любых сфер бизнеса.

❸ Онлайновое выполнение транзакций между компаниями, организациями или правительственными учреждениями.

B2B Portal (B2B-портал)

Портал, предназначенный для онлайнового взаимодействия между предприятиями. B2B-порталы могут быть вертикальными и горизонтальными. Вертикальные порталы строятся для обслуживания специфических рыночных ниш. Горизонтальные (функциональные) порталы обеспечивают определенные функции и сервисы независимо от отрасли. Например, сервисы логистики, страхования, юридических услуг и т.д.

B2C (business-to-consumer – бизнес-потребителю)

❶ Вид маркетинговых коммуникаций, которые ориентированы на работу с конечными физическими потребителями товаров или услуг.

❷ Онлайновое выполнение транзакций между компаниями и организациями, предлагающими товары общего назначения, с одной стороны, и потребителями этих товаров, с другой стороны.

B2G, G2C, G2G

Аббревиатуры, обозначающие новые сферы бизнеса, в которые, так или иначе, вовлечено государство (Government) - Business-to-Government, Government-to-Citizens, Government-to-Government. Являются следствием включения Государства в процесс электронизации всех видов деятельности. Концепция Electronic Government была оглашена в США на самом высоком правительственном уровне первого июля 1997 года.

Backbone (Магистраль. Бэкбон. Опорная сеть).

Магистральная сеть связи. Часть коммуникационной сети, которая передает трафик с использованием наиболее высокоскоростных (и часто наиболее протяженных) трактов в сети.

BASIC (beginner's all purpose symbolic instruction code)

БЕЙСИК, простой для изучения и применения язык программирования, ориентированный на диалоговую работу.

Back-End Systems (Исполнительные системы)

Унаследованные корпоративные системы, которые занимаются обработкой заявок, управлением материально-техническими запасами и взаиморасчетами, как для продавцов, так и для покупателей.

Bean (Java)

Повторно используемый программный компонент. Может служить составной частью при создании приложений.

BEDO (Burst Enhanced Data-Out RAM)

Более быстрая модификация памяти типа EDO.

Best-of-breed solution.

Решение нового поколения. Сюда относятся решения, которые фокусируются на узкой части общего спектра функций, реализуемых в пакете общего назначения. К примеру, сюда может относиться система, которая специализируется на взаимоотношениях с клиентами (customer relationships), в отличие от той, в которой эта функция является одной из многих.

BI (Business intelligence – интеллектуальный бизнес)

❶ Интеллектуальный анализ данных.

❷ Пользователецентрический процесс, который включает доступ и исследование информации, ее анализ, выработку интуиции и понимания, которые ведут к улучшенному и неформальному принятию решений. (Gartner)

❸ Методы, технологии, средства извлечения и представления знаний. Согласно первоначальным определениям, BI — это процесс анализа информации, выработки интуиции и понимания для улучшенного и неформального принятия решений бизнес-пользователями, а также инструменты для извлечения из данных значимой для бизнеса информации. Термин BI, включает также и технологию управления знаниями Knowledge Management (KM), которая также связана с анализом неструктурированной или слабоструктурированной информации (например, HTML страниц). KM обеспечивает категоризацию, разведку и семантическую обработку текстов, расширенный поиск информации и др. Сегодня категории BI-продуктов включают BI-инструменты и BI-приложения. Первые, в свою очередь, делятся на генераторы запросов и отчетов. Развитые BI-инструменты — это прежде всего инструменты оперативной аналитической обработки (online analytical processing, OLAP), корпоративные BI-наборы (enterprise BI suites, EBIS) и BI-платформы. Средства генерации запросов и отчетов в большой степени поглощаются и замещаются корпоративными BI-наборами. Многомерные OLAP-механизмы или серверы, а также реляционные OLAP-механизмы являются BI-инструментами и инфраструктурой для BI-платформ.

❹ Знания, добытые о бизнесе с использованием различных аппаратно-программных технологий. Такие технологии дают возможность организациям превращать данные в информацию, а затем информацию в знания. Это определение четко разграничивает понятия «данные», «информация» и «знания». Данные понимаются как реальность, которую компьютер записывает, хранит и обрабатывает — это «сырые данные». Информация — это то, что человек в состоянии понять о реальности, а знания — это то, что в бизнесе используется для принятия решений. В процессе организации информации для получения знания часто применяют хранилища данных, а для представления этого знания пользователям — инструменты бизнес-интеллекта.

BIOS (Basic Input/Output System, Базовая Система Ввода-Вывода)

Встроенное в персональный компьютер (ПК) программное обеспечение, которое доступно ему без обращения к диску. BIOS содержит код, необходимый для управления клавиатурой, видеокартой, дисками, портами и многими другими устройствами. Кроме того, он поддерживает выполнение экранных операций, тестирование устройств и начальную загрузку операционной системы (ОС). Обычно BIOS размещается в микросхеме ПЗУ (ROM–Read Only Memory), располагаемой на материнской плате компьютера (поэтому этот чип часто называют ROM BIOS). Эта технология позволяет BIOS всегда быть доступным пользователю и ОС, несмотря на повреждения, например, дисковой системы. Это также позволяет компьютеру самостоятельно выполнять загрузку ОС и поддерживать дальнейшее взаимодействие с устройствами ПК. Поскольку доступ к RAM (оперативной памяти) осуществляется значительно быстрее, чем к ROM (постоянной памяти), многие производители компьютеров создают системы таким образом, чтобы при включении компьютера выполнялось копирование BIOS из ROM в оперативную память. Задействованная при этом область памяти называется Shadow Memory (теневая память). В настоящее время, почти все материнские платы комплектуются Flash BIOS, т.е. BIOS’ом, который в любой момент может быть перезаписан в микросхеме ROM при помощи специальной программы прожига. BIOS PC максимально стандартизирован, поэтому, в принципе менять его, также как, например, операционные системы нет необходимости. Дополнительные возможности компьютера можно получать только использованием нового программного обеспечения – ОС, драйверов, системных утилит и т.д. BIOS, который поддерживает технологию Plug-and-Play, называется PnP BIOS. При использовании этой технологии BIOS должен быть обязательно прошит во Flash ROM.

BizTalk server

Сервер производства Microsoft, предназначенный для управления Web сервисами, а также выполнения функций упорядочения и управления запасами и производством.

BLOB (binary large object – большой двоичный объект)

Набор двоичных данных, имеющий большие размеры (как правило, 10–100 мегабайт) и хранящийся в виде отдельной сущности в базе данных или файловой системе (как файл). Обычно, используются для хранения мультимедийных объектов, таких как изображения, видеоданные и звуковые данные, а также фрагменты программ и кода.

BPEL (Business Process Execution Language)

Язык выполнения бизнес процессов (см. *XML-Related Terms and Definitions*).

BPML (Business Process Markup Language)

Язык разметки бизнес процессов (см. *XML-Related Terms and Definitions*).

- C -

CAD/CAM (computer-aided design/ computer-aided manufacturing)

Система автоматизированного проектирования и производства.

CALS

Концепция непрерывной компьютерной поддержки жизненного цикла изделия. Такая поддержка осуществляется созданием единой интегрированной модели изделия, сопровождающей изделие на всём протяжении его жизненного цикла. Наиболее важным событием в сфере использования новых информационных технологий в индустрии развитых стран представляется появление и развитие CALS-технологий. По мере развития этого направления информационных технологий интерпретация аббревиатуры CALS изменялось, отражая их постепенную эволюцию:

1985 – Computer-Aided of Logistics Support;
1988 – Computer Acquisition and Logistics Support;
1993 – Continuous Acquisition and Lifecycle Support;
1995 – Commerce At Light Speed.

Основным содержанием CALS-технологий является создание стандартных “интерфейсов” для различных промышленных технологий, бизнес-процессов, других сфер человеческой деятельности. Движущей силой развития этого направления информационных технологий стало осознание нарастающей сложности проблем, возникающих “на стыках” различных технологических процессов. К ключевым областям CALS в настоящее время относятся:

Реинжиниринг и управление проектами;
Параллельное проектирование;
Виртуальное предприятие;
Электронный обмен данными;
Распределённые системы поддержки принятия решений;
Интегрированная логистическая поддержка;
Многопользовательские базы данных;
Метаописание систем понятий и их хранение;
Репозитории метаописаний предметных областей;
Международные стандарты.

CASE (Computer Aided Software Engineering – Автоматизированное проектирова-ние и создание программного обеспечения (ПО))

Программные средства, поддерживающие процессы создания и сопровождения информационных систем (ИС), включая процессы анализа и формулировки требований, проектирование прикладного программного обеспечения (ПО) (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с

системным ПО и техническими средствами образуют функционально полную среду разработки ИС. Как правило, предполагают наличие репозиториев, предназначенных для хранения и дальнейшего использования артефактов разрабатываемого ПО.

Case study

❶ Показательный пример, отражающий типичные для сегодняшнего бизнеса проблемы обработки информации. Каждый пример начинается с вводной информации о бизнесе, после чего описываются проблема и ключевые фигуры.

❷ Обучение на реальных бизнес-примерах.

❸ В общественных или медицинских науках, термин обозначает анализ поведения одного представителя в популяции (совокупности) или одиночное событие в серии.

CBSD (Component-based software development – разработка компонентного программного обеспечения)

Разработка компонентного программного обеспечения направлена на построение больших программных систем, путём интегрирования ранее разработанных программных компонентов. Повышая гибкость и надёжность систем, такой подход одновременно позволяет снижать стоимость разработок программного обеспечения, ускорение интеграции систем и сокращение сроков прохождения этапов жизненного цикла больших программных систем на этапах поддержки и обновления. С данной технологией связан процесс компонентной разработки ПО (component-based software engineering, CBSE), а также коммерческие стандартные (commercial-off-the-shelf, COTS) программные компонентные продукты.

CC (Carbon Copy)

«Под копирку» – часть заголовка электронной почты, который показывает вторичных получателей сообщения.

CCM (content and collaboration management)

Система управления контентом.

CDI (Customer Data Integration – интеграция данных о потребителях)

Один из важнейших компонентов в структуре рынка CRM. Включает комбинацию технологий, программного обеспечения, процессов и сервисов, предназначенных для создания единого, точного и полного представления о потребителе в пределах предприятия.

CD-ROM (Compact Disc-Read Only Memory)

Термин, относящийся к устройству для чтения лазерных компакт-дисков и к самим компакт дискам с записанными на них данными.

CERN

Европейская лаборатория физики элементарных частиц в Женеве (Швейцария). Организация, ответственная за создание World Wide Web.

Chat (Чат)

Средство для обмена сообщениями в среде Internet в реальном времени с помощью клавиатуры компьютера.

Chipset (Chip Set) набор микросхем

Одна или несколько микросхем и таймеры, то есть система управления, специально разработанная для "обвязки" микропроцессора. Комплекс содержит в себе контроллеры прерываний и прямого доступа к памяти, соединения между памятью и шиной – то есть все те компоненты, которые в первоначально выпускаемых IBM PC были собраны на отдельных микросхемах.

CGI (Common Gateway Interface – общий межсетевой интерфейс)

Когда Web-сервер запускает программу для отправки документа, который может представлять собой HTML-текст, графическое изображение или иной тип данных, диалог сервера с программой-браузером определяется протоколом CGI, а запускаемая сервером программа называется программой CGI или сценарием CGI. Сервер сообщает программе CGI, какая страница была затребована, какие значения были переданы в HTML-формах, откуда поступил запрос, какие данные использовались при аутентификации и многое другое.

Chief Information Officer (CIO)

Наименование *лица* в коммерческой компании или неприбыльной (nonprofit) организации, являющегося ответственным за управление потоками официальной информации, эксплуатируемые компьютерные средства и другие, связанные с этим материальные ценности, а также определяющего процессы регулирования и использования применяемых в этих процессах компьютерных средств и их функций.

Clone (клон, имитация, аналог)

❶ Вычислительная система или персональный компьютер, совместимые с персональным компьютером IBM PC.

❷ Программа или вычислительная машина, реализующие возможности прототипа в упрощённом варианте.

CLR (Common Language Runtime – среда исполнения) (в архитектуре .NET)

Система, управляющая исполнением программы, выполненной на любом языке программирования. Управляет исполнением кода, адаптированного к системе .NET, и работает она следующим образом. Компилируя C# и любую другую .NET-программу, получает файл со специальным кодом, названным промежуточным языком Microsoft (Microsoft Intermediate Language, **MSIL**).

Cluster (см. *кластер*)

CMOS (Complementary Metal Oxide Semiconductor)

Эта аббревиатура относится к технологии изготовления полупроводниковых устройств, более экономичных с точки зрения потребления электроэнергии, но в настоящее время используется и в отношении устройств энергонезависимой памяти. Для систем класса PC обозначает 64-байтовую схему памяти с батарейным питанием, используемую для хранения параметров оборудования. Кроме того, обычно включает в себя часы реального времени (RTC - Real Time Clock).

COM (Component Object Model) (компонентная объектная модель)

Открытая архитектура для кросс платформенных разработок клиент/серверных приложений, которая лежит в основе технологий ActiveX, DirectX и OLE 2.0. Спецификация, модель и технология корпорации Microsoft, предназначенные для построения и разработки компонентов программного обеспечения и их интерфейсов. COM устанавливает абстракции и правила, необходимые для определения реализуемых объектов и их интерфейсов. В ее состав входит также программное обеспечение, реализующее ключевые функции. Сами компоненты легко объединяются в программы или могут быть добавлены к существующим программам, чтобы придать им большую функциональность. Компоненты пишутся на разных языках (чаще других при этом используется язык C++). COM сервер обычно является .DLL или .EXE файлом. Реализованный в виде DLL, COM сервер называется сервером "в процессе" (in-process), поскольку размещается в том же адресном пространстве, что и клиент. Клиент может напрямую вызвать запрашиваемый объект, что осуществляется быстрее и эффективнее. Реализованный в виде EXE-файла, COM сервер называется "вне процессным" (out-process), так как он запускается в своём собственном пространстве процесса.

COM+

Модернизация модели взаимодействия COM и Microsoft Transaction Server, которая упрощает разработку сложных распределенных приложений.

Common ground (Общая основа)

Формат приложения и файла, который позволяет просматривать документы на различных платформах. Например, документы в формате PDF можно просматривать в Windows, UNIX или Macintosh. Документы на "Общей основе" обычно включают в себя программу просмотра (viewer).

Compound document (см. *составной документ*)

Computer (см. *компьютер*)

Computer science (компьютерные науки)

❶ На начальных этапах развития информационных технологий, рассматривалась, как дисциплина, изучающая вычислительные машины, принципы их построения и использования. Включала исследования таких аспектов как: программирование, информационные структуры, разработку программного обеспечения, языки программирования, компиляторы и операционные системы.

❷ Теоретическая дисциплина, охватывающая теорию и методы построения вычислительных и программных систем. Знание компьютерной науки необходимо специалистам по программному обеспечению так же, как знание физики – инженерам-электронщикам.

❸ Отрасль знаний, изучающая информационные процессы, происходящие в компьютерах и отображаемые в них. "Компьютерные науки" концентрируют свое внимание на различных аспектах, связанных с протеканием и использованием информационных процессов, с теми структурами, в которых представляется информация, и теми процедурами, которые используются при её переработке. Последнее связывает область "компьютерных наук" с теорией машин для переработки информации - компьютеров - и методами их использования в системах переработки информации.

Configuration file (Файл конфигурации)

Файл, в который записываются различные параметры прикладной программы, например, тип используемого модема.

Connection

Соединение. Путь обмена информацией, который установили два взаимодействующих компьютера.

Connection-oriented (Основанный на соединении)

Модель связи, при которой сеанс связи проходит три фазы: установление соединения, передача данных, разрыв соединения. Примеры: X.25, Internet TCP, обычный телефонный звонок.

Context-sensitive (см. *контекстно-чувствительный*)

Cookbook

Тип книг «как-это-делать», которые содержат инструкции для приготовления пищи, включая рецепты для специфических блюд, замечания о продуктах и других ингредиентах и много другой полезной информации. В информационных технологиях публикации под таким названием содержат описания технологии решения специфических задач.

Cookie

Небольшие строки с данными, создаваемые Web сервером и передаваемые на компьютер клиент, соединяющийся с ним через Интернет и сохраняемые в специальном cookie-файле используемого Web браузера. Обычно применяются для сокращения времени, требуемого для идентификации и повторного соединения с соответствующим Web сайтом, путём использования cookie-файла, сохранённого на жёстком диске после предыдущего визита.

Cool Talk (Прохладный Разговор)

Телефонные переговоры по Интернет. Средство включено в Netscape Navigator. Обеспечивает высококачественную звуковую связь.

Copyright

Защита, обеспечиваемая законом, от несанкционированного копирования и распространения.

CORBA (Common Object Request Broker Architecture – Брокер запросов общей объектной архитектуры)

❶ Стандарт, который был предложен консорциумом OMG для организации взаимодействия распределенных объектов. Архитектура CORBA позволяет выполнять в сети программы, написанные на любом языке, независимо от того, на какой платформе они запускаются. Таким образом, крупные корпорации получают возможность в короткие сроки

создавать достаточно сложные системы. В архитектуре CORBA клиент выполняет запрос к общему интерфейсу, который называется брокером объектных запросов (Object Request Broker, ORB). Брокер ORB пересылает запрос соответствующему объекту, а затем возвращает клиенту полученные результаты. Является спецификацией кросс платформенных распределённых вычислений, продвигаемых на рынок ИТ консорциумом OMG, в противовес COM-архитектуре корпорации Microsoft.

❷ Открытая независимая от производителей архитектура, инфраструктура и спецификация распределенных объектов, реализуемых приложениями при работе в сетях. Использование стандартного протокола IIOP (Internet Inter Object Request Broker Pritocol) обеспечивает полную интероперабельность взаимодействия между любыми компьютерами, операционным системам, языками программирования и сетевыми структурами.

Country code

Большинство стран, имеющих выход в Интернет, имеют двухбуквенное обозначение по стандарту ISO 3166. Эти две буквы есть адрес основного домена для данной страны. Например: uk - Великобритания, fi - Финляндия, ru - Россия.

Cracker (Взломщик)

Пользователь, занимающийся поиском незаконных средств доступа к компьютерным ресурсам (в том числе и к сайтам, содержащим конфиденциальную информацию).

Cram, cramming (Враньё)

Практика некоторых телефонных компаний, которые добавляют фальшивые суммы в телефонный счёт за звонки, которых вы не делали.

CRC (Cyclic Redundancy Check – Циклический контроль по избыточности)

Процедура проверки на ошибку при передаче данных. Передающее устройство вычисляет некоторое число из передаваемых по сложному алгоритму и передаёт это число принимающему устройству. Приёмник производит аналогичные вычисления и сравнивает вычисленное и полученное от передатчика. Если они совпадают, считается, что передача прошла успешно. В противном случае считается, что данные изменились при передаче, то есть, возможно, они приняты с ошибкой.

CRM (Customer Relationship Management)

Служба управления отношениями с клиентами, реализованная средствами компьютерных технологий. Методология, программное обеспечение и возможности Интернета, которые помогают компании управлять и организовывать взаимоотношения с клиентами. Помогает определять (идентифицировать) и относить клиентов к какой-либо из употребляемых категорий.

CWM™ (Common Warehouse Metamodel – Общая Метамодель Хранилища Данных)

Общая Метамодель Хранилища Данных (CWM™) является созданной OMG архитектурой и технологией для управления сложным жизненным циклом корпоративных данных и контента в Интернете и Интранете, с полной интероперабельностью приёма и передачи их в данных средах.

- D -

DAO (Data Access Objects)

Интерфейс прикладного программирования для Microsoft Jet Database Engine, используемого в приложении Microsoft Access. DAO основывается на иерархической объектной модели, образованной всеми объектами инструмента Jet. Поскольку Jet позволяет подключаться к источникам данных ODBC, DAO можно использовать и для доступа к источникам данных ODBC.

Data-Based Knowledge (Знания, основанные на данных)

Знания, которые выводятся путем обработки данных интеллектуальными инструментальными средствами анализа из хранилища данных.

Data Mart (Киоск или витрина данных)

База данных, имеющая то же назначение, что и хранилище данных, но обычно меньшая по объему и сконцентрированная на данных одного подразделения или рабочей группы предприятия.

Data Mining (Извлечение смысла из данных)

❶ Процесс поиска скрытых зависимостей, взаимосвязей и потенциальных перспектив объединения по определённым критериям в больших скоплениях данных.

❷ Концепция, которая строится на базе систем создания и поддержки в актуальном состоянии хранилищ данных (DW-DATA Warehouse) и на системах «добычи» знаний (data mining) из DW. В отечественной литературе термин "data mining" трактуется как интеллектуальный анализ данных (ИАД). В дальнейшем эта технология переросла в Business Intelligence (BI) — знания, добытые о бизнесе с использованием различных аппаратно-программных технологий. Такие технологии дают возможность организациям превращать данные в информацию, а затем информацию в знания.

Data Warehouse (Хранилище данных)

База данных очень больших размеров (от 1 терабайта и выше), где собираются данные для последующего анализа, в частности, в масштабах предприятия.

DBMS — Data Base Management System (*см. СУБД*).

DCOM (Distributed Component Object Model - распределённая компонентная объектная модель)

Расширение модели корпорации Microsoft COM, обеспечивающее прозрачное взаимодействие объектов через локальную сеть или Интернет. Если клиент и вызванный им компонент (т.е. сервер), находятся на разных машинах, DCOM подменяет локальный механизм взаимодействия объектов сетевым протоколом. Ни клиент, ни компонент ничего не знают о том, что на самом деле они взаимодействуют, находясь на различных компьютерах.

DDB (Microsoft Digital Dashboard – электронная информационная панель)

Продукт, который позволяет создавать настраиваемые решения для сотрудников организаций, работающих с информацией. Основной составляющей электронной информационной панели, являются портлеты, в терминологии Microsoft называемые Web Part или DDB-компонентами. Таким образом, информационная панель Digital Dashboard может объединять персональную, групповую, корпоративную и внешнюю информацию, предоставляя доступ к различным инструментам анализа и источникам информации.

DDL (Data Definition Language – язык определения данных)

Операторы языка SQL, предназначенные для создания и манипулирования сущностями реляционной базы данных. Примером таких операторов являются операторы CREATE TABLE и DROP INDEX.

DDR (Double Data Rate)

Стандарт памяти и технология, приводящие к удвоению скорости передачи данных между памятью и процессором. Необходимость создания связана с пропускной способностью памяти, а точнее, шин память-контроллер и контроллер-процессор. Чем больше информации можно передавать по ним за единицу времени, тем активнее будет загружен процессор, и тем эффективнее он будет работать. В 2002 году принят новый стандарт памяти DDR-II, который, по задумке 120 крупнейших в мире компаний по производству чипов должен стать стандартом к 2003 году. Напряжение, которое требуется такой памяти для нормальной работы – 1,8 вольт, частоты работы – с 400 до 533 МГц, скорость передачи данных – от 3,2 Гб до 4,3 Гб в секунду.

Delimiter (Делимитер)

В компьютерных программах и передаваемых в сетях наборах данных, *делимитер* является символом (character), который обозначает начало или конец строки символов или конечной последовательности символов. В то же время, сам делимитер не является частью

обрамляемой с его помощью строки. В синтаксисе командных строк часто представляется пробелом, бэкслэшем (\) или слэшем (/).

Deployment (Deploying)

Инсталляция (развертывание) распределенной программной системы. Как правило, состоит из двух важных частей: *топологии развёртывания*, определяющей, на каких системах будут размещены те или иные компоненты решения, а также процесса развёртывания, описывающего шаги по непосредственному распределению частей по целевым системам.

Design pattern (проектный (конструкторский) шаблон) (см. *паттерн, паттерны проектные*)

Проектный шаблон является описанием коммуникационных объектов и классов, которые могут быть изготовлены по заказу, т.е. доработаны для решения конкретной проблемы.

DIMM (Dual In-Line Memory Module)

Наиболее современная разновидность форм-фактора модулей памяти. Отличается от SIMM тем, что контакты с двух сторон модуля независимы (dual), что позволяет увеличить соотношение ширины шины к геометрическим размерам модуля. Наиболее распространены 168-контактные DIMM (ширина шины 64 бит), устанавливаемые в разъём вертикально и фиксируемые защелками. В портативных устройствах широко применяются SO DIMM.

DIP (Dual In-line Package)

Микросхемы с двумя рядами контактов, расположенными вдоль длинных сторон чипа и загнутых "вниз". Чрезвычайно распространенная упаковка во времена "до" «модулей памяти».

DLL (Dynamic-Link Library – динамически компонуемая библиотека)

Технология формирования библиотек программ, используемых приложениями, разработанная Microsoft. В отличие от обычных библиотек, являющихся неотъемлемой частью приложения и присоединяемых к каждому приложению на этапе компоновки, DLL является самостоятельным компонентом приложения, загружаемым в оперативную память только тогда, когда осуществляется обращение к её внутренним компонентам и выгружается из оперативной памяти, когда необходимость в их использовании отпадает. Это и называется динамической компоновкой. Файлы динамически компонуемых библиотек имеют расширение DLL.

DMA (Direct Memory Access - прямой доступ к памяти)

Технология организации непосредственного доступа к памяти процессора. Способ обмена данными между внешним устройством и памятью без участия процессора, что может заметно снизить нагрузку на процессор и повысить общую производительность системы. Режим DMA позволяет освободить процессор от рутинной пересылки данных между внешними устройствами и памятью, отдав эту работу контроллеру DMA. Процессор в это время может обрабатывать другие данные или решать другую задачу в многозадачной системе.

DNA (Microsoft Windows DNA – Windows Distributed interNet Applications)

Модель многоярусного распределённого приложения, основанная на концепции кооперации компонентов (cooperating component). Эти компоненты создаются с использованием COM-моделей и технологий.

DNS (Domain Name System – система имен доменов или Domain Name Service – служба доменных имен)

Распределенный механизм имен/адресов, используемых в сети Internet. Используется для преобразования *логических имен* в IP-адреса. DNS используется в сети Internet, обеспечивая возможность работы с понятными и легко запоминающимися именами вместо неудобоваримых чисел IP-адреса.

DOM (Document Object Model – Объектная модель документа)

DOM является спецификацией W3C, определяющей представление элементов в документах на языке XML, а также обеспечивает языко-независимую и платформо-независимую объектную модель для XML-документов. DOM обеспечивает интерфейс прикладного программирования (API - Application Programming Interface) для упрощения доступа к XML документам, которые могут использоваться любыми приложениями, которые предназначены для манипулирования документами (к примеру, Word, Excel, Adobe Acrobat Reader и т.д.) (см. *XML-Related Terms and Definitions*).

Domain (домен, область)

❶ Термин, обозначающий группу хостов (компьютеров) сети. Деление на группы может осуществляться по физическим (местоположение в сети) или логическим (функциональное предназначение) критериям. В *OSI* термин домен используется как административное деление сложных распределенных систем, как в MHS PRivate Management Domain (PRMD) и Directory Management Domain (DMD).

❷ В сети Internet - часть иерархии имен. Синтаксически, доменное Интернет-имя содержит последовательность имен (меток), разделенных точками (.). К примеру, peterburg.net.

Dotcom

Компания, чья деятельность полностью протекает в Интернете.

dpi (dot per inch – точек на дюйм)

Единицы, характеризующие *разрешение* растровых графических изображений, то есть числа, указываемые в технических характеристиках принтеров, мониторов и других компьютерных устройств. Например, 640x480 dpi, 800x600 dpi и т.д. Первая цифра указывает общее количество единичных элементов растра отображаемой прямоугольной области по ширине, а вторая - по высоте. Чем выше разрешение, тем точнее растровая карта воспроизводит изображение и тем больше общее количество единичных элементов и, соответственно, размер файла, в котором хранится картинка.

DRAM (Dynamic RAM)

Динамическая память – разновидность RAM, единичная ячейка которой представляет собой конденсатор с диодной конструкцией. Наличие или отсутствие заряда конденсатора соответствует единице или нулю. Основной вид, применяемый для оперативной памяти, видеопамяти, а также различных буферов и кэшей более медленных устройств. По сравнению со SRAM заметно более дешевая, хотя и более медленная по двум причинам - емкость заряжается не мгновенно, и, кроме того, имеет ток утечки, что делает необходимой периодическую подзарядку её элементов.

DRAM module

Модуль памяти - устройство, представляющее собой печатную плату с контактами, на которой расположены чипы памяти (иногда заключенное в корпус), и представляющее собой единую логическую схему. Помимо чипов памяти может содержать и другие микросхемы, в том числе шунтирующие резисторы и конденсаторы, буферы и т.п.

DSS (Decision Support System – Система поддержки принятия решения) (см. *Система поддержки принятия решений*)

Программная система, сконструированная для помощи при принятии решений. Может включать аналитические, статистические, геоинформационные и многие другие функции и средства.

DTD (Document Type Definition –Определение типа документа)

DTD является описанием структуры и свойств класса файлов язіков XML или SGML. DTD определяет грамматику для класса документа. К примеру, DTD заказов на покупку (purchase orders), может определять элементы для количества, цены и т.д. Для более сложных структур можно использовать XML Схемы (XML Schemas), связанные с понятием XSD (см. *XML-Related Terms and Definitions*).

- E -

e-Business (электронный бизнес)

Повышение эффективности бизнеса, основанное на использовании информационных технологий, для того чтобы обеспечить взаимодействие деловых партнеров и создать интегрированную цепочку добавленной стоимости. Понятие «электронный бизнес» шире понятия «электронная коммерция», касающегося только коммерческой деятельности. Понятие «электронный бизнес» охватывает всю систему взаимоотношений с партнерами и заказчиками.

В состав программного обеспечения е–Бизнеса входят следующие составляющие:

• автоматизация продаж (SFA - Sales Force Automation);
• управление отношениями с клиентами (CRM - Customer Relationship Management);
• планирование ресурсов предприятия (ERP - Enterprise Resource Planning);
• планирование потребности в материалах (MRP - Material Requirements Planning)
• управление цепочками поставок (SCM - Supply chain management);
• управление конфигурацией ПО (SCM - Software configuration management);
• системы поддержки производственных процессов (MES - Manufacturing Execution Systems)
• планирование требуемой производительности (CRP - Capacity Requirements Planning)
• управление профсоюзами (Shop Floor Control)
• интеграции корпоративных приложений (EAI - Enterprise application integration);
• интеграция межкорпоративных систем (B2B - business-to-business)
• система управления контентом (CCM - Content and collaboration management)

Во втором квартале 2002 г. более одного миллиона малых и средних предприятий в США использовали одно или более бизнес-приложений автоматизации (SFA, CRM, ERP или SCM), что соответствует увеличению количества таких предприятий на 114 % по сравнению с тем же периодом 2001 г. Access Markets International (AMI) Partners, Inc. оценивает, что расходы малого и среднего бизнеса в США, связанные с приобретением лицензионного ПО автоматизации бизнес-процессов, вырастут до 4,2 млрд. долларов в 2006 г., что составит ежегодную норму роста (CAGR) 33 % в течение 2002–2006 гг.

e–Commerce (электронная коммерция) (см. *e-Business, Интернет-коммерция*)

❶ Общее определение нового явления: удаленных торговых операций, производимых при помощи телекоммуникаций.

❷ Маркетинг, подача предложений, продажа, сдача в аренду, предоставление лицензий, поставка товаров, услуг или информации с использованием компьютерных сетей или Интернета. Понятие «электронная коммерция» шире понятия «интернет-коммерция», поскольку в него входят все виды электронной коммерческой деятельности.

❸ Электронная коммерция является основополагающей системой электронизации и компьютеризации процессов развития бизнес-процессов в сетевой среде и подразумевает управление бизнесом он-лайн. Это включает, к примеру, покупку или продажу продуктов и товаров с переводом денег через цифровые сети с применением *электронного обмена данными* (Electronic Data Interchange, EDI).

e-Education (е-Образование) (см. *e-Learning*)

e-Learning (е-Обучение)

❶ Понятия e-Learning и e-Education означают процесс дистанционного образования в электронной среде и охватывают широкий спектр приложений и процессов, таких, как обучение, базирующееся на Web-технологиях и компьютерных технологиях, виртуальные классы, предоставляющие возможность совместного обучения. Эти понятия включают доставку обучающимся аудио- и видеоматериалов курсов посредством сети Интернет, сетей Intranet/Extranet (LAN/WAN), с помощью спутникового вещания, интерактивного

телевидения и записей на CD-ROM. В целом, данные термины объединяет три составляющие: открытое (широкодоступное) обучение, компьютерную поддержку всего процесса обучения и разветвлённую систему электронных коммуникаций, включая Интернет, для которых характерна асинхронность доступа (т.е. в любое удобное для пользователя время).

❷ Использование сетевых и Web-технологий для создания, доставки, отбора, администрирования, поддержки и распространения обучения в виде соответствующих элементов контекста. Другими словами, е-образование означает образование, осуществляемое частично или полностью с использованием средств электронных коммуникаций.

e-Trade (Электронная торговля)

Торговля, осуществляемая с помощью электронного документооборота в Интернете.

EAI (enterprise application integration)

❶ Интеграция корпоративных приложений. Комплекс мероприятий, направленный на оптимизацию решений корпоративных задач, путём объединения разнообразных и разноплановых приложений предприятия, а также используемых ими данных.

❷ Набор инструментальных средств, предназначенных для интеграции бизнес-процессов и приложений в рамках организаций, со структурой любой сложности. EAI повышает эффективность использования ИТ средств и оперативность обработки бизнес-данных, обеспечивая анализ и передачу достоверной информации в реальном масштабе времени, снижая тем самым время обмена данными в организации и обеспечивая эффективную инфраструктуру электронного проведения коммерческих операций (e-business).

ebXML (Electronic Business using eXtensible Markup Language–электронный бизнес, использующий язык XML)

Модульный набор спецификаций, позволяющий предприятиям любого размера и с любым географическим местоположением управлять своим бизнесом через Интернет. Компании, использующие ebXML, получают в распоряжение стандартный метод для обмена коммерческими сообщениями и документами, возможность единообразно управлять торговыми взаимоотношениями, представлять и обмениваться данными в едином поле представлений, а также определять и регистрировать бизнес процессы. Спецификация ebXML является совместной инициативой Организации Объединённых Наций (United Nations (UN/CEFACT)) и OASIS. Целью данной инициативы является создание спецификации, поддерживающую модульную модель электронного бизнеса, базирующуюся на XML. В настоящий момент разработаны подробные требования для электронного бизнеса, но сами технологии продолжают быстро изменяться и развиваться (см. *XML-Related Terms and Definitions*).

ECC (Error Correcting Code)

Система коррекции ошибок передачи или хранения данных для памяти, шин и других устройств. Позволяет автоматически на аппаратном уровне корректировать одиночные ошибки данных и обнаруживать двойные.

EDI (Electronic Data Interchange – Обмен электронными данными)

❶ Стандартный формат для обмена бизнес-данными. Разработан Data Interchange Standards Association (США). EDI-сообщение содержит строку *элементов данных*, каждый из которых представляет единственный факт, такой, как цена продукта, номер модели товара и т.д., отделяемые друг от друга *разделителями (делимитерами)*. Вся строка называется *сегментом данных*. Один или более сегментов данных обрамляется *заголовком* и концевой меткой *набор транзакций,* в совокупности представляющие блок EDI для передачи (эквивалент *сообщения*). Набор транзакций часто состоит из элементов так называемых бизнес-форм или бизнес-документов. Участники обмена EDI сообщениями называются торговыми партнёрами.

❷ Устаревший вариант электронной коммерции, более дорогой и громоздкий по сравнению с электронной коммерцией, базирующейся на Интернет. Доступен только для крупных компаний и их наиболее значительных торговых партнеров.

❸ Способ, с помощью которого компании могут использовать сети для делового взаимодействия. Если электронная переписка между компаниями - явление обычное, ЭОД подразумевает передачу больших объемов информации, заменяя большие бумажные документы такие, как счета и контракты.

EDO (Extended Data Out)

Расширенный выход данных.

EIDE (Enhanced Integrated Drive Electronics)

Расширенная интегрированная электроника для дисководов.

EIP (Enterprise Information Portal – корпоративный информационный портал)

❶ Способ собрать на одном экране всю необходимую сотруднику предприятия информацию для его эффективной работы. При создании порталов используется архитектурный шаблон на основе “толстого” Web-клиента, таким образом, для доступа к информации и поставляющим ее системам (финансовым, почтовым и другим) достаточно привычного Internet-браузера, поддерживающего апплеты Java и компоненты ActiveX. **(см. *портал*)**

❷ Компания Merrill Lynch одной из первых воспользовалась термином «корпоративный информационный портал» (enterprise information portal — EIP). Она так определяет это понятие: «Корпоративные порталы – программные приложения, позволяющие компаниям «расконсервировать» информацию, сохраняемую как внутри, так и вне их границ, а также предоставить каждому пользователю единую точку доступа к предназначенной для него информации, необходимой для принятия обоснованных управленческих решений». Следовательно, корпоративный информационный портал интегрирует внутренние приложения, такие как: приложения электронной почты, доступа к базе данных и управления документами, – с внешними приложениями, например службами новостей и потребительскими Web-узлами. Это Web-интерфейс, который дает пользователю возможность обращаться ко всем этим приложениям с экрана своего ПК.

EIS (Executive Information System – Оперативная информационная система)

Средства, разработанные для исполнительных руководителей высшего звена и обеспечивающие формирование заранее записанных отчетов или инструкций. Они предлагают мощные средства формирования отчетов и возможности для "углубления в данные" (drill-down). В настоящее время эти средства допускают формирование произвольных отчетов по многомерной базе данных, а большинство из них предлагает аналитические приложения, используемые в различных предметных областях, например при продажах или в финансовом анализе работы подразделения в контексте предприятия в целом.

EISA (Extended Industry Standard Architecture)

Расширенный стандарт подключения старых 8-ми и 16-ти разрядных адаптерных плат.

Enterprise (предприятие)

❶ Дословно, бизнес-организация, корпорация. В компьютерной индустрии, термин часто используется для описания любой большой организации, использующей компьютеры. Интранет, к примеру, является примером компьютерной системы предприятия.

❷ Общее понятие бизнеса, включающее в себя функции, подразделения или другие компоненты, используемые для полного формирования конкретных целей и задач.

Enterprise Data (Данные предприятия)

Данные, определенные для использования в корпоративной среде предприятия.

Enterprise Modelling (Моделирование предприятия)

Развитие общего согласованного представления и понимания элементов данных и их соотношений в рамках предприятия.

EPP (Enhanced Parallel Port – *улучшенный параллельный порт)*.

ERA (Entity Relationship Analysis – анализ сущностей и связей)

Процесс моделирования данных, направленный на выявление сущностей, их атрибутов и связей между ними.

ERP (Enterprise Resource Planning)

Средства планирования ресурсов предприятия. Интегрированные приложения, которые контролируют ежедневные бизнес операции такие, как управление запасами, продажа товаров, управление финансами и доходами, человеческими ресурсами и продвижение товаров от производства к потребителю.

ERP- система (Enterprise Resource Planning System)

Система планирования ресурсов предприятия, единая среда для автоматизации учета, контроля, анализа и планирования всех основных бизнес-операций корпорации. Система управления предприятием.

Ethernet

❶ Тип локальной вычислительной сети, разработанной корпорацией Xerox, в которой компьютеры взаимодействуют посредством передачи радиочастотных сигналов, посылаемых через коаксиальный кабель.

❷ Локальная вычислительная сеть на основе коаксиального кабеля, впервые описанная Меткалфом и Боггсом из Xerox PARC в 1976 г. В настоящее время признана стандартом отрасли.

Executive Information System (см. *EIS*)

EXPRESS

Язык EXPRESS является одним из разделов стандарта ISO 10303 STEP. Он описан в 11 томе стандарта ISO 10303. Язык EXPRESS предназначен для описания модели мира на концептуальном уровне. При этом принимается, что мир един и все в мире взаимосвязано. Следовательно, и описание мира тоже должно бы быть единым, цельным и неделимым. Но ввиду того, что мир велик и сложен, составить единое его описание пока трудно. Поэтому и приходится пока условно разбивать мир на предметные области. А описание мира пока приходится (также условно) разбивать на схемы (*SCHEMA*). Схема является самым верхним уровнем описания информационной модели. Вся информационная модель состоит из связанных между собой схем.

Extendibility (см. *Расширяемость*)

- F -

F2F (face-to-face) (лицом к лицу)

Термин, используемый для описания традиционной среды аудиторного обучения.

FAQ (Frequently Asked Questions – часто задаваемые вопросы).

Текстовый файл, содержащий список наиболее часто задаваемых вопросов и соответствующих ответов на них на любую тему по информационным технологиям. Является кратким введением в некоторую отрасль компьютерных знаний.

FAT (File Allocation Table)

Таблица размещения файлов и таблица индексов, указывающих местоположение файлов на диске. Поскольку каждый файл может занимать несколько блоков на диске, таблица FAT указывает последовательность блоков, занятых файлом. FAT создается для каждого тома. К примеру, операционная система NetWare разбивает каждый том на блоки выделения дискового пространства. Можно установить размер блока 4, 8, 16, 32 или 64 КБ. (Все блоки в пределах одного тома имеют одинаковый размер).

FDD (Floppy Disc Drive)

Устройство записи, считывания и хранения данных на гибком магнитном диске.

FDDI (Fiber Distributed Data Interface)

FDDI - распределенный интерфейс передачи данных по оптоволоконным каналам.

Предложенная комитетом ANSI стандартная спецификация сетевой архитектуры (X3T9.5), основанной на высокоскоростной передаче данных по оптоволоконным линиям связи. Сети FDDI обладают следующими особенностями:

- Для передачи данных используется многомодовое (multimode) или одномодовое (single-mode) оптоволокно.
- Максимальная скорость передачи данных составляет 100 Мбит/с.
- При организации сетей используется кольцевая топология. Сеть FDDI состоит из двух колец, информация по которым перемещается в противоположных направлениях.
- Для кодирования и передачи информации используются не электрические, а оптические сигналы.
- Кодирование данных осуществляется по схеме 4B/5B. При этом каждым четырем битам реальных данных ставится в соответствие пять передаваемых информационных битов. То есть, для достижения скорости передачи в 100 Мбит/с сеть должна работать с тактовой скоростью 125 Мбит/с.
- Сеть поддерживает до 1000 узлов, а протяженность одной сети может достигать 100 км.
- Максимальное расстояние между узлами может составлять до 2 км в случае применения многомодового, и до 40 км - для одномодового кабеля.

File–(в английском языке слово *file* имеет следующие значения) (см. *файл*)

I (технич.) 1) напильник 2) пилочка (для ногтей) 3) отделка, полировка 4) оглобля, дышло 5) (разговорное) ловкач

(глагол) 1) пилить, подпиливать 2) отделывать (стиль и т. п.)

II (существит.) 1) скоросшиватель (для бумаг); шпилька (для накалывания бумаг) 2) подшитые бумаги, дело; досье 3) подшивка (газет) 4) картотека

(глагол) 1) регистрировать и хранить (документы) в каком-л. определенном порядке; подшивать к делу 2) сдавать в архив 3) (амер.) представлять, подавать какой-либо. документ 4) принять заказ к исполнению

III (существит.); 1) ряд, шеренга; колонна (людей) 2) (в шахматах) вертикаль 3) очередь, хвост

IV (компьютерное) 1. файл (поименованная область на диске) || формировать [организовывать] файл; заносить в файл; хранить в файле

FLOPS (Float Operations per Second – операций с плавающей точкой в секунду)

Количество операций с плавающей точкой в секунду, которые характеризуют производительность микропроцессора при работе с вещественными (нецелыми) числами.

Freeware (бесплатные компьютерные программы)

Программное обеспечение, которое поставляется бесплатно, даже если автор сохраняет авторское право на программу. Авторы или компании создают бесплатные программы, руководствуясь принципами солидарности с другими компаниями либо с целью обеспечения продвижения других проектов, либо потому, что программа достаточно узкоспециализирована или коммерческое распространение её не имеет смысла.

FTP (File Transport Protocol)

Протокол передачи файлов. Способ перемещения файлов между различными компьютерами. В качестве транспортного механизма для передачи данных FTP применяет протокол TCP (см.TCP/IP). FTP позволяет передавать данные в обоих направлениях как между клиентом и FTP-сервером, так и между двумя удаленными компьютерами.

- G -

G2C, G2G (см. *B2G)*

Gadget (см. *гаджеты*)

GDI (Graphic Device Interface – интерфейс графического устройства)

Часть библиотеки Win API, которая служит для работы с графикой и обычно называется GDI. Ключевым в GDI является понятие контекста устройства (DC – Device Context). Контекст устройства – это специфический объект, хранящий информацию о возможностях устройства, о способе работы с ним и о разрешённой для изменения области.

Gb (гигабит), - 1024 мегабит, т.е. 1,073,741,824 бит.

GB *(*гигабайт) – 1024 мегабайт, т.е. 1,073,741,824 байт.

GPS (Global positioning system – Глобальная спутниковая навигационная система)

Навигационная система на базе спутников, позволяющая с очень высокой точностью определить местоположение на поверхности Земли.

Groupware (Групповое программное обеспечение)

Программное обеспечение, которое позволяет группе пользователей осуществлять сотрудничество по сети в рамках работы над общим проектом. Эта категория ПО охватывает электронную почту, а также средства для совместной разработки документов, планирования и контроля.

GUI (graphical user interface – графический интерфейс пользователя)

Интерфейс взаимодействия пользователя с компьютерной (как правило, операционной) системой, основанной на графических элементах управления, таких как пиктограммы, ярлыки, меню и т.д. Является стандартом "де-факто" в компьютерной отрасли. GUI разработан совместно корпорациями Microsoft и Apple для операционных систем Windows и Macintosh, соответственно. Стал также, стандартом для интерфейсов приложений, разрабатываемых для работы под управлением соответствующих операционных систем.

GUID (Globally Unique ID (identificator)) – глобальный уникальный идентификатор

Уникальное имя, присваиваемое каждому новому программному компоненту (COM–серверу). Представляет собой 128-битное уникальное число. К примеру, может иметь следующий вид: 7D785DE3-07C0-11D0-896C-444553540000.

- H -

Hacker (хакер)

❶ Программист, способный писать программы без предварительной разработки детальных спецификаций и оперативно вносить исправления в работающие программы, не имеющие документации.

❷ Пользователь компьютера, занимающийся поиском незаконных способов получить доступ к защищённым данным.

Hardware (аппаратные средства компьютера)

Материальная часть вычислительной системы (компьютера), включающая электрические, электронные и электромеханические элементы, включая стойки и корпуса.

Hosting (Хостинг) (см. *Хостинг*)

Размещение и поддержание интернет-сайта и необходимых приложений на сервере Интернет-провайдера. Сдача в аренду аппаратно-программного обеспечения.

HDD (Hard Disc Drive)

Устройство записи, считывания и хранения данных на так называемом жёстком диске.

HTML (Hypertext Markup Language).

Язык разметки, предназначенный для представления содержимого Web-страниц в Internet.

HTTP (Hypertext Transport Protocol)

Протокол, используемый Web-браузером для взаимодействия с Web-серверами.

HyperCube (Гиперкуб, многомерный куб)

Структура данных, хранящая многомерную информацию и имеющая по одному ребру для каждой возможной комбинации размерности.

- I -

IAC - провайдеры (Internet Application Collaboration)

Провайдеры приложений совместной работы.

IBM (International Business Machines)

Название крупнейшей американской фирмы-производителя вычислительной техники. Была создана в 1924 году американским инженером Германом Холлеритом, автором статистического табулятора, построенного им с целью ускорения результатов переписи населения в США в 1890 г. Основной продукт IBM – компьютеры серии IBM/360 и IBM/370.

IBM PC

Персональный компьютер ИБМ. 16-ти разядный компьютер фирмы IBM на базе микропроцессора Intel 8088 и её модификации – IBM PC XT с винчестерским диском, IBM PC AT на базе микропроцессора Intel 80286.

ICE (Internet Content Exchange)

Протокол обмена информационным наполнением в Internet. Элемент программного обеспечения программного продукта BizTalk, разработанного Microsoft.

IDE (Integrated Drive Electronics)

Интегрированная электроника для дисководов. Интерфейс подключения дисководов к портам компьютера.

IEEE (Eye-triple-E, Institute of Electrical and Electronics Engineers, Inc.) Институт инженеров по электротехнике и электронике

Неприбыльная, профессиональная техническая ассоциация 380 000 индивидуальных членов из 150 стран.

IIDL (Interoperable Interfase Definition Language – Интероперабельный Язык Определения Интерфейсов)

Независимый от языков программирования набор стандартов и стандартных интерфейсов, разработанных международной организацией OMG (Object Management Group) для приложений, написанных на C, C++, Java, COBOL, Smalltalk, Ada, Lisp, Python, and IDLscript.

IIOP (Internet Inter Object Request Broker Pritocol)

Компонент стандарта CORBA. Использование стандартного протокола обеспечивает полную интероперабельность взаимодействия между любыми компьютерами, операционным системам, языками программирования и сетевыми структурами.

IIS (Internet Information Server)

❶ Продукт Microsoft, предоставляющий средства для Web-публикаций и передачи файлов, а также поддержки Интернет- и Интранет-приложений. Кроме стандартных HTML-страниц, IIS поддерживает технологию активных серверных страниц (Active Server Pages, ASP) – не зависящую от языка среду выполняющихся на сервере сценариев, позволяющую создавать и выполнять Web-приложения.

❷ IIS является группой серверов Internet (включая серверы: *Web, Hypertext Transfer Protocol, File Transfer Protocol)* с дополнительными возможностями работы с операционными системами Microsoft's Windows NT и Windows 2000 Server. IIS представляет собой программный продукт, разработанный корпорацией Microsoft для пополнения линейки Web-серверов: Apache, Sun Microsystems, O'Reilly и др. Разработчики, использующие IIS,

могут в своей работе применять следующие продукты и технологии корпорации Microsoft: Front Page, Active Server Page (ASP), ActiveX controls, Internet Server Application Program Interface (ISAPI), Common Gateway Interface (CGI).

Input (входные данные)

Данные, передаваемые или вводимые в компьютерную систему для обработки, в противоположность результатам обработки, известным как выходные данные (*output*). Наиболее известными устройствами ввода (*input devices)* в персональном компьютере являются клавиатура (keyboard) и мышь (mouse). Сканнеры становятся также активно применяемыми устройствами при вводе информации, к которым начинают присоединяться технологии распознавания речи.

Intel (Integrated Electronics).

Компания Intel - крупнейший в мире изготовитель микропроцессоров, а также ведущий производитель оборудования для персональных компьютеров, сетевых и коммуникационных продуктов была основана в 1968 г. Робертом Нойсом и Гордоном Муром в калифорнийском городке Маунтин-Вью.

Internal web (внутренняя сеть)

Как правило, web является *неструктурированной клиент/серверной вычислительной сетью*, которая использует для передачи транспортный протокол HTTP. World Wide Web соединяет все узлы HTTP в общедоступный, открытый всем Интернет. Внутренний Web (внутренняя паутина- Internal Web) объединяет все узлы HTTP частной вычислительной сети, такой, как LAN или WAN. Если организация имеет корпоративную структуру (является корпорацией), внутренний Web, является корпоративным Web'ом. Если корпоративный web соединяет двух или более торговых партнёров, его часто называют экстранетом или "*бизнес-2-бизнес*" Web (business-to-business Web) сетью.

Internet (см. Интернет)

Internetworking (межсетевое взаимодействие)

Когда две или более сетей организуют совместную транспортную службу, то такой режим взаимодействия обычно называют *межсетевым взаимодействием (internetworking)*. Для обозначения составной сети в англоязычной литературе часто также используются термины *интерсеть* (*internetwork* или *internet).* Интерсеть обеспечивает только передачу пакетов, не занимаясь их содержанием.

Internet address (Интернет адреса)

Уникальные коды, присваиваемые конкретным компьютерам, подключённым к Интернету для идентификации его в качестве отсылающего и получающего данные и файлы программ. Существуют две категории используемых адресов: адреса электронной почты (*e-mail*) отдельных личностей (к примеру, presleyelvis@aol.com) и *URL* или *FTP* сайты, сайты *Telnet* и *Web* сайты (к примеру, *www.aol.com*). Форма и формат Интернет адресов регулируется службой Системы Доменных Имён (Domain Name System, DNS).

Interoperability (См. интероперабельность)

IP (Internet Protocol – протокол коммутации пакетов)

Протокол сетевого уровня из набора протоколов Internet, определенного в RFC 791. Описывает программную маршрутизацию пакетов и адресацию устройств. Стандарт используется для передачи через сеть базовых блоков данных и дейтаграмм IP. Обеспечивает передачу пакетов без организации соединений и гарантии доставки. Протокол был изначально разработан Министерством Обороны США для объединения в сеть разнородных компьютеров.

IP address (IP-адрес)

Адрес для протокола IP – 32 битовый (4 байта) адрес, определенный в STD 5 (RFC 791) и используемый для представления точек подключения в сети TCP/IP. IP-адрес состоит из номера сети (network portion) и номера хоста (host portion) - такое разделение позволяет сделать маршрутизацию более эффективной. Обычно для записи IP-адресов используют десятичную нотацию с разделением точками (например, 10.12.23). Новая версия протокола

IPv6 использует 128-разрядные адреса, позволяющие решить проблему нехватки адресного пространства.

