Бусигін Б.С.
Коротенко Г.М.
Коротенко Л.М.

Підручник

Прикладна інформатика

www.programmer.dp.ua

м. Дніпропетровськ, НГУ, 2004 рік

Книга є переможцем VI-го Обласного міжвузівського конкурсу на «Кращі наукові, навчальні, навчально-методичні та художньо-публіцистичні видання», 2006 р.

Книга затверджена Міністерством освіти і науки, молоді та спорту України як підручник для ВНЗ.

кафедра
Програмного забезпечення комп'ютерних систем
www.programmer.dp.ua

Відомості про авторів

Автор	*e-mail*
Бусигін Борис Сергійович доктор технічних наук, професор кафедри геоінформаційних систем НГУ	
Коротенко Григорій Михайлович доктор технічних наук, професор кафедри геоінформаційних систем НГУ	gkorotenko@rambler.ru
Коротенко Леонід Михайлович кандидат технічних наук, доцент кафедри програмного забезпечення комп'ютерних систем НГУ	leonid_korotenko@ukr.net

Детальніше про книгу на сайті:

www.programmer.dp.ua/books/

УДК 510.5
ББК 33.16
С16

Рецензенти:

Науково-методична комісія з комп'ютерних наук Науково-методичної ради Міністерства освіти і науки України.

О.Ф. Приставка, д-р техн. наук, професор (Дніпропетровський національний університет, професор кафедри математичного забезпечення і електронно-обчислювальних машин).

І.В. Жуковицький, д-р техн. наук, професор (Дніпропетровський національний університет залізничного транспорту ім. акад. В. Лазаряна, завідувач кафедри електронно-обчислювальних машин).

С16 **Б.С. Бусигін , Г.М Коротенко, Л.М Коротенко.** Прикладна інформатика: Підручник. Дніпропетровськ: Національний гірничий університет, 2004. – 559 с.

ББК 33.16
С16

ЗМІСТ

Якщо б за останні 25 років авіаційна промисловість розвивалася також стрімко, як обчислювальна техніка, то Боїнг-767 можна було б сьогодні придбати за 500 доларів і облетіти на ньому повз земну кулю за 20 хвилин, витратив при цьому 19 літрів пального.

Журнал «Scientific American», 1985 р.

Вступ

Сьогодні вже нікого не дивує той факт, що інформаційний джин, стрімко увірвавшись у сучасне суспільство, різко прискорив процес виробництва нових знань. З часу свого з′явлення біля п′ятидесяти років тому, інформатика стала визначною технологією нашого часу. Досягнення в області інформаційних технологій за короткий час стали однією з головних рухаючих сил сучасного суспільства. Людство з великими зусиллями намагається встигати за цими темпами розвитку.

Разом з тим, майже кожний з нас, підключившись за допомогою модему комп'ютера до всесвітньої мережі Інтернет, зараз же поринає у сплав багаторівневих, найновітніх, програмно-апаратних мережових технологій, що інтенсивно розвиваються. Скоріше за все, на Вашому персональному комп'ютері з процесором, який має частоту не менш за 1,7 ГГц (виробництва Intel, а може і AMD), вже розгорнута операційна система Windows XP (до речі, згадайте, з якого часу вона замінила у Вас попередні версії 95, 98. Millenium?), а набір пристроїв Вашого комп'ютера вже вимірюється десятками одиниць (відеокарта, аудіо колонки, мікрофон, TV-тьюнер, FM-тьюнер, аудіо - и відео-програвачі, привод CD-RW, сканер, принтер, флеш-пам'ять) і цей список Вас вже не дивує, а його можна продовжувати і продовжувати…

І якщо через декілька років у Вас виникне бажання ще раз відкрити цю книгу, а може вона просто випадково потрапить до Ваших рук, то Ви зможете переконатися, як змінилася перша частина списку і поповнилася друга.

Вже на 2002 рік, за думкою спеціалістів, з 132-х тем у сукупності знань з інформатики, що сформувалися, «ядро» обов'язкових складали 64![1] Уся

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

сукупність знань з інформатики умовно може бути розподілене на 14 областей (див. таблицю 1).

Таблиця 1. Структуризація знань з інформатики на 2002 рік

Дискретні структури (DS)	Графіка і візуалізація (GV)
Основи програмування (PF)	Інтелектуальні системи (IS)
Алгоритми і теорія складності (AL)	Управління інформацією (IM)
Архітектура і організація ЕОМ (AR)	Методи обчислень (CN)
Операційні системи (OS)	Людино-машинна взаємодія (HC)
Розподілені (мережові) обчислення (NC)	Соціальні и професійні питання програмування (SP)
Мови програмування (PL)	Програмна інженерія (SE)

Багато процесів, що впливають на інформатику, пов'язані з прогресом у розвитку нових сучасних технологій. Як еволюційні, так и революційні зміни у інформаційних технологіях (IT) підвищили важливість багатьох тем і, в особливості, наступних:

❶ WWW и його прикладення (програмування процесів обробки і обслуговування ресурсів "Всесвітньої павутини" (Web-сайтів, Web-вузлів, порталів і т.д.)).

❷ Використання програмних інтерфейсів додатків (API) (поміжплатформена взаємодія і переносність програмних компонентів).

❸ Мережові технології, які базуються на TCP/IP (взаємодія мереж з різними протоколами, операційними системами і рівнями взаємодії: локальні, глобальні і пірінгові).

❹ Бездротові (рос.-беспроводные) технології та їх програмування (WAP) (сумісне функціонування і обмін інформацією, поміж мобільними комп'ютерними і телефонними системами, не зв'язаних комунікаційними лініями).

❺ Геоінформаційні системи и технології (ГІС) (глобальне, всесвітнє, сумісне накопичування, обробка, взаємодія и виведення колосальних об'ємів цифрової інформації, зібраної про поверхню Землі і інфраструктуру просторових об’єктів, що розташовані на ній).

❻ Графіка і мультімедіа (сумісне кодування, уявлення, обробка, запис и вивід аудіо - і відеоінформації).

❼ Вбудовані системи (конструювання і програмування мікропроцесорів, що вбудовуються у сучасну техніку: автомобілі, побутову, радіо -, відео -, електро - і усяку іншу).

❽ Реляційні, об'єктно-орієнтовані та XML-орієнтовані бази даних (бази даних, які формуються і керуються за допомогою різних механізмів уявлення даних і обробки об'єктів).

❾ Властивість до взаємодії (інтероперабільність-interoperability) (програмування засобів забезпечення доступу практично до будь яких джерел інформації: ресурсам бібліотек, комп'ютерам, радіотелефонам, ноутбукам, локальним мережам, Internet, WWW та ін.).

❿ Об'єктно-орієнтоване і компонентно-орієнтоване програмування (ООР и СОР) (створення комп'ютерних, програмних і операційних систем, а також і мов програмування, що забезпечують адекватне уявлення у інформаційних системах об'єктів і сутностей реального світу для розширення можливостей їх реалізації і обробки).

❶❶ Надійність програмного забезпечення (створення механізмів забезпечення безперервного виконання програмних процесів уявлення і обробки даних у комп'ютерних і інформаційних системах будь якого рівня).

❶❷ Аналіз і управління контентом (Topic Maps) (розробка і впровадження теоретичних методів управління пошуком потрібної інформації у WWW і Internet).

❶❸ Безпека і криптографія (захист персональної, корпоративної і інформації у Internet від хакерських атак та несанкціонованого доступу, а також від комп'ютерних вірусів будь яких різновидів).

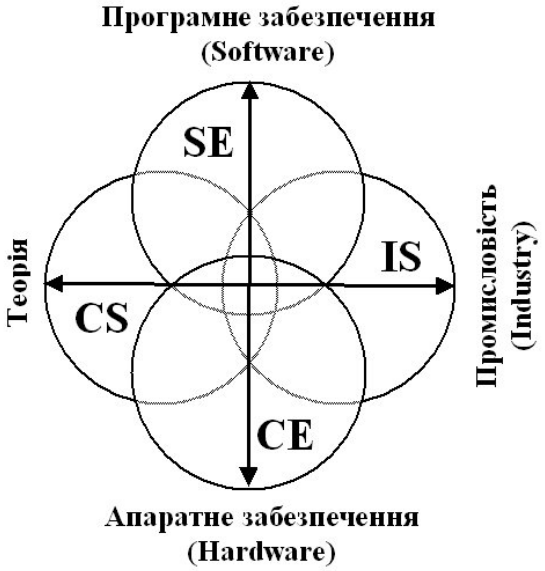


Рис. 1. Перехрешення деяких комп'ютерних галузей: Computer Science (CS), Software Engineering (SE), Information System (IS), Computer Engineering (CE)

Одне з можливих уявлень перехрещення і взаємодії найбільш великих сучасних комп'ютерних напрямів[2] наведено на рисунку 1. Безсумнівно, що один з найбільш містких секторів ринку комп'ютерних технологій належить інформаційним системам, які працюють у корпоративному просторі та спираються на наукові розробки (CS), програмну інженерію (SE) і апаратну технологічну базу (CE).

Не буде перебільшенням сказати, що переважна частина тем інформатики базується на використанні тих чи інших мов програмування, як виключно можливим інструменті управління різноманітними компонентами складних багаторівневих комп'ютерних систем, а також інформаційним контентом. І хоч ринок програмних продуктів постійно поповнюється все новими і новими додатками, покрити поле інтересів користувачів вони поки ще не в змозі…

На початкових етапах засвоєння їх можливостей, як правило, користувач застосовує інструменти стандартного інтерфейсу додатків (API), а потім, засвоїв головні функції, починає використовувати вбудовану мову системи

[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.

(типу *Visual Basic for Application*). І дійсно, навряд чи Вас надовго захопить процес використання чужих програм (доречи, і їх хтось створював у свій час!). А ще більшу незадоволеність викликає необхідність додержуватися багато чисельних керівництв для «чайників», постійно виконуючи рекомендації типу: «Підведіть курсор миші до меню вікна додатку, виберіть потрібну команду і клацніть лівою кнопкою миші ОК!». І неминуче наступить момент, коли Вам самому захочеться направити хід дій комп'ютера на свій розсуд. А це і будуть Ваші перші кроки у програмуванні и у програмування.

Рис. 2.
Андрій Петрович Єршов (1931-1988), радянський математик, академік АН СРСР (зліва) и Джон МакКарті (батько мови Лісп, США)

Але, перед тим, як входити до цієї таємничої і захоплюючої галузі людської діяльності, замислитесь над словами академіка А.П. Єршова (рис. 2), які він висловив на Об'єднаній обчислювальній конференції Американської федерації суспільств з обробки інформації в місті Атлантик-Сіті (США) ще у 1972 р., у своєму виступу під назвою «О людським і естетичнім факторах в програмуванні»: «Програмування стає масовою професією. Однак потрібно мати на увазі, що зараз це, мабуть, сама трудна з усіх масових професій, причому, нажаль, ця трудність ще не визнана у повній мірі».

Ця трудність полягає у тому, що саме програмісти безпосередньо впираються у межі людського пізнання у виді алгоритмічно не розв’язуваних проблем і глибоких таємниць роботи головного мозку.

Трудність полягає ще і у тому, що власний стек програміста повинен бути не у 56 позицій глибини, як це виявили психологи у середньої людини, а той же глибини, що і стек у його черговій задачі, яка підлягає програмуванню, плюс ще як мінімум дві-три позиції.

Трудність програмування полягає також і у тому, що програміст повинен володіти здібністю першокласного математика до абстракції і логічного мислення у поєднанні з "едісонівським" талантом споруджати все, що завгодно, тільки з нуля та одиниці. Він повинен поєднувати акуратність бухгалтера з проникливістю розвідника, фантазію автора детективних романів з тверезою практичністю економіста. А, крім того, програміст повинен мати смак (рос.-вкус) до колективної праці, розуміти інтереси користувача і багато, багато іншого».

Це один бік проблеми, але є ще і другий. А.П. Єршов звернув увагу присутніх на дуже важливий аспект, пов'язаний з проблемами програмування: «У час мого перебування в 1970 р. у Сполучених Штатах на мене призвели дуже велике враження нові ідеї професорів Масачусетського технологічного інституту Марвіна Мінського і Сеймура Пейперта в галузі навчання дітей. Вони викинули у корзину ходяче уявлення про те, що діти вчаться несвідомо методом наслідування (рос.-подражания). Мінський і Пейперт доказують, що

людина чомусь навчається тільки у тому разі, коли у нього в голові складається блок-схема дій, виділені підпрограми і прокладені інформаційні зв'язки. Професор Пейперт назавжди обернув (рос.-обратил) мене в свою віру на прикладі жонглювання двома м'ячами, коли, апелюючи до моїх здатностей програміста, він за десять хвилин навчив мене тому, що я сам не зробив би і за декілька часів. Таким чином, людина незмірно посилить інтелект, якщо зробить частиною своєї істоти спроможність планувати свої дії, виробляти загальні правила и засіб їх застосування до конкретної ситуації, організовувати ці правила у осознану і висловлювану структуру, – одним словом, зробиться програмістом.

Творча і конструктивна природа програмування не потребує особливих доказів. В своїй творчій природі воно йде набагато далі більшості інших професій, наближаючись до математики і письменництва. В більшості інших професій ми лише «приручаємо» за допомогою сил природи ті або інші фізичні чи біологічні явища, не обов'язково осягаючи їх сутність. В програмуванні ж ми у деякому сенсі йдемо до кінця. Один з тезисів сучасної теорії пізнання: «ми знаємо щось, якщо можемо це запрограмувати» – дуже опукло характеризує цей максималізм програмування.

У цьому і полягає фундаментальна роль головних понять інформатики у формуванні здібностей і інтелекту сучасної людини і у витікаючої з цієї ролі необхідності впровадження комп'ютерів в учбовий процес».

І якщо після цього Ви ще не передумали займатися програмуванням, то перед Вами постане ***питання***: «За допомогою якої книги и яку мову програмування вивчати»? Відповідь неоднозначна, але, на щастя, може бути «розкладена на складові». Умовно його можна розбити на три частини.

1. В принципі, програмувати можна на будь якій мові програмування. У класичній роботі Бома и Джакопіні (надрукована в журналі Communications of the ACM у травні 1966 р.) було показано, що для програми, у якій усякий модуль проектувався з єдиним входом і єдиним виходом, а сама програма є множиною вкладених модулів, кожний з яких також має один вхід и один вихід, остання може бути реалізована в мові, яка включає тільки дві головні управляючі конструкції. Фактична реалізація цих конструкцій може бути різною у різних мовах програмування. Принцип, який викладено в статті Бома и Джакопіні, зветься «структурною теоремою», має фундаментальне значення і складає основу більшості реалізацій алгоритмів при проектуванні програм.

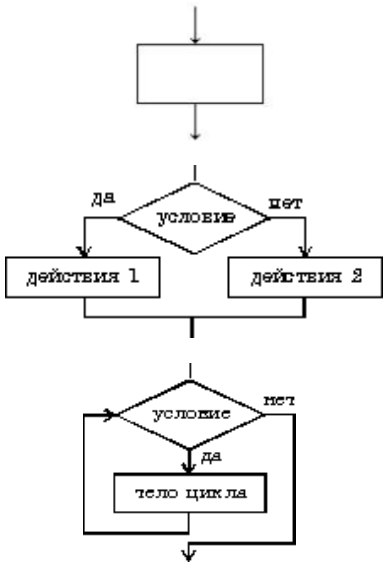


Рис. 3. Три головних блока в програмі

По Бому и Джакопіні для побудови програми потребується три головних складових блока (рис.3):

❶ функціональний блок або лінійна послідовність операторів;

❷ умовний оператор (конструкція

прийняття двоїчного або дихотомічного рішення);

❸ конструкція узагальненого циклу (цикл з параметром, з передумовою, з постумовою і т.д.).

Тоді для усякої програми, складеної з вказаних конструкцій, завжди може бути отриманим ще й доказ її коректності!

2. Якщо Ви маєте намір зразу ж вирішувати конкретні задачі у конкретній організації, то при обранні мови програмування, необхідно (і прийдеться!) враховувати масу факторів: накопичений досвід організації, специфіку задачі, яка вирішується, апаратне забезпечення, яке існує у цій організації, наявність ліцензійних програмних продуктів, встановлений термін виконання проекту і т.д., що і дозволяє вже обрати конкретну мову програмування.

3. Але, якщо Ви вирішили засвоїти систематичний і науковий підхід до програмування, тобто навчитися реалізовувати великі програми зі складними даними, то, безсумнівно, Вам потрібно починати з мови Турбо Паскаль.

Рис. 4.
Ніклаус Вірт перед тренувальним вилітом на бойовому винищувачі

Сама мова Паскаль з'явилася в м. Цюріх у «Школі програмування», на чолі котрої стоїть Ніклаус Вірт (1968-1999 рр. – професор інформатики в університеті м. Цюріх, Швейцарія) (рис. 4), який узагальнив свій багаторічний авторський досвід навчання програмуванню. Створений спеціально з педагогічними цілями Паскаль оказався дуже вдалим не тільки в силу того, що йому просто навчатися, але й як основа обговорення мов програмування взагалі. Мова проектувалася з врахуванням спрощення написання відповідного транслятора, а опис Паскалю, який займає біля 30 сторінок тексту, уявляє собою яскравий приклад систематичного и наукового підходу до навчання побудови програм. Для порівняння спробуйте зазирнути в книгу Б'ёрна Страуструпа з описом мови C++ і, як говориться, відчуйте різницю!

Ніклаус Вірт першим сформулював тезис про те, що програми уявляють собою, з наукової точки зору, конкретні формулювання абстрактних ***алгоритмів***, які основані на конкретних уявленнях і структурах ***даних***. Це і знайшло своє відображення в мові Паскаль. Так, засвоїв конструкцію опису масиву:

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

Ви далі впізнаєте, що конструкція `1..N` уявляє собою обмежений тип і може бути далі уявлена самостійним ідентифікатором (ім’ям чи назвою), наприклад `Border`. І тоді той же опис буде виглядати так:

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

Повільний перехід від простих (тобто базових для усіх мов програмування!) типів даних до складових (рос.-составных), а потім і до об'єктних (що зовсім не важко!) дозволяє навчитися легко переміщатися по рівням абстракцій, що є головним в програмуванні. Це повинно дозволити Вам програмувати або швидко навчитися програмувати практично майже на любій з головних мов програмування, котрі прийдеться використовувати в майбутньому на робочому місці.

Важливо підкреслити, що у програмуванні, на відміну від математики, немає формул, підставив до яких деякі початкові дані можна отримати гарантований результат. Але тут є звісний набір прийомів и прикладів їх використання. Фрагменти коду і приклади програм, звичайно, можуть багато чому навчити, але не слід забувати і про можливості направляти хід думок програміста, тобто пропонувати йому вже апробовані алгоритми проектування програм и проектування алгоритмів. Тим більш, що відомий у усьому світі Дональд Кнут (США)[3] (рис. 5) розробив і надрукував аж три томи найбільш ужитих (рос.-употребительных) обчислювальних алгоритмів. Таким чином у основі програмування лежить творчий акт і основною задачею при навчанні є розвиток здібностей людини творчо мислити. І тут потрібно не стільки розповідати, скільки показувати, як робиться «те» чи «інше». На цьому шляху велику роль грає вибір матеріалу і підбір прикладів, у котрих підкреслюються основоположні принципи програмування.

Рис. 5. Професор комп'ютерних наук Дональд Кнут, який розробив і надрукував три томи книжок з обчислювальними алгоритмами для комп'ютерів.

Як витончено підмітив Н. Вірт: «Програмування – це мистецтво конструювання». Звичайно, навчити конструкторській або винахідницький діяльності досить важко. Але можна скористуватися старим як світ способом. Подібно до того, як дитина вчиться говорити, складаючи з літер (рос.-букв) слова, а з слів – речення, програміст «складає» алгоритми програмних модулів, з модулів – програми, а з програм – системи[4]…

«Входження» у Паскаль суттєво полегшується присутністю компактного інтегрованого середовища програмування Турбо Паскаль (ТП). Ще однією з причин почати свої перші кроки у

[3] Професор Комп'ютерних наук Стендфордського університету (США), володар премії Т'юринга у 1974 р., член Британського комп'ютерного товариства з 1980 р., Почесний член організації IEEE з 1982 р. Член Американської академії Мистецьтва і Науки, Національної Американської Академії Наук, Національної Американської Інженерної Академії, закордонних академій: Французьської Академії наук (l'Académie des Sciences (Paris)) і Норвезьської Наукової академії (Det Norske Videnskaps-Akademi (Oslo)). Надрукував 160 статей і 19 книг, має багато наукових і державних нагород різних держав світу, є почесним професором Оксфордського, Паризького і Петербурзького університетів, а также багатьох коледжей світу.

[4] Бортове програмне забезпечення (ПЗ) ***усіх*** російських супутників зв'язку пишеться на мові Модула-2, розробленої Н.Віртом в 1979 р. після створення мови Паскаль в 1970 р.

програмуванні на Турбо Паскалі, є його багатофункціональне середовище розробки, котре уключає текстовий редактор для уводу програм, дбайливий компілятор, який у випадку присутності помилки в тексті Вашої програми, встановлює курсор у цьому місці і повідомляє про номер і назву цієї помилки, а також дебаггер, тобто засіб налагоджування програм (рос.-отладчик), котрий дозволяє досліджувати створені Вами програми. Простота його використання дозволить Вам зосередитися на головних тонкостях програмування на відміну від монстроподібних систем швидкої розробки програм, вивчення можливостей яких потребує окремих (а іноді і напружених) зусиль. Краще за усе цю ситуацію ілюструє історія, що приведена Джозефом Фоксом у його книзі «Программное обеспечение и его разработка»: «Коли на початку 1970-х рр. ми успішно завершили першу робочу версію нової програмної системи управління повітряними перевезеннями, ми відчули почуття величезного задоволення (при цьому група з 500-т програмістів працювала над її створенням 7 років).

Система була відправлена в Джексонвілл, шт. Флоріда, для випробувань в робочих умовах в нічну зміну в Центрі управління авіаперевезеннями. Однак диспетчери з Джексонвилла відмовились користуватися нею – вони заявили, що вона «ненадійна».

«Ненадійність» диспетчерської авіаслужби потребує особливої уваги. Нам вдалося вирішити цю проблему ***виключенням*** із складу системи ***значної частини функцій***, котрі були уключені до неї. Наступний варіант програми, який було поставлено у Джексонвіллі, містив значно менше системних можливостей. І диспетчерам він сподобався. А потім поступово, дуже повільно ми почали додавати функції, які вже були запрограмовані і протестовані нами раніш.

Зрозуміло, що якщо Вам запропонують пуд морозива, важко очікувати, що Ви зможете проковтнути його за один прийом. Це не вийде».

Рис. 6. Комп'ютер IBM System/360, 1974 р.

Можливо, Вам буде цікаво узнати, що автори цієї книги за своє життя «впізнали» усі принадності (рос.-прелести) програмістського ремесла. Ще на студентській лаві у 1970 р. прийшлося програмувати в машинних кодах для електронної обчислювальної машини (ЕОМ) «Урал-1». На переддипломній практиці в Інституті кібернетики АН УРСР ім. академіка Глушкова програмували в кодах трьох адресної машини М-220. Навчаючись у аспірантурі і потім, продовжуючи там же подальшу працю, програмували для ЕОМ БЭСМ-6 на мовах Алгол, Фортран і Автокоді «Мадлен», а для ЕС ЕОМ (лінія IBM, рис.6) – на мовах PL/1, Fortran і Assembler. Вже у епоху розповсюдження комп'ютерів IBM PC, працюючи у вищих навчальних закладах (ВНЗ), програмували на мовах Турбо Паскаль, Quick Basic, Пролог, Асемблер, Ліспо. Поточним часом перевагу віддають мовам Турбо Паскаль, C++, Visual Basic for Application,

Python. Вищевказаний мовний слалом «вивірив і розмітив» сам процес розвитку інформаційних технологій, а аж ніяк не ексцентричність у обранні підходу до вивчення деякої серії таких мов.

Ідея створення цього підручника виникла в процесі спілкування авторів з викладачами і студентами Національного гірничого університету, а також вчителями і школярами Регіональних учбових центрів НГУ у мм. Павлограді, Комсомольську, Олександрії і Дніпрорудному. Досвід цих зустрічей посвідчує, що, незважаючи на існування великої кількості навчальних книг і спеціальної літератури з інформатики, вивчення мови програмування у відриві від історії появи персональних комп'ютерів (ПК), головних тенденцій розвитку інформаційних технологій і зміни парадигми створення програм, виходить дуже «плоским». Крім того, звісно, що спеціалісти, які освоювали в студентські роки навчальний курс «з малюнками» і «без малюнків», як правило принципово відрізняються в подальшому один від одного по ***стилю мислення***. Тому автори приділили особливу увагу підкріпленню викладення матеріалу ілюстративними матеріалами.

Рис. 7.

Фрагмент з мультимедійної гри

Слід також мати на увазі, що певну трагікомічність ситуації з навчанням інформатиці на початкових курсах ВНЗ придає той факт, що персональний комп'ютер практично перетворився у багатьох сім'ях у настільний побутовий пристрій для підтримки дитячих ігор (рис. 7), а простота інсталяції операційної системи (ОС) Windows (спасибі Біллу Гейтсу та його команді!) дозволяє "юним талантам (рос.-дарованиям)" самостійно встановлювати цю ОС на ПК, а потім с легкістю користуватися всілякими ігровими, Інтернет - і мультімедійними програмами, що у простих (і не дуже) користувачів породжує ейфорію ***повного*** володіння інформаційними технологіями. Але одне діло освоювати майже за десять хвилин програми, котрі і ***створювалися*** у розрахунку на їх ***максимальну дружність*** непідготовленому користувачу, а зовсім інше – ***проектувати й розробляти такі програми***...

Головна частина популярної (і дуже потрібної!) літератури по оволодінню комп'ютерною грамотністю націлена на опис апаратних, програмних і мовних компонентів комп'ютерних систем. Стрімкий їх розвиток швидко «робить старим» матеріал, який у неї викладається. Тому у підручнику, що пропонується, особливу увагу було приділено висвітленню ряду питань, котрі, на думку авторів, з одного боку слабко відображаються у відомій літературі, а з

іншого боку – відносяться до розряду фундаментальних (тобто тих, які не старіють).

Наприклад, просте питання: «А що діється у процесі включення комп'ютера?» – викликає велике збентеження (рос.-замешательство) і становить у тупик багатьох юних комп'ютерщиків. Тому автори прийняли рішення заглянути усередину персонального комп'ютера і спорядили (рос.-снабдили) книгу докладним (рос.-подробным) глосарієм, який містить більш ніж 1000-і термінів з галузі комп'ютерних технологій, без якого багато речей зникають з поля зору при вивченні глибин і основ інформатики.

І до цього часу продовжується бурхлива полеміка про те, «що» і «як» потрібно у першу чергу викладати в інформаційних курсах (див. Додаток 1), тому на наш погляд тема ця невичерпна. Якщо ж казати у цілому про ціль даного підручника, то окрім безпосередніх задач навчання програмуванню автори прагнули показати становлення інформатики не як щось «дане згори (рос.-свыше)», а як боротьбу ідей, котрі висували і відстоювали живі й безсумнівно надзвичайно талановиті люди.

Звичайно, ряд глав можуть здатися при першому читанні досить складними для початківців. Попервах їх можна пропустити і перейти безпосередньо до теми, яка висвітлює особливості мови Турбо Паскаль. Але, повертаючи знов і знов до не у всьому зрозумілих глав, Ви переконаєтесь у тому, що багато речей не такі трудні, як це здається на перший погляд. Їх потрібно засвоїти тому, що на думку провідних закордонних фахівців, головною професійною функцією спеціалістів комп'ютерних напрямів на сучасному етапі є ***управління змінами у інфраструктурі інформаційних систем.***

Рис. 8. Самий перший переносний персональний комп'ютер, створений у 1981 р. у вигляді чемодану, який випускала фірма Osborne Computers (США). Вага у зібраному вигляді 10 кг.

Оскільки цей підручник не є монографією, посилання на літературу у тексті не наводилися, а список літератури до вступу знаходиться у його кінці. За невеликим виключенням, у більшості своїй, це улюблені книги авторів, котрі хочеться перечитувати неодноразово. Не дивлячись на відносну молодість інформатики як науки, в ній вже є свої класики. Їхні книги написані живою і образною мовою, вони повні реальних програмістських історій, котрі складають фольклор професії. Без цього фольклору важко бува зрозуміти багато тонкостей ремесла програміста. Сподіваємось, що вони визвуть і у читачів відповідний інтерес.

Зважаючи на широту «розкиду (рос.-разброса)» питань, які висвітлюються і досить високий рівень абстракцій, які містяться у деяких уявленнях, в таких випадках посилання на відповідні джерела і поняття подаються безпосередньо у виносках (рос.-сносках) до тексту.

Підручник, що пропонується, починається розділом про історію народження персонального комп'ютера з декількох причин (рис.8). По-перше, ми вважаємо її вельми захоплюючою (рос.-увлекательной) і корисною для входження до атмосфери стрімкого розвитку комп'ютерних технологій, де розділ про процес включення ПК служить містком до розуміння складності вирішених при цьому задач і поводом углибитися у фундаментальні питання обробки даних і командної основи роботи комп'ютера. По-друге, акцент скрізь робиться на фокусуванні Вашої уваги на точні формулювання в сфері існуючого різноманіття у визначенні багатьох термінів, велика кількість яких викликала появу багатьох комп'ютерних словників і Web-сайтів, що тлумачать багаточисельні неологізми, котрі з'являються лавино подібно…

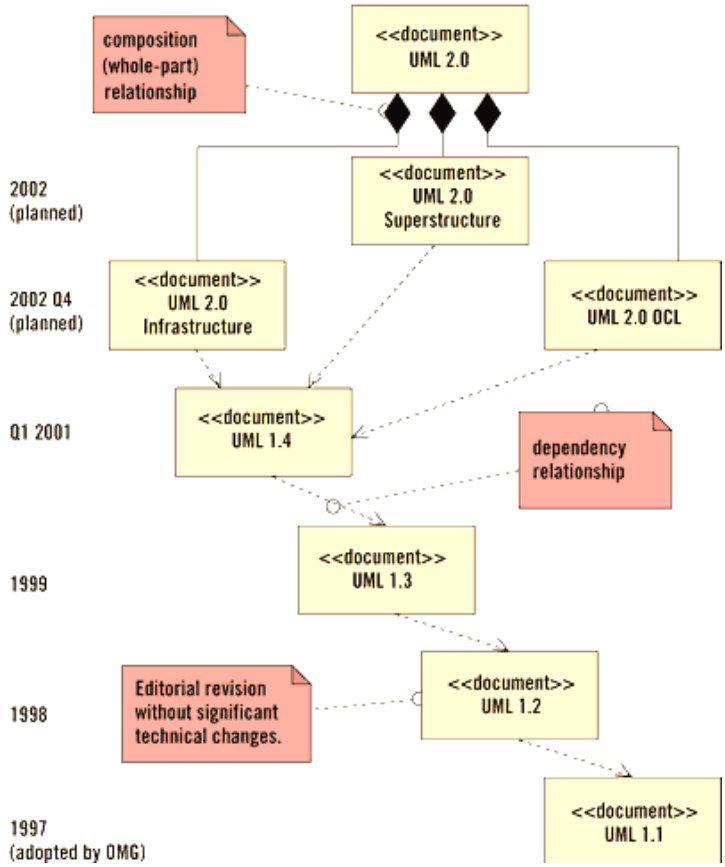


Рис. 9. Діаграма, що описує розвиток мови UML с 1997 г. по 2002 г., яка виконана за допомогою самої мови UML

Знання і уявлення значень постійно виникаючих термінів потрібно в першу чергу тому, що як заявив керівник японських програмістів на однієї з конференцій у 1977 р.: «Найбільш важливою мовою, котру потрібно знати програмісту, є не JCL або ПЛ/1, а японська мова». Коментарі, як говориться, зайві! (Хоча знання, в тому числі і мови UML аж ніяк не завадять (рос.-не помешают) (рис.9)).

У другому розділі описуються головні тенденції у розвитку інформаційних технологій и у фундаментальних уявленнях, що до них відносяться. Вам пропонується зануритися (рос.-погрузиться) в глибини різноманіття понять інтерфейсу і розглянути, як він мінявся від DOS до Windows…

Друга тема, котра примусить Вас по новому поглянути на вивчення Турбо Паскаля, це розвиток мов програмування: які з них і де застосовуються і в чому їх принципова відміна, скільки їх треба знати і

т.д. Розробляючи свої перші програми на ТП, Ви повинні вміти побачити перспективи розвитку своїх зусиль в галузі розробки програмних систем. Іншою, на наш погляд, темою, що рідко висвітлюється, є проблема різноманіття типів додатків і комп'ютерних архітектур, в рамках яких вони будуть функціонувати після створення. А розділ про мову UML націлить Вас на освоєння уявлень цього важливого засобу моделювання процесу розробки програмних систем, що спираються на об'єктно-орієнтовані уявлення реального миру для організації свого мислення і діяльності в галузі проектування і розробки програмного забезпечення.

У розділах, які присвячені особливостям мови Турбо Паскаль і програмуванні на неї, головну увагу було зосереджено:

❶ на розумінні абстракцій, що містяться в уявленнях різних типів чисел, які не мають аналогів в інших сферах діяльності людини (наприклад, таке число: *-2.34e-12*);

❷ на взаємодію рівнів абстракцій при використанні логічних: ***значень, змінних, операцій, виразів*** и т.д.

❸ на алгоритмічних особливостях конструювання програм обчислення нескінченних (рос.-бесконечных) рядів и добутків (рос.-произведений);

❹ на унікальних можливостях дослідження функціонування програм і їх фрагментів засобами дебаггера середовища Турбо Паскаль;

❺ на особливостях практичного застосування в програмах складних типів даних: масивів, записів, файлів, множин (рос.-множеств), черг (рос.-очередей) і ряду інших.

У конці кожного розділу для закріплення знань по головним пунктам обговорення і самоконтролю наводяться питання і завдання. На наш погляд, корисно виконання не тільки завдань, але й прикладів з розділів, які присвячені мові Турбо Паскаль, оскільки в програмуванні важливо не тільки споглядати (рос.-созерцать) фрагменти коду, а пророблять все «своїми руками». Звичайно, на перших порах, не все у Вас буде виходити, але програміст мужніє саме у боротьбі з компілятором…

Декілька слів про Додатки. Сюди винесені питання перетворення чисел з десятинної системи у двоїчну, восьмеричну, шістнадцятирічну і назад (рос.-обратно). Тут також розміщена таблиця характеристик усіх найбільш відомих мов програмування. Окремо наведені визначення (рос.-определения) назв мереж різних рівнів, котрі повсюдно (рос.-повсеместно) застосовуються, але у такому поєднанні у відомих авторам джерелах не об'єднувалися. Декілька частин додатків відведені для довідкових відомостей про характеристики системи Турбо Паскаль (поєднаннях клавіш і типах даних, що застосовуються), а також таблиця кодів ASCII.

Особливо треба виділити глосарій, що є одним з головних компонентів підручника. Зібраний по крупицям з матеріалів різних і багаточисельних вітчизняних і закордонних джерел, він може також служити самостійним інструментом для подавання допомоги початківцям з ціллю проникнення у світ інформаційних технологій, що так стрімко розвиваються. Крім того, у ньому зібрано ряд рідких і/або багатозначних термінів, що углибляє уявлення про

комп’ютерний світ. Більш глибоке знайомство з глосарієм крім всього іншого, дозволить зайвий раз переконатися в глибокому проникненні англомовних фундаментальних інформаційних термінів не тільки в нашу мову, але й в наші абстрактні уявлення про світ, що нас оточує, а також про його прояви (про що заявляють навіть продвинуті світові спеціалісти)[5].

Завершуючи вступ, хочеться згадати слова архітектора і керівника розробки однієї з значних (рос.-крупнейших) програмних систем – операційної системи OS/360 для комп'ютерів фірми IBM – професора університету Северної Кароліни (США) Фредерика П. Брукса (рис. 9), автора твору, що став класикою, під назвою «Мифический человеко-месяц или как создаются программные комплексы[6]»: «Чому програмування доставляє задоволення (рос.-удовольствие)? Як винагороджуються всі зусилля професіонала?».

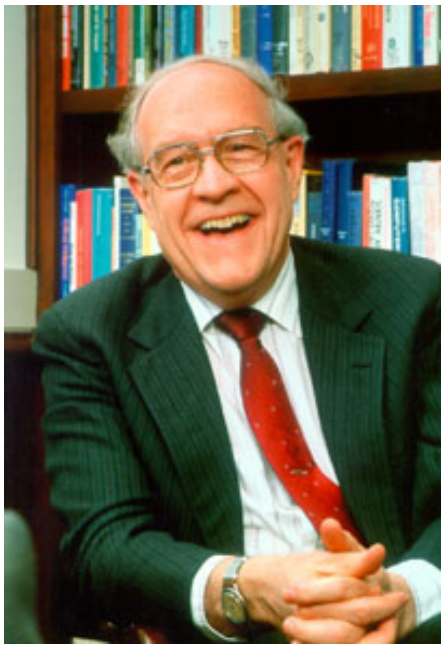


Рис. 9.
Фредерик П. Брукс

Тому, пише Ф.П. Брукс, що це задоволення працювати з дуже гнучким матеріалом. Програміст, як і поет, працює майже виключно головою. Він будує свої зáмки в повітрі і з повітря тільки силою своєї уяви (рос.-воображения). Дуже рідко матеріал для творчості допускає таку гнучкість, таку можливість таких частих покращень (рос.-улучшений) і переробок і такими простими засобами дозволяє здійснювати величезні задуми.

Матеріал поета – слова, і результат – ті ж слова; але на відміну від віршотворця (рос.-стихотворца), програміст створює програмний продукт, реальний у тому сенсі, що сам програміст двигається і працює, виробляючи видимий результат, відмінний (рос.-отличный) від нього самого. Він друкує результати, креслить рисунки, викликає звуки, управляє рухом руки. Чарівництво (рос.-волшебство) міфів і легенд стало явою (рос.-явью) у наші дні. Ви друкуєте на клавіатурі заклинання, і ось екран дисплея оживає, показуючи об’єкти, котрих не було і могло не бути ніколи!

Програмування завдає нам радість, тому що дозволяє задовольняти прагнення до творчості, яка глибоко закладена в кожному з нас, і розділити ці почуття радості з іншими».

Оскільки цей підручник є практичним керівництвом, у ньому міститься множина прикладів з покроковими інструкціями і докладними (рос.-подробными) вказівками. При цьому використовуються наступні погодження (рос.-соглашения).

---

[5] Terminology, and naming things in general, is always difficult. Web Services Architecture Working Group (http://www.w3.org/2002/ws/arch/). Дослівно: «Створення термінології і йменування сутностей взагалі і завжди протікають надзвичайно важко», Документ Робочої групи Архітектури Web-сервисів організації W3C, 2002 г.

[6] Тільки в Росії ця книга витримала дев’ять (!) видань з 1979 р. по 1999 р.

Слова, на яких ми хочемо акцентувати Вашу увагу, будуть виділятися ***жирним курсивом***.

Якщо необхідно натиснути клавішу **Ctrl** или **Alt** одночасно з якоюсь ще клавишей, назви цих клавиш в тексті об'єднуються знаком «плюс». Наприклад, «Компілюйте і запускайте свою програму, використовуючи **Ctrl+F9**». Такий же шрифт використовується для виділення елементів інтерфейсу інтегрованого середовища розробки Турбо Паскаль, наприклад, «**Save file as**».

Фрагменти коду програм і ідентифікатори у підручнику надруковані моноширінним шрифтом, наприклад «`Program MyFirst;`».

Список літератури до підручника розподілений по главам й розміщений наприкінці.

Література до Вступу

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. Ніклаус Вірт. Приветствие на открытии Международной конференции по преподаванию 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 C++, 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., Дэйкстра Э., Хоор К. Структурное программирование: Пер. с англ. –М.: Мир, 1975. –256 с.

16. Дейкстра 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.

18. Джонс Ж, Харроу К. Решение задач в системе Турбо Паскаль/Пер. с англ. Предисл. Ю.П. Широкого. – М.: Финансы и статистика,1991. – 720 с.

Марні такі наміри, що не збуджують прагнення до наслідування: доблесними справами люди не тільки захоплюються, але й бажають наслідувати тим, хто їх робить.

Плутарх (46 – 120 р.н.е.)

1. ТРИВАЛИЙ ШЛЯХ ДО ПЕРСОНАЛЬНОГО КОМП'ЮТЕРА

1.1 Джерела "інформаційного вибуху"

За початок відліку розвитку людської цивілізації як правило приймають час, коли люди почали створювати предмети праці і полювання. Таємниця "викрадення вогню" губиться у століттях, але вся наступна історія технічного прогресу від оволодіння вогнем до відкриття ядерної енергії - це історія послідовного підпорядкування людині все більш могутніх сил природи: тяглові тварини, вітряні і водяні двигуни, теплові двигуни, атомна енергетика. Задача, що стояла перед людиною, була гранично проста: множити різними інструментами і машинами свою мускульну силу. У той же час спроби створення інструментів, що посилюють природні можливості людини по обробці інформації, починаючи з камінчиків абака і до механічної машини Беббіджа вимірялися окремими сплесками людських доль, фактів, музейних механізмів і пристроїв, що наполегливо передвіщали небачений розвиток цієї галузі піднесення людини на нові вершини знань.

Рис. 1.1. Клінопис на глиняній табличці

На самих ранніх етапах формування трудових колективів для синхронізації виконуваних дій людині потрібні були кодовані сигнали спілкування, складність яких швидко зростала з підвищенням складності трудового процесу. Цю задачу людський мозок вирішував еволюційно, без будь-яких штучно створених інструментів: просто розвивалася і постійно удосконалювалася людська мова.

Мова і виявилася першим носієм людських знань, що накопичуються в усних розповідях і переказах, що передавалися з покоління в покоління. Першою технологічною підтримкою цього важливого процесу з'явилося створення писемності. Розпочатий тоді процес пошуку й удосконалювання носіїв інформації (а також інструментів для її реєстрації) продовжується і дотепер: камінь, кістка, дерево, глина (рис.1.1), папірус, шовк, папір, люмінофор, магнітні й оптичні носії інформації і т.д.

Одночасно з розвитком процесу накопичення знань

у людському суспільстві йшов процес формування відособленої професійної групи, для якої спочатку основним, а потім і єдиним "службовим заняттям" стає робота з інформацією. Цей живий бар'єр почав руйнуватися тільки після винаходу друкарства й оволодіння суспільством новою технологією - ***грамотністю***. У середині XV в. майже одночасно І. Гутенберг (рис.1.2) у Німеччині і І. Федоров у Росії винайшли друкувальний верстат, що прискорив темпи накопичування професійних знань.

Рис. 1.2. Винахідник першого друкувального станка І. Гутенберг (1397-1468)

По підрахунках наукознавців, загальна сума людських знань за останні 200 років змінювалася дуже повільно: до 1800 р. вона подвоювалася кожні 50 років, до 1950 р. подвоювалася кожні 10 років, а до 1970 р. - кожні 5 років. Явище різкого збільшення знань у 80-х–90-х роках, що одержав назву "інформаційний вибух", указується серед симптомів, що свідчать про початок століття інформатизації і що включають:

❶ швидке скорочення часу подвоєння обсягу накопичених наукових знань;

❷ перевищення рівня матеріальних витрат на збереження, передачу і переробку інформації над рівнем аналогічних витрат на енергетику;

І якщо на початку XX століття економічна міць держав вимірялася ***матеріальними ресурсами*** то в його кінці вперше в історії людства основним ***предметом праці*** в промислово розвитих країнах стає ***ІНФОРМАЦІЯ***. Інструментом для роботи з новим предметом праці і став персональний комп'ютер, що поклав початок освоєнню суспільством нової технології – ***комп'ютерної грамотності.***

1.2. Комп'ютер: від ідеї – до реалізації

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

Існує безліч версій історії появи персонального комп'ютера. Інакше і бути не могло. Адже шлях до нього вмістив у себе масу пошуків і відкриттів протягом багатьох століть. І все ж таки, спробуємо коротко зупинитися на деяких віхах зародження комп'ютера.

З усіх винахідників рахункових машин минулого, напевно, ближче усього до формулювання основних принципів побудови й організації обчислювального процесу в комп'ютері, в сучасному його розумінні, підійшов англієць Чарльз Беббідж (рис. 1.3). У свій час він прославився не тільки гостротою розуму, але і своїми дивацтвами. Беббідж був одним із засновників Королівського астрономічного товариства, автором усіляких творів на всілякі теми – від політики і математики до технології виробництва. У впродовж 13 років цей ексцентричний геній завідував

кафедрою математики у Кембріджському університеті, хоча жодного разу не з'явився в ньому і не прочитав там жодної лекції.

Найвищим досягненням Чарльза Беббіджа була розробка принципів дії і складу комп'ютера, за ціле сторіччя до того, як з'явилася технічна можливість його реалізації. Один з творців першого американського комп'ютера "Марк-1" молодий гарвардський математик Говард Ейкен, говорив у 1943 році, що для нього ознайомлення з описом Аналітичної машини, залишеного самим Беббиджем, виявилося більш ніж досить. "Якби Беббідж жив на 75 років пізніше періоду своєї діяльності, – заявив він, – я б залишився без роботи".

Рис. 1.4. Перша жінка-програміст, графіня Ада Лавлейс

Енергія, авторитет і уміння переконувати дозволили Чарльзу Беббіджу домогтися в 1822 році державного фінансування розробки і створення «Різницевої (рос.-разностной) машини», призначеної для розрахунків і друку великих математичних таблиць. Здійсненню цього проекту усі свої неабиякі математичні і літературні здібності присвячувала уроджена Огаста Ада Байрон графиня Лавлейс (рис. 1.4). Говорячи про «Аналітичну машину», Беббідж відзначав: "…графиня Лавлейс, очевидно, розуміє її краще за мене, а пояснює її пристрій у багато раз краще". Графиня Лавлейс допомагала Беббіджу проясняти його власні ідеї, надихала його, глибоко цікавлячи його роботою і заражаючи своїм ентузіазмом. Вона прекрасно зрозуміла революційну сутність розроблювальної машини і те, що це дійсно був «математичний верстат», споконвічно мов би бездумний, але здатний виконати будь-як програму, переведену на мову перфокарт. Саме тому, на її честь, була названа одна з найвідоміших і понині мов програмування реального часу - ADA.

Незважаючи на те, що результат не був досягнутий, у процесі роботи Ч. Беббідж прийшов до ідеї створення ще більш могутньої машини. Його нове творіння – «Аналітична машина», на відміну від своєї попередниці, повинна була не просто вирішувати математичні задачі вказаного типу, але й виконувати різноманітні обчислювальні операції відповідно до інструкцій, що задаються оператором. За задумом це була "машина самого універсального характеру" – в дійсності не що інше, як перший універсальний програмований комп'ютер. На той час, коли в 1833 році витрати на створення «Різницевої машини» досягли 17 000 фунтів стерлінгів, терпіння британських чиновників лопнуло і державне фінансування було припинено. Адже якщо «Різницева машина» мала сумнівні шанси на успіх, то «Аналітична» – й зовсім виглядала фантастикою. Розміром з залізничний локомотив, конструкція в проекті являла собою значне накопичення сталевих, мідних і дерев'яних деталей,

годинникових механізмів, що приводяться в дію паровим двигуном. Найменша нестабільність якої-небудь малюсінької деталі приводила до сторазово посилених порушень в інших частинах механізму, що по суті робило нереальним реалізацію цього проекту. Адже, як відомо, для реалізації науково-технічної ідеї потрібно виконання принаймні трьох основних умов:

1. Ідея не повинна суперечити відомим законам науки.
2. У її реалізації повинна бути гостро зацікавлена значна частина суспільства (іншими словами, потрібно "дозріти" соціальному замовлення).
3. Повинний бути досягнуть такий рівень технології і суспільного виробництва, що забезпечує ефективну реалізацію не тільки конкретних елементів, але й інших закладених в ідею технічних принципів.

Лише в середині XIX в. з'являються відкриття й винаходи, що наближають народження комп'ютера. Вільям Остін Барт першим в Америці (1829 р.) одержує патент на незграбну, але працездатну друкарську машинку, що не викликала тоді інтересу в промисловців. А знайома зараз будь-якому користувачу комп'ютерна клавіатура з розкладкою QWERTY (так на ній розташовані перші шість латинських літер), з'явилася лише в 1874 році. Але саме з неї почався переможний хід цього гучного знаряддя праці по столах секретарок малих і великих організацій. До речі, клавішу Shift, для переключення верхнього і нижнього регістрів, додали до друкарських машинок тільки в 1878 році, а до того часу заголовні літери розташовувалися на клавіатурі окремо.

Рис. 1.5. Електронна лампа

Експериментуючи з електричною лампою в 1883 р., Едісон вводить у вакуумний балон додатковий платиновий електрод, подає напругу і, до свого подиву, виявляє, що між електродом і вугільною ниткою протікає струм. Оскільки в той момент головною метою Едісона було продовження терміну служби лампи накалювання, цей результат його зацікавив мало, але патент заповзятливий американець все ж-таки одержав. Явище, відоме нам як термоелектронна емісія, тоді одержало назву "ефект Едісона" і на якийсь час забулося. В електронну лампу його ідея матеріалізувалася пізніше (рис. 1.5).

Соціальне замовлення на комп'ютери дозріло у розпал другої світової війни – були потрібні складні артилерійські балістичні розрахунки. І от, із благословення командування військово-морського флоту США, при фінансовій і технічній підтримці фірми IBM, почалася розробка обчислювальної машини, в основу якої лягли неперевірені ідеї XIX в. і надійні технології XX в.

На початку 1943 року розпочало свою роботу творіння Говарда Ейкена – обчислювальна машина "Марк-1" (рис. 1.6), що досягало у довжину майже 17 м и в висоту більш 2,5 м. та містило 750 000 деталей, з'єднаних проводами загальною довжиною близько 800 км. "Марк-1" міг "перемелювати" числа довжиною до 23 розрядів (рис. 1.7). На операції складання й віднімання витрачалося 0,3 з, а на множення 3 с. За день машина виконувала обчислення,

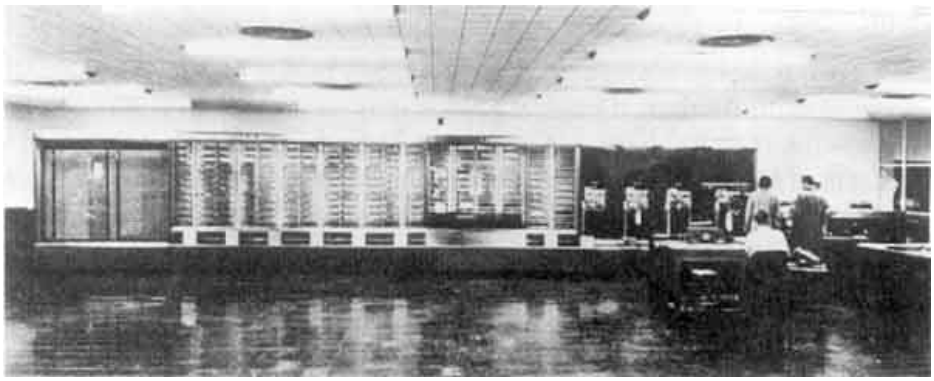


Рис. 1.6. Машина "Марк-1"

на котрі раніш йшло півроку (!). У ній, як і в її попередницях, використовувались прості електромеханічні перемикачі (реле), які широко застосовувались в той час у телефонному зв'язку, а команди машині у виді програм набивалися на перфострічці (рос.-перфоленте) (рис. 1.8).