IPC (Interprocess communication – механизм взаимодействия процессов)

Механизмы, обеспечивающие взаимодействие и обмен данными между процессами. Подобные механизмы просто необходимы для распределённых приложений, поскольку распределённые приложения являются совокупностью нескольких процессов, а зачастую и нескольких систем.

IRDA (InfraRed Data Association)

Объединение данных посредством инфракрасного порта.

ISA (Industry Standard Architecture)

Стандарт подключения старых 8-ми и 16-ти разрядных адаптерных плат.

ISAPI (Internet Server API – интерфейс прикладного программирования для интеграции приложений с Internet Information Server)

Поскольку ISAPI-программы выполняются на сервере, эту технологию можно использовать с большим числом Web-браузеров.

ISDN (Integrated Services Digital Network)

❶ Стандарт цифровой передачи для телекоммуникационных сетей, позволяющий с высокой скоростью передавать по каналам связи голосовые сообщения, видеоинформацию и другие сопутствующие данные одновременно.

❷ Продукт эволюции аналоговой телефонной сети. Обеспечивает передачу информации в цифровой форме на всем протяжении соединения. Данная сеть доступна через стандартизованный набор пользовательских интерфейсов.

ISO (International Organization for Standardization)

Основанная в 1947 г. всемирная организация, которая в международном масштабе осуществляет стандартизацию де–юре технологий и процессов. При участии ISO, организовано более 130 институтов национальных стандартов в разных странах. Главная задача ISO – развитие взаимодействия в сферах интеллектуальной, научной, технологической и экономической деятельности стран всего мира. Только в 2000 г. организацией опубликовано 986 (из общего количества 13 025, начиная с 1947г.) международных и типовых стандартов. На 31 декабря 2000 г. в разработке находились 4789 рабочих проектов стандартов, входящих в тематику так называемых технических комитетов (TC, technical committee).

ISO/TC 211 (ISO/ Technical Committee – Технический комитет ISO)

Международная организация, занимающаяся вопросами стандартизации в сфере геоинформатики. К концу 2003 года было разработано 32 стандарта на обработку и использование пространственной информации (геоинформации).

ISP (Internet Service Provider – провайдер услуг Internet)

Компания или другая организация, предлагающая услуги по подключению к Internet через свои компьютеры (являющиеся частью Internet).

IT (information technology) (см. *информационные технологии*)

- J -

J2EE Platform (Java 2 Platform, Enterprise Edition)

Среда для разработки и развёртывания корпоративных (Enterprise) приложений. Платформа J2EE состоит из набора сервисов, интерфейсов программирования приложений (API) и протоколов, которые обеспечивают функциональность для разработки многоярусных, Web-ориентированных приложений.

JAE (Java Application Environment – среда приложений Java)

Исходный код, подготовленный в интегрированной среде разработки JDK (Java Development Kit).

JAR Files (.jar)

Архив языка Java (Java ARchive). Формат файлов, используемый для объединения многих файлов в один.

Java

❶ Торговая марка фирмы Sun, относящаяся к ряду технологий, предназначенных для создания и безопасного выполнения программ в виде настольных и сетевых приложений.

❷ Объектно-ориентированный язык программирования, используемый в основном в Web-технологиях. Язык Java можно использовать для реализации двух типов программ: приложений и апплетов. Одна из особенностей Java состоит в том, что результатом работы компилятора Java является ***байткод***. Байткод - это оптимизированный набор команд, предназначенных для выполнения виртуальным устройством, которое эмулирует Java-система в процессе выполнения апплета. Байткод, как правило, интерпретируется. Интерпретация - это самый простой способ создания переносимых и безопасных программ. Java является простым, безопасным, переносимым, объектно-ориентированным, устойчивым к ошибкам, многопоточным, независимым от архитектуры, интерпретируемым, высокопроизводительным, распределенным и динамичным языком программирования.

❸ Платформа и архитектура (JavaBeans и Enterprise JavaBeans), развиваемые фирмой Sun Microsystem.

❹ Разработанный компанией SunSoft язык программирования, основные свойства которого – независимость от аппаратной платформы.

JavaBeans

❶Портабельная (переносимая), платформонезависимая модель программных компонентов.

❷ Компонентная программная архитектура, разработанная корпорацией Sun Microsystems и функционирующая в среде Java. Компоненты JavaBeans представляют собой независимые программные модули, написанные на Java, которые можно вызывать из других приложений. Архитектура JavaBeans конкурирует с моделью Microsoft COM. Компания Sun недавно представила спецификации серверных компонентов Enterprise JavaBeans, которые будут использоваться в распределенных приложениях. **Java Beans**

❸ Программные компоненты - независимые, повторно используемые программные модули, которые способны взаимодействовать друг с другом. В языке Java компоненты называются Beans (Бинс - Зерна). Специальная программа BeanBox используется для манипуляции компонентами и построения готового приложения.

Java virtual machine (см. *виртуальная машина*)

JPEG (Joint Photographic Experts Group)

Стандарт для сжатого (compressing) цифрового представления графических данных (фотографий и других изображений).

- K -

kb (kilobit – килобит)

1024 бит. Надо отметить, что при подсчете объемов информации для введения высших разрядов вместо привычной тысячи используется $1024=2^{10}$, что иногда порождает путаницу.

kB (kilobyte – килобайт)

1024 байт.

KDD (Knowledge Discovery in Databases)

Обнаружение знаний в базах данных, статистические алгоритмы выявления знаний.

KISS-principle (*Keep It Simple Stupid* – Будь попроще, глупыш…)

Принцип, запрещающий использование более сложных средств, чем необходимо.

KMP (от английского термина Knowledge Management Portal – Портал Управления Знаниями)

Информационно-технологическое решение, использующее технологии корпоративного информационного портала для управления взаимодействием на уровне знаний между сотрудниками организации, рабочими группами и собственно организацией. Кроме того, подобный портал предполагает наличие возможностей для поиска, извлечения и представления знаний.

KMS (knowledge management system – система управления знаниями) (см. *Knowledge Management*).

Knowledge Management (управление знаниями)

Систематический процесс регистрации, извлечения, сохранения и доставки (распространения) знаний во всей организации. Ноу-хау может быть извлечено как из анализа деятельности одного сотрудника, так и деятельности целого коллектива в целях улучшения деятельности всей организации в целом. Для управления вышеуказанными процессами применяются системы управления знаниями. Технологии, активно развиваемые компаниями Lotus, IBM и Xerox в своих программных продуктах.

- L -

L2 cache (Level 2 cache– Кэш 2-го уровня)

Кэш между процессором и подсистемой памяти. Работает, как правило, на частоте шины и смонтирован на материнской плате (хотя в поздних процессорах Intel его начали устанавливать в одной микросборке или модуле с процессором, а также увеличили частоту). Для кэша 2-го уровня практически всегда используется память типа SRAM. Характерные емкости такой памяти – от 256kB до 1MB на процессор. Объем и быстродействие кэша 2-го уровня оказывают значительное воздействие на быстродействие системы в целом. Следует иметь в виду, что иногда установка в систему дополнительной памяти (как правило, свыше 64MB) может заметно замедлить ее работу, если контроллер не поддерживает кэширование этой памяти.

LAN (Local Area Network)

Сеть, соединённых между собой рабочих станций (компьютеров), совместно использующих ресурсы процессора или сервера в пределах относительно небольшого географического пространства. Может обслуживать от нескольких до нескольких тысяч пользователей. Для взаимодействия компонентов и передачи информации используются средства (протоколы и языки) TCP/IP, HTML, XML, SMTP и другие открытые Internet ориентированные стандарты.

LED – Light Emitting Diode (светодиод).

Элементная база технологии некоторых фирм (OČE (Голландия), OKI (Япония)), позволяющая отказаться от сложной системы лазерной развёртки при формировании образа печатаемого листа для последующей лазерной печати. Обычно крепятся внутри печатных устройств в виде линейных конструктивов.

Legacy system (См. *наследуемая система*)

LMS (learning management system – система управления обучением)

Программное обеспечение, автоматизирующее процессы обучения и администрирования в процессах е–Обучения. LMS регистрируют пользователей, управляют размещением новых курсов в каталоге и записью данных, поступающих от пользователей для обработки лицами, ведущими соответствующие курсы. Как правило, LMS создаются для управления курсами многочисленными авторами и провайдерами.

Location Services (Адресные сервисы)

В существующем беспроводном мире мобильные телефоны (cell phones), персональные цифровые ассистенты (PDAs), лэптопы и автомобильные компьютеры становятся «беспроводными позиционными Интернет устройствами». Их позиционность основывается

на получении сигналов от множества взаимосвязанных антенн, встроенных систем GPS (Global Positioning System) или же на других позиционирующих их пространственное расположение технологиях.

Lpi (line per inch - линий на дюйм)

Единицы измерения разрешения печати при выводе на принтер указывается в lpi (line per inch - линий на дюйм). Под линией понимается так называемый полиграфический растр. Его отличие от обычного растра заключается в том, что при печати для воспроизведения оттенков используется прямоугольная матрица из точек, печатаемых принтером. Более светлому оттенку соответствует меньшее количество точек в матрице, более темному - большее количество точек. Размер такой матрицы может изменяться, а вот расстояние между точками матрицы фиксировано и зависит от разрешения принтера. В конечном итоге оказывается, что разрешение принтера, разрешение печати (lpi) и количество оттенков, доступных для воспроизведения, жестко связаны между собой.

- M -

Macintosh

Семейство компьютеров, представленных фирмой Apple в 1984 году для популяризации графического интерфейса пользователя (graphical user interface, GUI), ставшего отправной точкой для остальных фирм-производителей, начавших разработку своих, дружественных для пользователя (user-friendly) графических приложений и операционных систем. Хотя линейка компьютеров фирмы Apple составляет всего 5% от общего рынка настольных компьютеров, тем не менее, они представляют крупнейшие серии компьютеров, не совместимых с IBM-ориентированными ПК. Мак'и ("Macs") продолжают оставаться популярными в издательском деле и в школьных компьютерных классах США, где составляют более 60% от общей массы используемых компьютеров.

Marshalling (маршаллинг, транспортировка)

❶ (в распределённых вычислениях, DCOM) Представляет собой акт передачи данных (параметров функции и возвращаемых значений) за пределы процесса. Она включает в себя упаковку данных, передачу их за пределы процесса и распаковку данных по достижению ими места назначения. Применяется при распределённых вычислениях и совместной работе компонентов в моделях DCOM.

❷ (в обработке Web Сервисов) Процесс конвертирования типов данных исходного языка программирования в формат, удобный для передачи в сети.

Mb (Megabit – мегабит)

1024 килобит, т.е. 1,048,576 бит.

MB (Megabyte – мегабайт)

1024 килобайт, т.е. 1,048,576 байт.

MDAC (Microsoft Data Access Components)

Технологии, обеспечивающие универсальный доступ к данным (Universal Data Access, UDA). К ним относятся ADO (ActiveX Data Objects), RDS (Remote Data Services), OLE DB и ODBC.

MDI (multiple-document interface –многодокументный интерфейс)

Интерфейс приложения, состоящий из одного первичного окна, называемого родительским окном, внутри которого располагаются дочерние окна. По своей сути, любое дочернее окно является первичным окном, однако оно не может выходить за пределы родительского окна, являясь стандартным интерфейсом Windows-приложений, в котором одно главное окно, называемое родительским окном, визуально содержит множество дочерних окон.

MHz (Megahertz – мегагерц)

10^6 герц, т.е. операций в секунду. Единица измерения частоты, характерная для современных компьютеров, таймеры различных подсистем которых имеют частоты от нескольких мегагерц (шина ISA) до нескольких сотен мегагерц (процессоры). Обычно, системные шины компьютеров имеют частоту от нескольких десятков до 100 мегагерц. Вместе с тем, до недавнего времени максимальная, официальная частота для чипсетов Intel составляла 66 мегагерц.

Microsoft Jet Database Engine

Система управления реляционными базами данных, используемая в приложении Microsoft Access, а также в других продуктах Microsoft: Microsoft Office и Visual Basic.

Microsoft SQL Server (система управления реляционными базами данных)

Предназначена для развёртывания масштабируемых приложений баз данных на платформе Windows NT.

Middleware (промежуточное программное обеспечение (ПО), ПО среднего уровня)

❶ Промежуточное программное обеспечение (содействующее процессам обмена информацией между клиентом и сервером).

❷ Связующее ПО (ПО, обеспечивающее прозрачную работу программ в неоднородной сетевой среде). Посредническое обеспечение (программные средства, играющие роль посредника между прикладной программой и сетью).

❸ Слой программного обеспечения, который расположен между операционной системой и средствами управления компьютерными сетями снизу и прикладными системами сверху. В 7-уровневой модели ISO/OSI это находится на 6-7 уровнях (представления и прикладного).

❹ *Middleware* состоит из набора сервисов, которые позволяют многочисленным процессам выполняться на одной или нескольких машинах, взаимодействуя в вычислительной сети, объединяющей гетерогенные платформы. Эта технология эволюционирует с 1990 года в направлении достижения полной интероперабельности приложений, исполняемых на разных платформах и написанных на разных языках программирования. Наиболее известными являются следующие инициативы по созданию работоспособных моделей для разработки и реализации ПО среднего уровня (middleware): Distributed Computing Environment (DCE) (разработка Open Software Foundation), Common Object Request Broker Architecture (CORBA) (разработка Object Management Group) и Component Object Model COM/DCOM (разработка Microsoft).

MIDL (Microsoft Interfase Definition Language – Язык Описания Интерфейсов Микрософт)

Спецификация интерфейсов классов компонентов в распределенных гетерогенных средах.

MMX (MultiMedia eXtension)

Дополнительные возможности, ориентированные на обработку цифрового изображения и звука, анонсированные Intel в процессорах P55C. Включают в себя 57 новых команд, предназначенных для обработки звуковых и видеосигналов; команды могут использоваться в режиме SIMD (Single Instruction, Many Data - одна команда, много данных), когда одной командой одновременно обрабатываться несколько элементов данных.

Modeless (см. *немодальное окно*).

MP3

Формат для сжатия музыкальных файлов, позволяющий пользователям скачивать музыкальные произведения из Интернета.

MPEG (Moving Picture Experts Group)

Стандарт для сжатия цифровых видео изображений.

MSF (Microsoft Solution Framework – Модель решений Майкрософт)

Дисциплина разработки решений (программных продуктов), предоставляющая набор моделей и измеримых вех, которые можно использовать как рекомендации, равно как

и руководство по планированию, проектированию и ведению проектов в сфере информационных технологий.

MSIL (Microsoft Intermediate Language –промежуточный язык Microsoft)

Определяет для полученного от CLR файла набор переносимых между любыми платформами инструкций, независимых от конкретного процессора. По существу, MSIL является «переносимым ассемблером» и воплощает развитие концепции байт-кода Java. В файле скомпилированной .NET-программы кроме MSIL-кода содержится компонент метаданных — с его помощью CLR обеспечивает контроль и безопасность .NET-файлов. Далее CLR-среда, получив на исполнение .NET-программу (универсальный MSIL-код), запускает *JIT-компилятор (Just In Time — В Нужный Момент*), который-то и превращает MSIL во внутренний код, причем компилирует части программного кода по мере необходимости. Создается своеобразный «динамический вариант» исполняемого кода, а на вход процессора подаётся скомпилированная программа, которая выполняется «на лету» — с той же скоростью, что и обыкновенные программы, но на любом процессоре.

MTS (Microsoft Transaction Server)

Программное обеспечение, предоставляющее основанные на компонентной модели услуги компонентной модели услуги промежуточного программного обеспечения, поддерживающие распределённые транзакции. MTS является расширением COM-модели, призванным облегчить разработку, внедрение и сопровождение распределённых приложений, а также объединяет в себе возможности монитора обработки транзакций и брокера запросов объектов. Кроме того, MTS берёт на себя некоторые вопросы безопасности и управления потоками.

- N -

NASDAQ (National Association of Securities Dealers Automated Quotation)

Американская электронная биржа для торговли акциями высокотехнологичных компаний. Nasdaq была создана в 1971 году с более мягкими, по сравнению с классическими фондовыми биржами (NYSE и др.), условиями прохождения листинга (свод правил и условий, которые необходимо выполнять компании, для того чтобы её акции были допущены к торгам в системе, т.е. на бирже). В силу более либеральных правил прохождения листинга, практически все новые компании, желающие провести публичное размещение своих акций, делают это в рамках торговой системы Nasdaq. А поскольку большинство новичков последнего времени относились к высокотехнологичному сектору, эта торговая система и одноименный индекс NASDAQ, рассчитываемый по результатам торгов, стали ассоциироваться с состоянием дел в интернет-экономике.

Newbie

Новичок (новый пользователь сети), "чайник"

Northbridge (хаб северный мост) (см. *Southbridge*)

Среди производителей чипсетов так обозначается схема системного контроллера, включающая обычно контроллер системной шины, шин AGP и PCI, памяти и кэш-памяти. Обычно название чипсета соответствует обозначению этого устройства.

N-tier application (n- ярусное приложение)

Логическое расширение трёхъярусного приложения. *n*-ярусное приложение является распределённым приложением, в котором один или несколько из трёх первоначальных ярусов, разделены на дополнительные ярусы. Это представляет дополнительный уровень абстракции для описания модели приложений.

- O -

OCX

Расширение, которым снабжаются файлы с элементами управления ActiveX (ActiveX Controls) и специальные элементы управления OLE, представляющие OLE-сервер в качестве DLL.

ODBC (Open Database Connectivity)

❶ Стандартный интерфейс доступа к данным, основанный на спецификации SQL Access Group. Представляет собой технологию и спецификацию интерфейса для доступа к базам данных различных форматов и производителей, разработанных корпорацией Microsoft. По сути, это интерфейс API, такой же, как и Windows API, который имеет дело с программированием баз данных. Архитектура ODBC включает в себя четыре компонента:

–приложение (программа пользователя)
–ODBC менеджер
–ODBC драйверы
–источник данных (базы данных, например, Interbase, Oracle и др.).

В настоящее время компания Microsoft предлагает более совершенный стандарт OLE DB.

❷ Популярный стандарт для гарантированного многоплатформенного доступа к информации, располагающейся в базах данных разных производителей, таких как Oracle, PostgreSQL, Sybase, Informix и др.

ODMG (Object Database Management Group)

Стандарт, предназначенный для спецификаций объектных баз данных и объектной модели для Java-ориентированных платформ.

OEM (Original Equipment Manufacturer – основной производитель оборудования).

OEM – это компания, использующая наборы комплектующих и технологий их применения, произведённых другими известными производителями без покупки лицензии или патента, но под своей торговой маркой. К примеру, тайваньской фирмы Acer по технологии и из комплектующих американской компании IBM производила настольные компьютеры, известные под маркой Aptiva. Такая форма сотрудничества называется “OEM-партнерством”. Compaq, компания, выпускающая компьютеры под собственной торговой маркой, использует в качестве компонента процессоры производства корпорации Intel. Осуществляемые Intel поставки процессоров в технической упаковке называются *«OEM-поставками»*, а сам канал сбыта комплектующих сборщикам называется *«OEM-каналом»*.

Off-line (офф-лайн – автономный).

Работа с полученными данными Internet после отключения от Internet.

OGC –Open GIS Consortium (OGC)

Международный промышленный консорциум, объединяющий на 2002 год более 220 компаний, государственных организаций и университетов, участвующих в процессе разработки и согласования доступных общественности спецификаций в области геообработки данных с помощью информационных и геоинформационных технологий. Открытые интерфейсы и протоколы, определяемые Абстрактными Спецификациями (OpenGIS® Specifications) поддерживают интероперабельные решения, которые придают геоинформационность Web–приложениям, беспроводным и геопривязанным сервисам и другим господствующим в ИТ тенденциям. Важнейшим компонентом является предоставление на основе указанных Спецификаций возможности разработчикам технологий создавать сложные пространственно информационные (геоинформационные) приложения и сервисы, доступные и удобные во всех областях применения.

OLAP (Online Analytical Processing) (см. *Оперативная аналитическая обработка*)

OLE (Object Linking and Embedding – связывание и внедрение объектов)

❶ Архитектура, основанная на модели компонентных объектов Microsoft (COM). На её базе построена унифицированная технология системного уровня, которая базируется на объектах и реализует интеграцию приложений, а также предоставляет клиентам набор объектно-ориентированных услуг. Включает набор системных библиотек DLL-файлов, дающих прикладным программам возможность взаимодействовать друг с другом.

❷ Спецификация корпорации Microsoft, устанавливающая правила взаимодействия приложений, участвующих в подготовке и редактировании составных документов. Преимущества технологии OLE в сравнении с простым обменом данными между приложениями, заключается, в первую очередь, в возможности полноценной работы с каждым объектом в составном документе уже после формирования последнего. В частности, можно видоизменить некогда вставленный в документ рисунок, прослушать и внести правки в речевую аннотацию, просмотреть и отредактировать видеоклип и т.д.

❸ Основанный на COM протокол, позволяющий создавать составные документы. С помощью OLE объект, такой как электронная таблица, может быть внедрён или связан с контейнерным приложением, таким как форма Microsoft Access.

OLE DB (Object Linking and Embedding for Database)

Одна из ключевых технологий универсального доступа к данным (Universal Data Access, UDA). Набор стандартных COM-интерфейсов, позволяющих клиентскому приложению с помощью одинаковых методов одновременно работать с разными типами данных. Управление интерфейсами осуществляется на уровне COM-сервера. COM-сервер, поддерживающий OLE DB, называется OLE DB-провайдером. Архитектура OLE DB состоит из провайдера, потребителя и слоя сервисных компонентов между ними. Интерфейс OLE DB является встроенным интерфейсом SQL Server 7.0 корпорации Microsoft, т.е. тем интерфейсом, посредством которого процессор запросов (MS SQL Server Query Processor) общается с механизмом хранения. Основывается на трёхъярусной архитектуре поставщиков данных, необязательных поставщиков услуг и потребителей данных.

OMA (Object Management Architecture – Архитектура Управления Объектом)

Архитектура, построенная OMG на концепции управления объектом. Ее ключевыми составляющими являются:

- **CORBA** (*Common Objects Request Broker Architecture - общая архитектура объектных запросов*) - отвечает за базовые механизмы взаимодействия объектов в сети
- **Object Services** (*Объектные сервисы*) - системные службы для поддержки разработки приложений
- **Common Facilities** (*Универсальные средства*) - поддержка пользовательских приложений
- **Application Objects** (*Объекты приложений*) - собственно прикладные приложения

OMG

Основанная в апреле 1989 года одиннадцатью компаниями, Object Management Group™ (OMG™), является неприбыльной организацией, включающей в 2003 году более 800 организаций-членов. В рамках деятельности корпорации разрабатываются коммерчески перспективные и независимые от производителей спецификации для софтверной индустрии. OMG™ продвигает Архитектуру Ведомую Моделью (Model Driven Architecture ™), в качестве «Архитектуры Выбора для Связанного (коммуникациями) Мира» ("Architecture of Choice for a Connected World" ™) в рамках развиваемых ею стандартных всемирно известных спецификаций: CORBA®, CORBA/IIOP™, UML™, XMI™, MOF™, Object Services, Internet Facilities и Domain Interface.

On-line (он-лайн)

❶ Интерактивный, диалоговый, оперативный (об информации или программе, обрабатываемой или доступной в интерактивном режиме).

❷ Подключенный (о внешнем устройстве, работающем под управлением вычислительной системы).

❸ Сеанс работы в сети, в том числе и в Интернет.

On the fly ("на лету")

По отношению к компьютерным технологиям, термин "*на лету*" описывает действия, которые выполняются или происходят динамично, в отличие от чего-либо статичного. Наиболее распространённые технологии для разработки Web-страниц "*на лету*" применяются на стороне сервера в виде встраиваемых и выполняемых фрагментов кодов программ, а также использования *cookie* (информации, предварительно запоминаемой на жёстком диске клиентского компьютера) или технологии Microsoft *Active Server Page (ASP)*.

Outsourcing (аутсорсинг)

❶ Практика передачи части работ компании другим компаниям субподрядчикам. Привлечение внешних ресурсов для решения собственных проблем (напр., для разработки проекта, использование дорогого программного продукта, располагаемого на сервере поставщика сервисов приложений (ASP) и др.).

❷ Извлекать данные из внешних источников (в отличие от получения данных собственными силами).

❸ Переводить производство из региона с более дорогой рабочей силой в регион с менее дорогой, тем самым, снижая себестоимость.

- P -

P2P (Peer to Peer – равный к равному, пиринговый)

Пиринговые (peer-to-peer) вычисления или обработка данных вызывает и использует ресурсы и сервисы в группе компьютеров путём непосредственного обмена информацией. Эти сервисы и ресурсы могут включать, но не ограничиваются этим, циклы обработки, постоянные запоминающие устройства, информацию и принтеры. Объединение ресурсов в такой среде является более простой моделью по сравнению с моделью и архитектурой клиент/сервер.

Paradigm *(существительное)*

Любой пример или модель.

Pattern (см. *шаблоны проектирования*)

Образец, шаблон, модель; моделировать; схема, структура; образ, изображение.

PDF (portable document format – портабельный формат документа)

Формат файлов, разработанный компанией *Adobe Systems* для предоставления пользователям независимого от используемой платформы (кросс платформенного) просмотра документов в точно таком же виде, как они были созданы: то есть, с шрифтами, изображениями, форматированием и расположением элементов в первоначально выполненном виде.

PCI (Peripheral Component Interconnect)

Стандарт подключения 32-х разрядных адаптерных плат.

PCI Rev. 2.1 (Concurrent PCI)

Новая спецификация шины PCI, пришедшая на смену Rev. 2.0.

Peer-to-peer network (см. *P2P*)

Соединение равноправных узлов локальных вычислительных сетей (ЛВС). Сетевая среда взаимодействия, позволяющая пользователям осуществлять соединения непосредственно между своими компьютерами, минуя централизованные сервера WWW и обмениваться файлами, располагаемыми на их собственных компьютерах. Примером приложения, позволяющего работать в сетях в режиме P2P, является программный продукт *Groove*. В русском языке *Peer-to-peer network* именуются *пиринговыми сетями*.

PIO (Parallel Input/Output)

Параллельный ввод/вывод.

POST (Power-On Self Test)

Процесс определения системой своей конфигурации при загрузке (тестом фактически не является). В принципе, память с серьезными дефектами не будет распознана как таковая уже на этой стадии. Следует иметь в виду, что на результат POST могут повлиять установки BIOS Setup.

Plug-In (Плагин, дополнительный модуль)

Программный код или компонент, предназначенный для расширения возможностей программных систем или программных приложений (обычно основной, вызывающей плагин программе). Также используется на Web-страницах для отображения мультимедийного контента.

PLUG AND PLAY

Спецификация, созданная совместно фирмами Microsoft, Intel, Phoenix Technologies (разработчик BIOS), Compaq и некоторыми другими. Цель её создания состояла в сведении к минимуму проблем, связанных с настройкой и конфигурированием аппаратных средств. Технология PLUG & PLAY обеспечивает независимость подключаемых устройств от конкретной операционной системы и определяет расширения для любой существующей архитектуры IBM-совместимых компьютеров, включая новые BIOS и аппаратные возможности, которые призваны оградить пользователя от проблем с настройкой и конфигурированием. Кроме процесса физического подключения некоторого устройства к системе, интерфейс PLUG & PLAY выполняет все работы по идентификации подключенного устройства и по обеспечению данного устройства необходимыми аппаратными ресурсами (вроде уровня запроса прерывания) и по конфигурированию соответствующих драйверов устройств. Кроме того, интерфейс PLUG & PLAY не зависит от архитектуры системной шины и способен работать с ISA, EISA, MICRO CHANNEL, PCMCIA и любой другой шиной, используемой в персональных компьютерах.

PNG (Portable Network Graphics)

Беспатентный графический формат сжатия изображений, разработанный фирмой *Macromedia*, для замены формата GIF. Формат PNG обеспечивает новые возможности высококачественного отображения графики и, в том числе, 48-битные цвета.

POP (Point of Presence)

Региональный концентратор (точка входа в сеть), используемый провайдером услуг Internet (ISP) для соединения сетей.

PQFP (Plastic Quad Flat Package – плоский прямоугольный пластмассовый корпус с выводами по четырем сторонам)

Корпус микросхем для установки методом поверхностного монтажа.

Program (программа, синоним- Application; см. *приложение*)

Proxy (прокси, функция-заместитель)

Метод локального сервера, вызываемого клиентом. Перехватывается частью кода, которая называется функцией-заместителем. Прокси является представителем сервера и располагается в адресном пространстве клиента. (См. клиент-сервер).

- R -

RAD (rapid application development – быстрая разработка приложений)

Концепция, в рамках которой развивается технология и программная поддержка организации обеспечения быстрой и высококачественной разработки программных продуктов. Концепция включает следующие элементы:

–сбор и накопление требований в рамках проведения конференций и рабочих совещаний;

–прототипирование и раннее, многократное тестирование разрабатываемых для заказчиков программных продуктов;

–повторное использование программных компонентов;

–жёстко выдерживаемое расписание выполнения этапов разработки вместе с постоянным улучшением каждой новой версии продукта.

Софтверные компании предлагают продукты в большей или в меньшей степени удовлетворяющие выше указанным требованиям. RAD обычно опирается на методологию объектно-ориентированного программирования, обеспечивающего повторное использование компонентов. Для наиболее популярных объектно-ориентированных языков программирования C++, Java и Delphi разработаны так называемые среды визуального программирования в виде пакетов программ, называемых средствами быстрой разработки приложений (RAD).

RAID (Redundant Array of Independent Disks – избыточный массив независимых дисков)

Дисковый массив, консолидированная дисковая система для хранения данных большого объема путём использования массивов небольших (3,5- и 5,25-дюймовых) жестких дисков, что позволяет достичь показателей производительности, характерных для одного большого дорогостоящего диска. В массивах RAID значительное число дисков относительно малой емкости используется для хранения крупных объемов данных, а также для обеспечения более высокой надежности и избыточности данных. Подобный массив воспринимается компьютером как единое логическое устройство.

RAM (Random-access memory – Память с произвольной выборкой)

Аналог термина *оперативное запоминающее устройство (ОЗУ)*. Любое устройство памяти, для которого время доступа по случайному адресу равняется времени доступа по последовательным адресам. В этом смысле, термин практически утратил свое значение, так как современные технологии RAM используют методы оптимизации последовательного доступа и существенно ускоряют выборку данных.

RDO (Remote Data Objects)

Технология Microsoft RDO предоставляет высокопроизводительный объектно-ориентированный интерфейс к источникам данных ODBC без использования Microsoft Jet Database Engine. Таким образом, никаких накладных расходов, связанных с Jet, в данной модели нет.

RFP (request for proposal – запрос на предложения)

❶ Документ, разрабатываемый в ИТ-отрасли перед выполнением сложных научно-технологических перспективных разработок. Обычно публикуется в WWW и, после получения и обсуждения всех замечаний и предложений, является основой для выполнения последующих работ.

❷ Документ, разрабатываемый компанией, ищущей товары или услуги и рассылаемые перспективным производителям

Restart

❶ Перезапуск, повторный запуск операционной системы или компьютера.

❷ Перезапускать, возобновлять.

Risk, risc- analysis, risk assessment и др. (см. *риск*)

ROI (return on investment – возвращение вложений)

Обычно, уровень прибыли, получаемый от вложенных инвестиций по отношению к собственно объёму инвестиций. В е–Обучении, ROI, как правило, вычисляется сравнением материальных (реальных) результатов обучения (к примеру, увеличение числа выученных блоков курса или уменьшение уровня ошибок) к стоимости проведенного обучения.

ROM (Read-only memory)

Память только для считывания, постоянная память.

RPC (remote procedure call – удалённый вызов процедур)

❶ Во взаимодействиях программных компонентов, определяемых COM моделями, RPC определяет способ вызова COM–компонентами приложений или объектов, которые

выполняются в других процессах или на других компьютерах. Таким образом, осуществляются распределённые в сетевых средах вычисления.

❷ В DCOM моделях взаимодействия, сообщение, посылаемое по сети, которое позволяет программе, установленной на одном компьютере, инициировать выполнение необходимой операции на другом.

❸ Идея вызова удаленных процедур состоит в расширении хорошо известного и понятного механизма передачи управления и данных внутри программы, выполняющейся на одной машине, на передачу управления и данных через сеть. Средства удаленного вызова процедур предназначены для облегчения организации распределенных вычислений. Наибольшая эффективность использования RPC достигается в тех приложениях, в которых существует интерактивная связь между удаленными компонентами с небольшим временем ответов и относительно малым количеством передаваемых данных. Такие приложения называются RPC-ориентированными.

Характерными чертами вызова локальных процедур являются:

Асимметричность, то есть одна из взаимодействующих сторон является инициатором;

Синхронность, то есть выполнение вызывающей процедуры приостанавливается с момента выдачи запроса и возобновляется только после возврата из вызываемой процедуры.

Существует несколько реализаций процедур удаленного вызова процедур в различных операционных системах.

В операционной системе UNIX используется процедура под одноименным названием *(Remote Procedure Call - RPC).* Данная процедура внедрена в ядро системы. Ее выполнение обеспечивается протоколом RPC.

В операционных системах Windows удаленный вызов процедур начал развиваться на базе механизмов OLE, которые постепенно развились в технологию DCOM (Distributed Component Object Model). Данная технология позволяет создавать достаточно мощные распределенные сетевые вычислительные среды. В технологии используются фирменные протоколы Microsoft.

RTTI (run time type identification – идентификация на этапе выполнения)

Способность объектно-ориентированных языков автоматически определять тип объекта на этапе выполнения программ.

Run time

Время выполнения программы, время счёта.

Run-time

❶ Исполняющая система; модуль исполняющей системы.

❷ Динамический, то есть выполняемый или происходящий во время выполнения программы.

RWM (Read/write memory)

Память для считывания и записи.

- S -

Schema (Схема)

Описывает и ограничивает XML контент (см. *XML-Related Terms and Definitions*).

SCM (Software configuration management)

Управление конфигурацией программного обеспечения.

Scripting (создание сценариев) (См. *Скрипт*)

Использование языка сценариев для доступа к возможностям приложения на уровне программирования. Для этого в приложение встраивается специальная система обработки сценариев (scripting engine), позволяющая использовать определённый язык сценариев,

например VBA, VBScript или JavaScript. Примерами приложений со встроенными сценарными возможностями являются MS Excel, MS Word, MS Internet Explorer, Internet Information Server с активными серверными страницами (Active Server Page) и многие другие.

SCSI (Small Computer System Interface – читается «скази»).

Интерфейс подключения внешних устройств к компьютеру. Важнейшим преимуществом этого интерфейса является то, что можно подключить к ПК до 8-ми периферийных устройств, имея всего один слот подключения (расширения).

SDI (Single Document Interface – однодокументный интерфейс)

Некоторые из приложений операционной системы Windows, например Блокнот (Notepad), позволяют работать одновременно только с одним документом. Чтобы открыть другой документ, нужно закрыть текущий. Приложение, подобно Блокноту использующее одно главное и несколько дополнительных вторичных окон, называется *SDI-приложением (Single Document Interface — однодокументиый интерфейс).* Единственный способ работать одновременно с несколькими объектами в SDI-приложении — открыть несколько экземпляров этого приложения. Главные окна SDI-приложения можно свертывать и развертывать независимо друг от друга. Если делается попытка открыть уже открытый объект, активизируется существующее окно. В Windows 95 чаще всего встречаются именно SDI-приложения, поскольку в операционной системе сделан акцент на понятие документа.

SDRAM (Synchronous Dynamic RAM - динамическая оперативная память)

Память, работающая синхронно с системной шиной. Быстродействие обычно составляет 8, 10 или 12 нс, хотя эти значения нельзя сравнивать с 60, 70 или 80 нс для стандартной памяти DRAM, так как в случае SDRAM это не полное время выборки, а время одного такта. Синхронная DRAM - название синхронной памяти "первого поколения", широко применяющейся в настоящее время и имеющей пропускную способность порядка 100Mb/сек.

SDRAM clock

Часто встречающееся указание на то, что те или иные чипы или модули SDRAM являются 2 clock или 4 clock. Под *clock* здесь понимается линия ввода сигнала таймера.

SFA - Sales Force Automation

Автоматизация продаж.

SGML (Standard Generalized Markup Language)

Обобщённый стандартный язык разметки (см. *XML-Related Terms and Definitions*).

Shareware (условно-бесплатная программа) (см. также *программный продукт*)

Программа, которую вы можете бесплатно использовать в течение ограниченного времени. Если по истечении оговоренного срока вы продолжаете работать с программой, вам надо заплатить за нее.

SIMM (Single In-line Memory Module)

Наиболее распространенный в течение долгого времени форм-фактор для модулей памяти. Представляет собой прямоугольную плату с контактной полосой вдоль одной из сторон, фиксируется в разъеме поворотом с помощью защелок. Контакты с двух сторон платы на деле являются одним и тем же контактом (single). Наиболее распространены 30- и 72-контактные SIMM (ширина шины 8 и 32 бит соответственно).

Socket 4

Разъем для старых модификаций процессора Pentium с питанием 5 В.

Socket 7

Разъем, ставший практически промышленным стандартом де-факто. Впервые был применен Intel для процессоров P54C и P55C. Сегодня ему соответствуют процессоры AMD K5, K6, K6-2, IDT C6, Cyrix 6x86, 6x86L, 6x86MX и другие.

Socket 8

Запатентованный Intel разъем для процессора Pentium Pro.

Slot 1

Запатентованный Intel разъем для процессора Pentium II. Допускает также подключение процессора Pentium Pro с помощью специального адаптера.

Smart card (см. *смарт-карта*)

SMP (Symmetric Multi-Processing)

Метод, позволяющий более чем одному процессору распределять между собой вычислительную нагрузку. Intel Pentium и Pentium II поддерживают такой режим только для двух процессоров, Pentium Pro - для четырех. С использованием дополнительных схемных решений система может содержать и большее количество процессоров, однако это не всегда гарантирует равномерное распределение нагрузки между ними.

SOAP (Simple Object Access Protocol)

Средство обеспечения совместной работы для приложений через Интернет независимо от платформы. SOAP разработан совместно Microsoft, DevelopMentor и Userland Software и представлен организации Internet Engineering Task Force (IETF) для утверждения в качестве стандарта.

Software (см. *программное обеспечение*)

Southbridge (хаб южный мост) (см. *Northbridge*)

Обозначение схемы периферийного контроллера чипсета, включающего обычно контроллеры EIDE, клавиатуры, последовательных портов, шины USB и прочих подобных устройств.

SPD (Serial Presence Detect)

Система идентификации модулей памяти, включающая в себя интерфейс I2C и схему энергонезависимой памяти объемом 512 байт, в которой записаны параметры модуля.

SQL (Structured Query Language — язык структурированных запросов)

Является стандартным языком для работы с реляционными БД. Кроме стандартных реляционных операций, этот язык предоставляет возможности для изменений структуры таблиц БД. Как структурированный язык запросов и непроцедурный язык – ориентирован на операции с данными, представленными в виде логически взаимосвязанных совокупностей таблиц.

SQL-3 (см. также SQL)

Стандартизированный и расширенный ISO и ANSI – вариант SQL для работы с объектно-ориентированными базами данных.

SRAM (Static RAM)

Статическая память (разновидность RAM), единицей хранения информации в которой является состояние "открыто-закрыто" в транзисторной сборке. Используется преимущественно в качестве кэш-памяти 2-го уровня. Ячейка SRAM более сложна по сравнению с ячейкой DRAM, поэтому более высокое быстродействие SRAM компенсируется высокой ценой. Несмотря на низкое энергопотребление, является энергозависимой.

Star Schema (Схема "звезда")

Метод организации информации в хранилище данных, позволяющий рассматривать информацию во многих перспективах (процесс проектирования, который включает для каждой таблицы фактов одну или более таблиц размерности).

STEP (Standard for Exchange of Product Data – Стандарт для Обмена Данными о Продукции)

Международный стандарт ISO 10303. Стандарт описывает единую методологию, концептуальную и логические модели, а также форматы данных, используемых для построения модели изделия.

System memory (оперативная память)

Память (в подавляющем большинстве случаев – DRAM), использующаяся для хранения активных программ и данных. Количество и быстродействие оперативной памяти оказывают чрезвычайно серьезное воздействие на быстродействие современных компьютеров. Работает на частоте системной шины. Доступ процессора к оперативной

памяти происходит через кэш 2-го уровня. Некоторые подсистемы компьютера способны обращаться к оперативной памяти напрямую, минуя процессор.

- T -

TCO (Total cost of ownership – Совокупная стоимость владения)

Стоимость покупки, эксплуатации и техобслуживания компьютерной системы. TCO включает стоимость приобретения аппаратного и программного обеспечения плюс затраты на его установку, обучение персонала, поддержку, модернизацию и ремонт. В отрасли используются следующие средства снижения TCO: централизованное администрирование компьютеров и сетей, автоматизированное обновление и "самоисцеление" программного обеспечения.

TCP/IP (Transmission Control Protocol/Internet Protocol – Протокол управления передачей/межсетевой протокол)

❶ Протокол, обеспечивающий передачу данных по Internet. Применяется по умолчанию системами UNIX для маршрутизации пакетов информации в локальной или глобальной сети. Это стандартный протокол, на котором основана система передачи данных в Internet.

❷ Протокол управления передачей/протокол Internet - набор коммуникационных протоколов, используемый большинством главных компьютеров для обмена информацией. Метод передачи данных с коммутацией пакетов, который используется в сети Интернет. Протоколом определяется разделение сигнала на пакеты, а также добавление к каждому пакету адресной информации, необходимой для того, чтобы пакет достиг адресата и оригинальное сообщение было восстановлено.

TFT

Thin Film Technology *(тонкопленочная технология)*.

Topic Map

Навигация по XML контенту и Web-ресурсам в Интернете (см. *XML-Related Terms and Definitions*).

- U -

UDA (Universal Data Access – универсальный доступ к данным)

Стратегия Microsoft по предоставлению унифицированных методов доступа к данным, не зависящих от их типа и местоположения.

UDDI (Universal Description, Discovery and Integration)

Ориентированная на Web адресная книга, позволяющая бизнес –организациям самостоятельно размещать о себе информацию в Интернет и отыскивать там друг друга самостоятельно, как в традиционной телефонной книге.

UIU (Intellectual User Interface – Интеллектуальный Пользовательский Интерфейс) (См. *интерфейс интеллектуальный*)

Unicode (юникод)

Всемирный стандарт на кодирование символов. Кодировка символов 16-разрядными двоичными числами, в результате использования которой удаётся представить 65 536 различных знаков и символов. Это вполне достаточно для одновременного представления всех букв основных языков любой страны мира, где используются компьютеры, а также всевозможных небуквенных специальных символов.

UML (Unified Modeling Language – Унифицированный язык моделирования)

Единый язык объектно-ориентированного анализа и моделирования, предназначенный для спецификации, визуализации, конструирования и документирования отчуждаемых материалов программных систем, равно как и для моделирования бизнеса и

других не программных систем. UML включает в себя в унифицированном виде наилучшие практические методы графического моделирования, известные в настоящее время.

UNIX

Операционная система, разработанная в исследовательской организации Bell Labs в 1969 году. UNIX поддерживает многопользовательский и многозадачный режимы работы и имеет большое количество разнообразных версий (ОС HP/UX, Red Hat, Linux, IBM AIX, Solaris компании Sun Microsystems, SCO UNIX и др.). Они предназначены для функционирования на множестве разных платформ и популярны в научных и исследовательских организациях. Обычно устанавливаются на серверах, ввиду высокой надёжности работы и устойчивости против компьютерных вирусов.

UPS (Uninterruptible Power Supply)

Источник бесперебойного питания.

URI (uniform resource identifier)

Имя и адрес информации, представленной текстом, графикой, аудио, видео и другими данными в Internet'е. Обычно URI идентифицирует и приложение, используемое для доступа к ресурсу, располагаемому по указанному адресу, а также имя файла ресурса. Адрес Web-страницы или URL является наиболее часто используемым типом URI.

URL (Uniform Resource Locator) (произносится–ЮэРэЛ)

Является адресом файла (ресурса), доступного в Интернет. Тип ресурса зависит от протокола доступа к приложению в Интернет. Если используется World Wide Web's протокол, называемый *Hypertext Transfer Protocol* (HTTP), ресурс может быть страницей, написанной на языке HTML, файлом изображения, программой на скриптовом языке (PHP или др.) или любым другим файлом, поддерживающим HTTP. URL содержит имя протокола, требуемого для получения ресурса. Таким образом, *domain name* (имя домена) специфицирует конкретный компьютер в Интернет и иерархическое описание местоположение файла на этом компьютере. Пример URL-имени: http://www.mhrcc.org/kingston. Это имя описывает Web-страницу, доступную через протокол HTTP с помощью приложения, называемого Web-браузером, и расположенного на компьютере с именем www.mhrcc.org . Искомый файл располагается в директории с именем /kingston и является стартовой страницей в директории. HTTP URL может существовать для любой Web-страницы и не только домашней страницы или файла. URL для программы, создающей скрипт управления формами на листе (common gateway interface - CGI) и написанной на языке **Perl** может выглядеть следующим образом: http://whatis.com/cgi-bin/comments.pl

URL для файла, который должен быть загружен на компьютер-клиент посредством функции даунлоад (download), требует задания протокола "ftp" такого типа: ftp://www.somecompany.com/whitepapers/widgets.ps

USB (Universal Serial Bus)

Последовательный универсальный интерфейс для подключения внешних устройств, обеспечивающий скорость передачи данных 12 Мбит/с. Предназначен для замены RS-232 и низкоскоростного SCSI-интерфейса.

- V -

VBA (Visual Basic for Applications – Visual Basic для приложений)

Среда разработки и скриптовый язык программирования, аналогичный Visual Basic, встроенные в приложение. VBA имеется, например, в приложениях: Microsoft Word, Microsoft Excel и Microsoft Access, а также в геоинформационных продуктах фирмы ESRI, известных под общим наименованием ArcGIS.

VBScript (Visual Basic Script)

Подмножество языка Visual Basic, используемое в качестве языка сценариев для встраивания в Web-страницы. Эти сценарии могут выполняться как на компьютере клиенте, так и на сервере.

Visual Basic (VB, Visual Beginner's All-purpose Symbolic Instruction Code)

VB является и интерпретатором, и компилятором. Как интерпретатор, Visual Basic позволяет запускать приложения непосредственно в среде разработки (команда Run \ Start). Как компилятор, Visual Basic предоставляет возможность создавать независимые от среды разработки исполняемые EXE-файлы (команда File \ Make *имя_файла_проекта*…). Исторически различные приложения Microsoft включали различные языки макросов, значительно отличающиеся друг от друга (WordBasic, ExcelMacro, AccessBasic и т.д.). Начиная с Office 97, корпорация Microsoft стала включать в свои приложения общий язык макросов – VBA (Visual Basic for Applications).

VME (Virtual Mode Extension)

Расширение виртуального режима, т.е. набор аппаратных возможностей процессора, позволяющий оптимизировать обработку прерываний в режиме V86 (в частности - обрабатывать программные прерывания внутри VM-задачи, без переключения в режим ядра и виртуализовать флаг IF, отвечающий за разрешение/запрет внешних прерываний.

- W -

W3C (World Wide Web Consortium)

Международная неприбыльная организация, инициирующая и проводящая работы по разработке и внедрению интероперабельных спецификаций, программного обеспечения и инструментальных средств для WWW.

WAN (Wide Area Network) – территориально распределённая сеть

❶ Физическая коммуникационная сеть, связывающая географически удалённые друг от друга компьютеры и сетевые сегменты (LAN). При этом включает все средства передачи. Характеризует более широкую телекоммуникационную структуру, чем LAN. Может состоять из сетей частных компаний, а также включать государственные сети. Обязательно включает все средства передачи. Другими словами, компьютерная сеть, покрывающая достаточно большое территориальное пространство. Обычно строится на двух или более локальных вычислительных сетях (LAN). Примером WAN является Internet.

❷ Распределенная или глобальная сеть, обеспечивающая передачу информации на значительные расстояния с использованием коммутируемых и выделенных линий. Сеть WAN связывает офисы компаний или филиалы компании, находящиеся в разных городах или странах.

WAP (Wireless Application Protocol – протокол беспроводных приложений)

❶ Спецификация создания устройств и программной их поддержки для чтения контента из Internet без непосредственного подключения к нему, то есть в беспроводном режиме.

❷ Стандарт взаимодействия мобильных телефонов и других беспроводных устройств с сетями Интернет/Интранет для получения информации и услуг.

WAV (WAVeform-auto, аудиоинформация в волновой форме)

Категория звукового файла, который, подобно аудио компакт-диску, хранит непосредственные результаты преобразования звука из аналоговой в цифровую форму. Само преобразование выполняется звуковой платой компьютера (мультимедиа). Звуковые WAV-файлы имеют расширение WAV и различаются форматом хранения оцифрованного звука.

Web (паутина, мировая паутина)

Термин, используемый в качестве синонима WWW.

Web server (Web-сервер)

Программное обеспечение, предоставляющее сервисы для доступа в Интернет, интранет и экстранет. Web-сервер управляет работой Web-сайтов, обеспечивает поддержку протокола HTTP и других протоколов и выполняет серверные программы (такие, как

скрипты CGI или сервлеты), для обеспечения разных функций. В архитектуре J2EE Web-сервер обеспечивает сервисы Web-контейнерам.

Web services (Web-сервисы)

❶ Существует много вещей, которые могут быть названы "Web services" (Web-сервисами) в мире окружающих нас вещей. Тем не менее, рабочая группа W3C, основываясь на существующей архитектуре Web-сервисов, остановилась на следующем определении. ***Web-сервис является программной системой***, разработанной для поддержки интероперабельности межмашинного взаимодействия в компьютерной сети. Он имеет интерфейс, описанный в машинно-обрабатываемом формате (как правило, WSDL). Другие системы взаимодействуют с Web-сервисом заданным способом, соответствующим описанию, использующему SOAP-сообщение, типично передаваемым с использованием протокола HTTP с XML сериализацией в привязке к другим Web-ориентированным стандартам. Web Сервисы являются системами, базирующимися на информации, представляемой и манипулируемой с применением XML технологий, с использованием Интернета для непосредственного взаимодействия между приложениями (application-to-application). Эти системы могут включать программы, объекты, сообщения или документы. Web Сервисы обеспечивают независимый от данных механизм (data-independent mechanism) программной обработки бизнес сервисов в Интернете, с использованием стандартных XML протоколов и форматов. Доступ к Web Сервисам может обеспечиваться на уровне браузеров, но это требование не является обязательным и не требует применения HTML.

❷ Основой сервис-ориентированного Web является Web-сервис — набор логически связанных функций, которые могут быть программно вызваны через Internet. Информация о том, какие функции предоставляет данный Web-сервис, содержится в документе WSDL (Web Service Description Language), а для поиска существующих Web-сервисов предполагается использование специальных реестров, совместимых со спецификацией UDDI (Universal Description, Discovery and Integration).

❸ Web-сервисы иногда именуются сервисами приложений (*application services*). Сервисы (обычно включающие некоторую комбинацию программ и данных, а также человеческие ресурсы), создают возможность и условия для использования бизнес-ориентированных Web-серверов (business's Web server) Web пользователями (Web users) или другими Web-ориентированными программами. Поставщики Web-сервисов обычно называются *поставщиками сервисов приложений* (application service providers, ASP). Web-сервисы разделяются на такие главные сервисы, как управление хранением и *управление связью с покупателями* (customer relationship management, CRM) или проведение электронных аукционов. Ускорение создания новых таких приложений составляет главнейшее направление развития Web. Пользователи могут использовать некоторые Web-сервисы посредством пиринговых подключений (peer-to-peer – соединение равноправных узлов локальных вычислительных сетей (ЛВС)), вместо обращений к центральному серверу. Некоторые сервисы могут взаимодействовать с другими сервисами для обмена процедурами и данными, поддерживаемого классом программного обеспечения, называемом *middleware*. В последнее время термин "*Web services*" описывает стандартный способ интеграции Web-размещаемых (Web-based) приложений с использованием открытых стандартов *XML, SOAP, WSDL* и *UDDI*. XML используется для организации и использования данных. SOAP служит для передачи данных сетях, WSDL применяется для описания доступных данных, а UDDI используется для перечисления доступных сервисов.

❹ Web-сервисы являются новой Интернент-парадигмой, независимой от платформ и языков программирования. Web-сервисы являются автономными, модульными приложениями, которые могут быть описаны, опубликованы, размещены и быть вызваны через электронную вычислительную сеть для создания новых продуктов и сервисов.

Web site (см. *Веб-сайт*)

Web-провайдер

Организация, организующая поставки услуг Интернета для пользователей. Синоним –ISP (см. *ISP*).

Web-сервер (Web server)

❶ Компьютер, на котором хранится Web-узел и который делает его доступным пользователям Internet.

❷ Программное обеспечение, предоставляющее сервисы для доступа в Интернет, интранет и экстранет. Web-сервер управляет работой Web-сайтов, обеспечивает поддержку протокола HTTP и других протоколов и выполняет серверные программы (такие, как скрипты CGI или сервлеты), для обеспечения разных функций. В архитектуре J2EE Web-сервер обеспечивает сервисы Web-контейнерам.

Web-Сервисы (см. *Web services*)

Web-страница (Web page)

Отдельно взятый документ Всемирной Паутины (WWW). Представляет собой HTML-документ вместе с файлами, на которые из него есть ссылки. Как правило, текстовый файл с расширением .html.

Web-технологии

Средства создания, размещения и пересылки информации в World Wide Web в разных форматах. Предполагают использование скриптовых языков программирования и технологий работы на стороне клиента и на стороне сервера. К Web-технологиям в последнее время относят следующие элементы:

–именование Web ресурсов и их компонентов (Naming);
–распределённые вычисления в Web (Distributed computation);
–безопасность и перевод денежных средств в Web (Security and money);
–функциональное программирование (functional programming);
–Интернет технологии и организации (Internet technology and organizations);
–Web технологии: HTML, HTTP, WAIS (Wide Area Information Server);
–сценарные языки программирования на клиентской и серверной частях;
–технологии организации контента в Web (ASP (Active Server Objects), JSP);
–технологии анимации (Flash, MetaStream);
–базы данных на Web серверах и Web узлах (MySQL и др.);
–Web участники (компании, организации и люди)

Web-узел (синоним: Web site)

Совокупность взаимосвязанных Web-страниц. Синоним – Web-сайт, или просто сайт. Вместе с тем, Web-узел может означать структуру, существенно расширенную за счёт применения в нём баз данных и элементов сценарных технологий, располагаемых на Web-сервере, где также размещён авторский Web-узел.

White paper (дословно: "белая книга")

Термин, применяемый в сфере науки и техники для характеристики авторитетных докладов или реферативных изданий, как правило, описывающих технологические особенности и преимущества новых и перспективных разработок (программ, товаров, изделий и др.). Обычно размещаются на Web-сайтах для ознакомления он-лайн. Готовятся и публикуются работниками исследовательских организаций или фирм производителей, либо независимыми консультантами.

WIMP (Windows-Icons-Menus-Pointing device)

Тип интерфейса, используемый традиционно в оконно-ориентированных операционных системах и приложениях.

Windows

Дружественная пользователю операционная система, разработанная в 1985 году корпорацией Microsoft для персональных компьютеров (personal computers, PC), сначала устанавливаемая, как надстройка дисковой операционной системы DOS. На начальных этапах Windows эмулировала графический интерфейс пользователя (graphical user interface,

GUI), разработанный фирмой Apple и ставший с тех пор индустриальным стандартом для настольных (desktop) компьютеров.

Windows CE

Упрощенная версия операционной системы Windows, предназначенная для карманных ПК, других цифровых компаньонов и встроенных систем.

Wizard (Мастер, Помощник)

❶ Последовательность страниц, отображаемых во вторичном окне приложения, помогающих пользователю в выполнении конкретной задачи. Эти страницы, как правило, запрашивают у пользователя всю информацию, необходимую для выполнения данной задачи.

❷ Контекстно-чувствительное (context-sensitive) либо открываемое по команде окно диалога, которое автоматически появляется в некоторых компьютерных приложениях для помощи пользователю в необходимых случаях в особых местах программы либо при вызове из различных разделов меню программы. Помощник может быть отключён, если его помощь оказывается назойливой или не нужной. Примером наиболее часто используемого помощника является *Мастер диаграмм*, вызываемый в приложении MS Excel либо щелчком мыши по кнопке стандартных инструментов *Мастер диаграмм*, либо из главного меню выполнением последовательности команд Вставка/Диаграмма.

Workflow (последовательность выполняемых действий, поток работ)

Способ осуществления передачи работы от одного сотрудника организации – другому, либо от одного отдела – другому в компании или организации, обеспечивающий общий ход выполнения запланированных работ. Эффективность выполнения таких работ может быть повышена в результате систематического анализа потоков работ всей организации.

World Wide Web (или просто *Web*), сервис-ориентированный (см. *WWW*)

Новая ***модель Web–сервисов***, по которой Web состоит из набора серверов приложений, обменивающихся информацией в формате XML по протоколу SOAP. Основой ***сервис-ориентированного Web*** является Web-сервис — набор логически связанных функций, которые могут быть программно вызваны через Internet. Информация о том, какие функции предоставляет данный Web-сервис, содержится в документе WSDL (Web Service Description Language), а для поиска существующих Web-сервисов предполагается использование специальных реестров, совместимых со спецификацией UDDI (Universal Description, Discovery and Integration).

Wrapper (обёртка)

Объект, который инкапсулирует и делегирует некоторым образом другому объекту изменение его интерфейса или поведения.

WSCI (Web Service Choreography Interface)

Интерфейс предназначен для "увязки" событий и транзакций при взаимодействии различных систем и приложений в распределенной вычислительной среде.

WSDL (Web Services Description Language)

Язык описания Web-сервисов. XML-форматируемый язык, который используется для описания возможностей Web-сервисов, как коллекции конечных точек коммуникаций, способных обмениваться сообщениями. WSDL является интегральной частью UDDI (всемирной службы регистрации участников бизнес-процессов), базирующейся на технологии XML. UDDI использует язык WSDL, который был разработан совместно Microsoft и IBM.

WWW (синонимы: см. *World Wide Web, Web, см. Всемирная паутина*)

❶ Раздел Internet, образуемый всей совокупностью гипертекстовых (HTML) документов, размещённых на Web-серверах. Логически делится на множество Web-узлов и порталов. Для доступа к размещаемым на серверах документам используется протокол HTTP. Следует особо отметить, что WWW не является синонимом Internet.

❷ Одна из услуг Интернета, позволяющая публиковать информацию в сети. Использует протокол HTTP (Hyper Text Transfer Protocol, протокол передачи гипертекста). WWW-

информация обычно представляет собой гипертекст, создаваемый с помощью языка HTML (Hypertext Markup Language, язык разметки гипертекста). WWW позволяет создавать приложения, доступ к которым может получить любой пользователь, имеющий выход в сеть.

- X -

XML (Extensible Markup Language).

Язык и технология для описания принципов работы с любыми видами данных. Спецификация, разработанная организацией W3C. XML является упрощённой версией языка SGML, разработанного специально для создания и размещения в Интернете Web документов и Web контента. Язык XML позволяет разработчикам создавать свои собственные пользовательские тэги, реализующие определения, передачу, подтверждение правильности и соответствующую интерпретацию данных, циркулирующих между приложениями и между организациями.

XML-Related Terms and Definitions (XML-ориентированные термины и определения)

DTD: Document Type Definition (Определение типа документа).

DOM: Document Object Model (Объектная модель документа).

Schema: Описывает и ограничивает XML контент.

XSD: XML Schema Definition (Определение XML схем).

XSL: Extensible Style Sheet Language (Расширяемый язык стилей листов).

XSLT: Extensible Style Sheet Language Transformation (Преобразование расширяемого языка стилей листов).

XPath: Синтаксис, используемый для поиска элементов в XML.

SGML: Standard Generalized Markup Language (Обобщённый стандартный язык разметки).

eBXML: Electronic Business Extensible Markup Language (Язык разметки электронного бизнеса).

BPML: Business Process Markup Language (Язык разметки бизнес процессов).

BPEL: Business Process Execution Language (Язык выполнения бизнес процессов).

Topic Map: навигация по XML контенту и Web-ресурсам.

- A -

Аббревиатура (abbreviation)

Укороченная форма слова или фразы, используемая для сокращения места, занимаемого текстом при печати или упрощения произношения. Как правило, состоит из первых букв или первых нескольких букв, завершающихся точкой. Например, **assoc**. для слова *association*, **P.O.** для фразы *post office*. Некоторые термины могут иметь более чем одну аббревиатуру: **v.** или **vol.** для *volume (том книги или том жёсткого диска)*. В более простых случаях аббревиатура может состоять просто из первых букв фразы (*WWW – World Wide Web*), либо начальных или конечных фрагментов слов фразы: *системный администратор – сисадмин* (англ. *sysadmin*), *bit – binary digit*.

Абстрагирование

Процесс обобщения, при котором внимание сосредотачивается на сходстве объектов.

Абстрактная машина

❶ Представление о вычислительной машине в терминах информационных ресурсов и операций, доступных программе. Эти ресурсы и операции могут соответствовать реальным или имитироваться операционной средой. Абстрактная машина может не учитывать некоторые возможности реального компьютера, возможно определение абстрактной машины без её реального воплощения для описания семантики языка или доказательства свойств программ.

❷ Абстрактная спецификация для вычислительного устройства, которое может быть реализовано разнообразными способами, как программно (software), так и аппаратно (hardware). Компиляция набора инструкций (команд) на виртуальной машине, производится точно так же, как компилировался бы набор инструкций в микропроцессоре. Виртуальная машина Java (Java virtual machine) состоит из набора инструкций байткода, набора регистров, стэка, динамической сборки мусора и области для сохранения методов.

Абстрактное представление данных (data abstraction)

❶ Использование при работе с объектами только определенных над ними операций, без учета их внутреннего представления.

❷ Методология программирования, при которой программа описывается как совокупность абстрактных типов данных. Абстракция данных обеспечивает большую модульность, чем процедурная абстракция.

❸ Принцип определения типа данных (data type), через операции, которые могут выполняться над объектами данного типа. При этом вводится следующее ограничение**:** значения таких объектов могут модифицироваться и наблюдаться *только путем использования этих операций*. Такое применение общего принципа абстрагирования (abstraction) приводит к понятию абстрактного типа данных (abstract data type).

> Абстрактное представление данных имеет очень большую важность в современном программировании, особенно при грубом структурировании программ. Использование такого представления дает целый ряд преимуществ, в частности, возможность использовать естественные единицы для описания и верификации данных (module specification). Оно обеспечивает основу для высокоуровневого проектирования и хорошо согласуется с принципами утаивания информации (information hiding).
>
> Описание типа данных через имеющиеся операции предоставляет всю необходимую для использования этого типа данных информацию, в то же самое время обеспечивая максимальную свободу реализации. Это означает, что в случае необходимости способ реализации можно изменить прозрачно для пользователей. Кроме того, появляется возможность создания «библиотеки» полезных абстракций данных**:** стеков, очередей и т.д.

Типичная реализация абстрактного типа данных в программе – это реализация с помощью многопроцедурного модуля, реализуемого в языках объектно-ориентированного программирования (ООП) в виде компонентов (объектов) некоторых классов. Такой модуль имеет локальные данные, которые могут использоваться для представления значения данного типа, а каждая процедура, именуемая методом, реализует одну из операций, ассоциированных с этим типом.

Доступ к локальным данным модуля может осуществляться только со стороны этих процедур, так что пользователь этого типа данных может производить доступ только к операциям и не имеет прямого доступа к представлению. Программист, таким образом, имеет полную свободу действий при выборе представления, которое остается прозрачным (потайным – hiding) для пользователей и может при необходимости быть изменено. Для представления значения каждого абстрактного типа данных используется определенная часть локальных данных модуля.

Для обеспечения нормального функционирования таких многопроцедурных (объектных) модулей требуется, чтобы принципы абстрактного представления ***были заложены в самом языке программирования. Такой язык программирования называется объектно-ориентированным языком (ООЯ).*** Соответственно, такой язык должен допускать организацию модулей в виде кластеров и иметь определенные правила видимости, отражающие необходимые ограничения на доступ.

Первым языком, позволившим работать с абстрактными типами данных, стал язык SIMULA, в котором была реализована концепция *класса.* В настоящий момент наиболее развитыми ООЯ являются следующие языки: SmallTalk, Object Pascal, C++, Java, C# и некоторые другие.

Абстрактные спецификации OGC (The OpenGIS® Abstract Specification)

Постоянно редактируемые документы, в которых изменения и дополнения производятся по итогам каждой Встречи Технического Комитета OGC (OGC Technical Committee Meeting). Формально, только члены OGC могут вносить какие-либо предложения и изменения. OGC публикует очередные версии Абстрактных Спецификаций тогда, когда Рабочая Группа Технического Комитета OGC выпускает Плановые Запросы для проектирования спецификаций (Request for Proposals (RFP)), которые реализуют часть соответствующей Абстрактной Спецификации для конкретных распределённых вычислительных платформ (distributed computing platforms). В большей части Абстрактных Спецификаций с применением терминологии UML формулируются принципы реализации элементов *геоинформационных задач* в структуре информационных систем и технологий.

Абстракция

❶ Принцип игнорирования второстепенных аспектов предмета с целью выделения главных.

❷ Один из моментов процесса познания, заключающийся в мысленном отвлечении от ряда несущественных свойств, связей предмета и выделении основных, общих его свойств, связей и отношений. Результатом абстракций являются понятия, категории и др. (напр. материя, движение, развитие и т.п.).

❸ Важная характеристика сущности, отличающая её, от всех иных сущностей. (UML).

❹ Абстракция (при абстрагировании) выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов и, таким образом, четко определяет его концептуальные границы с точки зрения наблюдателя.

Абстракция данных (data abstraction)

❶ Использование при работе с объектами только определенных над ними операций, без учета их внутреннего представления.

❷ Методология программирования, при которой программа описывается как совокупность абстрактных типов данных. Абстракция данных обеспечивает большую модульность, чем процедурная абстракция

Автоматизированная система (computer-aided system)

Комплекс технических и программных средств, выполняющий определённые функции в автоматическом режиме.

Авторизация (Authorization)

Процесс, при котором эмитировавший платежную карту банк подтверждает транзакцию держателя карты путем выдачи кода авторизации в ответе на запрос приложения торгующей организации, в которой держатель карты осуществляет покупку.

Автоформализация знаний

Процесс формализации знаний специалиста в виде программы для компьютера.

Для обеспечения автоформализации знаний требуются специальные методы и инструменты (например, персональные компьютеры). Понятие введено Г. Р. Громовым и является очень важным для определения роли компьютера в современном обществе.

Агент (другие варианты термина: Droid, Intelligent Agent (Robot), Knowbots,)

❶ Устройство и/или программа, установленные в элементах компьютерной сети для централизованного управления этими элементами и всей сетью. Является частью системы сетевого управления. *Аппаратные агенты* – встроенная аппаратура со своим процессором и памятью, в которой хранятся программы управления – *программные агенты*. Программные агенты могут существовать как вместе с аппаратными, так и без них. Обычно представляют собой резидентную программу, выполняющую задачи по сбору статистики и передаче ее в стандартную информационную базу устройства (элемента сети). В этой базе хранятся все управляемые параметры и ресурсы устройства.

❷ Программа, действующая от лица другого субъекта, сущности или процесса. (W3C).

❸ Программный модуль с элементами искусственного интеллекта, функционирующий в фоновом режиме и осуществляющий автоматический поиск информации по предварительным запросам пользователя. Например, агент может использоваться на электронной бирже для он-лайн мониторинга цен и условий от имени продавца или покупателя, а в некоторых случаях и для заключения сделок.

Агрегат данных (Aggregate Data)

Данные, являющиеся результатом объединения элементов данных. Данные, предоставляемые в совокупности или в форме единого результата суммирования.

Ада, язык программирования (см. *Ada*)

Адаптер

❶ Переходное устройство, то есть устройство сопряжения компьютера с другим внешним устройством.

❷ Устройство для соединения устройств с разным способом представления данных либо использующих различные виды сопряжения.

❸ Устройства обеспечения соединения информационных каналов с разными интерфейсами.

❹ Устройство сопряжения *центрального процессора* и *периферийных устройств компьютера*; кроме этого, иногда осуществляет функции управления периферийным устройством. Обычно выполняется в виде микросхемы и помещается на *материнскую плату*, а также может быть представлен отдельной платой. Некоторые источники называют его картой или контроллером.

Адекватный

Равный, соответствующий, тождественный чему-либо.

Адрес

❶ Число, код или идентификатор, специфицирующие регистр, ячейку памяти, область запоминающего устройства, внешнее устройство или узел сети.

❷ Часть команды, указывающая операнд.

❸ Часть сообщения, указывающая адресата.

❹ Сведения, позволяющие найти и точно идентифицировать объект (адрес файла, папки, URL, адрес электронной почты и др.).

❺ Цифровое или буквенно-цифровое обозначение зоны запоминающего устройства или отдельной его ячейки, определяющее место хранения информации в памяти компьютера.

❻ Почтовый адрес.

Адрес команды

Адрес области памяти, которая занята командой. (ГОСТ).

Адрес операнда

Адрес ячейки или области памяти, откуда извлекаются обрабатываемые данные.

Адрес результата

Адрес, по которому записывается значение результата операции.

Адресация

Присвоение адресов объектам и фрагментам информации.

Адресные сервисы (*см. Location Services*)

Аккумулятор (синоним: *накапливающий сумматор, накапливающий регистр*)

❶ Устройство, вырабатывающее электричество путем преобразования химической энергии в электрическую. Имеется возможность многократной перезарядки. Используются в настольных компьютерах, как вспомогательное энергопитание, в компьютерах переносного типа, как основное, кроме этого – в устройствах бесперебойного питания.

❷ Ячейка памяти, используемая для хранения результатов вычисления; обычно так называют один из регистров в арифметико-логическом устройстве процессора.

❸ Узел арифметико-логического устройства, сохраняющий результаты предыдущих операций для использования их в последующих операциях. (См. *сумматор*).

Аксиома

❶ Основное положение, самоочевидный принцип. В дедуктивных научных теориях аксиомами называются основные исходные положения, той или иной теории, из которых путём дедукции, то есть чисто логическими средствами, извлекается всё остальное её содержание. (Мат. энц.)

❷ Положение, принимаемое без логического доказательства, в силу непосредственной убедительности: истинное исходное положение теории.

Актуализация

❶ Процесс, обеспечивающий постоянное внесение текущих изменений в состояние системы, базы данных.

❷ Осуществление, переход из состояния возможности в состояние действительности. В сетевом планировании — отражение в сетевом графике выполненных работ.

Алгебра

❶ Часть математики, посвящённая изучению *алгебраических операций*. Простейшими алгебраическими операциями являются арифметические действия (операции) над натуральными и положительными рациональными числами. Термин "Алгебра" происходит от названий сочинения Мухаммеда аль-Хорезми "Альджебр аль-мукабала" (9 в.), содержащего общие приёмы для решения задач, сводящихся к алгебраическим уравнениям 1-й и 2-й степеней. В этом понимании термин "Алгебра" употребляется в таких сочетаниях, как *гомологическая алгебра, коммутативная алгебра, линейная алгебра, полилинейная алгебра, топологическая алгебра*. (Мат. энц.)

❷ Частный случай *операторного кольца*: алгебра над полем, телом, коммутативным кольцом. Ассоциативная алгебра, не ассоциативная алгебра, альтернативная алгебра. (Мат. энц.)

Алгебра логики (синоним: булева алгебра)

Алгебра, в которой каждая переменная может принимать одно из двух значений: "истинно" или "ложно". (См. *алгебра*).

Алгоритм

❶ Набор правил или описание последовательности операций для решения определённой задачи или достижения определённой цели.

❷ Последовательность чётко определённых правил или команд (действий или шагов), исполнение которых позволяет решать конкретную задачу за конечное число шагов.

❸ Формальное описание способа решения задачи путем разбиения ее на конечную по времени последовательность действий (элементарных операций). Термин “формальное” подразумевает, что описание должно быть абсолютно полным и учитывать все возможные ситуации, которые могут встретиться по ходу решения. Под элементарной операцией понимается действие, которое по заранее определенным критериям (например, очевидности) не имеет смысла детализировать.

❹ Заранее определенное, точное предписание, которое задает дискретный (пошаговый) процесс, начинающийся определенным образом и приводящий к результату за конечное число шагов. Это понятие относится к исходным математическим понятиям, которые не могут быть определены через другие, более простые понятия. Иногда такое или подобное определение называют интуитивным, т.е. понятным из опыта.

Каждый алгоритм, в общем случае, должен задаваться:

–множеством допустимых исходных данных,

–начальным состоянием,

–множеством допустимых промежуточных состояний,

–правилами перехода из одного состояния в другое,

–множеством конечных результатов,

–конечным состоянием.

В зависимости от конкретного задания этих параметров, определяются классы алгоритмов. Например, алгоритмы линейные, циклические, сортировки и т.д. При разработке алгоритма всегда должен предполагаться его исполнитель. Слово алгоритм, является производным от имени среднеазиатского ученого Аль Хорезми, уроженца Хивы, жившего в IX веке нашей эры.

❺ Математическое определение алгоритма есть уточнение понятия алгоритма в интуитивном смысле, и представляется в виде машины Тьюринга, машины Поста, нормального алгоритма Маркова и пр.

Алгоритмизация процесса

Построение алгоритма, выполнение которого реализует модель данного процесса.

Алгоритмический язык

Язык, предназначенный для представления алгоритмов. (ГОСТ 19781-83).

Алгоритмы маршрутизации

Алгоритмы маршрутизации описывают процесс определения наиболее предпочтительного пути пакета к адресату в сети на основании данных таблиц маршрутизации. Простейшие алгоритмы маршрутизации выбирают путь с наименьшим числом переходов (транзитных узлов), более сложные учитывают задержку, пропускную способность или реальную стоимость различных физических или логических каналов связи.

Алфавит

Набор сисмволов, из которых может быть составлено любое сообщение на данном языке.

Анализ (от греческого *analysis* - разложение)

❶ (в программировании) Стадия в разработке системы, во время которой анализируются требования и предметная область. На стадии анализа разработчики фокусируют внимание на том, что им предстоит сделать, а на стадии проектирования – каким образом они будут это осуществлять.

❷ Дословно, разбиение целого на части. То есть расчленение (мысленное или реальное) объекта на элементы для последующего детального изучения. Анализ неразрывно связан с синтезом.

❸ Синоним научного исследования вообще.

❹ В математике, анализом называется исследование предельных процессов и отыскание устойчивых алгоритмов вычисления бесконечно малых значений. В кибернетическом анализе, целое описывается не просто в терминах его частей, а в основном в виде моделей, объединяющих его части в целое (т.е. в терминах: *отношений, зависимостей, связей, передачи сообщений, структуры и организации*). Такой анализ выявляет целостные (обобщённые) свойства системы без нарушения строения исследуемой системы или потери информации. (Krippendorff).

❺ Последовательный процесс, состоящий из следующих этапов:

–определения ответа на поставленный вопрос или некоторого результата,

–моделирования проблемы,

–изучение результатов моделирования,

–интерпретации результатов (и, возможно),

–выдачу рекомендаций.

Анализ контента (см. *контента анализ*)

Анализ системный (см. *прикладной системный анализ*)

Аналоговые (системы)

Системы, в которых регистрируемые, передаваемые и отображаемые сигналы могут представлять данные в аналоговом виде (т.е. как «действительные числа»).

Аналоговый (analog)

Представление объектов, физических условий или процессов, которое однозначно представляет исходный оригинал, отражая любые изменения его состояния. В технологиях, аналоговые устройства создаются для контроля процессов, таких как звук, движение или температура и преобразуют результаты измерений в электрические сигналы или механические перемещения, представляющие колебания исходного процесса.

Аналоговый сигнал

Форма электрического сигнала или колебательного процесса, амплитуда и/или частота которого изменяется непрерывно, т.е. сигнал содержит информацию в каждый момент времени, а не в определенные, дискретные. Аналоговые сигналы подвержены внешним воздействиям, которые могут изменять характер колебания.

Аналоговая вычислительная машина

Вычислительная машина, которая оперирует данными, представленными в аналоговом виде. Аналоговые вычислительные машины практически всегда жестко специализированы. Отличаются от цифровых большей скоростью выполнения операций и простотой программирования. Предполагается, что аналоговые вычислительные машины получат свое дальнейшее развитие при создании нейрокомпьютера.

Аналого-цифровая вычислительная машина

Вычислительная машина, которая оперирует как с данными, представленными в аналоговом виде, так и данными, представленными в цифровом виде.

Аналого-цифровой преобразователь

Устройство, преобразующее аналоговый сигнал в цифровой и обратно. Например, для передачи данных по цифровой телефонной сети с помощью модема, между модемом и цифровым телефонным каналом ставится аналого-цифровой адаптер.

Анимация

Процесс создания движущихся графических изображений на экране дисплея. Используется при программном проектировании различных движущихся объектов, моделировании физических явлений, а также в обучающих системах и игровых программах.

Аннотация

Краткая характеристика содержания документа, его части или группы документов с точки зрения назначения, содержания, формы и других особенностей.

Апертура

Порция адресов памяти PCI, выделенная в адреса графической памяти. Циклы, обращающиеся к этим адресам, не требуют трансляции и передаются напрямую в AGP. Кроме того, размер указывает максимальный объем системной памяти, выделяемый для хранения текстур. Это означает, что видеоплатам выделяется адресное пространство, причем независимо от фактической емкости видеопамяти платы. Размер апертуры незначительно сказывается на общей производительности системы. Но большинство современных 3D-акселераторов требует значительно больше, чем 8МБ апертуры для нормального функционирования.

Аппаратные средства (Hardware)

Материальная часть вычислительной системы (компьютера) включающая электрические, электронные, электромеханические и механические элементы (включая стойки и корпуса). (ГОСТ).

Апплет (applet)

❶ Программа, передаваемая при использовании WWW-технологий на компьютер клиента в виде отдельного файла и запускаемая при просмотре Web-страницы в браузере.

❷ Компонент, который обычно выполняется в Web-браузере, но может также выполняться в других разнообразных приложений и устройствах, которые поддерживают программную модель взаимодействия апплетов.

❸ Приложение, написанное на языке программирования Java, полученное компьютером-клиентом из сети Internet. Это программа, которая выполняется виртуальной машиной системы Java. Апплет практически изолирован от машины-клиента, но может общаться с сервером, с которого получен. Иногда употребляется термин "апплетка".

Аргумент

Переменная (независимая) от значения которой зависят значения функции.

Арифметико-логическое устройство (АЛУ)

Блок вычислительной системы, содержащий схемы выполнения арифметических и логических операций.

Арифметическое выражение

Выражение, где операндами являются объекты, над которыми выполняются арифметические операции. Каждый язык программирования задает свои правила образования выражений и свои обозначения операций (синтаксис).

Арифметическая операция

Простейшая вычислительная операция над числами. Во многих языках программирования определены двуместные арифметические операции: сложения (+), вычитания (-), умножения (*), деления (/), деления нацело (div, иногда \), деление по модулю (mod); одноместные операции присваивания знака (+,-). В языке программирования C, например, введена операция увеличения (++), которая увеличивает значение операнда на единицу. К примеру, выражение ***a***++ означает, что после выполнения операции значение переменной ***a*** увеличивается на 1.

Артефакт

❶ Объект, созданный или модифицированный путём выполнения определённой работы одним или несколькими лицами, в отличие от естественного объекта, называемого образцом или экземпляром.

❷ Часть информации, которая используется или производится в процессе разработки программной системы (рабочий проект, рабочий документ, рабочий продукт (изделие), исходный код, версия и т.д.). Артефакт может быть моделью, описанием или программным обеспечением. (UML).

❸ Часть цифровой информации. Артефакт может иметь любой размер и состоять из других артефактов. Примерами артефактов могут служить: сообщение; URI; XML документ; PNG изображение; поток битов (двоичных сигналов - *a bit stream*). (W3C).

Архивация

Процесс сохранения временно ненужных данных, либо создания резервных копий данных. При архивации файлы обычно записывают в более плотном виде для экономии памяти. Часто архивацией называют сам процесс упаковки, или сжатия данных.

Архитектура

❶ Методология объединения и организации взаимодействия элементов сложной структуры на логическом, физическом и программном уровнях.

❷ Организационная структура системы, включающая её декомпозицию на составляющие части (элементы), их связи, механизмы взаимодействия, а также основные признаки, сообщающие о конструкции системы. (UML).

❸ Обобщённое определение системы с точки зрения существующих в ней информационных потоков и способов их обработки.

❹ Описание вычислительной системы на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд и средств пользовательского интерфейса, организации памяти и системы адресации, операций ввода-вывода и управления и т.д. (Иллингуорт)

❺ Значение понятия "*Архитектура системы*" относительно недавно определено в индустрии программирования систем. Архитектура является пространством, в котором взаимодействуют объекты (objects operate). Она также определяет соглашения, используя которые, внутренние объекты взаимодействуют с компонентами внешней по отношению к ней системы и друг с другом. Основное назначение архитектуры системы проявляется в ответе на вопрос "А что если ...?".

❻ (зодчество) Проектирование и строительство зданий и других сооружений. Является одновременно областью материального производства и художественного творчества.

Архитектура информационных систем (Information Systems Architecture)

Официальное определение правил бизнеса, структур систем, технических ограничений и сути производимой продукции для информационных бизнес-систем. Архитектура информационных систем состоит из четырех уровней: архитектура бизнеса, архитектура систем, техническая архитектура и производственная архитектура.

Архитектура “клиент/сервер” (Client/Server Architecture – CSA)

❶ Архитектура сети, в которой мощные компьютеры (серверы) представляют функции баз данных, приложений и управления системой клиентам, работающим на рабочих станциях.

❷ Одна из наиболее популярных в компьютерных технологиях моделей взаимодействия и обмена данными между программными и аппаратными компонентами компьютеров и компьютерных систем и сетей.

❸ Технология взаимодействия компьютеров в сети, когда один компьютер (клиент) формирует запрос, например, поиск в базе данных, более мощному компьютеру (серверу), расположенному в другом месте. Клиент формирует и отправляет запрос, а сервер образует ответ, который передается клиенту для вывода на экран или на печать. (См. *клиент-сервер*).

Архитектура многоярусная (см. *многоярусная архитектура*).

Архитектура приложения

Архитектура приложения определяет то, как будет организован код представления, код обработки данных и код обращения к хранилищам данных. (См. *приложение*).

Архитектура программного обеспечения (ПО)

Описание структуры программной системы. Различные архитектурные модели, такие как структурная модель, модель управления и модель модульной декомпозиции, разрабатываются в процессе архитектурного проектирования. Большие системы редко

сводятся к одной архитектурной модели. Они неоднородны и на разных уровнях обобщения используют разные модели.

Архитектура производственная (Enterprise architecture)

Структурированное описание делопроизводства и бизнес-процессов предприятия, приложений и методов автоматизации, поддерживающих бизнес-процессы, а также информация, технологии и инфраструктура, необходимые для их выполнения. Производственная архитектура позволяет выработать целостный план работ и скоординированных проектов, необходимых для претворения в жизнь задач развития информационной инфраструктуры предприятия.

Архитектура систем (Systems Architecture)

Один из четырех уровней архитектуры информационных систем. Архитектура систем представляет определения и внутренние отношения между приложениями и архитектурой продуктов.

Архитектура Хранилища Данных (Data Warehouse Architecture)

Интегрированный набор продуктов, позволяющий извлекать и трансформировать оперативные данные для загрузки в базу данных для последующего анализа и формирования отчетов конечным пользователем.

Архитектура фон Неймана

Классическая архитектура построения компьютера, в которой выделены: оперативная, последовательно адресуемая память, где хранятся как данные, так и сама программа; процессор, последовательно выполняющий команды из программы. Большинство компьютеров в настоящее время имеют эту архитектуру. Примером другой архитектуры могут служить многопроцессорные компьютеры с параллельными вычислениями. Название дано в честь одного из разработчиков данной архитектуры, известного математика Джона фон Неймана.

Архитектурный элемент (architectural element)

Общий термин, относящийся к части архитектуры, такой как компонент, коннектор или данные. Взаимосвязи между такими элементами ограничены задачами достижения заданного набора архитектурных свойств. (W3C).

Аспект

Точка зрения, с которой рассматривается какое-либо явление, понятие, перспектива.

Ассемблер (см. *язык ассемблера*).

Атомарные данные (Atomic Data)

Элементы данных, представляющие собой самый низший уровень детализации. Например, в ежедневном отчете о продажах отдельные проданные предметы будут атомарными данными, а обобщенные понятия (такие, как счета-фактуры и общие итоги по ним) - агрегатами данных.

Атрибут (Attribute)

❶ Характеристика файла, которая может быть установлена или сброшена. Стандартные атрибуты**:** только считываемый, скрытый, системный, архивный.

❷ Дополнительная информация о метках (tag) гипертекстового документа HTML.

❸ Информация, определяющая способ вывода символа (attribute character).

Аутентификация (authentication)

❶ Средство защиты, определяющее подлинность пользователя и законность его работы.

❷ Часть процедуры верификации. Включает в себя проверку источника, уникальности и целостности сообщения. Процесс аутентификации определяет пользователя как истинного на основе цифровых аутентификационных сертификатов.

Аутентификационный сертификат

Цифровой сертификат. Содержит информацию о владельце, об организации, выпустившей его, уникальный серийный номер, срок действия и зашифрованный блок для

верификации содержимого сертификата. Сертификаты выпускаются определенными организациями, пользующимися доверием сторон, применяющих данные сертификаты.

Аутсорсинг (см. *Outsourcing, ASP*)

- Б -

Баг (bug – дословно - насекомое)

Сбой, ошибка в работе программы, вследствие наличия бага, т.е. прерывание работы программы вследствие обнаружения ошибки (синтаксической, семантической или на уровне выполнения).

База данных

❶ Совокупность данных, существенная для некоторой деятельности.

❷ Совокупность данных, организованных по определённым правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ (приложений). (ГОСТ)

База данных локального доступа (Local Access Database, LAD)

База данных, обслуживающая отдельные системы и рабочие группы; конечный пункт общего распределения данных. LAD'ы являются "розничными торговым точками" в сети хранилища данных. Они обеспечивают прямой доступ к данным, необходимым конкретным настольным системам или службам запросов. Данные попадают в LAD'ы из Хранилищ Данных согласно условиям поднаборов стандартных наборов. Эти данные обычно находятся на LAN-сервере. Если серверы отсутствуют и данные статичны, они могут находиться на рабочем столе пользователя. (См. Сеть Хранилища Данных).

База знаний

Информационная база, отражающая опыт конкретных людей, групп, обществ, человечества в целом, в решении творческих задач в выделенных сферах деятельности, традиционно считавшихся прерогативой интеллекта человека.

Базовая система ввода-вывода (BIOS- Basic Input Output System)

Группа программ, которые работают непосредственно с базовыми аппаратными средствами компьютера и с некоторыми периферийными устройствами, выполняя самые фундаментальные задачи в системе – обмен на уровне байта с клавиатурой, экраном, дискетой, жестким диском и т.д. (см. *BIOS*).

Байт (byte)

❶ Часть машинного слова, состоящая обычно из восьми битов.

❷ Минимальная адресуемая единица памяти.

❸ Общепринятая единица измерения информации, используемая для указания размера памяти, скорости обмена информации и других характеристик компьютера. Один байт состоит из восьми битов (восьми двоичных разрядов). При представлении символов текстовой информации каждая буква, цифра или знак занимает один байт.

1 байт = 8 бит.

1 килобайт (K) = 1,024 байта.

1 мегабайт (MB) = 1,024 килобайта или приблизительно один миллион байтов.

1 гигабайт (GB) = 1,024 мегабайта или приблизительно один миллиард байтов.

Байткод (в языке Java)

Оптимизированный набор команд, предназначенных для выполнения виртуальным устройством, которое эмулирует Java-система в процессе выполнения апплета. Байткод, как правило, интерпретируется. Интерпретация - это самый простой способ создания переносимых и безопасных программ.

Банк (памяти)

Группа модулей памяти одинаковой ёмкости, которые должны быть установлены одновременно, чтобы система могла работать. Количество модулей равняется отношению

ширины системной шины к ширине шины модуля (умноженному на коэффициент чередования (interleav)).

Баннер (banner)

Узкая полоса графических рекламных материалов, отображаемых на Web-сайте того, кто сдаёт внаём или продаёт пространство на своих страницах, всем желающим разместить рекламу.

Бесплатные компьютерные программы (см. *freeware*)

Бета-тестер (Beta tester)

Специалист, пытающийся выявить ошибки в программном продукте до его поставки. Обычно эта работа не оплачивается и строится на личной инициативе.

Бета-тестирование (beta test)

Тестирование программного обеспечения добровольцами из числа клиентов, проводимое непосредственно перед официальным выпуском продукта. Предназначается для выявления проблем, которые могут возникнуть в ходе реальной эксплуатации, но не были обнаружены при внутреннем тестировании. Если бета-тестеры обнаруживают серьезные недостатки, разработчик устраняет их и, прежде чем выпускать программное обеспечение на рынок, проводит еще одно бета-тестирование.

Бизнес-данные (Business Data)

Информация о людях, местах, вещах, деловом регламенте и событиях, используемых для управления бизнесом. Это не метаданные - метаданные определяют и описывают бизнес-данные.

Бизнес-инжиниринг

Деятельность, направленная на проектирование и реализацию бизнес-приложений, т.е. программных средств, предназначенных для решения деловых и экономических задач.

Бизнес логика

Код, который реализует функциональную часть приложения. В модели Enterprise JavaBaen, такая логика реализуется в виде методов объектов корпоративных компонентов (Bean).

Бизнес метод

Метод в корпоративном компоненте (Bean, Java), который реализует бизнес логику или правила приложения.

Бизнес-модель (Business Model)

Представление данного бизнеса в любой момент времени. Представление может быть сформировано на основе процесса, данных, события или ресурсной перспективы и может представлять собой прошлое, нынешнее или будущее состояние данного бизнеса.

Бизнес-процесс (business process)

❶ Совокупность или ряд взаимосвязанных действий, получающих на входе данные разных типов и продуцирующих результат, направленный на предоставление добавленной стоимости потребителю. Например, процесс выполнения заказа на входе получает заказ и выдает в качестве результата заказанные товары, т.е. доставка заказанных товаров потребителю и есть та ценность, которую создает процесс.

Бизнес-транзакция (Business Transaction)

Элемент процесса, исполняемый системой захвата данных для создания, изменения или удаления бизнес-данных. Каждая транзакция представляет собой отдельно оцениваемый факт, описывающий отдельное событие бизнеса.

Бинарный (двоичный) файл (Binary file)

Файл, содержащий информацию, которую нельзя представить или осмысленно интерпретировать как текст. Типичные примеры бинарных файлов**:** программные файлы, файлы большинства баз данных и электронных таблиц, упакованные файлы, графические файлы и т.д.

Бит (Binary digIT)

❶ Фундаментальная единица информации, используемая в теории информации. Обозначает количество информации, необходимое для различия двух равновероятных событий.

❷ Минимальная единица представления информации в компьютерной технике, занимающая один разряд байта и способная принимать только два значения – 0 и 1.

Блок (Block)

❶ В коммуникации – набор данных в конверте символов синхронизации, адресации, управления и контроля ошибок, передаваемый как одно целое.

❷ Выбранный фрагмент документа, с которым можно работать как с одним целым.

❸ Прямоугольная область пикселов (picsels).

❹ Группа последовательных байтов в памяти.

❺ (полит.) Партийный блок.

❻ (строит.) Строительный блок.

Браузер (броузер, browser) (*см .- контейнер*)

Клиентская программа-контейнер, дающая возможность пользователю читать гипертекстовые документы в WWW и перемещаться между ними (путём навигации в системе адресов WWW – гиперссылок). Таким образом, браузер является программным приложением, используемым для локализации и представления содержания Web-страниц.

Брокер (broker – посредник)

❶ (выч.техн.) Программное обеспечение, устанавливающее соответствие сервисных запросов клиента серверным реализациям. (CORBA).

❷ Лицо (предприятие), выполняющее чужие заказы на покупку-продажу. Другими словами – посредник в операциях с валютой, ценными бумагами, товарами, недвижимостью, в страховании и т.д., который заключает сделку от своего имени, но за счет клиента.

Брэнд (Brand – клеймо, фабричная марка)

❶ Обозначает не столько конкретный товар, сколько образ, который связан с определённым производителем (к примеру: IBM, Intel, Microsoft, Sony, Mersedes). Брэнд – это совокупность ощущений потребителя от впечатлений, связанных с тем или иным именем.

❷ Продукт, компания или концепция, выделенные общественным сознанием из массы себе подобных. Как правило, сам по себе "бренд" – это слово или фраза, которое законодательно защищено.

Брэнд нэйм *(brand name- фабричная марка)*

❶ Для модулей – подразумевает тот факт, что модуль сделан известным (соблюдающим стандарты и высокое качество) производителем и имеет его маркировку. Единого мнения о том, каких производителей считать *известными*, не существует.

❷ Для систем – тот факт, что система произведена крупным производителем компьютерной техники, который специально продает модули расширения со своей маркировкой и рекомендует использовать именно их.

Буфер *(buffer)*

Память для промежуточного (временного) хранения данных. Обычно используется для компенсации разницы в скорости обработки информации при передаче данных между двумя устройствами с различным быстродействием.

Буфер обмена (clipboard) *(в операционной системе Windows)*

Специальная область оперативной памяти, обслуживаемая операционной системой. Она легко доступна всем приложениям и используется для передачи данных между ними. Для обеспечения совместимости между приложениями, буфер обмена хранит передаваемые данные одновременно в нескольких разных форматах.

Буферный регистр

Регистр, через который происходит обмен между оперативной памятью и внешним устройством.

Быстродействие

Показатель скорости работы компьютера и его производительности в единицу времени.

- В -

Ввод (input)

❶ Внешнее по отношению к системе событие, переводящее систему в новое состояние.

❷ Команда исполнителю взять порцию данных из определённого места внешней среды и поместить её в устройствах компьютера для последующей обработки.

Ввод–вывод (данных)

Обмен данными под управлением компьютера. (ГОСТ).

Веб-сайт, сайт,– (см. *Site, web site*)

Веб-сайт - определенное место в Интернет, доступное из любой точки мирового пространства, представляющее компанию или индивидуума. Сайт состоит из одной или нескольких страниц, объединенных по смыслу, навигационно и по месту расположения (на логическом уровне), а так же, как правило, имеющий единый стиль оформления.

Веб-сервисы (см. *Web services*)

Веб-технологии (см. *Web-технологии*)

Векторный рисунок

Рисунок, элементы изображения которого описываются математическими формулами. Обычно такие элементы называются объектами и с каждым из них можно работать как с единым целым, то есть перемещать их, раскрашивать, изменять размеры и так далее. В векторных рисунках, сложнее, чем в растровых редактировать детали изображения.

Величина

❶ Одно из основных математических понятий, смысл которого с развитием математики подвергался ряду обобщений. Основное, сформулированное ещё в "Началах" Евклида (3 в. до нашей эры), представляется *положительными скалярными величинами* и является непосредственным обобщением более конкретных понятий: длины, площади, объёма, массы и т. д. Сюда же относится и система *действительных чисел*, в которой каждое из них допустимо называть величиной. (Мат. энц.).

❷ Размер, объем, протяжение вещи.

❸ (мат. физ.) Всё, что можно измерить и исчислить. К примеру, бесконечно малая величина, неизвестная величина, переменная величина.

❹ (книжн.) Всё, имеющее общественную ценность или значение. К примеру, литературная величина (о значительном писателе).

Верификация

❶ Процесс подтверждения выполнения программой заложенных в неё функций, т.е. проверка правильности программы путём формального доказательства соответствия программы заданной спецификации.

❷ Формальное (обычно полуавтоматическое) доказательство правильности программы, использующее предусловия и постусловия для процедур и операторы контроля.

Вещественное (реальное, дробное) число.

Тип числа с дробной частью в языках программирования. Буквально, ″число с точкой″, разделяющей целую и дробную часть числа (к примеру: 12.4183). В компьютерах часто представляется в форме *с фиксированной* или *плавающей точкой*. (См. *число с фиксированной точкой, число с плавающей точкой*).

Видеоадаптер

Видеоадаптер является устройством, непосредственно формирующим изображение на мониторе. Различают два режима работы видеоадаптера — текстовый и графический. В текстовом режиме на экране отображается текст в виде символов, внешний вид которых определяет знакогенератор карты. Каждому символу ставится в соответствие число — его порядковый номер в наборе матриц знакогенератора, что определяет раскладку таблицы символов. Всего таких символов в стандартной таблице 256 и нумеруются они от 0 до 255. Конкретное начертание набора называется кодовой страницей, а несколько таких наборов для различных режимов — символьной раскладкой или набором для соответствующей национальной спецификации. Графический режим предполагает изображение на экране монитора объектов произвольной формы и сложности. В графическом режиме изображение кодируется как набор пикселов.

Визуализация

Визуальное представление данных.

Винчестер, винчестерский диск (Winchester disk)

Дисковое внешнее запоминающее устройство, в котором носитель данных, магнитные головки и другие механические компоненты помещены в герметический кожух. Происхождение названия по одной версии, происходит от места первоначальной разработки – филиала IBM в г. Винчестере (Великобритания), а по другой – потому, что расположение двух дисков в первоначальном варианте исполнения напомнило разработчикам расположение стволов в одноименном ружье.

Виртуальная машина

Совокупность ресурсов, которые эмулируют поведение реальной машины. Концепция виртуальной машины появилась в Кембридже (шт. Массачусетс) в конце 60-х годов как расширение концепции виртуальной памяти.

Виртуальная память

Система, при которой рабочее пространство процесса частично располагается в быстродействующей памяти (типа ОЗУ), а частично в некотором более медленном устройстве (обычно на диске).

Виртуальная реальность (Virtual reality)

❶ Компьютерные системы, которые обеспечивают визуальные и звуковые эффекты, погружающие зрителя в воображаемый мир за экраном. Пользователь окружается порожденными компьютером образами и звуками, дающими впечатление реальности. Пользователь взаимодействует с искусственным миром с помощью различных сенсоров, таких как, например, шлем и перчатки, которые связывают его движения и впечатления и аудиовизуальные эффекты. Будущие исследования в области виртуальной реальности направлены на увеличение чувства реальности наблюдаемого.

❷ Новая технология бесконтактного информационного взаимодействия, реализующая с помощью комплексных мультимедиа-операционных сред иллюзию непосредственного вхождения и присутствия в реальном времени в стереоскопически представленном «экранном мире». Более абстрактно - это мнимый мир, создаваемый в воображении пользователя.

Виртуальная система (Virtual system)

Операционная система, которая обеспечивает в режиме разделения времени многих пользователей виртуальными ресурсами центрального процессора, памяти, каналов.

Виртуальная экономика

Проведение экономических операций в электронном пространстве.

Виртуальное предприятие

❶ Предприятие, создаваемое путем объединения (интеграции) людских, финансовых, материальных, организационно-технологических и прочих ресурсов с использованием компьютерных сетей. Это позволяет сформировать гибкую и динамичную организационную систему, наиболее приспособленную к скорейшему выпуску и

оперативной поставке новой продукции на рынок. Идея такого подхода к географически распределенным ресурсам в интересах общей работы над уникальными проектами или новыми продуктами стала общепризнанной трактовкой виртуальной организации. Классическими примерами виртуальных предприятий служат европейский консорциум Airbus Industries, изготавливающий широко известные аэробусы, объединившие усилия при работе над проектом Powerbook фирмы Apple и Sony, а также многие компьютерные фирмы, имеющие офисы в самых отдалённых уголках мира: Xerox, Hewlett-Packard, IBM и многие другие с количеством сотрудников от 100 000 и более..

❷ В абстрактном смысле – это наиболее передовая и эффективная форма организации предприятия из ряда «мысленно возможных», т.е. наилучшая с точки зрения имеющихся технических и экономических условий. Конкретнее, виртуальное предприятие означает сетевую, распределённую, компьютерно интегрированную организационную структуру, объединяющую неоднородные ресурсы, расположенные в различных местах. Нередко акцент делается на временный характер объединения ресурсов в виртуальной организации: тогда она понимается как *межорганизационное гибкое предприятие*, создаваемое на ограниченный период, главная цель которого – получение выгоды благодаря расширению ассортимента товаров и услуг. Важнейшей характеристикой виртуальной организации является гибкая, адаптивная, динамичная сетевая структура. Поскольку такая сеть не существует в реальном физическом пространстве, а создается путем информационной интеграции ресурсов партнеров, ее нередко называют *квазипредприятием*. В то же время виртуальное предприятие объединяет цели, культуру, традиции, ресурсы, опыт ряда предприятий-партнеров, координируя их развитие и представляя собой «предприятие над предприятиями» т.е. *метапредприятие*. Ключевой проблемой обеспечения эффективности виртуальных предприятий является управление знаниями, циркулирующими в сетях. (У.Дэвидоу и М.Мэлоун)

❸ Часто главная стратегия *виртуального предприятия* связывается с ориентацией на заказчика, поскольку ее основные характеристики – это быстрота выполнения заказа (Minimal Time-to-Market) и полнота удовлетворения требований клиента. С включением заказчиков и исполнителей в единую открытую сеть границы между взаимодействующими организациями становятся нечеткими, прозрачными и подвижными.

Виртуальные миры (Virtual worlds)

Моделируемые на экранах компьютеров явления и процессы реальности. С помощью таких моделей продумываются возможные варианты различных жизненных ситуаций и проекты в области градостроительства, прокладки коммуникационных линий, производства, торговли, образования, науки, медицины и многих других форм общественно-культурной деятельности.

Виртуальный

Не имеющий физического воплощения или воспринимаемый иначе, чем реализован. К примеру, программно реализованные кнопки команд на стандартной панели инструментов Windows–*приложений* Word, Excel, Access и многих других, при нажатии имитируют настоящие кнопки, хотя в природе их не существует, а реализованы они программно, т.е. посредством написания и последующего выполнения этих фрагментов программных кодов.

Витрина данных (Data Mart), Предметно-ориентированная база данных (Subject Oriented Databases) (См. также *База данных частного доступа (LAD*).

Вместо построения одного крупного централизованного Хранилища Данных многие компании создают несколько предметно-ориентированных хранилищ для обслуживания потребностей различных подразделений. Такие хранилища образуют систему, называемую Витриной данных (Data Mart).

Воксель

При объёмных построениях виртуальных компьютерных трёхмерных тел их элементы моделируются трёхмерными пикселами (кубиками), именуемых вокселями.

Волоконно-оптическая линия связи

Стеклянный или полимерный носитель, используемый для передачи данных. Передаваемые световые волны излучаются источником лазерного типа. Волоконно-оптические кабели обеспечивают высокую секретность связи, имеют широкую полосу пропускания и занимают мало места. Могут рассматриваться в виде физического носителя для всех наземных систем связи в будущем.

Волоконно-оптический разъём

Волоконно-оптические разъемы предназначены для организации физического соединения двух сегментов оптического волокна, диаметр которого не превышает нескольких нанометров, то есть значительно меньше диаметра человеческого волоса. Точность юстировки сегментов оптического волокна имеет при этом первостепенное значение, так как именно от нее зависит количество световых лучей, которые будут попадать в один сегмент волокна из другого.

Всемирная Паутина (World Wide Web) (см. также *WWW*)

❶ Служба в интернете, которая позволяет легко получать доступ к информации на серверах, расположенных по всему миру.

❷ Служба в интернете, организующая информацию с использованием гипермедиа. Каждый документ может содержать ссылки на образы, звуки или другие документы.

Всплывающие подсказки (tooltips)

Небольшие всплывающие окна, в которых выводится название элемента управления, не имеющего текстовой метки. Появляются автоматически после того, как указатель мыши некоторое время неподвижно простоит над элементом управления.

Вывод (output)

❶ Любое изменение, производимое системой в окружающей её среде. (Umpleby).

❷ Команда исполнителю передать текущее значение указанного выражения во внешнюю среду.

Выделенная линия (канал)

Линия/канал, зарезервированная/ый для исключительного использования заказчиком.

Выключка

Официальный термин, относящийся к верстке. Равномерное увеличение или уменьшение пробелов между словами (а иногда и между буквами) для доведения строки до заданной точно ширины. Используется как в простом варианте (колонка текста, выровненного по обоим краям), так и в более сложных (фигурная выключка, когда текст "обтекает" картинку со сложными контурами). Обычно термин употребляется в контексте возможности использования для выравнивания текста регулировки именно меж буквенных интервалов.

Выполнение программы

Последовательный процесс, состоящий из следующих шагов.

1) Исходный модуль программы, то есть текстовый файл, как правило, размещённый на жёстком диске и содержащий текст программы на языке программирования высокого уровня (Turbo Pascal, C++ и т.д.) обрабатывается компилятором соответствующего языка. В результате получается объектный модуль, т.е. новый файл с новым расширением, содержащий двоичные коды программы на машинном языке.

2) С применением программы компоновщика из объектных модулей и, возможно, библиотечных модулей, вызываемых из запускаемой программы, строится загрузочный модуль с расширением .EXE. Под вызываемыми модулями подразумеваются те, имена которых упоминаются в тексте исходной программы.

3) Загрузочный модуль помещается (загружается) в оперативную память и там выполняется. При этом осуществляется пооператорное выполнение программы, представленной в виде машинных команд используемого в

компьютере процессора. На этапе выполнения возможно подключение динамически загружаемых библиотек (DLL) и программных компонент COM и DCOM.

4) Программа выгружается из оперативной памяти компьютера обратно на жёсткий диск.

Выражение

❶ (выч.техн.) Элемент программы, вырабатывающий значение, то есть последовательность операндов, объединённая знаками операций (операторов).

❷ Закономерно построенный текст, образованный знаками операций, именами функций и величин, скобками, записями констант, задающий правило вычисления своего значения как функции текущих значений входящих в него величин.

❸ Оборот речи, принятый в каком-либо языке. Слово или слова, служащие для передачи мысли. К примеру, образное выражение, непонятное выражение.

❹ (мат.) Совокупность математических обозначений, соединённых знаками математических операций; формула, выражающая какие-либо математические отношения. К примеру, алгебраическое выражение.