24.712946369421639419437

Рис. 1.7. Двадцятитрьохрозрядне дрібне (рос.-дробное) десятичне число (тобто число, що містить у собі двадцать три десятичні цифри)

Війна продовжувалася, військові розробки вимагали прискорення і за справу взялися співробітники обчислювального центру Пенсільванського університету Джон У. Мочлі і Джон Преспер Екерт. Їхні зусилля не залишилися непоміченими і 9 квітня 1943 року армія США уклала з університетом, де вони працювали, контракт на суму в 400 000 дол., який перевищив вартість створення лампового комп'ютера "Еніак" (рис. 1.9).

Рис. 1.8. Перфострічка

Наприкінці 1945 року 30-тонний гігант був зібраний і продемонстрований представникам преси. По своїх розмірах (близько 6 м у висоту і 26 м у довжину) цей комп'ютер більш ніж удвічі перевершував по швидкодії "Марко-1" Говарда Ейкена і коштував 500 000 дол. Одночасно працювали 17 000 електронних ламп збільшили його швидкість в 1 000 разів! Це був прорив.

Подальше удосконалювання комп'ютерної техніки пішло по шляху підвищення надійності роботи, мініатюризації електронних пристроїв та скорочення енерговитрат. Інженери з фірми Bell Labs Вільям Шоклі, Джон Бардін і Уолтер Бреттен у 1947 році винайшли транзистор (рис. 1.10). Замітка в New York Times про цю подію була поміщена 1 липня 1948 року в самому кінці невеликого розділу "Новини радіо" поруч з оголошенням про час трансляції передачі "У ритмі вальсу".

Після майже трирічних досліджень, що коштували більше мільйону доларів, фірма Bell Labs отримала перший напівпровідниковий підсилювач (рос.-усилитель), який, до речі, не зробив особливого враження на представників засобів масової інформації.

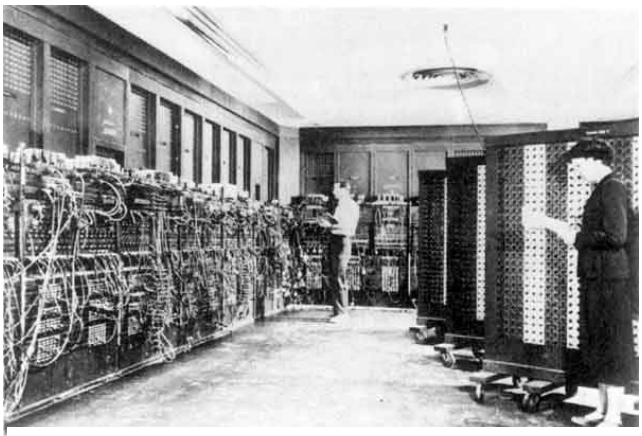


Рис. 1.9. Загальний вид машини "Еніак", 1946 р.

З появою транзисторів лампові комп'ютери морально застаріли. У 1959 році фірма IBM створила свій перший цілком транзисторний великий універсальний комп'ютер (мейнфрейм), здатний виконувати 229 тис. арифметичних операцій у секунду. Такі мейнфрейми дозволили військово-повітряним силам США створити систему раннього попередження про напад балістичних ракет. У 1964 році на основі двох мейнфреймів моделі 7090 американська авіакомпанія SABRE уперше застосувала автоматизовану систему продажу і бронювання авіаквитків у 65 містах світу.

Перші германієві транзистори коштували до 8 дол. за штуку, у той час як ціна лампи не перевищував 75 центів, тому що германій був і залишається досить рідким елементом, котрий отримують за особливою технологією і по собівартості він дорожче золота. Однак у 1954 р. американський фізик Гордон Тил виготовив транзистор з кремнієвого кристала замість германієвого. Це був черговий крок вперед, тому що кремній - найпоширеніший на Землі (після кисню) хімічний елемент. Удосконалювання технологій виробництва транзисторів теж сприяло зниженню їх вартості.

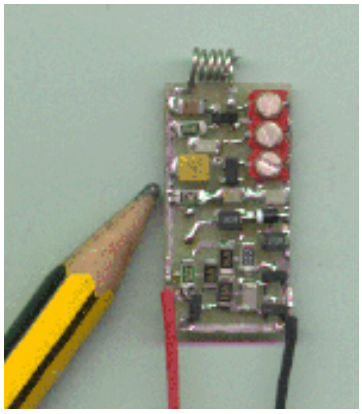


Рис. 1.10. Плата з транзисторами, 1947 р.

Та інтенсивні дослідження продовжувалися. Геніальна ідея спала на думку уродженцю Канзасу Джеку Кілбі. Рішення виникло в липні 1956 р., коли співробітники компанії Texas Instruments відбули в двотижневу літню відпустку, а він як новачок, що не заслужив на це права, залишився в лабораторії практично один. Кілбі зрозумів, що елементи електричних схем – резистори і конденсатори можна робити не тільки з того ж матеріалу, що і транзистори, але й виготовляти усі компоненти на одній напівпровідниковій пластині. Через кілька місяців тривалих роз'яснень він переконав у правильності своєї ідеї скептично налаштованого шефа, виготовивши перший експериментальний зразок. Перша у світі інтегральна схема (ІС) уявляла собою тонку пластинку з германію, площею в 1 $см^2$. Фірма повідомила про народження нового пристрою в січні 1959 р. А щоб продемонструвати потенційні можливості нової техніки, компанія побудувала для військово-повітряних сил США комп'ютер обсягом близько 40 $см^3$ на 587 ІС. Його розміри виявилися в 150 разів менше, ніж у машин старого зразка.

1.3. Стрибок у розвитку обчислювальної техніки

Наприкінці 60-х років велика група талановитих інженерів - електронників, залишивши фірму Fairchild Semiconductors, створили більш 50-ти компаній для розробки і створення інтегральних схем. На той час ІС, незабаром прозвані "чипами", уявляли кристали кремнію, що містять усі необхідні для роботи електронних схем компоненти: транзистори, резистори і конденсатори. В міру зменшення розмірів окремих компонентів на кристалі, кількість їх на одному чипі зростала з неймовірною для такого роду системних елементів швидкістю, приблизно подвоюючись щороку. Наприклад, у 1964 році на кристалі розміром 7 $см^2$ уміщалося 10 транзисторів і інших компонентів, а до 1970 р. у кристалі того ж розміру містилося вже не менш 100 елементів приблизно при тій же вартості ІС.

Рис. 1.11. Засновники фірми Intel

Тому не дивно, що бізнес-план обсягом в одну сторінку, що власноручно надрукував Боб Нойс у 1969 р. на "текстовому процесорі" того часу – друкарській машинці – викликав справжній інтерес в Артура Долі, фінансиста із Сан-Франциско. Покладаючись на найвищу репутацію засновників фірми Intel Енді Гроува, Гордона Мура і Роберта Нойса (рис. 1.11) і їхні глибокі знання, містер Рок протягом двох днів надав їм необхідні гроші – ні багато, ні мало – два з половиною мільйона доларів!

Корпорація Intel починала з виготовлення мікросхем оперативної пам'яті. Першим серійним виробом Intel була "мікросхема 3101" – 64-розрядна статична оперативна пам'ять. Цей виріб мав деякий успіх. Однак треба визнати: те особливе місце, що Intel зайняла в комп'ютерній індустрії нинішнього часу пов'язано з зовсім іншими мікросхемами (у 2001 році в щорічно надрукованому списку 500 найбільших компаній США американського журналу Fortune компанія Intel займала 41 місце з прибутком 33 млрд. доларів).

Все почалося з однієї чудової ідеї інженера Intel по імені Тед Хофф. Замість дванадцяти спеціалізованих мікросхем, замовлених японською фірмою Busicom для кишенькових калькуляторів він вирішив створити одну універсальну, яка змогла б їх замінити. Як виявилося, це рішення змінило історію, дозволивши зробити програмувальний інтелектуальний процесор настільки дешевим, що його можна було надалі застосовувати навіть для побутових цілей, і настільки універсальним, що кожний міг сподіватися мати свій індивідуальний комп'ютер.

Через 9 місяців напруженої роботи в 1971 році з'явився перший у світі мікропроцесор – Intel 4004 (рис. 1.12). У ньому містилося 2 300 напівпровідникових транзисторів, і він вільно уміщався на долоні. Малюсінький процесор 4004 мав таку ж швидкодію, як величезний комп'ютер

Еніак, що займає обсяг 85 кубічних метрів і що мав 17 000 вакуумних ламп (порівняйте рисунок 1.9 і рисунок 1.12).

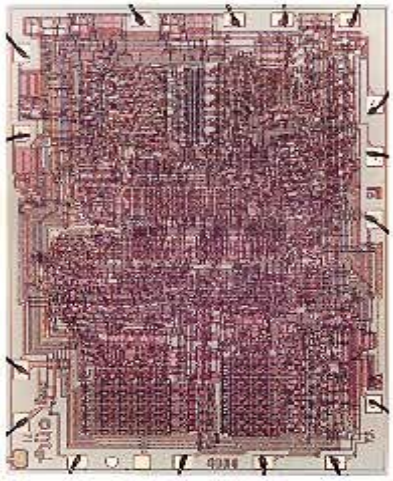


Рис. 1.12.
Зовнішний вид ІС
Intel 4004

У цей момент керівництво Intel зрозуміло, що зроблено видатний винахід. Але, до нещастя, авторські права на цей винахід належали не їм, а фірмі Busicom (Японія), що замовила й оплатила витрати на розробку. У результаті переговорів Intel викупила в Busicom усі права на мікропроцесор 4004 за 60 тисяч доларів. "Тепер питання полягало в тому, чи зможемо ми продавати процесори кому-небудь, крім Busicom і чи знайдеться досить багато інших застосувань для цих мікросхем, щоб вони стали приносити помітний прибуток", – згадував згодом Роберт Нойс

Фірмі Intel довелося взятися за непросту задачу: роз'яснити інженерному світу величезний потенціал використання програмувальних мікропроцесорів у безлічі областей. Компанії довелося проводити семінари для інженерів, публікувати рекламні оголошення і продавати довідкові посібники з мікропроцесорів. Один з співробітників Intel згадує: "Бували тижні, за які Intel продавала більше довідкової документації, чим самих мікропроцесорів". Одночасно вдосконалювалася конструкція самого чипа.

Рис. 1.13.
Перший ПК Altair

У 1974 році з'явилася нова модель – Intel 8008. Потужність цього процесора, у порівнянні з його попередником, зросла вдвічі. За повідомленням журналу Radio Electronics, відомий ентузіаст обчислювальних технологій Дон Ланкастер застосував процесор 8008 у розробці прототипу персонального комп'ютера – пристрою, що згаданий журнал назвав "гібридом телевізора і друкарської машинки". Використовувалося воно як термінал введення-отримання інформації.

І тільки наступний крок виявився самим вдалим. Процесор 8080 1974 р. випуску став "мозком" першого персонального комп'ютера (ПК) Altair (рис. 1.13), названого по імені зірки Альтаїр, до якої був запущений міжпланетний корабель Ентерпрайз з американського телесеріалу "Космічна одисея". По нинішніх мірках його параметри були сміховинні – оперативна пам'ять всього 256 байт, але вже десятки тисяч екземплярів комплекту для самостійної збірки Altair, за ціною $397, розійшлися усього за кілька місяців. І хоча його можливості були досить обмежені, на тільки що відкритому ринку ПК вперше утворився дефіцит. З розумінь економії багато покупців купували комп'ютер у

виді набору деталей, а потім збирали його власними силами. Щоб зібраний у такий спосіб комп'ютер був працездатний, від його власника були потрібні чималі пізнання і практичні навички в електроніці. Більш того, машина не мала ні клавіатури, ні екрану.

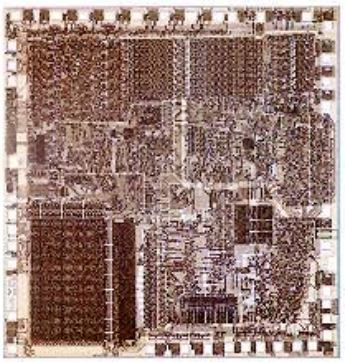


Рис. 1.14.
Intel 8088

Користувачі вводили програми та дані в двоїчній формі, клацаючи набором маленьких ключів, що могли займати два положення – верхнє та нижнє (що відповідало двоїчним кодам 1 то 0). Результати зчитували також у двоїчних кодах – по світлим та темним лампочкам. Більшість покупців замовляли комп'ютерний набір поштою, довідавшись про нього лише з журнальної статті. Їх не лякали технічні труднощі та недоліки Altair. Ентузіасти писали власні програми для машини та доповнювали її різними пристроями.

1978 рік ознаменувався для фірми Intel випуском процесорів-бестселерів – Intel 8086 і 8088 (рис. 1.14). На той час, створена в 1924 р. Германом Холлерітом фірма IBM (International Business Machines), займалася виробництвом великих обчислювальних машин для економічних і наукових розрахунків. У 1969-71 роках комп'ютери IBM забезпечували висадження американських астронавтів на Місяць. У 1973 році фірма виконала замовлення NASA на постачання комп'ютерного устаткування для програми «Союз-Аполлон». Згодом IBM взяв участь і в програмі польотів космічних човників «Шаттл». До кінця 1960-х років IBM панувала на ринку електронних обчислювальних машин, де обсяг збуту її продукції перевищував 3 млрд. доларів щорічно. У 2001 році в списку журналу ″Fortune″ 500 найбільших фірм США IBM займала почесне 8-і місце з доходом 88 млрд. дол. у рік.

Рис. 1.15. IBM PC XT,
1981 р.

У 1980 році керівництво IBM звернуло увагу на ринок мікрокомп'ютерів, що тільки но зароджувався, і прийняло революційне рішення про створення власної моделі персонального комп'ютера. При конструюванні був застосований принцип відкритої архітектури: складові частини були універсальними, що дозволяло модернізувати комп'ютер вроздріб. З метою зменшення витрат на створення персонального комп'ютера IBM використовувала для свого дітища розробки інших фірм, зокрема, програмне забезпечення фірми Microsoft і мікропроцесор фірми Intel. "Мозком" знову створеного ″хіта″ сезону – IBM PC XT (рис. 1.15) став процесор 8088, велику партію яких, придбав у Intel спеціально створений підрозділ корпорації IBM по розробці та виробництву у цей період надзвичайно популярної марки персональних комп'ютерів. Параметри нового ПК були вже досить значні – операційна система MS-DOS 1.0. фірми Microsoft, процесор Intel 8088 з

частотою 4,77 Мгц, 64 Кбайт оперативної пам’яті, розширюваної до 640 Кбайт. Поява IBM PC у 1981 році породила лавинний зріст попиту на персональні комп'ютери, що стали від того часу знаряддям праці людей самих різних професій. Поряд з цим виник гігантський попит на програмне забезпечення і комп'ютерну периферію. На цій хвилі виникли сотні нових фірм, що зайняли свої ніші комп'ютерного ринку.

Рис. 1.16.

Стив Джобс, засновник Apple Computer

З цього моменту почався переможний хід ПК по усьому світі. З'явилася безліч клонів (IBM-подібних комп'ютерів інших фірм-виробників). Їх характеризувала програмна й апаратна сумісність з IBM PC. Деякі з учасників марафону і нині продовжують розпочату на початку 80-х нескінченну гонку. У першій сотні найбільш щасливих, крім IBM, у 2001 році знаходилися: Hewlett-Packard (19 місце, прибуток 48 млрд. дол.), Compaq Computer (27 місце, прибуток 42 млрд. дол.), Lucent Technologies (28-е місце, прибуток 41 млрд. дол.), Motorola (34-е місце, прибуток 37 млрд. дол.), Dell Computer (48-е місце, прибуток 31 млрд. дол.), Microsoft (79-е місце, прибуток 22 млрд. дол.).

Розвиток обчислювальної техніки та інформаційних технологій вивело комп'ютери з великих машинних залів на робочі столи користувачів, зробивши їх предметом першої необхідності практично у всіх сферах праці – знаряддям для спрощення рутинних операцій і одержання нової інформації.

Цьому сприяло відкриття «феномена персонального комп'ютера і персональних обчислень», що у США зв'язують з ім'ям Стіва Джобса – засновника і керівника фірми Apple Computer (рис. 1.16). У 1980 році Джобс визначив цей тип електронної обчислювальної машини (ЕОМ), як індивідуальний інструмент для посилення природних можливостей людського розуму. У середині 1981 року Джобс сформулював концепцію ПК за допомогою простої аналогії. "Один раз, – писав він, – мені довелось розглядати список біологічних видів, розташованих за рівнем ефективності, з яким вони використовують свою мускульну енергію для пересування. На першому місці по ефективності в цьому списку знаходиться кондор (рис.1.17) – він майже не ворушить крилами при польоті, а людина перебуває в нижній третині цього великого списку.

У той же час, з досліджень біологів відомо, що людина, яка їде на велосипеді, по ефективності використання мускульної енергії набагато перевершує усіх відомих тварин, включаючи кондора". Іншими словами, персональний комп'ютер виконує для людини ті ж функції підвищення ефективності, що і велосипед, але в іншій, розумовій сфері людських можливостей, дозволяючи помітно збільшити ефективність його інтелектуальної діяльності.

Рис. 1.17
Кондор у польоті

З усього цього Стив Джобс робить висновок, що основне призначення ПК полягає в тім, щоб звільнити людини від гніта рутинної обробки інформації, залишаючи йому "…робити те, що він може робити значно краще, ніж кожний зі створених їм приладів***: концептуально мислити".***

Випускник вузу на виробництві, для рішення поставлених перед ним задач, як правило, виходить "один на один" з персональним комп'ютером. І тут від нього потрібно глибоке оволодіння комп'ютерною грамотністю, що базується на знаннях не тільки взаємодії програмних та апаратних засобів самого ПК (рис. 1.18), але й останнього з навколишнім його реальним світом, без чого неможливо очікувати максимальної віддачі від настільки корисного і наукомісткого пристрою за назвою "персональний комп'ютер".

Рис. 1.18. ПК фірми Apple
″зразку″ 2003 року.

З неймовірною швидкістю змінюються технології і концепції комплексування комп'ютерів і їхніх елементів, але загальні підходи в конструюванні моделей даних і програмних компонентів вже сформувалися.

Тому важливо засвоїти фундаментальні уявлення про інформаційні процеси в комп'ютерних системах для подальшого успішного застосування інформаційних технологій у повсякденній практиці кожного фахівця.

1.4. Розвиток операційних систем для персонального комп'ютера

Коли в 1981 році був випущений перший промисловий варіант ПК, ніхто не міг припустити, що до 2000 року 86% усіх ПК в усьому світі будуть працювати під керуванням операційних систем фірми Microsoft, а сама фірма – у 2001 році в списку журналу Fortune розташується на 79 місці серед 500 інших. Цьому сприяло щорічне збільшення її доходів і прибутку на 30%, що, після сплати всіх податків, у 2002 р. складало 250 000 дол. у рік на одного співробітника (у середньому в американській промисловості ця цифра не перебільшує 17 000 дол. на рік на одного робітника).

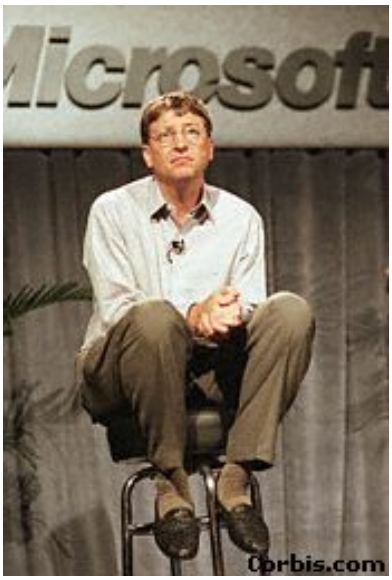


Рис. 1.19 Засновник Microsoft Білл Гейтс

Як стверджує один з численних життєписів, справа починалося так...

Вільям Генрі Гейтс III (рис. 1.19) народився в 1955 році в Сіетлі (шт. Вашингтон, США) у родині юриста і шкільної вчительки. Батьки віддали Білла у престижну приватну школу, де відкрилися його схильності до математики і неабиякі здібності до програмування.

Там же відбулося знайомство з Полом Аллєном – виявилося, що обидва вони захоплювалися на той час науковою фантастикою. Трохи пізніше в місцевому обчислювальному центрі утворився дитячий кружок, у який негайно записалися Пол та Білл. Інтерес до програмування потягнув за собою звичку витрачати весь вільний час (а як парвило і невільний) на збагнення програмістських премудростей. Після переходу у восьмий клас, Білл уперше приступив до створення програм для своєї рідної школи, за що остання платила машинним часом, який, у свою чергу, використовувався для нових програмних розробок.

Останній клас містер Гейтс акуратно поділяв поміж школою і роботою програмістом у аерокосмічній корпорації TRW, де отримував 20 тисяч доларів щорічно.

У 1973 році свої обійми юному Біллові відкрив Гарвард; куди незабаром з Вашингтонського університету перейшов і Пол Аллєн.

Якось раз, заволодівши журналом Popular Elerctonics, Гейтс і Аллєн вивудили відтіля статтю про настільний комп'ютер Altair. Його виробник компанія MITS з м. Альбукерка (шт. Нью-Мексико, США), бажала придбати мову програмування, що дозволяла б фахівцям з ним працювати. Пол і Білл негайно зв'язалися з MITS та повідали, що на даний момент мають готову версію мови BASIC, якого, скажемо відверто, у них не було й у спомині. Після домовленості про зустріч, на все про все Білл та Пол одержали у своє розпорядження три тижні, по закінченні яких повинен був відбутися очний контакт із представниками MITS. Тепер залишалася суща дрібниця – скласти обіцяний інтерпретатор, не маючи на руках самого комп'ютера Altair. Однак вихід був знайдений негайно! На великому комп'ютері, що мався під руками, була емульована система команд новинки - процесора Intel 8080. Отриманий у такий спосіб інтерпретатор мови BASIC Білл Гейтс та Пол Аллєн продали MITS. Праця компаньйонів була витрачена не даремно. Адже в продажі кожна копія програми від MITS коштувала "усього" 500 дол. (рис. 1.20).

Успіх окрилив Гейтса та у 1975 році він достроково розстався з гостинними стінами Гарварда, не потрудившись закінчити навчання й одержати диплом. Удвох з Полом Аллєном вони перекочували в штат Нью-Мексико, де і була 4-го квітня 1975 р. зареєстрована компанія Microsoft (Microcomputer Software). Спочатку Пол наполягав на спеціалізації в області апаратного

забезпечення, однак Білл запропонував зробити упор на розробку програм, не без підстави покладаючись на успіх програмного забезпечення в комп'ютерній індустрії.

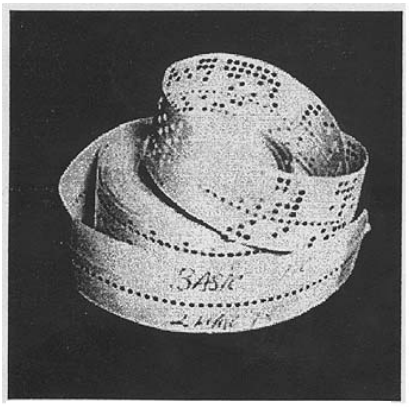


Рис. 1.20
Перфострічка з кодом інтерпретатора мови BASIC, написаний Б. Гейтсом і П. Аллєном для ПК Altair 8800 у 1975 р.

Спонукуваний природними амбіціями, Гейтс приступив до набору гідної команди. Він планомірно об'їжджав коледжі й університети в пошуках обдарованих студентів, чесно попереджаючи про виснажливу роботу, гарантуючи при цьому повну волю творчості. Ентузіасти потраплялися вкрай рідко, але їх усе-ж таки вистачило, щоб набрати необхідний склад. Спочатку справи у фірми йшли не дуже гарно. Компанія не могла дозволити собі найом менеджера по збуті продукції – цим займалася мати Білла, яка без найменшої зніяковілості пропонувала програми Microsoft таким величезним корпораціям, як IBM і AT&T.

Мало помалу, ліцензії на удосконалений BASIC стали купувати такі фірми як DTS, General Electric, NCR, Citibank, Radio Shack, Apple і навіть IBM. Не дивно, що 4 квітня 1979 р. BASIC 8080 став першим офіційно відміченим програмним продуктом фірми й одержав премію ICP (American College of Physicians) у 1 мільйон доларів. Ця нагорода, вручена Полу Аллєну, з'явилася заслуженою нагородою Microsoft за нелегку працю на ниві створення програмного забезпечення. Прояснився новий стратегічний напрямок і 18 червня 1979 р. Microsoft представляє новий BASIC для комп'ютерних систем на мікропроцесорі 8086. Це була перша мова програмування високого рівня, що з'явився для 16-бітних машин.

І не дивно, що вибір «блакитного гіганта» (так звуть IBM за колір її торгівельної марки) на розробку базової операційної системи (ОС) для усіх своїх майбутніх ПК упав саме на Microsoft. І це не дивно, бо ця складова комп'ютера є однією з найскладніших і важливіших (рис. 1.21).

Задача для Microsoft була непростою: до того моменту компанія розробляла і поставляла тільки компілятори та інтерпретатори для мов програмування і нараховувала всього 39 чоловік. Природно, що придатної операційної системи в Білла ще не було. Перше, що прийшло йому в голову – це порекомендувати IBM звернутися до суперника Microsoft, фірми Digital Research, що вже мала у своєму розпорядженні досить популярну операційну систему CP/M, установлену на багатьох 8-розрядних комп'ютерах.

Однак, недовго поміркувавши, Гейтс відразу зреагував і керівництву IBM була спрямована ціла дисертація про необхідність переходу на більш могутній для тих часів 16-рязрядний процесор 8080 від Intel. Не має сумнівів, що текст цієї пояснювальної записки виявився вкрай переконливим, унаслідок чого

CP/M була відкинута, і договір на розробку нової операційної системи дістався Microsoft.

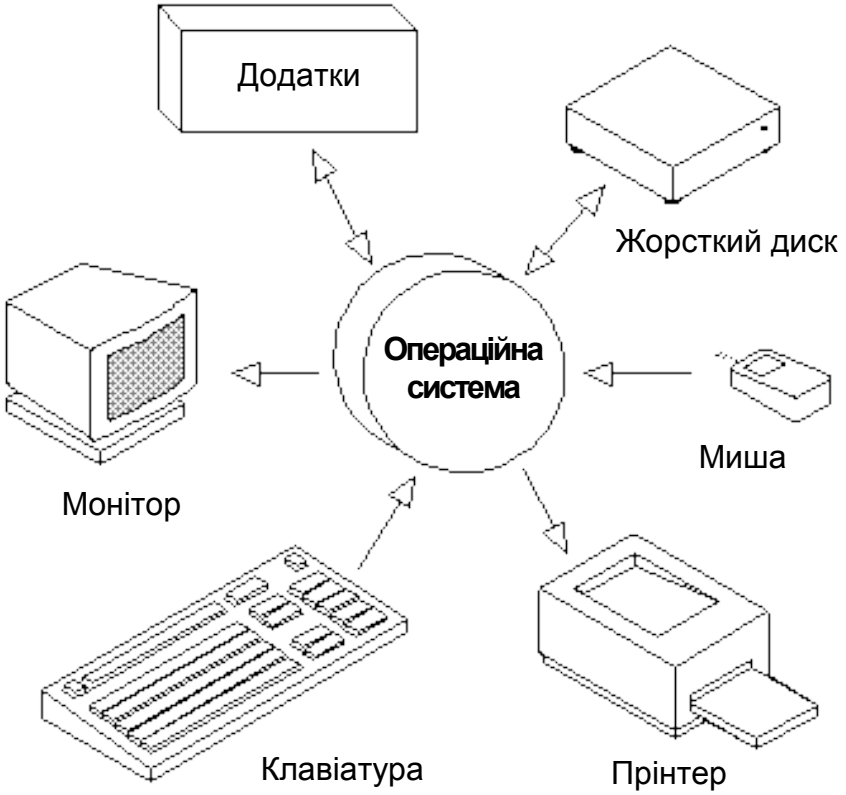


Рис. 1.21. Області застосування управлінських
″виконавчих обов'язків″ сучасної операційної системи

В умовах стиснутих донезмоги термінів Пол Аллєн установив контакт із маленькою компанією Seattle Computer Products, що мала придатну операційну систему 86-DOS, яка потребувала серйозній оптимізації для установки на IBM PC, випуск якого повинний був відбутися вже через місяць. Microsoft зробила мудро, не тільки купивши операційну систему 86-DOS, але і запросивши на роботу її творця Тіма Паттерсона. Гейтс та компанія активно доводили до розуму викуплену ОС, працюючи «по 25 годин на добу», відкіля пішла відома традиція зберігати в шафах фірми спальні мішки для цілодобово працюючих програмістів.

У результаті 2 серпня 1981 року IBM представила свій ПК, у якому використовувався комплекс програмних продуктів від Microsoft Inc: 16-бітна операційна система MS-DOS 1.0 і мови програмування BASIC, COBOL і Pascal.

Почався швидкий ріст кількості інсталяцій (установок) MS-DOS. За перші 16 місяців ліцензію у Microsoft купили 50 виробників подібних комп'ютерів. Одночасно ріс рівень виробництва ПК і в інших фірм. І хоча IBM зберігало лідерство, монополістом вона уже вважатися не могла. Були утворені і почали активно розвиватися Compaq, Hewlett-Packard, Texas Instruments.

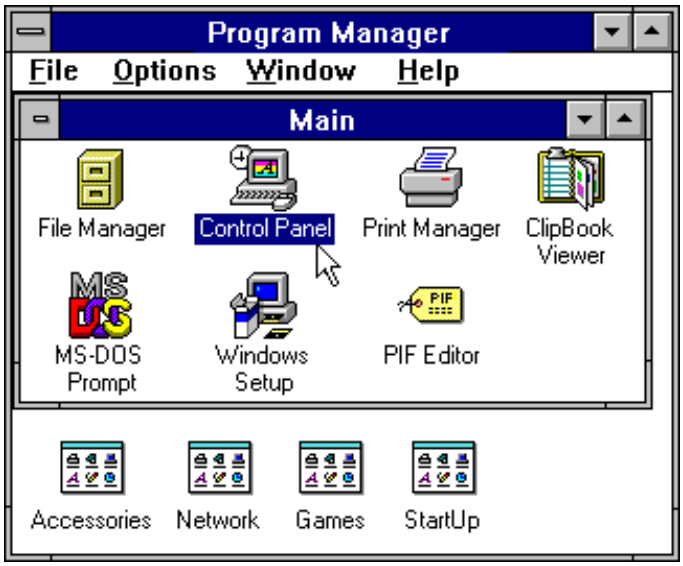


Рис. 1.22. Інтерфейс першої графічної ОС Windows 3.1

І в цей час, уловивши актуальність моменту, Білл Гейтс зробив геніальний хід, призвавши ведучих виробників ПК створювати продукцію, розраховану на повну сумісність з машинами IBM для того, щоб будь-яка програма, написана для ПК IBM, могла бути використана для інших комп'ютерів. Талант переконання Гейтса привів до того, що MS-DOS стали скуповувати оптом і вроздріб, ледве не встаючи за нею в чергу – на 80% усіх ПК була встановлена саме MS-DOS, завдяки чому і виникло поняття "IBM-сумісність" комп'ютерів.

10 вересня 1983 року Microsoft вперше анонсує Windows, що з'явився як графічний інтерфейс операційної системи для MS-DOS, що представляє собою універсальне операційне середовище для роботи з прикладними програмами (рис. 1.22).

У 1990 р. була укладена угода між Microsoft і Intel по створенню стандарту Wintel (абревіатура Windows+Intel) (рис. 1.23), що став популярним завдяки ПК на базі чипів від Intel, що працюють під керуванням ОС Microsoft Windows.

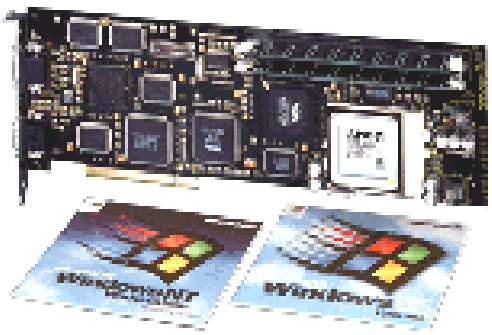


Рис. 1.23. Поєднання технологій Microsoft та Intel

До середини 2003 р. операційні системи Windows-9x/NT/200x були встановлені на 91% комп'ютерів в усьому світі.

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.

По даним аналітиків Ovum, п'ятірка лідерів продаж програмного забезпечення, у 2003 р. виглядала наступним чином: Microsoft ($25,9 млрд.), IBM ($13,1 млрд.), Oracle ($6,9 млрд.), SAP ($6,8 млрд.) і Hewlett-Packard ($2,6 млрд.).

Розуміючи велике значення розвитку інформаційних технологій деякі країни починають крокувати у цьому напрямі. Так, у порівнянні з вищенаведеними цифрами, по даним тих же аналітиків у 2005 р.

передбачається, що експорт програмного забезпечення з Індії буде перевищувати $35 млрд., що складатиме 5–7% усього її сукупного ВВП. Доречи, вклад усього російського ІКТ-сектора у економіку країни у 2003 р. не перевищував 1%. Такий швидкий підйом повинен стати головним фактором стрімкого економічного зростання країни. Тут Індія вже зараз має непорівнянні переваги перед іншими країнами, що тільки почали розвиватися у цьому напрямку. Це не тільки висока дисципліна працівників при низької вартості їх праці, але й відсутність мовного бар'єру (англійська мова в Індії давно отримала статус офіційної і стала мовою спілкування у сфері ІКТ), а також велику кількість сертифікованих спеціалістів. По цим параметрам країна випереджує навіть Китай. Індійці зараз працюють майже у кожній компанії Силіконової долини. Країна стоїть на порозі велетенського прориву уперед.

Вже сьогодні біля 60% обігу (рос.-оборота) ПЗ — експортні операції, на котрі працюють 400 компаній. На долю двадцяти самих крупних з них приходиться 67% усього обігу. При цьому 57% індійського експорту йде у США, а 22% — у західноєвропейські держави і тільки 21% — у інші країни.

Про потенціал країни говорять такі цифри. У кінці 1990-х років у Індії діяли 226 університетів (у 1972 р. було тільки 83 університети, 9 інститутів, с 2,6 млн. студентів). Усього у кінце XX віку в університетах Індії навчалися більш ніж 6,5 млн. студентів (5-6% молоді країни у віці 17-23 років). В усіх вищих навчальних закладах, окрім навчання студентів, ведуться і наукові дослідження. По кількості англомовних дипломованих спеціалістів у сфері інформаційних технологій Індія знаходиться на другому місці у світі після США, а у індійських компаніях над розробкою ПЗ працюють біля трьохсот тисяч спеціалістів. Таким чином інформаційна складова відіграє також важливу економічні роль у розвитку багатьох країн.

Запитання

1. Які носії використовувалися раніш і використовуються зараз для зберігання і передачі інформації?
2. У яких видах існують і передаються знання?
3. Завдяки чому постійно прискорюються темпи накопичення знань?
4. Чому мова, письменність, печатний станок та інші засоби фіксації потрібної людині інформації звуться технологіями?
5. Чому англійському винахіднику Чарльзу Беббіджу не вдалося реалізувати свою обчислювальну машину?
6. Які технології послідовно змінювали одна одну у конструюванні різних типів електронно-обчислювальних машин на етапах їх розвитку?
7. Які технологічні новинки створили можливість побудови перших персональних комп'ютерів?
8. Яку роль грає операційна система у роботі самого комп'ютера і у роботі користувача?

Якщо нічого не виходить, прочитай
у кінці кінців інструкцію!

Журнал CW/K №42/98

2. ВИТОНЧЕНІСТЬ ПРОЦЕСУ УКЛЮЧЕННЯ ПК

2.1 Універсальність комплектації персонального комп'ютеру

Рис. 2.1. Кнопки управління ПК

І елементи конструкції і програмне забезпечення ПК постійно удосконалюються и еволюціонують. Виробники стараються наблизити експлуатаційні характеристики ПК до рівня домашніх побутових приборів. Але, якщо натиснення кнопки "***Вкл***." на панелі телевізора тривіально для нас по своєму ефекту, то натиснення кнопки "Power" на системному блоку ПК (рис. 2.1) пов'язано з процесами, котрі потребують деякого осмислення.

Фундаментальна концептуальна відміна ПК від перших електронно-обчислювальних машин криється у багатоликому і ємному слові "***персональний***". Мало того, що у комп'ютерному комплекті, який уявляє собою роботоздібний (рос.-работоспособный) ПК, що об'єднує біля десятка різноманітних електронних пристроїв, аби яке з котрих може бути замінено в будь який момент на відповідне по характеристикам з величезної кількості існуючих зараз уніфікованих частин практично від будь якої фірми виробника (табл.2.1).

Таблиця 2.1.

Компоненти, необхідні для збирання сучасного ПК:

Апаратні компоненти	Склад системної (материнської) плати
Системна (материнська) плата	Гніздо процесора
Процесор	Перетворювачі напруги живлення процесору
Пам'ять (оперативна пам'ять)	Набір мікросхем системної логіки системної плати
Корпус	Кеш-пам'ять другого рівня (кеш L2)
Блок живлення (рос.-питания)	Гніздо пам'яті SIMM або DIMM
Дисковод для гнучких дисків	Роз'єми (слоти) шини
Жорсткий диск	ROM BIOS
Накопичувач CD-ROM, CD-RW або DVD-ROM	Батарея для живлення часів і CMOS
Клавіатура	Мікросхема уводу-виводу
Миша	
Відеоадаптер	
Монітор (дисплей)	
Звукова карта	
Акустична система	

Десятки тисяч програмних продуктів різного призначення можуть виконуватися на цьому конкретному наборі апаратних засобів. Така взаємозамінність апаратних компонентів та інтероперабельність (переносність) програмних засобів стає можливою завдяки тому, що ідеологія внутрішньої побудови і концепція функціонування ПК постійно узгоджуються і стандартизуються усіма зацікавленими фірмами-розробниками *hardware* і *software*. ***Зусиллями сотень тисяч фірм-виробників загальний принцип запуску і функціонування ПК уніфіковано повністю.***

2.2 BIOS усьому "голова"

Нічого при цьому не повинно завадити Вашому ПК приступити до виконання своїх прямих обов'язків – вирішувати для Вас Ваші задачі. Тому алгоритм початку його роботи "запаяне" у ***постійному запам'ятовуючому пристрої*** (ПЗП, ROM – Read Only Memory), який розміщено усередині системного блоку, а сама програма, що реалізує цей алгоритм, зветься ***базовою системою уводу - виводу*** ПК (BIOS – Basic Input/Output System)[7] (рис. 2.2). Цей модуль як Мінотавр (напівлюдина-напівзвір), наполовину відноситься до програмного, а наполовину до технічного забезпечення, одночасно виявляючись і частиною апаратури, і частиною аби якої операційної системи і, у тому числі, і MS DOS.

Рис. 2.2. ROM BIOS

Поточним часом при їх виготовленні отримали розповсюдження плати з перемичками у вигляді плавких уставок-запобіжників (рос.-вставок-предохранителей), котрі можна вибірково "перепалити" за допомогою зовнішнього джерела току достатньої сили, отримав при цьому пам'ять з відповідною направленістю в роботі (Programmable Read Only Memory). Цілеспрямоване перепалення формує наперед завдані властивості такого пристрою по мірі змін внутрішніх і зовнішніх вимог до нього. Незайве нагадати, що тільки у мікропроцесорі за період у яких нибудь 30 років починаючи з 1971 року кількість транзисторів збільшилась з 24 тисяч до 45 млн., а частота його роботи зросла з 4,77 Мгц до 3,2 ГГц (3 200 Мгц!), що викликало необхідність внесення відповідних змін і до BIOS.

Як правило, BIOS для сучасних системних плат розробляється однією з декількох фірм, які спеціалізуються на цьому: Phoenix Technology, Award Software, American Megatrends Inc. (AMI) та деяких інших.

BIOS у більшості PC-сумісних комп'ютерів виконує п'ять головних функцій:

❶ *POST* – самотестування при включенні живлення процесора, пам'яті, набору мікросхем системної логіки, відео адаптеру, контролерів диску, дисководу, клавіатури і інших життєво важливих компонентів системи.

[7] BIOS – група програм, котрі працюють безпосередньо з базовими апаратними засобами комп'ютера і з деякими периферійними пристроями, виконуючи основні фундаментальні задачі у системі – обмін на ровні байта з клавіатурою, екраном, дискетой, жорстким диском и т.і.

❷ ***Програма установки параметрів BIOS (Setup** BIOS)* – конфігурування параметрів системи. Ця програма запускається при натисненні визначеної клавіші (або комбінації клавіш) у час виконання процедури POST.

❸ ***Початковий завантажувач системи*** – виконання пошуку головного завантажувального сектору на дискових пристроях. Якщо останні два байти цього сектору (його сигнатура) дорівнюють *55ah*, даний код виконується.

❹ *BIOS* – набір драйверів, призначених для взаємодії операційної системи і апаратного забезпечення при завантаженні системи. При запуску DOS або Windows у режимі захисту від збоїв використовуються драйвери пристроїв тільки з BIOS.

❺ ***Обробка переривань***[2] при роботі з зовнішніми пристроями ПК.

Так як при кожному черговому включенні комп'ютера невідома поточна (рос.-текущая) конфігурація системи (деякі пристрої можуть бути на деякий час відключені або від'єднанні, а інші знаходитись у непрацездатному стані), тобто першою важливою задачею BIOS є автоматичне тестування, а просто важучі "опитування" усього підключеного до ПК обладнання і занесення у визначене місце пам'яті інформації про пристрої, що приймають участь у виконанні поточної роботи. Автотестування виконується програмою POST (Power On Self Test – автотест при включенні електричного живлення).

Рис. 2.3. Кнопка RESET ПК

При рестартуванні (повторному запуску) ПК за допомогою натиснення комбінації клавіш Ctrl-Alt-Del ще раз виконується програма POST, але вже без тесту пам'яті. Цей спосіб перезапуску має спеціальну назву – "теплий" перезапуск і може здійснюватися за допомогою спеціальної кнопки "Reset" (рис. 2.3), що знаходиться на передній панелі системного блоку ПК. Інший спосіб – "холодний" перезапуск, який реалізується шляхом виключення і повторного включення живлення ПК. Обидва ці способі є рятуванням у випадку "зависання" комп'ютера (а вірніше його операційної системи), за обставинами, що не залежать від користувача.

Перша інформація, котра з'являється на екрані комп'ютера після його включення – це назва фірми-виробника BIOS, а потім і сама BIOS на екрані дисплея показує список пристроїв, які підключені та їх характеристики. Більша частка часу самотестування витрачається на перевірку працездатності елементів оперативної пам'яті: чим більше мікросхем пам'яті встановлено в ПК, тим довше йде процес тестування. При виявленні яких нибудь негараздів, BIOS (рис. 2.4) виводить на екран ПК відповідні повідомлення (звичайно у вигляді умовного коду помилки) і сповіщає про це користувача звуковим сигналом. Подальша робота машини при цьому завершується і користувачеві

[2] Переривання (рос.-прерывание) – обрив нормальної послідовності виконання інструкцій у роботі комп'ютера. Переривання викликає автоматичну передачу управління на заздалегідь визначену адресу у пам'яті, де розташована послідовність команд, виконання яких і складає процес переривання.

потрібно прийняти міри до усунення несправностей, які були виявлені несправності. Іноді причиною помилки може послужити просте порушення контакту (наприклад, при сильному вигині (рос.-изгибе) кабелю, що приєднує клавіатуру, або при частковому виходу з гнізда однієї з печатних плат, які вставляються всередину системного блоку). Такі помилки ліквідовуються легко, але деякі несправності потребують заміни відповідних вузлів або мабуть цілих пристроїв.

Рис. 2.4. Зовнішний вид схеми BIOS

По закінченню процесу тестування, починається етап "розкрутки" операційної системи (bootstrapping). Буквально цей термін перекладається як "підняття себе за халяву (рос.-голенища) чобіт (рос.-сапог) або за шнурки ботинок". Поміж тим, мова йде не про відомий подвиг Барона Мюнхаузена, а про геніальну знахідку розробників – "затаскуванні (рос.-втаскивании)" практично аби якої ОС в комп'ютер зусиллями схеми BIOS. Це вона, як би навпомацки (рос.-наощупь), починає "обшарювати" можливі місця присутності невеликої, але дуже важливої програми початкового завантаження DOS (Disc Operating Sysytem) або ОС – *Boot record* на одному з дискових пристроїв комп'ютера.


Рис. 2.5. Вставка дискети у слот дисковода

Для чого це необхідно? Розглянемо реальну ситуацію. Припустимо Ви придбали комп'ютер і ще не вспіли привести в робочий стан його жорсткий диск. Тобто не відформатували[3] його і не встановили на ньому операційну систему, яка Вам потрібна. Як же у цьому випадку починати роботу?

По-перше, необхідно вставити у флоппі-дисковод (тобто дисковод для гнучкого диску) системну дискету (рис. 2.5), тобто дискету з записаною на ній, як правило, ОС DOS і нажати кнопку Power. Варто нагадати, що на аби яку дискету при форматуванні в Boot sector завжди записується Master Boot Record, а на системну –ще й усі модулі DOS. Тому, коли Ви уставите цю системну дискету у флоппі-дисковод, то при включенні ПК розумний BIOS знайде на ній Master Boot Record и завантажити його в оперативний запам'ятовуючий пристрій (ОЗП). Природно, що "аби яка" дискета Вас не врятує – адже один Boot Record в полі не воїн! Тому добрий (рос.-хороший) хазяїн завжди повинен мати в запасі ще одну системну дискету з якої-небудь програмою файл-менеджером (Norton або Volcov Commander, Far Manager та ін..). У комп'ютері звичайно передбачено (и це зручно), що роль

[3] Форматування – програмно кероване нанесення на поверхню дисків ділянок стандартної дліни (секторів) для наступного запису файлів.

системного диску у ПК може грати по вибору гнучкий або жорсткий магнітний диск. Тому BIOS спочатку робить спробу здійснити читання з гнучкого диску (часто званого флоппі-диском) (рис. 2.6).

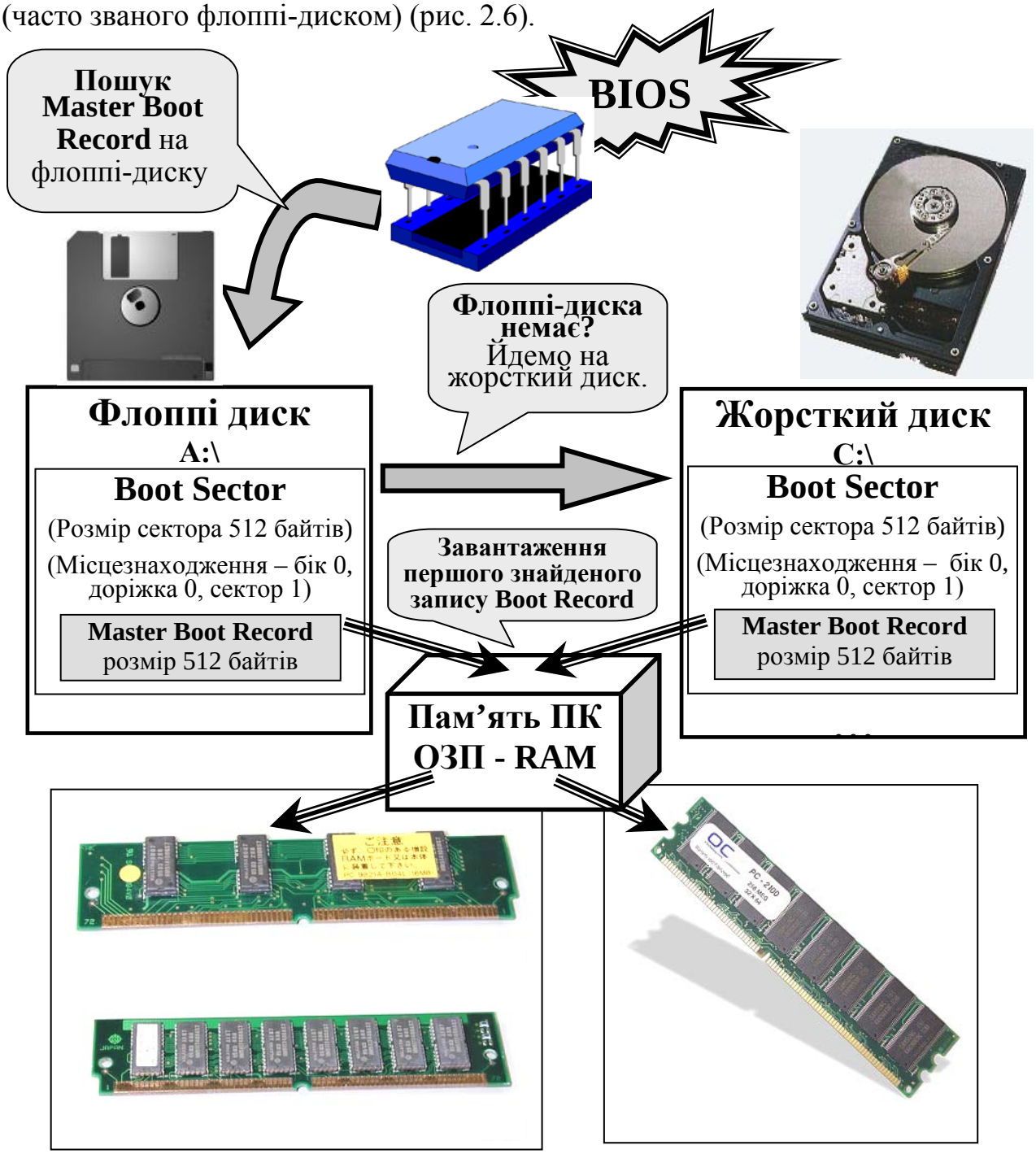


Рис. 2.6. Процес розміщення в ОЗП програми початкового завантаження Master Boot Record (початковий запис)

Якщо гнучкий диск встановлено, то саме він і вважається системним. Коли ж гнучкий диск відсутній, то BIOS звертається до жорсткого диску, вважаючи системним його. Зрозуміло, що до цього моменту він повинен бути вже відформатований і містити усі необхідні компоненти операційної системи, яка встановлюється (MS-DOS, Windows 9x/200x/XP або якась інша).

Треба відмітити, що послідовність пошуку програми початкового завантаження можна міняти, додавати нові пристрої (припустимо CD-ROM) і т.д. зробив відповідні зміни у установках програми Setup BIOS.