❺ Характерные внешние черты, отражающие душевное состояние, т.е. элемент мимики. К примеру, страдальческое выражение лица, грустное выражение глаз.

Высокие технологии (high technology)

Технологии, включающие прогрессивные специализированные системы или устройства. Относится к чрезвычайно быстро развивающейся отрасли разработки и производства средств электроники и компьютеров.

Вычисление

Выполнение арифметических и логических операций над данными с целью получения требуемого результата. (ГОСТ).

Вычислительная система (computing system)

Совокупность технических и программных средств, обеспечивающая выполнение вычислительных работ. (ГОСТ).

Вычислительный эксперимент

Проведение расчётов на компьютере с целью моделирования физических и инженерно-технических процессов. (ГОСТ).

- Г -

Гаджет (Gadgets - средства, приспособления (англ.))

❶ Gadgets & Widgets - элементы пользовательского интерфейса (термин применяется в основном в библиотеках Xt для X Window System).

❷ Миниатюрные, многофункциональные устройства: мобильные телефоны, пейджеры, плейеры, цифровые фотоаппараты, микрокомпьютеры и прочие «экзотические» электронные устройства.

❸ Реализация некоторого сервиса, запускаемая портальным сервером, которая содержит некоторые данные, набор собственных бизнес функций, а также стандартное представление на рабочих панелях портала. Гаджеты обычно (но не обязательно) выглядят как стандартные "окошки" на рабочей панели компьютера.

Гарнитура, шрифт и шрифтовое семейство

Согласно официальному значению, шрифт есть комплект литер, воспроизводящий какой-либо алфавит, а также цифры и знаки. Гарнитура есть обладающее собственным наименованием семейство начертаний шрифта, имеющих общие стилевые особенности и отличительные детали рисунка знаков. В полиграфии шрифтовым семейством называют совокупность начертаний, принадлежащих одной в официальном смысле гарнитуре. Гарнитура здесь есть уникальный по форме букв набор символов (букв, знаков, цифр),

составляющий в официальном смысле шрифт, обладающий собственным именем, возможно, не единственным. Шрифт в этом смысле есть любая электронная либо материальная реализация гарнитуры, т.е. файлы TrueType и Type1, отлитые литеры и т.д.

Геодезия – (греч. *geödaisia*, от – *geö* –Земля и *daisia*– делю, разделяю)

Наука об определении фигуры, размеров и гравитационного поля Земли и об измерениях на земной поверхности для отображения её на планах и картах, а также для проведения различных инженерных и народно-хозяйственных мероприятий.

Геология (от *гео...* и *...логия*)

Комплекс наук о земной коре и более глубоких сферах Земли. В узком смысле слова – наука о составе, строении, движении и истории развития земной коры и размещении в ней полезных ископаемых. Большая часть прикладных и теоретических вопросов, решаемых геологией, связано с верхней частью земной коры, доступной непосредственному наблюдению.

Геоиконика

Общий язык и метод исследования для картографии, аэрокосмического зондирования ГИС-технологий, телекоммуникационных сетей, а также наук о Земле и смежных с ними социально-экономических наук.

Геоинформатика

Интегрированная сфера знаний, изучающая закономерности возникновения и протекания пространственно-координированных процессов в природе и обществе.

Геоинформационная система (ГИС)

❶Комплекс интегрированных программно-аппаратных средств, обеспечивающих сбор, хранение, обработку, манипулирование, моделирование, анализ и отображение пространственно-координированных данных для поддержки принятия решений. Включает также и персонал, выполняющий все вышеуказанные действия.

❷ГИС является быстро развивающейся технологическим направлением, включающим развитые графические средства, опирающиеся на табулированные данные, предназначенные для решения практических задач (real-world problems).

Геоматика

❶ Сфера деятельности в науке и технике, имеющая дело с использованием информационных технологий и средств коммуникации для сбора, хранения, анализа, представления, распространения и управления пространственно-координированной информацией, обеспечивающей принятие решений;

❷ Суперсистема, охватывающая такие дисциплины, как математика, физика, информатика, картография, геодезия, фотограмметрия и дистанционное зондирование.

Гетерогенность

Только небольшое количество сетей обладает *однородностью (гомогенностью)* программного и аппаратного обеспечения. Однородными чаще являются сети, которые состоят из небольшого количества компонентов от одного производителя. Немногие организации имеют сети, составленные из оборудования, например, только IBM, SPARC (SUN), Macintosh или DEC. Как правило сети *неоднородны (гетерогенны)* и состоят из различных рабочих станций, операционных систем и приложений, а для реализации взаимодействия между компьютерами используют различные протоколы. Разнообразие всех компонентов, из которых строится сеть, порождает еще большее разнообразие структур сетей, получающихся из этих компонентов.

Гиперссылка, гипертекст, гипермедиа (hyperlink, hypertext, hypermedia) (линк, link)

Строка в html - документе, указывающая на фрагмент любого другого файла, который может быть расположен в Internet, и содержащая полный путь (URL) к этому файлу. Гиперссылками могут быть графическое изображение или слово, фраза или текст на странице сайта или в письме электронной почты, снабжённые соответствующими адресами, щёлкнув на которых мышью можно загрузить (другую), связанную с ними Web - страницу. Таким образом, гиперссылка есть связь между одним элементом документа - словом, фразой,

символом или изображением - и другим элементом этого же или другого документа. Гиперссылки также называют "горячими ссылками" или "гипертекстовыми ссылками". (См. также *HTML).*

Гипертекст (hypertext)

Слово или фраза в документе, которая связана с какой – нибудь частью этого или другого документа. Слова и фразы гипертекста обычно имеют голубой цвет и подчеркнуты. Часто под гипертекстовыми страницами подразумевают HTML-документы.

Глобализация

Процесс распространения информационных технологий, продуктов и систем по всему миру, несущий за собой экономическую и культурную интеграцию. Сторонники этого процесса видят в нем возможности дальнейшего прогресса при условии развития глобального информационного общества. Оппоненты предупреждают об опасностях глобализации для национальных культурных традиций.

Глобальная информационная инфраструктура (ГИИ)

Качественно новое информационное образование, формирование которого начала в 1995 году группа развитых стран мирового сообщества. По их замыслу ГИИ будет представлять собой интегрированную общемировую информационную сеть массового обслуживания населения нашей планеты на основе интеграции глобальных и региональных информационно-коммуникационных систем, а также систем цифрового телевидения и радиовещания, спутниковых систем и подвижной связи.

Глобальный

❶ Об объекте программы (идентификаторе переменной, константе и др.), описанный на внешнем уровне и доступный всем компонентам программы.

❷ О методе, применяемому ко всему объекту в целом.

Глоссарий

Еще в рукописях XI века (на полях или в самом тексте) можно встретить пояснения непонятных слов, чаще всего иноязычных или вышедших из употребления. Эти пояснения назывались *глоссами*, а собрания *глосс*, так называемые *глоссарии*, представляли собой первые небольшие *словарики*. Различают собрание глосс (непонятных слов или выражений) с толкованием (толковый глоссарий) или переводом на другой язык (переводной глоссарий). Существуют глоссарии к отдельным произведениям или к циклу. Например, глоссарий к Ведам, 1-е тыс. до н. э., к произведениям Гомера начиная с 5 в. до н. э. и т.д.

Гомогенность (однородность)

Однородными (гомогенными) чаще всего являются сети, которые состоят из небольшого количества компонентов программного и аппаратного обеспечения от одного производителя.

Грамматика (греч. grammatikȅ, от grámma – буква, запись)

Раздел лингвистики, изучающий строй языка (т.е. законы, по которым соединяются друг с другом в потоке речи языковые единицы – морфемы, словоформы, словосочетания и предложения), грамматические значения (обязательно выражаемые) и способы их выражения. Грамматику обычно подразделяют на *морфологию* (строение и классификация словоформ) и *синтаксис* (строение и классификация словосочетаний и предложений). Грамматика может быть описательной (констатирующей), исторической, сравнительной, сопоставительной, нормативной и т.д.

Грамотность

Определенная степень владения навыками чтения, письма в соответствии с грамматическими нормами родного языка. Применительно к характеристике населения – один из базовых показателей его социально-культурного развития. Конкретное содержание понятия грамотности исторически изменчиво, имеет тенденцию к расширению с ростом общественных требований к развитию индивида: от элементарных умений читать, писать, считать - к владению некоторым комплексом различных общественно необходимых знаний и

навыков, позволяющих человеку сознательно участвовать в социальных процессах (т. н. функциональная грамотность).

Граница

❶ Число, характеризующее минимальное или максимальное значение индексов в описании данных в виде массивов.

❷ Линия раздела между двумя административными единицами, владениями, областями. То есть линия, разделяющая территории государств – рубеж. Например, турецкая граница. (граница с Турцией).

❸ Предел, конец; допустимая норма. К примеру, "Нашим скитаниям не видно границ".

- Д -

Данные (*см. BI*)

❶ Зарегистрированные сигналы или факты.

❷ Форма существования и представления информации.

❸ Информация, представленная в виде пригодном для обработки автоматическими средствами при возможном участии человека. (ГОСТ).

❹ Представление фактов, понятий или команд в формализованном виде, удобном для интерпретации человеком или автоматически.

❺ Любое представление, дискретное или аналоговое, которому приписано или может быть приписано какое-либо значение.

❻ Информация, подготовленная для определенных целей (при этом часто подразумевается определенный *формат*).

❼ Информационный и стратегический ресурс организационных структур различного уровня.

❽ В вычислительной технике **термин «*данные*» имеет три различных значения:**

1) *Данные* – как объекты, отличные от команд. Подразумеваются все обрабатываемые программой операнды. Например, значения констант и переменных, файлы *данных* (в противоположность программным файлам). Однако приходится учитывать контекст: например, команды на исходном языке являются *данными* для компилятора, а результирующий объектный код – *данными* для компонующего загрузчика. Когда же начинается выполнение, тот же самый объектный код становится программой.

2) Слово *данные* в контексте отдельной программы или пакета программ может использоваться в более узком смысле, означая входные данные, в противоположность результатам (выходным данным), как, например, в случае подготовки и проверки данных. Вместе с тем, результаты, полученные при выполнении одного процесса, почти всегда являются данными для следующего процесса.

3) Когда говорят *данные*, часто подразумевают (особенно в последнее время) нечто отличное от текста, речи и изображений. Аналогичным образом обработка данных противопоставляется обработке текста, обработке речи и обработке изображений. При таком употреблении термина подчеркивается высокая форматированность данных в традиционных приложениях обработки данных, в противоположность более свободным структурам, используемым для представления текста (например, на естественном языке), при передаче речи или в процессе обработки визуальных изображений.

❾ Отдельные фрагменты информации, обычно форматируемые специальным образом для дальнейшего использования в соответствующих обрабатывающих программах.

Всё программное обеспечение (software) делится на две основные части: ***данные*** и ***программы***. При этом программы представляют собой наборы команд (инструкций) для манипулирования данными. Данные могут существовать во многих формах – в виде чисел или текста, размещаемых на листах бумаги, в виде битов и байтов, запоминаемых в электронной памяти или в виде фактов, запоминаемых в памяти человека. Строго говоря, данные являются множеством исходных фактов или отдельными фрагментами информации.

❿ Термин данные часто используется для того, чтобы отличать двоичную, читаемую компьютером информацию, от текстовой, воспринимаемой и удобной для чтения человеком. Разные приложения разрабатываются и настраиваются на работу с теми или другими видами данных и поэтому могут читать либо двоичные файлы данных (содержащие двоичные цифры 0 и 1), либо текстовые файлы (содержащие данные в кодах ASCII).

Данные-информация-знания (Data – Information - Knowledge)

Данные – факты, зарегистрированные с помощью различных носителей.
Информация – нет универсального определения. Используется и как синоним знаний, и как синоним данных. Однако есть специфика, лучше всего выражаемая через глагол «информировать», т.е. сообщать что-то новое. Получить информацию значит получить ответ на какой-то вопрос. Можно получить информацию и не имея вопроса, в этом случае сообщение будет информацией, если оно меняет сложившуюся у потребителя картину мира. Знания – результат познавательной деятельности человека.

Данные конечного пользователя (End User Data)

❶ Данные, отформатированные для обработки запросов конечного пользователя.

❷ Данные, создаваемые конечными пользователями.

❸ Данные, предоставляемые Хранилищем данных.

Данные пространственные (Spatial Data)

Любой тип данных, который включает формальную пространственную привязку – как, например, геодезическая сеть. К этой категории относятся как данные дистанционного зондирования, так и информация с карт.

Двоичный код

В цифровой технике способ представления данных (чисел, слов и других) в виде комбинации двух знаков, которые можно обозначить как 0 и 1. Знаки или единицы двоичного кода называют битами. Одним из обоснований применения ДК является простота и надежность накопления информации в каком-либо носителе в виде комбинации всего двух его физических состояний, например в виде изменения или постоянства магнитного потока в данной ячейке носителя магнитной записи. Наибольшее число, которое может быть выражено двоичным кодом, зависит от количества используемых разрядов, т.е. от количества битов в комбинации, выражающей число. Например, для выражения числовых значений от 0 до 7 достаточно иметь 3-разрядный или 3-битовый код:

Двоичный поиск

Алгоритм поиска, в котором элемент отыскивается путем последовательного деления упорядоченного списка пополам и просмотра той половины, которая должна содержать элемент.

Двухрядный коннектор (double-row connector)

Двухрядный соединитель, двухрядный разъем.

Дебаггер (debugger)

Отладчик, программа отладки, отладочная программа. Программа, которая помогает локализовать и исправлять ошибки в выполняемых программах. Когда в работе программы обнаруживается ошибка, *дебаггер* показывает позицию в исходном коде (модуле), в случае, если он является частью интегрированной среды проектирования. Если дебаггер является отдельной программой, то в случае ошибки он показывает вызвавшую её строку в дезассемблированном коде выполняемой программы.

Девелопер (developer) (жарг.)

❶ Относится к любому лицу, включённому в процесс разработки компьютерных игр. Это может относиться к любому сотруднику компании, которая производит игры. Одновременно термин может определять лицо, непосредственно вовлечённое в процесс разработки игры: артиста, разработчика, программиста, музыканта и др.

❷ Является синонимом понятия программист.

Дерево

❶ Конечное множество, в котором выделен один элемент (корень), а остальные элементы разбиты на непересекающиеся множества (поддеревья), каждое из которых является деревом.

❷ Ориентированный граф, в котором имеется ровно одна вершина (корень дерева), не имеющая входящих рёбер, а в каждую из остальных вершин входит ровно одно ребро.

Диаграмма

❶ Графическое представление множества (collection) элементов, обычно изображаемое в виде связного графа из вершин (сущностей) и ребер (отношений). Иначе говоря, система представляет собой разрабатываемую сущность, которая рассматривается с разных точек зрения с помощью моделей, многообразные представления которых отображены в форме диаграмм. Язык UML поддерживает следующие девять типов диаграмм: диаграммы классов, диаграммы объектов, диаграммы прецедентов (use case), диаграммы последовательностей (sequence), диаграммы сотрудничества (collaboration), диаграммы состояний (state), диаграммы видов деятельности (activity), диаграммы компонентов (component) и диаграммы развёртывания (deployment). (UML).

❷ Графическая проекция элементов, составляющих систему.

❸ Графическое представление функциональных зависимостей или числовых последовательностей.

Диалоговое приложение (См. *Приложение диалоговое*)

Дигитализация, Оцифровка - синоним.

❶ Перевод информации в цифровую форму.

❷ Более технологическое определение: Цифровая трансмиссия данных, закодированных в дискретные сигнальные импульсы.

Дилер (Dealer)

Финансовая компания или частное лицо, занимающееся куплей и продажей ценных бумаг или товаров. Вознаграждение дилера - разница между ценой покупки и продажи, агента (брокера) - комиссионные.

Динамическая страница (dynamic page)

В отношении к HTML-странице это страница элементов данных, которые сгенерированы из базы данных, но сама страница формируется «на лету», в процессе обращения к базе данных.

Динамические запросы (Dynamic Queries)

Динамически созданный SQL, обычно создаваемый настольными программами формирования запросов. Запросы не обрабатываются заранее, их подготовка и выполнение происходит во время работы.

Динамические языки высокого уровня

Языки программирования, которые имеют мощные средства контроля данных во время выполнения программы. К ним относятся: Lisp (основанный на структурах списков), Prolog (основанный на алгебре логики), Smalltalk (основанный на объектах).

Директорий (см. каталог).

Дисковод

Внешнее устройство, предназначенное для ввода информации с магнитных дисков в память компьютера.

Дискретные (системы)

Системы, в которых регистрируемые, передаваемые и отображаемые сигналы могут представлять данные в дискретном виде (т.е. как целые числа).

Дистанционное обучение

Новый способ реализации процесса обучения, основанный на использовании современных информационных и телекоммуникационных технологий, позволяющих осуществлять обучение на расстоянии без непосредственного, личного контакта между преподавателем и учащимся.

Дистанционное образование

❶ Целенаправленное и методически организованное руководство учебно-познавательной деятельностью лиц, находящихся на расстоянии от образовательного центра, осуществляемое посредством электронных и традиционных средств связи.

❷ Процесс получения знаний, умений и навыков с помощью специализированной образовательной среды, основанной на использовании информационно-коммуникационных технологий (ИКТ), обеспечивающих обмен учебной информацией на расстоянии, и реализующей систему сопровождения и администрирования учебного процесса.

Добыча данных (Data Mining)

Технические приемы, использующие программные инструменты, предназначенные для такого пользователя, который, как правило, не может заранее сказать, что конкретно он ищет, а может указать лишь определенные образцы и направления поиска. Data mining - это процесс просеивания большого объема данных для определения отношений между данными. Также известно как "скольжение по данным" (data surfing).

Документ (document)

❶ Любые данные, которые могут быть представлены в цифровой форме. (W3C).

❷ Набор текстовых и/или графических данных, организованных и форматированных для прямого восприятия человеком. Документ может иметь вид печатных страниц или находиться в цифровом виде в форме скомпонованных изображений страниц.

❸ Совокупность данных в памяти компьютера, предназначенная для восприятия человеком с помощью соответствующих программных и аппаратных средств.

❹ Физическая сущность, имеющая любой смысл и содержащаяся в записанных на носителях одной или нескольких взаимосвязанных частях. Основными характеристиками документа являются: содержание, представление и структура.

❺ Среда, в которой информация доступна для коммуникации.

❻ Набор пользовательских интерфейсов, интерпретируемых приложением.

❼ Совокупность данных, создаваемая и редактируемая некоторым приложением.

❽ Файл, содержащий некоторый документ.

Документирование

Фиксация документов на носителях (бумаге, магнитных или других типах), обеспечивающая их хранение и возможность воспроизведения. (ГОСТ).

Домен (реляционной базы данных)

Семантическое понятие. Домен можно рассматривать как подмножество значений некоторого типа данных имеющих определенный смысл. Домен характеризуется следующими свойствами:

- Домен имеет *уникальное имя* (в пределах базы данных).
- Домен определен на некотором *простом* типе данных или на другом домене.
- Домен может иметь некоторое *логическое условие*, позволяющее описать подмножество данных, допустимых для данного домена.
- Домен несет определенную *смысловую нагрузку*.

Например, домен D, имеющий смысл "возраст сотрудника" можно описать как следующее подмножество множества натуральных чисел:

$$D = \{n \in \mathbb{N} : n \geq 18 \text{ and } n \leq 60\}$$

Если тип данных можно считать множеством всех возможных значений данного типа, то домен напоминает подмножество в этом множестве.

Драйвер (driver)

❶ Программа, управляющая работой внешнего устройства (мышь, клавиатура, принтер и т.д.). Драйвер обычно является интерфейсом между программами ввода-вывода операционной системы и конкретным устройством (принтером, дисководом, дисплеем и т.д.). Каждое внешнее устройство характеризуется своим уникальным интерфейсом, согласование которого с операционной системой осуществляет драйвер. Наиболее характерным примером драйвера служит программа KEYRUS.COM, которая кириллизирует клавиатуру и монитор для обеспечения русскоязычного интерфейса пользователя с, в целом, англоязычным персональным компьютером.

❷ Программный компонент, позволяющий компьютерной системе взаимодействовать с устройством. Драйвер принтера, например, преобразует поступающие от компьютера данные в форму, понятную конкретному принтеру. Обычно драйвер, кроме того, управляет аппаратурой.

- Ж -

Жизненный цикл (ЖЦ) программы

Последовательность этапов, проходимых каждой программой от начала её зарождения до полной утилизации. Для описания ЖЦ существует несколько моделей. Под моделью ЖЦ понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ. Модель ЖЦ зависит от специфики информационной системы (ИС) и специфики условий, в которых последняя создается и функционирует. Стандарт ISO/IEC 12207 не предлагает конкретную модель ЖЦ и методы разработки ПО. Его регламенты являются общими для любых моделей ЖЦ, методологий и технологий разработки. Стандарт ISO/IEC 12207 описывает структуру процессов ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить действия и задачи, включенные в эти процессы.

К настоящему времени наибольшее распространение получили следующие две основные модели ЖЦ:

–каскадная модель (70-85 г.г.);

–спиральная модель (86-90 г.г.).

- З -

Загрузка операционной системы (boot)

Копирование компонентов операционной системы с внешнего носителя информации (с жёсткого или гибкого диска) в оперативную память и запуск её в работу.

Загрузчик

Обслуживающая программа, обеспечивающая начальную загрузку части программы или ядра операционной системы.

Задача

❶ Вопрос, требующий разрешения, то есть то, что задано для решения.

❷ Математический вопрос, для разрешения которого требуется путём вычислений найти какие-либо величины.

❸ Арифметическая, алгебраическая задача.

❹ Цель, то, что необходимо осуществить, чего необходимо достигнуть.

❺ Поручение, как заданная кому-либо цель.

❻ (старинное) Удача, успех, счастье (противоположное – незадача).

Запоминающее устройство (ЗУ) (storage unit)

Устройство, реализующее функции памяти данных.

Запоминающее устройство на магнитной ленте

Запоминающее устройство, в котором в качестве запоминающей среды используется магнитная лента, размещаемая в съёмных кассетах (картриджах).

Запоминающее устройство на магнитных дисках (Hard Disk Drive – HDD)

Запоминающее устройство, в котором в качестве запоминающей среды используется поверхность вращающегося диска или пакета дисков с нанесённым магнитным покрытием.

Запрос(query)

❶ Операция, которая возвращает значение, но не изменяет состояния системы. (UML).

❷ Операция, не имеющая побочных эффектов. (UML).

Знания(в философском понимании)

❶ Отражение семантических (смысловых) аспектов реальности в мозгу человека или в технической системе.

❷ Форма существования и систематизации результатов познавательной деятельности человека. Выделяют различные виды знания: обыденное ("здравый смысл"), личностное, неявное и др. Научному знанию присущи логическая обоснованность, доказательность, воспроизводимость познавательных результатов. Знание объективизируется знаковыми средствами языка.

❸ Постижение действительности человеком. Процессы получения, обоснования, проверки и распространения знания изучаются логикой, методологией, теорией познания, науковедением, социологией.

❹ Проверенный практикой результат познания действительности, верное ее отражение в мышлении человека, обладание опытом и пониманием, которые являются правильными и в субъективном, и в объективном отношении, на основании которых можно построить суждения и выводы, кажущиеся достаточно надежными для того, чтобы рассматриваться как знание.

❺ Знания – это изменчивая смесь практического опыта, индивидуальных ценностей, контекстной информации, интуиции экспертов, обеспечивающая базовую структуру для оценки и объединения нового опыта и новой информации. Знания появляются и обретают практический смысл в сознании экспертов. В организации знания запечатлены не только в базах данных и репозиториях, но и в укладе организации, ее процессах, правилах и нормах. (Томас Давенпорт и Лоуренс Прусак).

Значение

❶ Содержание, связываемое с тем или иным выражением (слова, предложения, знака и т.п.) некоторого языка. Значения языковых выражений изучаются в языкознании, логике и семиотике.

❷ (значение переменной в языке программирования) Константа, сопоставленная с именем переменной.

- И -

Идентификатор

❶ Символьное имя ячейки или области памяти.

❷ Строка символов, обозначающая или именующая объект программы или вычислительной системы.

❸ Литерная цепочка, выступающая в определённом контексте в роли символа или имени. (ГОСТ).

❹ Неделимая последовательность символов алфавита, образующая имя объекта, который используется. Идентификатор одновременно представляет:

–имя объекта;
–адрес (место) в памяти;
–тип объекта;
–размер занимаемого объектом места в памяти в байтах.

Идентификация сущностей (Entity Identification)

Идентификация сущностей предметной области. Идентификация сущностей - это процесс соотнесения сущностей данных с уникальными элементами данных, с которыми те могут идентифицироваться.

Иерархия (hierarchy)

Многоуровневая организация; древовидная организация.

Измерение

Приписываемое наблюдению число, которое отражает величину или значение некоторой характеристики.

Иконка (значок) ***(См. пиктограмма)***

Неотъемлемый атрибут любой кнопки или файла в операционной системе Windows, позволяющий легко распознать тип объекта либо конкретный объект. Более точно тип файла определяется по его расширению (.DOC, .EXE и т.д.). Значки могут храниться в отдельных файлах с расширением .ICO, в программных файлах (.EXE), в динамически формируемых библиотеках (.DLL) и т.д.

Импульс

❶ Мера механического движения (то же – что количество движения). Импульсом обладают все формы материи, в т. ч. электромагнитные и гравитационные поля.

❷ Импульс силы – мера действия силы за некоторый промежуток времени; равен произведению среднего значения силы на время ее действия

❸ Импульс волновой – однократное возмущение, распространяющееся в пространстве или среде. Например: звуковой импульс – внезапное и быстро исчезающее повышение давления; световой импульс (частный случай электромагнитного) – кратковременное (~ 0,01 с.) испускание света источником оптического излучения (см. *импульс электрический*).

Импульс (электрический)

Кратковременное отклонение напряжения или тока от некоторого постоянного значения.

Имя (сущности)

❶ Символическое представление сущности. Сущность может быть объектом или некоторым действием, которое выполняется. Сущность может иметь большое количество наименований. Каждое имя имеет смысл только в пределах некоторого именного контекста.

❷ Слово, используемое для обозначения объекта для отличия этого объекта от других ему подобных.

Именование

Является одной из самых важных и наиболее часто обсуждаемых проблем в информатике. В вычислениях это означает определение пространства имён, взаимодействующих при вычислениях.

Инженерия программного обеспечения (software engineering)

Сутью данной технической дисциплины является создание спецификации требований, разработка, модификация и сопровождение систем программного обеспечения (программных систем). При этом разработчики программного обеспечения используют теорию и методы компьютерных наук для успешного решения разнообразных нетривиальных задач. Включает широкий спектр различных средств, методов и технологий проектирования и построения больших масштабируемых программных систем. (См. компьютерная наука (computer science)).

Инкапсуляция (encapsulation)

Методика, в которой программный компонент реализует определённую часть функциональности приложения, предоставляя набор методов и свойств для доступа к этой функциональности. Инкапсуляция локализует все детали реализации в пределах одного компонента. Если потребуется изменение функциональности, то они ограничатся изменениями только данного компонента.

Инкрементный (incremental)

Процесс разработки моделей UML, при котором создание диаграмм осуществляется пошагово (поэтапно). (UML).

Инсталляция

❶ Установка, настройка с заданием параметров и указание состава компонентов программной системе для работы на конкретной вычислительной машине при её развёртывании.

❷ Процесс разворачивания (установки) программного продукта на компьютере под управлением операционной системы. Это предполагает прописывание соответствующих данных в установочные области ОС для правильного функционирования и взаимодействия продукта с комплексом имеющихся в системе программно-аппаратных средств.

Инструкция (см. *предложение*)

Предложение, специфицирующее операцию и значения её операндов или адреса тех ячеек, в которых они хранятся.

Инструментальные программные средства (software tools)

Программы, которые используются в ходе разработки, корректировки или расширения других программ. Обычно набор таких программных средств обеспечивает удовлетворение только самых необходимых потребностей и в самом общем случае может состоять из текстового редактора, компилятора, динамического загрузчика и каких-либо средств отладки. Такие инструментальные программные средства могут оказывать помощь во всех видах деятельности на всех стадиях жизненного цикла программного обеспечения, включая управление разработкой и обеспечение качества.

Интеллектуальные информационные технологии

Под интеллектуальными информационными технологиями обычно понимают такие информационные технологии, в которых предусмотрены следующие возможности:

- наличие баз знаний, отражающих опыт конкретных людей, групп, обществ, человечества в целом, в решении творческих задач в выделенных сферах деятельности, традиционно считавшихся прерогативой интеллекта человека (например, такие плохо формализуемые задачи, как принятие решений, проектирование, извлечение смысла, объяснение, обучение и т. п.);
- наличие моделей мышления на основе баз знаний: правил и логических выводов; аргументации и рассуждения; распознавания и классификации ситуаций; обобщения и понимания и т. п.;
- способность формировать вполне четкие решения на основе нечетких, нестрогих, неполных, не доопределенных данных;
- способность объяснять выводы и решения, то есть наличие механизма объяснений;
- способность к обучению, переобучению и, следовательно, к развитию.

Интеллектуальный агент (Intelligent Agent)

Программный механизм, работающий в фоновом режиме и срабатывающий только при наступлении отдельного события. Например, агенты могут передавать суммарные файлы в первый день месяца или отслеживать входящие данные и выдавать сигнал для пользователя при возникновении определенных транзакций.

Интеллектуальный анализ данных (ИАД) (см. *Data Mining*).

Интеллектуальный интерфейс (См. *интерфейс интеллектуальный*)

Интерактивный (диалоговый)

Режим, в котором пользователь задаёт программе команды и вводит данные во время её работы. Такой режим обычно предполагает обмен текстовыми командами (запросами) и ответами (приглашениями).

Интернет (Internet)

Высокоскоростная оптоволоконная сеть, объединяющая все остальные сети нижележащих уровней (национальные, региональные, *WAN, LAN и др.*) по всему миру и использующая для передачи данных транспортный протокол *TCP/IP*. Служит средством коммуникации пользователей посредством использования *e-mail*, средств *передачи данных* и *файлов программ* с применением протокола *FTP*, а также поиска информации в *World Wide Web*. Кроме того, Интернет обеспечивает удалённый доступ к компьютерным системам с целью использования программных компонентов в распределённых вычислениях, работы с он-лайн'овыми электронными каталогами и базами данных средствами технологии коммутации пакетов (packet switching). Интернет был основан в 1969 году под эгидой проекта министерства обороны США *ARPAnet* и представляет (в отличие от *World Wide Web) только средства коммуникации*, то есть линии связи и сопутствующие им аппаратные средства: маршрутизаторы, хабы, переключатели и др.

Интернет адреса (Internet address)

Уникальные коды, присваиваемые конкретным компьютерам, подключённым к Интернету для идентификации их в качестве отсылающих и получающих данные и файлы программ. Существуют две категории используемых адресов: адреса электронной почты (*e-mail*) отдельных личностей (к примеру, *presleyelvis@aol.com*) и *URL* или *FTP* сайты, сайты *Telnet* и *Web* сайты (к примеру, *www.aol.com*). Форма и формат Интернет адресов регулируется службой Системы Доменных Имён (Domain Name System, DNS).

Интернет-коммерция, торговля в Интернете

Коммерческая деятельность в Интернете, когда процесс покупки/продажи товаров или услуг (весь цикл коммерческой /финансовой транзакции или ее часть) осуществляется электронным образом с применением Интернет-технологий. Выделяют несколько основных классов систем для электронной коммерции:

Business-to-Business (B2B) — Бизнес-Бизнес. Взаимодействие одного бизнеса с другим (организация поставок, обмен документацией, заказы, финансовые потоки, координации действий, совместные мероприятия).

Business-to-Customer (B2C) — Бизнес-Потребитель. Взаимодействие продавца и покупателя (приобретение клиентом товаров, услуги, получение консультаций, приобретение страховок и пр.)

Выделяют также и другие категории:

Business-to-Partners (B2P) — Бизнес-Партнеры. Взаимодействие с филиалами и партнерами, совместные предприятия и общение с поставщиками услуг.

Business-to-Employee (B2E) – Бизнес-Персонал. Использование информационных технологий в сфере взаимоотношений с персоналом.

Интероперабельность (interoperability)

❶ Взаимная возможность/способность информационных систем или компьютеров обмениваться сообщениями, выполнять программы или пересылать данные между их разными функциональными блоками таким образом, чтобы пользователь при этом практически ничего не должен был бы знать об особенностях этих блоков. Поддерживается средствами развитых многоуровневых интерфейсов.

Два компонента *X* и *Y* (см. рис.1) могут интероперировать (т.е. являются интреоперабельными), если *X* может посылать запрос *R* для получения сервисов *Y*, базируясь на взаимном понимании сообщения *R* элементами *X* и *Y*, а *Y* может подобным образом возвращать взаимно понимаемые сигналы *S* компоненту *X*.

Это означает, что две интероперабельные системы могут взаимодействовать для выполнения совместно решаемых задач. В географической сфере интероперабельностью является способность информационной системы:

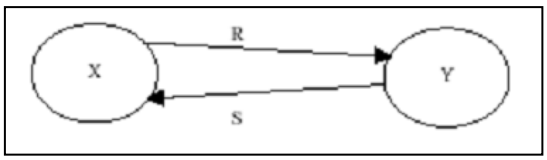


Рисунок 1. Интероперабельность двух систем

1)свободно обмениваться любыми видами пространственной информации о Земле, а также связанных с нею объектами и другими характеристиками, находящимися над её поверхностью и под ней;

2)совместно, посредством сетей, запускать и выполнять программные средства, способные манипулировать такой информацией.

Важной составляющей интероперабельности являются одинаковые форматы сообщений.

❷ Обеспечение согласования верхних уровней стека коммуникационных протоколов компьютерных сетей, реализуемых серверами и редиректорами операционных систем и некоторыми сетевыми приложениями.

Интерпретация

❶Самый простой способ создания переносимых и безопасных программ. Используется для обработки элементов в языках сценариев.

❷Выполнение в режиме интерпретации.

Интерпретатор

Программа контейнер, анализирующая команды или операторы подаваемой на её вход другой программы и немедленно выполняющая их.

Интерфейс

❶ Область, где встречаются и взаимодействуют друг с другом две независимые системы.

❷ Связь между двумя объектами, например прикладной программой и операционной системой, компьютером и модемом. Связь между компьютером и человеком называется пользовательским интерфейсом. Он определяет способ взаимодействия человека с компьютером и включает набор команд и правил их использования.

❸ Граница между компонентами, через которую они взаимодействуют. (W3C).

❹ Логическое группирование операций. Интерфейс представляет абстрактный тип сервисов, независимый протокол передачи и формат данных. (W3C).

❺ Средство сопряжения двух систем или частей систем, в которых все физические, электрические и логические параметры отвечают предварительным соглашениям и широко применяются в других устройствах. (Иллингуорт).

❻ Совокупность информационно-логических, электрических и конструктивных требований, выполнение которых обеспечивает работоспособное сопряжение различных компонентов системы.

❼ Совокупность средств и правил, обеспечивающих логическое или физическое взаимодействие устройств цифровых вычислительных систем или программ (ГОСТ).

❽ *Программный* интерфейс определяет совокупность допустимых процедур или операций и их параметров, список общих переменных, областей памяти или других объектов.

❾ *Физический* интерфейс определяет: тип стыка, уровни сигналов, импеданс, синхронизацию и другие параметры канала связи.

❿ *Системный* интерфейс (СИ) представляет комплекс средств сопряжения центрального процессора, оперативной памяти и внешних устройств. СИ представляет собой совокупность унифицированной магистрали для передачи информации, электронных схем, а также унифицированных алгоритмов (протоколов) обмена информацией между отдельными устройствами ЭВМ.

❶❶ Сочленение ожидаемого поведения и ожидаемых обязанностей, а также семантических связей с другими интерфейсами, дающее программистам и разработчикам конкретную сущность для использования при ссылке на осуществление необходимой связи. (COM).

❶❷ Именованное множество операций, описывающих поведение элемента и используемых для определения услуг класса или компонента. (UML).

❶❸ Точка, в которой два элемента соединяются между собой, чтобы иметь возможность работать друг с другом. В компонентной объектной модели под интерфейсом понимается группа взаимосвязанных функций, предоставляющих доступ к COM-объектам. Набор интерфейсов определяет контракт, позволяющий объектам взаимодействовать согласно спецификации COM.

❶❹ (существительное) Пользовательский интерфейс, состоящий из набора кнопок, списков команд (меню), команд операционной системы, форматов графических дисплеев и других устройств, обеспечиваемых компьютером или программой, которые обеспечивают пользователю возможность использовать и общаться с компьютером или программой. Графический интерфейс пользователя (GUI) обеспечивает использующим его более или менее "образно ориентированный" (picture-oriented) способ взаимодействовать с новыми техническими средствами и технологиями. GUI обычно более удобен или дружественен при работе с компьютерными системами.

❶❺ Как глагол, *соединять (связывать)* означает взаимодействие с другим пользователем или объектом. По отношению к аппаратным средствам интерфейс означает создание физического соединения, позволяющего двум частям оборудования сообщаться или работать вместе эффективно.

Интерфейс интеллектуальный (UIU, Intellectual User Interface)

Средство преодоления проблем, которые не в состоянии разрешить традиционные интерфейсы (типа WIMP, Windows-Icons-Menus-Pointing device). Интеллект в UIU должен сделать систему адаптивной к пользователю, допуская диалоги между пользователями и системой и представлять информацию в интегрированном исчерпывающем виде, используя несколько модальностей.

Интерфейс командной строки (Command line interface)

Разновидность интерфейса операционной системы, в которой пользователь вводит с клавиатуры специальные команды в текстовом виде. Таким интерфейсом характеризуется операционная система MS DOS.

Интранет (Intranet)

Является сетью, которая расположена в пределах предприятия. Он может состоять из многих связанных между собой локальных сетей (LAN), а также использовать арендованные линии в WAN. Он также может включать или не включать соединения через один или несколько шлюзов с внешним Интернетом. Основным назначением Интранета является объединение информации и вычислительных мощностей (средств) предприятия и обеспечения ими его работников. Интранет может также использоваться для обеспечения групповой работы и проведения телеконференций.

Информатизация

Системно - деятельностный процесс овладения информацией как ресурсом управления и развития с помощью компьютерных средств информатики с целью создания информационного общества и на этой основе - дальнейшего продолжения прогресса цивилизации.

По мнению ряда авторов, процесс информатизации включает в себя три взаимосвязанных процесса:

–медиатизацию, процесс совершенствования средств сбора, хранения и распространения информации;

–компьютеризацию, процесс совершенствования средств поиска и обработки информации;

–интеллектуализацию, процесс развития способности восприятия и порождения информации, т.е. повышения интеллектуального потенциала общества, включая использование средств искусственного интеллекта.

Информатика

❶ Система знаний, относящаяся к производству, переработке, хранению и распространению всех видов информации в обществе, природе и технических устройствах. (Философский словарь).

❷ Отрасль науки, изучающая структуру и общие свойства информации, а также вопросы, связанные со сбором, хранением, поиском, переработкой, преобразованием, распространением и использованием её в различных сферах деятельности человека.

Информационная система

❶ Система, предназначенная для решения задач поиска или логической обработки информации. (ГОСТ).

❷ Организационно упорядоченная совокупность документов (массивов документов) и информационных технологий, в том числе с использованием средств вычислительной техники и связи, реализующих информационные процессы.

Информационная среда

Совокупность носителей данных, используемых при какой-либо обработке данных.

Информационно-коммуникационная инфраструктура

Совокупность территориально распределенных государственных и корпоративных информационных систем, линий связи, сетей и каналов передачи данных, средств коммутации и управления информационными потоками, а также организационных структур, правовых и нормативных механизмов, обеспечивающих их эффективное функционирование.

Информационно-коммуникационные технологии (ИКТ)

Совокупность методов, производственных процессов и программно-технических средств, интегрированных с целью сбора, обработки, хранения, распространения, отображения и использования информации в интересах ее пользователей.

Информационное пространство

❶ Интегральное электронное информационное пространство, образуемое при использовании электронных сетей.

❷ Сферы в современной общественной жизни мира, в которых информационные коммуникации играют ведущую роль. В этом значении понятие информационного пространства сближается с понятием информационной среды.

Информационные технологии (information technology, IT)

Технологии разработки, развёртывания, установки и эксплуатации компьютерных систем и приложений (программ). (The American Heritage® Dictionary of the English Language, Fourth Edition. Copyright © 2000 by Houghton Mifflin Company. Published by the Houghton Mifflin Company.) Термин также относится к технологиям, имеющим дело с обработкой информации, хранением и передачей данных. Это включает использование компьютерных технологий и различных коммуникационных (сетевых) технологий (электронных, радиоволновых, оптических и др.). Другими словами, совокупность средств и методов обработки данных и информации (с помощью персональных компьютеров).

Информационные процессы

Процессы создания, сбора, хранения, обработки, отображения, передачи, распространения и использования информации.

Информационные ресурсы (Information resources)

Документы и массивы документов в информационных системах (библиотеках, архивах, фондах, банках данных, депозитариях, музейных хранениях и др.).

Информационные технологии (ИТ), информационно-коммуникационные технологии (Information and Communication Technologies (ICT))

Совокупность методов, производственных процессов и программно-технических средств, интегрированных с целью сбора, обработки, сохранения, распространения, отображения и использования информации в интересах её пользователей.

Информационные услуги Information services

Удовлетворение информационных потребностей пользователей путем предоставления информационных продуктов.

Информационный излишек в киберпространстве (Info-Glut in Cyberspace)

Слишком много данных! (более 30 миллионов электронных почтовых ящиков, более 7000 CD-ROM'ов по 650 Мб, более 5000 оперативных баз данных, более 500 кабельных каналов и т.д.).

Информация

❶ Сведения, неизвестные до их получения, являющиеся объектом хранения, передачи и обработки. (ГОСТ).

❷ Содержание сообщения или сигнала. Сведения рассматриваемые в процессе их передачи или восприятия, позволяющие расширить знания об интересующем объекте.

❸ Сведения о лицах, предметах, фактах, событиях, явлениях и процессах, независимо от формы их представления.

❹ Результат обработки *объективных* данных с помощью *субъективных* методов. При этом из одних и тех же данных можно получать разную информацию в зависимости от используемого метода.

❺ Основные свойства информации:

–информация приносит сведения, об окружающем мире которых в рассматриваемой точке не было до ее получения;

–информация не материальна, но она проявляется в форме материальных носителей**:** *дискретных знаков* или *первичных сигналов*;

–проявляется *только* "в процессе" обработки данных;

–знаки и первичные сигналы несут информацию *только* для получателя способного их распознать.

❻ Исходя из того, что получение информации обязательно должно производить изменение "тезауруса", можно утверждать, что человек получает информацию только в том случае, когда в его знаниях, т.е. в его тезаурусе после получения сообщения произошли какие-либо изменения. И чем больше изменений внесло сообщение в тезаурус приемника, тем большее количество информации он получил из этого сообщения. Изменить же тезаурус – это значит изменить его *смысловыражающие* элементы или *смысловые отношения* между ними. Поскольку информация проявляет себя только в *процессах*, то можно утверждать, что информация есть характеристика и качество этих специфических процессов, а не той или иной структуры данных. Одна и та же структура данных или сигнал в одном случае могут нести информацию, а в другом - нет.

Инфрастуктура (infrastructure)

❶ Обобщающий термин, пришедший из военного лексикона, обобщающий все компоненты, поддерживающие специфическую деятельность в постоянно функционирующих системах и структурах и составляющих их основу.

❷ (в информационных технологиях) Всё аппаратное (hardware) и программное (software) обеспечение, разработанное и эксплуатируемое для поддержания большой или маленькой системы в работоспособном состоянии.

❸ (в геоинформатике) Созданные человеком в окружающей природной среде искусственные сооружения и средства коммуникации, позволяющие функционировать микрорайонам, округам, городам, метрополиям, регионам и государствам.

Искусственный интеллект

Раздел информатики, связанный с разработкой интеллектуальных программ для компьютеров.

Исследование

Способ производства нового знания.

Исследование научное

Процесс выработки новых знаний, один из видов познавательной деятельности. Характеризуется объективностью, воспроизводимостью, доказательностью, точностью. Имеет два уровня - эмпирический и теоретический. Наиболее распространенным является деление исследований на фундаментальные и прикладные, количественные и качественные, уникальные и комплексные.

- К -

Кампус (амер.)

Территория университета, колледжа или школы, университетский городок.

Канал (Channal)

❶ Путь или соединение, по которому данные передаются между двумя устройствами.

❷ Морской канал.

❸ Зубной канал.

❹ Информационный канал.

❺ Телевизионный, радио канал.

❻ Оружейный канал.

❼ Каналом называется физический или логический путь для передачи сигналов. В контексте компьютерных сетей чаще всего встречаются упоминания каналов двух типов: коммуникационных и дисковых. Коммуникационным каналом называется маршрут, по которому происходит передача данных, речи или видеоизображения. Современные технологии передачи данных позволяют организовывать несколько коммуникационных каналов внутри одного физического кабеля. Дисковым каналом, в конфигурации с жестким диском, называются компоненты, посредством которых осуществляется взаимодействие операционной системы с накопителем на жестком диске.

Канал связи

Путь передачи сигналов между двумя или несколькими точками. По назначению различают телефонные, факсимильные каналы, каналы звукового вещания, телевизионные каналы, передачи цифровой информации.

Каротаж

Измерения параметров вдоль оси скважины с использованием скважинной аппаратуры**.** Гамма-каротаж – изучение естественного излучения горных пород в буровых скважинах для выявления радиоактивных руд, литологических расчленений, разрезов и т. п.

Карта (Card)

❶ Печатная плата (карта) адаптера.

❷ Карточка в гипертекстовых системах. Экранное представление картотечной карточки.

❸ Форма документа (медицинская карта).

❹ Географическая карта.

❺ Игральная карта.

Картография (от *карта* и *...графия*)

Наука о географических картах, о методах их создания и использования. С точки зрения на географические карты, как на наглядные образно-знаковые модели пространства, приводит к более строгому определению предмета и метода картографии. Картография – наука об отображении и исследовании пространств, размещения, сочетаний и взаимосвязей явлений природы и общества (и их изменений во времени) посредством картографических изображений, воспроизводящих те или иные стороны действительности. Это определение включает в круг интересов картографии карты небесных тел, звёздного неба, а также глобусы, реальные карты и другие пространственные модели в картографических знаках.

Каталог (Catalog)

❶ Справочник файлов и библиотек со ссылками на их расположение. Каталог может содержать другую информацию, такую как типы устройств, на которых хранятся файлы, пароли, коэффициент блокирования и др.

❷ Совокупность описаний элементов, включающих информацию, достаточную для обеспечения доступа к ним.

❸ Поименованное пространство на диске (каталог более высокого уровня), содержащее другие каталоги и файлы.

❹ (файлов) Каталог, хранящий информацию об имени, объёме, расположении и времени создания или последнего изменения файла. (ГОСТ).

❺ Компонента словаря данных базы данных, содержащая директорию хранения относящихся к нему объектов СУБД и их свойств.

Каталог файлов (синоним: *директорий*)

Каталог, хранящий информацию об имени, объёме, расположении и времени создания или последнего изменения файла. В области хранения и поиска данных, является каталогом файлов, записанных и хранящихся на жёстком или гибком диске компьютера или на любом другом носителе данных, обычно организуемом для облегчения доступа к ним в структуре иерархического дерева поддиректориев. Самый верхний по уровню директорий называется *корневым*.

Категория (греч. kategoria).

❶ (науч.) Высшее родовое понятие, обозначающее какой-либо наиболее общий, отвлеченный разряд явлений, предметов или их признаков. Например, "категория причинности", "категория количества".

❷ (книжн.) Разряд однородных предметов или лиц. К примеру, "он из категории тех людей, которые всегда всем недовольны".

Кванторы (от лат. quantum - сколько)

В логике и математике – логические эквиваленты слов "все", "каждый" и т. п. (кванторы общности), "некоторый", "существует" (кванторы существования) и др. Операторы, формализующие в исчислении предикатов логические свойства этих выражений.

Кегль (кегель)

Размер типографического шрифта, включающий высоту буквы и т.н. заплечики – свободные пространства над и под очком, образующие промежутки между строками в книге, газете и т.п. (см. Литера). Измеряется в пунктах (пункт равен 0.376 мм).

Кибернетика

Наука об общих законах получения, хранения, передачи и переработки информации в машинах, живых организмах и обществе.

Киберпространство (Cyberspace).

❶ Совокупность сервисных средств, доступных через Internet.

❷ Пришедшее из американской жизни понятие, введенное писателем Уильямом Гибсоном в пьесе «Le Neuromacien». Оно описывает виртуальное пространство, в котором циркулируют электронные данные всех компьютеров мира.

Киберсквоттер

Категория людей, занимающаяся самозахватом доменных имён в Web, относящихся к наиболее знаменитым и известным в мире личностям.

КИС (корпоративная информационная система)

Главная роль КИС — поддержать функционирование и развитие предприятия, цель существования которого — получение прибыли за счет некоторой основной деятельности. Сферы деятельности могут быть очень разными — производство, строительство, торговля и др., при этом на верхнем уровне абстракции задачи управления в подобных организациях будут весьма схожими — организовать управление поступающими на вход предприятия ресурсами таким образом, чтобы на выходе получить запланированный (ожидаемый) результат. Это означает, что в основу деятельности предприятия (и его ИС) должен быть заложен некоторый формально описываемый закон управления, позволяющий однозначно сказать, какой бизнес-результат будет получен, если на входе мы имеем определенное воздействие. Согласно современным требованиям корпоративные информационные системы должны включать:

- систему планирования ресурсов предприятия **ERP** (Enterprise Resource Planning);
- систему управления взаимоотношениями с клиентами **CRM** (Customer Relation Management);
- систему управления цепочками поставок **SCM** (Supply Chain Management);
- средства электронной коммерции и взаимодействия через **Интернет** (e-commerce);
- средства аналитики и поддержки принятия решений **BI** (Business Intelligence).

Система управления информацией **IMS** (Information Management System) объединяет все перечисленные системы, позволяет им свободно функционировать и обеспечивает единый интегрированный процесс обработки информации в корпоративной системе. Повышение уровня сложности во взаимодействии информационных потоков привело к понятию EIS — Enterprise Integration/Information System, как сейчас в мировой практике именуют КИС.

Класс

❶ Абстрактное описание данных и поведения ряда похожих объектов. Т.е. класс описывает новый, абстрактный тип данных (АТД). (Тиммоти Бадд).

❷ Определённый тип объектов, задаваемый при помощи описания класса, которое определяет переменные состояния и протокол доступа к объектам данного класса *(в языке C++)*.

❸ Обобщенное абстрактное описание множества однотипных объектов.

❹ Описание сущности, моделируемой в программе.

Классификация

Распределение предметов, объектов и понятий по группам (классам) по обнаруженным свойствам.

Кластер (группа, cluster)

❶ Группа блоков диска, распределяемая как единое целое. В DOS – минимальная единица распределения дискового пространства. Состоит из одного или нескольких соседних секторов. Размер сектора, как правило, кратен степени числа 2. Может иметь значения: 124, 256, 512 или более килобайт.

❷ Система из нескольких компьютеров, соединенных скоростными линиями связи. Для абонентов кластер выглядит как единое целое.

❸ Группа устройств (обычно терминалов) с общим контроллером.

❹ Группа объединённых конструктивно процессоров для повышения скорости решения практически важных задач (метеорологических, ядерных исследований и др.).

❺ Описатель абстрактного типа данных.

❻ В распознавании образов – группа объектов с общими признаками.

❼ Группа процессоров, собранных в единое устройство (обычно суперкомпьютер), для повышения производительности либо для решения сложных, специализированных задач.

Клиент

❶ Программа или компьютер, которые обслуживаются другими программой или компьютером (сервером). В COM моделях – клиентом называется приложение, которое пользуется услугами COM сервера.

❷ В объектно-ориентированном программировании (Object-Oriented Programming) член некоторого класса (class), то есть объект, пользующийся функциями или услугами другого класса или объекта.

❸ Клиент является рабочей станцией или персональным компьютером в клиент/серверной среде (окружении). Или же он представляет один вход в спектре взаимоотношений уровня запрос/доставка между программами.

Клиент автоматизации (automation client)

Приложение, использующее функциональность, предоставляемую сервером автоматизации в рамках модели и технологии COM. Часто называется называемым контроллером автоматизации.

Клиент БД

Так обычно называют пользовательское приложение, которое общается с сервером БД. Модель работы, в которой клиент общается непосредственно с сервером, не используя промежуточных приложений, называется архитектурой клиент/сервер.

Клиент/сервер (client/server)

❶ Архитектура региональных (WAN) и локальных (LAN) вычислительных сетей, обеспечивающая возможность компьютеру ***клиенту*** (обычно рабочей станции или персональному компьютеру) загружать информацию или обрабатывать данные с компьютера ***сервера***, в отличие от систем, которые использовали удалённые терминалы, присоединённые к миникомпьютерам или мэйнфреймам. Обычно клиент выполняет программное обеспечение (ПО) конечного пользователя (front-end software), представляющее собой любую прикладную программу или пакет, способные направлять запросы по сети серверу и обрабатывать получаемую в ответ информацию. Сервер, в свою очередь, получает запросы и предпринимает действия от имени клиента. Промежуточное обеспечение (middleware) предоставляет общий интерфейс для ПО конечного пользователя и сервера, проникающий сквозь слои графического интерфейса пользователя, операционной системы, вычислительной сети и собственных драйверов базы данных с помощью общих вызовов. Для завершения операции сервер базы данных выполняет запрос и передает клиенту затребованные данные для обработки их программой клиента.

❷ Модель вычислений, в которой нагрузка по обработке прикладных программ распределяется между компьютером-клиентом и компьютером-сервером, совместно использующим информацию с помощью сети. Обычно клиент - это программное обеспечение конечного пользователя, выполняющееся на ПК и способное установить связь с сервером (обычно, сервером баз данных). Производительность при использовании модели "клиент/сервер" выше обычного, так как клиент и сервер делят между собой нагрузку по обработке данных.

❸ В компонентных программных технологиях, модель взаимодействия и архитектура, в которой программные объекты, называемые серверами, предоставляют функции и данные объектам, называемым клиентами. Используется в технологиях DLL, COM, DCOM и некоторых других. При этом .EXE-файлы являются локальными серверами, а .DLL-файлы – внутризадачными.

Клиент/серверная модель (client/server model)

Модель клиент-сервер описывает взаимоотношения между двумя программами, из которых одна, клиент, выдаёт запрос на обслуживание другой программе, серверу, который выполняет этот запрос. В сети, модель клиент-сервер обеспечивает удобный способ связать программы, исполняющиеся на разных компьютерах. Например, чтобы проверить ваш

банковский счет, клиентская программа в вашем компьютере направляет запрос к обслуживающей программе банка. Эта программа может, в свою очередь, направить запрос к собственной клиентской программе, которая посылает запрос к серверу базы данных в другом компьютере банка, чтобы подвести ваш баланс на вашем счете. Баланс возвращается клиенту данных банка, который, в свою очередь, переправляет его клиенту в вашем персональном компьютере. Модель клиент-сервер является одной из центральных идей в сетевых вычислениях. Большинство деловых приложений (бизнес-приложений) используют модель клиент-сервер.

Клон (см. *clone*)

Книга

❶ Непериодическое издание в виде сброшюрованных листов печатного материала (объем более 48 страниц). Произведение художественной, научной, общественной литературы, средство массовой, научной и технической информации. Одна из древнейших форм книги - свиток (4 - 3-е тысячелетие до нашей эры), со 2 - 4 вв. заменялся кодексом (современная форма книги в виде книжного блока). Основные материалы для изготовления книги: папирус, со 2 в. до нашей эры - пергамент, с 13 в. в Европе - бумага. В античном мире и в средние века книги размножали переписыванием. Древнейшей печатной книгой считают текст, воспроизведенный ксилографическим путем (от ксило... и...графия, гравюра на дереве) в Корее в период с 704 по 751 гг. Первые опыты книгопечатания были предприняты в Китае в середине 11 в. Би Шэном. Новый период в истории книги связан с И. Гутенбергом (1397-1468), изобретателем европейского способа книгопечатания (середина 15 в.). Станок был изобретён Гутенбергом в 1440 – 1450 гг., что явилось началом эпохи книгопечатания. В Московской Руси первая русская печатная датированная книга "Апостол" была выпущена И. Федоровым и П. Мстиславцем в 1564 г.

❷ Фрагмент программы на языке COBOL.

Код (code)

❶ Код, система кодирования.

❷ Программа, текст программы, код.

❸ Кодировать, программировать, составлять программы.

❹ Набор символов, используемый для кодирования.

❺ Способ преобразования информации, записанной в некотором исходном алфавите (например, русском алфавите) в любой другой (например, двоичный).

❻ Набор условных обозначений для представления информации.

❼ Число, которому приписывают некоторый смысл.

Кодер, кодировщик (coder) (см. *программист, девелопер*)

❶ (жарг.) Программист, кодировщик. Лицо, разрабатывающее, составляющее и тестирующее компьютерные программы.

❷ Программист, составляющий программы по готовым, детальным спецификациям.

Кодирование

Процесс представления информации в виде кода (в том числе и программного).

Кодовая страница

Кодовой страницей называется комплекс кодов попиксельного изображения на экране компьютера конкретных начертаний каждого из 256-и 8-битных алфавитно-цифровых символов, включающих и символы национального языка любой страны мира. Несколько таких наборов для различных режимов работы дисплея называется символьной раскладкой или набором для соответствующей национальной спецификации. При этом таблица кодировки клавиатуры устанавливает, какой код (скэн-код) вырабатывается при нажатии клавиши или комбинации клавиш. Таблица знакогенератора дисплея устанавливает соответствие между скэн-кодом клавиши клавиатуры и кодом кодовой страницы, а соответственно и отображением (изображением) вводимого символа на экране (к примеру, грузинского или китайского). При работе в среде MS DOS используется так называемая

альтернативная кодировка, которая по классификации корпорации Microsoft называется кодовой страницей 866. При работе в среде Windows принята кодировка ANSI 1251. В операционной системе UNIX используется кодировка KOI-8R.

Количество

❶ (филос.) То в вещах и явлениях, что подлежит измерению и счету; одна из основных логических категорий, выражающая ту сторону действительности, которая определяет предмет со стороны его измеримости. Механическая философия сводит качественное многообразие мира к количеству. Количество переходит в качество.

❷ Число, величина, объем, масса. К примеру, количество людей, количество воды.

Команда (instruction, command)

❶ Предписание выполнить некоторое действие.

❷ Сигнал (импульс) – командный стимул в электронных устройствах и биологических системах.

❸ (*внутренняя команда процессора или устройства; синоним: инструкция, командное слово, машинная команда, элементарная операция*).

Предписание, определяющее элементарный шаг выполнения программы работы конкретного устройства, например, запись, считывание, пересылка и т.д. Содержит указание операции, адреса операндов в памяти и другие служебные признаки. Внутренние команды процессора являются основой архитектуры вычислительной системы и, соответственно, компьютерной платформы. Полный набор инструкций, выполняемых процессором, именуется «системой машинных команд».

❹ Оператор программы, предложение языка программирования.

❺ Предложение языка управления заданием.

❻ Предписание компьютеру или устройству выполнить определённую задачу.

Команды имеют различную форму и могут быть:

–специальным (ключевым, зарезервированным) словом, которое понимает программа или система;

–функциональными клавишами или их сочетаниями;

–элементами, выбираемыми из меню;

–кнопками или другими графическими объектами.

Набор команд и правил их использования, называется интерфейсом пользователя и изменяется от программы к программе.

Командный процессор

Часть операционной системы, обрабатывающая команды (предложения или операторы командного языка), вводимые с терминала или из командного файла, и запускающая задачи для их выполнения.

Командный файл (batch file) (синоним – бэт-файл)

Неформатированный текстовый файл, каждая строка которого содержит команду MS-DOS. Называется пакетным и имеет расширение .bat.

Коммуникация (Communication)

В переводе на русский язык это слово может обозначать связь, сообщение, средство связи, информацию, средство информации, а также контакт, общение, соединение.

Коммутация пакетов

Коммутацией пакетов называется методика передачи информации, при использовании которой передача пакетов данных от источника к приемнику осуществляется по общей (разделяемой) среде передачи. Для передачи может использоваться любой доступный маршрут (цепь), который после передачи пакета вновь становится свободным. При передаче следующего пакета может уже использоваться совершенно другой маршрут. Технология коммутации пакетов допускает одновременную передачу нескольких пакетов, относящихся к одному сеансу связи. При этом последовательность прибытия пакетов в узел-приемник может не соответствовать последовательности их передачи узлом-отправителем,

что приводит к необходимости применения средств восстановления исходной последовательности пакетов.

Коммутируемая линия связи

Линия связи, организуемая только на время проведения сеанса связи. Обычно коммутируемые линии связи организуются в телефонной сети.

Компилятор

Программа, переводящая текст программы на языке высокого уровня (C++, Object Pascal и др.) в эквивалентную программу на машинном языке (то есть в двоичные коды машинных команд конкретного процессора соответствующей платформы).

Компонент

❶ Составная часть, элемент чего-либо.

❷ Составная часть распределённого приложения.

❸ Физически заменяемая часть системы совместимая с одним набором интерфейсов и обеспечивающая реализацию какого-либо другого. (UML).

❹ Физическая упаковка логических элементов таких, как классы, интерфейсы и кооперации. (UML).

❺ Компонент является элементом архитектуры с определёнными (заданными) границами.

❻ Предварительно созданный программный объект, который представляет клиентам чётко определённый набор функций. Каждый компонент является самостоятельной отдельной сущностью, которая может быть определена и описана независимо от какого-либо программного пакета. Все COM-объекты являются компонентами. COM-объекты можно писать на любом языке, который поддерживает указатели на указатели, т.е. COM обеспечивает двоичный стандарт взаимодействия.

❼ В языке Java – программный модуль уровня приложения, поддерживаемый контейнером. Компоненты конфигурируются во время развёртывания. Платформа J2EE определяет четыре типа компонента: корпоративные (промышленные–enterprise) компоненты «зёрна» (beans), Web–компоненты, апплеты и приложения клиенты.

❽ Компонент является объектом программного обеспечения, предназначенный для взаимодействия с другими компонентами, инкапсулирующий некоторую функциональность или набор выполняемых функций. Компонент имеет чётко определяемый интерфейс и подчиняется правилам поведения, общим для всех компонентов в данной архитектуре. Таким образом, компонент является абстрактным набором программных инструкций (команд) и внутреннего состояния, которое обеспечивает преобразование данных через его интерфейс.

Компонентно–ориентированное программирование

Компонентно-ориентированное программирование было предложено Никлаусом Виртом в 1987 году. Основная идея заключалась в том, что функционально законченный фрагмент кода (компонент) должен компилироваться, даже в том случае, если ресурсы, на которые он ссылается (другие компоненты), недоступны в период компиляции. Более того, этот компонент должен штатно работать в отсутствие этих ресурсов, если они не требуются в текущем режиме, и выдавать сообщения, только в том случае, если внешние ресурсы нужны. Другими словами, компонентная программа компонуется во время исполнения, а не во время компиляции. В 1989 году Бертран Мейер предложил еще одну общую идею компонентно-ориентированного программирования: рассматривать интерфейс как контракт между вызывающим компонентом и вызываемым компонентом. В идеале такой подход способен привести к появлению компонентов, которые написаны на различных языках программирования и работают на разных платформах (и ОС!), но, тем не менее, способны общаться друг с другом. На практике это реализовано в протоколе SOAP для Web–сервисов.

Компонентное программное обеспечение (component software)

Иногда именуется (*componentware*), программное обеспечение, разработанное для функционирования в виде компонентов в составе более крупных приложений. Ярким

примером применения компонентов является персональный компьютер, который строится из стандартных компонентов: чипов памяти, процессоров, шин, клавиатур, мышей, дисководов и т.д. Так как все интерфейсы между компонентами стандартизированы, существует возможность варьировать компоненты разных производителей в рамках одной системы. Подобным образом, компоненты программного обеспечения стандартизированы по функциям интерфейсов и поэтому они также могут работать совместно в комплексе. Два стандарта OLE и OpenDoc разработаны для оказания помощи программистам при разработке компонентов, которые могут работать совместно. Аналитики считают, что компонентное программирование является естественным продолжением объектно-ориентированного программирования. (Webopedia)

Компоновка модулей

Построение загрузочного модуля из объектных модулей.

Компьютер (computer)

❶ Устройство для ввода, обработки и вывода информации.

❷ Устройство преобразования информации, посредством выполнения задаваемой программой последовательности операций.

Компьютеризация (Computerisation)

Процесс развития и внедрения компьютеров, обеспечивающих автоматизацию информационных процессов и технологий в различных сферах человеческой деятельности.

Компьютерная грамотность (Computer literacy)

Овладение минимальным набором знаний и навыков работы на персональном компьютере. Рассматривается сегодня как мастерство столь же необходимое, как чтение и письмо.

Компьютерная наука (см. *computer science*)

Компьютерная сеть (Network)

Компьютерной сетью называется система объединенных между собой компьютеров, а также, возможно, других устройств, которые называются узлами (рабочими станциями) сети. Все компоненты, входящие тем или иным способом соединены друг с другом и могут обмениваться различной информацией. На узлах сети работает программное обеспечение, которое обеспечивает инициализацию, обслуживание и администрирование сети. Основными компонентами компьютерных сетей являются следующее:

- Узлы: компьютеры и сетевые интерфейсные платы (карты).
- Топология: физическая и логическая.
- Соединительные элементы: кабели, монтажные центры, средства связи и т.п.
- Дополнительные компоненты: периферийные устройства, устройства защиты и инструментарий.

К программным компонентам компьютерных сетей относятся следующее:

- Сетевое программное обеспечение: сетевые операционные системы и программное обеспечение рабочих станций.
- Ресурсы: серверное программное обеспечение и драйверы.
- Инструментальные средства: утилиты, анализаторы, средства сетевого контроля и программы управления конфигурацией.
- Приложения: прикладное программное обеспечение, ориентированное на использование возможностей компьютерных сетей.

Компьютерные технологии

Сочетание программных средств, реализующих функции хранения, обработки и визуализации данных в определённой организационной структуре с использованием выбранного комплекса технических средств.

Конвейер (pipeline)

❶ Цепочка асинхронных процессов, в которой стандартный файл вывода каждого процесса (кроме последнего в цепочке) служит стандартным файлом ввода следующего

процесса в цепочке. Совершенствование конвейерной обработки при исполнении машинных инструкций позволяет выполнять больше инструкций за меньшее число машинных циклов (тактов);

❷ Метод доступа к данным, при котором можно продолжать чтение по предыдущему адресу в процессе запроса по следующему.

❸ Последовательность программ, в которой стандартный вывод каждой программы, кроме самой последней, связан со стандартным вводом следующей программы этой последовательности.

Конечная система (end system)

Конечная система (КС) является сетевым устройством, не выполняющим маршрутизацию или другие функции передвижения информации. Типичная КС включает такие устройства, как терминал, персональный компьютер (ПК) и принтер. Промежуточная система представляет собой сетевое устройство, выполняющее маршрутизацию, другие функции передачи информации и включает маршрутизаторы, коммутаторы и мосты сопряжения локальных вычислительных сетей (ЛВС).

Конечная точка (end point)

Ассоциация (связь) между привязкой (binding) сетевого адреса, специфицированного URI, который может быть использован для коммуникации с экземпляром сервиса. Конечная точка указывает целевое местоположение для доступа к сервису с использованием специального протокола и формата данных.

Конечный пользователь

Физическое лицо, использующее ресурсы в прикладных целях.

Коннектор (программный) (connector)

Коннектор является абстрактным механизмом, который служит для обеспечения связи, согласования или взаимодействия между компонентами. (W3C).

Коннектор (разъем) (connector)

❶ Соединитель; соединительное звено, как правило – многоконтактное.

❷ (логический) Блок объединения (на блок-схеме).

❸ Соединительный знак.

❹ Соединитель многоконтактный, (штепсельный) разъем. Средство соединения взаимозаменяемых частей (компонентов) компьютера. В частности, шестое поколение процессоров Pentium отличается большим разнообразием разъёмов-конструктивов. Одних только коннекторов имеется 4 типа: сокет 8, слот 1, слот 2 и сокет-370. Следует отметить, что в технологиях фирмы Intel, производителя данных марок процессоров, термины слот и сокет употребляются в более широком смысле. Они обозначают спецификацию электрических, программных и механических интерфейсов. В последнем случае имеется в виду число контактов (ножек) процессора и, соответственно, такое же по расположению и числу количество отверстий в приёмной панели на материнской плате ПК.

Коннектор двухрядный (double-row connector)

Двухрядный соединитель, двухрядный разъем.

Консоль

❶ (в вычислительной технике) Устройство ввода-вывода, предназначенное для связи оператора мэйнфрейма (большого компьютера) с её управляющей программой и заданиями. Под управляющей программой обычно подразумевается операционная система.

❷ (в технике) Балка, закреплённая в области одного из концов.

Консольное приложение (См. *Приложение консольное*)

Контейнер

❶ В разработанной Sun Microsystems компонентной архитектуре *JavaBeans* и в компонентной технологии Microsoft Component Object Model (COM), контейнер является прикладной программой или подсистемой, в которой выполняется выстроенный или встроенный блок программы, называемый компонентом (component). К примеру, компонент типа кнопки или другой элемент графического интерфейса пользователя или же маленький

калькулятор может быть выполнен с использованием JavaBeans, который позволяет выполнить их в контейнере Netscape, являющемся браузером либо в контейнерах Microsoft, таких как Internet Explorer, Visual Basic, Excel или Word.

❷ В архитектуре Common Object Request Broker Architecture (CORBA) Interface Repository, в иерархии для структуры метадата (metadata), Контейнер (Container) является одним из трёх абстрактных суперклассов (abstract superclasses) (вместе с IRObject и Contained).

❸ Объект, содержащий один или несколько других объектов. Примером контейнера может служить папка в структуре ОС Windows, предназначенная для хранения документов, папок, рисунков, звуковых-, видео- и других типов файлов.

❹ Сущность, которая обеспечивает жизненный цикл управления, безопасности, развёртывания и сервисы при выполнении компонента. Каждый тип контейнера также обеспечивает компонентно-конкретизируемые сервисы (например, EJB–Enterprise Java Beans, Web, JSP–Java Server Pages, сервлет, апплет или приложение клиент).

Контекст

❶ Окружение системы (т.е. сущности, находящиеся вне системы и взаимодействующие с ней, составляют её контекст). (UML)

❷ Фрагмент устной речи или документа, в пределах которого можно уяснить значение отдельного слова или объекта. Только в контексте слово или объект получают конкретное значение.

Контекстная справка (contextual help)

Текстовая строка около курсора мыши. Предоставляет пользователю информацию об объекте, с которым он взаимодействует в настоящий момент. Она принимает во внимание контекст текущих действий пользователя и пытается ответить на вопросы типа: «Что это такое?» и «Зачем мне это?». Такую справку также называют контекстно-зависимой справкой.

Контекстное меню

Меню, открываемое операционной системой или приложением в результате щелчка правой кнопкой мыши по некоторому объекту. Такие меню, в зависимости от контекста операционной обстановки, содержат разные наборы команд, которые могут быть применены в данный момент работы с данным объектом в текущей точке положения курсора.

Контекстно-чувствительный (context-sensitive)

В компьютерных технологиях, интерфейс, разработанный для обеспечения помощи пользователю именно в той точке, где это необходимо, в противоположность программам, где существует общий экран помощи, который предварительно должен быть пользователем открыт, а затем осуществлять перемещение по его содержимому в поисках ответа на специфический вопрос.

Контент (content)

❶ Основное содержание или суть литературной работы или устного изложения (discourse), в противоположность их форме или стилю. В более общем смысле, все идеи, темы, факты или утверждения, содержащиеся в книгах или других печатных изданиях. Синоним в этом случае – предмет изучения (subject matter). Понятие контент, также относится к элементам, содержащимся в курсах обучения по разным специальностям (course of study).

❷ Знания и интеллектуальная собственность заключенные в учебных курсах и распространяемые с помощью е-Образовательных технологий. е-Образовательный контент включает широкий спектр понятий от простых Web-страниц и документов до полностью интерактивных курсов, систем оценки получаемых с их помощью знаний и программных средств обеспечения их функционирования.

❸ Любое информационно значимое наполнение сервера - тексты, графика, мультимедиа. Контент организуется в виде страниц средствами гипертекстовой разметки. Существенными параметрами контента являются его объем, актуальность и релевантность.

Контента анализ (content analysis)

Строгий анализ явных (explicit) и неявных (implicit) передаваемых блоков информации (message), содержащихся в печатных работах или во внутренней части (теле) информационных сообщений, посредством классификации, дешифрирования или оценки главнейших концептов, обозначений, знаков и тем в них, с точки зрения оценки их значения и эффекта влияния на аудиторию.

Контингент

❶ Состав какого–либо коллектива.

❷ Норма, предельное количество (например, при приёме абитуриентов в ВУЗ).

Контроллер

Устройство согласования (по скорости передачи и уровням сигналов) системного интерфейса и некоторого стандартного интерфейса периферийного устройства с компьютером. Различают три группы интерфейсов периферийных устройств: параллельные, последовательные и интерфейсы внешних запоминающих устройств.

Контроллер памяти

Промежуточное устройство между системной шиной и модулями памяти. Контроллер определяет возможные тип и рабочий режим используемой памяти (в стандартных решениях зачастую и форм-фактор), организует interleave, контроль чётности или ECC и т.п. Иногда в контроллере имеется возможность настройки ряда параметров из BIOS Setup, в других случаях определение типа памяти и режима работы происходит автоматически. В настоящее время, как правило, контроллер памяти является частью чипсета.

Контроллер периферийного устройства

Устройство сопряжения компьютера с внешним устройством и управления обменом. Между понятиями «адаптер» и «контроллер» отсутствует чёткая грань. «Адаптер» подразумевает, в первую очередь, преобразование представления и скорость передачи информации, а «контроллер» обычно выполняет более сложные функции управления устройством.

Конфигурация (configuration)

❶ Совокупность функциональных частей компьютерной системы и связей между ними, обусловленная основными техническими характеристиками этих функциональных частей, а также характеристиками решаемых задач обработки данных. (ГОСТ).

❷ Конфигурация является структурой архитектурных взаимосвязей между компонентами, коннекторами и данными в период протекания процесса работы системы (system run-time). (W3C).

Конфигурирование

Поведение работ с операционной системой компьютера, связанной с подключением к ней драйверов используемых в данной системе внешних устройств (блоков памяти, принтеров, мониторов и др.), также (возможно) выполнение требуемых установок в BIOSe.

Концентратор

Устройство в сети, предназначенное для передачи данных, поступающих по нескольким входным каналам, в меньшее количество выходных каналов. Помимо этого, многие концентраторы могут хранить полученные данные до тех пор, пока выходной канал не освободится и не будет готов их принять.

Концепция

❶ Ведущий замысел в научной, технической, политической и других видах деятельности.

❷ Определённый способ понимания, трактовка какого-либо предмета, явления, процесса. Руководящая идея для их систематического освещения.

Концептуальное проектирование (conceptual design)

Процесс сбора, документирования и проверки информации, описывающей точку зрения пользователя в е-Бизнесе на проблему и её решение. Цель концептуального

проектирования заключается в понимании действий пользователя и выяснения потребностей бизнеса. Продуктом проведенного концептуального проектирования являются сценарии.

Корпоративная сеть

Сеть, обеспечивающая работу и взаимодействие сотрудников корпорации, независимо от размера компании, количества и удаленности филиалов, а, следовательно, функционирование компании в целом, путем использования современного оборудования и программного обеспечения, а также различных средств связи. Сеть, как правило, включает в себя компьютеры разных типов, начиная с настольных и кончая мейнфреймами, системное и прикладное программное обеспечение, сетевые адаптеры, концентраторы, коммутаторы и маршрутизаторы, кабельную систему и др.

Кроссплатформенный (кросс платформенный)

Термин, относящийся к проектированию программных систем и компонентов, способных выполняться на любой из платформ – либо Windows, либо Unix, либо Sun либо любой другой.

Кэш (сверхоперативная память) (Cache)

❶ ЗУ с очень малым временем доступа (в несколько раз меньше, чем время доступа к основной оперативной памяти). Используется для временного хранения промежуточных результатов и содержимого часто используемых ячеек более медленной оперативной памяти.

❷ Область памяти для хранения последних полученных данных. Используется для восстановления страницы непосредственно из локальной памяти, вместо того, чтобы перекачивать данные вновь из сети. Netscape Navigator запоминает в кэш последние посещенные сайты.

❸ Интеллектуальный буфер. Служит посредником в передаче данных между быстрым процессором и медленным внешним устройством.

- Л -

Латентность (latency)

Время, необходимое для получения данных по сети.

Лексема

❶ Минимальные единицы значений текста в программе.

❷ Минимальная единица языка, имеющая значение: идентификатор, буквенная константа, знак операции, разделители.

❸ Элементарное значение.

Лексика (от греческого lexikos - относящийся к слову)

❶ Вся совокупность слов, словарный состав языка.

❷ Совокупность слов, характерных для данного варианта речи (лексика бытовая, компьютерная, военная, детская и пр.), того или иного стилистического пласта (лексика нейтральная, просторечная и другая), для данного писателя (лексика А.С. Пушкина) или одного литературного произведения (лексика "Слова о полку Игореве").

Линия (от латин. linea, буквально – нитка)

❶ (в математике) Граница поверхности, имеющая только одно измерение (длину) и определяемая, как след движущейся точки или место пересечения двух поверхностей. К примеру, линия прямая, кривая, ломаная, перпендикулярная, наклонная, параллельная.

❷ Воображаемая черта, соединяющая две точки или являющаяся границей пересечения двух поверхностей. К примеру, линия горизонта, линия экватора.

❸ Черта, проведенная на какой-либо поверхности, узкая полоса.

Линия связи

Совокупность технических устройств и физической среды, обеспечивающая распространение сигналов от передатчика к приёмнику. Линия может быть проводная (воздушная и кабельная), радио, радиорелейная и др.

Литера

❶ Элемент алфавита.

❷ Рельефное трёхмерное изображение буквы или знака, применяемое для его печатного воспроизведения в типографическом наборе. Литеры изготовляются в виде металлических, деревянных или пластмассовых брусочков с рельефным изображением (очком) буквы или знака на одном из его торцов. При печати очко покрывается краской и даёт оттиск на бумаге. Размеры литеры определяются: кеглем (высотой самой буквы или знака), толщиной (шириной) и ростом (высотой ножки) (постоянным для всех литер).

Логическая модель данных (Logical Data Model)

Фактическая реализация концептуальной модели в базе данных. Для реализации одной концептуальной модели данных может потребоваться множество логических моделей данных.

Логическая структура

Совокупность определений, которая устанавливает порядок и принципы взаимодействия отдельных частей системы. (ГОСТ)

Логическая структура компьютера

Абстрактная модель, устанавливающая состав, порядок и принципы взаимодействия основных функциональных частей компьютера, без учёта их реализации. (ГОСТ)

Логическая схема

Блок-схема, представляющая в графическом виде логическую структуру процессов, программ, систем обработки данных. (ГОСТ)

Логический

❶ Рассматриваемый с точки зрения возможных операций, а не с точки зрения реальной организации. Понятие «виртуальный» обычно подразумевает большую степень абстракции. «Концептуальный» и «абстрактный» относятся больше к рассуждениям и проектированию, чем к функционированию программ.

❷ Предусматривающий использование логики.

❸ Концептуальный или виртуальный, т.е. включающий в себя концептуальные, а не реальные физические объекты. (Иллингуорт)

Логический адрес

Символьный или условный адрес ячейки или области памяти, устройства или узла сети, который переводится в физический адрес соответствующим программным или аппаратным обеспечением.

Логический диск

Часть физического жёсткого диска, рассматриваемая как отдельный жёсткий диск со своим именем накопителя.

Логический номер устройства

Число, используемое в качестве имени логического устройства в ряде систем программирования.

Логический проект (logical design)

Вид на решение с точки зрения проектной группы, определяющий решение (программный продукт) как набор взаимодействующих объектов и образуемых ими служб. Обычно службы делятся на три группы: службы представления, службы бизнеса и службы данных. Цель логического проекта заключается в описании структуры решения и взаимодействия её элементов.

Логическое имя

Абстрактное обозначение устройства компьютера в виде дополнительного текстового и/или графического имени/обозначения, приписываемых операционной системой для удобства их использования. Логическое имя допускает использование данного объекта таким образом, что не возникает необходимость вдаваться в особенности его физической реализации. (ГОСТ)

Логическое устройство

Абстрактное обозначение устройства ввода-вывода в виде дополнительного текстового имени, приписанного устройству в программе или в операционной системе. В ОС MS-DOS независимо от вида устройства все принтеры имеют логическое имя “PRN”. В ОС Windows логическое имя устройства, использующего гибкие диски, к примеру, - “3 ½ Floppy” и т.д.

Логограмма

Так называется определенное написание имени компании или ее продукта. Логограмма должна быть легкой для восприятия, а также в изображении и употреблении, т.е. должна легко сочетаться с текстом и другими элементами оформления.

Логотип

Сочетание Знака (графического изображения) и Логограммы (шрифтовой надписи).

Локализация

Набор информации, соответствующей данным языку и стране. Локализация влияет на язык компьютерных терминов и на различные параметры, зависящие от страны.

Локализация программного обеспечения (ПО)

Комплекс работ по доработке ПО, с целью сделать программный продукт удобным для пользователей того или иного культурно-географического пространства (т.е. перевод на соответствующий национальный язык всех меню и команд пользовательского интерфейса, элементов хелпа к системе, других текстов, а также доведение ПО до соответствия требований законодательства в регионе, перевод расчетов из одной валюты в другую и многие другие мероприятия).

Локальная вычислительная сеть (ЛВС)

Русскоязычный синоним региональных (WAN) и локальных (LAN) вычислительных сетей. LAN (Local Area Network) является сетью передачи данных, охватывающей небольшую территорию (например, предприятие, здание, этаж здания и т.п.). Данная сеть соединяет опеределенное количество компьютеров и других устройств между собой или с центральным сервером и обеспечивает между ними высокоскоростную передачу данных.

- M -

Макрогенератор

Программа или техническое средство, выполняющее преобразование макрокоманд в их макрорасширения. (ГОСТ).

Макрокоманда

Предложение языка программирования, вместо которого макрогенератор подставляет его макрорасширение. (ГОСТ)

Макрорасширение

Последовательность предложений (инструкций), порождаемая макрогенератором при обработке макрокоманды под управлением макроопределения и вставляемая в программу вместо макрокоманды. (ГОСТ).

Мантисса

Составная часть числа, содержащая его значащие цифры в представлении с плавающей запятой.

Маркетинг (Marketing)

❶ (в широком смысле) Философия управления, согласно которой разрешение проблем потребителей путем эффективного удовлетворения их запросов, ведет к успеху организации и приносит пользу обществу.

❷ (на уровне отдельных субъектов хозяйствования) Система, ориентированная на производство разнообразных благ и удовлетворение интересов производителей и потребителей посредством:

- планирования ассортимента и объема выпускаемых продуктов;

- определения цен;

- распределения продуктов между выбранными рынками и стимулирования их сбыта.

❸ (в предпринимательском смысле) Система управления производственно-сбытовой деятельностью организации, направленная на получение приемлемой величины прибыли посредством учета и активного влияния на рыночные условия.

Маршрутизатор (Router)

Специализированный компьютер или пакет программного обеспечения, отвечающий за соединение между двумя или несколькими сетями. Маршрутизаторы отыскивают в пакетах данных, передаваемых через них, адреса получателей и определяют, по какому маршруту следует передавать эти пакеты.

Маршрутизация

Выбор последовательности узлов сети передачи данных, по которой данные передаются от источника к приёмнику.

Масс медиа (Mass media) (Средства массовой информации – синоним)

Пресса (газеты, журналы, книги), радио, телевидение, кинематограф, звукозаписи и видеозаписи, видеотекст, телетекст, рекламные щиты и панели, домашние видеоцентры, сочетающие телевизионные, телефонные, компьютерные и другие линии связи. Всем этим средствам присущи объединяющие их качества – обращенность к массовой аудитории, доступность множеству людей, корпоративный характер производства и распространения информации.

Массив (данных) (array)

Конструкция данных, компоненты которой идентичны по своим характеристикам и перечисляются как значения функции от фиксированного количества целочисленных аргументов. Количество аргументов определяет размерность массива. (ГОСТ 20886-85).

Массовая коммуникация

Процесс передачи информации группе людей одновременно с помощью специальных средств – масс медиа.

Масштабируемость (Scalability)

❶ Характеристика, указывающая, насколько хорошо система будет работать при решении некоторой проблемы, когда размеры проблемы будут увеличиваться.

❷ Масштабируемость показывает возможность системы не терять производительность в выполнении пользовательских сервисов в условиях возрастания количества пользователей.

❸ Характеристика компьютерного приложения или компонента, подтверждающая его возможности для увеличения размера, производительности или числа обслуживаемых им пользователей, при неизменных функциональных возможностях.

Математика

Наука о количественных отношениях и пространственных формах действительного мира. (Математическая энциклопедия).

Математическая модель

❶ Приближённое описание какого-либо класса явлений внешнего мира, выраженное с помощью математической символики. (Математическая энциклопедия)

❷ Согласно стандарту ISO 10303 STEP математическая модель состоит из объектов. Поскольку в реальном мире объекты связаны между собой, объекты математической модели тоже должны быть между собой связаны. В реальном мире не существует двух одинаковых объектов, т.е. явлений или предметов (“В одну реку нельзя войти дважды”), но для отображения объектов в сознании человека и в памяти компьютера объекты систематизируются и классифицируются. Любой объект воспринимается через его свойства, поэтому объекты, имеющие одинаковый набор свойств можно считать объектами одного типа. Отличаются такие объекты не набором свойств, а *значениями свойств*. В реальном мире может существовать множество отличающихся объектов, имеющих один и тот же

набор свойств (т.е. много экземпляров объектов одного типа). Следовательно, и в математической модели тоже может существовать множество экземпляров объекта одного типа.

Матрица

Система элементов (чисел, функций или других величин, над которыми производятся алгебраические операции), расположенных в виде прямоугольной системы.

Междисциплинарный (interdisciplinary)

Подход в исследованиях или направление деятельности, требующие использования более чем одной академической дисциплины и потому часто называемые интегрированными. К примеру, такими сферами знаний и исследований являются геоинформатика, геоматика и др.

Межсетевое взаимодействие (См. *internetworking)*

Меню

❶ Совокупность команд, обеспечивающих управление диалоговыми программными продуктами, операционными системами и приложениями. В ОС Windows пользователю доступны меню четырёх разновидностей:

–главное меню системы;

–контекстные меню всевозможных объектов;

–меню приложений;

–меню управления окнами приложений, окнами документов и диалоговыми окнами.

❷ Список свойств объектов и применяемых к ним команд (т.е. способов преобразования) в операционных системах и работающих под их управлением приложениях, а также в средах быстрой разработки приложений (RAD-средствах).

Метаданные

Графическая или текстовая информация о содержании, качестве, условиях, источниках, происхождении, свойствах и характеристиках данных. Другими словами – это данные о данных.

Метамодель

❶ Модель, описывающая способ (язык) выражения модели.

❷ Модель модели (к примеру – языковая).

Метаобъекты

❶ *(информационные)* Типы объектов данных информационной системы (ИС): элемент данных, группа, запись, файл, база данных и т. д.;

❷ *(системные)* Элементы самой ИС: подпрограммы, модули, подсистемы и т. д.;

❸ *(среды)* Объекты среды функционирования ИС: предприятия, подразделения предприятия, линии связи, терминалы, пользователи и т. д.

Одни и те же объекты могут быть описаны в различных аспектах. Например, человека можно описать с информационной, биологической, социальной, медицинской, юридической и других точек зрения.

Метафора–(от греч. metaphora – перенос)

Троп или механизм речи, состоящий в употреблении слова, обозначающего некоторый класс предметов, явлений и т.п., для создания характеристики или наименования объекта, входящего в другой класс, либо наименования другого класса объектов, аналогичного данному в каком-либо смысле. В расширительном смысле термин "метафора" применяется к любым видам употребления слов в непрямом значении.

Метаязык

Язык, используемый для описания языка программирования (ЯП).

Метод (method)

❶ Способ достижения какой-либо цели, решения определённой задачи.

❷ Приём или совокупность приёмов труда в какой-либо области.

❸ (в науке) Способ познания действительности.

❹ Приёмы исследования.

❺ Последовательный процесс создания моделей, которые описывают вполне определёнными средствами различные стороны разрабатываемой системы. (UML).

❻ Реализация операции (в объекте). Описывает алгоритм или процедуру, которая формирует результат операции. (UML).

❼ Функция или процедура, которая выполняет некоторое действие и может быть вызвана программным обеспечением, использующим объект.

❽ Операция над объектом, определенная как часть описания класса. Не любая операция является методом, но все методы - операции. Термины "метод", "сообщение" и "операция" обычно взаимозаменяемы. В некоторых языках методы существуют сами по себе и могут переопределяться подклассами; в других языках метод не может быть переопределен. Он служит как часть реализации обобщенных или виртуальных функций, которые можно переопределять в подклассах.

Методика

Совокупность способов и приёмов, применяемых при выполнении какой-либо работы (исследовательской, учебной, воспитательной и т.д.). Расширением понятия *методика* является термин ***технология***.

Методология–(от *метод* и *…логия*)

❶ Учение о методе.

❷ Принципы приёмов исследования, применяемых в той или иной отрасли науки.

❸ Совокупность методов, применяемых в жизненном цикле разработки программного обеспечения (ПО) и объединённых одним общим философским подходом. (Гради Буч).

❹ Система принципов, действий и процедур, применяемых в конкретной области знаний.

Метрика (metric)

Метрика является атрибутом архитектурного компонента и может быть определена в процессе конфигурации архитектурного компонента, может быть измерена в процессе использования этого архитектурного компонента, либо её значение может быть оценено. (W3C).

Микроволновая сеть

Один из типов беспроводных сетей. Для передачи информации в микроволновых сетях используются сигналы гигагерцевого диапазона электромагнитного спектра. В микроволновых сетях антенны широковещательно передают лучи сигналов на остальные узлы. В качестве ретрансляторов могут использоваться как наземные антенны, так и геостационарные спутники Земли. При использовании спутниковых ретрансляторов расстояние передачи сигнала может составлять тысячи километров; при использовании антенн оно, обычно, ограничено единицами километров. Микроволновые сети обеспечивают передачу данных на скоростях до нескольких гигабит в секунду.

Многомерная база данных, СУМБД (Multi-dimensional Database, MDBS and MDBMS)

Мощная база данных, позволяющая пользователям анализировать большие объемы данных. База данных со специальной организацией хранения - кубами, обеспечивающая высокую скорость работы с данными, хранящимися как совокупность фактов, измерений и заранее вычисленных агрегатов.

Многоярусная архитектура (multitier architecture, N-tier architecture)

Архитектура, основанная на разделении приложения на несколько различных функциональных частей. Обычно приложение проектируется вокруг трёх ярусов: представления (пользователя), бизнеса и данных. В среде Windows многоярусные приложения, называемые также трехъярусными приложениями, реализуются обычно на базе компонентной объектной модели (COM). (См. *N-tier application*, (*n*-ярусные приложения)).

Мобильность (программ)

Возможность переноса прикладного ПО с минимальными изменениями в широком диапазоне компонентов, платформ, информационных систем и компьютерных систем, приобретаемых у одного или нескольких поставщиков.

Модальное окно (modal window)

Тип вторичного окна. Модальное вторичное окно не позволяет пользователю переключаться на другие окна, пока он не закончит работать с этим окном и не закроет его. Вторичное окно может быть модальным по отношению к приложению или к системе (операционной). (см. *немодальное окно*).

Моделирование

❶ Исследование физических процессов и явлений на моделях.

❷ Класс методов, использующих модель реальной ситуации, в условиях которой проводится эксперимент.

❸ (математическое) Абстрагированное и упрощённое отображение действительности логико-математическими формулами, передающими в концентрированном виде сведения о структуре, взаимосвязях и динамике исследуемых явлений.

❹ Представление некоторых характеристик поведения физической или абстрактной системы поведением другой системы, например, представления физического явления с помощью операций, выполняемых компьютером, или представление работы одного компьютера работой другого компьютера. Возможности моделирования, то есть перенос результатов, полученных в ходе построения и исследования модели, на оригинал основаны на том, что модель в определенном смысле отображает (воспроизводит, моделирует, описывает, имитирует) некоторые интересующие исследователя черты объекта. Моделирование как форма отражения действительности широко распространено, и достаточно полная классификация возможных видов моделирования крайне затруднительна, хотя бы в силу многозначности понятия "модель", широко используемого не только в науке и технике, но и в искусстве, и в повседневной жизни.

❺ Языковое или графическое описание (к примеру, средствами языка UML), сложной, многомерной, в том числе и информационной системы с целью её исследования и программной реализации на компьютере.

Моделирование данных (data modeling)

❶ Анализ объектов данных и их связей с другими объектами данных. Моделирование данных, как правило, является первым шагом в разработке баз данных и объектно-ориентированных программ, когда разработчик в начале создаёт концептуальную модель того, как элементы данных соотносятся и взаимодействуют друг с другом. Моделирование данных включает продвижение от концептуальной модели к логической модели, а затем к физической модели (схеме). (Webopedia)

❷ Метод, используемый для определения и анализа требований к данным, необходимым для поддержки бизнес-функций предприятия. Эти требования записываются как концептуальная модель данных с конкретными определениями. Моделирование данных определяет отношения между элементами и структурами данных.

Модель

❶ Образец чего-либо.

❷ Материальное подобие какого-либо предмета в уменьшенном или увеличенном виде (модель самолёта или молекулы).

❸ Упрощённое преставление реальности.

❹ Интерпретация формального языка. (Математическая энциклопедия).

❺ Представление чего-либо в некоторой среде (на бумаге, из папье-маше, в виде математических выражений или спецификаций). (UML).

❻ Формализованная абстракция. При этом модели представляют два важных аспекта: *смысловую информацию* (семантику) и *визуальное представление* или *нотацию* (нотация–представление условными знаками). (UML).

❼ Модель есть семантически (по смыслу) полная (замкнутая) абстракция системы. Модель UML представляет пакет иерархий, обосновывающих единственный и единый взгляд на систему. (UML).

❽ Представление реальности, используемое для имитации (воспроизведения) процесса, понимания ситуации, прогнозирования последствий или анализа проблемы. Модель структурируется на наборы правил и процедур, включающих средства пространственного моделирования, доступные в географических информационных системах. (ГИС).

❾ Набор суждений или уравнений, описывающих в упрощённой форме некоторые аспекты нашего опыта. Каждая модель основывается на теории, но теория может не быть сформулированной в лаконичной и чёткой форме. При этом модели подразделяются на следующие категории:

–физические;
–предметно–математические;
–знаковые;
–абстрактные (представляемые). (Umpleby).

❿ Объект или процесс, который комплексно использует важнейшие свойства оригинала, моделируемого объекта или процесса, но при этом значительно проще в манипулировании или понимании.(Arbib).

❶❶ Система, которая символизирует или представляет другую, моделируемую и обычно более всеобъемлющую. Модель состоит из набора объектов, описываемых в терминах переменных и отношений, определяемых:

-выбранной теорией, описывающей выбранный фрагмент реальности, предназначенный для представления;

-соответствующими данному фрагменту реальности выбранными свойствами гомоморфизма и изоморфизма между параметрами модели и заданными данными. Известны четыре типа моделей:

1) дискретные модели, состоящие из простого подмножества взаимно не пересекающихся объектов из большего множества объектов;

2) модели, точно повторяющие исходный объект, являющиеся линейной трансформацией каждого объекта из множества подобных, имеющих общие свойства, которые включаются в образец;

3) поведенческие модели, в которых взаимосвязи являются преобразованиями, уравнениями или правилами управления и представления основываются на убеждениях, что поведение модели соответствует поведению моделируемой системы.

4) символические модели, в которых наборы объектов являются символами и связи выражаются в форме алгебраических, вычислительных или алгоритмических утверждений, не представляющих их собственного поведения. (Krippendorff).

Модель архитектуры производственных приложений (Enterprise Application Model)

Модель проектирования и разработки приложений масштаба предприятия. Состоит из шести моделей: бизнес, пользователь, логика, технология, физическая модель и модель разработки.

Модель данных (Data Model)

❶ Обобщённый, определяемый пользователем взгляд на данные, соотнесённые с программным приложением.

❷ Формальный метод организации данных, описывающих поведение сущностей реального мира. Полностью разработанные модели данных описывают типы данных, правила целостности для типов данных и операции над типами данных.

❸ Представление объектов реального мира в виде сущностей в базе данных.

❹ Логическая карта, представляющая наследуемые свойства данных независимо от программного и технического обеспечения, а также от особенностей работы приборов. Модель демонстрирует элементы данных, сгруппированные в записи, а также связи, окружающие эти записи.

Модель жизненного цикла программы (см. *жизненный цикл программы*)

Модель производственной архитектуры (Enterprise Architecture Model)

Одна из шести базовых моделей методологии проектирования и разработки программных продуктов Microsoft (Microsoft Solution Framework, MSF). Модель производственной архитектуры включает совокупность рекомендаций, предназначенных для быстрой разработки производственной архитектуры на основе выпуска версий. Позволяет найти компромисс между бизнес-требованиями и технологическими возможностями, анализируя проблему с четырех точек зрения: бизнес, приложение, информация и технология.

Модем

❶ Компьютерное устройство, которое позволяет компьютерам взаимодействовать друг с другом через телефонные линии. При этом осуществляются преобразования цифровых сигналов в аналоговые для передачи и обратно в цифровые для получения и последующей обработки в компьютере.

❷ Модемами (сокращенно от «модулятор/демодулятор»), называются коммуникационные устройства, предназначенные для преобразования цифровых сигналов в акустические с последующей передачей их по обычным телефонным линиям, а также для восстановления исходного цифрового сигнала на приемном узле. Процесс преобразования данных в акустические сигналы называется «модуляцией», а обратный процесс их восстановления – «демодуляцией». Различные типы модемов отличаются друг от друга реализуемыми методами модуляции, а также коммуникационными и прочими стандартами, которым они соответствуют. Модемы принято разделять по следующим признакам:

–по классу: узкополосные, речевые, широкополосные и для физических линий;

–по используемому методу модуляции: с частотной, амплитудной, фазовой, квадратурно-амплитудной;

–по методике передачи сигнала: несколько типов методик, описанных в стандартах Bell и ITU-T;

–по методам коррекции ошибок: без коррекции, MNP;

–по конструктивному исполнению: внешние и внутренние.

Модуляция

Модуляцией называется процесс преобразования информационного (модулирующего) сигнала в форму, пригодную для передачи с использованием другого сигнала (несущего). Для этого, информация исходного сигнала накладывается на неизмененный несущий сигнал. Результирующий сигнал используется для передачи данных по той или иной среде передачи. В модуляции могут принимать участие только цифровые или только аналоговые сигналы, или же те и другие вместе. Существует несколько видов модуляции:

- Аналоговая модуляция – используется для преобразования одного аналогового сигнала (информационного) в другой (несущий);
- RF-модуляция – используется для преобразования цифровых сигналов в аналоговую форму;
- Цифровая модуляция – используется для преобразования аналоговых сигналов в цифровую форму, пригодную для передачи по цифровым линиям связи и для записи на цифровые носители.

Модуль

❶ Программа, которая рассматривается как отдельное целое в процессах сохранения, трансляции и объединения с другими программными модулями при их загрузке в оперативную память компьютера для выполнения.

❷ В языке Java, программный модуль, который состоит из одного или более компонентов J2EE, имеющих одинаковый тип контейнера и дескриптора (признака) развёртывания. Имеются три типа модулей: EJB (Enterprise Java Beans), Web и приложение клиент.

❸ Функционально законченный узел, являющийся частью определённой системы, оформленной как самостоятельное изделие и обладающий свойством заменяемости.

Моникер (moniker)

Имя, однозначно определяющее COM-объект, подобно полному (с путём) имени файла. Моникеры поддерживают такую операцию, как привязка (binding), под которой понимается процесс нахождения объекта, на который указывает моникер, активизации этого объекта или загрузки его в память (если он ещё не загружен) и возвращения указателя на его интерфейс.

Мониторинг

❶ Система непрерывного наблюдения, измерения и оценки состояния окружающей среды. Т.е. мониторингом окружающей среды называют регулярные, выполняемые по единообразной программе наблюдения природных сред, природных ресурсов, растительного и животного мира, позволяющие выделить изменения их состояния и происходящие в них процессы под влиянием антропогенной деятельности.

❷ Постоянное наблюдение за каким-либо процессом с целью выявления его соответствия желаемому результату или первоначальным начальным условиям, а с другой стороны, как наблюдение, оценку и прогноз состояния окружающей среды в связи с деятельностью человека

Мост

Устройство, предназначенное для передачи пакетов данных из одной сети в другую. С функциональной течки зрения, мосты относятся к канальному уровню эталонной модели OSI. Мосты позволяют программам и протоколам, работающим на более высоких уровнях, рассматривать объединение нескольких сетей, как одно целое. Наряду с передачей данных, мосты могут, также, выполнять их фильтрацию. Это означает, что в сеть N2 будут попадать только те пакеты, которые предназначены для узлов этой сети. А пакеты, предназначенные для узлов сети N1, из которой они поступают, будут возвращаться обратно. Значения терминов «мост» и «маршрутизатор» во многом сходно.

Мультимедиа (Multimedia)

Комбинация графических, звуковых и анимационных файлов, представленная компьютерными средствами. Также, интерактивное взаимодействие с компьютером в комплексном представлении текста, изображений, звука и цвета. Мультимедиа взаимодействие может быть представлено простым слайд-шоу, созданным в PowerPoint или сложным интерактивным моделированием.

Мультиплексор

Устройство уплотнения, позволяющее передавать по одной линии несколько сигналов (потоков данных) одновременно.

Мэйнфрейм (Mainframe)

❶ Большая ЭВМ.

❷ Центральный процессор. Часть вычислительной системы, в которую входят оперативная память и собственно процессор.

❸ Базовое централизованное вычислительное устройство, объединяющее все данные, программное обеспечение и оборудование, находящиеся в одном месте.

❹ Универсальная многопользовательская вычислительная машина.

❺ Большая, пришедшая из прошлого компьютерная система. Мэйнфреймы до сих пор используются во многих видах деятельности. В основном мэйнфреймы занимаются пакетной обработкой, но есть и такие, на которых выполняются критические диалоговые приложения обработки транзакций бизнес-процессов.

Мышление

Внутреннее активное стремление овладеть своими собственными представлениями, понятиями, побуждениями чувств и воли, воспоминаниями, ожиданиями и т. д. Мышление, которое по своей структуре может быть познающим или эмоциональным, состоит в постоянной перегруппировке всех возможных составляющих сознания и образовании или разрушении существующих между ними связей.

Мышления формы (Человеческий интеллект)

Способы и виды формальной организации мыслительного процесса, абстрагированные от его содержательного компонента.

- H -

Наблюдатель

❶ Лицо, которое следит за действием, не участвуя в нём.

❷ Источник очевидных фактов. Лицо, которое сообщает о своих чувственных ощущениях о внешней окружающей среде.

Набор данных (data sets)

Совокупность значений, принадлежащих одиночному объекту.

"На лету" (см. *on the fly*)

Наследуемая система (унаследованная система) (legacy system)

❶ Приложение или решение, выполняющие функции, критичные для функционирования бизнеса. Как правило, наследуемые системы выполняются на мэйнфреймах, однако они могут выполняться также на мини-компьютерах или настольных системах. Сюда же может относиться большая программная система, прослужившая длительный срок (10-20 и более лет) и всё ещё эксплуатируемая, ввиду своей большой стоимости. Как правило, такая система обычно нуждается в дальнейшем реинжиниринге, что редко бывает экономически оправданным.

❷ Программно-аппаратная среда (как правило мощный мэйнфрейм со своей инфраструктурой), продолжающая использоваться для решения критических для бизнеса задач.

Настольные приложения (Desktop Applications)

❶ Инструменты формирования запросов и анализа, которые получают доступ к исходной базе данных или Хранилищу данных по сети с использованием соответствующего интерфейса базы данных.

❷ Приложение, управляющее пользовательским интерфейсом для поставщиков данных и потребителей информации.

Немодальное окно (modeless)

Тип вторичного окна. Вторичное немодальное окно позволяет пользователю переключаться на другие окна, включая другие вторичные или первичные окна.

Нечеткая математика (Нечеткая логика — Недоопределенные данные)

Раздел математики, связанный с нечеткими объектами, данными и алгоритмами.

Непрерывное обучение (Lifelong learning)

Комплекс государственных, частных и общественных образовательных учреждений, обеспечивающих организационное и содержательное единство и преемственную взаимосвязь всех звеньев образования, удовлетворяющий стремление человека к самообразованию и развитию на протяжении всей жизни.

Нормализация данных (data normalization)

❶ Процесс реструктуризации базы данных в процессе разработки, направленный на устранение избыточности данных, посредством изменения количества и структуры её таблиц. Всего определено пять уровней нормализации, иначе говоря, *нормальных форм. С* каждым из уровней связан свой набор правил и ограничений на организацию данных в таблицах базы, строгость которых увеличивается от уровня к уровню.

❷ Процесс уменьшения комплексной структуры данных до простейшей, наиболее стабильной структуры. В целом, процесс вызывает удаление излишних атрибутов, ключей и отношений из концептуальной модели данных.

Нотация

Графический язык для описания моделей при разработке программного обеспечения.

- O -

Область

❶ Часть страны, территория, пространство. К примеру, область за областью покрываются сетью дорог. На севере целые области покрыты лесами.

❷ Крупная административно-территориальная хозяйственно-политическая единица. К примеру, Ленинградская область. Московская область и т.д.

❸ Район, пределы, в которых распространено какое-нибудь явление; зона, пояс. К примеру, область вечных снегов, область распространения пшеницы.