Важливо також і те, що системна BIOS містить драйвери головних компонентів ПК (клавіатури, дисководу, жорсткого диску, послідовного і паралельного портів і т.д.), які необхідні для початкового запуску комп'ютера. По мірі появлення нових пристроїв (відео адаптерів, накопичувачів CD-ROM, жорстких дисків з інтерфейсом SCSI і т.д.) їх процедури ініціалізації не додавалися у системну BIOS. Гостра необхідність у таких пристроях при запуску комп'ютера відсутня, тому необхідні драйвери завантажуються з диску під час запуску операційної системи. Це відноситься до звукових адаптерів, сканерів, принтерів, пристроям PC Card (PCMCIA) і т.д.

Однак, деякі пристрої вкрай необхідні при запуску комп'ютера. Наприклад, для відображення інформація на екрані монітору необхідна активізація відео адаптеру, але його підтримка не вбудована у системну BIOS. Окрім того, зараз існує величезна кількість відео адаптерів, і усі їх драйвери неможливо розмістити у системну BIOS. У таких випадках необхідні драйвери розміщаються у мікросхемі BIOS на платі цього пристрою. А системна BIOS при завантаженні шукає BIOS відео адаптеру і завантажує її до запуску операційної системи. Таке розташування BIOS запобігає (рос.-предотвращает) необхідність постійної модернізації системної BIOS при з'явленні нових моделей пристроїв, особливо які використовуються на початкових етапах завантаження комп'ютера.

Оскільки у цілому принципи завантаження операційних систем за допомогою BIOS подібні один до одного, то подальший опис процесів, що супроводжують підготовку комп'ютера до подальшої роботи, буде проведено на прикладі операційної системи MS DOS.

Отже, після самотестування пристроїв ПК, найважливішою задачею BIOS є пошук запису Master Boot Record, розміщення його у пам'яті ПК і передача на нього управління. А вже Master Boot Record, у свою чергу, починає процес завантаження модулів операційної системи. А коли Ви вже приступили до роботи на комп'ютері, уключається п'ята важлива функція BIOS – обслуговування переривань ОС, котрі виробляються програмними або апаратними засобами з ціллю виконання операцій обміну інформацією поміж пристроями і вузлами ПК.

Переривання можна розподілити на три основні групи: апаратні, логічні і програмні. Джерелами ***апаратних*** переривань – падіння напруги живлення, натиснення клавіш на клавіатурі, прихід чергового імпульсу від лічильника часу (таймера), виникнення спеціальних сигналів від накопичувачів на гнучких або жорстких дисках та ін.

Логічні, або процесорні, переривання виникають при різних нестандартних ситуаціях у роботі головного мікропроцесора — ділення на нуль, переповненні регістрів, появи «точки останову» та ін.

Програмні переривання — найбільш велика категорія. Виробляються вони, коли одна програма хоче отримати визначений (рос.-определенный) сервіс з

боку іншої програми, причому цей сервіс як правило пов’язаний з роботою апаратних засобів, а також роботою програм з ними.

Але давайте повернемося до процесу завантаження DOS. Напевно, краще усього його можна описати формулою "дідка за ріпку, бабка за дідку і т.д.". Тільки у нашому випадку це буде виглядати приблизно так: "BIOS за Boot Record, Boot Record за файл IO.SYS, IO.SYS за COMMAND.COM і т.д. (рис. 2.7).

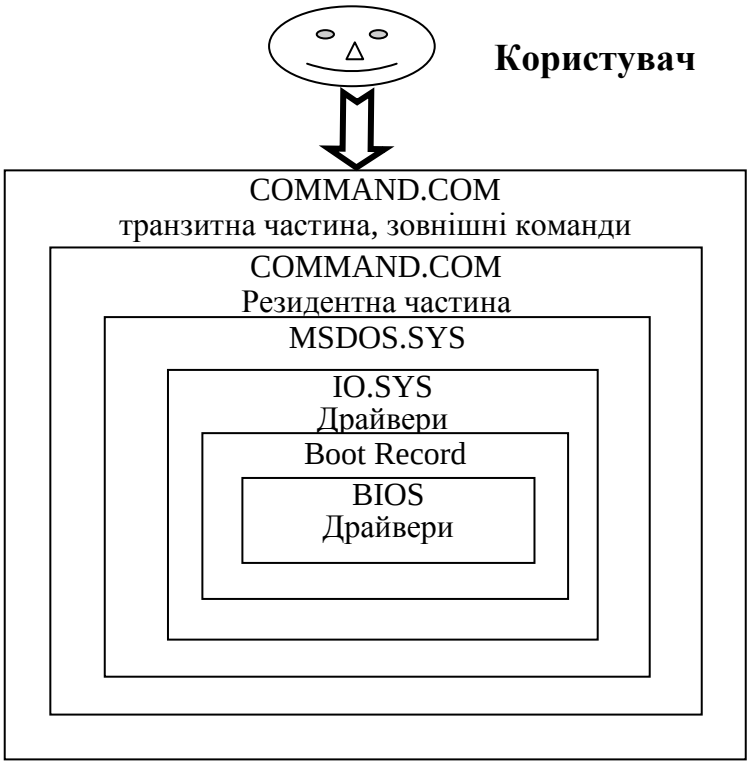


Рис. 2.7. Ієрархія модульної структури MS-DOS

Оскільки BIOS знаходиться у постійному запам’ятовуючому пристрої, зміни переривань, що у ньому знаходяться є складною задачею. Тому розробниками був запропоновано механізм, котрий дозволяє доповнювати можливості роботи BIOS з новою апаратурою.

Для цього у DOS призначено модуль IO.SYS, котрий зветься модулем розширення BIOS. У ньому можна увести нові переривання, котрі "перекривають" (заміняють) старі, або підключити програми-драйвери[4] для забезпечення роботи з аби якими новими пристроями, або працювати по-новому зі старими.

Таким чином, IO.SYS завантажує у ОЗП Boot Record разом з модулем обробки переривань DOS під назвою MSDOS.SYS. Щоб спростити роботу Boot Record обидва цих модуля розміщуються починаючи з першого сектору

[4] Драйвер – програма, яка управляє роботою деякого зовнішнього пристрою (миша, клавіатура, принтер і т.д.); драйвер як правило є інтерфейсом поміж даним пристроєм і програмами увода-вивіда ОС. Найбільш характерним прикладом драйвера служить програма KEYRUS.COM, котра кирілізує клавіатуру і монітор для забезпечення російськомовного інтерфейсу користувача з ПК.

системного диску. Першим з цих двох модулів починає роботу IO.SYS, котрий уважно читає і обробляє інформацію, яка знаходиться у файлі CONFIG.SYS. У цьому файлі повідомляється о необхідності підключення нових драйверів зовнішніх пристроїв, а також конфігурується операційна обстановка. CONFIG.SYS редагується (змінюється) користувачем заздалегідь як простий текстовий файл і його зміст буде розглянуто далі.

Завершив читання і обробку файла CONFIG.SYS, модуль розширення BIOS (IO.SYS) передає управління на вже завантажений до цього моменту у ОЗП модуль обробки переривань DOS під назвою MSDOS.SYS, у котрому робляться системні установки і провадиться підготовка до завантаження у ОЗП командного процесора COMMAND.COM, який ще поки знаходиться на системному диску. Після цього управління повертається у модуль розширення BIOS IO.SYS, котрий робить завантаження командного процесору (файл COMMAND.COM) з диску у ОЗП і передає йому управління.

Командний процесор COMMAND.COM при завантаженні виконує командний файл AUTOEXEC.BAT і на відміну від своїх вимогливих (рос.-требовательных) побратимів (рос.-собратьев) Boot Record, IO.SYS і MSDOS.SYS може розташовуватися на системному диску у аби якому місці. При цьому він розглядається як звичайна програма. Доречи, це він печатає стрічку-запрошення на екрані дисплея і виконує усі побажання користувача, які завдаються у вигляді команд DOS. Його головні функції полягають у наступному:

❶ аналізувати ті різноманітні команди, що уводяться користувачем з клавіатури або з командного файлу з розширенням .BAT;

❷ завантажувати у ОЗП і виконувати зовнішні програми DOS і різноманітні прикладні програми (файли з розширенням .COM и .EXE) (рис. 2.8).

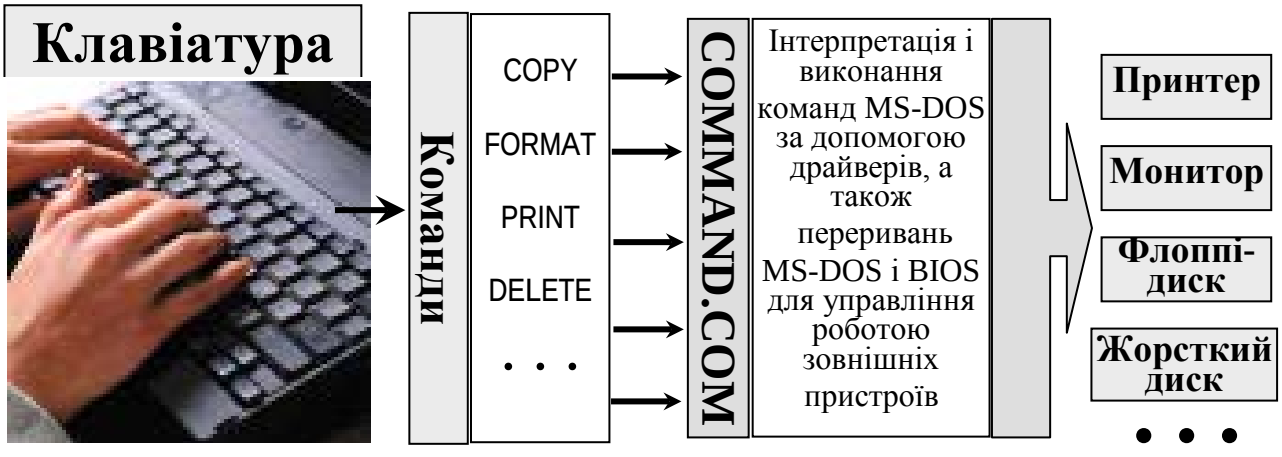


Рис. 2.8. Шлях (рос.-путь) команд від їх уводу з клавіатури до виконання їх ОС MS DOS

Образно кажучи, комп’ютер можливо уявити істотою (рос.-существом), яка спілкується з зовнішнім світом за допомогою своїх периферійних (зовнішніх) пристроїв. Його органи почуттів (рос.-чувств) – це клавіатура, миша і сканер (рис.2.8), а засоби зворотнього спілкування – принтери, дисплей,

диски та інші периферійні пристрої, через котрі провадиться контакт комп'ютера з зовнішнім світом конкретних об'єктів.

Всі вони, по суті, є буферами[5] для прийому, переробки і передачі потоків інформації поміж усіма пристроями комп'ютера, які мають ***різні швидкості*** роботи (мікропроцесор – до 3,2 ГГц, принтер – декілька стрічок в секунду і т.д.) (рис.2.9).

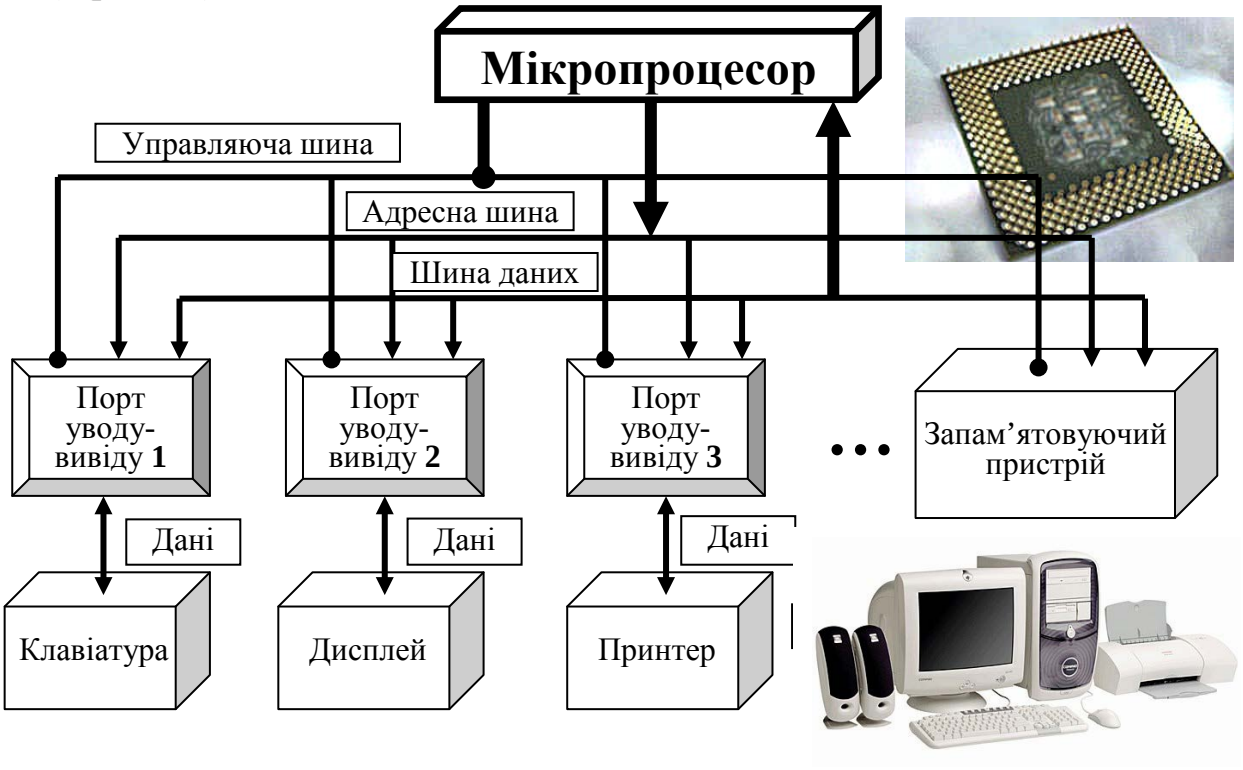


Рис. 2.9. Організація взаємодії поміж пристроями комп'ютера

Пристрої уводу отримують із зовнішнього світу команди і пов'язані з ними дані, котрі поступають у пам'ять для обробки. Пристрої виводу отримують обчислені результати і передають їх користувачеві або іншому пристрою для виводу або збереження (рос.-сохранения). Точки контакту поміж пристроями уводу - виводу і мікропроцесора звуться ***портами уводу-виводу*** (рис. 2.10).

У відповідності з загальноприйнятими угодами ***портом уводу*** зветься аби яке джерело даних, а ***портом виводу*** – приймач даних. Шини – це інформаційні канали, які з'єднують пристрої всередині комп'ютера. По управляючим шинам подаються команди на виконання різноманітних дій. Адресні шини служать для вибору потрібних портів і розміщення у пам'яті інформації, що передається. Зрозуміло, що по шині даних передаються дані (потоки даних).

Порти уводу-виводу мають свої адреси у пам'яті (ОЗП), так що до одного мікропроцесора може бути підключено декілька пристроїв уводу-виводу. По цієї адресі у пам'яті знаходиться поле чарунок (рос.-ячеек), через котрі і провадиться обмін інформацією поміж пристроями і користувачем. Процесор

[5] Буфер – область пам'яти для тимчасового (рос.-временного) зберігання (рос.-хранения) даних.

черпає інформацію з одного місця (наприклад, з клавіатури) і передає її у інше місце (наприклад, на дисплей).

Рис. 2.10. Конектори портів входу-виходу ПК і сам порт (знизу)

Таким чином, ми приходимо до осмислення того факту, що мозок комп'ютеру – це процесор, серце – таймер, який задає ритм роботи усього його організму, кровоносні судини (рос.-сосуды), які несуть інформацію – це шини адреси і даних, а органи почуттів – це порти, які приймають інформацію (управляючі сигнали і дані) від зовнішніх пристроїв і відсилають її назад.

Кожного разу, коли ми натискуємо або відпускаємо одну з клавіш клавіатури ПК, схеми клавіатури генерують однобайтне число, яке називається скен-кодом і котрий показує тільки одне – нажата клавіша або відпущена, оскільки скен-коди натиснення і відпускання клавіші різні. При цьому, ні один з скен-кодів ще не пов'язаний з визначеним символом, а лише відмічає місцезнаходження клавіші на клавіатурі. Адже за клавіатурою може працювати і англієць, і китаєць (рис. 2.11), і фін, і, звичайно ж – українець. Окрім того, звісно, що населення Землі говорить сьогодні більш ніж на 3 000 різних мов, а пише тільки на 100 з них. На англійської мові говорять 1 млрд. 400 млн. жителів планети, з котрих тільки 400 млн. вважають її за свою рідну.

Рис. 2.11. Уявлення імен папок (каталогів) на китайській мові

Присвоювання клавіші символу мови – це робота підпрограм (переривань) ROM BIOS за допомогою портів, а відображення рідної для користувача мови виконується додатковими драйверами клавіатури (PROKEY, KEYRUS і ін.).

Підводячи підсумки вищесказаного, можна сказати, що при включенні комп'ютера у його пам'ять завантажується операційна система DOS. Цей процес зветься ***початковим завантаженням (рос.-начальной загрузкой)*** (або перезавантаженням).

У ході цього процесу:

❶ перевіряється правильність роботи пристроїв комп'ютера;

❷ у оперативну пам'ять комп'ютера завантажується ОС DOS або компоненти іншої операційної системи;

❸ робиться (рос-происходит) настройка DOS на обранні параметри конфігурації і ті пристрої, ***які підключені до комп'ютера у цей поточний момент***;

❹ виконуються команди і програми, які вказані користувачем у файлі *autoexec.bat;*

❺ на екран виводиться запрошення (рос.-приглашение) DOS, яке указує, на те, що DOS готова до прийому команд користувача (рис.2.12).

2.3. Як операційна система управляє процесом уводу-виводу

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

C:\>_
```

Рис. 2.12. Запрошення ОС DOS для уводу чергової команди для роботи з диском С:

BIOS – це термін, котрий використовується для опису базової системи уводу-виводу. По суті, BIOS уявляє собою "***проміжний шар (рос.-слой)***" поміж програмною і апаратною частинами системи, яка складається з комбінації усіх типів BIOS, а також драйверів пристроїв, які завантажуються. Частина BIOS, яка міститься у мікросхемі на системній платі або платах адаптерів, зветься ***firmware***[8]. Стандартна PC-сумісна система складається з декількох шарів, котрі пов'язані поміж собою (рис. 2.13).

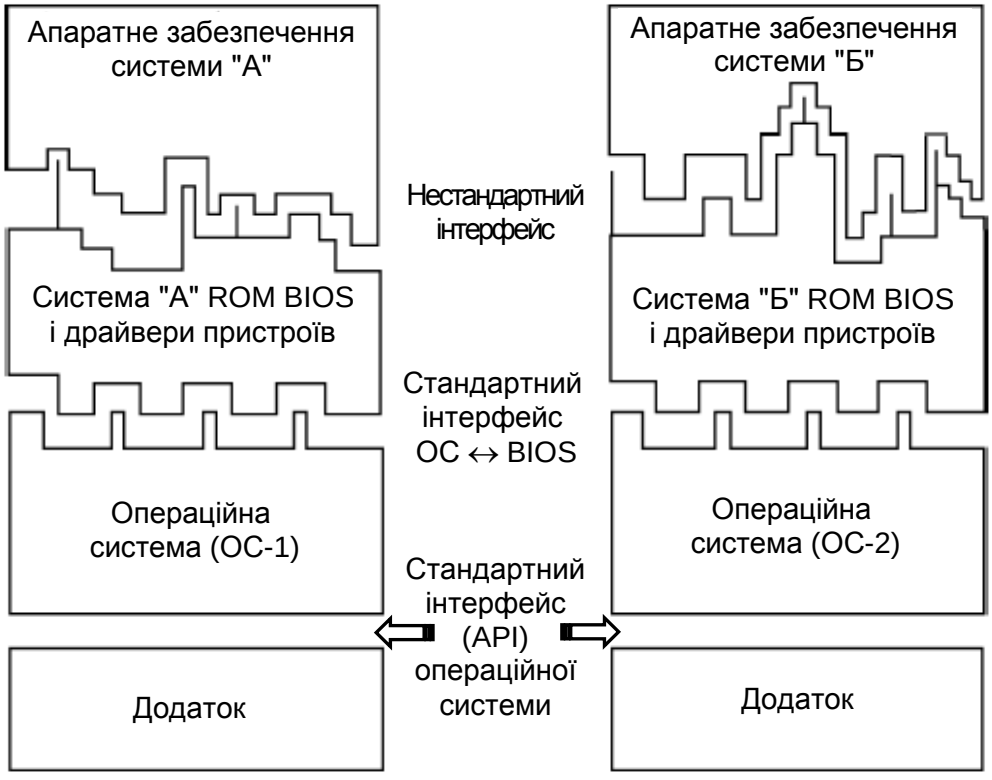


Рис. 2.13. Умовний розподіл ПК на декілька шарів (рос.-слоёв)

На цьому рисунку показані два різних комп'ютера, у котрих використовується унікальна BIOS у ролі інтерфейсу поміж апаратним

[8] firmware-програмно-апаратне забезпечення, програмно-апаратні засоби, апаратно-програмне забезпечення.

забезпеченням, операційною системою і її додатком (рос.-приложением). Тобто BIOS настроюється на роботу з конкретною конфігурацією апаратних засобів аби якого ПК. Таким чином, на цих комп'ютерах може бути встановлено різне обладнання (процесори, жорсткі диски, монітори і ін.), на котрому можна запускати однакове програмне забезпечення.

Зв’язок поміж додатками і операційною системою здійснюється за допомогою відповідного API (Application Programming Interface). Цей інтерфейс визначає, наприклад, як виконується запис і зчитування даних на диск, печать і інші функції. Оскільки додаток не залежить від встановленого апаратного забезпечення, то і усі його виклики обробляє операційна система, котра вже містить інформацію про встановлене обладнання. Операційна система, у свою чергу через BIOS звертається безпосередньо до апаратного забезпечення. Цей зв'язок реалізован у вигляді драйверів пристроїв. Причому у кожній операційній системі – DOS, Windows9x, Windows NT, Windows 2000, OS/2, Linux або іншій – для одного і того ж пристрою необхідні свої драйвери. Як видно з рисунку 2.13, додатки і операційна система ідентичні у більшості комп'ютерів, а BIOS "підстроюється" під визначене апаратне забезпечення і, незалежно від встановленого обладнання, забезпечує стандартний інтерфейс для операційної системи. Таким чином, BIOS уявляє собою інтерфейс поміж апаратним забезпеченням і операційною системою.

Як правило прикладна програма не працює сама з апаратурою, а користується послугами операційної системи. Виключення (рос.-исключение) складають випадки, коли користувач самостійно забезпечує доступ до апаратури зі своєї програми. При зверненні до стандартної апаратури з програми користувача у комп’ютерах фірми IBM або сумісних з ними використовується механізм драйверів. Однак драйвери ОС не завжди звертаються прямо до апаратури. Як правило вони викликають функції BIOS, і вже BIOS виконує усі дії щодо уводу-виводу на рівні своїх переривань. Звичайно, BIOS містить програми обслуговування тільки стандартних пристроїв уводу-виводу, а нестандартні пристрої обслуговуються драйверами безпосередньо.

Використання BIOS як додаткового інтерфейсу поміж драйверами стандартних пристроїв і апаратурою різко підвищує "живучість" DOS на не "зовсім сумісних" з IBM PC персональних комп’ютерах інших виробників. Це можливо завдяки тому, що виробники сумісних комп’ютерів ураховують (рос.-учитывают) у програмах BIOS особливості всієї апаратури, яка постійно з'являється. У цьому випадку і DOS, і програма користувача тим більш не бачать ніяких відмін у нових компонентах ПК.

З іншого боку, користувачі можуть легко доповнювати ОС своїми особистими драйверами, які складені (рос.-составлены) для нестандартних пристроїв, або заміняти стандартні драйвери і функції BIOS. При цьому треба відмітити, що оскільки драйвер повинен враховувати усі деталі конструкції кожного пристрою і працювати у режимі реального часу, тому хочаб частина його повинна бути написана на машинно-орієнтованій мові програмування.

2.4 Управління пристроями за допомогою драйверів

Управління зовнішніми пристроями - це одна з важливіших функцій аби якої операційної системи. Система повинна забезпечувати ефективний і зручний доступ до периферійних пристроїв, а також забезпечувати можливість уніфікованої розробки програмного забезпечення для зовнішніх пристроїв, які знов підключаються. Тому після завантаження операційної системи, остання, у відповідності зі вказаними у спеціальних областях жорсткого диску даними, завантажує в ОЗУ драйвери підключених до ПК пристроїв. Номенклатура таких зовнішніх пристроїв надзвичайно велика. До них можна віднести і нові типи пристроїв зовнішньої пам'яті (Zip, Jazz, стримери, магнітні диски підвищеної ємності і др.), бездротові клавіатури і маніпулятори типу миша (рис. 2.14), тьюнери, відео приставки, цифрові фотокамери, мікрофони і багато іншого.

Рис. 2.14. Різні пристрої комп'ютера

Більшість з них обмінюються даними с мікропроцесором асинхронно, тобто через нерівні відрізки часу. Незважаючи на це, технології створення відповідних драйверів вирішують практично аби які задачі. Розглянемо процес взаємодії комплексу пристроїв, які підтримуються відповідними драйверами. Отже, аби який зовнішній пристрій характеризується унікальним уніфікованим інтерфейсом обміну даними з комп'ютером, а також набором внутрішніх команд. У їх склад, як правило, входять наступні:

❶ ініціалізації, які приводять пристрої у готовність до роботи;

❷ управління компонентами (механічними, електричними, електронними та ін.) даного пристрою;

❸ управління обміном даними від комп'ютера до пристрою і назад;

❹ завершення процесу сумісної роботи (очистка регістрів, буферів, скидання (рос.-сброс) прапорців (рос.-флагов), відключення живлення елементів, які завершили роботу).

Кожна версія операційної системи концептуально розробляється один раз, а зовнішніх пристроїв кожного року у світі з'являються десятками тисяч. Тому уявляється важливим з боку ОС розглядати зовнішній пристрой як деякий абстрактно узагальнений (рос.-обобщённый) об'єкт, який має незмінний інтерфейс, тобто уніфіковані засоби доступу до нього і обміну даними з ним. А усі тонкості, які відносяться до специфіки конструкції і функціонування реального пристрою, розробники намагаються (рос.-стремятся) «припрятати» у тілі драйвера, який програмується.

При реалізації вказаної концепції для доступу до зовнішнього пристрою в DOS і у багатьох інших ОС використовується універсальна абстракція файла. Важливо відмітити, що як правило файл практично у більшості операційних систем уявляє собою комплексну структуру, яка уключає (рис. 2.15):

❶ ім'я з розширенням з трьох символів, яке міститься у каталогу (FAT);

❷ байт атрибутів файлу (файл тільки для читання, системний і т.д.);

❸ час (рос.-время) і дата створення файлу або її модифікації;

❹ розмір файлу у байтах;

❺ посилання на перший кластер[6] магнітного диску, з котрого починається розташування усього файлу;

❻ у таблиці розташування файлів містяться номери кластерів, у яких міститься файл цілковито (!) і т.д.

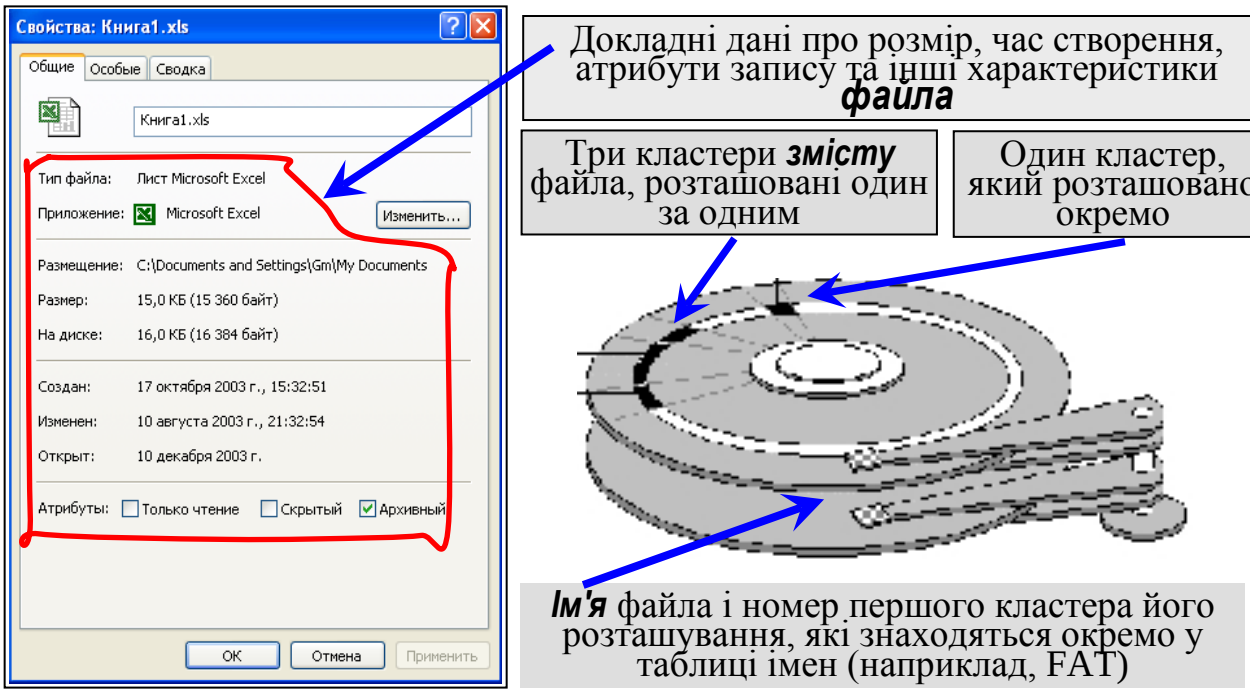


Рис. 2.15. Комплексні характеристики, які описують файл, що зберігається на магнітному диску

Окрім справжніх файлів (звичайних файлів або каталогів), котрі реально займають пам’ять на магнітних дисках, файлова система містить так звані спеціальні файли, для котрих, як і для справжніх файлів, відводяться окремі (логічні) імена, але котрим реально (рос.-на самом деле) відповідають зовнішні пристрої. Таке рішення дозволяє зручним способом (рос.-естественным образом) працювати у одному і тому ж інтерфейсі з аби яким файлом або зовнішнім пристроєм. (Насправді, у деяких випадках використовування нестандартних зовнішніх пристроїв нерідко може виходити за межі стандартного інтерфейсу.) Зрозуміло, що проста об’ява зовнішнього пристрою

[6] Кластер – група блоків диску, яка розглядається як єдине ціле. У MS DOS і деяких інших операційних системах– мінімальна одиниця розподілу дискового простіру. Складається з одного або декількох сусідніх секторів. Розмір сектора, як правило, кратний ступені числа 2. Може мати значення: 124, 256, 512 або більше кілобайт.

спеціальним файлом не дає можливості працювати з цим пристроєм, якщо не створений і відповідним чином не є підключеним до системи спеціальний програмний код, який відповідає специфіки даного пристрою.

Як і у більшості сучасних операційних систем, такого роду програмний код у ОС DOS зветься драйвером пристрою (у цьому контексті слово драйвер краще усього розуміти у значенні "управляюча програма"). Також, як і у аби який іншій системі, драйвер пристрою - це багато входовий програмний модуль зі своїми статичними даними, котрі повинні і вміють ініціювати роботу з пристроєм, виконувати обміни, які заказує користувач (на увід або вивід даних), термінувати роботу з пристроєм і обробляти переривання від пристрою.

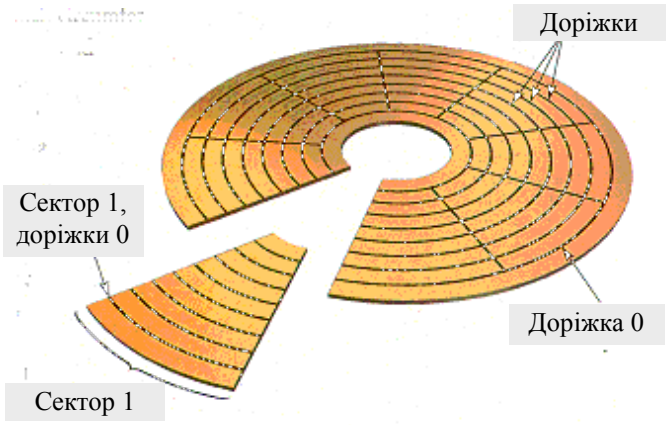


Рис. 2.16. Розташування секторів і доріжок на поверхні пластини магнітного диску

Як приклад можна розглянути команду ОС DOS «*COPY*». Її адресні частини можна розглядати під різними кутами зору і тоді вони мають різни значення (варіанти, назви і т.д.), а також і об'єкти перепису з одного місця магнітного носія на інше (рис. 2.16, 2.17).

№ пп	Ім'я команди	Перший об’єкт перепису	Другий об'єкт перепису
1	**COPY**	Джерело (рос.-источник) даних	Приймач (рос.-приёмник) даних
2	**COPY**	Ім'я файлу	Адреса, куди його потрібно записати
3	**COPY**	Путь до файлу, що переписується	Адреса кінцевого запису
4	**COPY**	Ім'я каталогу	Адреса призначення
5	**COPY**	Ім'я пристрою, що передає дані	Ім'я файлу, який отримує дані
6	**COPY**	Ім'я файлу, з котрого витягаються (рос.-извлекаются) дані	Ім'я пристрою, куди дані виводяться/розміщуються
7	**F5**	Виділений об'єкт або група об'єктів	Каталог або ім'я пристрою
8	**COPY** **(з контекстного меню об’єктів робочого столу Windows)**	Виділений об'єкт або група об'єктів	Місце призначення

Рис. 2.17. Деякі трактовки використання команди копіювання даних *COPY* у різному контексті

У загальному значенні команда COPY забезпечує перепис об’єктів з одного місця диску на інше. Об’єктами перепису як правило виступають файли і каталоги. Але в указаному користувачем контексті, джерелами і приймачами даних можуть бути також і пристрої, яким присвоюється логічне ім'я CON (табл. 2.2).

Таблиця 2.2

Використання пристроїв, як джерел (рос.-источников) і приймачів об’єктів файлового типа

Поточний (рос.-текущий) диск	Команда, що вводиться і її адресні частини	Результат виконаної дії
C:\>	**COPY** A:\ PROG C:\ FRAG	Каталог PROG с диску A: перепишеться у каталог FRAG на диску C:
C:\>	**COPY** CON FILE1.txt	Дані, які набираються на клавіатурі, уводяться у файл з іменем FILE1.txt. (CON – файл-джерело даних). Після закінчення уводу текстових даних у файл FILE1.txt потрібно натиснути сполучення клавіш Ctrl+Z, що призведе до закриття файлу з додаванням у його кінці признаку його кінця.
C:\>	**COPY** FILE1.txt CON	Дані, які містяться у файлі з ім'ям FILE1.txt, виводяться на екран дисплея комп'ютера (CON – файл приймальник даних)

Таким чином, якщо логічне ім'я CON стоїть на першому місці у команді COPY, то воно трактується як логічне ім'я системного пристрою уводу-виводу, тобто ***клавіатури***. Після завершення уводу текстових даних у відкритий файл з завданим ім'ям і закриття цього файлу сполученням клавіш [Ctrl+Z] можна продивитися його зміст на экрані дисплея. У цьому останньому випадку, CON – логічне ім’я пристрою виводу, хоча по суті, дія яка виконується позначає (рос.-обозначает) вивід (перепис) одного файлу (FILE1.txt) у другий (CON).

Концепція абстрактного файлу у цьому випадку витримується повністю, так як вказуючи стандартне логічне ім’я CON ми абсолютно не замислюємося про реальний тип, назву і фірму виробника клавіатури або дисплею. Їх драйвера підтримують концептуально відповідний файловий обмін даними.

Запитання.

1. Як звуться головні апаратні компоненти комп'ютеру?
2. Які компоненти входять до складу материнської плати?
3. Що собою уявляє базова система уводу - виводу ПК (BIOS – Basic Input/Output System)?
4. Які п'ять головних функцій виконує BIOS після включення ПК?
5. Чим ″теплий″ перезапуск ПК відрізняється від ″холодного″?
6. Які головні функції запису Master Boot Record?
7. Де може розміщуватися Master Boot Record?
8. Що таке порт ПК?

Оператор (мови) – базова одиниця ***дії*** в мовах програмування і алгоритмічних мовах.

Програма – упорядкована послідовність ***команд***, які підлягають обробці.

ГОСТ.

3. КОМАНДНА ОСНОВА РОБОТИ КОМПЬЮТЕРА

3.1. Роль команд у процесі управління комп'ютером

Персональні комп'ютери (ПК) сьогодні можна побачити на робочому місці спеціаліста практично аби якої професії. Вони стали невід'ємною частиною виробничої діяльності організацій и підприємств, розширюючи область промислової обробки даних. Ця ситуація виникла з багатьох причин, і в першу чергу тому, що ПК став доступним, мініатюрним і зручним пристроєм для уводу, зберігання, обробки і виводу інформації практично у всіх відомих предметних областях[1] (рис. 3.1). Більш того, перше питання при прийомі спеціаліста на роботу в організацію аби якої форми власності формулюється дуже просто: "Комп'ютером володієте?". Так що мова йде про нову форму грамотності – ***комп'ютерної грамотності.***

Рис. 3.1. Процес рішення задачі користувачем на комп'ютері

І щоб оволодіти цією грамотністю в умовах, коли функції комп'ютерної техніки постійно удосконалюються і ускладнюються, потрібно добре уявляти особливості його роботи і принципи функціонування.

Як відомо, процес взаємодії користувача з ПК в процесі рішення задачі ґрунтується (рос.-основывается) на декількох ключових моментах.

По-перше, при роботі з комп'ютером їм необхідно постійно ***управляти,*** тобто направляти його дії у потрібне русло. Бо, якщо після його включення не робити ніяких дій, сам комп'ютер не виконає ні однієї конкретної задачі, поки не дочекається конкретних дій (***команд***) користувача. Останні уводяться, як

[1] Предметна область – клас задач, які вирішуються програмним засобом або програмною системою.

правило, шляхом натиснення клавіш на клавіатурі ПК або кнопок миші, після наведення курсору на управляючі елементи додатків або операційної системи. Ці дії примушують ті компоненти, котрі їх отримали, породжувати (рос.-порождать), у свою чергу, командні стимули на рівні цих програмних компонентів і так далі униз по ієрархії. Щоб уникнути в суть процесів, які діються, необхідно розглянути самий верхній рівень уявлення поняття і терміна "команда" для ПК, котрий відповідає рівню взаємодії останнього з користувачем.

Як правило, під ***управлінням*** розуміють процес цілеспрямованого впливу (рос.-воздействия) на систему, який забезпечує підвищення її організованості, з ціллю досягнення того або іншого корисного результату. При цьому аби яка система управління розподіляється на дві підсистеми: ***управляючу*** і ту, якою ***управляють***. Зв'язок від управляючої підсистеми до тієї, якою управляють зветься прямим зв'язком. Такий зв'язок існує завжди. Протилежний по напряму зв'язок зветься зворотнім (рос.-обратным). Поняття зворотнього зв'язку е фундаментальним[2] в техніці, природі і суспільстві. Досвід показує, що управління без сильних зворотних зв'язків не ефективно, так як не володіє (рос.-обладает) властивістю до самовиявлення помилок, формулюванню проблем, не дозволяє використовувати можливості саморегулювання системи, а також досвід і знання спеціалістів. У комплексі ***користувач-комп'ютер*** прямим зв'язком є діяння (рос-воздействие) на комп'ютер з ціллю отримання рішення деякої задачі, а зворотньою – повідомлення про хід її рішення (рис. 3.2).

Рис. 3.2. Управляючі зв'язки при взаємодії користувача з комп'ютером

По-друге, здійснення прямих і зворотних зв'язків поміж двома системами реалізується на рівні ***інтерфейсу***[3] (комп'ютера), який забезпечує

[2] Фундаментальний – дещо, що складає головну частину або основу чогось. Як правило, застосовується як характеристика основоположних елементів наук або прикладних галузей знань. Наприклад, фундаментальні дослідження, фундаментальні поняття.

[3] Інтерфейс – засіб об'єднання двох або більше частин декількох систем.

інтероперабельність[4] поміж системами, що взаємодіють, тобто користувачем і ПК. Головні задачі підтримки інтерфейсу ПК покладається на операційну систему (ОС) і будуть обговорені у подальшому у главі 4. Інтерфейс додатків, які працюють під управлінням ОС, створюється їх розробниками і залежить від конкретних задач, які вирішуються за їх допомогою.

І, нарешті, по-третє, як управління комп'ютером, так і забезпечення роботи його і усіх його компонентів здійснюється шляхом виконання різноманітних ***команд, з котрих формуються програми або послідовності наборів команд для опису дій комп'ютера, які приводять до рішення задач користувача.*** Так, у багатьох сучасних додатках (MS Word, MS Excel та ін.) рішення аби якої типової задачі реалізується послідовністю клацань мишею на назвах меню, командах у списках, що розкриваються, командних кнопках і інших управляючих елементах, котрі у своєї сукупності і виконані у визначеній (рос.-определённой) послідовності грають по сутності роль програми з «елементів мови» даного додатку (рис.3.3).

ЗАДАЧА	**ПРОГРАМА ДІЙ.**
Відформатувати діапазон комірок (рос.-ячеек) B4:F16 у грошовому форматі.	1. Виділити діапазон комірок **B4:F16**. 2. В меню ***Формат*** оберіть команду ***Ячейки***. 3. Клацніть по вкладці ***Число***, якщо вона не є активною. 4. У списку ***Числовые форматы*** оберіть грошовий формат. 5. Установіть в полі ***Число десятичных знаков*** значення 0. 6. Нажміть кнопку **ОК**, щоб повернутися у лист.

Рис. 3.3. Програма для форматування діапазону комірок в додатку (мові) Excel

За час тривалого періоду розвитку комп'ютерів термін *команда* увібрав у себе багато значень і отримав не менше відтінків. Поза межами комп'ютерного світу слово ***команда*** має два головних значення:

❶ колектив людей, які об'єднані загальною (рос.-общей) ціллю (футбольна, пожежна та інші команди);

❷ усний наказ виконати які нибудь дії (наприклад, наліво, направо, стій, кругом і т. і.).

У інформатиці ***значень*** і ***змістів (рос.-смыслов)*** слова ***команда*** Ви можете найти значно більше, чим у всіх інших сферах діяльності людини. Вони присутні на різних ***рівнях***[5] існування і представлення компонентів

[4] Інтероперабельність – можливість (рос.-способность) обмінюватися повідомленнями, виконувати програми або пересилати дані поміж різними функціональними елементами систем.

[5] Спеціалісти розглядають дев'ять значень поняття ***рівень***, який є головним для концепції ієрархії систем. Термін «рівень» може означати: 1) ступінь взагалі (рос.-степень вообще); 2) ступінь складності; 3) ступінь глибини аналітичного дослідження; 4) виникнення (рос.-возникновение) (істотної (рос.-естественной), тваринної) організації більш високого рівня у порівнянні з наявною (рос.-имеющейся); 5) систему

комп'ютерних систем і ***інформаційних*** потоків и означають ініціалізацію того чи іншого процесу, який діється на стадіях ***проектування, реалізації*** або ***роботи*** багаточисельних елементів ПК.

3.2. Фізичний і логічний рівні застосування команд

Слід відмітити, що найбільш важливими у інформаційних технологіях є ***фізичний*** і ***логічний*** рівні уявлення компонентів комп'ютера, який перетворився за останні 20 років у технологічно и технічно надзвичайно складний комплекс різноманітних взаємодіючих поміж собою пристроїв: клавіатури, миші, різноманітних чипів і чипсетов, мікропроцесорів, дисководів для зйомних та незйомних носіїв інформації, дисплея, принтера, сканера і т.д. (рис. 3.4).

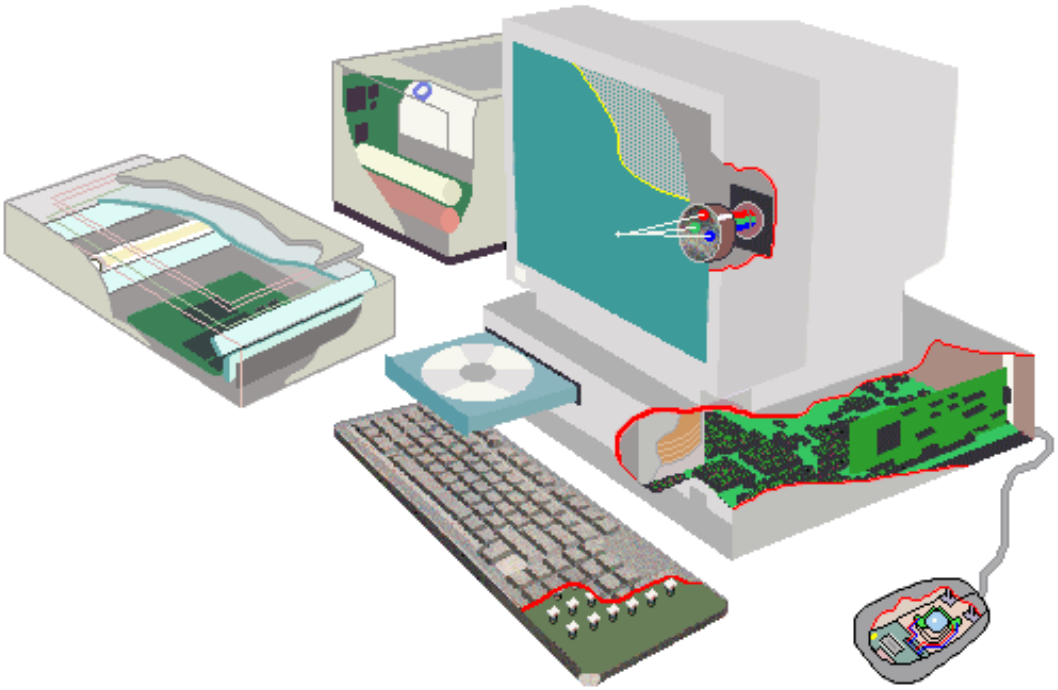


Рис. 3.4. Персональний комп'ютер у комплексі

Фізичний рівень визначає конкретну технологічну реалізацію того або іншого пристрою або елемента. Наприклад, дисплей ПК може бути конструктивно реалізован на електронно-лучової трубці (рис. 3.4) або на основі жидкокрісталічної або плазмової технологій.

Зовнішній запам'ятовуючий пристрій комп'ютера може уявляти собою флоппі-дисковод або жорсткий диск і кожний зі своєю системою команд доступу, кількості секторів, доріжок, циліндрів і т.д. Уявіть собі ситуацію, коли Вам для того, щоб переписати файл з одного пристрою на інший буде потрібно вникати до всіх технічних тонкостей різниць (рос.-отличий) принципово різних дискових пристроїв і до того ж самому писати програми обміну поміж ними.

взаємопов'язаних властивостей, або змінних; 6) розряд; 7) шар (рос.-слой); 8) головний шар (рос.-слой); 9) рівень.

Для спрощення розуміння і безпеки використання пристроїв ПК, при роботі користувача з ОС вводиться поняття логічного рівня. Це дозволяє абстрагуватися від усіх технічних і програмних тонкостей роботи з цими пристроями і оперувати, наприклад, іменами логічних дискових пристроїв у ОС MS DOS (A:, C:, D: и т.д.), або графічними уявленнями цих же дискових пристроїв на робочому столі ОС Windows. Таким чином, можна сказати, що логічний рівень – це концептуальний[6], тобто віртуальний[7] рівень, котрий уключає в себе концептуальні, а не реальні фізичні об'єкти і ховає (рос.-скрывает) принципи їх реалізації.

Це означає, що коли Ви звертаєтесь до логічного імені флоппі-диска або жорсткого диску для Вас не мають значення ані їх конструктивні особливості, ані тонкості технічної реалізації (може тільки об'єм вільного дискового простору (рос.-пространства)!) (рис. 3.5).

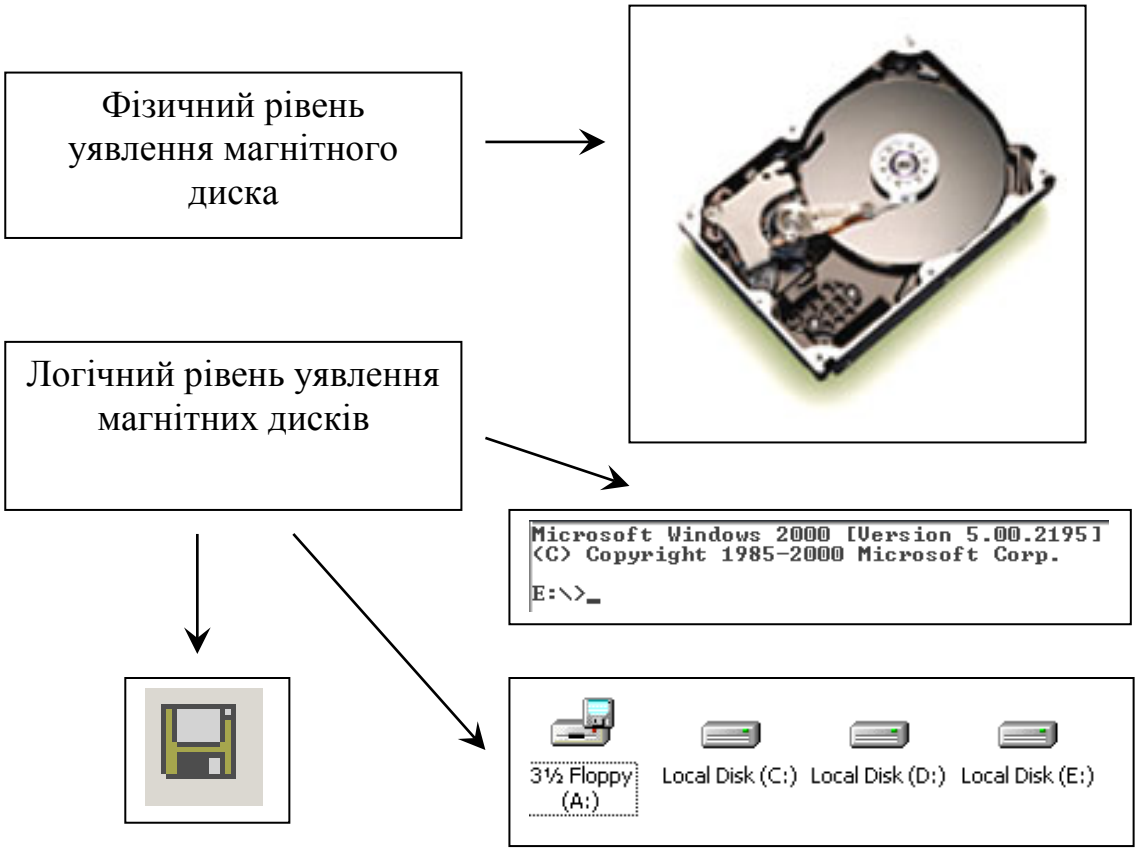


Рис. 3.5. Рівні логічного уявлення магнітних дисків у ПК, які забезпечують роботу з даними, що на них розташовані

Таким чином, логічним іменем диска може служити абстрактне позначення пристрою комп'ютера у вигляді додаткового текстового і/або графічного імені/позначення, які приписуються операційною системою для зручності їх використання. Логічне ім'я дає можливість користуватися об'єктом не

[6] Концепція – визначений (рос.-определённый) спосіб розуміння, трактовка якого нибудь предмета, явища, процесу.

[7] Віртуальний – який не має фізичного втілення (рос-воплощения) або який сприймається (рос-воспринимаемый) інакше, ніж реалізован.

поглиблюючись у особливості його фізичної реалізації. Наприклад, кожний ***логічний*** диск ПК є частиною ***фізичного*** жорсткого диску, котра розглядається як окремий жорсткий диск зі своїм іменем накопичувача (рис. 3.6).

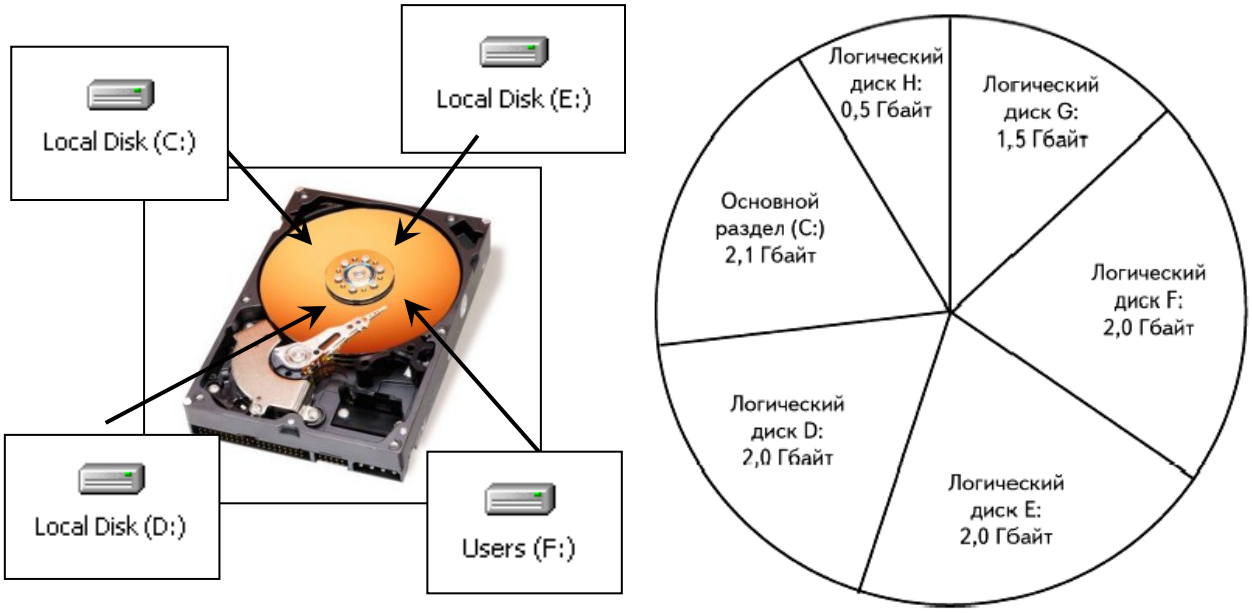


Рис. 3.6. Розміщення логічних дисків на фізичному

3.3. Команди фізичного і логічного рівнів

Багато які компоненти і пристрої ПК містять в своєму складі різноманітні інтегральні схеми, сама велика з котрих у ПК звичайно є процесором. З часів винаходження першого в світі мікропроцесора (МП) Intel 4004 у 1971 році, у котрому містилося 2 300 напівпровідникових транзисторів пройшло вже достатньо багато часу. Але й і зараз технічно процесор реалізується у вигляді великої інтегральної схеми (ВІС), структура котрої постійно ускладнюється, а кількість функціональних елементів (типу діод або транзистор) на ній постійно росте (від 5 мільйонів у процесорі Pentium II до 42 млн. транзисторів у Pentium 4). І це не межа (рос.- предел). У кінці 2002 року співробітники компанії Intel розробили абсолютно новий тип кремнієвих транзисторів, швидкість переключення яких приблизно у 1000 разів перевершує (рос.- превосходит) аналогічний показник транзисторів, які використовуються у сучасних чипах. Новий транзистор має ширину усього у 80 атомів (20 нм), що дозволить умістити в майбутньому на одному кристалі більш ніж мільярд таких елементів (рис. 3.7).

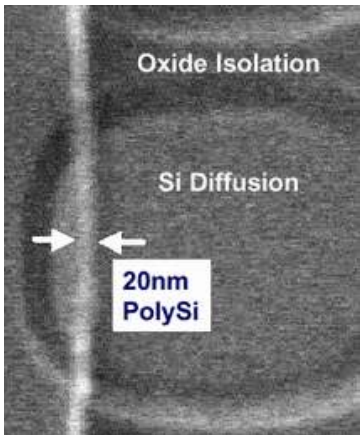


Рис. 3.7. Транзистор, який є елементом інтегральної і електричної схем, що представляють мікропроцесор.

Не дивлячись на такий неймовірний ріст концентрації електронних елементів у процесорі, робота усіх цих транзисторів у електричних схемах і самі електричні схеми практично не змінилися з часів їх винаходу.

Командами відкриття і закриття транзисторів на рівні електричних схем і ланцюгів є електричні ***сигнали***[8], які звуться ***електричними імпульсами***[9] (рис. 3.8). Таким чином, необхідно чітко уявлять, що на самому нижньому рівні абстракції, тобто на фізичному рівні конструкції елементів ПК, поняття ***команда*** відноситься до засобів управління електронними компонентами пристроїв, таких як ***процесор***[10], ***чипсет***[11] або ***материнської плати***, ***схеми пам'яті*** та інших, які побудовані на тріодах, діодах, опорах (рос.-сопротивлениях), провідниках і т.д. (рис. 3.8 и 3.9).

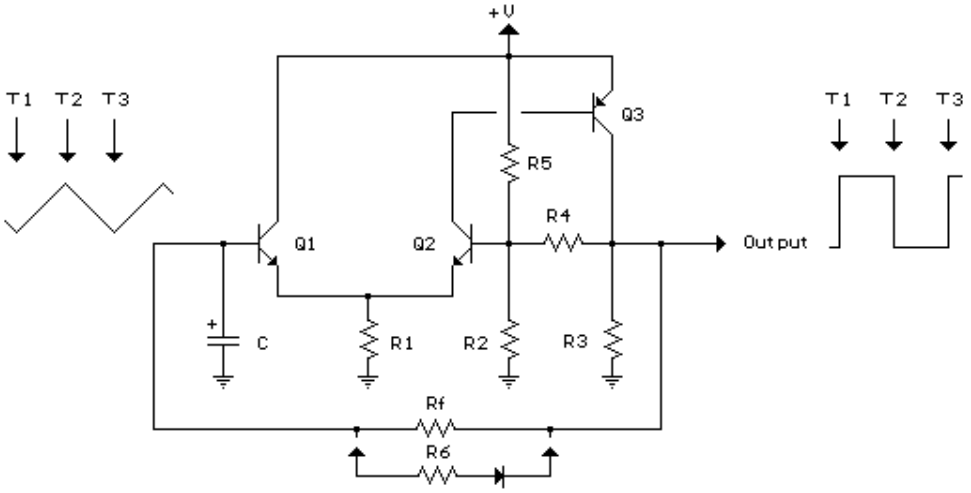
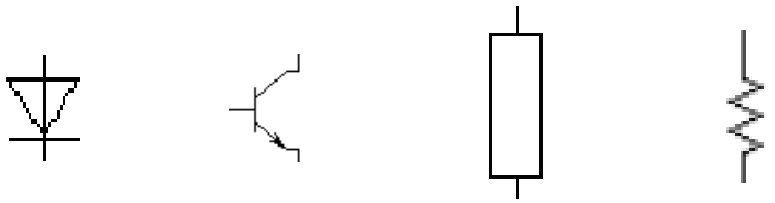


Рис. 3.8. Електрична схема перетворення
Пилкоподібного (рос.-пилообразного) сигналу у прямокутні імпульси



А) Б) В) Г)

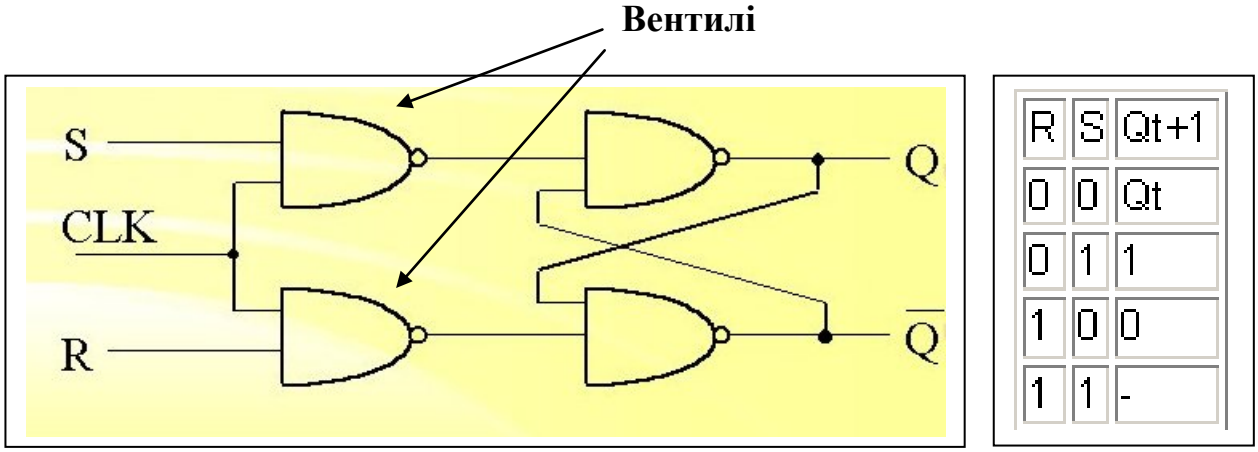
Рис. 3.9. Позначення діода (А), транзистора (Б) і опорів
(В) и (Г) на електричних схемах.

Імпульси забезпечують не тільки злагоджену (рос.-слаженное) взаємодію, включення і виключення самих пристроїв і блоків, побудованих на цих пристроях, але й моделюють особисто основу функціонування ПК – цифрові значення 0 и 1, які використовуються для кодування усіх даних і інформації .

[8] Сигнал – мінливий у часі фізичний процес, що відображає повідомлення, яке передається.
[9] Електричний імпульс – короткочасне відхилення напруги або струму від деякого постійного значення.
[10] Процесор (мікропроцесор) – це мікросхема, яка реалізує функції центрального процесора персонального комп'ютера.
[11] Чипсет – набір мікросхем материнської плати, які реалізують архітектуру комп'ютера.

Проектування комп'ютерних і мікропроцесорних пристроїв починається на логічному рівні шляхом розробки логічних схем роботи нового пристрою по завданим вимогам. Складні логічні системи будуються з ***типових логічних вузлів***, які уявляють собою схеми з логічних вентилів. До типових логічних вузлів, що відіграють основоположну роль у мікропроцесорних і комп'ютерних системах, відносяться: ***тригери*** (рис. 3.10***), регістри, лічильники, дешифратори, селектори, схеми виконання арифметичних операцій (суматори і віднімателі), а також системи шин.***

Транзистори і інші елементи мікропроцесорів групуються у логічні схеми, елементами яких є так звані ***вентилі***, сполучення сигналів на входах и виходах яких реалізує модель поведінки або протікання того або іншого реального процесу. Рисунки з зображеннями елементарних логічних схем і їх зв'язків називаються логічними діаграмами (рис 3.10), у котрих при розробці логіки роботи комп'ютера і його пристроїв повідомлення моделюються у вигляді ***сигналів (команд)***, які подаються на їх входи.



R	S	Qt+1
0	0	Qt
0	1	1
1	0	0
1	1	-

Рис. 3.10. Логічна схема RS-тригера і таблиця сигналів і станів (рос.-состояний) на його входах

Одними з найголовніших елементів МП є ***тригери*** - найпростіші цифрові схеми послідовного типу. У залежності від алгоритму роботи, який реалізується, тригер може мати установочні, інформаційні або управляючі входи. Він також має два стійких стани: Q=1 и Q=0 (див. рис.3.10) и тому його іноді називають бістабільною схемою. У якому з цих станів опиниться тригер, залежить від сигналів на входах тригера і від його попереднього стану, тобто він має пам'ять. Таким чином, командами логічного рівня для цього і інших подібних пристроїв є ***значення*** (сигнали) 0 и 1 на входах R и S (рис. 3.10).

Слід пам'ятати, що логічний (вентильний) рівень уявлення тригера, на фізичному рівні може реалізовуватися транзисторами і резисторами за допомогою різних технологій і з різних, взагалі говорячи, матеріалів.

З моменту винаходу у 1959 року інтегральних схем МП або як їх щё звуть «чипів», лавино образно росте інтерес до створення на їх базі електронних пристроїв, які знатні виконувати складні функції управління. Після упровадження (рос.-внедрения) мікропроцесорів у виробництво кишенькових калькуляторів вони почали широко застосовуватися і в інших галузях.

Головними якостями цих пристроїв стали ***низька вартість*** і спроможність виконувати ***програми***. Іншими словами, мікропроцесор є логічним пристроєм, яке ***програмується*** *і* виготовляється у монолітному кристалі. Сам по собі МП не може вирішити ту або іншу конкретну задачу. Для її вирішення його потрібно запрограмувати, забезпечити необхідними даними і з'єднати з іншими пристроями. Наприклад, МП може управляти світлофорами на складному перехресті, забезпечуючи їх переключення через задані проміжки (рос.-промежутки) часу (рис. 3.11).

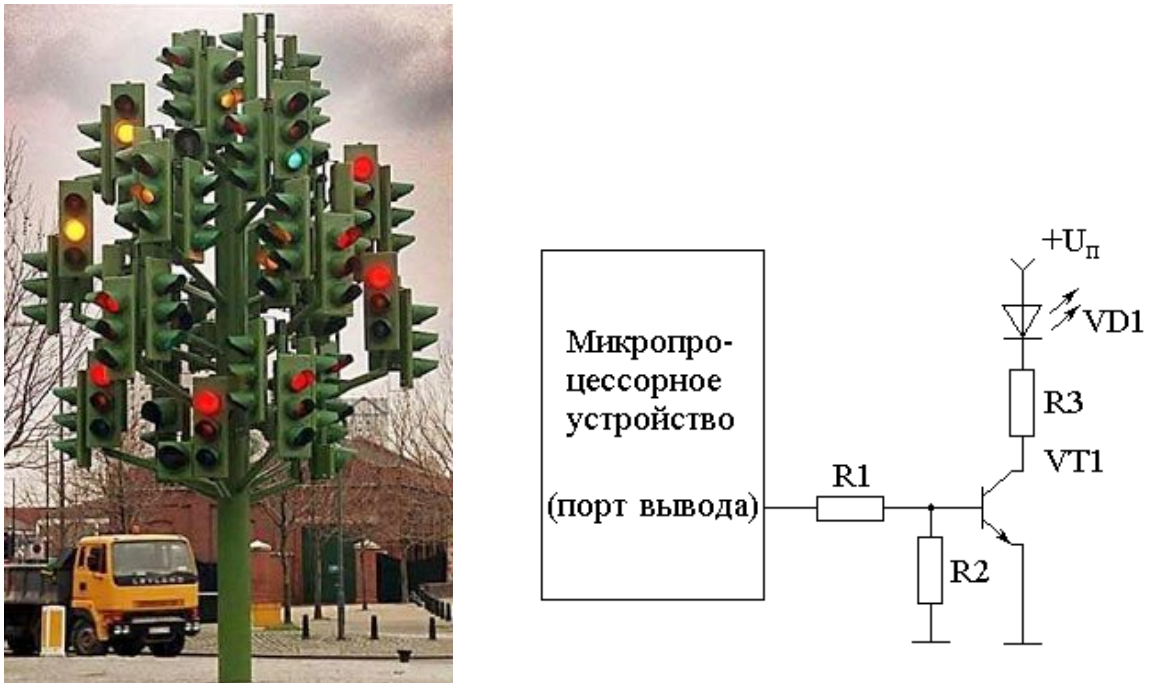


Рис. 3.11. Світлофор на складної ділянці дорожньої розв'язки у Англії, який управляється за допомогою мікропроцесора і схема типового його підключення

3.4. Команди програмних рівнів і рівнів роботи з ОС

Наступним, більш високим рівнем уявлення команд, є ***приписання (рос.-предписания), які визначають елементарні шаги виконання програмованих дій у роботі конкретних пристроїв***, наприклад, запис, зчитування, пересилання даних і так далі у регістрах і інших блоках процесора. Для їх обробки у інтегральних схемах служать ***програмний лічильник (рос.-счетчик) (лічильник команд), стек і регістр команд***. Зрозуміло, що ці команди також управляють і роботою усіх інших пристроїв та компонентів пристроїв ПК.

Команда мікропроцесора як правило складається з коду операції (КОП) і так званої адресної частини, де можуть указуватися адреси операндів у оперативної пам'яті (ОП, ОЗУ). Операндами звуться данні, над якими команда робить які нибудь дії. Як правило, у адресній частині команд вказуються не

самі операнди, а їх адреси в операційних пристроях ПК (регістрах, суматорах, оперативній пам'яті і ін.).

Формати команд достатньо сильно залежать від конструкції процесора. Для процесора, який розроблено по архітектурі[12] Фон-Неймана, котрій відповідають всі великі чипи фірми Intel, команди будуються наступним чином (рис. 3.12).

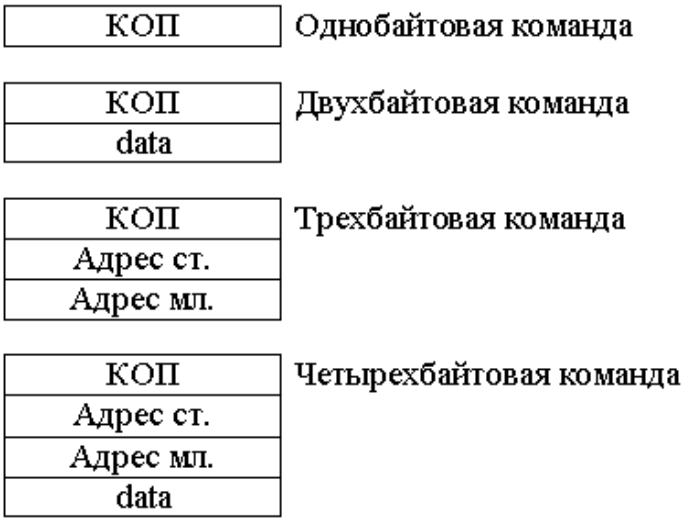


Рис. 3.12. Побудова команд процесорів Intel

Внутрішні команди процесора є основою ***архітектури обчислювальної системи*** і, відповідно, ***комп'ютерної платформи***[13]. На їх базі будуються практично усі мови високого рівня (МВР), такі як C, Pascal, C++, Object Pascal, Java і багато інших. В мікропроцесорі команди мають цифрову кодировку(рис. 3.13), а дані обробляються у двоїчної, восьмирічній, шістнадцятирічній і десятинній системах обчислення (див. також Додаток 2).

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

Рис. 3.13. Фрагмент програми комп'ютера у цифровому вигляді (шістнадцятирічний код).

[12] Архітектура – опис обчислювальної системи на деякому загальному рівне, який включає опис користувальницьких можливостей програмування, системи команд і засоби користувальницького інтерфейсу, організації пам’яті і системи адресації, операцій уводу-виводу і управління и т.д.

[13] Платформа – сукупність апаратних засобів, програмного забезпечення і інтерфейсів, які використовуються у конкретних комп'ютерах. Як правило, платформа визначається операційною системою, яка застосується, а також і процесором.

Таким чином, кожної команді відповідає деякий цифровий код (наприклад, 01, 02 і т.д.), котрий спеціальним пристроєм ***дешифратором*** інтерпретується у відповідну послідовність елементарних дій (посилань (рос.-посылок), імпульсів) виконуючим пристроям (тригерам, суматорам, регістрам і ін.).

Якщо для коду операції використовується восьми бітне слово (байт), то за допомогою цього слова можна закодувати 256 операцій. У процесі розробки системи команд нового пристрою для операції може бути назначено аби який код. Саме системою команд і визначається конкретне сімейство процесорів. Однобайтні команди дозволяють робити з внутрішніми програмно доступними регістрами процесора. При цьому для виконання однієї і той же операції над ***різними регістрами*** процесора назначаються ***різні коди команд.***

С ціллю зручності (рос.-удобства) написання програм для конкретних пристроїв комп'ютера розробляються спеціальні мови (програмування) нижнього рівня (МНУ) – асемблери. Точно так же звуть і програму або пакет програм, які здійснюють трансляцію (перетворення) початкового (рос.-исходного) тексту програми (початковий (рос.-исходный) модуль) у машинні коди (об'єктний модуль). Далі, як правило, код, що отримано трансформується у загрузочний модуль, тобто програму у вигляді пригодному для завантаження (рос.-загрузки) і виконання. Такий файл містить програму у машинному коді, інформацію для настройки адресів у пам’яті ПК і має розширення імені **.**EXE.

У асемблерних програмах, цифровим кодам команд, доводяться у відповідність (рос.-соответствие) мнемонічні (текстові) позначення (наприклад, ADD - скласти, LOOP–цикл, MOV–переслати і т.д.) з відповідними адресними частинами. Таким чином, для повного позначення (рос.-обозначения) команди використовуються мнемонічне позначення операції і символічні імена операндів, що вона використовує, котрі перелічуються (рос.-перечисляются) через кому. При цьому, у більшості процесорів операнд приймач (рос.-приёмник) даних записується першим, а операнд джерело (рос.-источник) даних – другим.

Наприклад:

```
MOV   R0, A
```

; Скопіювати вміст (рос.-содержимое) регістра A у регістр R0

Інструкція (команда) складання вмісту двох регістрів для 16-розрядного процесора Intel 80286 на мові асемблера виглядає наступним чином:

```
ADD   AX,BX
```

; Скласти вміст регістрів AX и BX

Команда організації циклу має вигляд:

```
LOOP  MET1
```

; MET1 – ім'я мітки начала циклу

Команда зсуву (рос.-сдвига) вліво на 4 біта вмісту (рос.-содержимого) регістру AX показана нижче.

```
SHL     AX,4        ; AX ім'я регістру
```

Важність мови асемблеру полягає ще і в тому, що, наприклад, в дійсності команда пересилки даних MOV – це ціле сімейство машинних команд мікропроцесора. Для різних типів адресів і операндів, існує від семи (Intel 8080) і більше різних варіантів даної команди. Асемблер породжує правильний машинний запис, основуючись на типах операндів, котрі вказують програміст. И це одна з причин, по котрій асемблер потребує для операндів зазначення типів, тобто асемблер повинен знати, що уявляє (рос.-представляет) собою кожний операнд – регістр, байт пам'яті, слово пам'яті, сегментний регістр, або щось інше.

Основою формування програм для виконання на комп'ютерах задач з різних предметних областей є мови (програмування) високого рівня (МВР). У їх структуру обов'язково входять ***команди обробки даних, які іменуються також операторами програми або пропонуваннями (рос.-предложениями) мови програмування***. Сюди можна віднести оператори (команди): циклу, умовні, вибору і деякі інші. Кожний з таких операторів реалізується у комп'ютері у вигляді набору базових машинних команд процесора. Переклад кожного оператора і всієї програми у цілому у набори відповідних машинних команд здійснюється ***компілятором*** або ***інтерпретатором*** мови програмування, яка використовується.

Компілятором зветься програмний засіб для перетворення тексту програми з опису на вхідній мові (мові програмування) у її уявлення на вихідної мові (у машинних командах). Програмна система, яка аналізує команди або оператори програми і негайно (рос.-немедленно) виконує їх зветься інтерпретатором. Достатньо часто переклад програми при трансляції здійснюється спочатку на мову асемблера, а потім вже у машинні команди.

Наприклад, розглянемо простий математичний вираз:

f(n) = n! (обчислити значення факторіала числа *n*)

Операторна реалізація його в вигляді фрагмента програми на мові Турбо Паскаль буде виглядати наступним чином:

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

Команди (інструкції) мови ТП

Фрагмент програми на мові Турбо Паскаль, який реалізує обчислення факторіала числа *n* и занесення цього значення у змінну Fct (ячейку з ім’ям Fct)

Так як роботу і взаємодію користувача з пристроями комп'ютера організує ***операційна система***, вона повинна враховувати, що на кожній комп'ютерній платформі існує свій, ***стандартний набір команд машинного рівня***

для управління обчислювальною системою (комп'ютером) у цілому. У даному контексті ***командою*** є припис (рос.-предписание) комп'ютеру або пристрою виконати визначений (рос.-определённый) крок на путі рішення задачі. Нижче приведено фрагмент наведеної вище програми обчислення факторіала на мові асемблера для процесора платформи Intel (рис. 3.14)..

	Код операції	**Адресна частина**	**Коментарі**
START:	mov	i, 1	*; **Заносимо значення** 1 **у комірку (рос.- ячейку)** i*
	mov	Fct, 1	*; **Заносимо значення** 1 **в комірку** Fct*
	mov	CX, n	*; **У регістр** CX **заносимо** n*
	sub	CX, i	*; **Обчислюємо різницю і приміщуємо (рос.-помещаем) значення** n-i **у** CX*
	mov	AX, Fct	*; **У регістр** AX **заносимо значення** Fct*
	mul	i	*; **Там же у** AX **множимо** Fct **на** i*
	mov	Fct, AX	*; **Повертаємо добуток (рос.- призведе-ние) в** Fct*
	loop	START	*; **Якщо вміст (рос.-содержимое)** CX **не дорівнює нулю, оператор** loop **віднімає з нього одиницю і переходить на мітку** START*
	...		*; **У протилежному випадку цикл завершується і управління передається наступної за циклом команді***
Примітка: mov, sub, mul і loop – команди мови асемблера.			

Рис. 3.14. Фрагмент програми обчислення факторіалу на мові асемблера

На відміну від команд різних програмних рівнів – команди операційної системи і додатків, які працюють під її управлінням, мають різноманітну зовнішню форму, хоча практично всі також працюють з пристроями ПК (!).

У командній стрічці операційної системи MS–DOS команди уявлені (рос.-представлены) спеціальними (ключовими) словами, котрі розуміє програма або система (наприклад, команди MS DOS: *MD, CD, COPY, DIR* і т.д.) (рис. 3.15).

Ряд важливих системних команд ПК уявлений (рос.-представлен) функціональними клавішами (Tab, Esc, Shift, Ctrl, Alt) або їхніми сполученнями (рос.-сочетаниями) (Ctrl+Alt+Del, Ctrl+Tab і т.д.).

Команди різних версій операційної системи Windows реалізуються у вигляді піктограм і елементів, розташованих у різних частинах екрану, що обираються з меню і списків, які розкриваються на клацання мишою на відповідних управляючих елементах (рис. 3.16).

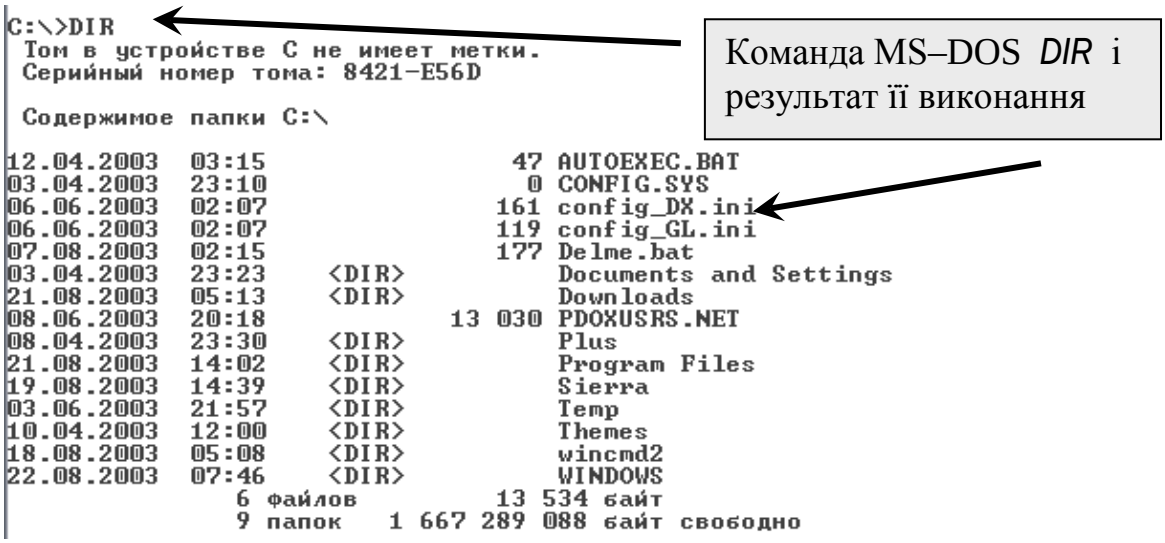


Рис. 3.15. Результат виконання команди MS DOS *DIR*.

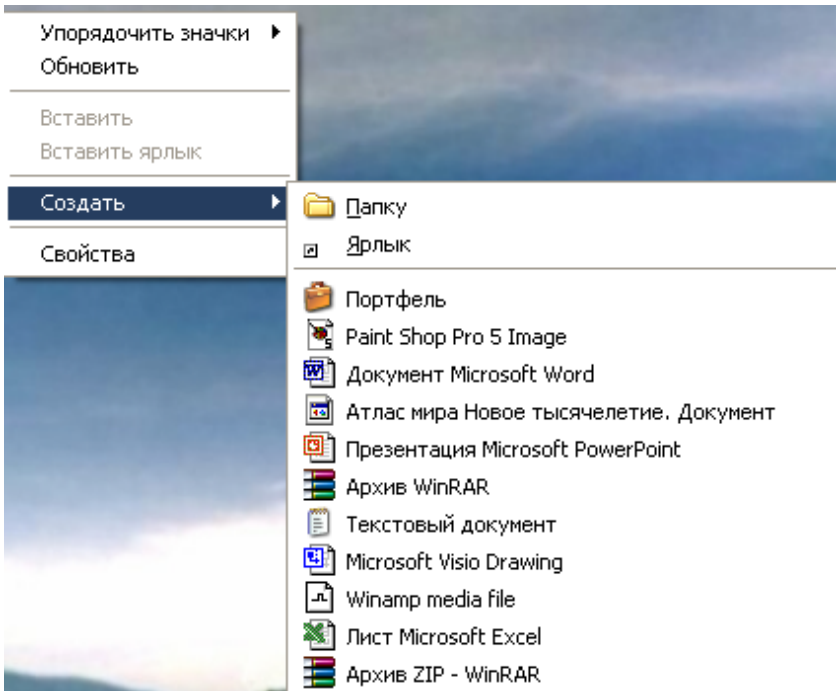


Рис. 3.16. Команди списку команди ***Создать*** контекстного меню робочого столу ОС Windows

Команди додатків ОС Windows реалізуються у вигляді кнопок, команд меню і багатьма іншими графічними об'єктами (комбобокси, чекбокси, радіокнопки і ін.), що розташовані на екрані ПК (рис. 3.17, 3.18).

Рис. 3.17. Кнопки стандартної панелі інструментів додатків MS Office

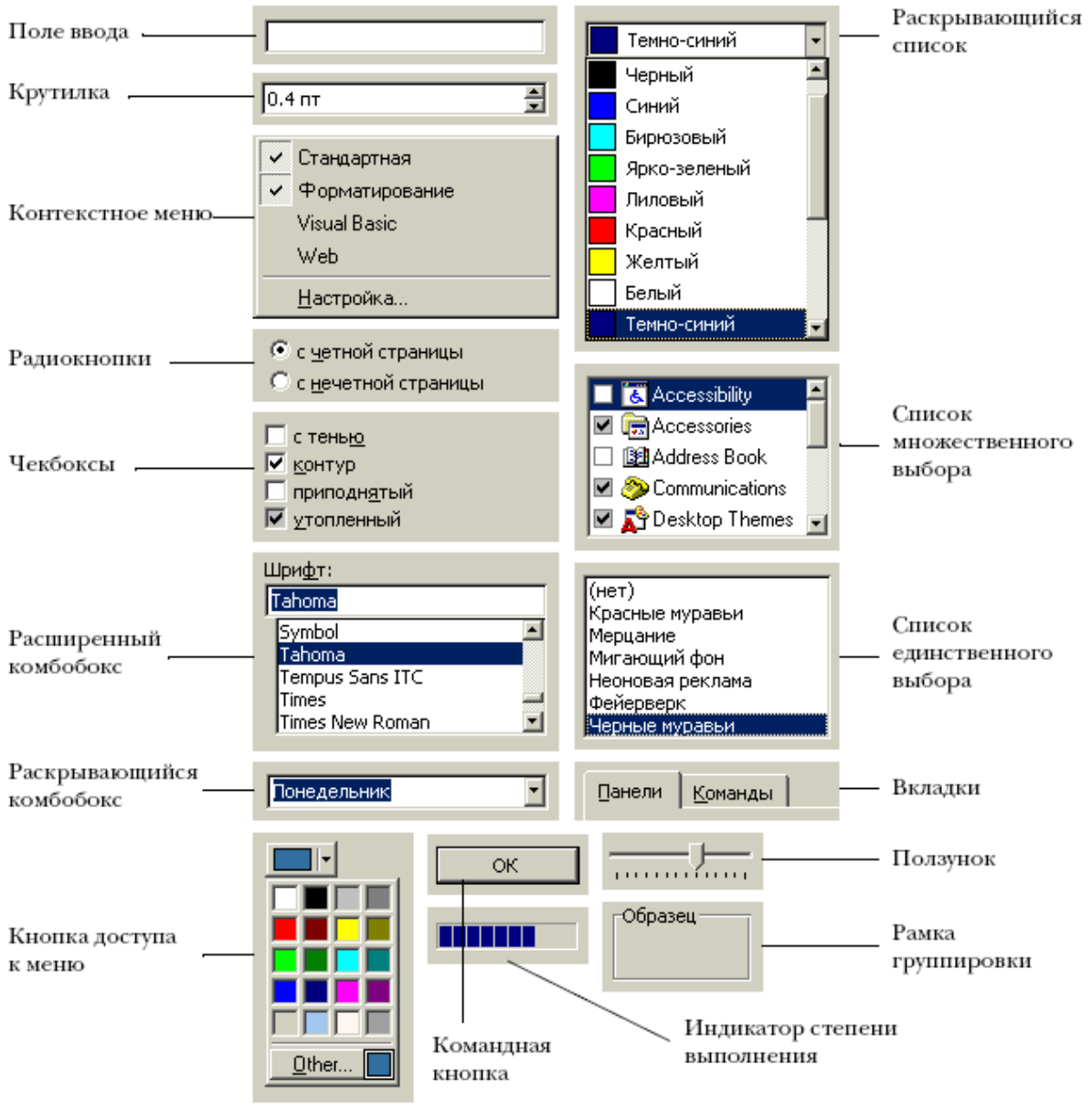


Рис. 3.18. Вигляд різноманітних ***команд*** роботи з операційними системами і додатками, які виконані у стандарті графічного інтерфейса користувача (GUI)

Таким чином, ***набір команд***, який розроблен для роботи з додатками (програмами) під управлінням ОС, а також ***правила їх використання***, звуться у комплексі ***інтерфейсом користувача*** і може мінятися від програми до програми. Самі команди реалізуються різними засобами, але по своїй сутності є ***макрокомандами***, які містять велику кількість інших елементарних команд, а також звернень (рос.-обращений) до підпрограм і переривань (рос.-прерываниям). Чим інтерфейс є більш розвинутим з точки зору його зручності для користувача, тим більше він зважається (рос.-считается) «дружнім (рос.-дружественным)».

Усі дії і операції, котрі доступні у вигляді макрокоманд (тобто великих команд), функцій додатків і бібліотек програм, користувач може використовувати шляхом виконання простих клацань мишею на кнопках і

вибору команд з випадаючих списків і деяких інших, які передбачені інтерфейсом операційної системи дій. В таблиці 3.1 подані (рос.-представлены) деякі види команд, які використовуються і функціонують на різних рівнях ієрархії управління компонентами і об'єктами при розробці, використанні і роботі елементів комп'ютерних пристроїв.

Таблиця 3.1.

Команди в структурі ПК

№	Найменування команди	Рівень (логіч./ фізич.)	Пристрої, які управляються	Рівень уявлення (рос.-представление)
1	Імпульс	Фізичний	Транзистори, діоди	Електричний
2	Значення	Логічний	Вентилі	Логічний
3	Значення	Логічний.	Логічні діаграми	Логічний.
4	Сигнал	Логічний	Логічні схеми тригера і ін. пристроїв	Логічний.
5	Імпульс	Фізичний	Фізичні схеми тригера і ін. пристроїв	Електричний
6	Команда процесора	Фізичний	Елементи пристроїв комп'ютера	Електричний
7	Команда мови асемблера	Логічний.	Елементи пристроїв і пристрої комп'ютера	Логічний Текстовий Дані
8	Команда алгоритмічної мови МВР (оператор, інструкція)	Логічний.	Елементи пристроїв і пристрої комп'ютера	Логічний Текстовий Дані
9	Команда МВР з графічним інтерфейсом* (меню, кнопка и ін.).	Логічний.	Елементи пристроїв і пристрої комп'ютера, інші комп'ютери	Логічний Текстовий Графічний Дані
10	Команда управління операційною системою (текстова стрічка, сполучення клавіш, меню, кнопки, списки і ін.)	Логічний.	Пристрої комп'ютера	Логічний Текстовий Графічний. Дані

Примітка (*):
Мови високого рівня (МВР) з графічним інтерфейсом є, як правило, інтегрованими середовищами розробки (ІСР) (IDE-Integrated Development Environment) або засобами швидкої розробки додатків (RAD-Rapid Application Development). В останній час деякі з цих засобів стали називати мовами: Delphi, C#, Excel та ін. Окрім того, їх ще відносять до мов так званого четвертого покоління – 4GL-мови.

Все те, що уявляється ще невирішеним або не реалізованим у функціях існуючих операційних систем і додатків, або потребує додаткової автоматизації

або спеціальної настройки, як правило, програмується безпосередньо користувачем або професіональними програмістами. Таким чином, головний вектор напряму зусиль останніх направляється на програмування всіх тих необхідних в роботі функцій, котрі ще не реалізовані. І не дивлячись на колосальний ріст кількості різноманітних додатків (по деяким даним тільки до 1990 року для комп'ютерів Apple було розроблено більше 13 тис. прикладних програм), необхідність у застосуванні користувачами мов програмування різних рівнів не тільки не зменшилась, але й продовжує постійно зростати. Доречи, одним з важливіших прикладень (рос.-приложений) програмування є автоформалізація знань спеціалістів різних прикладних областей, яка приводить до проникнення комп'ютерів в галузі, де раніш це казалося неможливим.

Запитання.

1. Що таке управління?
2. Для чого потрібно управляти комп'ютером?
3. Наведіть приклади уявлень фізичних рівнів систем?
4. Наведіть приклади уявлень логічних рівней систем?
5. Які завдання виконують логічні імена пристроїв?
6. Наведить приклади фізичних команд?
7. Що грає роль команд у мовах програмування?
8. Які комади операційної системи Windows Ви знаєте?
9. Якими пристроями комп'ютера за допомогою яких команд можна управляти?

Інтерфейс – це іменована множина операцій, які описують поведінку елемента.

Комп'ютер – це інтерфейс доступу до цифрових даних.

Мова UML.

4. КОНЦЕПЦІЇ ІНТЕРФЕЙСУ

4.1.Хто і як управляє комп'ютером

Рис. 4.1. Незабаром з комп'ютером можна буде розмовляти

Починаючи працювати з ПК, як правило, людина не замислюється над тим, наскільки складний процес здійснюється при трансформації інформації у ланцюзі "думка – результат" (рис. 4.1, 4.2). Адже для того, щоб організувати процес рішення задачі обробки інформації, потрібно попередньо сформулювати "завдання машині" у термінах та діях, які зрозумілі самій машині, тобто програмно. І якщо інтерфейс поміж розумовими (рос.-мыслительными) процесами людини і його руховими (рос.-двигательными) функціями і органами природою організовано споконвічно (рос.-изначально), то з комп'ютером усе значно складніше, так як він складається з комплексу різнорідних та достатньо складних різноманітних електромеханічних пристроїв. Тому, доводиться на межі взаємодії не тільки поміж людиною і комп'ютером, але й на межах (рос.-границах) взаємодії між усіма його компонентами вирішувати проблеми узгодження їхньої роботи на рівні стандартизації і спряження відповідних інтерфейсів, як програмних, так і апаратних.

Рис. 4.2. Переходні перетворення інформації при її обробці на ПК

Під межами взаємодії, природно, розуміються ділянки простору, на яких відбувається передача даних від однієї сутності до іншої (наприклад, з'єднання (рос.-разъём) є ділянкою, де спрягаються вхідний штекер зовнішнього пристрою і гніздо виходу пристрою, до котрого підключається останнє).

Іншими словами, головна задача інтерфейсу – забезпечити ***двосторонню*** передачу даних, а значить і інформації.

Само по собі складне і багатофункціональне англомовне (!) поняття "***інтерфейс***" достатньо багатозначне. Енциклопедія пропонує наступні трактування і змістовні відтінки цього терміну (рис. 4.3).

ІНТЕРФЕЙС (як іменник (рос.-существительное) позначає):

1) область, де зустрічаються і взаємодіють дві системи;

2) інтерфейс користувача, який складається з набору кнопок, списків команд (меню), команд операційної системи, форматів графічних дисплеїв та інших пристроїв, які підтримуються комп'ютером або програмою. Графічний інтерфейс користувача (ГІК)(англійською мовою – GUI-Graphic User Interface) забезпечує більш менш "образно орієнтований" (picture-oriented) спосіб взаємодії з новими технічними засобами і технологіями. Звичайно, GUI більш зручний або дружній при роботі з комп'ютерними системами.

3) програмний інтерфейс складається з набору тверджень, функцій, опцій і інших засобів, які висловлюють програмні інструкції і дані, які передбачені програмою або мовою для використання їх програмістом.

4) фізичний або логічний інтерфейс підтримує з'єднання будь-якого пристрою з будь-яким з'єднанням або іншим пристроєм.

5) як глагол (з'єднувати, зв'язувати), означає взаємодію з іншим користувачем або об'єктом. По відношенню до апаратних засобів інтерфейс означає створення фізичного з'єднання, що дозволяє двом частинам обладнання сполучатися або працювати разом ефективно.

Рис. 4.3. Багатозначність поняття ***інтерфейс***

Рис. 4.4. Інтерфейс дороги – дорожні знаки

Щоб краще зрозуміти зміст і деякі окремі значення цього терміну розглянемо інтерфейс автомобільної дороги (рис. 4.4). Розроблений сумісними зусиллями спеціалістів різних країн світу він зрозумілий усім автомобілістам тому, що система дорожніх знаків вивчається у час проходження відповідних курсів. Дивлячись на той або інший дорожній знак водій розуміє, що чекає його далі на дорозі і розуміє, які дії необхідно виконувати у час руху по різним її ділянкам. Більш того, проїжджаючи на автомобілі по автостраді у любій країні, йому нема необхідності знати мову цієї країни бо загальні для водіїв усіх

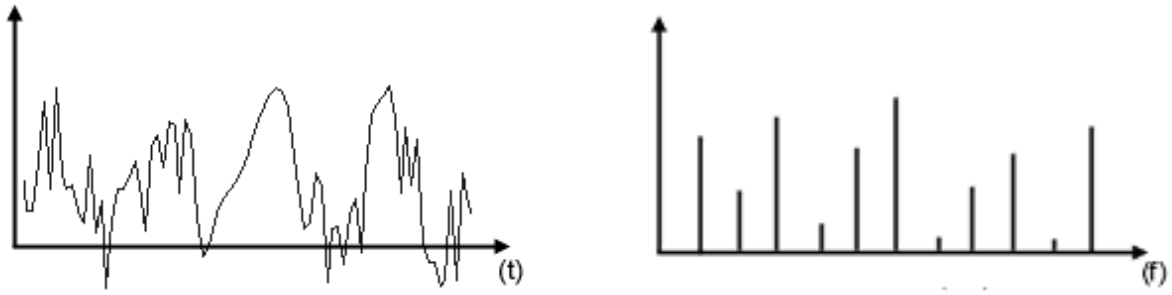
країн дорожні знаки і є мовою взаємодії з системою під назвою "дорога".

Природа комп'ютера значно складніша багатьох інших систем і об'єктів, з якими людині приходиться взаємодіяти. До того ж вона є надзвичайно динамічною у багатьох відношеннях.

По-перше, конфігурація і склад пристроїв ПК надзвичайно мінливі. Сюди ж потрібно віднести різноманітність та багатофункціональність програмних компонентів, які використовуються.

По-друге, природа, масштабі змін у формі існування і розміри об'ємів даних і інформації, якими обмінюються користувач и комп'ютер іноді важко піддаються осмисленню і формалізації (на даному етапі розвитку науки).

У самому загальному наближенні (рос.-приближении), можна сказати, що ***даними для комп'ютера являються зареєстровані об'єктивні сигнали або факти.*** І ті і інші уводяться в його пам'ять у вигляді послідовностей цифрових дискретних (а не аналогових!) сигналів (рис. 4.5), або текстових символів з деякого їх носія або іншого пристрою (клавіатури, жорсткого або гнучкого диску, CD-ROM, бази даних, цифрових або аналогових датчиків та ін.).



а) **б)**

Рис. 4.5. Приклад аналогового сигналу (а) і
його дискретного (цифрованого) образу (б)

Прикладами сигналів можуть служити дані, які передаються супутниками з планет Сонячної системи, куди вони були послані. Сюди ж можна віднести і фотографії у цифровому вигляді, і температурні режими на поверхні планет, які досліджуються і багато іншого.

Прикладами зареєстрованих фактів можуть служити результати перепису населення або зведення (рос.-сведения) про геологічні характеристики Землі, визначені у конкретний проміжок (рос.-промежуток) часу і уведені в комп'ютер. У свою чергу, кожна дискретна величина (дискретний знак, цифра або число) уявляються у схемах і пристроях комп'ютера також у вигляді дискретних двоїчних сигналів 0 і 1 (рис. 4.6).

Інформацією ж, являється ***суб'єктивна динамічна складова*** результату обробки даних.

Взагалі кажучи, ***інформацією називаються зведення (рос.-сведения), які невідомі до їх отримання і є об'єктом зберігання, передачі і обробки***. Тобто інформацією є ***динамічний результат*** застосування суб'єктивних методів обробки до об'єктивних даних, який ***усвідомлений одержувачем (рос.- осознан получателем)***. Якщо результату (осознання) немає, це значить, що інформація

не сприйнята. Наприклад, щоб узнати, як працює той чи інший складний пристрій, необхідно відкрити інструкцію и почати її ***читати*** (динамічна складова процесу отримання інформації). Однак, якщо інструкція надрукована на китайській мові, ніякої інформації людині отримати не вдасться, якщо вона не володіє цією мовою (рис.4.7).

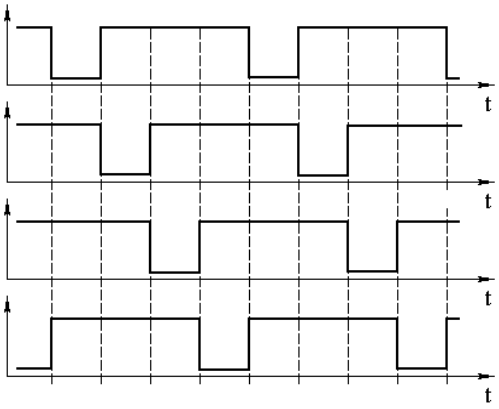


Рис. 4.6. Уявлення дискретних двоїчних сигналів 0 и 1, яке реалізується у вигляді синхронних імпульсів (0-нема імпульсу, 1-є імпульс).

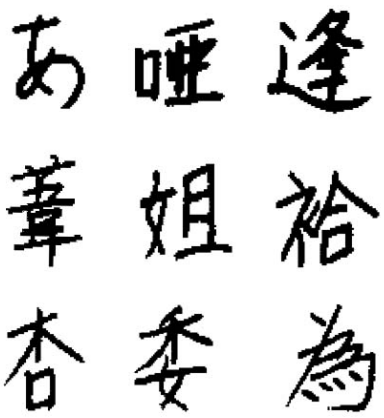


Рис. 4.7.
Китайькі ієрогліфи (складні у перекладі і вимові).

Дані завжди об'єктивні, майже якщо і несуть у собі неправдиву інформацію. У якій би формі вони не були б виражені, це завжди зареєстровані сигнали, тобто об'єкти матеріальної природи. А ось інформація може бути як об'єктивною, так і суб'єктивною, як істинною, так і неправильною – вона не являється об'єктом матеріальної природи. Властивості інформації залежать не тільки від змісту даних, але і від властивостей методів, за допомогою яких вона отримана, і більш за все від погодження (рос.-согласования) уявлення інформації з можливостями одержувача (реципієнта).

При всьому цьому не треба забувати, що практично уся інформація, що зараз накопичується на сучасних носіях інформації запам'ятовується, зберігається а потім використовується у вигляді нулей і одиниць. Тому, комп'ютер є інтерфейсом доступу до використання усієї цієї безлічі даних, які продовжують постійно накопичуватися! (рис. 4.8).

Які б дані не уводилися у комп'ютер: текст на любій мові, звук, зображення, фотографії, дані про різноманітні об'єкти Землі– все це перетворюється в процесі уводу у комп'ютер тільки у ***нулі і одиниці***. У такому ж вигляді все це знаходиться на жорстких і гнучких магнітних і лазерних компакт- дисках і обробляється у пристроях ПК програмами, які уявлені теж у вигляді цифр 0 і 1.

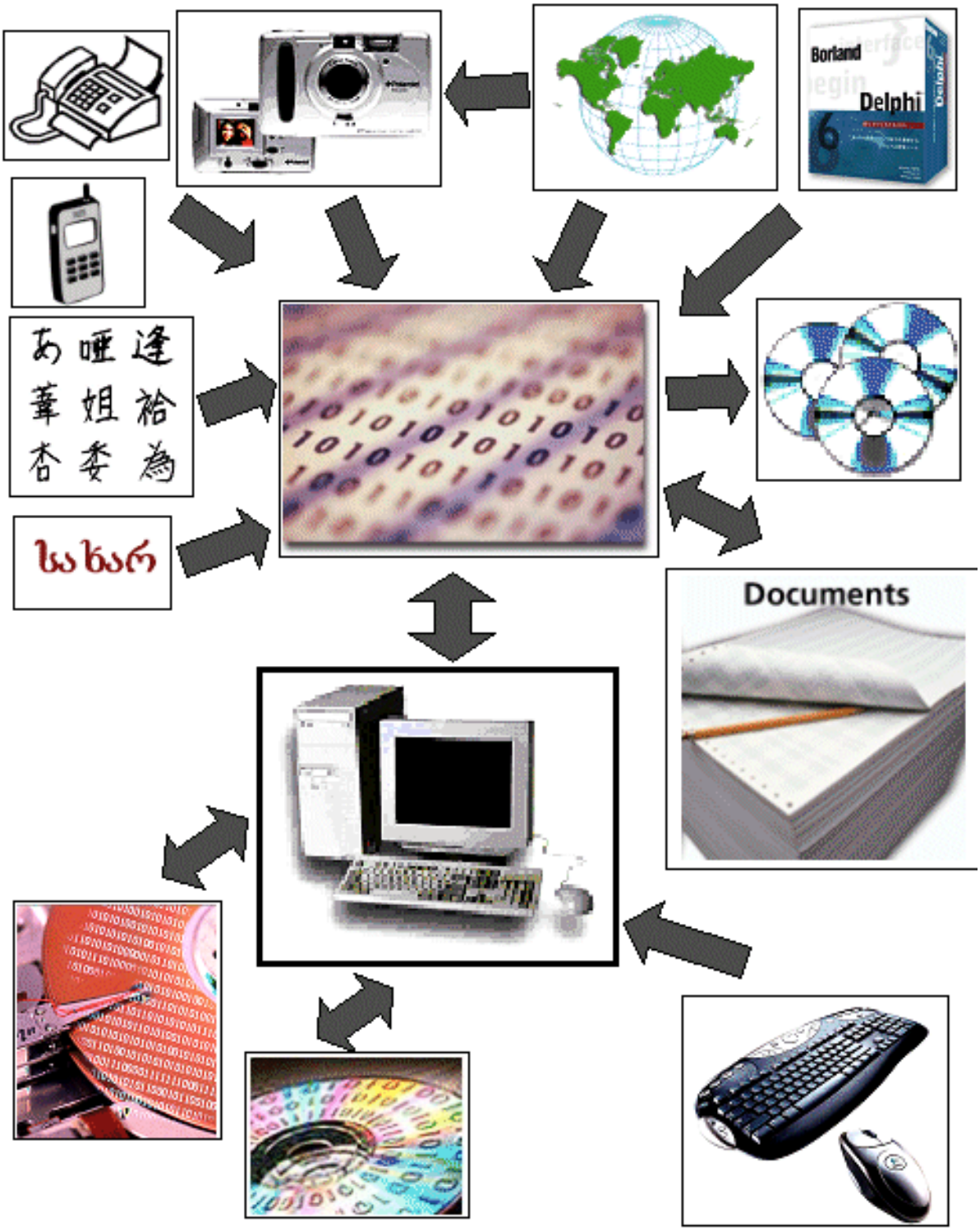


Рис. 4.8. Двоїчні цифри *0* і *1* – універсальний хоронитель й передавач інформації і даних

Об'єм інформації необхідний для нормального функціонування сучасного суспільства росте приблизно пропорційно квадрату (!!!) розвитку промислового потенціалу ведучих країн світу. Доля робочої сили, що зайнята питаннями накопичування і обробки інформації вже давно і багатократно перевищує долю робочої сили, зайнятої безпосередньо у матеріальному виробництві.

У даному контексті, ***процес взаємодії користувача з комп'ютером*** у самому загальному вигляді можна уявити наступним чином (рис.4.9). Користувач уводить інформацію для себе (для запису на диск, обробки і т.д.), а також і для управління комп'ютером, чергуючи (рос.-чередуя) дані і команди. Комп'ютер, виконуючи вказівки, виконує завдання, відображаючи для користувача на екран дисплея результати роботи і різноманітні повідомлення (поточний стан, повідомлення про помилки, підказки і інш.)

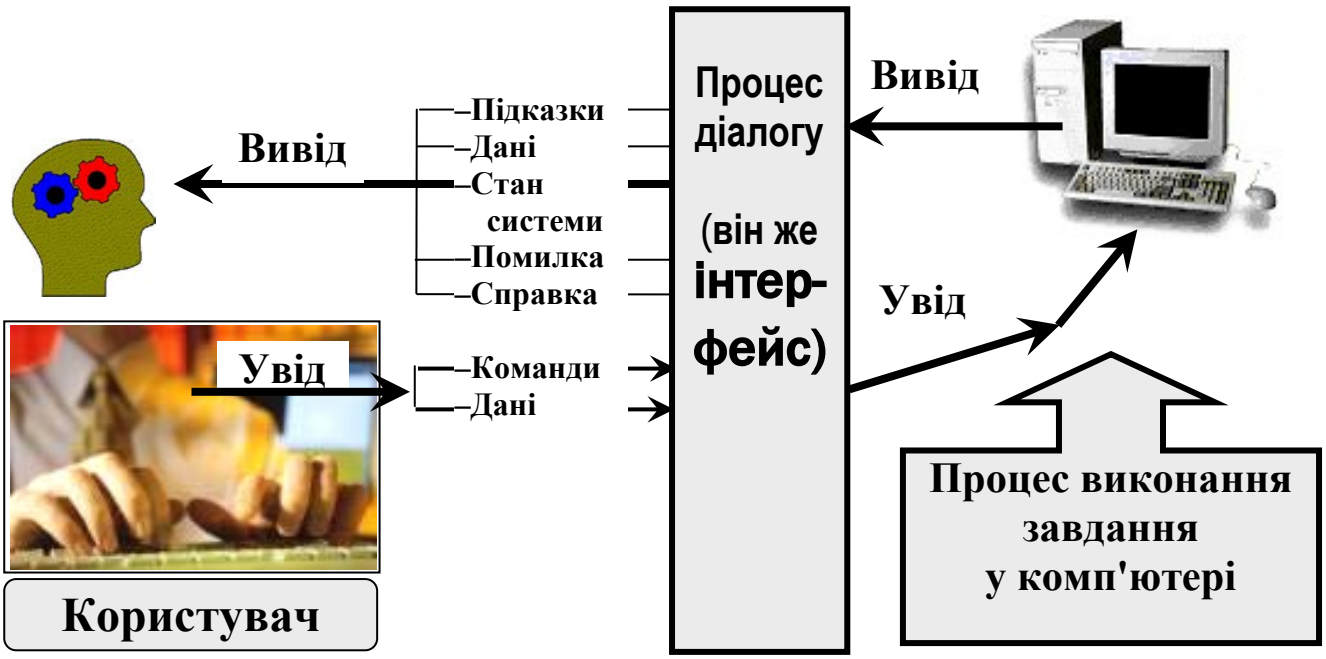


Рис. 4.9. Процес діалогу користувача з комп'ютером

Що головне у цьому процесі? Щоб відповісти на це питання, потрібно чітко уявляти о якої взаємодії йде мова. Очевидно, що у випадку адекватної реакції з боку другої системи, необхідна присутність наступних концептуальних складових:

❶ наявність інформаційних потоків з обох сторін;
❷ угода про протокол[1] обміну даними;
❸ додержання специфікацій[2] протоколу обміну даними;
❹ забезпечення інтерфейсу взаємодії у рамках увіду-виводу[3] з боку обох систем.

4.2. Принципи формування інтерфейсу користувача

Взагалі кажучи, UI (User Interface – інтерфейс користувача) у комп'ютерних системах складається з екрану, миші, клавіатури і підтримується операційною системою. У сучасних ПК інтерфейс користувача може складатися з динаміків, мікрофонів, а також, у випадку спеціалізованих

[1] Протокол обміну даними – набір правил і угод, які визначають формати даних і процедури передачі для обміну інформацією поміж системами (процесами), що взаємодіють.
[2] Специфікація – повний опис вимог.
[3] Увід-вивід – обмін даними під керівництвом комп'ютера.

пристроїв, з ручок їх управління, датчиків, що фіксують напрям погляду або чутливих до торкання місць на екрані. Такі екрани часто йменуються сенсорними і були вперше широко застосовані у копіювальних апаратах фірми Xerox, які практично не мали механічних і електричних елементів управління. Користувач бачив на рідкокристалічному екрані зображення кнопок з розшифровкою їх функціонального призначення. Натиснення пальцем у місці розташування потрібної кнопки викликало розкриття нового, наступного екрану з зображеннями кнопок більш низького рівня. І так далі. Тому вони значно перевершували по простоті управління апарати багатьох інших фірм, панелі управління котрих, були густо (рос.-обильно) усіяні багатьма кнопками і клавішами, число котрих іноді доходило до 80-ти.

На початку свого розвитку операційні системи забезпечували як пакетний, так і діалоговий режими роботи з користувачем. У пакетному режимі операційна система автоматично виконує задану послідовність команд, яка записується попередньо у так звані командні "бет-файли" з розширенням .BAT. Сутність же діалогового або інтерактивного[4] режиму полягає у тому, що операційна система знаходиться у стані очікування команди користувача і, отримав її, розпочинає виконання, а виконав, повертає відгук і чекає чергову команду. Властивість операційної системи перервати поточну роботу і відреагувати на подію, що визвана користувачем за допомогою управляючих пристроїв, сприймається нами як діалоговий (інтерактивний) режим роботи.

Сучасні операційні системи можуть одночасно забезпечувати інтерфейс командної стрічки і графічний інтерфейс користувача. У першому випадку пристроєм управління є клавіатура. Управляючі команди, котрі користувач повинен попередньо вивчити (!) і навчитися застосовувати, уводяться у поле командної стрічки, де їх можна також і редагувати. Відправка команди на виконання ОС здійснюється після натиснення клавіші Enter. Для комп'ютерів платформи IBM PC інтерфейс командної стрічки забезпечується сімейством операційних систем під загальною назвою MS-DOS (версій від 1.0 до 6.2).

За останні роки методи організації інтерфейсу у системі "людина–комп'ютер" отримали значний розвиток і набули визначеної логічної завершеності. Проте, в операційній системі ПК Windows і до наших часів збережено споконвічний інтерфейс командної стрічки, вікно котрої відкривається після вибору з меню[5] кнопки ***Пуск*** команди ***Командна строчка (Command prompt)*** (рис. 4.10).

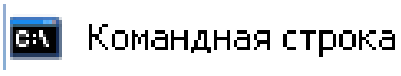


Рис. 4.10. Команда ***Командная строка*** з меню кнопки ***Пуск***

У вікні, що відкривається можуть виконуватися усі команди MS-DOS (рис. 4.11).

[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).

[5] Меню – список якостей об'єктів і команд, що до них застосується (тобто способів перетворення) у операційній системі і в працюючих під її управлінням додатків.

Графічний інтерфейс користувача забезпечується операційними системами Windows xx (різних версій), у котрих як орган управління окрім клавіатури може використовуватися миша або адекватний пристрій позиціювання. Доступ до пристроїв та їх властивостям може бути забезпечено з вікна ***Панель управления*** (Control panel) (рис. 4.12)

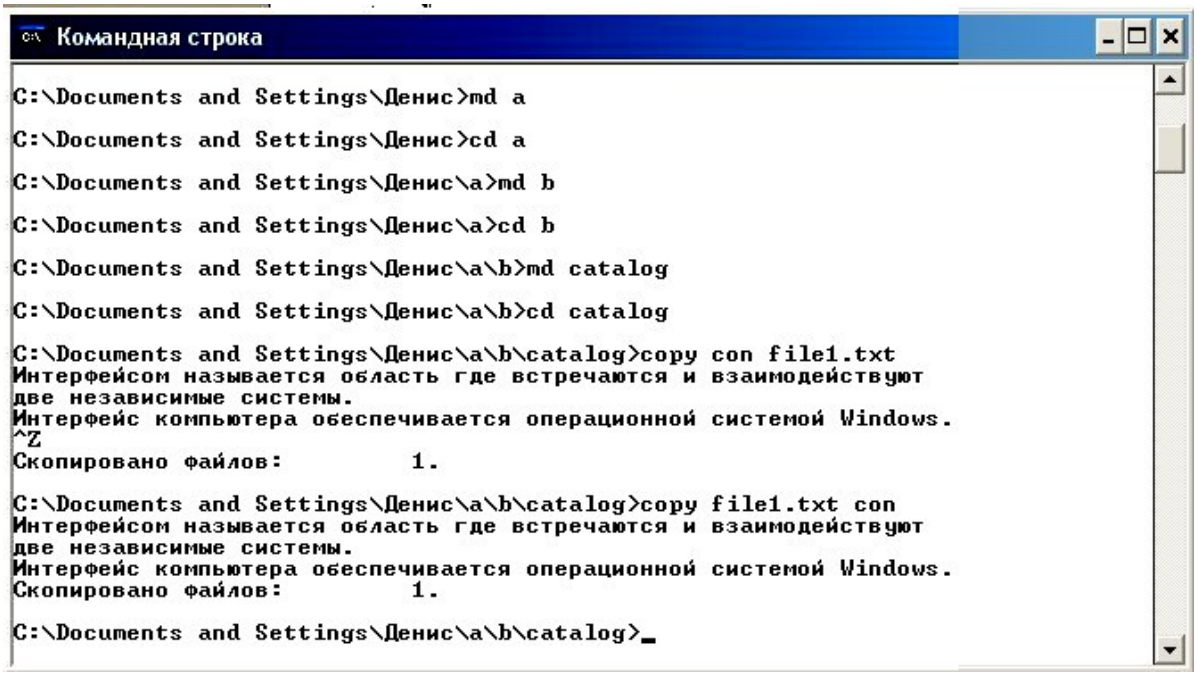


Рис. 4.11. Створення каталогів і текстових файлів у командній стрічці операційної системи Windows XP за допомогою команд MS-DOS (*md, cd, copy*) і логічного імені пристроїв (клавіатури і дисплея) *con*

Рис. 4.12. Інтерфейс доступу до пристроїв комп'ютера та їх властивостям в ОС Windows

Таким чином, робота з графічною операційною системою заснована на взаємодії активних і пасивних екранних елементів управління. Активним елементом управління є покажчик (рос.-указатель) миші (курсор) – графічний об'єкт, переміщення якого на екрані синхронізовано з переміщенням миші. Пасивними елементами управління роботою додатків (рос.-приложений) служать слідуючи графічні елементи управління: вікна ОС і додатків, екранні кнопки, значки, перемикачі (рос.-переключатели), прапорці (рос.-флажки), списки, що розкриваються, стрічки меню і багато іншого. Програмно, доступ до елементів інтерфейсу ОС здійснюється на рівні бібліотеки компонентів Windows API.

Для створення такого інтуїтивно зрозумілого і комфортного інтерфейсу спочатку необхідна ***метафора***[6], відштовхнувшись від котрої можна починати реалізацію функцій, які потребуються від нього. Адже для користувача "з вулиці", незнайомому з усіма тонкостями апаратних і програмних засобів комп'ютера, необхідно усі "болти і гайки" операційної системи сховати всередину зручної і привабливої графічної упаковки. В основу WIMP-інтерфейсів лягли три наступні метафори: "конкретний об'єкт" (WIMP), "що бачиш – то й отримаєш" (WYSIWYG) і "робочій стіл" (Desktop) (рис. 4.13).

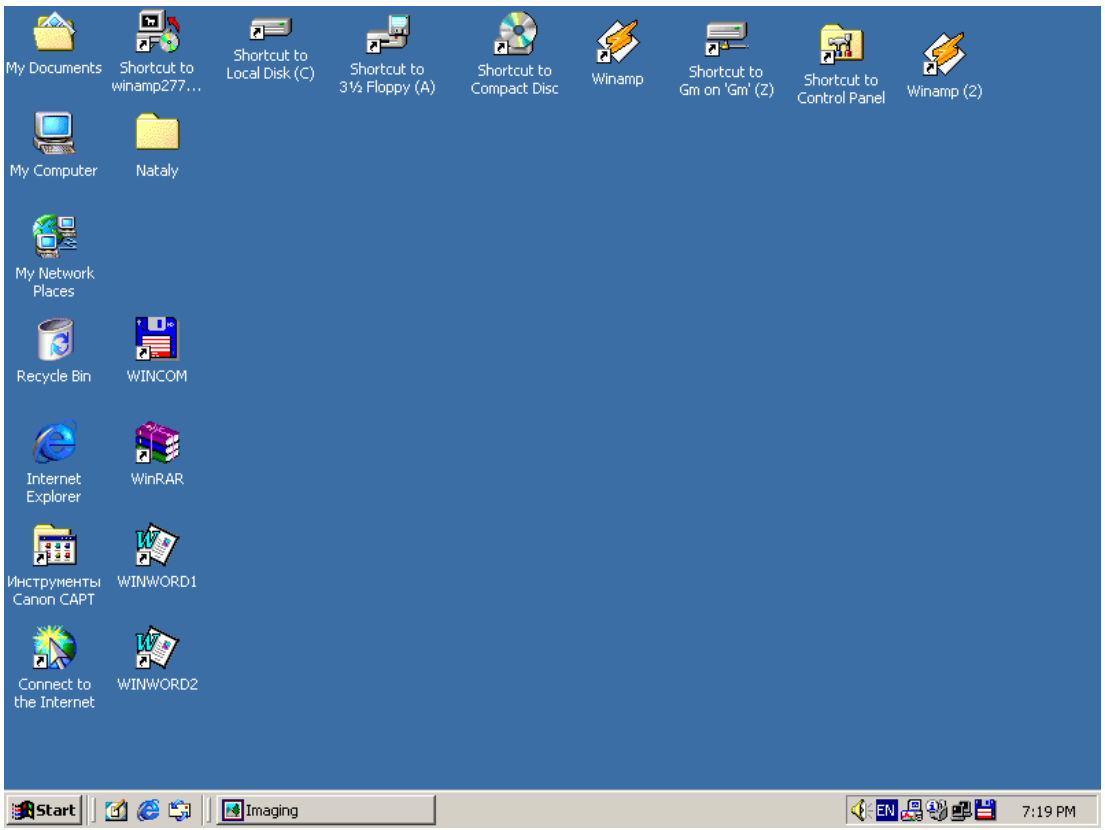


Рис. 4.13. Робочій стіл (Desktop) операційної системи Windows 2000 з піктограмами папок, а також зовнішніх та внутрішніх пристроїв

[6] Метафора – (від греч. metaphora – переніс (рос.-перенос)) – троп або механізм мови, який входить до ужитого (рос.-употреблённого) слова, що визначає (рос.-обозначает) деякий клас речей (рос.-предметов), явищ і т.і., для характеризування або найменування об’єкту, що входить у інший клас, або найменування іншого класу об’єктів, аналогічного даному у якомусь сенсі (рос.-смысле). У розширительному змісті термін "метафора" використовується до будь яких видів уживання (рос.-употребления) слів у непрямому значенні.

Абревіатура *WIMP* уявляє наступні поняття:

W (windows – вікна) – інформація візуалізується користувачеві на екрані дисплея у вигляді одного або декількох вікон;

I (icons – іконки[7]) – об’єкти, з котрими система і користувач мають справу і зображуються піктограмно[8] у вигляді іконок;

M (mouse – миша) – вибірка провадиться за допомогою маніпулятора типу "миша";

P (pop-up – випливати) – означає розкриття контекстних[9] меню, котрі автоматично випливають на екрані при клацанні правою кнопкою миші у будь який момент роботи з системою у будь якому місці екрану (рис. 4.14).

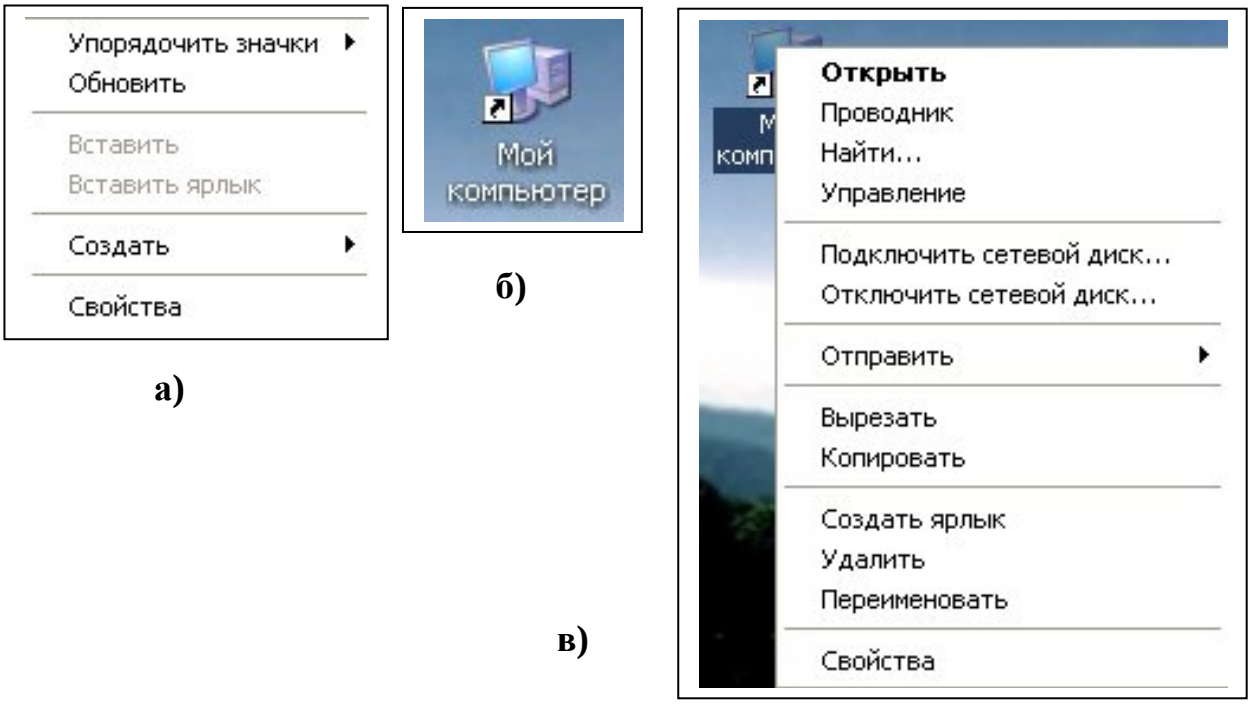


Рис. 4.14. Контекстне меню Робочого столу, яке розкривається (а). Об’єкт ***Мой компьютер*** (б) і відповідне йому контекстне меню (в)

Метафора "конкретний об’єкт" уведена для того, щоб користувач не вивчав ні семантику комп'ютерної системи (стрічки команд, об’єкти комп'ютерної системи і їх зв'язки з об’єктами задач, що вирішуються), ні синтаксис команд комп'ютерної системи, а користувався фізичними (графічними) уявленнями об’єктів їх піктограмами (іконками).

На робочому столі метафорою файлів є пачки бумаг або папок, а каталогів – ящики в картотеці, які видвигаються і т.д. Канцелярські операції

[7] Іконка – невід'ємний атрібут аби якої кнопки або файлу в операційній системі Windows, що дозволяє легко розпізнати тип об'єкту або конкретний об'єкт. Більш точніше тип файлу визначається за його розширенням (.DOC, .EXE и т.д.). Значки можуть хранитися у окремих файлах з розширенням .ICO, у програмних файлах (.EXE), у динамічно формуємих бібліотеках (.DLL) и т.і.

[8] Піктограма – графічне уявлення (зображення) об'єкта на екрані комп'ютера (аналог – іконка). Піктограми мають сам комп'ютер (значок «Мой компьютер»), файли, логічні диски, принтери і т.і.

[9] Контекстне меню – меню, що відкривається операційною системою або додатком у результаті клацання правою кнопкою миши по деякому об'єкту. Такі меню містять команди, які використовуються у контексті операційної обстановки у даний момент роботи з даним об'єктом.

припускають (рос.-предполагают) наявність фізичних дій над цими об'єктами. Так, для зберігання файлу людина збирає пачку ділових бумаг, підходить до шафи для зберігання, відкриває ящик і кладе пачку всередину. При перейменуванні файлу людина постирає старе ім'я і напише нове, а запис файлу у новий каталог може бути виконана шляхом перетаскування його за допомогою миші на іконку потрібного каталогу и т.д.

Метафора WYSIWYG (What You See Is What You Get-"що бачиш – то й отримаєш") описує такі інтерфейси, у котрих фактично ефект аби якої дії миттєво відображається на дисплеї. Це значить, що екран повинен імітувати засоби друку, і якщо користувач хоче надрукувати частину тексту курсивом, він і на екрані повинен набрати цей текст курсивом. Якщо файл знищується, користувач бачить, що файл зникає з відображеного на екрані списку файлів, тобто інтерфейс ефективно забезпечує інформацію про статус об'єкта, підтверджуючи, що дія була виконана.

Метафора "робочій стіл" дозволяє користувачеві мати перед собою усі необхідні для роботи документи і легко переключатися з одного на інший, тобто "тасувати папери на столі" і здійснювати з ними всі необхідні дії. На такому столі легко уживаються засоби для роботи з текстами, електронні таблиці для фінансового планування і різні допоміжні засоби: калькулятор, годинник, щоденник, записна книжка и т.д.

Інтерфейсом доступу до об'єктів і компонентів комп'ютера служать їхні ярлики[10], які уявляються операційною системою відповідними іконками. Розташовані у необхідних місцях на робочому столі, у каталогах та інших місцях доступу вони дозволяють користувачу оперативно зв'язувати різнорідні компоненти комп'ютера. Наприклад, операційна система забезпечує можливість, коли переміщення (буксирування) ярлику документа текстового редактора Word (розташованого одночасно і у деякому каталогу і на Робочому столі Windows (теж моделі каталогу!)) на іконку принтера приведе до друку файла (рис. 4.15). Природно, якщо принтер підключено до даного ПК!

Рис. 4.15. Перетаскування іконки документа на іконку принтера з метою друкування документа

Перетаскування іконки того ж документа на іконку флоппі-дисководу здійснить його запис на дискету (якщо вона уставлена у щілину (рос.-щель)

[10] Ярлик – файл, який містить покажчик (рос.-указатель) (посилання (рос.-ссылку) на деякий об'єкт в дереві ресурсів – інший файл, папку, принтер и т.і. Забезпечує безпосередній доступ до об'єкту з іншої папки, особливо з робочого столу. Має розширення .LNK і розпізнається по загнутій стрілці у лівому нижнему вуглу його значка. Роль значків виконують також PIF и URL файлы.

(слот) дисководу). Один простий рух клавіші миші ініціалізує у надрах ОС послідовність програмних дій для реалізації зчитування файла з жорсткого диска у пам'ять, а потім перепис його з пам’яті на гнучкий диск.

Інтерфейсом доступу користувача до інформації, розташованої у файлах здійснюється на рівні розширень їх імен. При інсталяції чергового програмного продукту, відповідні до нього розширення (.DOC, .XLS, .EXE и т.д.) заносяться у системний реєстр і їм у відповідність ставляться додатки (продукти), що їх породили: .DOC – додаток MS WORD, .XLS – додаток MS Excel і т.д. У каталогах, де вони розміщаються, таким файлам приформовуються відповідні значки (рис. 4.16).

Рис. 4.16. Значки файлів додатків MS Excel, MS Visio и MS Word у каталогах Windows

Після подвійного клацання мишою по значку (і імені) необхідного файла операційна система у відповідності з записом в реєстрі викликає функціонально відповідальний за роботу з даним файлом додаток. Воно відкриває файл і відображає його зміст у форматі додатка: документ, таблицю, музику, відеофільм і таке інше. Програми, що підтримують декілька форматів файлів даних (а всього в світі налічується більш за 5 000 різних форматів) вказує це в реєстрі також. Доречи архіватор файлів RAR має можливість розкривати також і файли, що внесені у архів у форматі ZIP.

Якщо користувач клацне мишою по файлу, розширення якого невідомо операційної системі, то на екран буде виведено вікно діалогу, де потрібно самостійно підібрати додаток, котрий може «справитися» зі змістом (рос.-содержимым) невідомого системі файла (рис. 4.17).

Початкові (рос.-исходные) (текстові, записані у ASCII-кодах) файли інтегрованого середовища розробки Турбо Паскаль у комп'ютері з ОС Windows, де розгорнуто середовище швидкої розробки Delphi, представляються у вікнах зі значками початкових текстів цього середовища (Delphi Source File), а початкові тексти Delphi – зі значками файлів Delphi Project (рис. 4.18).

Програми на мові Турбо Паскаль з деякими обмеженнями можуть відкриватися і виконуватися у вікні консольних додатків (Console Application) середовища Delphi. Значно простіше виконуються у цьому вікні аналогічні або більш складні по функціональності програми на мові Object Pascal (мові самого середовища). Для кодування програм необхідно виконати у Delphi послідовність наступних команд: *File|New|Other* і у вікні діалогу *New Items* обрати об'єкт *Console Application*. В той же час фізичний інтерфейс поміж пристроями ПК забезпечується різноманітними конекторами[11], штекерами, гніздами, сокетами[12] і слотами[13] (рис. 4.19).

[11] Конектор – з'єднувач багатоконтактний, (штепсельний) роз'єднувач (рос.-разъём). Засіб з'єднання взаємозаміняємих части (компонентів) комп'ютера. Доречі, шосте покоління процесорів Pentium відрізняється

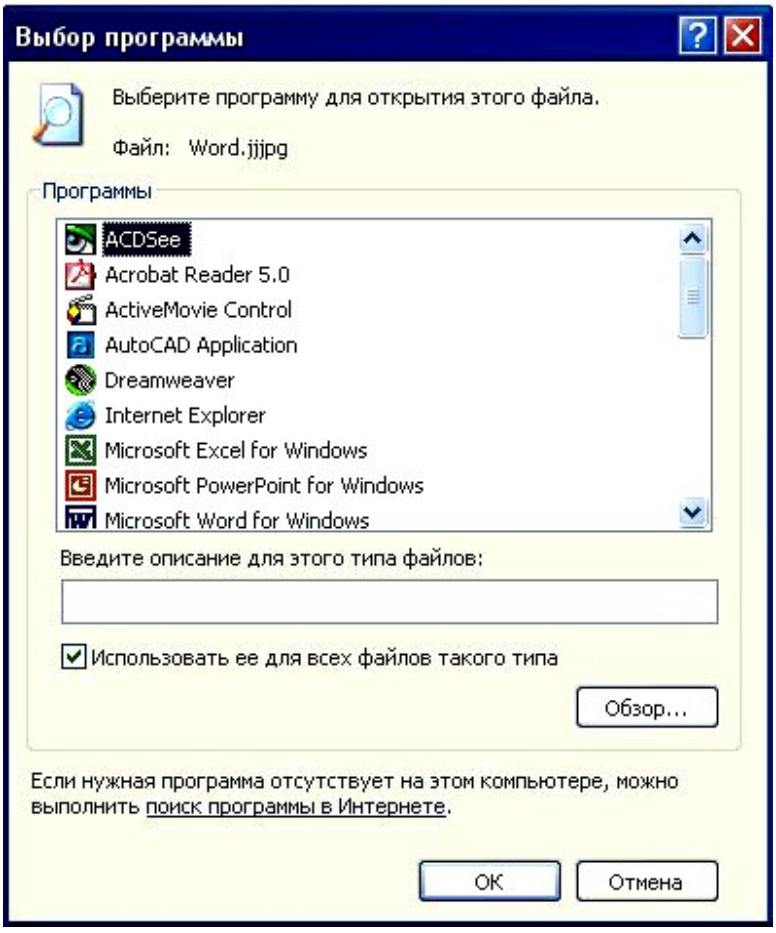


Рис. 4.17. Вікно діалогу операційної системи Windows XP для пошуку «невпізнаного» файла з невідомим розширенням імені.

а) **б)** **в)**

Рис. 4.18. Іконки (піктограми) файлів:
а) драйвера кірілізатора KEYRUS з розширенням .COM,
б) тексту ТП KURSOWRT з розширенням .PAS і
в) тексту консольного додатка WWProject2 з розширенням .DPR

великим різноманіттям раз'єднувачів-конструктивів. Одних тільки конекторів існує 4 типи: сокет 8, слот 1, слот 2 і сокет-370. Потрібно відмітить, що у технологіях фірми Intel, виробника ціх марок процесорів, терміни слот и сокет використовується в більш широкому розумінні. Вони позначають (рос.-обозначают) специфікацію електричних, програмних і механічних інтерфейсів.

[12] Сокет – гнездо; (з'єднувальна) панель; розетка (гнездова частина р'єднувального з'єднання).

[13] Слот —1) гнездо (у т.ч. і для плати), вхід, розетка; 2) роз'єднувач для елементів пам'яти. Як правило, ця назва використовується для роз'єднувачів, куди "уставляються (рос.-вставляются)" платы розширення, у тому числі модулі типу SIMM і DIMM. Роз'єднувачі, куди "утикаються (рос.-втыкаются)" ніжки (чипів або роз'єднувачів "протилежного полу"), звуться сокет (socket).

4.3. Конструкції і призначення фізичних (апаратних) інтерфейсів

Фізичні (апаратні) інтерфейси відіграють велику роль у процесі комплектуванні базової основи персонального комп'ютера. Вони визначають: типи стику (рос.-стыка), рівні сигналів, імпеданси, синхронізацию і багато інших параметрів каналів зв'язку (рис.4.19).