❹ (медицинское) Место, занимаемое каким-либо органом тела с прилегающими к нему частями, или ограниченный по каким-нибудь признакам участок тела, какого-нибудь предмета. К примеру, ранение в области сердца, боль в области ранения.

❺ Определенная сфера знаний, деятельности или представлений; какая-нибудь отрасль наук, искусств, техники. К примеру, "быстрое развитие области компьютерных технологий".

❻ Непустое, связное, открытое множество точек топологического пространства *X*. (Математическая энциклопедия)

❼ Математическое представление множеств объектов: область значений, область определения, область целостности. (Математическая энциклопедия)

Область видимости имён

Принцип использования объектов программ (переменных, констант и др.) при котором одни объекты доступны всем частям программы (глобальные объекты), а другие используются локально, т.е. внутри подпрограмм и объектов.

Образ

❶ (в психологии) Субъективная картина мира, включающая самого субъекта, других людей, пространственное окружение и временнýю последовательность событий.

❷ Образ художественный – категория эстетики, средство и форма освоения жизни искусством; способ бытия художественного произведения.

Обратное проектирование (reverse engineering)

❶ Предполагает процесс преобразования кода, написанного на каком либо языке программирования в модель (в том числе языковую). (UML).

❷ Получение с помощью различных технологий исходного текста программы на языке высокого уровня или ассемблера из имеющихся машинных кодов.

Объект (Object)

❶ То, на что направляется творческий, созидательный труд человека (объект исследования, строительный объект, промышленный объект).

❷ Предмет или явление, существующие в реальной действительности.

❸ Нечто, имеющее чётко очерченные границы. (Буч).

❹ Осязаемая сущность, имеющая чётко определяемое поведение. (Буч).

❺ Инкапсулированная абстракция, которая включает информацию о состоянии и чётко определённое множество элементов протокола доступа (т.е. сообщения, которые обрабатывает объект). (Гради Буч).

❻ Форма представления данных.

❼ Некоторая структура данных либо понятие, устройство или процесс, который выражен в программе в виде совокупности характеризующих его данных. В любой программе можно выделить следующие основные классы действий над объектами:

–конкретное представление (описание);

–генерация (создание объекта или его компонентов);

–модификация (изменение состояния объекта);

–доступ (обращение к атрибутам объекта и их анализ);

–преставление, вывод (преобразование информации об объекте и его состоянии в форму, наиболее удобную для восприятия человеком или программой для дальнейшей обработки объекта).

❽ Конкретный представитель определённого класса, называемый экземпляром класса.

❾ Самодостаточный программный модуль, который абстрактно описывает физическую или логическую сущность реального мира. Он скрывает (инкапсулирует) детали своей реализации и имеет общедоступный интерфейс. (COM объект).

❿ (объект данных) Любой элемент данных, хранимый в базе данных и дающий информацию о конкретных объектах или явлениях реального мира. Связи между объектами данных в БД определены отношениями. (ГОСТ).

❶❶ Дискретная сущность с чётко определёнными границами и индивидуальностью, инкапсулирующая состояние и поведение. Экземпляр класса. (UML).

❶❷ (в объектно-ориентированном программировании) Конкретный представитель определённого класса, методами, свойствами и событиями которого можно управлять. В данном контексте его характеризуют:

Метод - это процедура или просто набор команд, сообщающих объекту, что нужно выполнить некоторую задачу и реализующих алгоритм её выполнения.

Свойство - это некоторый вид параметра объекта.

Событие - сигнал, подаваемый, если с объектом что-то происходит или необходимо сделать.

❶❸ Лицо, место, вещь или понятие, имеющее характеристики, значимые для среды. В терминах объектно-ориентированных систем объект - это сущность, объединяющая описания данных и описание их поведения.

Объектно-ориентированная технология

Комплекс методик создания программных систем, основывающихся на так называемой объектной модели. Основными ее принципами являются: абстрагирование, инкапсуляция, модульность, иерархичность, типизация, параллелизм и сохраняемость. Каждый из этих принципов сам по себе не нов, но в объектной модели они впервые применены в совокупности.

Абстрагирование - процесс выделения существенных характеристик некоторого объекта, отличающих его от всех других видов объектов и, таким образом, четко определяющих его концептуальные границы с точки зрения наблюдателя.

Инкапсуляция - процесс разделения устройства и поведения объекта; инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации.

Модульность - состояние системы, разложенной на внутренне связные и слабо связанные между собой модули.

Иерархия - ранжирование или упорядочение абстракций.

Типизация - способ защититься от использования объектов одного класса вместо другого или, по крайней мере, способ управлять такой подменой.

Параллелизм - свойство, отличающее активные объекты от пассивных.

Сохраняемость - способность объекта существовать во времени и (или) в пространстве.

Объектно-ориентированное программирование (см. *управление объектом*)

Тип программирования, при котором программисты определяют не только типы данных (data type) и структур данных (data structure), но также и типы операций (функции-functions), которые могут применяться к структурам данных. Таким образом, структуры данных становятся объектами (object), которые включают одновременно и данные и функции. В дополнение, программисты могут создавать отношения между одним и другим объектом. К примеру, объект может наследовать (inherit) характеристики другого объекта. Одним из принципиальных преимуществ технологии объектно-ориентированного программирования по сравнению с технологией процедурного программирования заключается в возможности создания модулей (modules), которые не нуждаются в изменении при добавлении новых типов объектов. Это позволяет легко модифицировать объектно-ориентированные программы, для создания которых требуется применение объектно-ориентированного языка программирования (object-oriented programming language (OOPL). К наиболее популярным ОО языкам относятся Java, C++, Object Pascal и Smalltalk. (Webopedia)

Объектно-ориентированное проектирование (OO design)

Способ проектирования, включающий в себя описание процесса объектно-ориентированной декомпозиции и объектно-ориентированную нотацию для описания различных моделей системы (логической и физической, статической и динамической).

Объектно-ориентированный (object oriented)

Широко используемое выражение, которое может представлять много понятий, в зависимости от способа употребления. Объектно-ориентированное программирование относится к отрасли программирования, в котором комбинируются структуры данных с функциями для создания объектов повторного использования (re-usable objects). Объектно-ориентированная графика обозначает то же, что и векторная графика. В других случаях, термин объектно-ориентированный относится для описания систем, оперирующих в основном с разного типа объектами, в связи с чем, действия пользователя существенно зависят от типа объекта, которым он манипулирует. К примеру, программа объектно-ориентированного рисования может обеспечивать рисование много типов объектов, таких как окружности, треугольники, прямоугольники и др. Применение одинаковых методов к этим объектам, тем не менее, даёт разный результат. К примеру, если применяется метод Выполнить в 3D (Make 3D), результатами будут разные объекты: шар, пирамида и параллелепипед соответственно. (Webopedia).

Объектно-ориентированный анализ (OO analysis)

Способ анализа, изучающий требования к системе с точки зрения создания будущих классов и объектов, основанного на словаре предметной области.

Оверклокер

Пользователь компьютера, использующий разные технологии увеличения частоты работы процессора (разгона процессора), с целью превышения его паспортных характеристик.

Одноранговая сеть

Компьютерная сеть, все узлы которой обладают, примерно равными вычислительными возможностями и могут, по мере необходимости, выступать как в роли серверов, так и в роли рабочих станций.

Окно модальное (см. модальное окно)

Онлайн (on-line) (см. также *on-line*)

Интерактивный, диалоговый режим работы с системой.

Оперативная аналитическая обработка (OLAP - On-line Analitic Processing)

❶ Анализ многомерных данных, хранящихся в больших базах данных. OLAP позволяет легко и избирательно извлекать и обозревать данные для рассмотрения их с разных точек зрения.

❷ Технология аналитической обработки информации в режиме реального времени, включающая составление и динамическую публикацию отчетов и документов.

Оперативная память (work storage)

Память, в которой размещаются данные, над которыми непосредственно производятся операции процессора (ОП). Синонимы: ОЗУ–Оперативное Запоминающее Устройство, RAM–Random Access Storage.

Оперативное запоминающее устройство (ОЗУ) *(*RAM– random access memory*)*

Часто называется: ЗУ с произвольным доступом, ЗУ с произвольной выборкой, ЗУ с непосредственным доступом. Является быстрым запоминающим устройством, непосредственно связанным с центральным процессором и предназначенным для хранения данных, оперативно участвующих в выполнении арифметико-логических операций.

Оператор

❶ (языка) Базовая единица действия в языках программирования и алгоритмических языках. (ГОСТ).

❷ Элемент текста программы, выражающий целостное законченное действие (предложение).

❸ Действие, которое может быть выполнено над одним или несколькими операндами для получения результата в предложении языка программирования.

❹ Лицо, ответственное за текущий контроль состояния аппаратных средств вычислительных систем.

❺ (в кино) Один из создателей фильма, производящий съемку картины.

❻ (математическое) Закон, сопоставляющий одной функции или последовательности из определенного класса – другую функцию или последовательность. Например: оператор дифференцирования сопоставляет каждой дифференцируемой функции f(x) – функцию f′(x), являющуюся ее производной.

Оператор арифметический

Знак арифметической операции.

Операционная обстановка

Устанавливаемые пользователем параметры операционной системы, определяющие её рабочий интерфейс.

Операционная система

❶ Система программ, предназначенная для обеспечения определенного уровня эффективности вычислительной системы за счет автоматизированного управления ее работой и предоставляемых пользователям определенного набора услуг.

❷ Комплекс программных компонент, которые совместно управляют ресурсами вычислительной системы и процессами, использующими эти ресурсы. (Иллингуорт).

❸ Совокупность программных средств, обеспечивающих управление аппаратными ресурсами вычислительной системы, а также взаимодействие программных процессов с аппаратурой, другими процессами и пользователем. Операционная система выполняет следующие действия: управление памятью, управление вводом-выводом, управление файловой системой, управление взаимодействием процессов, обеспечение защиты данных и др.

❹ Комплекс программ, постоянно находящихся в памяти компьютера, позволяющих организовать управление устройствами машины и её взаимодействие с пользователями.

Операция (operation – действие)

❶ Действие, которое может быть выполнено над одним или несколькими операндами для получения результата. Обычно это действие обозначается символом операции, которая должна быть выполнена, а переменная задаёт конкретное значение данных для этой операции.(Иллингуорт)

❷ Знак операции, операция (обозначение операции в тексте).

❸ Операция элементарная – действие отдельного узла компьютера при выполнении им основных (базовых) операций типа запись, считывание, пересылка и т.д. (ГОСТ)

❹ (операция вычислительной системы) Действия, в совокупности составляющие выполнение команды процессора или другого электронного компонента.

❺ Элементарный шаг взаимодействия системы с окружением. Описание операции является шагом к этапу проектирования. (Гради Буч).

❻ Ряд действий, связанных с решением какой либо задачи, достижением определенной цели. (UML).

❼ Реализация услуги, которая может быть запрошена у любого объекта класса. (UML).

❽ Набор сообщений (messages), относящихся к одиночному действию Web-сервиса. (W3C).

Операция арифметическая

Операция, аргументы и результат которой являются числами.

Операция логическая

Операция, аргументы и результат которой принимают логические значения.

Операция, метод, функция

Элементы, происходящие от разных традиций программирования (Ada, Smalltalk, C++ соответственно). Фактически обозначают одно и то же. (Гради Буч)

Описание

Предложение в языке программирования, определяющее характер интерпретации высказываний, операторов или данных и их структур в этом языке. (ГОСТ).

Описание (descriptor)

Хранимый в памяти информационный объект, указывающий в каком виде запоминаются те или иные данные (например: в массиве, записи или в файле). Обратившись к дескриптору, программа получает возможность интерпретировать характеризуемые им данные.

Определение

❶ (математическое) Действие по конкретизации численных значений, вычисление по формулам.

❷ Формулировка, раскрывающая содержание понятия.

❸ (грамматическое) Второстепенный член предложения, отвечающий на вопросы: "Какой? Чей? Который?".

❹ постановление суда, вынесенное по частному вопросу, частное определение.

Оптимизация

Нахождение наибольшего или наименьшего значения какой-либо функции, выбор наилучшего (оптимального) варианта из множества возможных, например оптимизация управления.

Открытые системы (Open Systems)

В базовой концепции понятие открытая система, подразумевает комплекс средств, реализующих открытые (т.е. свободно распространяемые) спецификации или стандарты для интерфейсов, служб и форматов, с целью обеспечения вновь созданному прикладному программному средству следующих возможностей функционирования:

–перенос прикладного ПО с минимальными изменениями в широком диапазоне компонентов, платформ, информационных систем и компьютерных систем, приобретаемых у одного или нескольких поставщиков (мобильность);

–совместную работу с другими прикладными системами, расположенными на местных или удаленных платформах (интероперабельность);

–взаимодействие с пользователями в стиле, облегчающим им переход от системы к системе (портабельность, мобильность).

Отладка

Процесс выполнения программы с целью обнаружения ошибок.

Оферта

Товарное предложение.

Очко

Рельефное изображение буквы, при оформлении её в виде литеры, применяемой в типографском деле.

Ошибка

Неправильность в умозаключении, рассуждении, определении понятий, доказательстве и опровержении, вызванная нарушением законов и искажением форм мышления.

- П -

Пакет

В сети передачи данных представляет собой блок данных, имеющий определенную структуру, которая зависит от используемого протокола. Обычно включает в себя управляющую информацию (адрес получателя и и т.п.), передаваемые данные, биты контроля и исправления ошибок.

Память

❶ *(данных)* Функциональная часть компьютера, предназначенная для приема, хранения и выдачи данных;

❷ *(оперативная)* Память, в которой размещаются данные, над которыми непосредственно производятся операции процессора;

❸ *(кэш)* Запоминающее устройство (ЗУ) с малым временем доступа, используемое для временного хранения промежуточных результатов и содержимого часто используемых ячеек и используемого как буфер между процессором и оперативной памятью;

❹ *(динамическая) З*апоминающее устройство, в котором необходима периодическая регенерация хранимых данных.

Парадигма

Исходная концептуальная схема. Модель постановки проблем и их решения, а также комплекс методов исследования, господствующих в течение определённого исторического периода в научном сообществе. Смена парадигм представляет собой научную революцию.

Параллелизм (Parallelism)

Способность выполнять несколько функций одновременно.

Параметр (parameter) (от греч. parametreo - меряю, сопоставляя)

❶ (математическое) Величина, входящая в математическую формулу и сохраняющая постоянное значение в пределах одного явления или для данной частной задачи, но при переходе к другому явлению, к другой задаче меняющая свое значение.

❷ (математическое) Величина, числовые значения которой позволяют выделить определенный элемент (напр., кривую) из множества элементов (кривых) того же рода. Например, в уравнении $x^2 + y^2 = r^2$ величина r является параметром окружности.

❸ (физике и технике) Величина, характеризующая то или иное свойство какого-нибудь явления, например, теплопроводность, электропроводность тела, коэффициент его расширения или преломления и так далее. Параметры могут быть сосредоточенными (например, емкость электрического конденсатора, масса подвешенного к балке груза) и распределенными в пространстве (например, индуктивность линии электропередачи).

❹ То, что определяет структуру системы. Собственно параметры могут быть изменены входными значениями, но обычно параметры определяют, как входные воздействия или сигналы будут трансформироваться в выходные. В линейном уравнении $y = ax + b$, коэффициент *"a"* и откладываемое на оси *y* значение *"b"* являются параметрами; *"x"* является независимой переменной, а *"y"* – зависимой переменной. (Umpleby)

❺ (в абстрактном плане) Параметр это то, что вносит определённость. Параметр – это постоянная, чьё значение может меняться. Параметр – переменная, придающая определённость системе. (New York Times Magazine, l3 мая l979 г.).

❻ Объект, над которым выполняется процедура или от которого зависит её выполнение.

❼ Переменная, которой присваивается постоянное значение в рамках указанного применения и которая может указывать на применение. (ИСО 2382/2–76)

Параметр фактический и формальный (см. *фактический, формальный параметр*).

Паттерн (pattern) (также – шаблон) (см. *design pattern*)

Описание проблемы и метода её решения, позволяющее в дальнейшем использовать это решение в разных условиях. Представляет собой описание, в котором аккумулированы знания и опыт.

Паттерны проектные (см. *design patterns*)

Абстракции высокого уровня, которые документируют успешные проектные решения.

Первая нормальная форма (first normal form)

Первый уровень нормализации данных подразумевающий отсутствие повторяющихся групп. Таблица находится в первой нормальной форме, если она не содержит повторяющихся групп.

Перегрузка (overloading)

В языках программирования, свойство, которое позволяет объекту иметь различные значения или смысл, в зависимости от контекста, в котором он используется. Термин наиболее часто используется по отношению к операторам, которые могут по разному зависеть от типов данных, классов или операндов. К примеру, *x+y* может означать разные вещи, если *x* и *y* просто целые числа или сложные структуры данных. Не все языки программирования поддерживают перегрузку, но эта операция наиболее характерна для объектно-ориентированных языков программирования, включая C++ и Java. Перегрузка является одним из типов полиморфизма. (Webopedia)

Переменная

Программный объект, обладающий именем и значением, которое может быть получено и изменено программой.

Персептрон (Нейронные сети)

Обучаемая система, моделирующая восприятие и распознавание образов.

Персональные вычисления

Предоставленная миллионам людей возможность работать без посредников «один на один» с инструментом автоматизированной обработки информации – персональным компьютером.

Персональный компьютер (ПК)

❶ Устройство цифровой обработки информации (микроЭВМ универсального назначения), разработанное для использования одним человеком (пользователем) и предназначенное для ввода, обработки и вывода данных и информации. Типовой ПК состоит

из центрального процессора и основной памяти, а также долговременного запоминающего устройства на жёстком диске (винчестер), периферийных устройств ввода/вывода, включающих монитор, клавиатуру, мышь и принтер, а также операционную систему. Более мощные системы ПК, которые разработаны для обеспечения в компьютерных сетях данными, сервисами (услугами) и функциями широкого круга пользователей, называются серверами.

❷ Массовый инструмент активной формализации профессиональных знаний. По возможному влиянию на развитие индустриально развитого общества феномен персональных вычислений можно сравнить с началом эры всеобщей грамотности, которая стала возможной после изобретения книгопечатания.

❸ Интерфейс доступа к цифровым данным. Имеется ввиду, что без наличия ПК, в том числе и мобильного (беспроводного), использование цифровых данных невозможно.

Пиксел (pixel – Picture Element)

Наименьший элемент поверхности визуализации, которому может быть независимым образом заданы цвет, интенсивность и другие характеристики изображения. (ГОСТ)

Пиктограмма (иконка) (Icon)

❶ Ресурс, который можно добавить в загрузочный модуль приложения ОС Windows, представляющий собой графическое изображение небольшого размера, состоящее из отдельных пикселов. Обычно *пиктограммы* используются для обозначения свернутых окон приложений. По сути, это небольшая картинка с пояснительной надписью, которая связана с какой-либо программой или действием. Щелчок мышью по *пиктограмме* вызывает выполнение требуемого действия или программы, связанных с ней.

❷ Графическое представление (изображение) объекта на экране компьютера (аналог – иконка). Пиктограммы имеют компьютер (значок «Мой компьютер»), файлы, логические диски, принтеры и т.д.

Пин (pin)

Контакт для пайки или установки в разъём, не обязательно в виде проволочки.

Пиринговые подключения (см. peer-to-peer)

Платформа

❶ (в вычислительной технике) Совокупность аппаратных средств, программного обеспечения и интерфейсов, используемых в конкретных компьютерах. Обычно платформа определяется применяемой операционной системой и процессором.

❷ Помост, площадка.

❸ Грузовой вагон открытого типа с невысокими бортами.

❹ Небольшая ж.-д. станция, полустанок.

❺ Ж.-д. платформа – возвышенная площадка на станциях и остановочных пунктах у ж.д. путей.

❻ (в геологии) Область земной коры, характеризующаяся малой интенсивностью тектонических движений.

❼ (в политике) Программа, задачи или требования, выдвигаемые какой-либо партией, организацией, группой.

❽ Фасон женской обуви.

Плата *(или карта – сетевая, памяти, видеокарта и др.)*

❶ Плоская панель, содержащая набор интегральных схем, выполняющих определённые функции (платы расширения, материнские, сетевые, звуковые и другие карты). Как правило, имеет краевой печатный или штырьковый разъем, которым она соединяется со слотами шин ввода-вывода, а также металлическую скобу, которая закрепляет плату (карту) на корпусе.

❷ Пластина определённого размера из электроизоляционного материала, обычно прямоугольной формы, применяемая в электротехнической и электронной аппаратуре в

качестве основания для установки и механического закрепления навесных электро- и радиоэлементов (ЭРЭ) или нанесения печатных ЭРЭ.

Площадь

❶ Часть плоскости, ограниченная ломаной или кривой линией. К примеру, площадь прямоугольника, площадь криволинейной фигуры.

❷ Пространство, поверхность, естественно ограниченная или специально выделенная, отделенная для какой-нибудь цели. Количество эксплуатируемой, полезной площади. Также для обозначения в помещении, где пространство обычно измеряется в квадратных метрах поверхности пола. К примеру, жилая площадь, площадь дома.

❸ Большое, ровное и незастроенное место в пределах города или села. К примеру, базарная площадь, Красная площадь в Москве и др..

Поведение

❶ Любая последовательность состояний системы. (Ashby, Handout, 1961).

❷ Протокол наблюдаемых в системе изменений при переходе из текущего состояния в следующее. (Krippendorff)

Подпрограмма, процедура, (subroutine, procedure)

❶ Часть программы, предназначенная для выполнения определённой задачи (синонимы: routine, procedure, function, subroutine).

❷ Поименованная часть программы, которая вызывается и получает параметры, выполняет определённые действия и возвращает результат своей работы и управление в точку вызова. Во многих языках программирования различают два вида подпрограмм:

–*процедуры*, действие которых заключается в изменении значений параметров и некотором побочном эффекте. Обычно являются операторами или инструкциями языка программирования;

–*функции*, которые возвращают зависящий от параметров результат. Являются операндами в конструкциях языка программирования и описываемых с их помощью выражениях.

❸ Реализация метода в объектно-ориентированных программах, представляющая процедура или просто набор команд, сообщающих объекту, что нужно выполнить некоторую задачу и реализующих алгоритм её выполнения.

Подсистема

❶ Совокупность элементов, часть из которых задаёт спецификацию поведения других элементов.

❷ Система, являющаяся частью полной системы, выделенная по определённому аспекту или другим признакам деления. (ГОСТ).

❸ Часть большей системы, определяемая в подмножестве переменных этой большей системы. (Krippendorff)

Полиграфический растр (см. *lpi*).

Полиморфизм (polymorphism)

Обычно относится к способности проявления во многих формах. В объектно-ориентированном программировании, полиморфизм относится к способности языков программирования по-разному обрабатывать объекты, в зависимости от их типа данных или класса. Более точно, это можно назвать способностью языков переопределять методы для производных классов. К примеру, в заданном базовом классе фигура (shape), полиморфизм даёт программисту определять разные методы площадь (area) для любого количества производных классов, таких как окружности, треугольники и прямоугольники. Не имеет значения, какова форма объекта. Применение метода площадь всегда будет возвращать корректный результат. Наличие полиморфизма в языке программирования рассматривается в качестве необходимого условия, чтобы он считался действительно объектно-ориентированным. (Webopedia).

Полоса

❶ (математическое) Совокупность точек плоскости, лежащих между двумя параллельными прямыми этой плоскости.

❷ Длинная узкая форма, длинный узкий кусок чего-либо. К примеру, полоса железа, полоса материи.

❸ Одна из чередующихся параллельно частей какого-нибудь пространства. К примеру, полосы спектра, обои с белыми и голубыми полосами.

❹ Длинный узкий след, образуемый чем-нибудь.

❺ (типографское) Страница в наборе или уже отпечатанная.

❻ Период, эра, промежуток времени. К примеру, "…кончилась полоса неудач".

Полоса пропускания

Полосой пропускания (пропускной способностью) оценивается количество информации, которое может быть передано по каналу. Ширина полосы пропускания измеряется в битах в секунду (бит/с) – для цифровых сигналов или в герцах (Гц) – для аналоговых сигналов, например звуковых волн. Ширина полосы пропускания для аналоговой системы равна разности вычитания самой низшей передаваемой частоты из наивысшей. Например, ширина полосы пропускания, необходимой для передачи человеческого голоса составляет, примерно, 2700 Гц (3000-300). Чем шире полоса пропускания канала, тем больше данных может быть по нему передано. В цифровых коммуникациях это означает большую битовую скорость. В то же время, увеличение полосы пропускания, а, следовательно, повышение частоты сигнала, уменьшает длину волны. При более широкой полосе пропускания (выше частоты сигнала) возможна более скоростная передача. В этом случае, происходит уменьшение длительности импульсных сигналов, что приводит к их искажению и повышению вероятности возникновения ошибок. Этот эффект учитывается для сведения к минимуму искажения сигналов.

Ряд примерных полос пропускания цифровых каналов различного типа:

- цифровые телефонные линии – менее 100кбит/с
- сети ARCnet – 2,5 Мбит/с;
- сети ARCnet Plus – 20 Мбит/с;
- сети Ethernet – 10 Мбит/с;
- сети Token Ring – 1,4 или 16 Мбит/с;
- сети Fast Token Ring – 100Мбит/с;
- оптоволоконные сети (FDDI) – около 100 Мбит/с;
- сети ATM - около 655 Мбит/с.

Полоса частот

Частотный диапазон, в пределах которого происходит передача. Например, полоса частот стандартного телефонного канала связи находится в полосе между 300 и 4000 Гц.

Пользователь

❶ Человек или юридическое лицо, применяющие вычислительную систему или программное средство.

❷ Модуль программы или процесс, использующие средства, предоставляемые другим модулем или процессом.

Понятие

❶ (в философии) Форма мышления, отражающая существенные свойства связи и отношения предметов и явлений. Основная логическая функция понятия – выделение общего, которое достигается посредством отвлечения от всех особенностей отдельных предметов данного класса.

❷ (в логике) Мысль, в которой обобщаются и выделяются предметы некоторого класса по определённым общим и в совокупности специфическим для них признакам.

Порт

❶ Точка подключения внешнего устройства компьютера (принтера, сканера и др.) к внутренней шине процессора. Таким образом, программа или устройство могут посылать данные в порты или получать их из портов для обработки.

❷ Аппаратура сопряжения, содержащая цепи управления и позволяющая подключать устройства ввода-вывода к внутренней шине микропроцессора.

❸ Физический интерфейс для подключения компьютера, модема или другого коммуникационного оборудования.

Порт параллельный (логическое имя LPT)

Средство сопряжения процессора с устройствами низкого и среднего быстродействия при небольших объёмах передаваемой информации (принтер, сканер и др.).

Порт последовательный (логическое имя COM)

Поддерживает связь с низкоскоростными устройствами (модем и др.) в асинхронном режиме.

Портабельность

Взаимодействие программных систем и компонентов с пользователями в стиле, облегчающим им переход от системы к системе.

Портал (Portal)

❶ Web-сайт, который функционирует как "вход" ("doorway") в Интернет или как часть Интернета, представляющий определённую предметную область. В настоящее время существует множество разнообразных порталов и, в том числе, коммерческие, образовательные, программистские и другие порталы.

❷ Портал является одним из эффективных сценариев интеграции распределенных приложений в единую систему. В качестве интерфейса пользователя с системой в этом случае выступает, как правило, браузер, а поисковая машина обеспечивает «ворота» в Internet. Порталы, сочетающие набор служб, поисковую машину и службу новостей (актуализации), могут быть ориентированными на определенную сферу деятельности (вертикальными) или многоцелевыми (горизонтальными). Согласно Gartner, можно выделить следующие виды порталов.

- Мегапорталы (горизонтальные). Возникли одними из первых (Lycos, America Online, Yahoo!). Они обращались к сообществу Сети, а не отдельным группам пользователей. Основная функция таких порталов – быть специфически сетевым средством массовой информации.
- Вертикальные порталы. Иногда их называют нишевыми порталами или ворталами (vortals). Они предназначены для специфических групп пользователей – например, медицинские порталы, порталы для женщин и т.п.
- B2B порталы. Это электронные торговые площадки, которые разрабатываются для ведения бизнеса в Сети.
- Корпоративные порталы (Enterprise Portals). Разрабатываются для нужд одной компании, для решения, как внутрикорпоративных задач, так и для коммуникации с внешним миром – с покупателями, поставщиками, партнерами.

❸ Организация интегрированного подхода, состоящего в предоставлении пользователю унифицированного интерфейса для доступа к различным приложениям. Как правило, в роли подобного интерфейса выступает портал, обеспечивающий функции однократной регистрации в интегрированных системах, и "нулевой клиент". Портальное программное обеспечение может производить также адаптацию контента для устройств разного формата, его перевод на разные языки и персонализацию для каждого пользователя. Типичными примерами порталов являются Plumtree Portal, IBM WebSphere Portal и Microsoft SharePoint Portal. Стоит заметить, что современный портал – это некое обобщение рабочего экрана обычного ПК.

Портлет (стандартный портальный компонент)

❶ Реализация некоторого сервиса, запускаемая портальным сервером, которая содержит некоторые данные, набор собственных бизнес функций, а также стандартное представление на рабочих панелях портала. Портлеты обычно (но не обязательно) выглядят как стандартные "окошки" на рабочей панели браузера. С точки зрения пользователя, портлет – это небольшое окно на странице портала, которое предоставляет специфические функции или информацию, такие как календарь, заголовки новостей и др. С точки зрения разработчика, портлеты являются подключаемыми модулями (фактически – отдельными приложениями), которые разрабатываются для работы внутри портлет-контейнера портала.

❷ Реализация некоторого сервиса, запускаемая портальным сервером, которая содержит некоторые данные, набор собственных бизнес функций, а также стандартное представление на рабочих панелях портала. Портлет может содержать встроенный контент, который, в свою очередь, может быть представлен в самых разнообразных форматах, или ссылки на контент, находящийся на удаленном сервере. Тип разрабатываемого портлета зависит от его назначения, местоположения и объема информации, которую он должен отображать.

❸ Формат, разработанный корпорацией Oracle, в соответствии с которым на основе ее продуктов можно создавать, так называемые портлеты (portlets) — готовые компоненты, предназначенные для построения корпоративных порталов. Oracle также предлагает различные средства связи порталов, разработанных с помощью инструментария других фирм.

❹ Портлеты для программного продукта WebSphere Portal (IBM) представляют собой Java-сервлеты, разработанные на базе API портал-сервера. Дополнительная функциональность реализуется путем разработки новых портлетов. Помимо этого, на сервере IBM представлена библиотека портлетов, расширяющих функциональность портала.

Поток, нить (thread)

❶ Код из адресного пространства процесса выполняется в виде потоков. Поток (еще говорят «поток выполнения») — это единица выполнения, используемая операционной системой при планировании многозадачности (распределении ресурсов процессора). Первый поток процесса создается операционной системой и называется первичным потоком. Остальные потоки процесса порождаются первичным потоком. В Windows NT выполнение процесса завершается по завершении выполнения всех его потоков.

❷ Подпрограмма, выполняемая параллельно с главной программой (при этом главная программа тоже считается потоком, но этот поток ассоциирован с целым процессом). Поток может выполнять любую подпрограмму, а одна и та же подпрограмма может одновременно выполняться несколькими потоками. Все потоки имеют одно и тоже виртуальное адресное пространство, обращаются к одним и тем же глобальным переменным и ресурсам своего процесса. Поток является базовой единицей, которой операционная система выделяет время процессора.

Прагматика

В контексте языка UML – это та специфика объектно-ориентированного подхода, которая проявляется в организационных вопросах создания программного обеспечения (ПО). Сюда относятся: управление проектом, персоналом, рисками, версиями системы, конкретные программные средства поддержки разработки ПО и т. п. Важность этих вопросов обусловлена тем, что проектирование и анализ не являются строгой и формально определенной наукой. Поэтому для решения значительной части проблем не удается найти подходящих формализаций и остается только обсудить их на неформальном уровне. Таким образом, эта часть метода является самой неформальной.

Предложение (синоним – инструкция)

Базовая единица языка программирования, обладающая определённой для данного языка синтаксической и смысловой законченностью. (ГОСТ).

Предметная область

Класс задач, решаемых программным средством или программной системой.

Предметно-ориентированная база данных (Subject Oriented Databases) (см. *Витрина данных (Data Mart)*)

Представление

Образ ранее воспринятого предмета или явления (представление памяти, воспоминание), а также образ, созданный продуктивным воображением.

Представление данных

❶ Характеристика, выражающая правила кодирования элементов и образования конструкций данных на конкретном уровне рассмотрения в вычислительной системе. (ГОСТ).

❷ (в цифровой форме) Представление данных, при котором используются только цифровые знаки. (ГОСТ).

Представление знаний

Процесс структурирования предметных знаний с целью облегчения поиска решения задачи.

Представление чисел

Запись чисел при помощи заранее выбранного набора знаков и по заранее установленным правилам. (ГОСТ).

Преобразование

Процесс перехода от одной формы представления объекта к другой. (ГОСТ).

Преобразование типа

Операция программы, преобразующая значение одного типа в соответствующее значение другого типа. (ГОСТ).

Прерывание

❶ Обрыв нормальной последовательности выполнения инструкций в работе компьютера. Прерывание вызывает автоматическую передачу управления на заранее предопределённый адрес в памяти, где расположена последовательность команд, выполнение которых и составляет процесс прерывания.

❷ Внешний или внутренний сигнал, сообщающий процессору о необходимости прервать выполняемую программу и переключиться на процедуру обслуживания прерывания. Внешние прерывания обычно поступают от периферийных устройств, а внутренние вызываются ошибочными ситуациями. После обслуживания прерывания возобновляется выполнение прерванной программы.

Прикладной системный анализ

Научная дисциплина, которая на основе системно организованных, системно взаимосвязанных и функционально взаимодействующих эвристических процедур, методологических средств, математического аппарата, программного обеспечения и вычислительных возможностей компьютерных систем и сетей обеспечивает в условиях концептуальной неопределённости получение и накопление информации об исследуемом предмете для последующего формирования знаний о нём как едином, целостном объекте с позиции поставленных целей исследования и принятия рационального решения в условиях разнородных многофакторных рисков. (Панкратова)

Приложение (application, program) (см. *модель архитектуры производственных приложений*)

❶ Прикладная программа, то есть программа, выполняемая под управлением операционной системы.

❷ Компьютерная программа, выполняемая на командный стимул или из пакетного файла и позволяющая осуществить на компьютере конкретную работу.

–В широком смысле означает любую программу, отличающуюся от командного процессора (Command processor).

В более узком смысле подразумевает конкретную программу, например программу текстового процессора, базы данных, электронных таблиц, автоматизированного проектирования и т.д.

❸ В соответствии с подходом Microsoft, разработка приложения состоит из проектирования, моделирования, создания прототипа и в конечном итоге реализации и тестирования. На фазах проектирования и моделирования разрабатывается *архитектура приложения*. Почти все приложения содержат код представления, код обработки данных и код обращения к хранилищам данных. Архитектура приложения определяет то, как будет организован этот код. Для описания характеристик или типа приложения используется целый ряд терминов, в том числе:

–SDI;
–MDI;
–консольное;
–диалоговое;
–настольное;
–распределённое;
–одноярусное;
–двухъярусное;
–многоярусное;
–клиент/серверное;
–Web-приложение;
–Web-сервис;
–компонент;
–совместно работающее.

❹ (в языке Java) Программа, собранная в момент выполнения из отдельных компонентов, соединённых через сеть в отдельной конкретной среде выполнения, обычно располагаемой на разных платформах. Распределённые приложения поддерживают модели: двухъярусную (клиент/сервер), трёхъярусную (клиент/промежуточное ПО (middleware)/сервер), и многоярусную (клиент/ множественное промежуточное ПО/множество серверов).

Приложение диалоговое (См. *Wizards*)

Диалоговое приложение ведет пользователя через последовательность шагов к выполнению определенной задачи. Диалоговые приложения обычно активно взаимодействуют с пользователем через набор экранов (или диалоговых окон), посредством которых пользователь делает свой выбор. Хорошим примером диалоговых приложений являются довольно распространенные в среде Windows мастера (Wizards).

Приложение консольное

Приложение, у которого нет графического интерфейса. Вместо него взаимодействие пользователя и консольного приложения происходит путём ввода текстовых символов через командный интерфейс или интерфейс командной строки. Обычно у консольного приложения есть набор команд, которые можно использовать для доступа к функциональности приложения. Однако запомнить, какие команды, за что отвечают и каков их синтаксис, порой бывает затруднительно. В Windows чаще всего используются SDI- и MDI-интерфейсы, однако есть и другие варианты. Большинство приложений для мэйнфреймов и унаследованных приложений относятся к категории консольных. Эти приложения предназначены для использования с алфавитно-цифровым терминалом. В среде Windows терминал обычно заменяется программой эмуляции такового в рамках приложения Командная строка (Command Prompt) из главного меню кнопки Пуск Стандартные.

Приложения *модульные* (modularized)

Приложения, состоящие из нескольких меньших приложений, которые можно выполнять на различных вычислительных системах.

Приложение. Разработка концептуального и логического проектов приложения.

В соответствии с подходом, предлагаемом корпорацией Microsoft, считается, что разработка бизнес приложения, которое в полной мере удовлетворяло бы бизнес-требованиям и могло бы развиваться вместе с бизнесом, – процесс гораздо более сложный, чем простое написание компьютерной программы. Разработка решения (см. решение) бизнес-проблемы требует нескольких этапов: определения бизнес требований, выбора архитектуры решения, организации данных и размещения их по устройствам хранения.

Примитив (Primitive)

❶ (*primitive data tipe: целые, вещественные числа, логические и символьные переменные*) Типы данных, которые может использовать пользователь какого-либо конкретного типа вычислительного оборудования. Из них строятся более сложные структуры данных.

❷ (в компьютерной графике) Элементарный объект (отрезок прямой, треугольник, окружность и др.).

❸ (в программировании) Базовый элемент языка, используемый для создания сложных программ.

❹ Элемент, который нельзя разложить на более простые формы.

Принципы

Безусловные требования, которые должны быть удовлетворены в проекте. (Европейские правила геотехнического проектирования).

Проблема

❶ (в широком смысле) Сложный теоретический или практический вопрос, требующий изучения, исследования и разрешения.

❷ (в науке) Противоречивая ситуация, выступающая в виде противоположных позиций, в объяснении каких-либо явлений, объектов, процессов и требующая адекватной теории для её разъяснения.

❸ Буквально, «нечто, заброшенное вперёд (во времени)». В частности, распознанная неустойчивость или диспозиция, которые подвигают организм что-либо предпринять для изменения его текущего поведения либо смены существующего состояния. (Krippendorff)

Провайдер Интернет услуг (Internet Service Provider, ISP)

Компания или другая организация, предлагающие услуги по подключению к сети Интернет через свои компьютеры, которые являются частью всемирного Интернета.

Программа

❶ Последовательность операций или несколько параллельных последовательностей операций, выполняемых компьютером для достижения определённой цели.

❷ Последовательность команд или операторов, которая после декодирования её компьютером и транслирующей программой заставляет последний выполнить некоторую работу. (Фокс).

❸ Данные, предназначенные для управления конкретными компонентами системы обработки данных в целях реализации некоторого алгоритма.

❹ Упорядоченная последовательность команд, подлежащая обработке.

❺ Описание действий выполняемых компьютером, записанное на языке программирования или в машинных кодах.

❻ Программа описывает операции, которые нужно выполнить для решения поставленной задачи. Действия, предписываемые программой, называются операторами. Командой именуют элементарное предписание, предусматривающее выполнение какой-нибудь операции.

❼ Программа есть последовательность действий (операций), предложенная в целях достижения конкретного результата.

В зависимости от возлагаемых на них задач, в информатике различают много видов программ:

–системные, входящие в состав *операционных систем* (ОС);

–управляющие, предназначенные для управления работой систем либо их частей;

–прикладные программы, призванные выполнять задания пользователей;

–для определения качества программного обеспечения;

–начальной загрузки, восстановления, обеспечения запуска систем после отказов или ошибок;

–для ввода/вывода, осуществляющие ввод/вывод данных в/из компьютера (обычно называются драйверами);

–передачи данных;

–модем-программы;

–управления сетью;

–диагностики, локализации и объяснения неисправностей либо ошибок в работе;

–связи со специалистами и операторами, предназначенные для приёма и выполнения их команд и т.д.

Программа прикладная

Программный продукт, предназначенный для решения конкретной задачи пользователя.

Программа резидентная

Программа, постоянно размещаемая в оперативной памяти во время функционирования компьютера.

Программирование

❶ (в широком смысле) Все технические операции, необходимые для создания программы, включая анализ требований и все стадии разработки и реализации.

–(в узком смысле) Кодирование и тестирование программы в рамках некоторого конкретного проекта.

❷ Программирование (в «малом»). Для него характерны следующие признаки:

–код разрабатывается единственным программистом или небольшой группой. Отдельный индивидуум может понять все аспекты проекта от начала до конца.

–основная проблема при разработке состоит проектировании программы и написании кодов алгоритмов для решения поставленной задачи.

❸ Программирование (в «большом»). Наделяет проект следующими свойствами:

–программная система разрабатывается большой командой программистов. При этом одна группа может заниматься проектированием (или спецификацией) системы, другая – осуществлять написание кода отдельных компонентов. А третья – объединять фрагменты в конечный проект.

–нет ни одного человека, который бы знал всё о выполняемом проекте. Основная проблема в процессе разработки ПО – управление проектом и обмен информацией между группами и внутри групп.

❹ Деятельность, целью которой является описание процессов обработки данных.

Программист (programmer или programer)

❶ Человек, который программирует для компьютера, то есть пишет компьютерные программы.

❷ Лицо, которое разрабатывает, кодирует, тестирует и документирует компьютерные программы или Web-сайты.

❸ Юридическое лицо, работой или профессией которого является создание компьютерных программ.

❹ Профессия, связанная с написанием программных кодов. (См. *кодер, девелопер*).

Программист системный (systems programmer, аббревиатура – sysprog)

❶ Лицо, занимающееся написанием системных программ, предназначенных для обеспечения функционирования компьютерных систем, в противоположность тем, кто занимается разработкой программ-приложений.

❷ Технический эксперт в больших корпорациях, занимающийся поддержанием работоспособности компьютерных систем, а также ответственный за инсталляцию и интеграцию новых программных продуктов и аппаратных решений.

❸ Общий термин для широкого диапазона знаний и возможностей специалистов, включающих написание низкоуровневых кодов программ, относящихся к операционным системам или серверам. Круг знаний системного программиста должен включать следующие вопросы: конкретные операционные системы, сетевые технологии (TCP/IP, ATM, Ethernet, DNS), электронная почта (POP, IMAP, SMTP), Web-серверы, СУБД, операционные системы и безопасность в сетях, а также аппаратное обеспечение (SCSI, жёсткие диски и устройства долговременного хранения данных (бэк-ап – back-up devices)).

Программная инженерия (см. *инженерия программного обеспечения*)

Программная система

Программная продукция, представляющая собой совокупность программ и/или подсистем, имеющих общее целевое назначение. Связь между компонентами устанавливается разработчиком, пользователем или другими специалистами при инсталляции.

Программно-аппаратные средства (firmware)

❶ Программное обеспечение, хранимое, как правило, в постоянных запоминающих устройствах.

❷ Понятие, используемое одновременного указания на программные и технические средства.

Программное изделие (ПИ)

Экземпляр или копия разработанного программного средства. Изготовление ПИ – это процесс снятия копии программы и программных документов ПС с целью их поставки пользователю для применения по назначению. (ГОСТ).

Программное обеспечение (ПО) (software)

❶ Комплекс взаимосвязанных программных модулей, предназначенных для решения конкретной задачи или определённого класса задач, отчуждаемый от программистов-разработчиков, снабжённый в соответствии с заданными требованиями необходимой технической и технологической документацией и обладающий товарной стоимостью. (ГОСТ).

❷ Продукт интеллектуальной деятельности, включающий в себя информацию, выраженную через средства поддержки. ПО может быть представлено в форме концепции, протоколов, спецификаций или методик. Компьютерная программа является конкретным примером программного обеспечения. (Терминология ISO 9000)

❸ Комплекс программ или программный продукт, обеспечивающие обработку или передачу данных**,** а также разработку новых программ.

❹ Программное обеспечение – это не только программы, но и вся сопутствующая документация, а также конфигурационные данные, необходимые для корректной работы программ. Программные системы, как правило, состоят из совокупности программ и файлов конфигурации (необходимых для установки этих программ), а также документации, которая описывает структуру системы и содержит инструкции для пользователей, объясняющие работу с системой.Сюда же включается адрес Web-узла, где пользователь может найти самую последнюю информацию о данном программном продукте и его обновления. (Иан Соммервилл).

Программное средство (ПС)

Программа или логически связанная совокупность программ на носителях данных, снабжённая программной документацией. (ГОСТ).

Программный продукт

❶ Программа, предназначенная на продажу и реализуемая подобно любой другой продукции.

❷ Современное законодательство определяет авторское право на создание программы. Авторская копия программы называется **программным продуктом.**

Программный продукт это не просто программа, записанная на диске и проданная пользователям. Это **система мероприятий**, связанных с её использованием. Такая система мероприятий включает:

–техническую и информационную поддержку пользователя;

–предоставление гарантий, что программа будет нормально работать в соответствии с прилагаемой инструкцией;

–продажу последующих версий программы по сниженным ценам.

Из-за невозможности заранее узнать эксплуатационные характеристики программы, она может быть возвращена продавцу в срок, обычно не более 60 дней.

Программным продуктом обычно называют программу, предназначенную на продажу и реализуемую подобно любой другой продукции.

Программы, созданные для себя, для собственных нужд автора и не предназначенные для широкого распространения называются утилитарными.

Распространяемые программы и программные продукты могут оказаться коммерческими (платными), свободными (бесплатными) (freeware), демонстрационными (demo), условно-бесплатными (shareware) и общедоступными (public domain).

Демонстрационные программы обычно не дают использовать наиболее интересные из заложенных в них возможностей, это только образцы, чтобы потенциальный покупатель составил представление об продукте.

Условно-бесплатные программы даются пользователю для работы на определённое время, как правило, на срок от 2 до 6 недель, или до совершения им покупки. Если после того, как пользователь испытал программу и совершил покупку, он становится зарегистрированным пользователем. Это сулит известные преимущества: техническую поддержку разработчика, скидки при покупке новых версий, возможность бесплатного обновления и т.д.

Общедоступное, так же как и свободное, программное обеспечение бесплатно и может свободно распространяться между пользователями. К тому же, оно не защищено авторскими правами.

Проектирование

❶ Процесс создания проекта - прототипа, прообраза предполагаемого или возможного объекта или состояния. Наряду с традиционными видами проектирования (архитектурно-строительное, машиностроительное, технологическое и др.) начали складываться самостоятельные направления. Особенно активно в последнее время развиваются: проектирование компьютерных, программных, информационных, человеко-машинных и других систем, трудовых процессов, организаций, экологическое, социальное, инженерно-психологическое, генетическое и другое проектирование.

❷ (*прямое*) Преобразование модели в исполняемый код, выполненное на каком либо языке программирования. (UML).

❸ (*обратное проектирование – reverse engineering*) Процесс преобразования кода, написанного на каком-либо языке программирования (C++, Delphi, Java и др.) в модель (в том числе и в языковую!). Например, на язык ассемблера. (UML).

Проектирование концептуальное (см. *концептуальное проектирование*).

Проектный (конструкторский) шаблон (см. *Design pattern*)

Производительность (performance)

Показатель того, насколько быстро работает система, то есть насколько быстро она выполняет возложенные на неё задачи. Производительность измеряется по множеству различных показателей, учитывающих влияние целого ряда факторов, таких как рабочая нагрузка, аппаратная конфигурация и операции над базой данных. Но то, как именно влияют на производительность системы эти факторы, зависит от архитектуры решения, а также как оно спроектировано.

Пространственные данные (Spatial Data) (см. *данные пространственные*)

Протокол

❶ (протокол взаимосвязи) Набор семантических и синтаксических правил, определяющих взаимосвязь логических объектов уровня при обмене данными (ГОСТ).

❷ Соглашение, касающееся управления процедурами информационного обмена между взаимодействующими объектами. Информация передаётся в отдалённый пункт с использованием протокола самого нижнего уровня и далее продвигается вверх через систему интерфейсов, пока не достигнет соответствующего уровня в пункте назначения. Набор интерфейсов управляет обменом между уровнями протоколов. В совокупности с набором протоколов, управляющих обменом информацией между связанными объектами на данном уровне, они вместе образуют систему, называемую иерархией протоколов; (Семиуровневая модель OSI).

❸ Полный набор операций, которые объект может осуществить над другим объектом. (Гради Буч).

❹ (*передачи данных*) Набор правил и соглашений, определяющих форматы данных и процедуры передачи для обмена информацией между взаимодействующими процессами, функциональными или логическими модулями, абонентскими станциями и т.д.

❺ Набор правил, управляющих коммуникациями между процессами. Протокол определяет формат и содержание сообщений, которыми обмениваются процессы.

Прототип

Начальная версия программной системы, которая используется для демонстрации концепция, заложенных в системе, проверки вариантов требований, а также поиска проблем, которые могут возникать как в ходе разработки, так и при эксплуатации системы.

Прототипирование

Процесс создания начальной версии программной системы.

Профилировка

Сбор данных о ходе выполнения программы, т.е. количество выполнений для каждого оператора, число обращений к переменным, число вызовов подпрограмм и т.д.

Профиль (профайл) потребителя (Consumer Profile)

Идентификация лица, группы или приложения, а также профиль (профайл) необходимых им и используемых ими данных: виды хранимых данных, физические реляционные таблицы, расположение и периодичность данных (когда, где и в какой форме они должны быть предоставлены).

Процедура (см. *подпрограмма*)

Процесс (от лат. processus - продвижение)

❶ Последовательность сменяющих друг друга состояний некоторой информационной среды.

❷ (в естественных науках) Означает изменение системы во времени (то есть её "движение"), которое (в общем случае) заранее не известно чем кончится и кончится ли вообще.

❸ Выполняемая программа, имеющая собственное виртуальное адресное пространство, код, данные, а также потребляющая ресурсы операционной системы, такие, как файлы, окна и т.д. Процессы порождаются запуском новых экземпляров приложений.

❹ Последовательность операций при выполнении программы или части программы, а также данные, используемые этими операциями.

❺ (Процесс) разработки программного обеспечения, т.е. шаги и указания, по которым разрабатывается система.

❻ Последовательная смена явлений, состояний в развитии чего-нибудь.

❼ Совокупность последовательных действий для достижения какого-либо результата (напр., производственный процесс).

❽ Описание целей, видов деятельности, результатов и мер прогресса для различных фаз объектно-ориентированного анализа и проектирования. Процесс не формализуется как набор процедур, а делится на части, для которых описываются интерфейсные характеристики.

Процесс разработки (development process)

Ряд целенаправленных и заранее определенных шагов, предпринимаемых для производства программного обеспечения, как процесса, поддающегося управлению и тиражированию. Основной целью процесса разработки программного обеспечения является достижение успешного и качественного завершения реализации системы в целом. (UML).

Процесс в бизнесе (см. *бизнес-процесс*)

Процессор (микропроцессор)

❶ Программируемое логическое устройство, изготовленное в монолитном кристалле.

❷ Микросхема, реализующая функции центрального процессора персонального компьютера, называется микропроцессором. Обязательными компонентами микропроцессора является арифметико – логическое устройство (АЛУ) и блок управления.

Прямое проектирование

Подразумевает преобразование модели программной системы в исполняемый код на каком-либо языке программирования. (UML).

Пункт

Единица измерения высоты шрифтов, равная 0.376 мм.

- P -

Развёртывание (См. *deployment*)

Размер

❶ (в вычислительной технике) (размер, размерность массива) Атрибут описания, определяющий число элементов для вектора, число индексов или диапазон значений определённого индекса для многомерного массива. (См. массив (данных)).

❷ Величина чего-нибудь в одном измерении. К примеру, линия размером в пять сантиметров. Размер палки - полметра.

❸ Величина какого-нибудь предмета во всех измерениях. К примеру, дом огромных размеров, размер участка.

❹ Количество, величина денежной суммы. К примеру, размер зарплаты, налога.

❺ Мерка какого-нибудь изделия, номер вещи, обозначающий ее величину. К примеру, размер ботинок, размер воротничка, размер костюма.

❻ (в переносном смысле) Степень, пределы охвата, величина какого-нибудь явления. Безработица в странах капитала достигает небывалых размеров.

❼ (литературное) То или иное количество и расположение слогов в стихе (напр. чередование через определенные промежутки слогов ударного и неударного, или кратких и долгих, или повторение одинакового количества слогов через равные промежутки и т. п.), от которого зависит звуковой мелодический строй стиха. Двухдольные, трехдольные размеры

(создаваемые повторением одинаково построенных двух, трех слогов). В русской поэзии наиболее употребительные размеры *ямб* и *хорей*.

❽ То или иное количество и расположение ритмических единиц в музыкальном такте, создающее ритмический строй музыкального произведения, музыкальный счет.

Разработка

❶ Процесс систематического изменения структуры системы. В кибернетическом плане напрямую связан с изменением организации. (Krippendorff).

❷ Событие, вызывающее изменение, то есть процесс, приводящий к изменению или прогрессу в ситуации. Создание чего-либо нового.