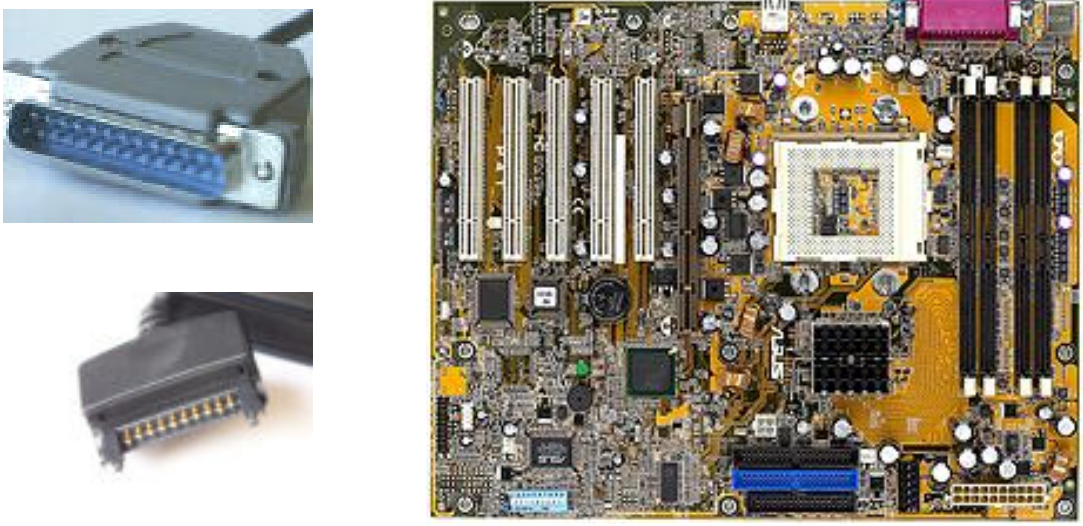


Рис. 4.19. Конектори, слоти і сокети ПК

Інтерфейс самого процесору з зовнішніми для нього, а також зовнішніми і внутрішніми для системного блоку пристроями зорганізовано через порти[14] увода–вивода. Звичайно порт уявляє собою точку підключення зовнішнього пристрою компьютера (принтера, сканера і т.д.) до внутрішньої шини процесора. Таким чином, програма або пристрій можуть посилати дані у порти або отримувати їх з портів для обробки процесором. Для багатоканальних пристроїв порт уявляє групу ліній увода-вивіда (шин[15]), по яким діється (рос.-происходит) передача інформації поміж центральним процесором (ЦП) і відповідним пристроєм уводу-виводу, як правило по одному біту на кожну лінію. Більшість пристроїв уводу-виводу інформації підключаються до комп'ютера за допомогою зовнішньої шини. Такі шини прийнято називати паралельними (LPT1) або послідовними (COM) портами. Природно роз'єднувачі (рос.-разъёмы) зовнішніх шин розташовуються на задньої панелі системного блоку комп'ютера. Велика кількість різноманітних зовнішніх пристроїв і відповідно до цього, різнотипних інтерфейсних механічних з'єднувачей, що багатократно виросла за останній час стає на перешкоді (рос.-затрудняет) користувачам і примушує виробників маркірувати відповідні вилки і розетки логотипами (значками або піктограмами), які їх ідентифікують. Наприклад, роз'єднувачі так званого порта USB (Universal Serial Bus –

[14] Порт – точка підключення зовнішнього пристроя до внутрішньої шини процессора. Таким чином, програма або пристрой можуть посилати дані у порти або отримувати їх з портів.

[15] Шина – група ліній електричних з'єднань, які забезпечують передачу даних і управляючих сигналів поміж компонентами комп'ютера.

Універсальна Послідовна Шина) мають логотип у вигляді структури, що розгалужується (рос.-разветвляется) (рис. 4.20). Сама система передачі даних USB була розроблена у 1995 році фірмами Compaq, DEC, IBM, Intel, Microsoft, NEC и Northern Telecom. USB порт практично знівелював існуючі раніш різниці поміж послідовним і паралельним портами ПК і має наступні характеристики:

❶ Швидкість передачі даних 1,5 - 12 Мбіт/с.

❷ Максимальна довжина шнура - 5 метрів.

❸ До 127 одночасно пристроїв, що підключені.

❹ Підключення пристроїв без вимикання комп'ютеру (так зване "гаряче" підключення), а також підключення у режимі Plug&Play[16].

Рис. 4.20. Піктограма позначення і роз'єми (вилки и розетки) послідовного порта USB

Іншим надзвичайно розповсюдженим інтерфейсом зовнішніх шин є стандарт SCSI (Small Computer System Interface –читаються як «сказі»). Найважливішою перевагою цього інтерфейсу уявляється те, що за його допомогою можна підключити до ПК до 8-ми периферійних пристроїв, маючи усього один слот розширення (рис. 4.21).

4.4. Інтерфейси у клієнт-серверних моделях взаємодії програм і пристроїв

При збільшенні кількості програмно-апаратних засобів, що одночасно використовуються, як правило вони об'єднуються з'єднуючими компонентами у мережу (рос.-сеть). Тому, комп'ютери и програми які взаємодіють і одночасно входять у склад інформаційної системи або задіяні у мережі, не являються

[16] Технологія PLUG & PLAY забезпечує незалежність пристроїв, що підключаються від конкретної операційної системи, а також визначає розширення для аби якої існуючої архітектури IBM-сумісних комп'ютерів, включая нові BIOS і апаратні можливості, котрі покликані відгороджувати користувача від проблем з настройкою і конфігуруванням пристроїв та інтерфейсів.

рівноправними. Деякі з них володіють ресурсами[17] (файлова система, процесор, принтер, база даних і т.д.), інші мають можливість тільки звертаються до цих ресурсів.

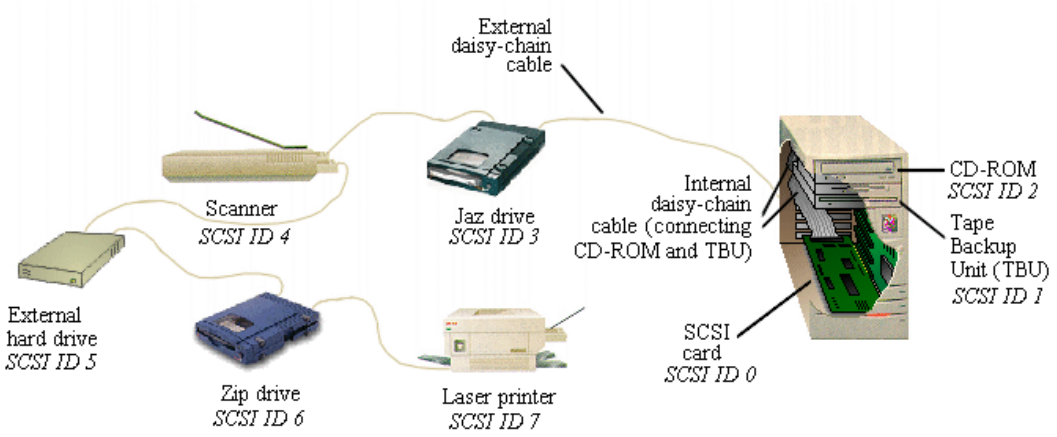


Рис. 4.21. Схема підключення семи пристроїв через SCSI порт

За такою схемою працюють багато комп'ютерних класів, де більшість комп'ютерів використовують спільні ресурси (програми, файли, послуги бази даних та ін.) (рис. 4.22).

Рис. 4.22 . Комп'ютерний клас з комп'ютерною мережею

Комп'ютер або програму, що управляє ресурсом, звуть ***сервером*** цього ресурсу (файл-сервер, сервер бази даних, обчислювальний сервер і т.д.). ***Клієнт,*** тобто компонент котрий користується послугами ***сервера***, і сам сервер якого нибудь ресурсу можуть знаходитися як у рамках однієї обчислювальної системи (комп'ютері), так і на різних комп'ютерах, які пов'язані мережею.

[17] Ресурс – засоб (рос.-средство) обчислювальної системи або комп'ютера, котре може бути виділено процесу обробки даних (програмі користувача) на визначений (рос.-определённый) момент часу (рос.-времени). Головними ресурсами комп'ютера є: процесори, області головної (рос.-основной) пам'яти, набори даних, периферийні (зовнішні) пристрої, програми і т.і.

У самому загальному випадку, термін "***клієнт-сервер***" означає таку архітектуру програмно-апаратного комплексу, у якій його функціональні частини взаємодіють за схемою "запит-відповідь". Якщо розглянути дві взаємодіючі частини цього комплексу, то одна з них (клієнт) виконує активну функцію, тобто ініціює запити, а інша (сервер) пасивно на них відповідає. По мірі розвитку системи ролі можуть мінятися, наприклад деякий програмний блок буде одночасно виконувати функції сервера по відношенню до одного блоку і клієнта по відношенню до іншого.

Найбільш проста форма архітектури клієнт-сервер - це розділення обчислювального навантаження (рос.-нагрузки) поміж двома окремими процесами: клієнтом і сервером.

Компанією Garthner Group, яка спеціалізується в галузі дослідження інформаційних технологій, запропонована наступна класифікація ***дволанкових (рос.-двухзвенных) моделей*** взаємодії клієнт-сервер (дволанковими ці моделі звуться тому, що три компоненти додатка по різному розподіляються поміж двома вузлами) (рис.4.23).

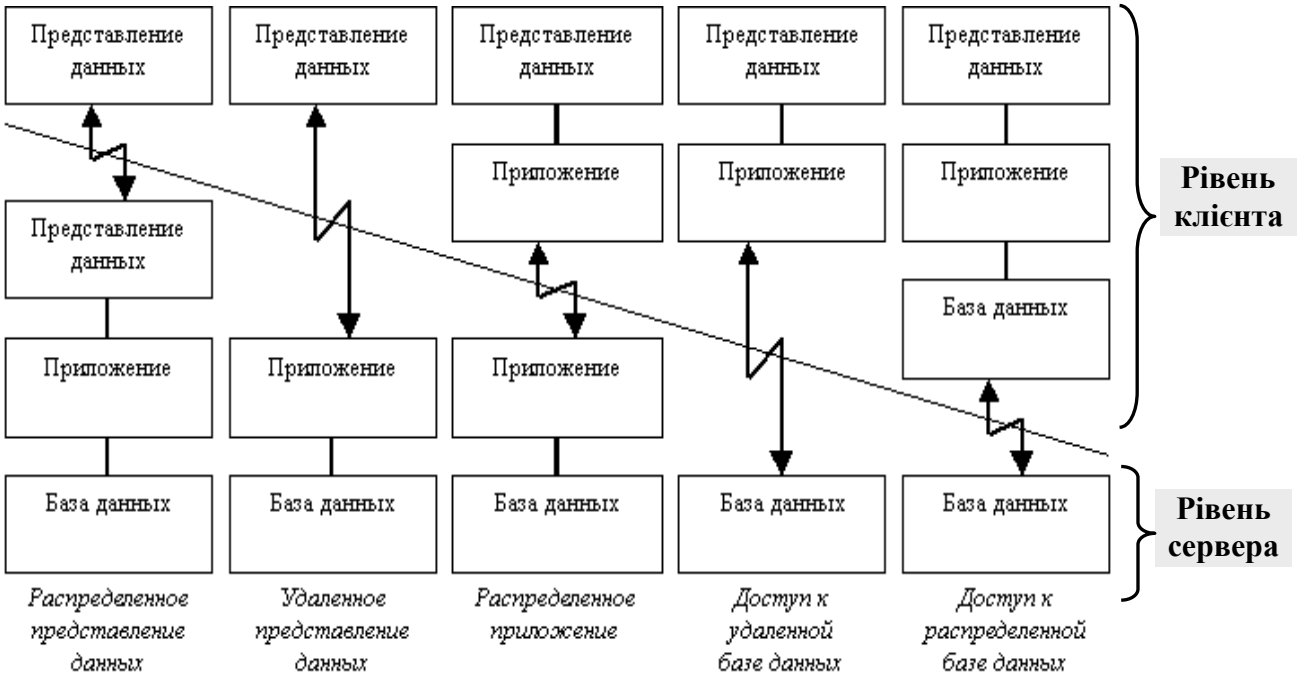


Рис. 4.23. Уявлення обчислень в мережі за допомогою дволанкових моделей клієнт/сервер

Як правило, ланкою-***клієнтом*** служить настільний ПК, на якому запускається ***програмне забезпечення*** (ПЗ) ***кінцевого користувача (рос.-конечного пользователя)*** (front-end software), або будь яка прикладна програма або пакет, що здатні (рос.-способные) направляти питання (рос-запросы) по мережі до сервера і обробляти інформацію, що буде отримана у відповідь. Сервер (back-end software, серверна частина), у свою чергу, отримує питання і робить (рос.-предпринимает) дії від імені клієнта.

ПК, що працює під управлінням Windows 95 і виконує програму клієнт-сервер Delphi, може, наприклад, подати (рос.-представить) на розгляд питання

серверу баз даних (скажімо, програмі Oracle7, версії 8.1, яка запущена на сервері Windows NT). Як правило, клієнт посилає питання базі даних у вигляді інструкцій на мові структурованих запитів (SQL- Structured Query Language), користуючись зрозумілим серверу бази даних діалектом.

Проміжне забезпечення (рос.-промежуточное обеспечение) (middleware) дає (рос.-предоставляет) загальний інтерфейс для ПЗ кінцевого користувача і сервера, що проникає скрізь шари (рос.-слои) графічного інтерфейсу користувача GUI, ОС, обчислювальної мережі і власних драйверів бази даних за допомогою загальних викликів і з застосуванням технології ODBC[18]. Для завершення операцій сервер бази даних виконує питання (рос.-запрос) і передає клієнту назад дані, що були викликані з бази даних для обробки їх програмою клієнта.

У зв’язку з постійним ростом кількості мов і систем програмування, програмних систем і додатків у комп'ютерному світі назріло питання: ***"Як забезпечити взаємодію програмних компонентів, які були написані у різний час в різних частинах світу різними компаніями та особистостями?"*** Практично для цього рішення потрібно забезпечити:

❶ Можливість виробників писати свої власні програмні компоненти і при цьому буту упевненими, що вони зможуть взаємодіяти поміж собою на різних платформах і з компонентами інших виробників.

❷ Погодженість (рос.-согласованность) нових версій компонент зі старими їх релізами.

❸ Незалежність програмних компонентів від типів і діалектів мов програмування, що використовуються при їх написанні.

❹ Вільний зв'язок поміж компонентами, які працюють як у одному, так і у різних процесах (і, можливо, на різних машинах), з використанням єдиної простої програмної моделі.

Вперше ***клієнт/серверну*** модель програмної взаємодії фірма Microsoft розробила і застосувала у технології створення ***динамічно зв’язаних бібліотек*** (DLL–Dynamic Linking Library) операційної системи Windows. У цієї моделі клієнтами служать:

❶ додатки, які звертаються до елементів-серверів бібліотек DLL;

❷ елементи-клієнти бібліотек DLL, які звертаються до елементів-серверів, які знаходяться у своїй або чужій бібліотеці (рис. 4.24).

Розвиток цієї технології втілився (рос.-воплотился) у програмній моделі, специфікації, архітектурі і технології, які розроблені Microsoft під загальною назвою COM – Component Object Model (Об'єктна Модель Компонентів) і заснованій також на моделі клієнт/сервер.

У COM будь яка частина програмного забезпечення реалізує свої сервіси як один або декілька об'єктів COM (не слід плутати (рос.-путать) об’єкти COM

[18] ODBC (Open Database Connectivity) – уявляє собою технологію і специфікацію інтерфейса для доступу до баз даних різних форматів, яка розроблена Микрософт. По своєї сутності, це інтерфейс API, такий же, як і Windows API, котрий має стосунки з програмуванням баз даних. Архітектура ODBC уключає в себе чотири компоненти: додаток (програма користувача), ODBC менеджер, ODBC драйвера і джерело (рос.-источник) даних (бази даних, наприклад, Interbase, Oracle та ін.).

з об'єктами у мовах програмування типу C++. Не дивлячись на те, що в них є деякі загальні риси, це абсолютно різні речі).

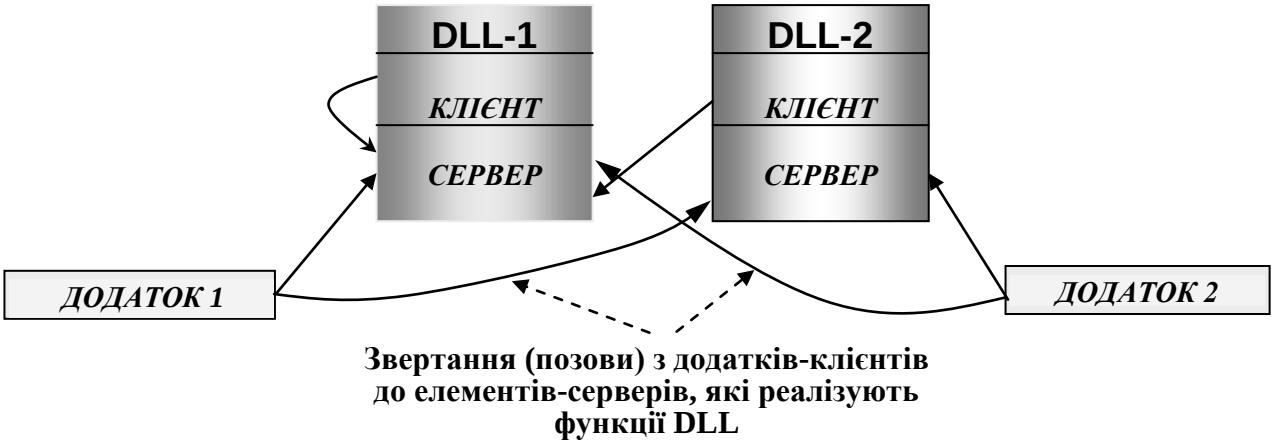


Рис. 4.24. Клієнт-серверні зв'язки поміж додатками: зовнішні (від додатка – до DLL) і внутрішні (від клієнт-функції DLL– до сервер-функції DLL) у ОС Windows

Одним з наріжних (рос-краеугольных) каменій технології COM є принцип взаємодії програмних компонентів виключно через інтерфейси (рис. 4.25).

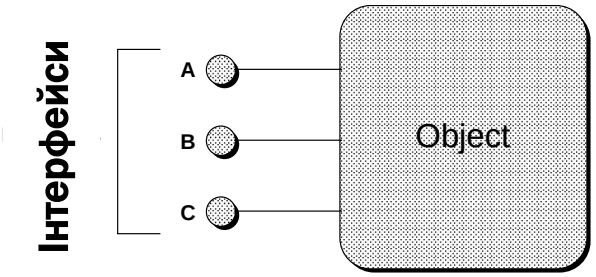


Рис. 4.25. Компонент з групою своїх інтерфейсів.

Кожний з об'єктів COM підтримує один або декілька інтерфейсів, які реалізують ті або інші методи. Клієнти можуть отримати доступ до сервісів сервера об'єкта COM тільки через виклики методів інтерфейсів об'єкта — у них немає безпосереднього доступу до даних об'єкта.

Використання інтерфейсів COM дозволяє:

❶ уніфікувати (рос.-унифицировать) процес перевірки можливостей програмного компонента;

❷ забезпечити зв'язок поміж компонентами, що знаходяться у різних нитках (рос-нитях), процесах і, можливо, на різних комп'ютерах;

❸ знизити витрати на розробку програмного забезпечення.

В основі моделі COM лежать декілька важливих концепцій:

❶ двоічний стандарт виклику функцій (для аби якої платформи);

❷ забезпечення суворої (рос.-строгой) типізації при об'єднанні функцій у інтерфейси з унікальними ідентифікаторами (GUID);

❸ управління часом (рос.-временем) життя об'єкту;

❹ механізм однозначної ідентифікації компонентів і інтерфейсів COM по унікальним ідентифікаторам (GUID).

Компоненти COM реалізуються у вигляді бібліотек DLL, елементів ActiveX, кнопок и команд графічних інтерфейсів додатків (Word, Fotoshop та ін.), елементів форм і таке інше. Специфікація на рівні структур даних дозволяє зв'язувати компоненти, написані для різних платформ (Microsoft® Windows®, Microsoft Windows NT™, Apple® Macintosh®, UNIX® і деяких інших) Стандарт є відкритим, що дозволяє створювати власні компоненти і використовувати компоненти, які створені іншими фірмами.

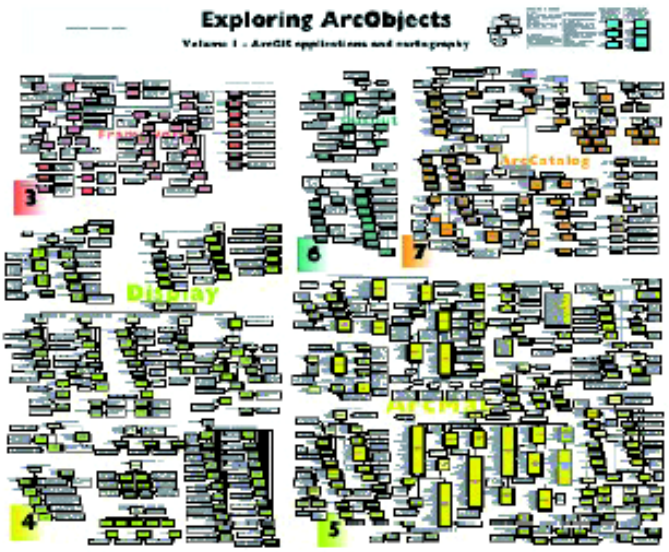


Рис. 4.26. Уявлення невеликої частини компонентів ArcObjects з їхніми інтерфейсами

Поточним часом COM є однією з найбільш поширено розповсюджених у світі програмною компонентною моделлю. По даним Інтернет у 2000 р. COM компоненти використовувалися на більш ніж 150-и мільйонах систем по усьому світу. Однією тільки фірмою ESRI (Environment System Research Institute) (США) на кінець 2002 року було розроблено більш за 1200-т компонентів під загальною назвою ArcObjects для потреб геоінформаційних систем (ГІС) (рис.4.26).

Подальшим продовженням моделі COM є модель DCOM, тобто розподілена COM, яка розширює границі взаємодії компонентів у мережному просторі (рис. 4.27).

При виклику одним компонентом іншого, згідно моделі DCOM (див. рис.4.27) діється наступне.

1. Компонент програми на клієнтському ПК здійснює виклик, який звертається до віддаленого серверного компонента, такому, наприклад, як система авторизації кредитних карток. При цьому він звертається до операційної системи Windows.

2. DCOM встановлює з'єднання через мережу поміж двома компонентами, використовуючи механізм віддаленого виклику процедур (RPC).

3. Клієнтський компонент може взаємодіяти з серверним компонентом так, як коли б вони розміщувалися на однієї і тієї же машині.

Така модель дуже добре підтримує модель ***розподілених обчислень***, яка є парадигмою організації додатків, у котрій різні частини програми можуть виконуватися на різних комп'ютерах в мережі.

Паралельно з DCOM розвивається подібна архітектура мережних обчислень під назвою Common Object Request Broker Architecture (CORBA), яку підтримують такі крупні компанії, як IBM, Sun Microsystems і ряд інших

виробників. Сьогодні вона застосовується багатьма великими організаціями для розгортання розподілених обчислень у масштабах підприємства на базі Unix-серверів і мейнфреймів.

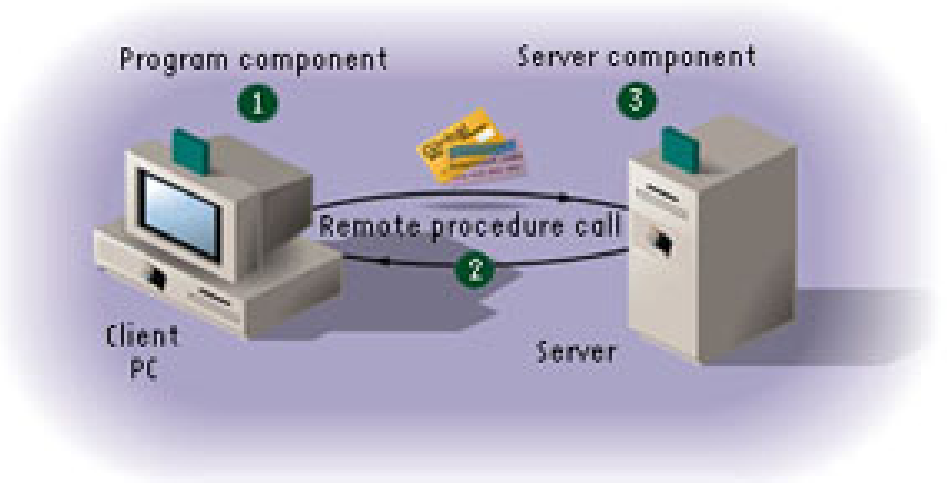


Рис. 4.27. Принцип взаємодії компонентів за допомогою моделі DCOM

Архітектура розподілених інтероперабельних інформаційних систем CORBA (див. підрозділ 6.4), базується на концепції ***проміжного шару (рос.-промежуточного слоя)*** *(middleware)*, що містить служби і засоби підтримки ***глобального простіру об'єктів, їх життєвого циклу*** та ***інтероперабільності***. Цей шар міститься поміж операційною системою (уключаючи засоби управління комп'ютерними мережами) і прикладними системами, специфічними для конкретних областей застосування. Компонентами такого глобального простору є довільні (рос-произвольные) інформаційні ресурси – програмні компоненти, бази даних, бази знань, файли даних, які у тому числі містять мультімедійну інформацію, компоненти існуючих інформаційних систем і т.д., які уявлені (рос.-представлены) колекціями об'єктів, незалежних від апаратурно/програмних платформ іх реалізації і розміщення у просторі. Уніфіковані інтерфейси взаємодії таких компонентів забезпечує створення інтероперабельних середовищ інформаційних ресурсів, які забезпечують як динамічне утворення композицій ресурсів, що зорганізуються для вирішення конкретних задач (реалізація інформаційної системи, реалізація проекту, підтримка групової діяльності і т.д.), так і мегапрограмування – програмування у середовищі готових компонентів для забезпечення технологій ***повторного використання*** інформаційних ресурсів (наприклад у Інтернеті).

Головними компонентами архітектури проміжного шару, який розвивається консорціумом OMG (див. підрозділ 7.2), є:

❶ Загальна Архітектура Брокера Об'єктних Запитів – Common Object Request Broker Architecture, тобто - CORBA;

❷ Брокер Об'єктних Запитів (Object Request Broker - ORB), який грає вирішальну роль "***загальної шини (рос.-общей шины)*** " у глобальному просторі об'єктів, за допомогою якої об'єкти цього простору можуть взаємодіяти один з одним, маючи широку підтримку цілим рядом продуктів, які розповсюджуються знаними виробниками: Digital, IBM, SunSoft, IONA Technologies та деякими іншими для різних типів платформ. Загальна система Об'єктних Служб и Загальних Об'єктних Засобів (CORBA) організує нижній шар архітектури проміжного шару, який забезпечує технологічну платформу інтероперабельності (властивість до взаємодії).

Важливим компонентом CORBA є ***мова визначення інтерфейсів (IDL - Interface Definition Language)***. У версії стандарту CORBA 2.0 ця мова почала називатися OMG IDL. Таким чином, OMG IDL є засобом, за допомогою якого потенційним клієнтам об'єкта-сервера повідомляється, які операції доступні на його ***інтерфейсі*** і як їх потрібно викликати.

IDL – є мовою однорідної специфікації інтерфейсів різноманітних інформаційних ресурсів, інкапсульованих засобами технологій CORBA. Ця мова є чисто описова. Визначення (рос.-определения) на мові IDL можуть бути відображені стандартним засобом у конкретні мови програмування, такі як C, C++, Smalltalk. Репозіторій Інтерфейсів, що містить визначення інтерфейсів на мові IDL, дозволяє бачити інтерфейси доступних інтерфейсів у мережі і програмувати їх використання у програмах-клієнтах з урахуванням можливостей Брокера. Специфікації, що містяться у репозіторії, можуть бути також використані у період виконання клієнта. Ролі "клієнта" і "серверу" слід розглядати як відносні: клієнт (сервер) у архітектурі CORBA може бути сервером (клієнтом) по відношенню до інших клієнтів (серверів).

Поширене розповсюдження і повсюдне (рос.-повсеместное) застосування (рос.-применение) клієнт-серверних комп'ютерних моделей розширили і поглибили поняття інтерфейсу в цілому.

Дійсно, якщо у інформаційних структурах зв'язуються дві системи (або більше), то це робиться у цілях сумісної роботи. У такому разі, по сутності, ***інтерфейс*** - це специфікація (опис) і реалізація процедури або процесу їх взаємодії. Інтерфейс визначає параметри і характеристики границь (або виявляється їх описом). Параметри можуть бути логічними, фізичними, функціональними, електричними і можливо іншими, а характеристики – конструктивними (тобто характеристики можуть визначатися конструкцією пристрою).

Таким чином, за зовнішньою простотою і комфортністю аби якого існуючого інтерфейсу ховається (рос.-скрывается) багатошарова (рос.-многослойная) структура багаточисельних інтерфейсів і реалізуючих їх програмно-апаратних засобів. Опис взаємодії різних логічних, фізичних і програмних компонентів ***моделі клієнт-сервер*** в узагальненому (рос.-обобщённом) вигляді може бути представлено так, як показано на рисунку 4.28.

Рис. 4.28. Багаторівнева модель структури уявлень інтерфейсу

На рисунку зображені різні "уявлення" складного поняття інтерфейс у моделі клієнт-сервер. Дійсно, по-перше, у його структурі представлені реалізації сумісної реальної взаємодії об'єктів самої різної природи (людина-комп'ютер, людина-програма, програма-принтер и т.д.).

По-друге, в процесі проектування і моделювання складних процесів взаємодії інформаційних систем (наприклад, засобами мови UML) інтерфейси розглядаються на концептуальному, абстрактному, логічному і архітектурному рівнях (логічні імена пристроїв, опис взаємодії комп'ютерів у мережі, взаємодія програм і розподілених компонентів, які знаходяться на декількох комп'ютерах одночасно і т.д.).

По-третє, у зв'язку з розширенням способів передачі даних поміж комп'ютерами, і у тому числі и безпробудно, ускладнюються протоколи і специфікації взаємодії, а таким чином і інтерфейс.

Якщо ж розглядувати послідовно принципи взаємодії сутностей або об'єктів у процесі їхньої взаємодії і наступної реалізації інтерфейсів поміж ними, то згідно рисунку 4.28 послідовність розглядання з ціллю подальшої реалізації такої взаємодії може виглядати наступними чином.

1. Принциповий розгляд можливостей реалізації взаємодії двох компонентів або елементів на рівні питань (рос.-запросов) необхідних сервісів (***клієнт***) і забезпечення цими сервісами (***сервер***) провадиться на концептуальному і архітектурному рівнях. Тут вивчаються можливості принципових підходів для реалізації взаємодії на програмному і фізичному рівнях, тобто на рівнях підтримки процесів інформаційної взаємодії сутностей шляхом розробки конкретних кодів програм, електричних, електронних і механічних компонентів, а також здійснення інформаційної взаємодії за допомогою створення відповідних ***інтерфейсів***.

2. Далі йде етап аналізу і проектування, тобто уявлення і моделювання компонентів, які приймають участь у взаємодії, реалізація якої розробляється. На даному рівні застосовуються абстракції даного етапу, як правило, при підтримці засобів мови UML.

3. При виборі засобів реалізації в рамках моделей інтерфейсів і мов реалізації застосовуються логічні абстракції.

4. На рівні програмної реалізації моделей, які розроблені (класів, об'єктів і їх багаторівневої взаємодії) застосовуються засоби конкретної мови програмування: C++, Object Pascal, Visual C++, Java, C# і інші (програмні абстракції).

5. По мірі розвитку технологій все більша кількість різноманітних електронних компонентів мають можливість використовувати сервіси один одного (технологічні абстракції). Наприклад, мобільні телефони вже мають можливість обмінюватися даними з серверами Інтернет і іншими службами WWW.

6. Фізичні абстракції дозволяють виділяти набори компонентів у логічно зв'язані комплекси, які уявляють собою елементи з більш високим рівнем організації.

Розглядаючи комплексно сукупність інтерфейсів, що існують поміж складними системами (наприклад, людиною і комп'ютером) можна назвати їх системними.

Підводячи підсумки усьому вищесказаному, види інтерфейсів, що підтримуються у складній, багаторівневої архітектурі клієнт-сервер, можна описати наступним чином (таблиця 4.1).

Таблиця 4.1

Взаємодіючі сутності, об'єкти, системи, компоненти, елементи і їхні інтерфейси

№ пп.	Взаємодіючі сутності, об'єкти і т.д.	Вид (тип) інтер-фейсу	Спосіб і засоби підтримки	Реалізація
1	Людина – реальний об'єкт	Системний	Комп'ютерний, мови програмування, мова UML, CASE-засоби, фізичні пристрої(мікропроцесори)	Логічна, програмна, фізична
2	Людина – комп'ютер (ПК)	Систем-ний	Командний (текстова стрічка, сполучення клавіш клавіатури, клацання клавішами миші); графічний (графічні об'єкти, кнопки, меню); мови програмування; RAD-засоби, фізичні пристрої	Логічна, програмна, фізична
3	Людина – пристрій ПК	Систем-ний	Логічні імена у ПК, мови програмування, RAD-засоби, кнопки пристроїв	Програмна Фізична
4	Людина – програма (додаток)	Систем-ний	Командний (текстова стрічка, сполучення клавіш клавіатури, клацання клавішами миші); графічний (графічні, кнопки, меню); мови програмування; RAD-засоби	Програмна
5	Комп'ютер – комп'ютер	Системний	Мережний, пристрої комунікації, програми, мови програмування	Програмна, фізична
6	Програма – програма	Програм-ний	Програмний, імена, мови програмування	Програмна
7	Об'єкт (програми) – об'єкт	Програм-ний	Програмний, імена, GUID, UUID, CLSID, мови програмування	Програмна
8	Програма – пристрій	Програм-ний	Програмний, імена, мови програмування, пристрої комунікації, протоколи	Програмна, фізична
9	Пристрій – програма	Програм-ний	Програмний, імена, мови програмування, пристрої комунікації, протоколи	Програмна, фізична
10	Пристрій – пристрій	Фізичний	Пристрої комунікації, протоколи, угоди	Фізична, програмна

З проникненням комп'ютерів у все більшу кількість галузей поміж системний інтерфейс продовжує зоставатися "гарячою" точкою концентрації

зусиль розробників апаратних і програмних засобів, але саме головне – продовжують удосконалюватися інформаційні моделі і технології їх реалізації. В останній час з'явилися візуальні засоби програмування (Delphi, Visual C++, Visual Basic та ін.), у яких закладені об'єктно-орієнтовані концепції програмування і інструменти реалізації відповідних моделей (COM, DCOM, OLE Automation, ActiveX і т.д.). Швидко розвиваються компонентні моделі і технології, які базуються на специфікаціях і архітектурах .NET і Java, що реалізують широкі функціональні можливості обробки інформації в Інтернеті і WWW.

Одночасно з ними удосконалюються нові технології аналізу і проектування (UML, CASE-технології, Web-технології і багато інших). Не відстають і розробники апаратних засобів, які пропонують усе нові й нові технічні рішення доступу до усе зростаючих об'ємів різноманітної інформації (DVD, безпровідні Internet-сервіси, радіо і телебачення в Internet і т.д.), а гонка усе продовжується...

Запитання.

1. Чому уявлення ″інтерфейс″ має декілька значень?
2. Наведіть приклади інтерфейсів різних систем?
3. Чим відрізняється аналогове уявлення процесу від цифрового?
4. Що таке дані?
5. Що таке інформація?
6. Які концептуальні складові повінні бути присутніми для забезпечення процесу взаємодії користувача з комп'ютером?
7. Які екранні об'єкти, що забезпечують графічний інтерфейс користувача є пасивними, а які активними?
8. Які концепції лягли у основу розробки WIMP-інтерфейсів?
9. Наведіть приклади метафор, які використовуються для створення графічного інтерфейсу користувача?
10. Які задачі виконують у комп'ютері розширення імен файлів?
11. Чим відрізняються апаратні інтерфейси від програмних?
12. Які головні елементи входять до моделі інтерфейсу клієнт-сервер?

У корні невірною є ідея, що існує тільки один правильний спосіб або мова, щоб вирішити аби яку проблему для аби якого користувача.

Б'єрн Страуструп, автор мови C++.
(з інтерв'ю 21.02.2001)

5. ЕВОЛЮЦІЯ МОВ ПРОГРАМУВАННЯ

5.1. Початок розвитку мов програмування

Творці перших мов програмування високого рівня для комп'ютерів прагнули зробити їх у меншій мірі подібними на середовище спілкування поміж людиною і комп'ютером і у більшій – на упорядкований набір знаків і символів. Споконвічно (рос.-изначально) держати курс на традиційну і вже улаштовану (рос.-устоявшуюся) математичну символіку запропонував Х. Рутисхаузер (у 1952 р.), який став родоначальником ідеї мов програмування і одним з авторів мови Алгол-60. Поширене розповсюдження і застосування його ідеї отримали і були втілені у життя лише у 1957 р., після того, як корпорація IBM опублікувала опис мови Фортран і реалізувала для нього компілятор, який транслював програми у машинний код. По суті, з цього моменту і почалася епоха мов програмування (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)

Рис. 5.1. Фрагменти тексту програм на мовах
Фортран (а) і Алгол (б)

До нинішнього часу програмування, як процес, полягає у створенні комп'ютерних програм, які містять конкретні інструкції (команди) для виконання їх комп'ютером. Різні частини програми можуть бути написані на різних мовах програмування (МП). Мова програмування уявляє собою стандартизований засіб комунікації для повідомлення комп'ютеру команд на виконання конкретних задач. Він також надає програмісту можливість точно указати, якими видами даних комп'ютер повинен маніпулювати, а також послідовність дій у різних обставинах. Таким чином переслідуються дві головні цілі:

❶ висловлювати програму на логічному рівні, який значно перевершує логіку низькорівневих кодів блока центрального процесора (CPU-Central Processing Unit);

❷ забезпечувати сумісність і переносність програм, що розроблюються поміж різними комп'ютерними платформами, котрі, як правило, мають різне програмне забезпечення для їх трансляції у машинну мову конкретної системи.

Якщо механізм трансляції застосовується до усього тексту програми ціликом для перекладу у внутрішній формат комп'ютера, то такий вигляд обробки зветься ***компіляцією***. В результаті компіляції фрагмента коду програми ми одержуємо файл з розширенням .EXE. В протилежному випадку текст програми транслюється крок за кроком, и кожний крок виконується негайно і такий механізм зветься ***інтерпретацією***. Програми, що інтерпретуються, виконуються значно повільніше (рос.-медленее) за ті, що компілюються, але мають значно більшу гнучкість при взаємодії з програмно-апаратними засобами комп'ютера (рис. 5.2).

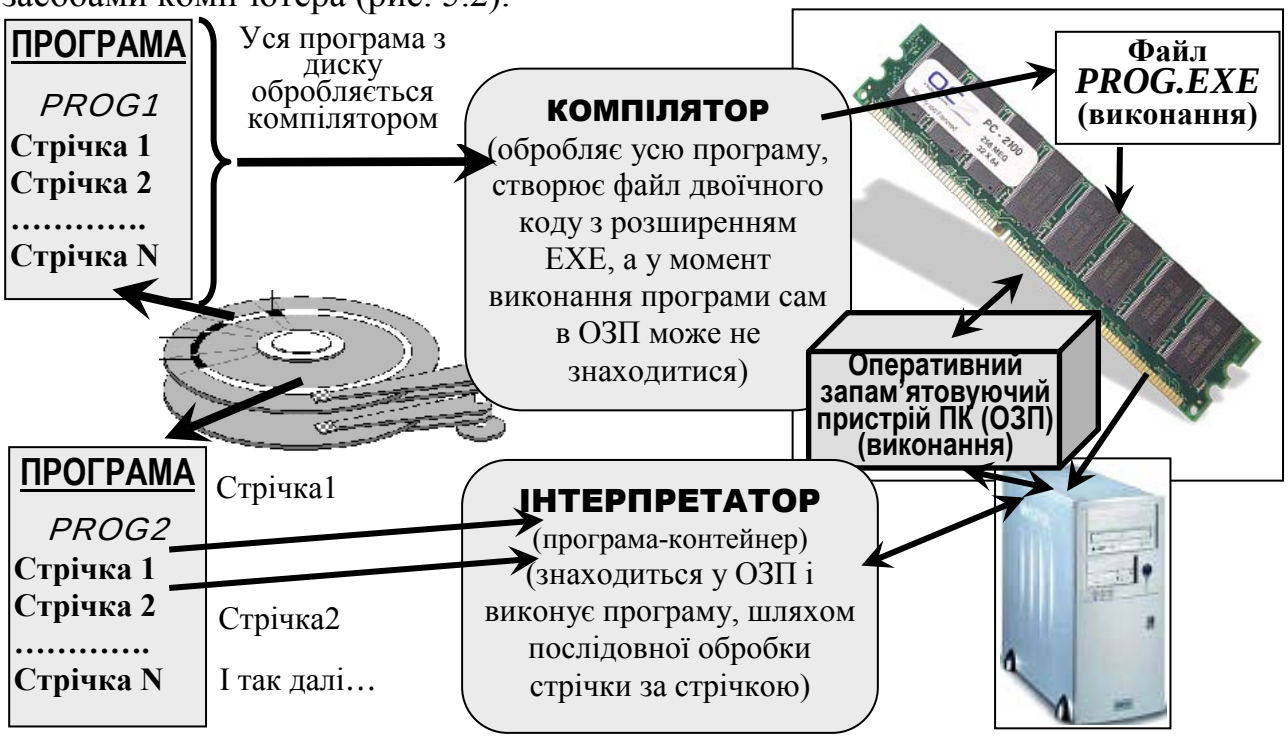


Рис. 5.2. Технологія обробки початкових (рос.-исходных) текстів програм компіляторами і інтерпретаторами з наступним виконанням

У сучасних комп’ютерах інтерпретатори трансформуються у програми (додатки)-контейнери, які володіють (рос.-обладают) значно більшою функціональністю. До них можна віднести додатки MS Excel, MS Internet Explorer і багато інших (див. главу 6).

5.2. Розширення функціональності мов програмування

У комп’ютерах 70-80-х років на початкових етапах їх розвитку велике значення мали мови програмування систем (system programming language) (МПС). До них можна віднести мови асемблерів і ряд мов високого (high level) рівня (Паскаль, Сі, C++, Java та ін.). Поступово останні майже повністю віти снили (рос.-вытеснили) мови асемблера при розробці великих додатків. При компіляції текстів програм цих мов як правило виходять достатньо компактні двоїчні коди, які дають високу швидкість виконання програм. Однією з головних якостей МПС завжди вважалась так звана типізація, при котрій:

❶ кожна змінна для схову (рос.-хранения) даних повинна бути відвічно (рос.-изначально) декларована з тим, щоб бути приписаною до визначеного типу: ціле (Integer), реальне (Real, Double), укажчики (рос.-указатели) на строку (String) і т.д., а також змінна повинна використовуватися тільки тими способами, котрі цьому типу відповідають;

❷ дані і код розділені – важко, якщо взагалі можливо, створити новий код у час виконання;

❸ змінні можуть бути згруповані у об’єкти з добре визначеною структурою и процедурами для маніпулювання ними. Об'єкт одного типу не може використовуватися там, де очікується використання об'єкту іншого типу (рис. 5.3).

(Число типу ***Integer***) не можна (!!!) розділити на (Число типу ***Real***)
якщо результат присвоюється змінної ***I*** типа ***Integer***.
Тобто, оператор виду: `I := (24 / 5.37e-2);` – заборонено!

Рис. 5.3. Приклад типізації в мові Турбо Паскаль

Типізація забезпечує цілий ряд переваг (рос.-преимуществ):

❶ великі програми вона робить більш технологічними завдяки точному визначенню сутностей, які використовуються і їх відміну від інших;

❷ компілятори використовують інформацію о типах для виявлення (рос.-обнаружения) визначених (рос.-определенных) видів помилок, таких як спроб, задіяти величину з плаваючою точкою як покажчик (рос-указатель);

❸ типізація підвищує ефективність виконання, дозволяючи компілятору генерувати спеціалізований компактний двоїчний код.

У сильно типізованій мові програміст обов'язково декларує, як кожна порція інформації буде використовуватися, а мова запобігає (рос.-

предотвращает) її використання іншим способом. У слабо типізованій мові не існує апріорі завданих обмежень на використання інформації; її зміст (рос.-смысл) визначається тільки способом, котрим вона використовується.

Разом з тим, самі по собі сучасні комп'ютери принципово безтипові. Будь яке слово пам'яті може зберігати величину будь якого типу, будь то ціле або реальне (рос.-вещественное) число, покажчик (рос.-указатель) або команда. Зміст (значення) величини визначається тим, як вона застосовується. Одне і те ж слово може використовуватися у різних випадках по-різному. Наприклад, в мові Visual Basic (VB) 6.0 спеціальний тип даних ***Variant*** дозволяє зберігати дані усіх інших типів (з можливістю наступної обробки операціями, які відповідають типу, що обробляється) (рис. 5.4).

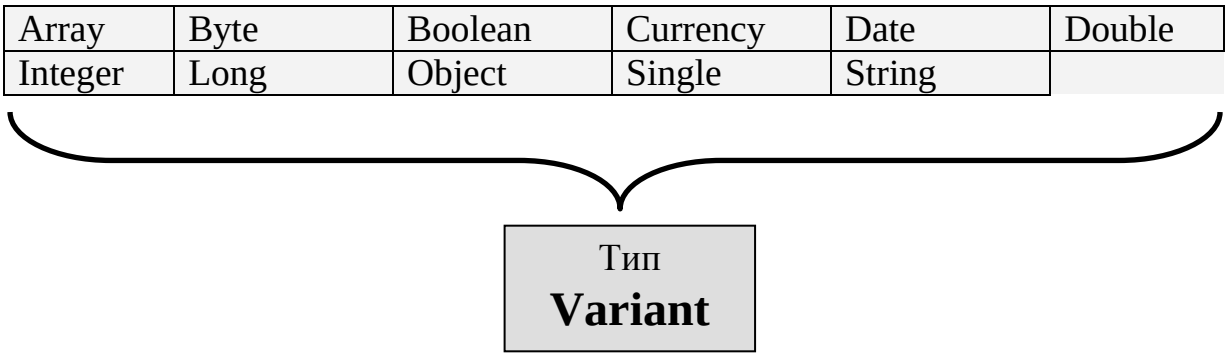


Рис. 5.4. Багатофункціональність типу Variant
в мові Visual Basic

З середини 90-х років почали з'являтися так звані мови сценаріїв (scripting languages, скриптові мови) (СМ). В першу чергу це було пов'язано з втягненням (рос.-вовлечением) в мережові структури усе більшої кількості комп'ютерів, активізацією процесу розподілених обчислень у мережових середовищах і різким ростом об'єму інформації, що обробляється через Інтернет засобами World Wide Web. Мови програмування розроблялись для інтерактивного використання і мали у своєму складі багато команд, які уявляли собою міні-програми, які були призначені, як правило, для комбінування вже існуючих компонентів. З більш, чим 30-и найбільш популярних СМ (див. табл. 5.1) можна особливо відмітити мови, Rexx, Tcl, Perl, Visual Basic і оболонку Unix (shell).

Таблиця 5.1.

Список головних мов скриптов у алфавітному порядку

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	Perl	Rexx	Visual Basic (VB)
CobolScript	(ECMAScript)	PHP	Ruby	VBScript

И хоча мови сценаріїв часто використовуються для розширення

властивостей компонентів, вони мало придатні (рос.-пригодны) для програмування складних алгоритмів і структур даних, які як раз і забезпечуються компонентами.

Ось чому мови сценаріїв часто називають ***склеюваючими*** (рос.-склеивающими) мовами (glue languages) або мовами інтеграції систем (system integration languages).

Процес розробки мов програмування постійно продовжується и у таблиці 5.2 приведені деякі характеристики найбільш популярних серед користувачів мов програмування різного призначення. Більш повна інформація приведена у Додатку 3.

Таблиця 5.2.

Найбільш розповсюджені мови програмування (МП) для ПК

Найменування МП	**Рік створення**	**Тип мови**	**Автор (и) розробки**
Pascal	1970	МПС/–	Ніклаус Вірт
C	1972	МПС/–	Денніс Рітчі
C++	1980	МПС/ОО	Б'ёрн Страуструп
Turbo Pascal	1983	МПС/–	Андерс Хейльсберг, Філіпп Кан (Borland Int.)
Perl	1986	СМ/ОО	Ларри Вол
Visual Basic	1991	СМ/ОО	Microsoft
Delphi/Object Pascal	1995	МПС/ОО	Андерс Хейльсберг (Borland Int.)
Java	1995	МПС/ОО	Патрік Нотон і Джеймс Гослінг
Java Script	1995	СМ/ОО	–
VBScript	1997	СМ/ОО	Microsoft
XML	1998	–/–	W3C (консорціум)
C#	2000	МПС/ОО	Андерс Хейльсберг (Microsoft)
ПРИМІТКИ: МПС–мова програмування систем, СМ–скриптова мова; ОО–об'єктно-орієнтований.			

Однією з важливіших ознак (рос.-признаков) класифікації мов програмування є принадність їх до одного з стилів, головні з котрих приведені у таблиці 5.3.

Таблиця 5.3.

Стилі мов програмування.

Стиль (тип) мов програмування	**Найменування мови**
Процедурний	FORTRAN, COBOL, BASIC, PL/I, FORTH, Pascal, Turbo Pascal
Декларативний (функціональний)	LISP, Common Lisp
Логічний (реляційний)	Prolog
Паралельне програмування	C++, Ada, Concurrent Pascal, Occam
Об'єктно-орієнтований	Smalltalk, Object Pascal, C++, Python, Java, Simula, Java Script, VBScript, C#
Модульний (компонентний)	Visual Basic, C++, C#, Object Pascal, Java, Ada, Modula-2, Eiffel, Component Pascal
Спеціалізовані мови	HTML, XML, Tcl, Perl, Cold Fusion

Для уявлення виразних можливостей мов програмування різних стилів у таблиці 5.4 приведені фрагменти програм для реалізації виводу на екран

дисплея текстового повідомлення “Hallo World!”, яке стало традиційним у курсах вивчення практично всіх мов уявлення сигнатур.

Таблиця 5.4.

Програми виводу на екран дисплея стандартної фрази “Hallo, World!”, виконане на мовах програмування різних стилів.

Процедурна мова Turbo Pascal

```
Program Hello;

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

Процедурна мова FORTRAN

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

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

Декларативна (функціональна) мова LISP

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

Логічна (реляційна) мова Prolog

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

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

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

Мова паралельного програмування Ada

```
    with Text_Io; use Text_Io;

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

Модульна (компонентна) мова Visual Basic

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

Об'єктно-орієнтований. C++

```
hallo.cpp
      #include <iostream> //
      ::std::cout
      #include <ostream>  // <<

      int main(){ ::std::cout <<
      "Hallo, world!" << '\n'; }
```

Об'єктно-орієнтована мова Java

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

Спеціалізована мова HTML

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

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

Спеціалізована мова Perl

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

З точки зору класифікації мов програмування по типам задач, що за їх допомогою вирішуються, деякі автори пропонують наступне розподілення (табл. 5.5).

Таблиця 5.5.

Класифікація мов програмування по типам задач, що за їх допомогою вирішується.

Тип задачі, що вирішується	**Найменування мов програмування**
Задачі штучного інтелекту	Lisp, Prolog, Multilisp, Commonlisp, Рефал, Planner, QA4, FRL, KRL, QLisp
Паралельні обчислення	Fun, Apl, Alfl, PARAlfl, ML, SML, PPL/1, Hope, Miranda, Occam, PFOR, Glypnir, Actus, паралельний Cobol, ОВС-ЛЯПИС, ОВС-Алгол, ОВС-Фортран, PA(1), PA(G)
Задачі обчислювальної математики і фізики	Occam, PFOR, Glypnir, Actus, паралельний Cobol, ОВС-ЛЯПИС, ОВС-Мнемокод, ОВС-Алгол, ОВС-Фортран, PA(1), PA(G)
Розробка інтерфейсу	Forth, c, C++, Асемблер, Макроасемблер, Simula-67, OAK, Smalltalk, Java, РПГ
Розробка програм-оболонок, розробка систем	Forth, c, C++, Асемблер, Макроасемблер, Simula-67, OAK, Smalltalk, Java, РПГ
Задачі обчислювального характеру	Algol, Fortran, Cobol, Ada, PL/1, Фокал, Basic, Pascal
Оформлення документів, обробка великих текстових файлів, організація віртуальних трьохвимірних інтерфейсів у Інтернеті, розробка баз даних	HTML, XML, Perl, Tcl/Tk, VRML, SQL, PL/SCL, Informix 4GL, Natural, DDL, DSDL, SEQUEL, QBE, ISBL
Мови мережної обробки (реалізації Web–сервісів)	Java, C++, C#, Visual Basic .NET
Клієнт-серверні мови програмування у WWW	Java, JavaScript, VBScript, XML, ASP (Active Server Pages), PHP, Perl, Python, ScriptBasic, HTML, CGI, CSS
Мови програмування геоінформаційних систем (мови вбудованих систем)	Avenue, Visual Basic for Application, ArcXML, MapBasic, AutoLisp, Visual Basic, VBScript, GML

Додатки і програми пишуться для рішення усе більш складних задач з використанням мережних інформаційно-комп'ютерних технологій та Web– і Інтернет технологій, які стрімко розвиваються (див. Додаток 4). По деяким даним[1], світ прийшов до осмислення фундаментальної важливості мережної обробки даних, и щоб це оцінити, потрібно тільки вдуматися в наступну приголомшуючу цифру: у 2000 року ринкова вартість 200-т крупніших мережних американських компаній, акції яких котируються на фондовій біржі, перевищувала 5 трлн. дол. За цей же рік їх сумарний оборот збільшився на 100 млрд. дол. (тобто на 14%) — з 709,7 у 1999 р. до 809,6 млрд. дол. у 2000 р.

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

І, як стало ясно з розвитку подій, ключ до Інтернет і Web-сервісів — це мови програмування, орієнтовані на створення компонентних додатків, програмування безпровідних інтерфейсів і мобільних пристроїв.

Ідеолог мов програмування з корпорації Microsoft і колишній створювач мов Turbo Pascal і Delphi, Андерс Хейльсберг, охарактеризував C# (“сі шарп”) як “першу справжню компонентно-орієнтовану мову програмування у сімействі C/C++”. Він також висловив упевненість, що модель програмування, яка заснована на компонентах з асоційованими с ними даними (відповідають якостям (properties), у деяких інших мовах) і варіантами поведінки (відповідають подіям, events), підтримуються у C# більш істотно (рос.-естественно), ніж у мові Java. “Java емалює (рос.-эмулирует) властивості з використанням спеціальної схеми іменування методів доступу, — каже Хейльсберг, — а обробники подій — за допомогою різних “перехідників” і зв’язуючого коду”. Він згоден з тим, що і Java, і C++ також підтримують компонентно-орієнтований стиль програмування, але вважає що “компоненти у них не володіють (рос.-обладают) громадянством першого класу”. Деякі відміни цих мов приведені в табл. 5.6.

Таблиця 5.6.

Порівняння об'єктно-компонентно-орієнтованих мов

Риса	**C++**	**Java**	**C#**
Управління системними ресурсами	Виділення і звільнення пам'яті вручну, з використанням спеціальних операторів	Автоматичний «збір сміття (рос.-мусора)»	
Продуктив-ність труда програмістів	Невелике число високорівневих механізмів	Добре структуровані визначення структур даних і оператори передачі управління	Розширений словник визначених типів даних; Підвищена гнучкість у передачі управління
Надійність ПЗ	Обмежені можливості контролю типів; незручні і часто ігноруємі програмами механізми обробки помилок	Жорсткий контроль типів дозволяє уникнути помилок з непередбаченим перевизначенням операцій; механізми обробки виняткових ситуацій дозволяє обробляти помилки тільки структурованим образом	
Продуктив-ність додатків	Свобода використання покажчиків (рос.-указателей) на дані дозволяє раціоналізувати багато операцій, але в той же час створює можливості для багаточисельних програмістських помилок; обмежений набір механізмів, які характерні для об'єктно-орієнтованих систем, зводить до мінімуму додаткові обчислювальні витрати, пов'язані з їх реалізацією	Покажчики не використовуються; процедури перетворення типа поміж об'єктами і примітивними типами бувають порой достатньо незграбні	Використання покажчиків можливо, але тільки у коді, поміченому спеціальним ярликом “unsafe”; ефективне перетворення типів об'єктів і значень з використанням механізму «боксів»
Переноси-мість додатків	Компіляція у платформено-незалежний код	Компіляція в універсальний байт-код, придатний (рос.-пригодный) для використання на аби який платформі, оснащеною віртуальною Java-машиною	Компіляція у код .Net Intermediate Language

Слід додати, що у рамках концепції створення всесвітньої архітектури використання Web–сервісів – .Net (дот нет), яку розвиває Microsoft, цією

корпорацією реалізовано ціле сімейство продуктів розробки з відповідними мовами: Visual Basic .NET 2003, Visual C# .NET 2003, Visual C++ .NET 2003,Visual J# .NET 2003 і планується продовжувати ці роботи.

5.3. Деякі можливі порівняння

Разом з тим, до сих пір спеціалістами дебатуються питання ефективності і глобальності з точки зору широти спектра проблем, що покриваються тою чи іншою мовою програмування. На цьому фоні Міжнародна Комісія у складі членів комітетів по освіті організацій IEEE-CS[2] и ACM[3] прийшли до думки (рос.-мнению), що спеціалістам у області інформатики на початковому етапі необхідно знання не менш двох мов. При цьому, багато хто з експертів схиляється до думки, що освоєння мов природніше призводити, починаючи з більш простої, до якої як раз і відноситься Turbo Pascal (див. далі табл. 5.7).

Чим же «вимірюється» складність мови? Крім стильових і концептуальних особливостей, що пов'язані з галуззю його використання, у більшості мов присутній так званий лексемний рівень. Лексеми подають (рос.-представляют) мінімальні одиниці мови, які мають значення у його структурі: ідентифікатори, буквені константи, знаки операцій (оператори) і розподілювачі. В мовах процедурного напряму велике значення мають операції: арифметичні (+, –, *, /), порівняння, логічні і ряд інших. У деяких мовах їх йменують операторами або функціями. У складних виразах з їх участю необхідно враховувати пріоритет кожної з операцій. У приведеному нижче операторі привласнення мови Турбо Паскаль спочатку буде виконана операція ділення (/), яка має більш високий пріоритет, а потім результат відніметься з числа 20:

```
R := 20–8/4
```

У таблиці 5.7 приведені співвідношення кількості операцій и рівнів їх пріоритету у деяких популярних мовах програмування.

З таблиці слідує, що найбільш «легкими», тобто ті що мають відносно невелику кількість операцій і рівнів приоритету, а також з точки зору простоти засвоєння можливостей мови є Turbo Pascal і Visual Basic. Однак, з точки зору об'ємів засобів підтримки, які включають опис мови і безпосередньо середовище розробки IDE, Turbo Pascal зостається неперевершеним до сих пір. Крім того, у основі любого мови програмування лежать принципи організації структур даних і, як правило, деякий набір операторів мови. Засвоїв принципи їх використання на базі мови Turbo Pascal, можна застосовувати ці знання для вивчення інших мов, які безпосередньо відносяться до сфери професійної діяльності.

[2] Institute of Electrical and Electronics Engineers-Computer Science – Інститут інженерів з електротехніки і електроніки-комп'ютерні науки.
[3] Association for Computing Machinery – Асоціація з обчислювальної технікі.

Таблиця 5.7.

Порівняльна таблиця деяких характеристик мов програмування

Назва мови програмування (МП)	Кількість операцій/ операторів (типу +, –, *, /, >, < і т.д.)	Кількість рівнів пріоритету операцій/ операторів
Turbo Pascal	20	4
Visual Basic	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

Вже здавна триває полеміка серед спеціалістів щодо порівняння мов Turbo Pascal та C++ (яка по деяким даним входить у число трьох найбільш популярних на мировому рівні мов програмування разом з VB і Java), тому є інтерес відмітити деякі особливості мови C++. Суттєва направленість мови C++ на роботу з апаратурою і компонентами персональних комп'ютерів забезпечується 52-мя операторами роботи з даними, які об'єднані у 18 груп різного пріоритету. Сюди, доречи, входять постфіксні і префіксні інкременти (++) і декременти (--), оператори присвоювання: з множенням (*=), діленням (/=), діленням по модулю (%=), сумою (+=), різністю (-=), зсувом уліво (<<=) і зсувом вправо (>>=). Сюди ж входять побітові операції: І (&), АБО (|), побітове, що виключає АБО (^), логічні І (&&) і АБО (||), тернарна операція (?) і багато інших. У той же час у мові Turbo Pascal 7.0 використовується усього 20 операцій роботи з даними, які об'єднані у 4 групи за пріоритетами виконання, що усього у ***чотири рази менш*** за кількість пріоритетів у мові C++ (!!!).

У алгоритмічних мовах ***програма*** – це "***алгоритми + структури даних***".

Об'єктно-орієнтовані (ОО) мови уявляють собою середовище, у якому програмісти визначають не тільки типи структур даних, але й типи операторів (функцій), котрі застосовуються до цих структур даних. У такому контексті утворюються об'єкти, які уключають і дані, і функції (так звані методи). Функціонально подібні об'єкти утворюють класи. У рамках інформаційної системи, що утворюється, програмісти повинні спроектувати зв'язки поміж об'єктами, які моделюються з урахуванням властивостей, що наслідуються з класів.

Надзвичайно важливим уявляється також той факт, що мова C++ є суттєво ***об'єктно-орієнтованою*** і визначально припускає у того, хто її вивчає чітке уявлення про ОО принципи, які уключають наступні фундаментальні поняття: ***ОБ'ЄКТ, ПОВІДОМЛЕННЯ, КЛАС, НАСЛІДУВАННЯ І МЕТОД***. У контексті мови постійно і широко використовуються принципи, які лежать в основі об'єктної моделі уявлення систем, які програмно моделюються: ***абстрагування, інкапсуляція, поліморфізм, модульність, ієрархічність, типізація, паралелізм і збереженість (рос.-сохраняемость)***. Механізмами реалізації вказаних

абстракцій є віртуальні і не віртуальні функції, функції і методи, які перевантажуються, функції-члени базових класів і їх об'єкти, потоки, буфера і їх класи, шаблони, які об'являють параметризовані класи масивів, класи і екземпляри шаблонів і т.д. Іншими словами, алгоритмічна складова у цієї мові служить основою реалізації методів об'єктів, при вельми високої ступені абстракції уявлення елементів ієрархічної структури організації взаємодії абстрактних даних у вигляді базових і віртуальних класів систем, що реалізуються і об'єктів, що ними породжуються. Все це робить мову C++ достатньо складною для засвоєння.

Розширення горизонтів застосування комп'ютерів і удосконалення мов програмування потребує вивчення усе нових й нових не тільки мовних інструментів, але й їх діалектів.

Можна назвати як мінімум три десятка мов, котрі відіграли помітну роль у розвитку програмування, але все ж тільки три пари — Алгол-60 і Фортран, Паскаль і C, Java і C++ — стали найбільш яскравими, найбільш помітними на комп'ютерному небосхилі.

Говорячи про відведене (рос.-отстранённое) і упереджене (рос.-предвзятое) відношення людей до «чужих» мов, Ніклаус Вірт, автор мови Паскаль відмічав: «Багато хто відноситься до стилів і мов програмування, як до релігійних конфесій: якщо ви належите до однієї з них, то вже не можете належати до іншої. Але це неправильна аналогія, і вона свідомо підтримується по причинам комерційного порядку».

Опонент Ніклауса Вірта — Денніс Рітчі— недавно відмітив: «Паскаль — дуже елегантна мова. Вона до сих пір жива. Вона породила чимало своїх послідовників і зробила (рос.-оказала) глибоке діяння (рос.-воздействие) на проектування інших мов».

У інтерв'ю, яке було дане 21.02.2001 системному аналітику Денні Калєву (Danny Kalev), творець одної з найбільш поширених мов програмування C++, Б'єрн Страуструп сказав наступне:

«Багато чого у програмуванні краще зробити методами, котрі не містяться всередині вузької смуги (рос.-полоски) методів, що іменуються "об'єктно-орієнтованими". І якщо Ви не виходите за границю "об'єктно-орієнтованих" методів, щоб зостатися у рамках "доброго (рос.-хорошего) програмування і проектування", Ви отримаєте дещо, що є у цілому безглуздим (рос.-бессмысленным). Майбутнє за ***мультіпарадигматичним програмуванням***».

Далі Б'єрн Страуструп додав:

«… Я думаю, що аби яка мова, котра прямує (рос.-стремится) до пануючого (рос.-господствующего) положення в усіх областях, повинен забезпечити широку основу для декількох методів, уключаючи об'єктно-орієнтоване програмування (на основі ієрархії класів) і узагальнене (рос.-обобщенное) програмування (параметризовані типи і алгоритми). Зокрема, необхідно забезпечити добрі (рос.-хорошие) засоби для створення програм з окремих (незалежних) частин (можливо, написаних на різних мовах). Я також думаю, що для управління складним процесом обробки помилок необхідні

виключення (рос.-исключения). Мова, у котрій відсутні такі засоби змушує його користувачів моделювати їх (що веде до додаткових помилок і витрат)».

Не дивно, що продовжують жити багато старих добрих мов: Fortran, COBOL, Ada та інші. Зокрема, в червні 2001 року фірма COMPAQ випустила на ринок інтегроване візуальне середовище розробки з оптімізуючим компілятором Compaq Visual Fortran 6.6 (рис. 5.5).

Рис. 5.5. CD с компілятором Compaq Visual Fortran 6.6

У проведеному в 2001 році дослідженні[3] була зроблена оцінка програмного забезпечення, що відноситься до так званих "відкритих кодів" (Open Source) і розташоване на європейському сайті «The Site for Libre Software Developers [http://libre.act-europe.fr/]. Оказалося, що у комплексі досліджуваних програмних розробок, що займали у запакованому вигляді простір на диску більш ніж 1 гігабайт і число розробників котрих з 41-єї країни перевищує 5300 особистостей, мови програмування, які всі вони використовували розподілилися наступним чином (рис. 5.6).

З нього витікає, що більшість професійних програмістів володіють мовами C і C++, але використовують у своїй роботі і мову Паскаль.

Це і не дивно, бо мова Паскаль, яка створена у 1968 р. Ніклаусом Віртом, розроблялася визначально для навчання написанню надійних програм з розвинутою системою даних і є учбовою мовою високого рівня, для котрої у 1982 року була створено унікальне інтегроване середовище розробки Turbo Pascal. Версія Turbo Pascal 7.0 фірми Borland International:

❶ Має вбудований редактор, компілятор, засоби запуску програм на виконання, дебаггер (налагоджувальник) (рос.-отладчик), який дозволяє не тільки тестувати програми, але й *досліджувати* їх роботу.

❷ Дозволяє систематично і точно висловлювати концепції і структури елементів програмування.

❸ Показує, що використання машинно-незалежної мови програмування з гнучкими структурами даних і управляючими конструкціями приводить до

[3] 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

підвищення "удобочитаємості", верифицируемости[4] і, відповідно, надійності без позбавлення (рос.-потери) ефективності.

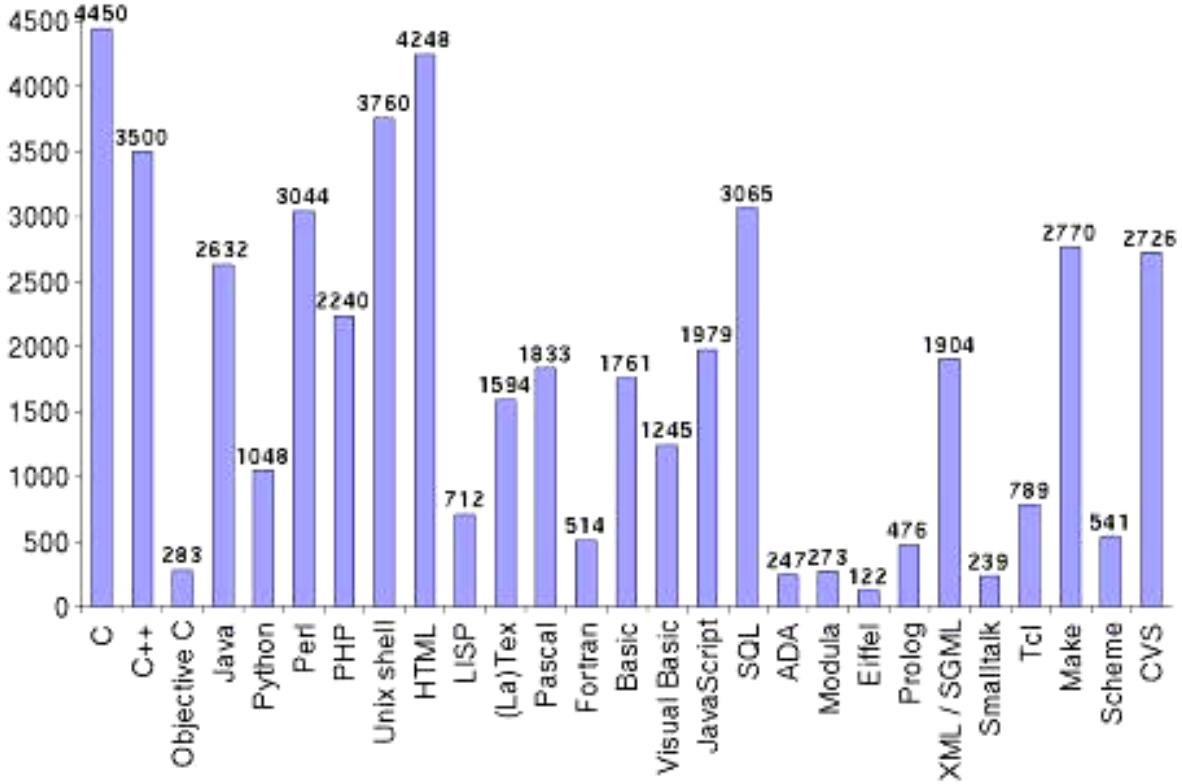


Рис. 5.6. Чисельне співвідношення мов програмування і програмістів, що їх використовували (більшість програмістів працювали з декількома різними мовами)

❹ Turbo Pascal 7.0 сприяє поліпшенню розуміння методів організації великих програм і методів управління програмістськими проектами.

❺ Має розвинуті засоби діагностики помилок, ефективні інструменти налагодження і по цим причинам є дуже зручним засобом для навчання програмуванню.

❻ Включає бібліотеку Turbo Vision з потужними засобами об'єктно-орієнтованого програмування.

❼ Має компактні розміри інтегрованого середовища розробки і документації з описом її роботи і конкретно конструкцій мови Borland Паскаль.

Оскільки мова Паскаль фактично реалізує віртуальну машину високого рівня (Паскаль-машину) з підсиленим мовним обмеженням, це затрудняє користувачу вихід за її границі, що у процесі навчання і є важливим позитивним моментом.

На рисунку 5.7 представлена просторово-часова (рос.-пространственно-временная) картина розвитку мови Паскаль у структурі решти (рос.-остальных) найбільш значимих мов програмування.

[4] Верифікація –процес підтвердження виконання програмою функцій, що у неї закладені, тобто перевірка правильності програми шляхом формального доказу відповідності програми завданій специфікації.

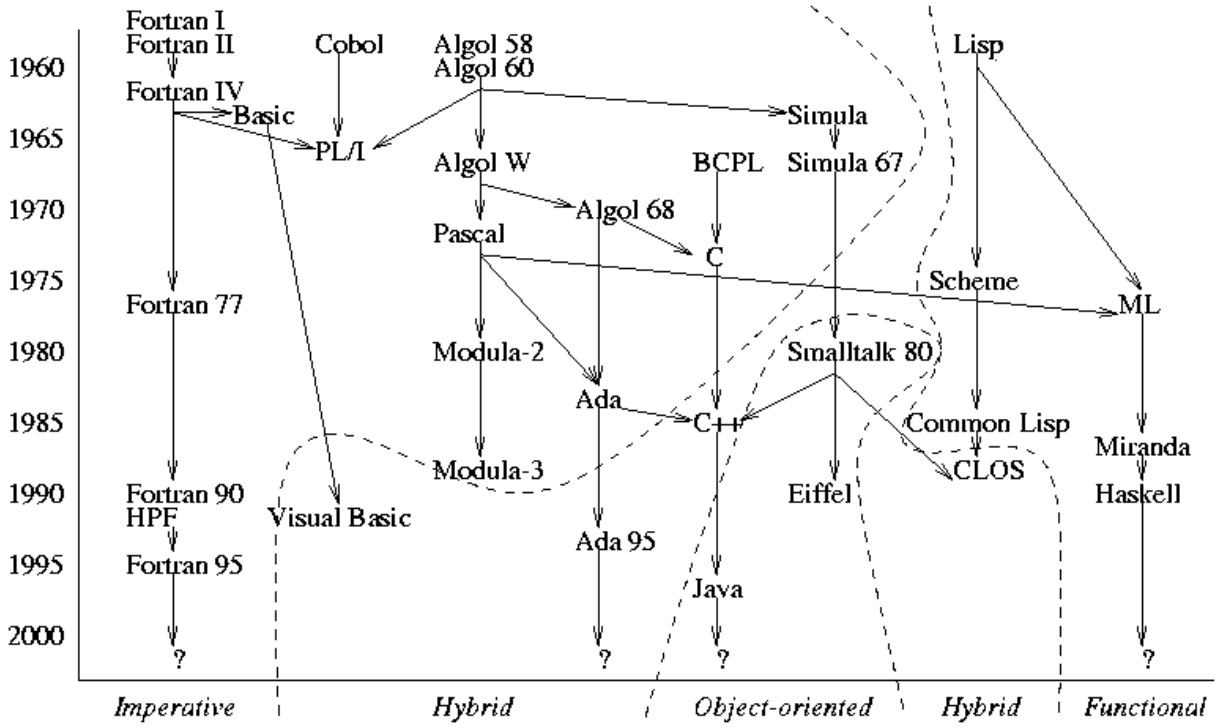


Рис. 5.7. Уявлення процесу розвитку мов програмування
з поділенням їх на наступні групи мов:
–імперативні (imperative);
–гібридні (hybrid);
–об'єктно-орієнтовані (object-oriented);
–функціональні (functional).

Якщо дивитися на мови програмування, як на інструмент обробки даних, всі вони мають наступні ***головні базові елементи***:

❶ базові оператори опису даних і їх структур;

❷ базові інструкції, за допомогою яких реалізуються процедурні компоненти програм;

❸ базові операції (оператори): арифметичні, логічні і інші.

❹ рівні пріоритету їх виконання.

Тому, можна стверджувати, що при знаннях принципів використання і взаємодії базових елементів, хоча б одної мови програмування, можна значно облегшити освоєння практично аби якого іншого.

Запитання.

1.	Що взагалі зветься мовою?
2.	Чим відрізняється мова людини від мови програмування?
3.	Для чого потрібно компілювати аба інтерпретувати мову програмування?

4.	Чим відрізняється процес і результат компіляції від процесу і результату інтерпретації?
5.	Чим відрізняється мова програмування системи від мови сценарію?
6.	Які мови програмування зараз найбільш розповсюджені?
7.	Які головні стилі програмування застосовуються при розробці програм?
8.	Як пов'язані мови програмування з типами задач, що вирішуються?
9.	Що визначають пріоритети виконання операцій у мовах програмування і як вони впливають на складність організації програм?
10.	Які спільні базові елементи мають практично усі мови програмування?

Розробка рішення бізнес-проблеми — це більше, ніж просто написання найскладнішого додатку (рос.-приложения) з застосуванням новітніх технологій. Розробка додатку повинна вестися, на основі бізнес-потреб. Якщо ж воно розроблене, додаток повинен бути прийнятий кінцевими користувачами, котрим він зобов'язаний допомагати у рішенні поточних проблем.

Дональд Р. Брандт. "Архітектура. Екзамен Microsoft – екстерном".

6. ЗМІНА МЕТОДОЛОГІЇ СТВОРЕННЯ ПРОГРАМ

6.1. Тенденції розвитку інформаційно-комп'ютерних технологій

Багато змін, які впливають на інформатику пов'язані з прогресом у розвитку комп'ютерних (електронних) технологій (КТ) і нерозривно пов'язаних з ними інформаційних технологій (ІТ, а у англійської транслітерації, IT-Information Technologies), що у комплексі має назву інформаційно-комп'ютерних технологій (ІКТ). Більшість нових досягнень у цієї галузі уявляє собою частину постійного еволюційного процесу, котрий продовжується вже довгі роки. Закон Мура (прогноз, зроблений у 1965 році фундатором фірми Intel Гордоном Муром, і стверджуючий, що щільність (рос.-плотность) транзисторів на кристалі мікропроцесора і частота його роботи буде підвищуватися удвічі кожних вісімнадцять місяців) і по сей день залишається у силі. Як результат дії цього закону можна спостерігати експоненціальний зріст обчислювальних можливостей комп'ютерів, завдяки чому з'явилась можливість вирішення задач, котрі були не розв'язувані усього декілька років тому. Треба також додати, що ємність (рос.-ёмкость) дискової пам'яті комп'ютерів зростає удвічі ще скоріше – кожні 9 місяців. І зовсім фантастичними темпами збільшується пропускна здібність ліній зв'язку – удвічі кожні 4-6 місяців. Тільки у Китаю, у секторі телекомунікацій сукупний об'єм інвестицій у цю сферу, згідно статистичного звіту китайського уряду, склав у 2003 році $30,1 млрд. Зараз Китай займає друге місце у світі після США по протяжності своїх телекомунікаційних та інформаційних мереж.

Поточним часом практично для усіх організацій характерно критичне нарощування об'ємів інформації. Зріст об'єму даних у інформаційних системах ставить перед ІТ-персоналом задачу побудови сучасних систем зберігання даних. По даним компанії Meta Group (Стемфорд, шт. Коннектикут, США) у 2002 році, об'єм інформації, акумульованої у майже будь якій організації США зростає удвічі кожні 18 місяців. По даним International Data Corp., платня за підтримку об'ємів даних, які неминуче збільшуються, складає сьогодні

приблизно половину усіх видатків на IT-рішення. Настільки дорогі засоби забезпечення постійного доступу до інформації розробляються для того, щоб уникнути ще більших витрат, пов'язаних з утратою доступу до сховищ даних (до декількох мільйонів доларів у годину у залежності від додатку). Ці утрати складаються з суми збитків від зменшення ефективності праці співробітників з причини простою системи зберігання даних, коштовності робіт, котра не могла бути виконаною у період простою, а також коштовності ремонту елементів, що вийшли з ладу.

Галузь інформатики, у котру вкладаються величезні кошти і яка приносить, у свою чергу, надвисокі прибутки, постійно знаходиться під пильною (рос.-пристальной) увагою ведучих наукових і державних організацій практично усіх розвинутих країн світу (рис. 6.1, тут і далі дані з серверу: http://www.nsda.ru/home.asp).

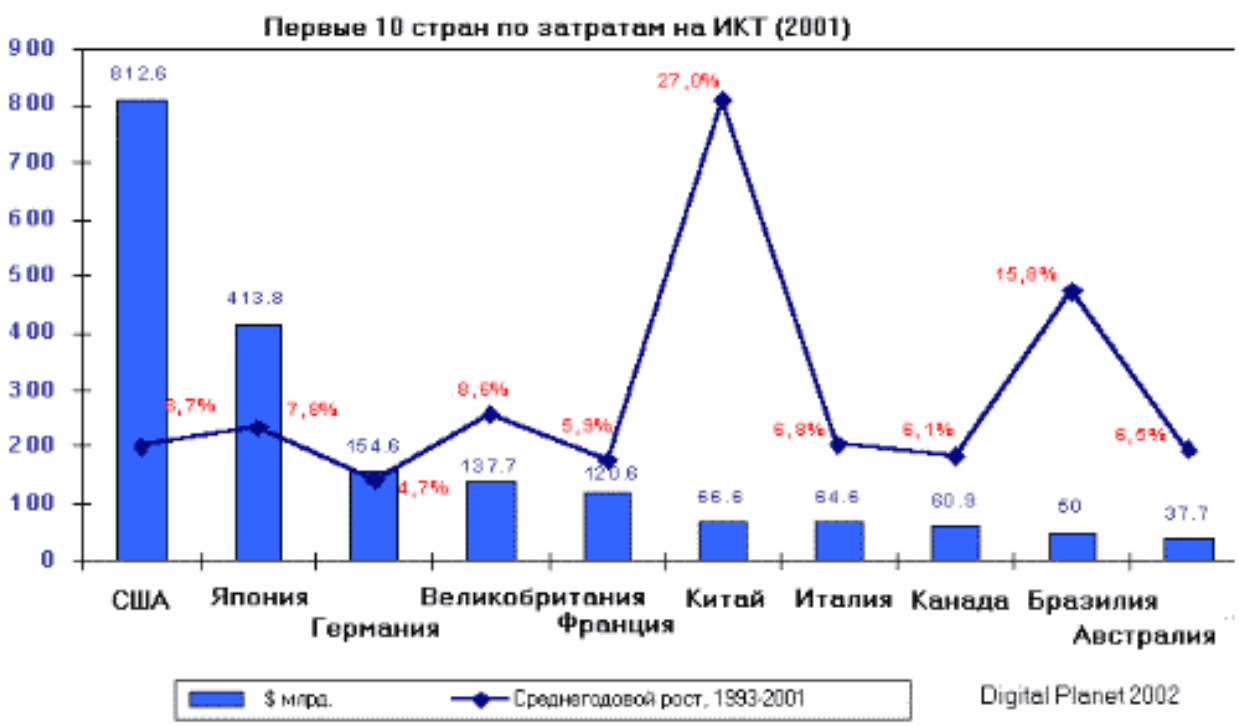


Рис. 6.1. Порівняння коштів, що вкладалися розвинутими країнами світу у ІКТ галузь у 2002 році

До головних факторів, які визначають розвиток галузі ІКТ, можна віднести наступні:

❶ активне впровадження ІКТ у всі структури і галузі матеріального виробництва поточним часом є стратегічним напрямом, який підтримується урядами усіх розвинутих країн світу;

❷ рівень складності проблем, які вирішуються комп'ютерними засобами постійно підвищується;

❸ кількість спеціалістів, які володіють потрібним комплексом знань з інформатики значно менше потрібного (тому тільки у 2002 р. у ряді розвинутих країн були додатково виділені десятки і сотні тисяч так званих "зелених карт" для в'їзду до розвинутих країн "просунутих (рос.-продвинутых)" спеціалістів у

галузі інформаційних технологій;

❹ постійно розширюється спектр нових технологій і, відповідно, нових галузей застосування ІКТ.

Вже давно не секрет, що пріоритети і рівень розвитку комп'ютерної індустрії визначає група держав, яку очолює США. Красномовніше (рос.-красноречивее) за все це говорять цифри. На кінець 2002 г. географічний розподіл доходів ринку інформаційних технологій виглядало так: США –46%, Європа – 29%, Японія – 13%, Азія – 4%; Канада –3%, інші країни – 5% (рис. 6.2).

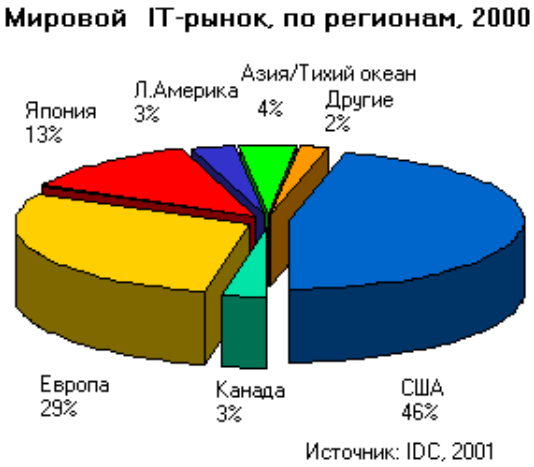


Рис. 6.2. Розвиток мирового ринку IT по регіонам світу у 2001 р.

У перегоні (рос.-гонке) за інформаційне панування (рос.-господство) виграють капітали, що вкладаються у галузь. Не дивно, що серед «гегемонів» – виробників базового компонента інформаційних систем – мікропроцесорів, з декількох десятків фірм, які стартували у 70-і роки лідерами зосталися фактично двоє: Intel і AMD (США), котрі вивели їх робочі частоти у 2003 році за відмітку 3 ГГц. Ринком операційних систем і офісних додатків у 2003 р. монопольно володіла Microsoft (США) (по даним 2002 г. відповідно 90% і 86%), а серед розробників баз даних – Oracle (США). Це і не дивно, бо за експертними оцінкам 2003 року, у російській галузі ІКТ працювало біля 0,54 млн. чоловік, а у США – біля 10,5 мільйонів. Потрібно відмітити, що у це число, ясна річ, не увійшли ти фахівці IT, котрі за гроші США виконують їх замовлення на створення програмного забезпечення на своїй батьківщині: у Індії[4], Ірландії[5], Україні та інших країнах світу. По оцінкам міжнародної дослідницької компанії International Data Corporation (IDC), об'єм світового

[4] У 2003 р. Індія входила до першої десятки країн світу по ряду показників виробництва. По об'єму ВВП вона вже посідала 4-е місце у світі після США, КНР і Японії. Щорічно тут випускається біля 2,5 млн. ПК, 10,5 млн. телевізорів, 20 млн. радіоприймачів і магнітол, 35 млн. електронних годинників, 1 млн. відеомагнітофонів. Країна посідає 3-е місце у світі по чисельності науково-технічного персоналу і друге - по кількості професіоналів комп'ютерного програмування.

[5] Однією з причин 10-відсоткового економічного зросту в Ірландії у 1999 році явилось то, що вона перетворилась у справжню "Силіконову долину" Європи. Тут розмістилась європейська штаб-квартира Microsoft. Фірма Intel виготовляє тут свої процесори Pentium, а Apple, IBM і Dell ведуть тут наукові дослідження. По виробництву програмного забезпечення Ірландія вийшла на 2-е місце у світі після США. Дослідницькі центри Ірландії приймають участь у розробці нових мікрочипів і мов програмування.

ринку інформаційних технологій у 2000 р. досягнув 395 млрд. дол., а у 2001 р. - 440 млрд. дол. і у перспективі буде продовжувати рости (див. рис. 6.3).

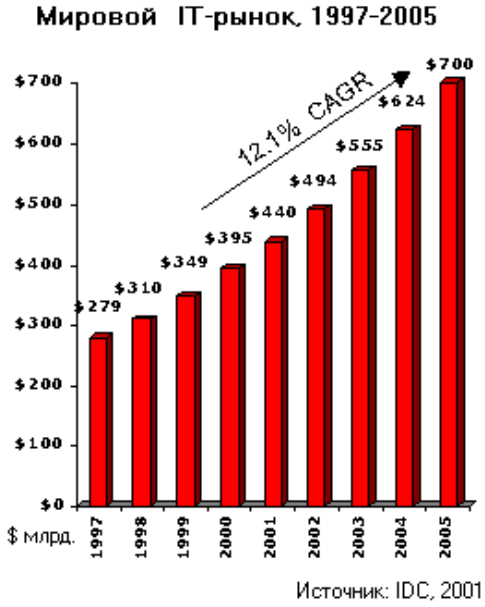


Рис. 6.3. Прогноз розвитку світового ринку IT у часу (2001 г.)

Зріст сукупного доходу ІКТ індустрії до 2002 р. ще більше вражає (рис. 6.4). Дані на 2003 р. 2680 млрд. Євро є прогнозом.

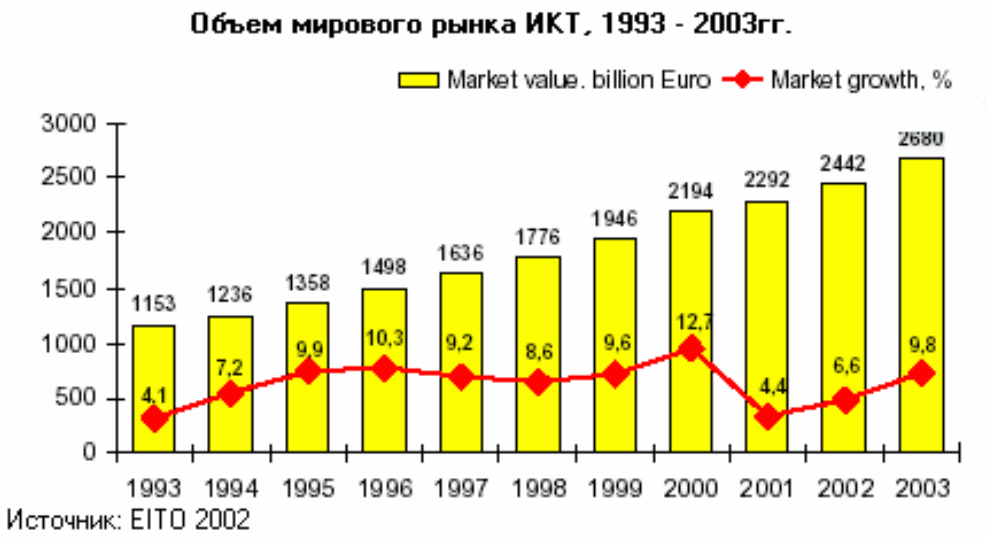


Рис. 6.4. Дані по розвитку світового ринку ІКТ на 2002 р.,(у млрд. Євро)

Межнародна дослідницька корпорація IDG/IDC для визначення рівня розвитку високотехнологічної галузі обчислює зведений індекс ISI (Information Society Index, зведений індекс розвитку інформаційного суспільства) на основі ряду показників:

❶ кількості персональних комп'ютерів на душу населення;
❷ рівня використання ПК у промисловості, державних органах і освіті;
❸ вартості програмного забезпечення на один комп'ютер;

❹ частини ПК, що підключені до Інтернету; пропускну властивість і вартість каналів зв'язку.

Згідно цим показникам, зведений індекс розвитку і валового внутрішнього продукту (ВВП) на душу населення деяких країн у 2003 році виглядав наступним чином (таблиця 6.1):

Таблиця 6.1.

Зведені індекси ISI і ВВП на душу населення по країнам світу у 2003 році.

Країна	ISI	ВВП	Країна	ISI	ВВП
Швеція	5062	20,7	**Польща**	1808	7,2
США	5041	**33,9**	**Чилі**	1635	12,4
Японія	4093	**23,4**	**Аргентина**	1651	10
Сінгапур	4014	**27,8**	**Малайзія**	1583	10,7
Германія	3558	22,7	**ПАР**	1537	6,9
Тайвань	3177	16,1	**Венесуела**	1491	8
Ірландія	3144	20,3	**Росія**	1444	4,2
Ізраїль	3140	18,3	**Філіппіни**	1012	3,6
Франція	3140	23,3	**Китай**	915	3,8
П.Корея	2931	13,3	**Індонезія**	888	2,8
Італія	2703	21,4	**Індія**	871	1,8
Чехія	2130	11,7	**Пакистан**	719	2

Цікаво відмітити, що, перші три місця по ВВП на душу населення займають США, Сінгапур і Японія. Не дивно, що у недавно опублікованому у США (2003 р.) звіті під назвою "The Digital Work Force: Building InfoTech Skills at the Speed of Innovation", підкреслено, що потреба в США у працівниках, які мають знання з інформаційних технологій збільшиться більш ніж удвічі до 2006 року, у той час як потреба в усіх інших спеціальностях збільшиться у середньому тільки на 14% (!).

Згідно даним багаточисельних закордонних і вітчизняних джерел головну роль у всіх комп'ютерних напрямах (див. Вступ) грає у першу чергу ***програмування***.

Зараз на зміну програмуванню процедурному, з котрого і починалося програмування як напрям, прийшло програмування об'єктно-орієнтоване, а за ним і компонентне. Об'єктно-орієнтований метод має настільки суттєві переваги, що переважна більшість програмного забезпечення, що створюється використовує переваги об'єктних технологій у той або іншій формі. У деяких мовах програмування є можливість створення об'єктів різними способами: або з "нуля", або використовуючи механізм наслідування і об'єкти заготовки – класи. Мови сценаріїв застосовуються для роботи с готовими об'єктами: об'єктам можна посилати повідомлення і отримувати від них реакцію у вигляді методів, що виконуються або даних і т.д.

Разом з тим, продовжує оставатися відкритим і найбільш широко розповсюдженим на усіх рівнях спеціалістів головне питання ІКТ галузі:

Яка ж мова найбільш універсальна для написання програм і, відповідно, яку ж мову у першу чергу потрібно вивчати?

Щоб спробувати об'єктивно підійти до відповіді на це питання, спочатку потрібно відповісти на наступні декілька питань:

❶ Що собою уявляє програма?
❷ У якому середовищі пишуться програми і додатки?
❸ У якому середовищі працюють програми і додатки?
❹ Як проектуються додатки?
❺ Які існують додатки?

6.2. Що собою уявляє програма?

Для відповіді на це питання реалізуємо на мові Турбо Паскаль стандартний вивід знайомого усім програмістам тексту "Hallo, World!" на дисплей комп'ютера (рис. 6.5).

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

Рис. 6.5. Вивід на дисплей ПК тексту "Hallo, World!" у мові ТП

Фрагмент тексту, який приведено на рисунку 6.5 і який складається з чотирьох інструкцій (пропозицій (рос.-предложений) або операторів) мови ТП, з точки зору сучасних інформаційних уявлень можна ***одночасно*** назвати наступним чином (тобто розглядати з таких точок зору) (табл. 6.1).

Таблиця 6.1.

Найменування набору інструкцій в мовах програмування

№ пп	Найменування набору інструкцій мови програмування і його сутність (рос.-сущность)
1	***Інструкції***, це ***дані*** *(рос.-данные)* призначені для управління конкретними компонентами системи обробки даних у цілях реалізації деякого алгоритму.
2	***Код*** – опис дій, які повинен виконувати комп'ютер, записаний на мові низькорівневого програмування або у машинних кодах.
3	***Програмний код*** або просто ***код*** – послідовність команд або операторів аби якої мови програмування, котра після декодування її комп'ютером і транслюючою програмою примушує (рос.-заставляет) останній виконувати деяку роботу.
4	***Фрагмент коду*** – те ж саме, що і ***код.***
5	***Програма –*** опис операцій, котрі потрібно виконати для рішення поставленої задачі. Дії, що описуються програмою, звуться *операторами*. *Командою* іменують елементарний припис (рос.-предписание), який передбачає виконання якої-небудь *операції.*
6	***Послідовність операторів*** – упорядкована послідовність команд, яка підлягає (рос.-подлежит) обробці.
7	***Модуль*** – програма, котра розглядається як окреме ціле у процесах збереження, трансляції і об'єднання з іншими програмними модулями при їх завантаженні у оперативну пам'ять комп'ютера для виконання.

Продовження таблиці 6.1.

№ пп.	Найменування набору інструкцій мови програмування і його сутність (рос.-сущность)
8	***Підпрограма*** – пойменована частина програми, котра викликається і отримує параметри, виконує визначені дії і повертає результат своєї роботи і управління у точку виклику. У багатьох мовах програмування відрізняють два види підпрограм: –***процедури***, дія котрих полягає (рос.-заключается) у зміні значень параметрів і деякому побічному ефекті. Як правило є операторами або інструкціями мови програмування; –***функції***, котрі повертають залежний від параметрів результат. Є операндами у конструкціях мови програмування і описуємих за їх допомогою виразах.
9	***Процедура-підпрограма, процедура-функція (у мові VBA)***– відповідно підпрограма і функція. Підпрограма може виконувати роль програми.
10	***Метод*** - це ***процедура*** або просто ***набір команд***, які повідомляють об'єкту, що потрібно виконати деяку задачу і реалізують алгоритм її виконання.
11	***Додаток (рос.-приложение)*** – прикладна програма, тобто програма, що виконується під управлінням операційної системи.
12	***Рішення (рос.-решение) (термін з Microsoft Solution Framework (MSF))*** 1)Програмний продукт. 2)Координована поставка набору елементів (таких, як програмно-технічні засоби, документація, навчання і супроводження), необхідні для задоволення деякої бізнес–потреби конкретного замовника. (MSF). 3)Множина (рос.-множество) файлів різного типу, у рамках одного *рішення (solution)* складають ***додаток*** *(application)* Visual Studio.Net 7.0.
13	***Операція*** *(operation)*-сервіс або послуга, котрі можуть бути заказані у об'єкта у вигляді результата його поведінки. Операція має ***сігнатуру***, котра може мати параметри. ***Сігнатура*** *(signature)*- ім'я і параметри властивості з поведінкою (рос.-поведенческого свойства). Сігнатура може включати опціонально повертаємі параметри. ***Метод*** *(method)*- реалізація операції. Вона специфіцірує алгоритм або процедуру, пов'язані з операцією. (UML).

Перший тезис, що програма це дані, підтверджується тим простим фактом, що ***усі без виключення програми зберігаються на магнітних дисках або у базах даних (також на дисках !)*** і починають виконувати свою роботу під управлінням операційної системи тільки на командний стимул і, як правило, після завантаження у оперативну пам'ять комп'ютера (рис. 6.6).

Більш того, текстові стрічки програмного коду є ***вхідними даними*** для трансляторів (компіляторів) і інтерпретаторів, які перетворюють їх у конкретні машинні команди, що їх виконує мікропроцесор.

Крім того, продививши таблицю 6.1. зверху униз, можна побачити наскільки многогранним є уявлення ***програмної компоненти*** у вигляді ***коду*** на аби якій ***мові програмування***.

При тому, у більшості своїй, інструкції програм і сучасних програмних систем записуються у восьмибітних ASCII ("аскі") кодах. І тому, взагалі кажучи, коди програм на багатьох мовах програмування можна набирати у аби яких текстових редакторах файл-менеджерів (Norton Commander, Far Manager, Total Commander і ін.), або редакторах операційних систем типу Блокнот

(Notepad). Те ж відноситься і до кодів мов розмітки: HTML, XML і деяких інших.

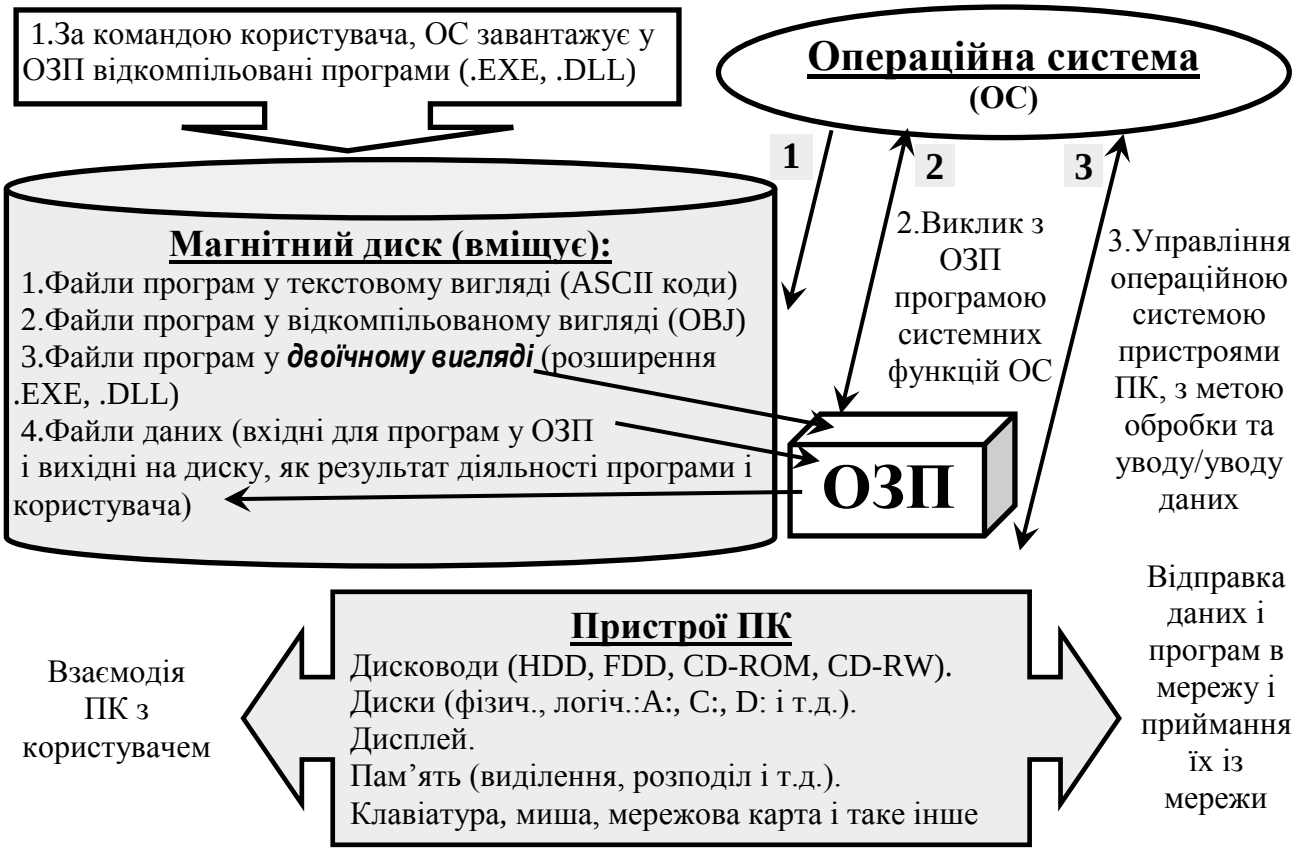


Рис. 6.6. Типовий процес виконання програми на ПК

6.3. У якому середовищі пишуться програми і додатки?

Якщо текст програми вже є, то яким же чином можна подати текстовий фрагмент коду на вхід транслятора?

Найбільш простими у використанні для цих цілей є Інтегровані системи розробки (ІСР, а англійською мовою – IDE – Integrated Development Environment,), які широко представлені продуктами фірми Borland International (Turbo Pascal, Turbo C, Turbo Prolog), Microsoft Visual Basic, PowerBuilder, Inprise Delphi Object Pascal, Oracle Developer PL/SQL і деякі інші, котрі відносяться до мов 4-го покоління (4GL, Generation Language 4). Ці продукти є продовженням розвитку мов програмування (типу Фортран, Алгол, С, С++, COBOL, Ada, Pascal), які у минулому були позбавлені багатьох сервісних можливостей і тому покладавши на програмістів виконання усіх дій по організації процесу уводу програмного коду у ПК, визову транслятора, завантаження відтранслёваних модулів у пам’ять і т.д. Їх звуть мовами 3-го покоління 3GL (Generation Language 3), тому що вони не були встроєні у інтегровані інтерактивні середовища розробки програмного забезпечення.

Щоб краще уяснити принцип роботи ІСР, розглянемо роботу з типовим додатком IDE Турбо Паскаль. По-перше, користувач повинен підготувати текст додатку (програми), подати його на вхід транслятора, отримати двоїчний код і

записати його на диск у файл з розширенням EXE (рис. 6.6). Потім, зчитати двоїчний код додатку у пам'ять комп'ютера, виконати ту частину коду, котра відповідає за організацію діалогу з користувачем, котрий повинен визначити необхідний для роботи набір даних. Потім додаток читає набір даних з диску (який можливо вже був ліквідований). Далі користувач взаємодіє з додатком, виконуючи маніпуляції з уявленням даних у головній пам'яті. Ця фаза продовжується більшу частину часу роботи додатку до тих пір, поки у кінці кінців модифікований набір даних не запишеться на диск, як результат вирішення задачі користувача.

Для організації комплексності такої технології роботи у ІСР включаються усі необхідні інструменти, які забезпечують здійснення усього фронту робіт, що були описані вище. Тому проходження рішення типової задачі, наприклад, у рамках ІСР Турбо Паскаль, можна уявити наступним чином (рис. 6.7).

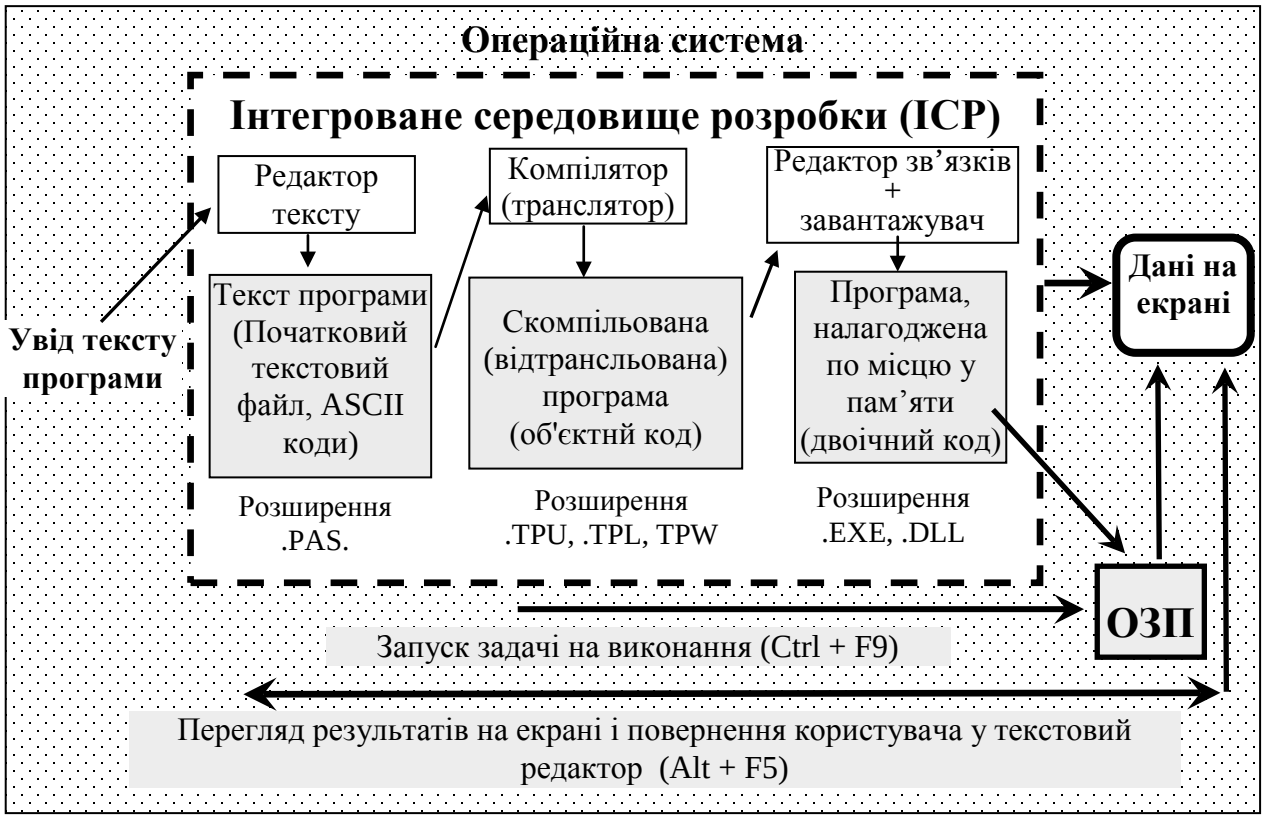


Рис. 6.7. Етапи виконання програми у ІСР ТП

У ІСР, як правило, інтегровані: текстовий редактор для набору кодів програм, транслятор, редактор зв'язків і завантажувач, тому усі файли (PAS, TPU, EXE, DLL та інші), які створюються, зберігаються на магнітному диску. Окрім того, у середовищі завжди присутні засоби відлагодження програм (дебаггери) і багато інших додаткових сервісних можливостей. Оскільки розробка подібних засобів велася ще до періоду розквіту графічних можливостей комп'ютерів, робота з ІСР активно підтримується командами, що уявляються окремими клавішами і їх сполученнями (Esc, Ctrl + F9, Alt + F5 і так далі). Однак, не дивлячись на те, що у режимі MS DOS з Турбо Паскаля можна напряму працювати з портами, відеопам'яттю, секторами і доріжками

дисків і т.д., слід чітко уявляти, що всі ці дії підтримуються системними функціями відповідної операційної системи (MS DOS, Windows та ін.).

Поява операційної системи Windows суттєво ускладнила програмування, так як з позиції програміста ОС Windows дає останньому тільки набір системних функцій, які утворюють так званий API–інтерфейс прикладних програм (API–Application Programming Interface). Тому, щоб виконати ту чи іншу дію (наприклад, відкрити вікно на екрані ПК, накреслити лінію, створити кнопку та ін.), програма повинна звернутися до відповідної функції API.

Функції API операційної системи, котрих тільки у ОС Windows 95 нараховувалося більш ніж 800-т, утворюють її ядро і зберігаються у багаточисельних DLL-бібліотеках. Серед великої кількості інших вони реалізують наступні ***системні функції***:

❶ управління пам'яттю, завантаження/вилучення програм з пам'яті і забезпечення безпосередньої підтримки виконання програм;

❷ управління вікнами (створення, зміна розмірів, переміщення, вилучення) і т.д.;

❸ управління уводом з клавіатури і роботою з мишею;

❹ взаємодія з зовнішніми пристроями (екраном, принтером і ін.).

Ускладнення взаємодії з такою великою кількістю системних функцій привело до необхідності спрощення для користувачів самого процесу програмування і створення для цього інтегрованих середовищ ***швидкої розробки додатків*** (RAD – Rapid Application Development) для візуального і швидкого програмування (створення Windows-додатків). Піонерами цього процесу виступили корпорація Microsoft, яка створила RAD середовища Visual Basic і Visual Studio (для мов C++, Visual Basic та ін.), а також Borland International (потім Inprise), яка створила середовище Delphi[6] (з мовою Object Pascal). Будівними блоками програми стали компоненти – тобто стандартні об'єкти, які мають візуальне уявлення на стадії проектування і у час роботи (рис.6.8).

Популярність середовищу швидкої розробки додатків Delphi принесли потужна внутрішня об'єктно-орієнтована мова програмування Object Pascal і унікальні по своїй простоті і гнучкості засоби роботи з базами даних. У подальшому, починаючи від версії Delphi 5 і вищє у середовищі з'явилися можливості створення багатофункціональних додатків (Windows, Internet, WWW, багатоярусних (Multi-Tier) і ін.), серверів COM і CORBA, а також окремих компонентів. Компоненти, що їх використовують у розробці, уявлені у вигляді ієрархії класів, реалізовані на мові Object Pascal і організовані у Бібліотеку візуальних компонентів (Visual Component Library, VCL), яка підключається до Delphi IDE (або RAD). Велика популярність Delphi серед розробників пояснюється також і тим, що мова Object Pascal по своїм можливостям перевершує мову Visual Basic, мало поступається мові C++, а усвоюється у два-три рази швидше останньої. Код, який виконується у Delphi, є

[6] Автором Delphi у 1995 г. став Андерс Хейльсберг (Anders Hejlsberg), соавтор розробки системи Турбо Паскаль, а потім у 2000 р. у статусі ведучого розробника (distinguished engineer) корпорації Microsoft, він розробив мову C# (С-шарп) і став соавтором створення платформи .NET (Дот НЕТ) (http://www.microsoft.com/presspass/press/2001/apr01/04-11AndersPR.asp).

достатньо ефективним, швидкість компіляції дуже висока, а відладка, особливо у порівнянні з відладкою у C++, значно простіше і протікає швидше.

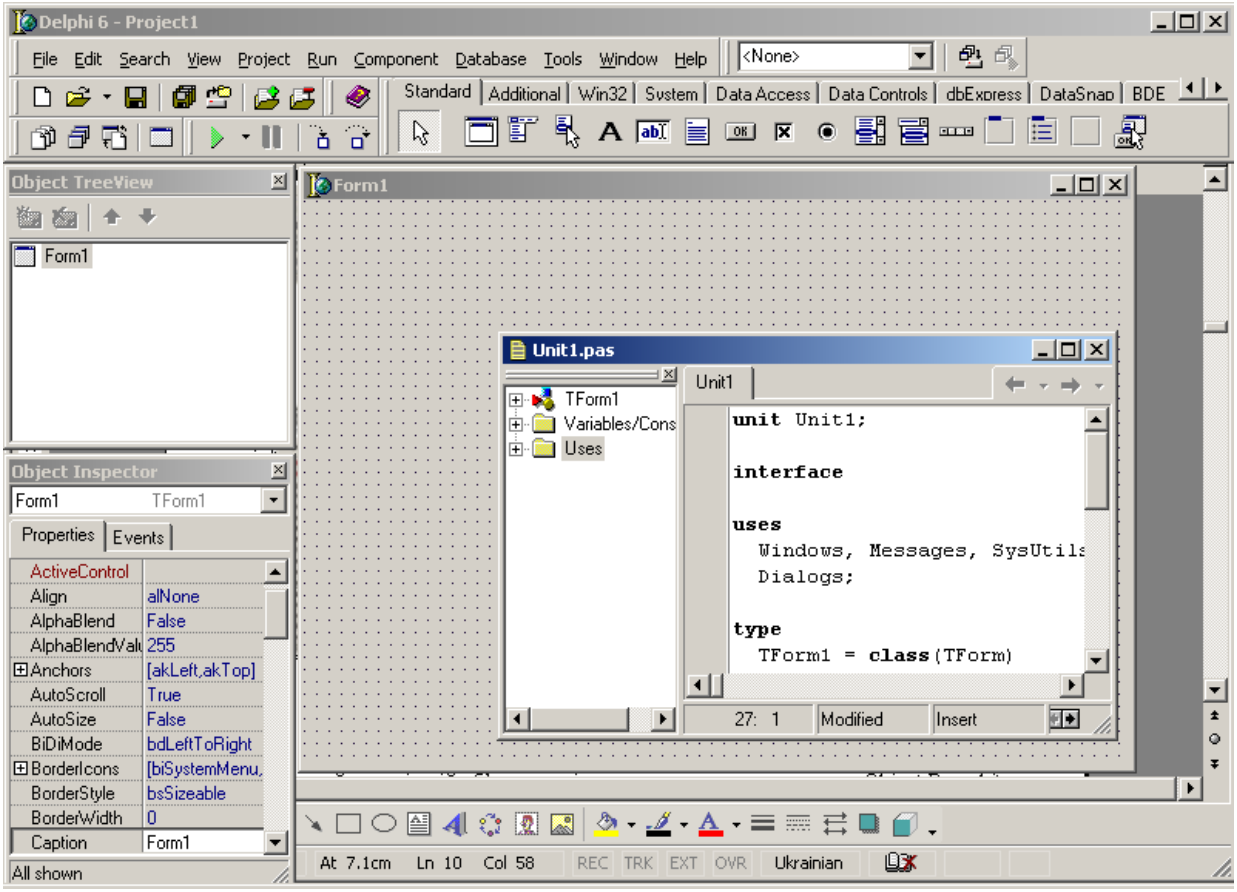


Рис. 6.8. Загальний вигляд візуального середовища програмування Delphi 6. На задньому плані форма, попереду – вікно з кодом. У лівому нижньому куту – список властивостей форми.

Найважливішим елементом усіх RAD, і Delphi у тому числі, є належним чином організоване зручне візуальне середовище, котре забезпечує ***двонаправлену розробку додатків з вже розроблених компонентів*** (тобто зміни, які зроблені у візуальному середовищі, відображаються на початковому (рос.-исходном) коді програми, а зміни початкового коду відображаються у візуальному середовищі). При цьому, частина стандартних фрагментів коду дає RAD, а потрібні з конкретної ситуації коди програміст дописує самостійно.

Головним елементом практично аби якого додатку є так звана ***форма*** (тобто вікно довільної (рос.-произвольной) форми), на котрій розміщуються усі інщі візуальні компоненти, а також меню, кнопки і відповідні команди, функціональність котрих забезпечує програміст (рис. 6.9).

Зважаючи на те, що взаємодія користувача з компонентами і компонентів поміж собою діються на рівні ***подій (рос.-событий)*** і ***відгуків (рос.-откликов)*** на них, у програмний код, окрім алгоритмічних складових, входять так звані ***обробники подій (рос.-обработчики событий)***. У фрагментах коду, що реалізує їхні функції, програміст вказує, як система повинна сприймати (рос.-воспринимать) й обробляти увід тих чи інших даних у тому чи іншому місці візуальних компонентів (вікон, кнопок і т.д.), як розцінювати клацання мишею