❸ Разработка чего-либо: процесс разработки.

❹ Быть разрабатываемым: состояние, в котором нечто разрабатываемое ещё не завершено.

❺ (музыкальное) Развитие музыкальной темы.

❻ Определение лучших методов при применении новых устройств или процессов для производства товаров или услуг.

Разработка приложений

Процесс, подразумевающий три различных вида деятельности:

–разработку концептуального и логического проектов приложения;

–проектирование интерфейса и служб пользователя;

–разработку физического проекта;

На окончательном этапе производится написание и отладка кода приложений.

Разработчик

Человек или компания, выполняющий работы по создания чего-либо.

Разрешение (при выводе на экран дисплея, на принтер и т.д.)

Термин "разрешение" используется для определения количества единичных элементов растровой карты изображения, приходящихся на единицу длины изображения или для определения общего количества единичных элементов для фиксированных значений длины и ширины при выводе на экран дисплея (монитора) компьютера. К примеру, обычно разрешение мониторов записывается в виде 640x480, 800x600 и т.д. Первая цифра указывает общее количество единичных элементов по ширине, вторая - по высоте. Чем выше разрешение, тем точнее растровая карта воспроизводит изображение и тем больше общее количество единичных элементов (пикселов) и, соответственно, размер файла, в котором хранится картинка. Каждое периферийное устройство (принтер, сканер, дисплей), которое вводит или выводит изображение, имеет конкретное разрешение. Профессиональные принтеры и сканеры имеют самое высокое разрешение, обычно выражающееся в DPI (dots per inch - точек на дюйм) или в ppi (pixels per inch - пикселов на дюйм). Разрешение показывает сколько точек (или пикселов) размещается в одном линейном дюйме. Разрешение компьютерного монитора составляет примерно 72 dpi, а у принтеров разрешение бывает в диапазоне от 150 до 1440 dpi (для моделей с наивысшим разрешением). У сканеров обычно разрешение составляет от 300 dpi и выше.

Разряд

❶ (в вычислительной технике) Место, которое может занимать литера в позиционном представлении числа и которое можно идентифицировать своим порядковым номером. (ГОСТ 19781–83)

❷ (математическое) Место, занимаемое цифрами при письменном обозначении числа.

❸ (ботаническое) Отдел, группа, род, категория в каком-нибудь подразделении предметов или явлений, различающихся по тем или иным признакам. К примеру, разряд растений.

Распознавание (identification, recognition)

Процесс отождествления объекта с одним из известных системе объектов.

Распределённое предприятие (distributed enterprise)

Объединённая инфраструктура (управленческая и информационная), представляющая весь комплекс подразделений крупных организаций или транснациональных корпораций, расположенных географически и территориально удалённо друг от друга. К таким организациям относятся, к примеру, транснациональные корпорации, налоговые инспекции, крупные международные банки и нефтедобывающие компании, а также и многие другие.

Распределённое приложение (distributed application)

❶ Модель приложения, в которой несколько приложений, выполняющихся раздельно, но совместно друг с другом, работают над задачей сообща. Различные приложения могут быть распределены в пределах одной системы или по нескольким вычислительным системам.

❷ Приложение, собранное в момент выполнения из отдельных компонентов в отдельной конкретной среде выполнения, обычно располагаемой на разных платформах, соединённых через сеть. Распределённые приложения поддерживают модели: двухъярусную (клиент – сервер), трёхъярусную (клиент – промежуточное ПО (middleware) – сервер), и многоярусную (клиент – множественное промежуточное ПО – множество серверов).

Распределённые вычисления (distributed computing)

❶ Парадигма организации приложений, в которой различные части программы могут исполняться на разных компьютерах в сети.

❷ Тип компьютерных вычислений, при выполнении которых различные компоненты и объекты, включённые в приложение, располагаются на разных компьютерах, соединённых в сеть. К примеру, приложение-процессор обработки слов (Word) может состоять из компонента редактора, находящегося на одном компьютере, объекта проверки правописания на втором, а словарь – на третьем. В некоторых системах распределённых вычислений каждый из трёх компонентов может выполняться под управлением трёх разных операционных систем. Одним из требований к процессу распределённых вычислений является необходимость наличия набора стандартов, которые специфицируют требования, как объекты взаимодействуют друг с другом. В настоящее время существуют два ведущих стандарта компьютерных распределённых вычислений: CORBA и DCOM. (Webopedia)

Распределённые системы (distributed systems)

Системы, состоящие из нескольких частей, обычно взаимодействующих друг с другом по сети. Разбивать систему на части приходится потому, что возможности автономной компьютерной системы не способны покрыть потребности серьезной информационной системы в вычислительных возможностях и хранении данных. К тому же разбиение на части с внесением избыточности в эти части обеспечивает механизм восстановления системы после сбоев.

Растр

Двумерная прямоугольная сетка пикселов, соотнесённая с процессом вывода их на экран компьютера.

Растровая графика

Технология представления графических изображений на экране компьютера в виде набора квадратных ячеек, причём, каждый квадрат содержит какой-либо элемент исходного изображения. Для каждой ячейки выбирается некое постоянное значение цветового оттенка, например, методом простого усреднения. Если теперь пронумеровать ячейки от первой до последней, будет получен набор пар цифр - первая представляет собой номер квадрата, вторая описывает усредненный оттенок цвета. Именно такой метод лежит в основе описания любого растрового изображения. Сетка, создаваемая для реальной оцифровки произвольного изображения, содержит огромное количество ячеек настолько малых, что глаз человека их не видит, воспринимая все изображение как целое. Сама сетка получила название растровой карты, а ее единичный элемент (квадратная ячейка) называется растром или пикселом (от

английского picsel - PICture Single ELement). Растровая карта представляет собой набор (массив) троек чисел: две координаты растра на плоскости и его цвет.

Растровая карта

Сетка, создаваемая для реальной оцифровки произвольного изображения, которая содержит огромное количество ячеек настолько малых, что глаз человека их не видит, воспринимая все изображение как целое. Растровая карта представляет собой набор (массив) троек чисел: две координаты растра на плоскости и его цвет.

Растровый рисунок

Рисунок, рассматриваемый как матрица точек, с каждой из которых можно работать отдельно. Растровые рисунки получаются в результате сканирования и фотографирования.

Расширяемость (extensibility)

❶ Способность легко добавлять новые функции к существующим службам без изменения основных программ или без переопределения основной архитектуры.

❷ Приспособленность *решения* (программы) к дальнейшему расширению и улучшению его функций.

Реализация (implementation)

❶ Реализация спецификации (Realization of a specification). (W3C).

❷ Отношение между спецификацией и её программной реализацией. (UML).

❸ Указание на то, что поведение наследуется без структуры. (UML).

Регистр (register)

❶ Внутреннее запоминающее устройство процессора или адаптера для временного хранения обрабатываемой или управляющей информации.

❷ Использование бистабильных устройств для хранения информации в вычислительных системах и обеспечения быстрого доступа к ней.

Редиректор

❶ Программные средства клиентской части операционной системы (ОС), используемые для запроса доступа к удаленным ресурсам и услугам, а также и их использования в компьютерных сетях. Эта часть ОС выполняет распознавание и перенаправление в сеть запросов к удаленным ресурсам от приложений и пользователей, при этом запрос поступает от приложения в локальной форме, а передается в сеть в другой форме, соответствующей требованиям сервера. Клиентская часть также осуществляет прием ответов от серверов и преобразование их в локальный формат, так что для приложения выполнение локальных и удаленных запросов неразличимо.

❷ Компонент клиентской части, который перехватывает все запросы, поступающие от приложений, и анализирует их.

Реентерабельность

Свойство программы, корректно выполняться при рекурсивном вызове из прерывания. Операционная система DOS – нереентерабельная программа, поэтому для вызова её функций из резидентных программ необходимо использование специальных средств синхронизации (проверки " занятости" DOS). Так, как DOS была спроектирована исключительно как однозадачная операционная система, понятие задачи или процесса в ней вообще не предусмотрено. Это значит, что при выполнении конкретной программы в DOS ее текущее состояние и характеристики занимаемых ресурсов " размазаны" по различным переменным. Эти разрозненные переменные и составляют ***контекст*** текущей выполняемой программы (задачи). В то же время резидентная программа, хотя и вызывается по прерыванию, при выполнении функций DOS, должна быть заявлена как отдельный процесс (задача).

Резидентная программа

Программа, постоянно размещаемая в оперативной памяти во время функционирования компьютера.

Реинжиниринг (Reengineering, BPR - business process reengineering)

❶ Перепроектирование.

❷ (в бизнесе) Реинжиниринг определяется как фундаментальное переосмысление и радикальное перепланирование бизнес-процессов компаний, имеющее целью резкое улучшение показателей их деятельности, таких как затраты, качество, сервис и скорость.

❸ (в программной инженерии) Повторная реализация *наследуемой системы* в целях повышения удобства её эксплуатации.

Релевантный

Имеющий отношение к данному вопросу, проблеме, предметной области.

Реляционная база данных (relational) БД (РБД)

❶ Реляционная модель базы данных была разработана в конце шестидесятых годов Эдгаром Ф. Коддом. В ее основе лежат теория множеств и исчисление предикатов, являющиеся ответвлениями теоретической математики. Основная идея реляционной модели следующая: данные, организуются в таблицы, над которыми можно производить операции для получения новых таблиц. Кодд назвал эти таблицы *связями* (relations), подразумевая связанный набор информации, отсюда и термин *реляционная база данных* (relational database).Таким образом, реляционной называется база данных, в которой все данные, доступные пользователю, организованны в виде таблиц, а все операции над данными сводятся к операциям над этими таблицами. Реляционные базы моделируют некоторую часть реального мира. В частности, в них хранятся сведения о различных объектах некоторой предметной области.

❷ РБД называется набор отношений. Данные в такой базе хранятся в плоских таблицах. Каждая таблица имеет собственный, заранее определенный набор именованных колонок (полей). Поля таблицы обычно соответствуют атрибутам сущностей, которые необходимо хранить в базе. Количество строк (записей) в таблице неограниченно, и каждая запись соответствует отдельной сущности. Каждая таблица должна иметь первичный ключ (ПК) — поле или набор полей, содержимое которых однозначно определяет запись в таблице и отличает ее от других. Связь между двумя таблицами обычно образуется при добавлении в первую таблицу поля, содержащего значение первичного ключа второй таблицы. Реляционные СУБД (РСУБД) предоставляют средства для всевозможных пересечений и объединений любых таблиц, отбора записей по разнообразным условиям, группировки и сортировки результатов. РБД сочетает наглядность представления информации с простотой (относительной) реализации своей концепции и является наиболее популярной структурой для хранения данных на сегодняшний день.

Репликация

Распределенные компьютерные системы часто обеспечивают репликацию (тиражирование) файлов в качестве одной из услуг, предоставляемых клиентам. Репликация - это асинхронный перенос изменений данных исходной файловой системы в файловые системы, принадлежащие различным узлам распределенной файловой системы. Другими словами, система оперирует несколькими копиями файлов, причем каждая копия находится на отдельном файловом сервере и видоизменяется самостоятельно.

Репозиторий (repository)

❶ Хранилище метаинформации (описательной информации) о разрабатываемых приложениях, компонентах, хранилищах данных, базах данных и т.д. Содержит также, модели процессов, происходящих в системе, модели данных, используемых системой и объектную модель с соответствующим описанием каждой из компонент. Как правило, специализированный крупный программный продукт или часть другого программного обеспечения (ПО).

❷ Электронное хранилище структурированной информации.

Ресурс (вычислительной системы)

Средство вычислительной системы или компьютера, которое может быть выделено процессу обработки данных (программе пользователя) на определённый момент времени. Основными ресурсами компьютера являются: процессоры, области основной памяти, наборы данных, периферийные (внешние) устройства, программы и т.д. (ГОСТ).

Решение (decision)

Намеренное наложение ограничений на набор первоначально возможных альтернатив.. (Krippendorff).

Решение (solution)

В повседневном смысле **решение (solution)** – это просто стратегия или метод, позволяющие решить проблему. На жаргоне IT-индустрии "решениями" все чаще называют программные продукты, являющиеся крупным приложением или комплексом крупных приложений. Поэтому время от времени возникает недопонимание или даже скептицизм в отношении того, что в действительности понимается под решением. В MSF (Microsoft Solution Framework) термин "решение" имеет очень специфическое значение. Это скоординированная поставка набора элементов (таких как программно-технические средства, документация, обучение и сопровождение), необходимых для удовлетворения некоторой бизнес–потребности конкретного заказчика. Хотя MSF и используется при разработке коммерческих продуктов для массового потребительского рынка, он концентрируется главным образом на поставке решений, предназначенных для определенного заказчика. В состав завершённого решения, как правило, входят следующие компоненты:

–программно-технические средства / разрабатываемый код
–процесс внедрения
–документация
–коммуникации
–обучение
–поддержка.

Риск (risk, hazard)

❶ В теории принятия решений и в статистике, риск означает неопределённость, для которой известно распределение вероятности. Соответственно анализ рисков относится к области знаний, в которой определяется результат и последствия принятых решений, вместе с их вероятностью. В системном анализе, лицо принимающее решение часто рассматривает вероятность того, что проект (при выбранных условиях и альтернативах или переменных) не может быть выполнен в запланированные сроки и за выделенную сумму денег. Риск неблагоприятного исхода (risk of failure) может отличаться от альтернативы к альтернативе и может быть оценен или вычислен при частном анализе.

❷ В некоторых случаях, риск означает неизвестное и чрезвычайно вредное воздействие, как, например «риск от влияния ядерных предприятий на здоровье населения может быть…». В этом случае анализ рисков (risk analysis) или оценка рисков (risk assessment) может представляться исследованием, состоящим из двух частей. Первая занимается определением собственно степени вредного воздействия, а вторая – определения ожидаемых вероятностей от воздействия.

❸ Согласно глоссария организации Society for Risk Analysis (SRA) (США), различаются следующие понятия, относящиеся к термину «*риск*».

Риск (risk) – потенциальная возможность для реализации (осуществления) нежелательных, неблагоприятных, вредных или пагубных последствий для челевеческой жизни, здоровья, собственности, земельного хозяйства или окружающей среды. Оценка (estimation) риска обычно базируется на ожидаемом значении (величине) условной вероятности появления события, важностью события и количеством возможных случаев его проявлений.

Риск-анализ (Risk analysis) – подробное исследование (изучение, рассмотрение), включающее оценку риска, определение риска и варианты управления риском, выполняемые для понимания природы и особенностей нежелательных, негативных последствий для человеческой жизни, здоровья, собственности, земельного хозяйства или окружающей среды и аналитическую обработку для получения информации, относящейся к этим нежелательным событиям, а также обработку

количественного описания вероятностей и ожидаемых пагубных последствий для идентификации рисков.

Оценка риска (Risk assessment) – процедура (процесс) выяснения информации в отношении допустимых уровней риска и/или уровней риска для индивидуальности, группы, общества или окружающей среды.

Вычисление риска (Risk estimation) – научное вычисление (определение) свойств и характеристик рисков, обычно в количественной форме, если это возможно.

Таксация риска (Risk evaluation) – составная часть (компонент) оценки (assessment) риска, в которых суждения (оценки) выполнены с учётом значимости и приемлемости риска.

Относительный риск (Relative risk)– относительный показатель степени отрицательных последствий (поражения) (обычно в сфере действия или срока службы) среди тех, которые обнаружены, относительно тех, которые не произошли.

Роль

❶ Именованное поведение некоторой сущности в конкретном контексте или, другими словами, лицо, которым абстракция обращена к миру (например, субъект в информационной системе может описываться как: сотрудник организации, юридическое лицо, покупатель, пассажир и т.д.). (UML).

❷ Именованный слот в объектной структуре, который представляет поведение элемента, находящегося в определённом контексте. (UML).

Роутер (маршрутизатор) (router)

Также как и другие сетевые устройства (мосты (bridge), шлюзы (gateway), хабы (hub) и переключатели (switch)), роутер в Internet служит устройством обеспечения функциональности и работоспособности сетевых коммуникаций. Является либо аппаратным устройством или программным обеспечением. Определяет следующую точку в вычислительной сети, куда должен направляться очередной пакет с информацией.

- C -

Сайт (см. также *Web-сайт*)

Определенное место в Internet, располагаемое на одном из миллионов серверов, доступное из любой точки мирового пространства, представляющее компанию или частное лицо. Сайт состоит из одной или нескольких страниц, объединенных по смыслу, навигационно и по месту расположения (на логическом уровне), а также, как правило, имеющих единый стиль оформления.

Сбор данных (data acquisition)

❶ Выделение и первичная обработка параметров физического или информационного процесса для последующей обработки на компьютере. Обычно подразумевается ввод данных с терминалов и датчиков.

❷ Процесс идентификации, выделения и накопления исходных данных, подлежащих централизованной обработке.

Свойство (особенность, черта, признак) (feature)

❶ (в вычислительной технике) Важные свойства устройств или программных приложений. Многие аналитики бурно обсуждают появившийся термин «улучшизм» (*featuritis)*, приводящий к многочисленным новым улучшениям и упрощениям приложений. Одно из принципиальных преимуществ современных приложений заключается в предложении массы новых свойств, без усложнения их кода. (Webopedia).

❷ Качество, признак, способность, характеризующие объект или составляющие отличительную особенность какого-либо объекта.

❸ Абстрактная часть функциональных возможностей (An abstract piece of functionality). (W3C).

❹ (в геоинформатике) Пространственный объект. Представление объекта реального мира (real-world object) в слое карты (layer on a map).

Свойство (property)

Характеристика объекта. Во многих языках программирования, термин свойства употребляется для описания атрибутов (attributes), ассоциируемых со структурами данных.

Связывание (имени и сущности)

Отображение имени на соответствующую сущность. В рамках контекста имя может иметь не более одного связывания. Допускается ссылка с помощью другого имени на контекст. Структурированное или составное имя формируется в виде комбинации имени контекста и имени сущности в данном контексте, которые разделены точкой. К примеру, в языке Visual Basic for Application связывание имени объекта с его свойствами прозводится через разделение точкой следующим образом: *Object.property*. То есть, обращение к свойству *StatusBar* объекта *Application* запишется в виде: *Application.StatusBar*.

Связь (connection)

Виртуальная связь (контур) транспортного уровня, установленная между двумя программами с целью коммуникации (связи и совместной работы). (W3C).

Семантика

❶ Значение слова, оборот речи или грамматической формы.

❷ Формальная спецификация значения и поведения чего-либо. (UML).

❸ Значение или значения языковых единиц (слов, фразеологизмов, словосочетаний, предложений).

❹ Система строгих правил, определяющих смысл, назаначение и функции элементов языка программирования.

Сервер

❶) Компьютер, предоставляющий свои ресурсы другим компьютерам. В любой программе удаленного доступа ***сервером*** называется управляемый компьютер, клиентом - управляющий. Также под этим термином подразумевается компьютер, на котором предлагается *хостинг*.

❷ Приложение (программа). Код, который обеспечивает другое запрашивающее приложение данными и методами.

❸ (*сервер автоматизации-automation server*) Приложение, которое предоставляет некоторую повторно используемую функциональность в рамках модели и технологии COM (также часто называемым сервером COM). Сервер автоматизации может не быть "чистым" сервером автоматизации, так же как и клиент автоматизации может не быть "чистым" клиентом автоматизации. В действительности сервер автоматизации может использовать сервисы другого приложения, которое также является сервером автоматизации. Клиент автоматизации, предоставляющий свои сервисы другому клиенту, также может являться как клиентом, так и сервером автоматизации. Глубинные механизмы (сетевые и транспортные протоколы), с помощью которых клиент автоматизации взаимодействует с сервером, уже являются частью собственно COM. Сервер автоматизации - это просто двоичный исполняемый модуль, который может состоять из нескольких объектов автоматизации. Объект автоматизации (также называемый объектом COM, хотя технически объект автоматизации является объектом COM особого сорта) - это отдельный, самодостаточный объект, спроектированный для выполнения специфической задачи или функции. Пример сервера автоматизации – приложение MS Excel.

Сервер баз данных (БД)

Обычно под сервером БД подразумевается СУБД, запущенная на той же машине, где находятся файлы БД, и монопольно распоряжающаяся этими файлами. При этом, все пользовательские приложения должны работать с базой только через эту СУБД, используя ее язык запросов.

Сервер приложений (Application server)

Сервером приложений называется выделенный компьютер (узел сети), который используется для запуска приложений, необходимых пользователям отдельных рабочих станций. При традиционном подходе на рабочих станциях работают клиентские приложения, которые интенсивно обмениваются данными и командами с файл-сервером. Использование сервера приложений позволяет снизить нагрузку на файл-сервер и, тем самым, увеличить его производительность. Сервер-ориентированные приложения состоят из двух компонентов: пользовательской части, которая работает на рабочей станции, и серверной части, которая работает на сервере. Управление работой приложения и ввод информации осуществляются при помощи пользовательской части, а реальная обработка и передача данных – при помощи серверной части. При этом сервер работает с исходными данными и возвращает рабочей станции только необходимые ей результаты.

Сервис (service)

❶ Компонент, способный выполнять задачу.

❷ WSDL сервис. Набор конечных точек.

❸ См. Web-сервис.

Сервлет (servlet)

Java программа, расширяющая функции Web-сервера, генерирующая динамический контент и взаимодействующая с Web-клиентом на основе парадигмы запрос/ответ. Сервлеты разрабатываются с помощью продукта Java Servlet Development Kit (JSDK) и выполняются в рамках серверов. Они способны обрабатывать сложные клиентские запросы и динамически генерировать ответы на них. Примером использования сервлетов может служить расширение, читающее запрос к базе данных на языке SQL, анализирующее его и делающее выборку данных из хранилища, а затем пересылающее клиенту HTML-страницу, сгенерированную автоматически на основе полученных данных.

Сервлет-контейнер (распределённый) (servlet container, distributed)

Контейнер сервлета, который может запускать Web-приложения, которое скомпоновано (связано) как распределённое и может выполняться в среде множества виртуальных машин Java (Java virtual machine), выполнямых на одном хосте или на разных хостах.

Сетевая плата

Компьютерная плата, установленная в каждую рабочую станцию для предоставления возможностей коммуникации с другими станциями и с серверами. Некоторые принтеры имеют собственные сетевые платы, позволяющие подключать их непосредственно к кабельной системе. Сетевые карты обычно представляют собой стандартные платы расширения, для персонального компьютера, выполненные по стандарту ISA или PCI.

Сетевое соединение

Процесс передачи данных между двумя компьютерами.

Сетевой протокол

Совокупность правил, регламентирующих передачу информации в сети.

Сетевые ресурсы

Ресурсами называются сетевые компоненты, поддающиеся учету и управлению, в частности, следующие:

- Сетевое оборудование – серверы, рабочие станции, кабели, повторители, узлы, концентраторы и сетевые интерфейсные платы.
- Другие устройства – жесткие диски, принтеры, модемы.
- Сетевое программное обеспечение – сетевые операционные системы, сетевые службы (коммуникации, очереди печати, обслуживание файлов) и т.п.
- Дополнительные программы – драйверы, протоколы, программное обеспечение мостов, маршрутизаторов, шлюзов, средства контроля и управления, приложения.

- Прочие объекты – процессы, средства защиты, структуры данных, пользователи, тома и т.д.

Сетевые технологии

Технологии, позволяющие компьютерам, программным компонентам и программно-аппаратным комплексам общаться совместно в сетевом режиме.

Сеть (Network).

❶ Этот термин может употребляться в широком смысле (сеть - это совокупность связанных между собой компьютеров) и в узком смысле (сеть - это совокупность компьютеров, соединенных между собой в соответствии с одной из стандартных типовых топологий - *шина, звезда, кольцо*, и использующих для передачи пакетов один из протоколов канального уровня, определенный для этой топологии). Каждая сеть имеет свой номер, который используется на сетевом уровне при выполнении маршрутизации. Когда две или более сетей организуют совместную транспортную службу, то такой режим взаимодействия обычно называют *межсетевым взаимодействием (internetworking)*. Для обозначения составной сети в англоязычной литературе часто также используются термины *интерсеть* (*internetwork* или *internet).* Интерсеть обеспечивает только передачу пакетов, не занимаясь их содержанием.

❷ Два или более компьютеров, связанных между собой и предназначенных для совместного использования данных и приложений.

Сеть Хранилища данных (Data Warehouse Network)

Интегрированная сеть Хранилищ данных, содержащая совместно используемые данные, переданные из исходного Хранилища данных на основе запроса потребителя информации. Управление хранилищами осуществляется с целью контроля избыточности данных и поддержки эффективного использования данных совместного доступа.

Сигнал

❶ Изменяющийся во времени физический процесс, отражающий передаваемое сообщение. В электрических цепях и схемах роль сообщений (*команд)* выполняют *импульсы.*

❷ Сигнал (импульс) – командный стимул в электронных устройствах и биологических системах.

❸ Спецификация асинхронной коммуникации между объектами. У сигналов могут быть параметры, выраженные в виде атрибутов. Представляет собой именованный классификатор, который служит для явной коммуникации между объектами. (UML).

❹ Знак, физический процесс или явление, несущие сообщение о каком-либо событии, состоянии объекта либо передающие команды управления, оповещения и т.д.

Синтез (от греческого synthesis - соединение)

Соединение (мысленное или реальное) различных элементов объекта в единое целое (систему). Синтез неразрывно связан с анализом (расчленением объекта на элементы).

Символ (symbol)

❶ Символ, обозначение.

❷ Символ, идентификатор.

❸ Символ, знак, литера при вводе с клавиатуры или в обозначениях элементов синтаксиса языков программирования.

Синергетика (Synergetics)

Наука, которая занимается изучением процессов самоорганизации и возникновения, поддержки, стойкости и распада структур (систем) разной природы на основе методов математической физики (“формальных технологий”). Синергетический подход также используется при изучении такой сложной и неструктурированной системы, как сетевое информационное пространство.

Синтаксис

❶ Способы соединения слов (и их форм) в словосочетаниях и предложениях, соединение предложений в сложные предложения.

❷ Система правил записи сообщений из символов или более простых конструкций на каком либо языке программирования.

Система (см. *создание систем*)

❶ Совокупность методов, процедур, программ или технических средств, объединенных определенными взаимоотношениями с целью выполнения заданных функций. (ГОСТ).

❷ Набор подсистем, организованных для достижений определённой цели и описываемых с помощью совокупности моделей.

❸ Упорядоченное множество структурно взаимосвязанных и функционально взаимодействующих однотипных элементов любой природы, объединённых в целостный объект, состав и границы которого определяются целями системного исследования. (Панкратова.)

❹ Набор переменных, выбранных наблюдателем. (Ashby, 1960).

❺ Набор или расположение сущностей так взаимосвязанных или так соединённых, что образуют единое или органическое целое.(Iberall).

❻ Любой, поддающийся определению набор компонентов. (Maturana and Varela, 1979).

❼ Совокупность взаимодействующих компонентов, работающих совместно для достижения определённых целей. Функциональные компоненты систем делятся на следующие типы: сенсорные, исполнительные, вычислительные, координирующие, коммуникационные и интерфейсные. Интегрированные свойства системы – это свойства, которые присущи системе как единому целому, а не отдельным её компонентам. К интегрированным системным свойствам относятся безотказность, удобство эксплуатации, безопасность и защищённость системы. (Соммервилл).

❽ Обычно операционная система или её компонент (например, файловая система). Если из контекста не ясно, о какой системе идёт речь, как правило, имеется в виду операционная система.

❾ Совокупность взаимосвязанных частей, объединённых по признаку описываемого процесса. (Дэвид Васкевич)

❿ (в геологии) Основное подразделение общей стратиграфической шкалы, отвечающее естественному этапу в развитии земной коры и органического мира. Соответствует геологическому периоду.

Система поддержки принятия решений, СППР (Decision Support Systems, DSS)

❶ Программное обеспечение, поддерживающее формирование отчетов по исключениям, стоп-сигналам, стандартным хранилищам, анализу данных и анализу, основанному на системе правил.

❷ База данных, созданная для формирования незапланированных запросов конечным пользователем.

Системная дискета (startup disk)

Гибкий диск, сформированный таким образом, что с него можно загрузить операционную систему Windows.

Системное мышление

Высшая форма человеческого познания, в которой процессы отображения, анализа и исследования объективной реальности с позиции достижения поставленных целей, базируются на умении из разрозненных, разнесённых в пространственно-временной среде материальных объектов, ситуаций, событий и процессов формировать целостное представление объекта исследования, а также на умении в условиях концептуальной неопределённости формализовать и решать задачу его системного исследования на основе системного использования возможностей математического и методологического инструментария, знаний, опыта, интеллекта, интуиции и предвидения исследователя. (Панкратова.)

Системный администратор (system administration, сокр. "admin", "sysadmin", "site admin")

Распространенная профессия (в развитых странах мира). Включает постоянное проведение мониторинга в компьютерных и сетевых системах с целью поддержания работоспособных и безопасных их конфигураций. Также включает управление распределением и размещением пользовательских имён и паролей, управление и мониторирование дисковым пространством и другими ресурсами совокупных компьютерных сетевых образований (интранет, экстранет, Internet и др.). Системный интегратор обязан обеспечивать целостность и безопасность всех данных организации, сохранять их резервные копии (бэк-ап), а также устанавливать новые компоненты программного обеспечения и аппаратных средств.

Системный анализ (см. *прикладной системный анализ*)

Системный программист (См. *программист системный*)

Системотехника

Комплексная технология создания систем, которая требует привлечения многих инженерных дисциплин.

Ситуация

Совокупность (сочетание) условий и обстоятельств, создающих определённую обстановку, положение.

Склад данных (Data Store)

Место хранения данных, неподвижные данные. Родовой термин, включающий в себя понятия базы данных и плоских файлов.

Скриншот (screenshot)

Сохранённое на диске графическое представление экранного изображения момента работы компьютера.

Скрипт (script) (См. *Scripting*)

❶ Программа, включенная в состав HTML-кода Web-страницы для расширения ее возможностей.

❷ Фрагмент программы или программа, написанная на скриптовом языке. Обычно небольшая программа-сценарий, написанная на каком-либо скриптовом языке программирования. Различают скрипты клиентской и серверной стороны. Скрипты клиентской стороны обычно пишутся на JavaScript для улучшения интерактивности Web-странички. Серверные сценарии используются для динамической подготовки информации на сервере и используют множество технологий (ASP, PHP, CGS, XML и др.).

❸ Программа или набор инструкций, которые выполняются не процессором компьютера, а некоторой другой программой контейнером. Код интерпретируется на этапе выполнения (at run time), а не запоминается в виде двоичного кода в исполняемом файле с расширением .exe. К контейнерам такого типа обычно относят интерпретаторы, браузеры или системы со встроенными языками, к которым относятся приложения MS Excel или ESRI ArcGIS с встроенным языком Visual Basic for Application (VBA).

Скриптовый язык (язык скриптов, ЯС) (scripting language – язык сценариев, glue language – склеивающий язык, system integration language – язык интеграции систем)

Язык, имеющий в своём составе много команд, представляющих собой мини-программы, предназначенные, как правило, для комбинирования уже существующих компонентов. Из более чем 30-и наиболее популярных ЯС можно особо отметить языки, Rexx, Tcl, Visual Basic, Python, VBScript и оболочка Unix (shell).

Словарь

❶ (в вычислительной технике) Таблица, содержащая такие атрибуты элементов данных, как их *имена*, *типы*, *размерности*, *форматы*, *диапазоны* изменения значений, допустимые *способы* использования.

❷ Лексика, словарный состав языка или диалекта какой-либо социальной группы, отдельного писателя и т.д.

❸ Система слов, расположенных по определённому принципу, по характеру содержащейся в словаре информации. Различают переводные, толковые, энциклопедические, частные словари, тезаурусы. В последнее время существует большое количество электронных словарей и энциклопедий по различным вопросам и тематике.

❹ Справочная книга, содержащая собрание слов, словосочетаний, идиом и т.п., дающая сведения об их значениях, употреблении, переводе на другой язык и др. Существуют лингвистические и энциклопедические словари.

Словарь данных (Data Dictionary)

База данных, содержащая информацию о самих данных и структуре баз данных. Каталог всех элементов данных, содержащий их имена, структуру и информацию по их использованию. Центральное месторасположение метаданных. Обычно словари данных разрабатываются для хранения ограниченного набора имеющихся метаданных, сфокусированных на информации по элементам данных, базам данных, файлам и программам установленных систем.

Слово (word)

❶ Вектор битов, рассматриваемый аппаратной частью компьютера как единое целое. Число битов в слове, называется длиной ли размером слова. Распространённая длина слова равна двум байтам. (Иллингуорт)

❷ Строка знаков или строка битов, рассматриваемая как единое целое. (ГОСТ).

❸ Законченная последовательность знаков определённой длины, воспринимаемая как элемент обработки с определённым семантическим содержанием.

❹ Единица речи, представляющая собой звуковое выражение отдельного предмета мысли. К примеру, произнести слово, написать слово.

❺ Название понятия, в отличие от самого понятия. К примеру, спорить о словах какого либо учёного.

❻ Текст к вокальному произведению. К примеру, музыка Чайковского, слова А. К. Толстого.

❼ Ораторское выступление, речь на собрании. Прошу слова! Дать, предоставить кому-нибудь слово.

Слот (slot)

❶ Гнездо (в т.ч. и для платы), вход, розетка;

❷ Разъем для элементов памяти. Как правило, это название используется для разъемов, куда «вставляются» платы расширения, в том числе модули типа SIMM и DIMM. Разъемы, куда «втыкаются» ножки (чипов либо разъемов «противоположного пола»), называются сокет (socket).*(*Другие значения см. *коннектор)*.

❸ Минимальная составляющая фрейма, заполняемая конкретной информацией об объекте или свойстве. (ГОСТ)

Смарт-карта (smart card)

Пластиковая карточка, содержащая интегральную схему, которая обеспечивает достаточный уровень программируемости и нбольшой объём памяти. Смарт-карти используются для идентификации, а также для кодирования таких сведений, как, к примеру, история болезни и т.д.

Снимок (снэпшот) (snapshot)

Конкретная статическая конфигурация системы в данный момент времени. Снимок включает в себя объекты и другие экземпляры, значения и связи. (UML)

Событие (event)

❶ Действие, на которое реагирует программа (например, щелчок левой или правой клавиш, а также перемещение мыши, нажатие клавиши или сочетаний клавиш клавиатуры).

❷ Любое значительное происшествие в компьютерной программе, системе или компьютерной сети, о котором следует сообщить системному администратору, пользователю или записать в журнал.

Совокупность

❶ (математическое) Объединяющая количественная характеристика. Например, совокупность значений аргумента.

❷ Сочетание, соединение, представляющее общую сумму чего-нибудь. К примеру, совокупность условий, совокупность данных.

Создавать

❶ Созидать, делать, творить или производить, вызывать из небытия в бытие.

❷ Изобретать, сочинять, составлять мысленно, или на деле, воздвигать, строить. Вновь создают только талантливые художники, а прочие подражают им. Созидание храма, дворца, столицы, дорог.

❸ Рождаться, являться, получать бытие. Созидание длительное, создание – окончательное действие по глаголу. По смыслу глагола создают мгновенно, а созидают исподволь.

Создание систем (процесс создания систем)

Последовательность действий, включающая следующие этапы: составление спецификации, проектирование, разработку, интеграцию (сборку) и тестирование. Наиболее ответственным этапом является сборка системы, когда различные подсистемы, подчас от разных производителей, интегрируются в единую систему.

Сокет (socket)

❶ Гнездо; (соединительная) панель; розетка (гнездовая часть разъемного соединения).

❷ Объект, являющийся конечным элементом соединения, обеспечивающего взаимодействие между процессами транспортного уровня сети.

❸ Двунаправленный канал (канал двухсторонней связи между компьютерами, объединенными в сеть). (Другие значения см. *коннектор)*.

❹ Конечный пункт передачи. То есть, когда программы используют сокет, он для них является абстракцией, представляющей одно из окончаний сетевого соединения. Для установки соединения в абстрактной модели сокетов необходимо, чтобы каждая из сетевых программ имела свой собственный сокет. Связь между двумя сокетами может быть ориентирована на соединение, а может быть и нет. Сетевая модель сокетов использует цикл: открыть - считать - записать - закрыть. Для открытия сокета нужно указать его описание и запросить у операционной системы дескриптор. Одновременно может быть открыто несколько сокетов.

Сокрытие информации (не существенной для пользователя) (information hiding)

❶ В программировании, процесс сокрытия деталей объектов и функций. Сокрытие информации является мощным методом и технологией программирования, поскольку таким образом снижается сложность. Одним из ведущих механизмов для сокрытия информации является инкапсуляция – комбинированные элементы для создания более крупных сущностей. В случае применения этой методики программист может сосредоточиться на новых объектах, не волнуясь и не заботясь о скрытых к этому времени прочих деталях. В этом контексте, полная иерархия языков программирования от машинного языка до языков высокого уровня может рассматриваться как форма сокрытия информации. Сокрытие информации используется также для предохранения программистов в случае необходимых в работе изменения программ или их частей.

❷ Синоним термина цифровое выполнение водяных знаков (*digital watermarking*). (Webopedia).

Сообщение (message)

❶ Основной блок связи (общения) между Web сервисом (Web service) и инициатором запроса (requester): данные передаются от или к Web сервису в виде отдельной логической пересылки. (W3C).

❶ Специальный символ, идентификатор или ключевое слово с или без параметров, которое представляет выполняемое объектом действие. (UML).

Сопровождаемость (maintainability)

Способность к поддержанию приложения в работоспособном состоянии.

Составной документ–(compound document)

❶ Документ, содержащий информацию, созданную более чем одним приложением.

❷ Документ, содержащий многочисленные типы информации, такие как текст, графические изображения, видео- и звуковые фрагменты и т.д.

❸ Организованная коллекция пользовательских интерфейсов, которые формируют единую интегрированную познавательную среду. Содержит разные данные и объекты, которые могут взаимодействовать между собой и пользователем.

Состояние

❶ Состояние системы в данный момент представляет собой набор численных значений, которые находятся в соответствующих переменных. (Ashby, l960).

❷ Любые хорошо определяемые режимы, ситуации или свойства, которые могут быть распознаны при неоднократных повторных наблюдениях.

Сотовая связь

Методика организации беспроводной связи. Район, в пределах которого необходимо обеспечить связь, делится на небольшие области обслуживания, называемые «сотами», и сеанс связи между двумя абонентами может осуществляться через промежуточные соты. Каждая сота обслуживается индивидуальным пунктом обслуживания, куда входит антенна и приемопередатчик, обеспечивающий прием сигналов с других сот и отдельных абонентов, а также передачу этих сигналов соседним сотам и абонентам, находящихся внутри соты. Диаметр соты, обычно, составляет от нескольких до 32 км.

Сотовая сеть

Сотовая сеть представляет собой пример организации беспроводной компьютерной сети. Для передачи данных в таких сетях используются радиосигналы частотами в диапазонах от 825 до 890 МГц, а так же специальные базовые станции (соты), которые обеспечивают трансляцию сигнала от передатчика к приемнику в пределах зоны обслуживания.

Преимущества сотовых сетей:

- Узлы могут быть мобильными, то есть перемещаться с места на место.
- Используемые диапазоны частот обеспечивают относительно высокую пропускную способность.
- Многие территории, в особенности крупные города, располагают уже существующей инфраструктурой базовых сот.

Недостатки сотовых сетей:

- Приемопередатчики и узлы должны находиться в переделах прямой видимости друг от друга.
- Передача данных по открытому воздушному каналу увеличивает риск перехвата и загрязнения различными помехами.
- Стоимость компонентов и услуг достаточно высока.
- Передача сигналов от соты к соте может привести к возникновению существенных задержек.

Спецификация

❶ Совокупность действий по специфицированию чего-либо.

❷ (программы) Полное описание требований по составлению исходной программы для данного компьютера с учётом ограничений на используемые средства и представление

данных, идентификаторов программы, связей с другими программными модулями и др. (ГОСТ).

❸ (спецификация программы–program specification) Точное описание того результата, который необходимо достичь с помощью программы. Это описание должно точно устанавливать, что должна делать программа, не указывая, как она должна это делать.

❹ Текстовое объявление синтаксиса и семантики некоторого строительного блока; декларативное описание того, чем является или для чего служит некоторая сущность. (UML).

❺ Разработка, заключающаяся в перечислении деталей, частностей.

Спулинг (Spooling)

Способ повышения производительности компьютера, в котором вывод программы помещается в быструю память (обычно на диск), а затем печатается параллельно с другими операциями. Термин происходит от сокращения выражения "**S**imultaneous **P**eripheral **O**perations **O**n-**L**ine" – одновременные периферийные операции.

Среда

❶ Вещество, заполняющее пространство, окружающие тела или явления; сфера. К примеру, воздушная среда, упругая среда.

❷ Совокупность природных или социальных условий, в которых протекает развитие и деятельность человеческого общества. К примеру, географическая среда, природная среда, экологическая среда.

❸ Социально-бытовая обстановка, в которой живет человек, окружающие условия; совокупность людей, связанных общностью условий, обстановки. К примеру, в среде ученых, в среде студентов. Литературная среда. Окружающая среда.

Стек (магазин, стековый список)

❶ Способ организации памяти, реализуемый программно или аппаратно в виде массива или списка элементов. Добавление или извлечение элементов возможно только с одного конца последовательности, поэтому данные, добавляемые к последовательности последними, извлекаются из неё первыми.

❷ Линейный список, все записи в котором выбираются, вставляются и удаляются с одного конца, называемого вершиной списка (стека). Это подразумевает обеспечение доступа к записям по принципу «последним вошёл, первым вышел» (last in, first out – LIFO). То есть, последний вставленный в список элемент первым удаляется из списка.

Стек протоколов

В сетевых технологиях таким образом называется совокупность связанных протоколов, используемых в той или иной компьютерной сети. Протоколы, входящие в один комплект, полностью или частично обеспечивают функциональность, соответствующую всем уровням используемой коммуникационной или сетевой модели.

Страница

❶ (в вычислительной технике) Условная единица деления непрерывного пространства памяти на поля фиксированной длины.

❷ Одна сторона листа бумаги в книге, тетради. К примеру, в книге около ста страниц, перелистывать страницы.

❸ (в переносном смысле) Период, часть (целого, образно представляемого в виде книги; книжн.). К примеру, новая страница моей жизни. Славные страницы его биографии. Новые страницы в исследовании Арктики.

Страница виртуальной памяти

Единица фиксированного объёма виртуальной памяти, используемая при перемещении данных виртуальной памяти в реальную память и обратно.

Стриммер (streamer)

Запоминающее устройство (накопитель) на магнитной ленте. По принципу действия аналогичен кассетному магнитофону, но записывает данные не в аналоговой, а в цифровой форме. Наибольшее распространение получили накопители для кассет (картриджей)

стандартов QIC. Применяются для периодической записи критически важных данных с целью их последующего восстановления (back-up) в случае непредвиденных сбоев в системе.

Строка (string)

❶ Структура данных, элементы которой линейно упорядочены.

❷ Тип данных в языках программирования, переменные и константы которого могут содержать последовательности произвольных символов. К примеру, переменная типа String в языке Visual Basic for Application (v.2002) может содержать до миллиарда символов.

Строка меню (menu bar)

Наиболее распространённый тип меню. Отображается вверху окна под его заголовком в виде набора пунктов меню, выбор каждого из которых приводит к появлению ниспадающего меню со списком команд. К примеру, в приложениях MS Office таковым является главное меню, содержащее команды: Файл, Правка, Вид, Вставка, Формат, Сервис, Таблица, Окно, Справка.

Строка текста

Последовательность символов исходного файла, завершающаяся символом "перевод строки".

Структура (structure)

❶ Определённая взаимосвязь, взаиморасположение составных частей целого; строение, устройство.

❷ Фактические связи (отношения), зафиксированные между компонентами, интегрально представляющими конкретную систему в заданном пространстве. (Maturana and Varela, 1979)

Структура данных (или информации) (data structure, information structure)

❶ Способ объединения нескольких элементов данных в один (массив, файл, список).

❷ Множество элементов данных, упорядоченных одним из принятых способов, например вектор, список и т.д.

❸ Агрегат данных, представляемый корневым ориентированным деревом.

❹ Представление пользователя о данных, выраженное в терминах их логического взаимодействия. С другой стороны, это аспект типа данных (data type), выражающий природу величин, которые являются составными, т.е. отличными от атомарных (atom).

Такие величины состоят из элементов (которые сами по себе не обязательно являются атомами), и структура данных выражает, как из этих элементов может быть составлена некоторая величина. Либо как составную величину разделить на элементы. Таким образом, например, структура данных "дата календарная" – это набор, содержащий член для каждого возможного календарного дня совместно с операциями для составления даты из ее элементов (года, месяца и числа), а также и выбора желаемых элементов.

Реализация структуры данных включает, как выбор определенной структуры хранения (storage structure), так и обеспечение набора процедур/функций, которые реализуют соответствующие операции с использованием выбранной структуры хранения.

Формально структура данных определяется как некоторая хорошо обозначенная область в абстрактном типе данных (abstract data type), которым (абстрактным типом) задается эта структура. Решение на компьютере задач реального мира включает определение некоторой идеальной структуры данных и ее последующее отображение на имеющиеся структуры данных [например массивы (array), записи (record), списки (list), очереди (queue), деревья (tree) и т.д.], в результате чего достигается ее (идеальной структуры) реализация.

Важно, что термин «структура данных», используется как для обозначения самой структуры, так и собственно данных, имеющих эту структуру.

Структура программы

Совокупность функционально обособленных частей программы и связей между ними.

Структура хранения

Представление данных на физических носителях.

Структурное программирование

❶ Программирование, технология которого определяет работу программиста как суперпозицию допустимых структур. Любой алгоритм на любом уровне программирования должен быть записан только с помощью трёх допустимых структур: линейной, структуры выбора и циклической.

❷ Методология программирования, основанная на предположении, что логичность и понятность програмы обеспечивает надёжность, облегчает модификацию и ускоряет разработку. Характерными чертами структурного программирования является отказ от неструктурных передач управления, ограниченное использование глобальных переменных и повсеместное применение модульности.

Структурный взгляд

Взгляд на полную модель с целью определения структуры объектов в системе, включая их типы, классы, отношения, атрибуты и операции. Структурный взгляд также относится к взгляду на представляемую модель, с целью фокусирования внимания на структуру частных случаев (вариантов) системы, которая моделируется средствами UML.

Стек

Способ организации памяти, реализуемый программно или аппаратно в виде массива или списка элементов. Добавление и извлечение элементов возможно только с одного конца последовательности, поэтому данные, добавляемые к последовательности последними, извлекаются из неё первыми.

СУБД (Система Управления Базами Данных, DBMS — DataBase Management System).

Программа, либо комплекс программ, предназначенных для полнофункциональной работы с данными. Как правило, включает в себя инструменты для создания и изменения структуры хранения наборов данных, а также средства доступа к хранимым данным, с возможностью их чтения, добавления, изменения и удаления. При этом, у большинства СУБД имеется собственный встроенный язык (возможно не один) для работы с данными. Сама база данных (БД) обычно находится просто в файлах закрытого, либо открытого формата.

Сумматор

❶ Накапливающий регистр.

❷ Регистр процессора, в котором остаётся результат выполнения команды (сложения, сравнения и т.д.). Таким образом, перед выполнением действия он содержит один из операндов, а затем, после выполнения операции, содержит результат произведённого действия.

❸ Операционный узел компьютера, выполняющий операцию арифметического сложения двух чисел. Для сложения двух многоразрядных двоичных чисел на каждый разряд необходим один полный сумматор. Только в младшем разряде можно обойтись полусумматором.

❹ Устройство, осуществляющее арифметическое суммирование двух двоичных ***n***-разрядных чисел X=(x(n-1),..,x0) и Y=(y(n-1),..,y0) по следующим правилам сложения двух одноразрядных двоичных чисел:

0 (+) 0 = 0

0 (+) 1 = 1 (+) 0 = 1

1 (+) 1 = 0 и перенос 1 в старший разряд.

Суперкомпьютер

Компьютер с множеством процессоров, созданный для достижения высоких скоростей вычисления. Используется для суперсложных вычислений, таких как астрономические расчёты, метеорологические прогнозы, моделирование глобальных военных операций и т.д.

Суперконвеерный

Архитектура процессора, в которой результаты разных этапов вычислений максимально совмещены по времени выполнения на уровне схемных решений.

Суперскалярный

Обычно процессор с высокой степенью распараллеливания процессов вычислений, что приводит к увеличению производительности компьютера.

Сущность, сущности (entity, entities)

❶ Абстракции, являющиеся основными элементами модели.(UML).

❷ Информационное проявление объекта в данном контексте. К примеру, музыкальный центр при наличии компакт-диска (контекст) проявляет себя как CD-плеер (одна из его сущностей), при помещении в него кассеты с магнитной лентой он проявляет себя как кассетный магнитофон, а при отсутствии носителей информации проявляет себя как радиоприёмник (третья сущность).

❸ Применительно к реляционным базам данных, таблицы, являющиеся средствами хранения данных об объектах, моделируемых в базах данных. Состоят из строк и столбцов.

❹ Коллекции объектов (субъектов, местоположений, предметов), описываемых одинаковыми атрибутами. Сущности идентифицируются в процессе концептуального проектирования баз данных и разработки приложений. (ESRI)

Схема (Schema)

Логическое и физическое определение элементов данных, физических характеристик и внутренних отношений.

Схема базы данных (Database Schema)

Логическое и физическое определение структуры базы данных.

Сценарий (scenario)

❶ Система фреймов, описывающая определённую ситуацию. (ГОСТ).

❷ Последовательность взаимодействий между субъектом и объектом. Сценарий позволяет определить текущее состояние процесса и цели его развития.

Сэмплинг (Sampling)

В контексте музыки относится к процессу выделения из музыкальной записи определённого шаблона (sample) и повторного его использования в качестве инструмента внутри другой записи. Этот процесс осуществляется с помощью Сэмплера (Sampler), который может быть или электронным устройством (hardware) или компьютерной программой.

- T -

Таймер

Системные часы. Устройство (регистр процессора) для измерения или индикации текущего времени или интервала.

Такт

Длительность одной операции синхронного (управляемого таймером) устройства. Обычно подразумевается процессор компьютера.

Тактовая частота (процессора)

Величина, обратная длительности такта, измеряется в мегагерцах, гигагерцах и т.д.

Тезаурус (thesaurus) (от греческого «θβσανρος» (*thesauros)* **- сокровище),**

❶ Знания приемника информации о внешнем мире, влияющие на его способность воспринимать те или иные новые сообщения. (см. системы распознавания образов).

❷ Словарь, в котором максимально полно представлены слова языка с примерами их употребления в тексте (в полном объеме осуществим лишь для мертвых языков).

❸ Словарь, в котором слова, относящиеся к какой-либо области знания, расположены по тематическому принципу и показаны семантические отношения (родо-видовые, синонимические и др.) между лексическими единицами. В информационно-поисковых тезаурусах лексические единицы текста заменяются дескрипторами.

❹ Совокупность терминов, связанных системой ссылок с соответствующими синонимами, а также их близкими смысловыми и ассоциативными значениями.

❺ Множество знаний, которое формируется в программах систем искусственного интеллекта (artifical intelligence).

❻ Совокупность знаний, накопленных человеком или некоторым коллективом.

❼ Предварительная осведомлённость в информационных и естественных системах.

Текстура

Растровый файл, обычное двумерное изображение, используемое в качестве "обоев" - иллюстрирующих фактуру поверхности виртуальных тел. Другими словами, текстура является разновидностью детализации, называемой иногда детализацией цветом и заключающаяся в нанесении некоторого узора на поверхность. Различают два основных типа текстур: точечные (Bitmap) и процедурные. В отличие от растровых файлов, процедурные или аналитические текстуры представляют собой запрограммированные математические методы генерирования изображения текстуры, которые основываются на формулах, позволяющих получать текстуры типа мрамора или иных материалов, имеющих разный узор на разных поверхностях. Такие текстуры можно произвольно масштабировать и совершать над ними любые аффинные преобразования.

Теория (от греч. *theoria* **- рассмотрение - исследование)**

Система основных идей в той или иной отрасли знания; форма научного знания, дающая целостное представление о закономерностях и существенных связях действительности.

Теория познания (гносеология, эпистемология)

Раздел философии, в котором изучаются закономерности и возможности познания, отношения знания к объективной реальности, исследуются ступени и формы процесса познания, условия и критерии его достоверности и истинности. Обобщая методы и приемы, используемые современной наукой (эксперимент, моделирование, анализ, синтез и т.д.), теория познания выступает в качестве ее философско-методологической основы.

Термин (от латинского terminus - граница, предел)

Слово или сочетание слов, обозначающее специальное понятие, употребляемое в науке, технике, искусстве. В современной логике слово "термин" часто употребляется как общее имя "существительных" языка логико-математических исчислений (т. н. термов), выражающих при интерпретации элементы предметной области.

Терминология (от *термин* и *...логия*)

Совокупность, система терминов какой-либо науки, области техники, вида искусства, предметной области и т. п.

Террейн (terrain)

❶ Термин употребляемый для описания слоя суши, почвы, Земли.

❷ Местность; территория.

❸ Почва; грунт.

❹ Террейн является элементарно соединяемой (т.е. отдельными разъединёнными участками поверхности). Таким образом, визуализация террейна есть особый случай визуализации объёма, иногда описываемого как 2,5D.

Тестирование программы

Деятельность, направленная на поиск ошибок в программном средстве, допущенных при его проектировании и разработке.

Технология

❶ (от греч. têchnë–искусство, мастерство + …логия) Наука или совокупность сведений о различных способах обработки или переработки сырья, полуфабрикатов, изделий, а также сами процессы этой обработки. (Энц.)

❷ Описание способов в виде инструкций, графиков, чертежей и пр.

❸ Строго научное понятие, означающее комплекс научных и инженерных знаний, воплощённых в способах, приёмах труда, наборах материально-вещественных факторов производства и способах их соединения для создания какого-либо продукта.

❹ Объект или последовательность операций, созданных человеком для достижения намеченных целей.

❺ Совокупность методов обработки, изготовления, изменения состояния, свойств, формы сырья, материалов или полуфабриката в процессе производства

❻ Наука о способах воздействия на сырье, материалы или полуфабрикаты соответствующими орудиями производства.

Технологии создания распределённых приложений

К технологиям создания распределённых приложений относятся следующие: CORBA, EJB, WebServices, COM/DCOM, ActiveX, .NET.

Тип данных

❶ Характеристика класса порций данных, выражающая общие для этих порций представление и способ использования.

❷ Определяет множество допустимых (возможных) значений, которые может иметь тот или иной объект, а также множество допустимых операций, которые могут применяться к нему. Кроме того, тип определяет также и формат внутреннего представления данных в памяти компьютера, т.е. длину в байтах.

Тип файла (file type)

Совокупность файлов, к которым применим единый набор действий, доступных из контекстного меню операционной системы либо которые открываются по команде *Открыть (Open)* из главного меню любого приложения. Тип файлов определяется по расширениям их имён. К одному типу может относиться как одно, так и несколько расширений. К примеру, приложение Word открывает файлы с расширениями .DOC и .RTF. Вместе с тем, никакие другие приложения файлы с расширением .DOC не могут ни открывать, ни, тем более, работать с ними.

Товарный знак (торговая марка, эмблема)

Официально принятый термин, означающий зарегистрированное в определенном порядке оригинально оформленное художественное изображение (оригинальные названия, художественные композиции и рисунки в сочетании с буквами, цифрами, словами и без них и т.п.). Товарный знак является эмблемой товара - графическим оформлением его содержания.

Топография (от греч. tòpos – место и ...графия)

Научно-техническая дисциплина, занимающаяся географическим изучением местности путём создания топографических карт на основе съёмочных работ (наземных, с воздуха, из космоса). В сферу топографии входят вопросы классификации, содержания и точности топографических карт, методики их изготовления и обновления и получения по ним различной информации о местности.

Транзакция (transaction)

❶ Неделимый (атомарный) фрагмент работы, который модифицирует данные. Транзакция включает одно или более программных инструкций, которые могут быть выполнены или не выполнены. В последнем случае происходит откат системы в исходное

(до выполнения транзакции) состояние. Транзакции дают возможность множеству пользователей получать доступ к одним и тем же данным одновременно.

❷ Взаимодействие между клиентом и сервером. Например, транзакцией может являться последовательность операций: запрос, передача данных или разрыв соединения. Сеанс системы АТМ (automated teller machine – автоответчик) также является примером транзакции. При использовании SQL (Structure Query Language – язык структурированных запросов) транзакция является наименьшим завершенным выполняемым действием для поиска или модификации баз данных. Если в SQL какой-либо из шагов транзакции не может быть выполнен, то она отменяется полностью.

Транслятор (компилятор)

❶ Программа, транслирующая (переводящая) текст на одном языке программирования в текст на другом языке программирования.

❷ Программа или техническое средство, выполняющее преобразование программы, представленной на одном из языков программирования, в программу на другом языке и в определённом смысле равносильную первой (в общем случае производится перевод программы во внутренний язык машины).

Трансляция, компиляция, (compilation)

Преобразование программы из описания на входном языке (языке программирования) в ее представление на выходном языке (в машинных командах).

Трансцендентное число

Число, не удовлетворяющее никакому алгебраическому уравнению с целыми коэффициентами. Трансцендентными числами являются: число π = 3,14159...; десятичный логарифм любого целого числа, не изображаемого единицей с нулями; число *e*=2,71828... и др.

Трафик

Применительно к сетям, степень сетевой активности. Например, один из способов измерения трафика состоит в определении количества сообщений, которые передаются через сеть в данный момент либо на протяжении определенного временного интервала.

Трейдер

В международной практике трейдером называется лицо (предприятие), совершающее сделки от собственного лица. Лицо (предприятие), выполняющее чужие заказы на покупку-продажу, называется брокером. Целями индивидуального предпринимателя, которого называют в экономической литературе трейдером, является получение прибыли, выживание и развитие своего бизнеса, а также игра на бирже на котировке акций.

Трекинг (tracking)

Пропорциональное изменение пробелов между словами и, главное, между буквами текста. Термин относится к верстке. Одно из применений трекинга – регулировка оптических характеристик текста при изменении кегля, способствующая лучшей воспринимаемости и читаемости. Например, текст, набранный малым кеглем, для удобочитаемости требует наряду с "просветлением" контраста, еще и увеличения меж буквенных интервалов. Крупный же кегль, наоборот, смотрится красивее, если меж буквенные интервалы сократить, а гарнитуру "утяжелить".

Триггер

Последовательная электронная схема с ***двумя состояниями***, каждое из которых при определённых условиях на входах поддерживается постоянным (т.е. стабильным). Каждому из этих состояний ставится в соответствие некоторое логическое значение (к примеру, ""ИСТИНА″ или ″ЛОЖЬ″, ″0″ или ″1″), которое триггер и «хранит».

Троп

Прием речи, состоящий в таком замещении речения (слова или словосочетания) другим подобным, при котором замещающее речение, используясь в значении замещенного, обозначает последнее и сохраняет смысловую связь с ним. Выражения *"черствая душа",*

"линия понимания вещей", "море смеялось", "столица мгновенно прервала свои занятия" содержат тропы, т.е. включают замещение одного слова другим словом и потому используются в несобственном, переносном значении. (См. *метафора*).

Тэг (Tag)

Команда, вставляемая в документ с целью указания, как документ или часть документа должна быть форматирована для последующего представления. Тэги применяются во всех используемых формальных спецификациях форматирования, которые сохраняются вместе с документом в текстовом формате в файлах. Применяются обычно в языках SGML и HTML.

- У -

Узел (Host)) (см. хост).

Любой компьютер, подключенный к сети Internet. Узел, предоставляющий информацию, называется сервером. Узел, потребляющий информацию, называется клиентом.

Унаследованная система (legacy system) (См. *наследуемая система*)

Управление

❶ Процесс целенаправленного воздействия на систему, обеспечивающий повышение ее организованности с целью достижения того или иного полезного результата. Любая система управления разделяется на управляющую и управляемую подсистемы. Связь от управляющей подсистемы к управляемой называется прямой связью. Такая связь существует всегда. Противоположная по направлению связь называется обратной. Понятие обратной связи является фундаментальным в технике, природе и обществе.

❷ Функция организованных систем различной природы (биологических, социальных, технических), обеспечивающая сохранение их определенной структуры, поддержание режима деятельности, реализацию их программ и целей.

❸ Выбор входных сигналов к системе с целью создания определённого её состояния или выходных сигналов с той же целью, некоторым заданным образом.(Arbib).

Управление версиями (Versioning)

Возможность одним определением описывать информацию о множественных физических реализациях.

Управление знаниями (см. *Knowledge Management)*

Управление объектом

OMG определяет управление объектом как разработку программного обеспечения, которое моделирует реальный мир через представление "объектов". Эти объекты есть инкапсуляция атрибутов, отношений и методов программного обеспечения опознаваемые программными компонентами. Ключевое преимущество объектно-ориентированной системы - ее способность расширять свои функциональные возможностях расширяя существующие компоненты и добавляя новые объекты к системе. Объектное управление имеет результатом более быструю разработку приложений, более легкое обслуживание, огромную масштабируемость и многократное использование программного обеспечения (см. *OMA*).

Уровень

Абстрактное понятие в структуре представления сложных образований и систем. Существует девять значений понятия ***уровень***, являющегося основным для концепции ***иерархии.*** Термин «уровень» может означать:

1) степень вообще;
2) степень сложности;
3) степень глубины аналитического исследования;
4) возникновение организаций более высокого уровня по сравнению с имеющейся (концепция, используемая биологами и физиологами для выражения

идеи превращения целостных образований более низкого уровня в элементы целостных образований более высокого уровня);

5) систему взаимосвязанных свойств, или переменных (poistem);

6) разряд;

7) слой;

8) основной слой;

9) уровень.

Последнее определение равнозначно определению уровня как "градации упорядоченности, достигаемой не путем произвольного перебора, а путем реализации целенаправленной последовательности действий". Именно этот смысл лежит в основе понятия уровня организации.

Уровень абстракции

Способ систематического представления и описания объектов на определенном концептуальном уровне.

Уровень логический

Концептуальный или виртуальный уровень, т.е. включающий в себя концептуальные, а не реальные физические объекты и скрывающий принципы реализации (к примеру, логическое устройство диск, логическое имя устройства и др.)

Уровень физический

Определяет конкретную технологическую реализацию того или иного устройства или элемента системы.

Уровень языка программирования

При определении уровня языка программирования имеется в виду то, насколько язык программирования приближен к машинным кодам (командам для процессора) компьютера. Выстраивается иерархия вида:

Самый нижний уровень - собственно машинные коды

Средний уровень - Ассемблер

Высокий уровень -

- 3GL языки (C, C++, COBOL, Ada, Pascal)
- 4GL языки (Microsoft VisualBasic, PowerBuilder, Inprise Delphi Object Pascal, Oracle Developer PL/SQL)

Самый верхний уровень (моделирования, анализа, проектирования) - OMG UML.

Условно-бесплатная программа (Shareware) (см. также *программный продукт*)

Программа, которую вы можете бесплатно использовать в течение ограниченного времени. Если по истечении оговоренного срока вы продолжаете работать с программой, вам надо заплатить за нее.

Утилита

Любая специальная программа, предназначенная для выполнения определенного задания по обслуживанию операционной системы (например - Windows 2003).

- Ф -

Файл (file) (см. также *file*)

❶ Поименованная область на магнитном диске.

❷ Поименованная область памяти на каком–либо физическом носителе (FDD, HDD, CD-ROM и т.д.…). В файле может содержаться любая информация доступная для кодирования в двоичном виде. Файлы бывают: защищенные, скрытые, системные, архивные, каталог, обычный и др. Каталог имеет имя и может находится в другом каталоге, являясь при этом подкаталогом или подчиненным каталогом. Так образуется иерархическая древовидная файловая система. На каждом дисковом носителе существует корневой каталог, тот в котором регистрируются обычные файлы и подкаталоги 1-го уровня. В последних, в свою

очередь, регистрируются файлы и подкаталоги 2-го уровня и т.д. Полное имя файла образуется из двух слов имени и типа, разделенных точкой.

❸ Во многих системах, таких как UNIX и MS–DOS, файлом называется не интерпретируемая последовательность байтов. Значение и структура информации в файле является заботой прикладных программ и операционную систему это не интересует.

Файл свопинга (swap file)

Файл, в котором хранится содержимое виртуальной памяти. В Windows 98 (к примеру), файл свопинга (подкачки) называется Win386.swp и обычно находится в основной папке системы. Обычно имеет большую длину

Файловая система

В операционных системах - это структура, используемая для хранения файлов и связанной с ними информации - названий, атрибутов и данных о расположении на диске. Ниже приведен ряд примеров популярных файловых систем:

- CDFS (CD-ROM File System, файловая система компакт-дисков) - используется для хранения данных и файлах и самих файлов на компакт-дисках
- FAT (File Allocation Table, таблица размещения файлов) - используется в различных версиях операционной системы DOS.
- HPFS (High Performance File System, высокопроизводительная файловая система) - используется в операционной системе OS/2.
- NTFS (NT File System, файловая система New-Technologies) - используется в операционных системах Windows NT и Windows NT Advanced Server.

HFS (Hierarchical File System, иерархическая файловая система) - используется в операционной системе Macintosh System 7.

Фактический параметр

Элемент языка, присутствующий в момент вызова процедуры, который поставлен в соответствие некоторому формальному параметру для обеспечения выполнения процедуры. (СТ ИСО 2382/15-85).

Физический уровень

Уровень взаимодействия открытых систем, обеспечивающий установление, поддержание и разъединение физического соединения между логическими объектами сетевого уровня звена данных и передачу битов данных между этими объектами. (ГОСТ 24402–88)

Физическое соединение

Соединение, обеспечивающее взаимодействие двух либо более объектов физического уровня.

Фильтры (Filters)

Сохраненные наборы параметров отбора, определяющих поднабор информации в Хранилище Данных.

Флопсы, Floating-point operations per second (FLOPS – количество операций с плавающей точкой в секунду)

Является общей оценкой (тестом), измеряющим скорость работы *микропроцессора*. Операции с плавающей точкой включают любые операции, которые включают дробные числа. При вычислениях с дробными числами требуется значительно больше промежуточных операций, чем для вычислений с простыми или целыми числами. Проверить скорость их выполнения можно только на базе выполнения некоторых специально разработанных программ (приложений) или тестов. Большинство современных микропроцессоров включают блок операций с плавающей точкой (floating point unit (FPU)), который выполняет все такие операции. В современных микропроцессорах быстродействие измеряется в производных единицах : мегафлопах(megaflops – 10^6 оп./с), гигафлопс (gigaflops – 10^9 оп./с), терафлопс (teraflops 10^{12} оп./с) и петафлопс (petaflops – 10^{15} оп./с).

Флэш (flash)

Разновидность энергонезависимой памяти с низким (сопоставимым с DRAM) временем доступа по чтению и относительно высоким временем записи. Используется для компактных внешних запоминающих устройств, а также для хранения редко перезаписываемых программных компонент (например, BIOS или операционной системы некоторых узкофункциональных устройств). Существует, в частности, в виде форм-фактора SIMM.

Форма (лат. forma)

❶ Внешнее очертание, наружный вид, контуры предмета.

❷ Внешнее выражение какого-либо содержания.

❸ Установленный образец чего-либо (напр., написать отчет по форме).

❹ Приспособление для придания чему-либо определенных очертаний (напр., литейная форма).

❺ Одинаковая по цвету и покрою одежда (напр., форма военнослужащих).

❻ Совокупность приемов и изобразительных средств художественного произведения (напр., стихотворная форма).

Формальный параметр

Параметр, определяемый в заголовке процедуры и используемый в теле процедуры. Получает значение при активизации процедуры.

Формат (данных) (data format)

❶ Спецификация размещения данных в памяти, базе данных, на внешнем носителе, а также при вводе–выводе. (ГОСТ).

❷ Определённая структура информационного объекта, подвергаемого обработке, записываемого на носитель или выводимого в виде твёрдой копии. (Иллингуорт).

❸ Англ. слово *Format* используется как глагол, указывающий на такие действия, как запись информации в предписанной форме или разбиение запоминающей среды (диска) на сектора с целью приёма информации.

Форматирование

Программно управляемое нанесение на поверхность магнитных дисков участков стандартной длины (секторов) для последующей записи файлов.

Формула (от латин. formula, букв. уменьш. от forma) (Formula)

❶ (математическое) Выраженный условными знаками ряд математических величин в их функциональных зависимостях.

❷ Общее краткое и точное выражение (мысли, закона), то есть определение. К примеру, формула изобретения, формула закона тяготения.

❸ (химическое) Условное буквенное выражение состава сложных веществ и химических процессов. Химическая формула. Формула серной кислоты. Формула горения.

❹ Объект базы данных, представляющий собой вычисление, правило или другое выражение для операций с данными в многомерной базе данных. Формула определяет отношения между элементами измерения и используется разработчиками баз данных OLAP для обеспечения большего по количеству наполнения для сервера базы данных. Формула используется конечными пользователями для моделирования отношения внутри предприятия и для персонализации данных с целью обеспечения большей наглядности и точности отображения.

Формфактор (форм-фактор)

Механический конструктив компонентов компьютера, определяющий их физический интерфейс (формфактор на корпуса (tower, desktop и др.), микросхемы и т.п.).

Формы мышления (см. *Мышления формы*)

Фотограмметрия (от фото..., греч. gràmma –запись, изображение и ...метрия)

Научно-техническая дисциплина, занимающаяся определением размеров, формы и положения объектов по их изображениям на фотоснимках. Последние получают как

непосредственно кадровыми, щелевыми и панорамными фотоаппаратами, так и при помощи радиолокационных, телевизионных, инфракрасно-тепловых и лазерных систем.

Фракталы (Fractals)

Группа форм, которые не идентичны друг другу, но похожи общим узором, например снежинки или листья на дереве. Фракталы можно создавать программно, и они применяются дизайнерами и иллюстраторами для получения интересных и красивых изображений.

Фрейм (кадр) (Frame)

❶ Блок данных фиксированного формата, передаваемый по каналу связи и содержащий управляющую информацию, например адреса и контрольные байты. Обычно сеть рассчитана на несколько типов кадров со стандартными форматами. Термины «кадр» и «пакет» все чаще употребляются как синонимы.

❷ Отдельный кадр движущихся на экране изображений.

❸ Временной интервал от стартового бита до последнего стопового бита асинхронной, последовательной передачи.

Фундаментальный

❶ Прочный, крепкий, большой, значительный.

❷ Основательный, солидный. К примеру, фундаментальные рассуждения.

❷ Нечто, составляющее основную часть или основу чего-либо. Обычно применяется в качестве характеристики основополагающих элементов наук или прикладных отраслей знаний. К примеру, фундаментальные исследования, фундаментальные понятия.

Функция (function) (от латинского functio - исполнение, осуществление)

❶ Одно из основных понятий математики. Пусть заданы два множества X и Y и каждому элементу $x \in X$ поставлен в соответствие элемент $y \in Y$, который обозначен через $\boldsymbol{f(x)}$.

В этом случае говорят, что на множестве X задана функция $\boldsymbol{f}$ (а также – что переменная y есть функция переменной x, или что y зависит от x) и пишут $\boldsymbol{f : X \to Y}$.

То есть задано отображение, ставящее в соответствие *одному значению аргумента* ровно *одно значение отображения*. Функции могут быть заданы, например, формулой, графиком, таблицей, правилом и так далее. (Мат. энц.)

❷ Функция в математике - соответствие между переменными величинами, в силу которого каждому значению одной величины x (независимого переменного, аргумента) соответствует определенное значение другой величины y (зависимого переменного, функции).

❸ (в вычислительной технике) Процедура, возвращающая результат. В некоторых языках программирования, функция не должна иметь побочного эффекта.

❹ (в вычислительной технике) Величина, зависящая от других величин.

❺ Деятельность, обязанность, работа; внешнее проявление свойств какого-либо объекта в данной системе отношений (например, функция органов чувств, функция денег).

❻ Функция в социологии, т.е. роль, которую выполняет определенный социальный институт или процесс по отношению к целому (например, функция государства, семьи и т.д. в обществе).

Функциональная зависимость

Считается, что A функционально зависит от B, если в любой момент времени каждому значению B соответствует не более одного значения A.

Функциональный язык, функциональный язык программирования

Декларативный язык программирования, основанный на понятии функции, – описания зависимости результата от аргументов с помощью других функций и элементарных операций. Функции только задают зависимость и не определяют порядок вычислений. В функциональных языках нет понятия переменных и присваивания, поэтому значение функции зависит только от её аргументов и не зависит от порядка вычислений.

- X -

Хаб (hub)

Устройство, которое изменяет передаваемые сигналы таким образом, что протяженность сети может быть увеличена или обеспечивается возможность подключения дополнительных рабочих станций. Существует два типа хабов:

- *Активный хаб.* Усиливает передаваемые сигналы в топологии сети. Активные хабы используются для добавления рабочих станций и увеличения расстояния между станцией и сервером.
- *Пассивный хаб.* Используется в некоторых топологиях для разделения передаваемого сигнала для подключения дополнительных рабочих станций. Пассивный хаб не усиливает передаваемый сигнал, поэтому его необходимо подключать непосредственно к рабочей станции или к активному хабу.

Хакер (см. *hacker*)

Хеджирование

Хеджированием называется практика заключения на фьючерсной или опционной бирже срочных сделок на продажу или покупку валюты или ценных бумаг для страхования от предполагаемых в будущем колебаний цен или процентных ставок. Сущность хеджирования заключается в покупке или продаже фьючерсных или опционных контрактов одновременно.

Хеширование

Процесс определения местоположения файла на большом томе посредством вычисления адреса файла в кэше и на диске.

Хинт (Hint)

Совет, разъяснение функций, реализуемых приложением, представленных данным ярлыком или пиктограммой, или же командой соответствующего меню.

Холодная печать (Cold type)

Компьютерный набор текста.

Холодный запуск (Cold start)

Процесс запуска (boot) компьютера, который в результате серьезной ошибки не реагирует на нажатие клавиш Ctrl+Alt+Del. Выключение и повторное включение компьютера.

Хореография

Для Web Сервисов предполагает описание порядка следования сообщений с точки зрения одного узла или группы узлов.

Хост (Host)

–1) Любой компьютер, непосредственно подключённый в вычислительную сеть и принимающий участие в обеспечении других компьютеров Web-сервисами, такими, как (e-mail, Usenet newsgroups, FTP или World Wide Web).

–2) Центральный компьютер.

Хостинг

Принцип размещения и поддержки Web-страниц (ресурсов клиента) на сервере Web-провайдера. Вы можете разместить на оборудовании провайдера виртуальный www-сервер вида www.name.ru (*.com или *.zel.ru), где "name" - это идентификатор, предлагаемый Вами и не занятый в выбранном домене. При этом материалы становятся доступными для других пользователей всемирной сети. Провайдер берет на себя работу по регистрации доменного имени в организациях, администрирующих соответствующий домен.

Хранение данных (data holding)

Режим запоминающего устройства (ЗУ) после записи или регенерации данных, обеспечивающий возможность их последующего считывания в произвольный момент времени.

Хранилище данных (см. *Data Warehouse*)

Хранимые процедуры баз данных (stored database procedures)

Наборы оттранслированных операторов языка SQL и необязательных операторов передачи управления, хранимых в базе данных и выполняемых на сервере баз данных. Хранимые процедуры часто используются для определения бизнес правил.

- Ц -

Цветовая палитра

Набор цветов, используемых для воспроизведения *растровых* изображения. Наиболее употребительными являются пять типов цветовых палитр:

–черно-белая, или bitmap (битмап) - любой из единичных элементов имеет только либо черный, либо белый цвет;

–grayscale (оттенки серого) единичный элемент может иметь один из 256 оттенков серого цвета;

–8-битный цвет - из всей доступной человеческому глазу цветовой гаммы вибираются 256 цветов, которые и формируют изображение;

–16-битный цвет предоставляет набор из приблизительно 65000 цветовых оттенков,

–24-битный, true color или "истинный", цвет делает доступным 16 миллионов цветовых оттенков. Основным преимуществом растровых изображений является возможность передавать огромное количество оттенков цвета и плавных переходов между ними (например, в фотографии).

Целое число

Тип чисел в языках программирования. Буквально, ″число без точки″, т.е. не содержащее дробной части, которую необходимо отделять от целой части. К примеру: 15, -123 и т.д.

Целый

❶ От которого не убавлено. Например, целая машина зерна.

❷ Указывающий на величину. К примеру, "Я готов обнять целый мир!".

❸ О чём-то значительном. К примеру, целый ряд новых проблем.

❹ (математическое) Целое число, то есть, измеряемое в единицах целых элементов.

Центрально устройство управления

Блок процессора компьютера, содержащий схемы, управляющие расшифровкой и выполнением инструкций (команд).

Цифра (digit) (см. *цифры*)

Представление числа, занимающего одну позицию в соответствующей системе исчисления. В десятичной системе исчисления цифрами являются символы 0-9, в восьмеричной – 0-7, в двоичной – 0-1.

Цифры

Условные знаки для обозначения чисел. Могут отличаться в разных системах счисления. К примеру, значение ***одиночной цифры*** A в *шестнадцатеричной* системе счисления соответствует числу 10 в ***десятичной*** системе счисления, которое, в свою очередь, как видно, обозначается ***двумя цифрами.***

Цифровая линия (digital line)

Линия связи, передающая информацию только в двоичной (цифровой) форме. Для минимизации искажений и влияния помех вдоль цифровой линии периодически подключаются повторители, которые восстанавливают форму сигнала.

Цифровая подпись (digital signature)

Средство подтверждения авторства зашифрованного сообщения, файла или любой другой зашифрованной цифровой информации. Скрепление цифровой подписью подразумевает преобразование информации и некоторых конфиденциальных сведений, которыми обладает отправитель, в метку, называемую подписью. Цифровые подписи применяются в средах с открытыми ключами и предоставляют функции обеспечения целостности и предотвращения неавторизованного изменения передаваемой информации.

Цифровая среда (Digital media)

Термин "цифровая среда" относится к цифровым представлениям аудио и видео данных в World Wide Web и другим технологиям, которые могут использоваться для создания и распространения цифрового контента.

Цифровой (digital)

❶ Численное значение.

❷ В наиболее общем смысле, понятие «цифровой» относится к форме представления, в которой отдельные объекты (или цифры) используются для выражения или представления объектов «реального мира» (к примеру, времени или температуры).

❸ Свойство или способность обрабатывать дискретные значения в противоположность значениям непрерывного спектра.

❹ Термин "цифровой" описывает электронную технологию, с помощью которой генерируются, хранятся и обрабатываются данные в терминах двух состояний: положительного и отрицательного. Положительное состояние выражается и представляется цифрой 1, а отрицательное цифрой 0. Таким образом, данные передаются и сохраняются в виде строк нулей и единиц. Каждое из этих состояний или цифр представляется в виде *бита*, а строка битов в компьютере может адресоваться по отдельности в виде группы битов, называемых *байтом*.

Цифровой сертификат (Digital certificate)

Документ, подписанный с помощью цифровой сигнатуры, который устанавливает, что заданный открытый ключ соответствует объекту, имеющему определенное имя. В рамках OBI сертификаты подписываются или исходят от доверенной третьей стороны (Certificate Authority) к покупателям, серверам и авторизованным сторонам, подписавшим документы заказа.

Цифровой сигнал

Сигнал, например голос, представленный последовательностью дискретных уровней (например, 0 и 1) и мало подающийся искажению при его передаче. Цифровой сигнал более устойчив к помехам и его использование позволяет повысить качество связи.

Цифровые продукты (Digital Products)

Неосязаемые продукты, которые могут передаваться по цифровым сетям. Имеют нулевую стоимость копии и пользователи могут получать отличного качества их копии по сети. Структура стоимости цифровых продуктов, для которых высокая цена производства занижена, а предельные издержки производства стремятся к нулю, делают их значительно конкурентоспособнее обычных продуктов.

- Ч -

Частота процессора

Рабочая частота процессора, называемая иногда также "внутренней" частотой. Равняется произведению частоты шины на фактор умножения частоты.

Чек бокс (Check box)

Небольшой квадратик в диалоговом окне или заполняемой форме. Его можно пометить или очистить. Такой квадратик означает, что соответствующую опцию можно включить или выключить.

Чип (chip)

Укороченная форма термина микрочип (*microchip).* Так называют высокоскоростные миниатюрные интегральные схемы (integrated circuit), изготовленные из полупроводникового (semiconducting) материала, обычно кремния (silicon). Чипы используют в качестве микропроцессоров или памяти в персональных компьютерах и других электронных устройствах.

Чипсет

Набор микросхем материнской платы, реализующих архитектуру компьютера. Как правило, контроллер памяти входит в состав чипсета, поэтому, зная какой именно чипсет применен в компьютере, можно сделать выводы о применяемой памяти.

Число

❶ Одно из основных понятий математики, которое зародилось в глубокой древности. В связи со счетом предметов возникло понятие о целых положительных (натуральных) числах: 1, 2, 3,... Задачи измерения длин, площадей и т.п. привели к понятию рационального (дробного) числа. Потребность в точном выражении отношений величин (например, отношение диагонали квадрата к его стороне) привела к введению иррациональных чисел, которые вместе с рациональными числами составляют совокупность действительных чисел. В связи с решением уравнений 1-й степени (линейных уравнений) были введены отрицательные числа, а квадратных уравнений - комплексные числа. Затем были введены р-адические, алгебраические, кардинальные, трансцендентные и др.

❷ Тот или иной день месяца в его порядковом ряду, месте (при названии месяца слово "число" в речи обычно опускается, например, "первое мая" – вместо "первое число мая"). К примеру, "Первого числа (т. е. в первый день месяца) он возвращается из отпуска". Какое сегодня число? Какого числа твой день рождения? И т.д.

❸ Количество (кто (что)-либо, считаемые отдельными элементами, единицами, штуками). Собралось большое число гостей. Число книг в библиотеке сильно возросло.

❹ Совокупность, ряд известного количества кого (чего)-нибудь. К примеру, "в числе присутствующих не оказалось ни одного математика". Все дружно принялись за работу, и новички в том числе.

❺ (грамматическое) Грамматическая категория, показывающая, об одном или о большем числе предметов идет речь. К примеру, единственное число, множественное число (указывает на число предметов больше одного или, в языках, имеющих формы двойственного числа, - на число предметов больше двух). Изменяться в роде, числе и падеже.

Число «е», *неперово* число

Предел, к которому стремится выражение (1+1/n)n при неограниченном возрастании n: *e* = 2,718281828459045...; является основанием натуральных логарифмов; *e* - трансцендентное число. Название числа ***e*** по имени Дж. Непера мало обосновано.

Число с плавающей точкой (синоним – число с плавающей запятой)

Концепция представления чисел в компьютерах. Применяются также термины: экспоненциальная форма (или логарифмическая форма) представления нецелого числа. Число с плавающей точкой является формой представления вещественных (синонимы: рациональных, реальных, дробных, нецелых) чисел в компьютере, при которой положение точки или запятой в записи числа относительно разрядной сетки пространства ячеек запоминающего устройства не фиксировано. Представляется в виде:

–знака числа;

–целой части числа;

–точки, отделяющей целую часть числа от дробной;

–дробной части числа;

–признака порядка числа (обычно большая или маленькая буква E);

–знака порядка;

–значения порядка, то есть степени основания 10, на результат возведения в которую необходимо умножить число, чтобы получить окончательной значение числа..

Например: –2.4925E-2. Это же число может принимать формы (то есть может быть записано в разных видах):

–2492.5E-5 = –0.024925E0 = $–0.024925 \times 10^{0}$= $–24.925 \times 10^{-3}$.

Число с фиксированной точкой

В отличие от числа с плавающей точкой, не содержит множителя в виде основания 10, возведённого в соответствующую степень, определяемую показателем порядка. К примеру: 315.6391. Это же число в виде с плавающей точкой может быть представлено в следующих видах: $315.6391 \times 10^{0} = 3.156391 \times 10^{2}$ и т.д.

- Ш -

Шаблоны проектирования (см. *design patterns*)

Шина (bus) (магистраль)

Группа линий электрических соединений, обеспечивающих передачу данных и управляющих сигналов между компонентами компьютера.

Шина внешняя

Проводные соединения между внешним устройством компьютера и его системным блоком.

Шины частота

Как правило, термин применяется к главной шине компьютера, т.е. той, на частоте которой работает память. Для современных процессоров и чипсетов Intel пока официально не превышает 120MHz, однако ожидается увеличение её значения. Иногда называется внешней частотой.

Ширина шины

Количество линий ввода-вывода, т.е. число бит, которое может быть передано одновременно (для устройств с контролем чётности из этого количества иногда исключают линии, "отвечающие" за четность, как не передающие информации). Для системной шины определяется в первую очередь типом процессора. Увеличение ширины системной шины - простой способ увеличить общую производительность системы, однако это требует коренной перестройки программного обеспечения и периферии. Все процессоры, начиная с Pentium, имеют ширину шины не менее 64 бит. Размер одного банка памяти кратен (как правило, равняется) ширине системной шины.

Шифрование

Преобразование данных с целью защиты от несанкционированного просмотра, использования или модификации, особенно при передаче по линиям связи или транспортировке на сменных магнитных носителях. Для обратного преобразования - расшифровки - нужен специальный ключ.

Шкала измерений

Способ, которым приписываются численные значения. Обычно определяет тип анализа, который может быть осуществлен с этими данными.

Шлюз (Gateway)

❶ Компьютер, который обеспечивает объединение разнотипных сетей, использующих различные сетевые протоколы. Перед передачей данных из одной сети в другую шлюзовой интерфейс преобразует их, обеспечивая совместимость протоколов.

❷ Специальная программа, устанавливаемая на пользовательских машинах, которая через сетевой протокол, обеспечивает связь с сервером БД. Через такие шлюзы, приложения передают запросы серверу и получают обратно результаты. Часто, дополнительно устанавливается библиотека (ODBC, OLE DB и т.п.), предоставляющая приложениям API для работы с сервером БД.

❸ Internet-узел, подключенный одновременно к двум сетям, которые используют различные протоколы. Шлюзом также называются аппаратные и/или программные средства, объединяющие сети с различными протоколами.

❹ Соединение между двумя сетями. Шлюз позволяет обеспечить передачу данных между сетями, использующими различные протоколы (например, между сетями NetWare и не-NetWare) используя стандартные протоколы, такие как TCP/IP, X.25 и SNA.

Шрифт (font)

Набор символов, обладающих одной гарнитурой, кеглем и стилем, а также, возможно, снабжённых некоторой совокупностью спецэффектов. Операционная система Windows обеспечивает работу с четырьмя типами шрифтов, различающихся способом формирования символов и, как следствие этого, назначением: с растровыми, векторными и принтерными шрифтами, а также шрифтами TrueType. В зависимости от особенностей начертания большинство шрифтов можно отнести к одной из двух категорий: к шрифтам с засечками (T) или к рубленным шрифтам (T). Также шрифты делятся на равноширинные или пропорциональные.

- Э -

Эвристика (Эвристическое программирование)

Эмпирическое правило, упрощающее или ограничивающее поиск решений в предметной области, которая является сложной или недоступной всякому пониманию.

ЭВМ (электронная вычислительная машина)

Старое наименование компьютеров в те времена, когда они производились на электронных лампах, а затем на диод-транзисторной элементной базе.

Экземпляр

❶ Копия чего либо (вещи, информационного фрагменты и т.д.).

❷ (instance) В объектно-ориентированном программировании, экземпляр объекта некоторого типа.

Эккаунт (см. *account*)

Эксперимент (от латинского experimentum - проба, опыт)

Метод познания, при помощи которого в контролируемых и управляемых условиях исследуются явления природы и общества. Нередко главной задачей эксперимента служит проверка гипотез и предсказаний теории (так называемый решающий или натурный эксперимент).

Эксперт

Человек, который за годы обучения и практики научился чрезвычайно эффективно решать задачи, относящиеся к конкретной предметной области.

Экспертная система (ЭС)

Компьютерная программа, аккумулирующая знания экспертов и фундаментальные знания в той или иной предметной области, обладающая способностью к логическим выводам и выступающая в качестве электронного консультанта для лиц, принимающих решения, а также обеспечения высокоэффективного решения задач в некоторой узкой предметной области. Такие программы, как правило, представляют знания символически, исследуют и объясняют свои процессы рассуждения и предназначены для тех предметных областей, в которых людям для достижения мастерства необходимы годы специального обучения и практики.

Экспоненциальная форма представления числа (см. *число с плавающей точкой; синоним – число с плавающей запятой*)

Экстранет (Extranet)

Объединённая сеть, которая использует интернет технологии для соединения фирм и предприятий с их поставщиками, клиентами или другими фирмами, связанными общими целями. Экстранет можно представить в виде части Интранет'а компании, которая сделана доступной для других компаний или уже является собственность нескольких компаний. Общая для них информация доступна только для участников комплекса или может открываться для доступа по особым соглашениям.

Электронная библиотека (Digital library)

Распределенная информационная система, позволяющая надежно сохранять и эффективно использовать разнородные коллекции электронных документов (текст, графика, аудио, видео и т.д.) через глобальные сети передачи данных в удобном для конечного пользователя виде.

Электронная цифровая подпись (ЭЦП) (Digital signature)

Аналог собственноручной подписи физического лица, представленный как последовательность символов, полученная в результате криптографического преобразования электронных данных с использованием закрытого ключа ЭЦП, позволяющая пользователю открытого ключа установить целостность и неизменность этой информации, а также владельца закрытого ключа ЭЦП.

Электронный обмен данными (ЭОД) (Electronic data interchange (см. *EDI*))

Элемент

❶ Составная часть сложного целого.

❷ Атомарная составляющая модели. (UML).

Элиэс (Alias)

❶ Псевдоним, альтернативное имя.

❷ Дескрипторы одного и того же сегмента памяти. Каждый псевдоним может определять для сегмента различный тип и права доступа (access right).

❸ Сокращенное название команды или последовательности команд.

❹ Другое имя приложения или документа. Псевдоним пиктограммы занимает на диске меньше места чем оригинал.

❺ Паразитный сигнал, появляющийся при восстановлении аналогового сигнала из цифрового при недостаточно высокой частоте дискретизации.

❻ Короткий псевдоним можно использовать вместо более сложного адреса электронной почты (electronic mail).

Эмблема(от греч. Emblema - вставка, выпуклое изображение)

Условное прояснение отвлеченного понятия с помощью графического изображения, обладающего интернациональной или, по крайней мере, общенациональной узнаваемостью.

Эмпирический

❶ Основанный на опыте.

❷ Следующий эмпиризму.

Эмуляция

Выполнение вычислительной машиной программ, записанных в системе команд другого компьютера.

Эпистемология (от греческого *episteme* (знания) и *logos* (теория)

❶ Отрасль философии, связанная с теоретическим изучением природы, систем и действительности в представлении человеческих знаний. Включает взаимоотношения между исследователем (субъект), знанием (объект) и процессом познания.

❷ То же, что теория познания. (*см. теория познания*).

- Я -

Явление

❶ (явление и сущность) Категории материалистической диалектики. *Сущность* выражает глубинные связи, внутреннюю основу вещей, а явление – её обнаружение. Сущность раскрывает себя в явлениях. Познание идёт от явления к сущности, от менее глубокого к более глубокому знанию. Примером физического явления может служить появление электрического тока в рамке, пересекающей силовые линии магнитного поля.

❷ Выделенный в тексте отрывок драматического произведения, на протяжении которого состав действующих на сцене лиц остаётся неизменным.

Ядро операционной системы

Резидентная часть операционной системы, управляющая процессами операционной системы и распределяющая для них физические ресурсы. Как правило, постоянно находится в оперативной памяти компьютера.

Язык

❶ Важнейшее средство человеческого общения, возникшее в процессе совместной трудовой деятельности людей. Является специфической особенностью человека. Возникая и развиваясь вместе с мышлением, язык является его основным и специфическим орудием и естественным материальным выражением.

❷ Система знаков, общая для членов данного языкового коллектива.

❸ Множество символов и совокупность правил, определяющих способы составления из этих символов осмысленных сообщений.

❹ Естественная или искусственная знаковая система, предназначенная для передачи информации. К естественным знаковым системам относятся языки общности: украинский, русский, английский и др., а к искусственным, в основном языки программирования: C++, Java и др.

Язык ассемблера (ассемблер) (assembly language, assembler)

❶ Язык программирования, понятия которого отражают архитектуру компьютера. Он обеспечивает доступ к регистрам, указание методов адресации и описание операций в терминах команд процессора. Ассемблер может содержать средства более высокого уровня: встроенные и определяемые макрокоманды, соответствующие нескольким машинным командам, автоматический выбор команды в зависимости от типов операндов, а также средства описания структур данных.

❷ Ассемблер, транслятор с языка ассемблера.

Язык высокого уровня (ЯВУ)

Язык программирования, характеризующийся высоким уровнем обобщения понятий, соответствующих некоторой области применения и позволяющий лаконично и ёмко определять задание для вычислительной машины в терминах, близких к использованию в профессиональной деятельности людей.

Язык низкого уровня (ЯНУ)

Язык программирования, отличающийся высокой степень детализации шагов при определении инструкций для компьютера. Как правило, каждой команде языка соответствует одна машинная команда. Промежуточными между языками низкого и высокого уровней, являются языки ассемблеров.

Язык программирования (ЯП)

❶ Язык программирования представляет собой стандартизированное средство коммуникации для сообщения компьютеру команд на выполнение конкретных задач. Он также предоставляет программисту возможность точно указать, какими видами данных компьютер должен манипулировать, а также последовательность действий в различных обстоятельствах.

❷Система обозначений, служащая для точного описания программ или алгоритмов для компьютеров. Языки программирования являются искусственными языками, в которых синтаксис и семантика строго определены. Поэтому при применении их по назначению они не допускают свободного толкования выражения, характерного для естественного языка.

❸ Формальный язык описания донных (информации) с целью их обработки на компьютере.

Язык сценариев (scripting language) (см. *скриптовый язык*)

Язык, элементы словаря которого являются макроопределениями, то есть они имеют много команд, представляющих собой мини-программы, предназначенные, как правило, для комбинирования уже существующих компонентов. К языкам сценариев относятся Visual Basic, Java, Perl, Python, Rexx и Tcl и т.д. Языки сценариев часто называют склеивающими языками (glue languages) или языками интеграции систем (system integration languages).

Ярлык (shortcut)

Файл, содержащий указатель (ссылку) на некоторый объект в *дереве ресурсов* Windows – другой файл, папку или принтер. Обеспечивает непосредственный доступ к объекту из другой папки, в частности с рабочего стола Windows. Имеет расширение .LNK и распознаётся по загнутой стрелке в левом нижнем углу его значка. Роль ярлыков выполняют также PIF и URL файлы.

Ячейка, ячейка памяти (cell, storage cell)

❶ Минимальная адресуемая область памяти данных. (ГОСТ 15971-84).

❷ Элемент памяти, обеспечивающий хранение элемента данных, таких как БИТ, БАЙТ, СЛОВО и т.д..

❸ Основная единица хранения данных в электронных таблицах (spreadsheet) в приложениях типа MS Excel. Клетка, ячейка, элемент.

❹ Позиция для интерфейсных плат.

❺ Элемент рыболовной сети.

ЛИТЕРАТУРА К ГЛОССАРИЮ

1. Bollen J., Heylighen F. Cybernetics Glossary. Создан: 18 марта, 1994 г., последняя модификация 6 октября 2003 г. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: (modified)http://pespmc1.vub.ac.be/index.html

2. Client/Server Software Architectures — Software Technology Review. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.sei.cmu.edu/str/descriptions/clientserver_body.html

3. Dictionary word count = 4136779 words in 739 online dictionaries now indexed. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.onelook.com/browse_en.shtml

4. E-Learning Glossary. Kaplan-Leiserson Eva. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.learningcircuits.org/glossary.html

5. Encyclopedia Britannica. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.Britannica.com

6. Glossary of e-Commerce. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.ifsworld.com/about_ifs/glossary.asp

7. Glossary of internet & intranet. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.nur.yamal.ru/operating_systems/internet_intranet/nbg2iig.shtml

8. Glossary of terms for internet resources. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.ucm.es/INET/hytelnet_html/glossary.html

9. Glossary of terms found in the Web services architecture. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://www.w3.org/TR/ws-gloss/

10. Java Glossary. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://java.sun.com/docs/glossary.html

11. Krippendorff Klaus. Web Dictionary of Cybernetics and Systems. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://pespmc1.vub.ac.be/ASC/Kripp.html (http://www.asc.upenn.edu/usr/krippendorff/)

12. On-line Encyclopedia. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://dic.academic.ru/

13. On-line Encyclopedia. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://eco.rea.ru/misc/enc3p.nsf/ByID/NT00017B52

14. Society for Risk Analysis (SRA) glossarium. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://www.sra.org/news.php

15. Wikipedia, the Free Encyclopedia. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.wikipedia.org/wiki/Development

16. Баранов Ю.Б., Берлянт А.М., Капралов Е.Г., Кошкарев А.В.,Серапинас Б.Б., Филиппов Ю.А. Геоинформатика. Толковый словарь основных терминов. – М.: ГИС-Ассоциация, 1999. –204с.

17. Богумирский Б. Энциклопедия Windows 98 (второе издание). –СПб.: Питер, 2001. – 896с.

18. Болотова Людмила, Любкин Сергей, Резер Владимир. Интеллектуальные информационные технологии (история и тенденции развития). WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://www.osp.ru/cio/2002/05/031_1_print.htm

19. Большая советская энциклопедия. В 22 томах. Т. 59. – М.: Изд-во «Советская энциклопедия», 1976. – 600 с.

20. Бордовский Г.А., Извозчиков В.А., Исаев Ю.В., Морозов В.В.. Информатика в понятиях и терминах. Кн. для учащихся ст. классов сред. шк./ Под ред. В.А. Извозчикова. -М.: Просвещение, 1991. – 208 с.

21. Борковский А.Б. Англо-русский словарь по программированию и информатике (с толкованиями). –М.: Рус. Яз., 1990, 335 с.

22. Брандт Д. Architectures. Экзамен – экстерном. (экзамен 70-100). СПБ.: Питер, 2001. – 432 с.

23. Буч Г., Рамбо Д., Джекобсон А. Язык UML: Руководство пользователя : Пер. с англ. –М.: ДМК, 2000. –432с.

24. Виртуальное предприятие как эффективная форма организации внешнеэкономической деятельности компании. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://www.ptpu.ru/issues/4_03/16_4_03.htm

25. Вычислительная техника и обработка данных. Терминологический и толковый словарь фирмы IBM. Пер. с англ.. Т. Тер–Микаэляна. – М.: Статистика, 1978. – 231 с.

26. Гейтс Билл. Бизнес со скоростью мысли. - М.: ЭКОЛОТ, 2001. –273с.

27. Глоссарий B2B. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://b2b.ibs.ru/analyst/glossary.asp

28. Глоссарий COMSTAR Communications. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://www.comstar.ru/glossarium.shtml?letter=C

29. Голобуцький О., Шевчук О. "Електронний уряд". Словник термінів. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://golob.narod.ru/glossary.html

30. Григорьев В.Л. Англо-русский толковый словарь PC. –М.: Компьютер, ЮНИТИ, 1997. –471с.

31. Клуб знатоков DWH, OLAP и XML. Словарь технологических терминов. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://www.iso.ru/club/rsh.html

32. Математическая энциклопедия. / Ред коллегия И.М. Виноградов (гл. ред.) и др. (в пяти томах). Т. 1 –М.: «Советская энциклопедия», 1977. т. 1. А–Г. 1977. –1152 с.

33. Мирончиков И.К., Павловцев В.А.. Англо-русский толковый словарь по Интернет.– Минск, М:. – 2000. –134с.

34. Митчелл Ш. Толковый словарь компьютерных технологий. –СПб.: ООО ″ДиаСофтЮП″, 2002. – 720 с.

35. Он-лайн геоинформационный глоссарий ESRI. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.esri.com/library/glossary/a_d.html

36. Панкратова Н.Д. Становление и развитие системного анализа как прикладной научной дисциплины // Системні дослідження та інформаційні технології, 2002, №1. –С. 65-94.

37. Поляков А. Глоссарий терминов, имеющих отношение к компьютерной памяти WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.htc.ukrtel.net/ixbt/memgloss.html

38. Программные средства вычислительной техники: Толковый терминологический словарь-справочник. – М.: Издательство стандартов, 1990. – 368 с.

39. Рамбо Джеймс, Якобсон Айвар, Буч Грэди. UML: Специальный справочник. СПб.: Питер, 2002. –656с.

40. Русско-английский глоссарий по информационному обществу. Сто базовых терминов. Совместный проект Британского Совета в России, Института развития информационного общества и проекта "Российский портал развития" (грант # CG 012 программы *info*Dev Всемирного Банка). WEB-сайт (Электронн. ресурс) / Способ доступа: http://www.iis.ru/glossary/governance.en.html

41. Словарь Java-терминов. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://asuxxi.narod.ru/oradoc/Java/java007.htm

42. Словарь информационных терминов. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.udel.edu/alex/dictionary.html#ext

43. Соммервилл Иан. Инженерия программного обеспечения, 6-е издание.: Пер. с англ. –М.: Издательский дом «Вильямс», 2002. – 624 с.

44. Терминологический словарь по основам информатики и вычислительной техники/А.П. Ершов, Н.М. Шанский, А.П.Окунева, Н.В. Баско; Под ред. А.П. Ершова, Н.М. Шанского. –М.: Просвещение, 1991. – 159 с.

45. Толковый словарь по вычислительным системам / Под. Ред. В. Иллингуорта и др.: Пер. с англ. А.К. Белоцкого и др. – М.: Машиностроение, 1990. – 560 с.

46. Фридланд А. Я., Ханамирова Л. С., Фридланд И. А. Информатика и компьютерные технологии: Основные термины: Толковый словарь. 3-е изд., испр. и доп. – М.: ООО «Издательство Астрель»: ООО «Издательство АСТ», 2003. – 272 с.

47. Хаббард Рон Л. BASIC STUDY MANUAL (Руководство по основам обучения). WEB-сайт (Электронн. ресурс) / Способ доступа: URL:: –NY.: Prentiss Hall. 1988. –250p. http://ufo.knet.ru/bibliot/00700/00700/00100.txt

48. Экзотариум. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://www.handy.ru/gadgets/gadgets.html

49. Экономический глоссарий. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://www.ndc.ru/seminars/piter/10.htm

50. Экономический глоссарий. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://uic.nnov.ru/~chep/html/comopr.htm

51. Экономический глоссарий.. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://www.glossary.ru/

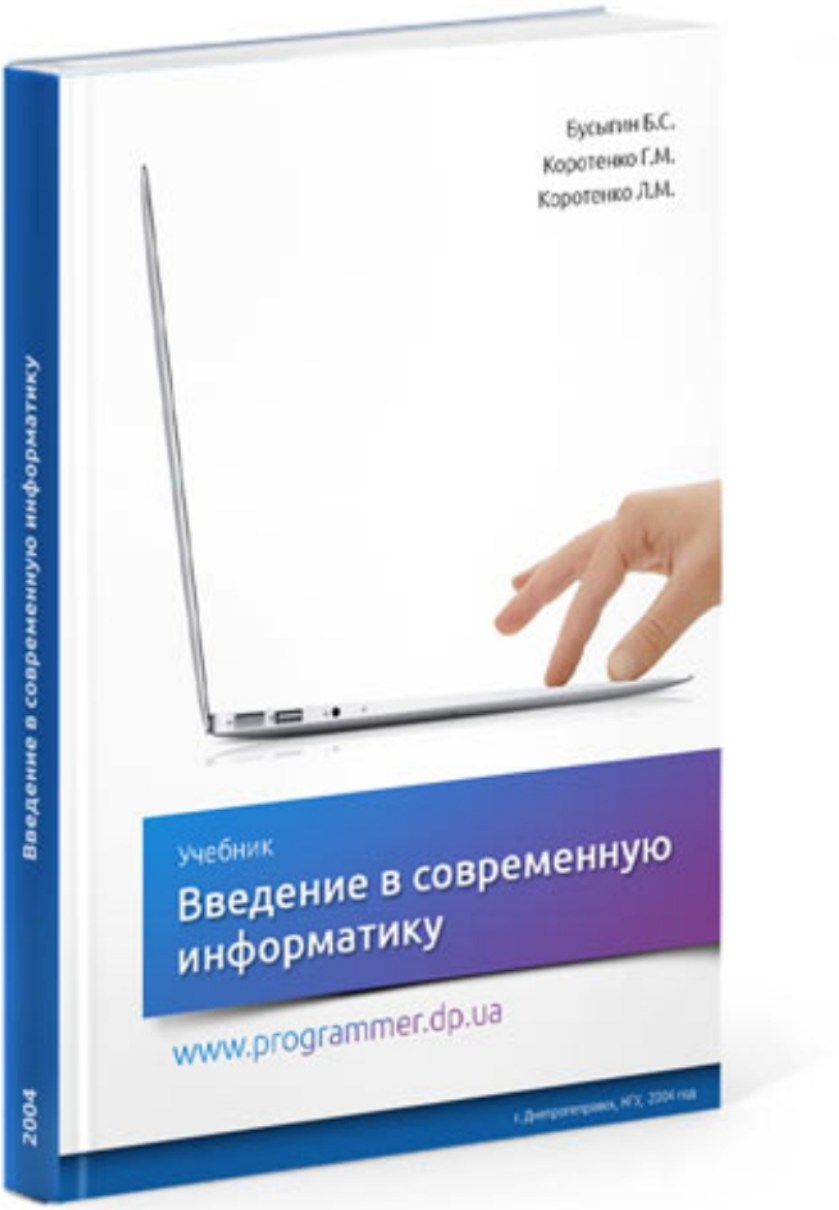
52. Энциклопедия библиотечного дела. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.wcsu.edu/library/odlis.html#A

53. Энциклопедия персонального компьютера. WEB-сайт (Электронн. Ресурс) / Способ доступа: URL: http://www.megabook.ru/pc/encyclop.asp?TopicNumber=562

кафедра

Программного обеспечения компьютерных систем

www.programmer.dp.ua



Электронная книга издана при поддержке
Студии брендинга “Свой стиль”
www.svoy-style.com.ua