у конкретній точці ***форми*** і які дії при цьому необхідно розпочинати (рос.-предпринимать).

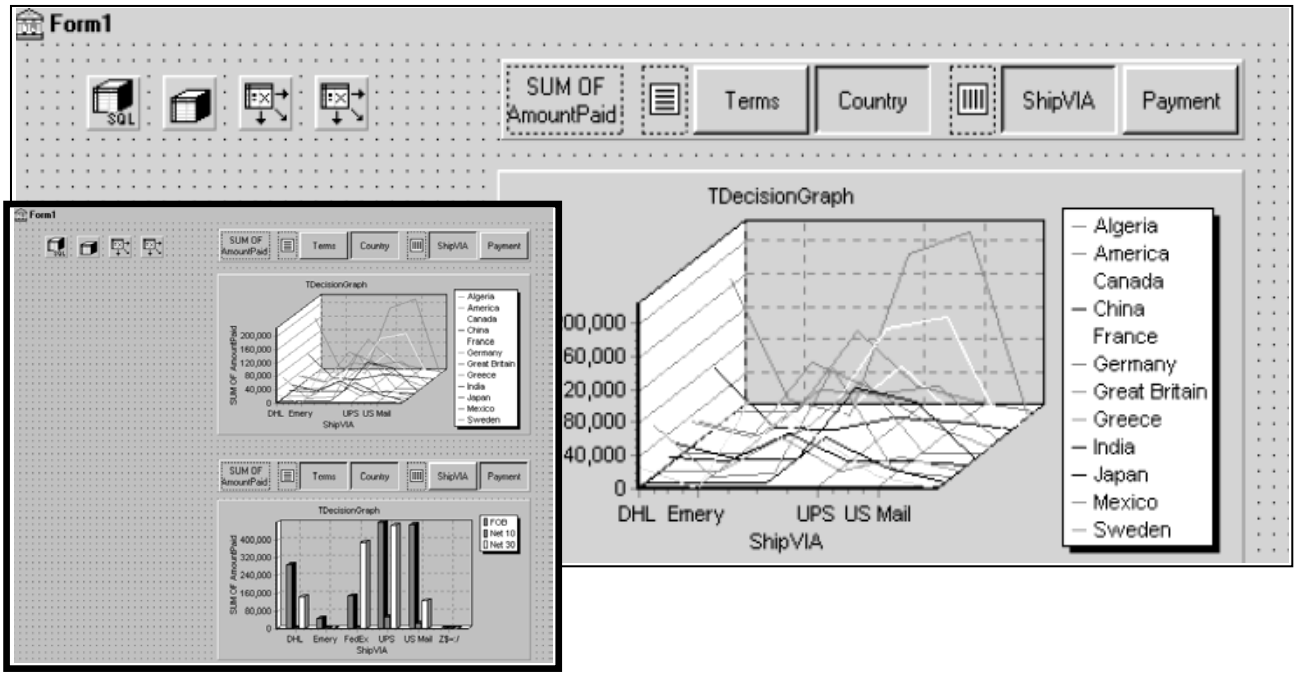


Рис. 6.9. Форма з компонентами, меню, кнопками і командами, які розроблені у IDE RAD Delphi для вирішення конкретної задачі (на передньому плані повна форма, на задньому – фрагмент)

Спрощення розробки додатків таким чином досягається уніфікацією взаємодії компонентів поміж собою і забезпечується на рівні уніфікованих інтерфейсів COM-компонентів (рис. 6.10).

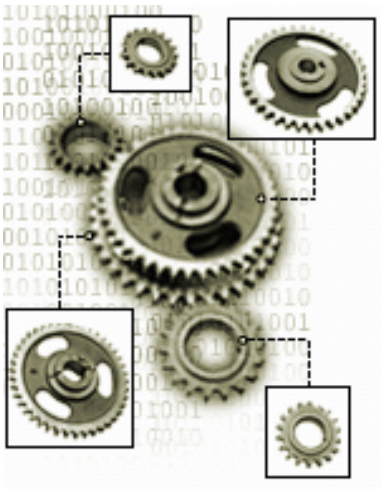


Рис. 6.10. Образне уявлення взаємодії компонентів (і відповідних модулів) на рівні інтерфейсів

Як правило, аби яка задача, що вирішується на комп'ютері, реалізується у Delphi у вигляді ***додатку***. Додаток створюється з різних частин. Кожна частина розміщується у окремому файлі і виконує строго визначені функції. Набір файлів, які необхідні для створення додатку, зветься ***проектом***. Тому, головною одиницею розробки додатку є ***проект***, який зберігається у файлі з розширенням DPR (скороч. від Delphi PRoject). Він уявляє собою головний програмний файл на мові Object Pascal, котрий підключає за допомогою операторів `uses` всі файли модулів, що входять до проекту. Кожній формі, що проектується у RAD, відповідає свій програмний модуль (`unit`), який містить усі відносні до форми ***об'яви*** і ***методи подій***, що пишуться програмістом на мові Object Pascal. Кожній формі відповідає свій двоічний файл з розширенням DFM, який містить її опис. Програмні модулі розміщуються в окремих файлах з розширенням PAS. Окрім них існує ще ряд додаткових файлів.

Файл ресурсів з розширенням RES (скороч. від RESource). У ньому, доречи, зберігається піктограма додатка, котра буде видна на Панелі Задач Windows. У підключених у вигляді модулів ***об'єктних файлах*** (мають розширення OBJ), розміщуються компоненти, написані на інших мовах програмування (наприклад, на C++). ***Файл опцій*** з розширенням DOF (скороч. від Delphi Option Files), містить параметри компіляції і компоновки проекту, які задає програміст. Існує ще цілий ряд файлів, які виконують службові функції, котрі у цьому посібнику не обговорюються.

У цілому, схематично процес створення додатка у RAD Delphi можна уявити як набір послідовних кроків по створенню проекту (див. рис. 6.11). Для цього на відкриту у вікні RAD Delphi форму наносяться потрібні візуальні компоненти і пишеться відповідний код. В подальшому, завантажений у пам'ять комп'ютера у вигляді двоїчного файла EXE або DLL додаток виконується як програмний модуль. Ідеологічно, таким же чином працюють багато інших середовищ візуального програмування для мов C++, Java, C# і деяких інших.

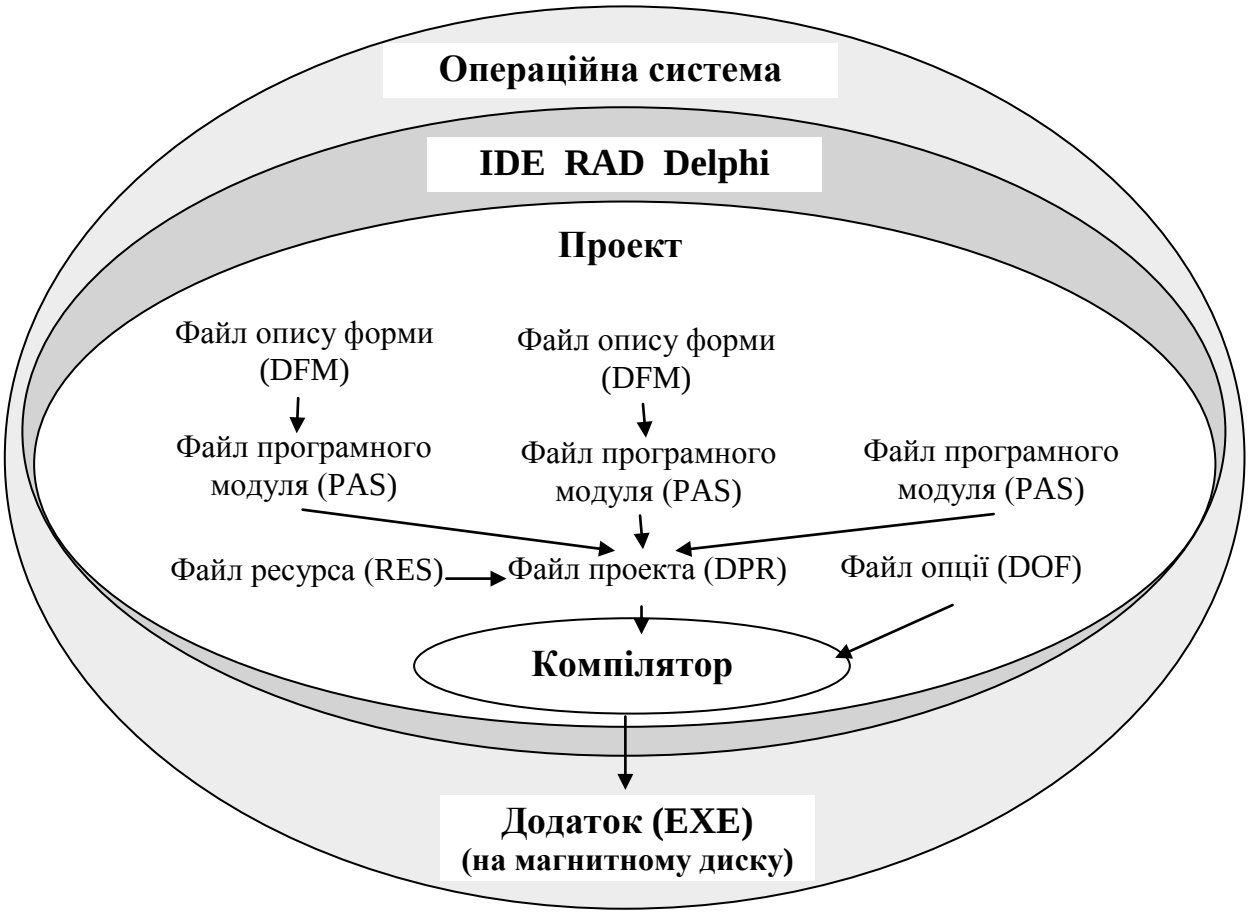


Рис. 6.11. Створення додатка в IDE RAD Delphi

Дещо відрізняється уявлення взаємодії програмних компонентів у IDE MS Visual Studio.Net, де найбільшою програмною одиницею є ***рішення (рос.-решение)*** *(solution).* У продовж роботи з IDE цей термін асоціюється з уявленням ***робочий простір (рос.-рабочее пространство)***, так як буквальний переклад — ***рішення*** — не завжди однозначний. Концепція ***рішення*** допомогає

об'єднати ***проекти*** та інші інформаційні елементи у одному ***робочому просторі***. Множина файлів різного типу, у рамках одного рішення складають ***додаток (рос.-приложение)*** *(application)* Visual Studio.Net 7.0. Робочій простір може містити декілька ***проектів***, бути пустим або містити файли, котрі мають сенс і поза контекстом ***рішення***. У аби якому випадку, користувач повинен починать роботу у студії з відкриття існуючого або створенняя нового ***робочого простіру***. ***Проект*** як частина ***рішення*** складається з окремих компонентів, наприклад файлів, які описують форму вікна або шаблон діалогу ***(re-файл)***, файлів з ***початковими (рос.-исходными) кодами*** програмних модулів (.cpp, .cs) і/або файлів, які уявляють собою опис ***запитів (рос.-запросов)*** до ***бази даних*** ***(database script)***.

Принципово відрізняються від додатків з візуальних RAD середовищ ті програми (підпрограми, фрагменти кодів і т.д.), які реалізовані на скриптових мовах і які уявляють собою звичайні набори текстових стрічок. До найбільш популярних скриптових мов, без сумніву, можна віднести мову Visual Basic for Applications (VBA), яка вбудована в усі без виключення офісні додатки Microsoft Office а, в останній час, і у геоінформаційну систему ESRI ArcGIS. При зовнішньої подібності інтерфейсу VBA (рис. 6.12) до звичайних IDE, коди процедур-підпрограм і процедур-функцій ***не компілюються у двоїчні файли з розширенням EXE і DLL***, а також ***не завантажуються для виконання у оперативну пам'ять комп'ютера***.

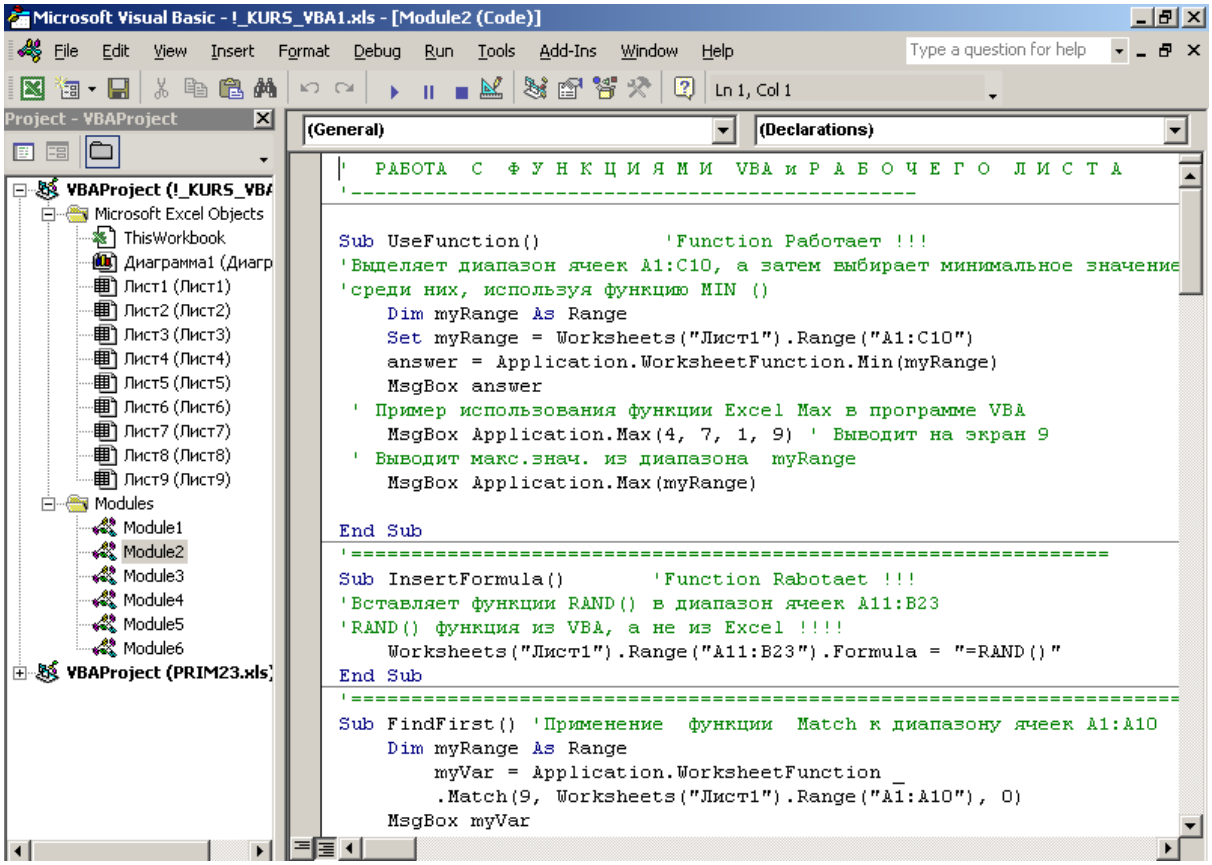


Рис. 6.12. Інтерфейс IDE Visual Basic for Applications

Виконання інструкцій скриптового коду, що уведені в тіло модуля, а потім викликаються у вікно проекту, провадиться шляхом їх інтерпретації і негайного виконання додатком-контейнером (наприклад, на рис. 6.12. MS Excel), котрий в цей час знаходиться у оперативній пам'яті комп'ютера.

По такому ж принципу працюють і інші додатки-контейнери і, у тому числі, MS Internet Explorer (IE). Знаходячись у оперативній пам'яті, IE приймає на вході стрічки з тегами (кодами) мови HTML і відображає закодовані таким чином елементи сторінки у відповідності з заданими параметрами розмітки, а якщо зустрічає фрагменти програм (кодів) на скриптових мовах, з котрих найбільш популярні JavaScript і VBScript, інтерпретує їх і безпосередньо виконує (рис.6.13).

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

Рис. 6.13. Приклад коду на мові JavaScript, що розміщено на HTML-сторінці

З деякими, доречи кажучи невеликими різницями, працюють і багато інших IDE і RAD для створення програмних кодів на різних мовах програмування. Не удаючись у подробиці тонкостей їх роботи, приведемо думку (рос.-мнение) одного з авторитетних спеціалістів з програмування на Visual Basic, Андрія Колесова, автора багаточисельних програмних розробок і статей у журналах "Комп'ютерПресс", PCWeek/RE і на сервері "Советы VBA Форуму". Його думка така[7].

> "Ще однією ілюзією є дуже поширене зараз уявлення про те, що кваліфікація програміста визначається тим, яким інструментом він володіє. Більш конкретно, С– програміст — це сильно, Basic — не дуже... Така думка не просто невірна: вона формує неправильне уявлення про дійсність. Насправді, кваліфікація розробника визначається його рівнем загально методичної підготовки і глибиною знань інструмента (мови). Короче кажучи, є більш і менш кваліфіковані спеціалісти. Тобто, відповідно, є слабкі С-програмісти і сильні VB-програмісти, і навпаки. При зміні інструменту, скоріш за все, слабкий розробник зостанеться слабким, а сильний — сильним.
>
> І все ж таки, не дивлячись на більш вузьку і глибоку спеціалізацію, "багатомовність" програміста і сьогодні є, якщо не обов'язковою, то, у крайньому випадку, досить необхідною потребою. Звичайно, зараз важко однаково добре знати два інструменти, тому, напевно, краще узяти за основу підхід "володіння головним і допоміжним інструментами". Очевидно, що "другим" повинен стати той засіб, котрий краще дозволяє прикрити слабкі місця "першого". При цьому навіть не дуже глибокі

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

знання додаткового інструмента може різко підвищити ефективність застосування головного. Плюсом від вивчення другої мови (як синоніма засобу розробки) є також більш повне розуміння можливостей першого. Для Fortran-користувачів (и, напевно, для C теж) є дуже корисними вміння створення інтерфейсу користувача за допомогою одного з RAD-інструментів. Delphi є у значній мірі самодостатнім інструментом, але навряд чи його користувач зможе обійтися без застосування VB або Java. VB-розробнику знання основ Java також зовсім не зашкодить (рос.-повредит). Усім їм також потрібне уявлення про HTML, а найближчим часом — і про XML".

Деякий час тому явною позитивною якістю (рос.-достоинством) програміста вважалося уміння використовувати технологію змішаного програмування (використання декількох мов для реалізації одного проекту). Сьогодні, в умовах більш чіткого розподілу праці, знання основ суміжних інструментів є таким же необхідним для афективної взаємодії з колегами.

І взагалі, у той час, як ми довго спорилися: "Програмування — це наука або мистецтво?, американці вже давно прийшли до висновку, що це — технологія".

6.4. У якому середовищі працюють програми і додатки?

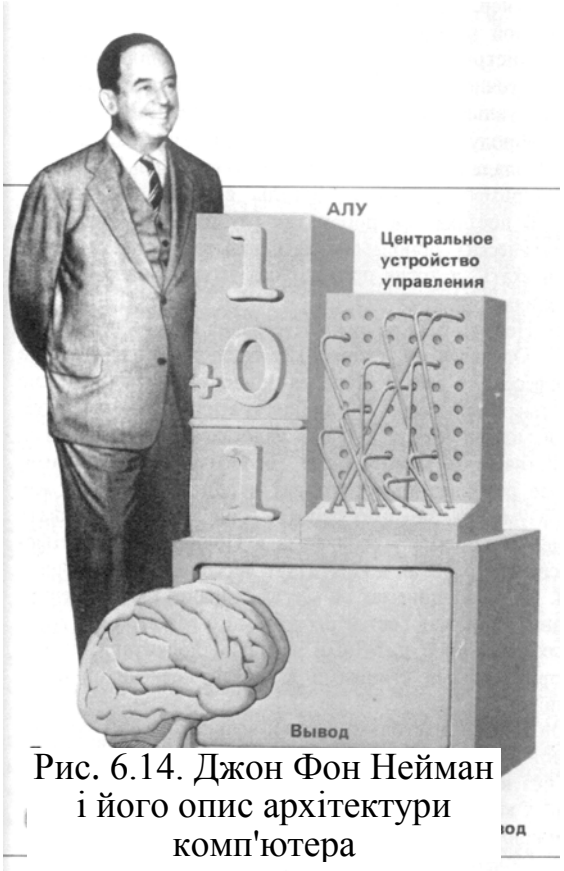


Рис. 6.14. Джон Фон Нейман і його опис архітектури комп'ютера

Розробка програм нерозривно пов’язана з ***архітектурою обчислювальних середовищ***, для котрих вони і створюються. Тут під архітектурою розуміється методологія об’єднання і взаємодії елементів складної обчислювальної структури на логічному (л.р.), фізичному (ф.р.) і програмному (п.р.) рівнях. Наприклад, для мережі LAN (див. Додаток 4), що складається з 5-ти комп'ютерів архітектура визначає, яка використовується схема поєднання ПК (л.р.), які компоненти апаратури з'єднані (ф.р.) і який набір програмних продуктів забезпечує роботу цієї системи: ОС, сервери і т.д. (п.р.). Тому розробнику слід чітко уявляти у яких "***вимірюваннях (рос.-измерениях)***" буде функціонувати кінцевий програмний коді що робити**:** управляти апаратурою, реалізувати математичний метод або клієнт-серверний Web-додаток, COM-сервер, сервлет, аплет і т.д..

Слід окремо відмітити, що з точки зору архітектурної побудови за Фон Нейманом власне сам комп'ютер складається з 5-ти наступних ***головних вузлів*** (рис.6.14):

❶ ***арифметико-логічного пристрою (АЛП);***

❷ ***центрального пристрою управління (ЦПУ);***

❸ ***оперативної пам'яті (ОП, ОЗП- оперативного запам'ятовуючого пристрою***);

❹ ***пристроїв уводу;***

❺ ***пристроїв виводу.***

У даному випадку, під архітектурою розуміється опис обчислювальної системи на деякому загальному рівні, який включає опис користувальницьких можливостей програмування, системи команд і засобів інтерфейсу користувача, організації пам'яті і системи адресації, операцій уводу/уводу, управління і т.д. І якщо прості, однокомп'ютерні (тобто ті, які працюють з одним мікропроцесором) додатки достатньо просто інтегруються у структуру обчислювального середовища, то вихід у мережу, де починають взаємодіяти багатопроцесорні і багатогомережові структури, як правило, істотно ускладнює задачу і приводить до необхідності використання моделі клієнт/сервер або компонентної моделі Web-сервісів.

Термін ***клієнт/сервер (або клієнт-сервер)*** уперше було застосовано у 80-х роках, по відношенню до процесу використання і взаємодії персональних комп'ютерів у мережах LAN (local area network). Сучасна клієнт/серверна модель виявила себе у кінці 80-х. Ця архітектура є гнучкою і заснованою на обміні повідомленнями (рос.-сообщениями) у модульній інфраструктурі, яка призначена для підвищення практичності (usability), гнучкості (flexibility), інтероперабельності (interoperability) і масштабованості (scalability) програмних додатків і єлементів комп'ютерних систем. ***Клієнт*** є ініціатором запитань (рос.-запросов) на виконання деяких послуг (services), а ***сервер*** – дає необхідні послуги (services) клієнту, які були потрібні і запитані клієнтом.

Отже, розглянемо послідовно найважливіші обчислювальні архітектури. Архітектурі клієнт/сервер передували дві архітектури, які існують і до наших часів у зв'язку з достатньо великою кількістю ***успадкованих систем (рос.-наследуемых систем),*** які продовжують використовуватися у повсякденних роботах у багатьох великих організаціях різних країн світу.

Архітектура мейнфреймів (*mainframe architecture*) (рис. 6.15). Характеризується зосередженням (рос.-сосредоточением) усіх вбудованих обчислювальних засобів на центральному хост-комп'ютері. Користувачі взаємодіють з хостом за допомогою уводу ключових командних стрічок через термінали. Хости не підтримують графічний інтерфейс користувача (GUI), але можуть взаємодіяти з ПК і робочими UNIX-станціями (рис.6.15). Останнім часом у зв'язку з постійним зростанням навантаження і об'єму інформації на серверах WWW, мейнфрейми почали активно використовуватися і у розподілених клієнт/серверних архітектурах. Самі ж сучасні мейнфрейми будуються на кластерах багаточисельних процесорів великої потужності.

У минулому, мейнфрейми звалися універсальними електронно-обчислювальними машинами (ЕОМ) і системи програмного забезпечення, що писалися до них були часто ***монолітні.*** У них інтерфейс користувача, ділова (бізнес) логіка і функціональні можливості доступу до даних зберігалися у одному великому додатку. Це було пов'язано з тим, що термінали увода-виводу, які використовувалися для звернень до універсальних ЕОМ не робили ніяких

операцій по обрабці даних – всі ці операції виконувалися у мейнфреймі безпосередньо. Таким чином створювалася ***монолітна архітектура додатків***.

Мейнфрейм та його компоненти:
– Операційна система
– Системи програмування
– Додатки
– Дані

Рис. 6.15. Архітектура мейнфреймів

Типова ***монолітна архіте-ктура додатка*** илюструється на рисунку 6.16. Подальший розвиток архітектури додатків був пов'язаний з розвитком персональних комп'ютерів, мережних технологій і технологій WWW та Internet (див. далі підрозділ 6.6).

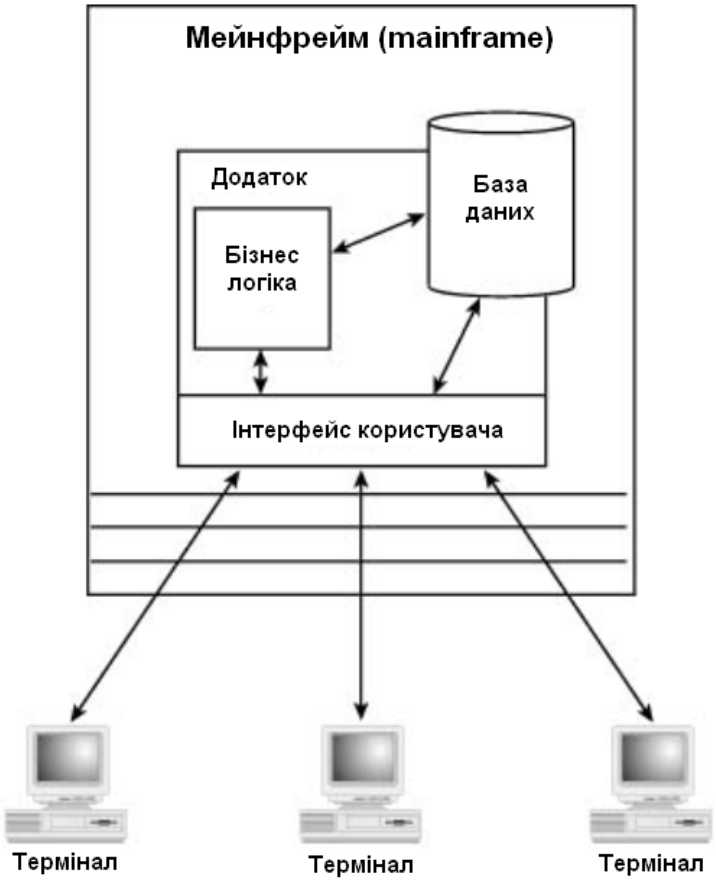


Рис. 6.16. Монолітна архітектура додатків мейнфреймів

Файл-орієнтована архітектура (*file sharing architecture*) пов'язана у першу чергу з звичайними комп'ютерними мережами LAN, у котрих комп'ютер-сервер завантажує файли з загального дискового простору у середовище, яке використовується настільними ПК. Далі додаток користувача запускається (разом з логікою і даними) у середовищі настільного ПК. Найбільш активно обчислення з використанням файл-орієнтованої архітектури застосовується у локальних мережах LAN, які розгортаються в учбових закладах і наукових

організаціях, а також у невеликих приватних фірмах.

Клієнт/серверна архітектура (*client/server architecture*) виникла як результат розвитку файл-орієнтованої архітектури, яка має деякі обмеження по продуктивності (рос.-производительности) і масштабованості. У нової архітектурі потужний комп'ютер, який використовується як сервер баз даних, замінив файл-сервер. Застосування систем управління базами даних СУБД (relational DataBase Management System, DBMS) дозволило суттєво прискорити і спростити зв'язки користувача з даними. Клієнт/серверна архітектура різко знизила мережовий трафік, а також забезпечила багатокористувальницький доступ до редагування одних і тих же записів даних у базах даних, які сумісно використовуються. Головним способом взаємодії клієнтів і серверів стали віддаленні (рос.-удалённые) виклики процедур (Remote Procedure Calls, RPCs) або речення (рос.-предложения) на стандартній мові запитань (рос.-запросов) (Standard Query Language, SQL) (рис.6.17).

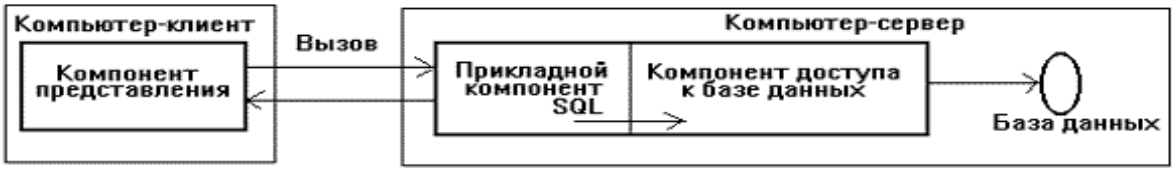


Рис. 6.17. Модель клієнт/серверної архітектури з сервером баз даних

Рис. 6.18. Класична дворівнева архітектура клієнт-сервер

У ***двоярусній архітектурі*** (*two tier architectures*) системний інтерфейс користувача як правило розташовано у середовищі настільного комп'ютера користувача, а служби управління базами даних – на більш потужнім комп'ютері сервері, котрий забезпечує зберігання додатків, а також процедур і тригерів баз даних (рис.6.18). На відміну від дуже коштовних мейнфреймів ця архітектура є добрим рішенням для розподілених обчислень у робочих групах до 100 користувачів, які працюють в LAN відділів і організацій одночасно.

Компоненти додатків у клієнт/серверних обчисленнях як правило розподілені так, щоб база даних постійно знаходилася на сервері (UNIX або мейнфреймі), інтерфейс користувача на ПК - клієнті, а ***ділова (бізнес) логіка*** знаходиться на обох компонентах.

Триярусна архітектура (*three tier architectures*), відома також як багатоярусна (multi-tier architecture), розроблялась для знімання обмежень, які існували у двоярусній. У ній середній ярус був доданий поміж системним інтерфейсом середовища клієнта і середовищем сервера, яка управляє базою даних. Існує достатньо багато реалізацій цього середнього рівня, до яких можна віднести монітори обробки транзакцій (transaction processing monitors), сервери повідомлень (message servers) і сервери додатків (application servers). Триярусна архітектура дозволяє одночасно працювати групам, які об'єднують декілька тисяч користувачів. Гнучкість у настроюванні потрібної конфігурації тут досягається простим застосуванням технології "drag and drop" для фрагментів кодів, які використовують додатки на різних комп'ютерах аби якого ярусу з трьох існуючих. Обмеження триярусної архітектури пов'язані з тим, що використання середовищ розробки додатків тут значно складніша, ніж візуально-орієнтована розробка у двоярусній архітектурі. Останнім часом, у триярусній архітектурі у якості серверів починають активно застосовуватися мейнфрейми.

Триярусна архітектура з технологіями монітора обробки транзакцій (*three tier architecture with transaction processing monitor technology*) є однією з найбільш розповсюджених і включає на середньому ярусі монітор обробки транзакцій (Transaction Processing, TP). У його задачі входить організація черг (рос.-очередей) запитань (рос.-запросов), розподіл транзакцій і пріоритетів виконання задач. Окрім того, він забезпечує:

❶ можливість оновлення багаточисельних різних СУБД шляхом одноразово виконаної транзакції;

❷ з'єднання і обробку багаточисельних джерел даних, і у тому числі двовимірні (рос.-двумерные) файли (flat files), не реляційні або об'єктно-орієнтовані БД, а також і мейнфрейми;

❸ можливість присвоювання різних пріоритетів транзакціям;

❹ забезпечення робастної (стійкої (рос.-устойчивой)) безпеки.

Триярусна архітектура з сервером повідомлень (рос.-сообщений) (*three tier with message server*) уявляє реалізацію процесу асинхронного обміну повідомленнями (messaging) з різними пріоритетами. Повідомлення, як правило, містять заголовки, які складаються з даних про рівень пріоритету, а також адреси і ідентифікаційного номера повідомлення. Якщо у даній архітектурі інтелектуальність, що програмується, зосереджена у повідомленнях, то у середовищі моніторів транзакцій безпосередньо у моніторі, який обробляє транзакції у вигляді не інтерпретованих пакетів даних. ***Триярусна архітектура з сервером повідомлень є кращім рішенням для безпровідних інфраструктур.***

Триярусна архітектура з сервером додатків (рос.-приложений) (*three tier with an application server*) забезпечує розміщення головного коду додатка для виконання не у середовищі клієнта, а на деякому обраному для використання хості, тобто комп'ютері вузла мережі. Сервер додатків не має і не керується GUI, а тільки виконує бізнес-логіку, обчислення і моделює машину пошуку даних. Перевагою такого підходу є зменшення компонентів програмного

забезпечення користувачів на комп'ютері клієнті і підвищення безпеки функціонування рішень у цілому (рис.6.19).

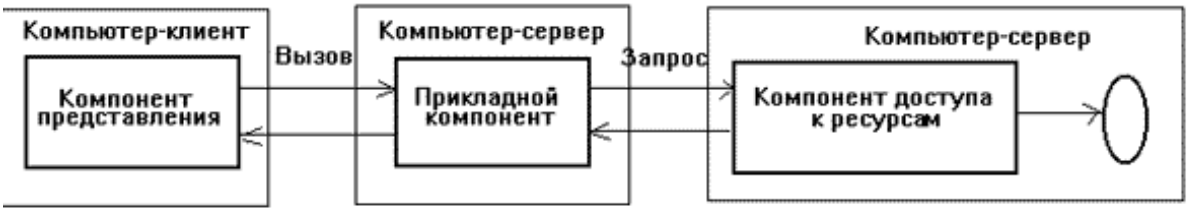


Рис. 6.19. Триярусна архітектура з сервером додатків

Додатки стають більш масштабованими (рос.-масштабируемыми), а вартість (рос.-стоимость) розгортання і підтримки програмних продуктів і систем на одному сервері значно менше, ніж на багаточисельних клієнтах. Таким чином, така архітектура найбільш усього вживається (рос.-применима) при умовах необхідності виконання умов безпеки, масштабованості і зниження вартості роботи системи.

Триярусна архітектура з ORB (*three tier with an ORB architecture*) для розподілених обчислень активно розвивається промисловими консорціумами на рівні стандартів, які підвищують інтероперабельність і кросплатформеність додатків, які розроблялися споконвічно (рос.-изначально) на різних мовах програмування для різних платформ. Головною проблемою для стандартів, що обговорюються, є визначення механізмів і елементів реалізації загальних для взаємодіючих систем об'єктних брокерів запитів (Object Request Broker, ORB). На даному етапі розробка клієнт/серверних систем з використанням технологій, що підтримують розподілені об'єкти має величезні перспективи в умовах повсюдного (рос.-повсеместного) розвитку і використання мережних, Інтернет і WWW технологій у суттєво гетерогенних середовищах. Крім підтримки інтероперабельності серед різних мов програмування і платформ, дана архітектура суттєво підвищує експлуатаційну надійність та зручність супроводження (maintainability), а також адаптованість (рос.-адаптируемость) (adaptability) програмного забезпечення, яке створюється . До найбільш розвинутих і тих, які частіше усього використовуються поточним часом відносяться дві наступні технології:

❶ Брокер запитів загальної об'єктної архітектури (Common Object Request Broker Architecture, CORBA).

❷ Компонентна об'єктна модель / Розподілена COM (COM/DCOM, Component Object Model/Distributed Component Object Model).

Промислові круги активно працюють над стандартами, котрі забезпечили би інтероперабельність поміж CORBA і COM/DCOM. Найбільш активним учасником даного процесу є консорціум Object Management Group (OMG), котрий розробив відображення відповідності (рос.-соответствия) даних і процесів поміж CORBA і COM/DCOM на рівні декількох програмних продуктів.

Розподілена/сумісна промислова архітектура (*distributed/collaborative enterprise architecture*) була розроблена у 1993 році. Ця програмна архітектура базується на технології застосування об'єктних брокерів запитів (Object Request Broker, ORB), які є продовженням розвитку архітектури CORBA, шляхом застосування сумісно (share) і повторно (reusable) уживаних (рос.-используемых) бізнес-моделей (а не просто об'єктів!) у масштабі розподілених підприємств (enterprise-wide scale). Переваги даного підходу містяться у тому, що стандартизовані бізнес-об'єктні моделі (business object models) і розподілені об'єктні обчислення (distributed object computing) комбінуються разом для забезпечення організаційної гнучкості для досягнення операційної і технологічної ефективності у масштабах цілих підприємств.

Глобальна XML архітектура Microsoft (*Microsoft Global XML Architecture, MS GXA*) є структурою і основою (framework) протоколу, який спроектовано для забезпечення моделі повної сумісності при побудові протоколів інфраструктурного рівня для Web-сервісів і Web-додатків. ***Web – сервіси*** – це прикладні сервіси, які уключають програми (додатки), програмування, дані і людські ресурси, котрі зроблені доступними з Web-серверів для Web-користувачів або інших Web-приєднаних програм. У доповнення до цієї головної структури протоколу, GXA визначає сімейство протоколів інфраструктури, які підключаються і котрі забезпечують для функціонуючих у GXA-середовищі додатків найбільш необхідними сервісами, такими, як безпека, надійність і підтримка багаторівневих і багатозв'язних служб погодження (рос.-согласования) їх роботи. GXA підтримує принципи, які об'єднують окремі протоколи GXA-інфраструктури у єдину зв'язану платформу для функціонування сервісів і додатків. Основною передумовою (рос.-предпосылкой) GXA є ліквідація необхідності для розробників додатків перебудовувати платформу цілком для кожного нового додатку. Ця передумова домінує в моделях розробки на однокомп'ютерних робочих місцях, де 95% програмістів усього світу використовують різні платформи розробки. Традиційний світ однокомп'ютерної розробки ПЗ десятиріччями формалізував методи модуляризації програмного коду і даних, котрі дозволяли здійснити (рос.-осуществить) розподіл поміж додатком і платформою. Світ комунікаційних протоколів також створив свої принципи модулярізації, серед яких найбільш відомим є 7-и рівневий протокол OSI.

Головною ціллю GXA є:

❶ створення більш загальної моделі для структурування протоколу;

❷ розробка сімейства протоколів платформного рівня для розробників додатків, які використовують Web-сервіси і Web-додатки.

Так же, як операційна система:

❶ забезпечує засоби ***загального використання (рос.-употребления) для роботи*** програмістам, програмним системам і додаткам;

❷ управління процесами і потоками;

❸ забезпечує контроль доступу до ресурсів та їх розподіл;

❹ управляє пам'яттю,

так і GXA забезпечує комплекс загальних засобів, які потрібні у широкому спектрі розробки і використання Web-сервісів і Web-додатків. Мова XML використовується тут для організації і використання даних. SOAP служить для передачі даних у мережах, WSDL застосовується для опису доступних даних, а UDDI використовується для перелічення (рос.-перечисления) доступних сервісів (рис.6.20.).

Сама мова XML служить технологією і засобом організації складно структурованих даних різної природи (програми, текстові, звукові, графічні, відео файли і т.д.), які розташовані у різних точках світового простору на різних ПК у єдине ціле (!!!).

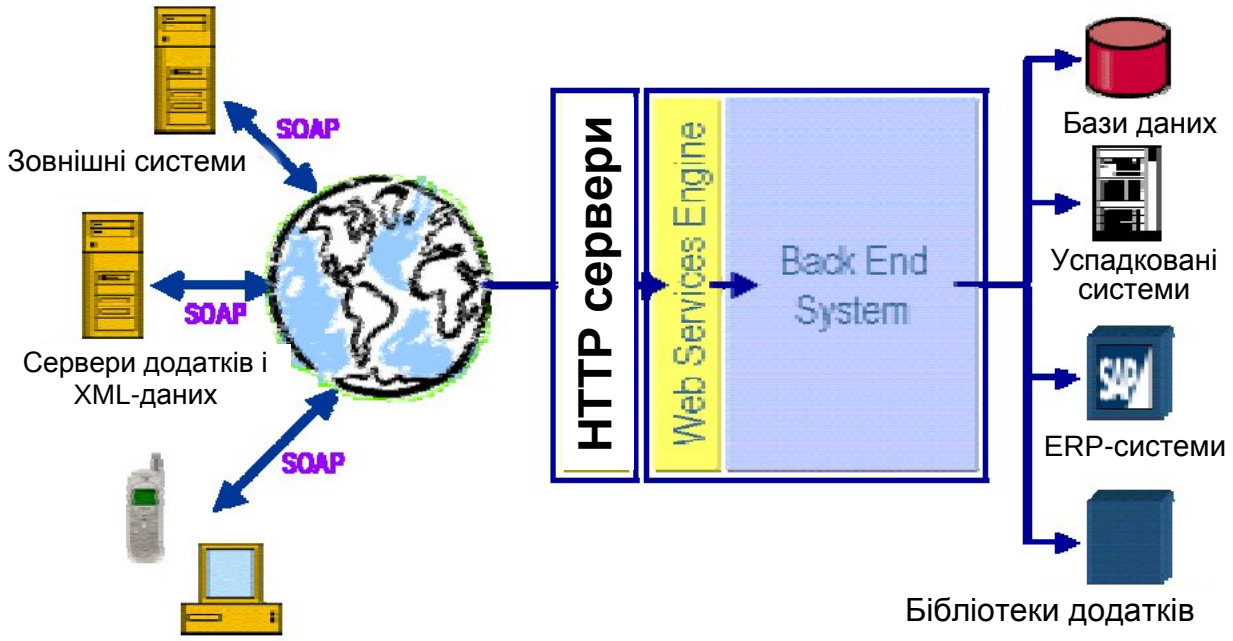


Рис. 6.20. Типова архітектура Web-сервісів

Підводячи підсумки слід додати, що вищевказані архітектури не єдині (рос.-единственные) і не останні. Зараз активно розвивається архітектура ***безпровідних комунікацій*** *(wireless architecture)* на базі ***адресних сервісів*** *(location services).* Ця і деякі інші архітектури розподілених обчислень підтримуються двома потужними архітектурами і відповідними технологіями: Jini™ від Sun Microsystems і .NET™ від Microsoft (див. підрозділ 6.6).

Тому розробнику програмного забезпечення необхідно постійно слідкувати за розвитком і взаємодією архітектур, а також компонентів і засобів комп'ютерної техніки і своєчасно корегувати методології розробки своїх додатків і рішень з відповідним засвоєнням нових: мов програмування, інструментальних засобів і технологій.

6.5 Як проектуються додатки і рішення?

Інформаційні системи грають основоположну роль у справі (рос.-деле) визначення (рос.-определения) успіху практично аби якого сучасного бізнесу і широкого спектру прикладних проблем з різних предметних областей. З появою персональних комп'ютерів комп'ютерні системи і додатки, створені на ї базі стійко закріпилися у більшості державних і бізнес-організацій світу.

Ці нові системи разом з існуючими успадкованими системами (legacy systems) визначають (рос.-предоставляют) критичні для функціонування організацій засоби зберігання, організації, отримання і обробки інформації у тому числі і на урядовому рівні[8],. Потенційно, ці системи можуть стати джерелом детальної різнопланової інформації, яка сприяє (рос.-способствует) прийняттю кращих рішень.

У своєї більшості сучасні інформаційні системи є розподіленими, тобто складеними з декількох частин, і як правило взаємодіють одна з одною через мережу. Розбивати систему на частини необхідно тому, що можливості автономної комп'ютерної системи не мають здібностей (рос.-не способны) покрити потреби серйозної інформаційної системи у обчислювальних можливостях і зберіганні даних. До того ж, розбивання на частини з унесенням (рос.-внесением) надмірності (рос.-избыточности) у ці частини забезпечує механізм відбудування (рос.-восстановления) системи після збоїв (рос.-сбоев).

У софтверній індустрії з урахуванням складності і великого об'єму виконуваних при реалізації ***крупних додатків*** додаткових заходів часто останні йменують ***рішеннями***[9]. Свій великий досвід у розробці бізнес–рішень спеціалісти корпорації Microsoft узагальнили у вигляді комплекса документів з опису Дисципліни розробки рішень (Microsoft Solution Framework, MSF[10]). MSF містить в собі набір моделей і чітко визначених проектних віх, котрі можна розглядати як рекомендації, рівно як я керівництво по плануванню, веденню і управлінню проектами у сфері інформаційних технологій. Ці моделі — результат інтеграції у єдину систему найбільш успішних і багаторазово застосованих практик, виявлених у процесі аналізу досвіду розробки програмних продуктів, накопиченого не тільки Microsoft, але й її замовниками і партнерами.

Вони не є єдиними (рос.-единственніми) у своєму роді, але на відміну від багатьох інших можуть бути отримані безкоштовно на російській мові.

З точки зору MSF, розробка рішення бізнес–проблеми — це більше, ніж просте написання суперскладного додатку з застосуванням найновітніших технологій. Розробка додатка повинна вестися, виходячи з попередньо

[8] B2G, G2C, G2G. Ці абревіатури обозначають нові сфери бізнесу, в котрі, тим чи іншим чином, залучається (рос.-вовлечена) держава (Government) і розкриваються як Business-to-Government, Government-to-Citizens, Government-to-Government. Є наслідком включення Держави у процес електронізації усіх видів діяльності. Концепція *Electronic Government* була оголошена у США на самому високому урядовому рівні першого липня 1997 року.

[9] Брандт Д. Architectures. Экзамен – экстерном. (экзамен 70-100). СПБ.: Питер, 2001. – 432 с.

[10] Матеріали Microsoft MSF: (http://www.microsoft.com/msf и http://www.microsoft.com/rus/msf)

складених бізнес–вимог. Коли ж його розроблено, додаток повинен бути прийнятий кінцевими користувачами, котрим він зобов'язаний допомагати в рішенні повсякденних проблем.

Тобто, ***додаток обов'язково пишеться під конкретний процес і відповідає його бізнес-правилам***.

Після того, як будуть підготовані і уточнені усі вимоги і проявляться контури майбутнього ***рішення***, ці високо рівневі абстракції зможуть надати інформацію, необхідну для створення детальних проектів, котрі ляжуть в основу реалізації. Такі кроки, показані на рис. 6.21.

Рис. 6.21**.** Процес розробки прикладної інформаційної системи

6.5.1.Підготовчий етап: аналіз вимог. Розробка рішення починається з аналізу вимог. Цей високорівневий процес збору інформації повинен відповісти на питання: що потрібно зробити, чому це потрібно зробити і який очікується результат. Після чого можна підвести підсумки і зафіксувати вимоги у документації. Повний ***аналіз вимог*** дає інформацію, яка описує:

❶ границі проекту;
❷ суть проблеми (потреби користувачів);
❸ потрібний рівень безпеки;
❹ продуктивність (рос.-производительность) додатку;
❺ потреби у супроводженні;
❻ можливості розширяємості;
❼ доступність;
❽ масштабованість;
❾ людський фактор;
❿ інтеграція з існуючим оточенням;
❶❶ методології.

6.5.2.Визначення технічної архітектури. Після вияснення вимог потрібно приступати до оцінки можливостей і обмежень технологій, які дозволяють реалізувати рішення, і починати формулювати структуру технічної архітектури, яка підходить для цього.

Тут слід визначитися також з кількістю користувачів, які обслуговує додаток і відповідно об'ємами і розташуванням джерел даних, що будуть використовуватися у майбутньому. Важливо оцінити існуючі стандарти і технології для працюючих у майбутньому рішень. І, на кінець, технічна архітектура повинна у загальних рисах давати уявлення о стратегії розгортання рішення після його реалізації.

6.5.3.Розробка моделі даних. Технічна архітектура дає уявлення о вимогах до даних, однак вона не дає ніяких вказівок про те, якими повинні бути елементи даних і як їх слід організовувати. Це визначається моделлю даних,

котра, в свою чергу, визначається ***метаданими*** (metadata), тобто ***даними о даних***. Модель даних визначає усі елементи даних, тобто ***сутності*** (entities), а також характеристики — ***атрибути*** (attributes) — кожної з них і організацію цих сутностей. Для організації даних існують спеціальні ***правила нормалізації даних*** (data normalization rules). Окрім цього модель даних визначає ***зв'язки*** (relationships) поміж сутностями, а також спеціальні правила, ***обмеження*** (constraints), які використовуються для забезпечення ***цілісності*** (integrity) даних.

6.5.4.Разробка проектних частин додатків і рішень. Наступна задача, після того як були розроблені технічна архітектура і модель даних, це розробка проектних частин додатків ***рішення,*** тому що власне розробка додатків має на увазі три різних видів діяльності:

❶ розробку ***концептуального*** і ***логічного проектів додатків***;

❷ проектування ***інтерфейсу*** і ***служб користувача***;

❸ розробку ***фізичного проекту***.

Більшість людей вважає, що основна діяльність при розробці додатку полягає у написанні і відлагодженню (рос.-отладке) коду (рис. 6.22). Хоча процес написання і відлагодження коду і уявляє собою невід'ємну частину процесу розробки, ця діяльність не може початися до тих пір, поки не будуть готові концептуальний, логічний і фізичний проекти додатків.

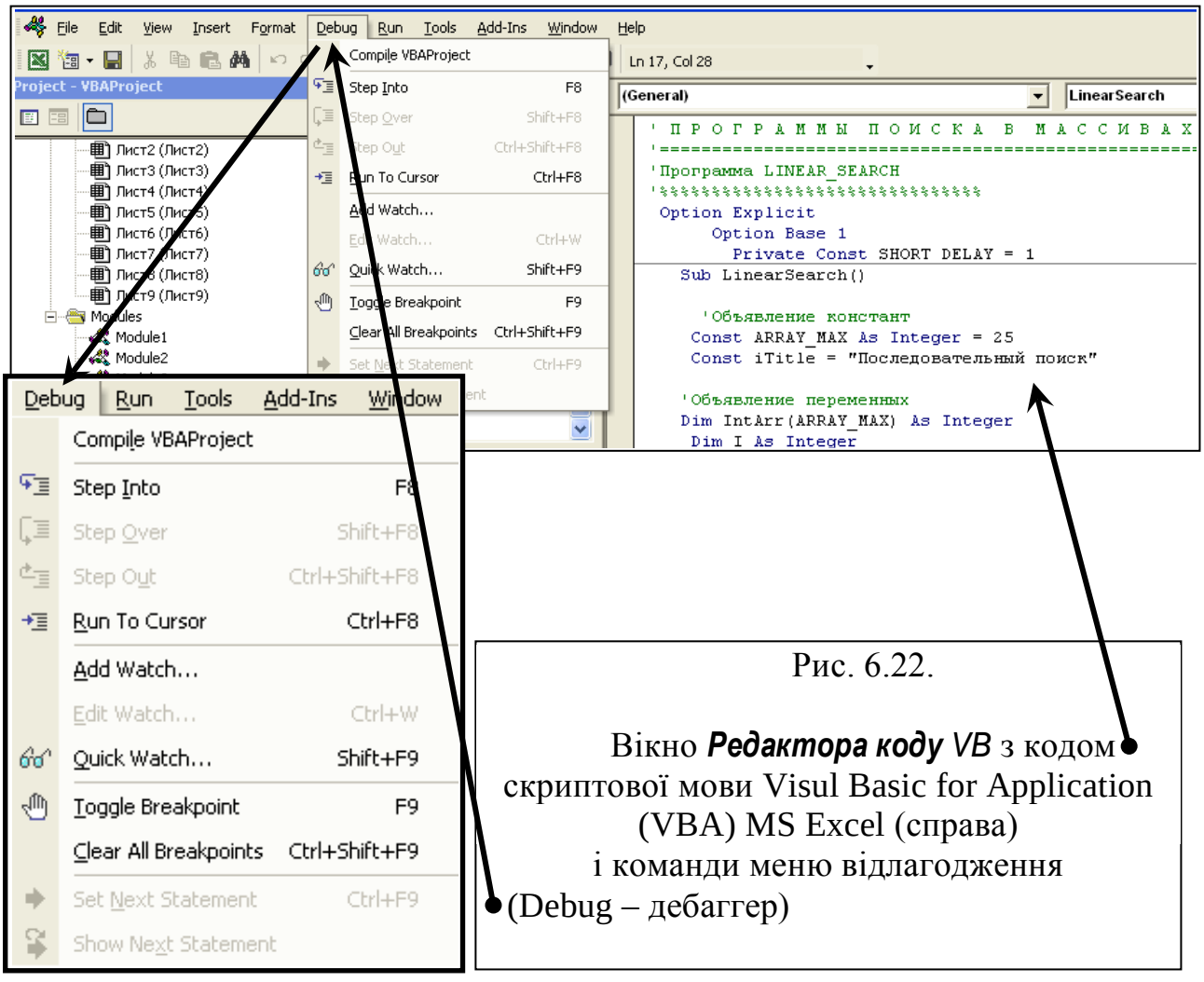


Рис. 6.22.

Вікно ***Редактора коду*** *VB* з кодом скриптової мови Visul Basic for Application (VBA) MS Excel (справа) і команди меню відлагодження (Debug – дебаггер)

Отже, проектування додатків починається з розробки концептуального проекту, котрий базується на бізнес-вимогах і як правило містить ***сценарії***, ***моделі послідовності дій/процес*** *(workflow/process model)* **і** ***моделі задача/послідовність (task/*** *sequence model).*

Логічний проект, який витікає далі з концептуального проекту, дає абстрактний високорівневий опис додатку. Логічна модель розбиває додаток на функціональні модулі, які відповідають бізнес-компонентам додатку. Центральною задачею логічного проектування є вичленення і специфікація бізнес-об'єктів, їх властивостей і методів. У термінах MSF моделі додатка ці бізнес-об'єкти звуться ***службами.*** Логічний проект визначає ***служби додатку*** *(application services)*, що дає додаток. Використання моделей, отриманих у результаті концептуального і логічного проектування, дозволяє дивитися на додаток у термінах ***бізнес-компонентів*** і ***програмної архітектури***. Як правило, оцінка логічного проекту робиться по семі ключовим факторам. Більш точні критерії оцінки визначаються конкретним рішенням, однак для логічного проекту найбільш складними є обмеження на:

❶ ***продуктивність (рос.-производительность) (performance),***
❷ ***супроводжуваність (рос.-сопровождаемостъ) (maintainability),***
❸ ***розширяємість (рос.-расширяемость) (extensibility),***
❹ ***доступність (availability),***
❺ ***масштабуємість (рос.-масштабируемость) (scalability)***
❻ ***безпека (рос.-безопасность) (security),***
❼ ***відмовостійкість (рос.-отказоустойчивость)*** *(failure-resistance).*

До інших, що також впливають на логічний проект вимогам, відносяться:

❶ ***практичність*** *(usability)*
❷ ***гнучкість (рос.-гибкость)*** *(flexibility)*
❸ ***інтероперабельність*** *(interoperability)*
❹ ***кросплатформеність*** *(crossplatform)*
❺ ***переносимість (мобільність, портабельність) (mobility)***
❻ ***адаптованість (рос.-адаптируемость)*** *(adaptability)*

6.5.5.Проектування інтерфейсу і служб користувача. ***Інтерфейс користувача*** *(UI)* уявляє служби для організації повнофункціональної роботи з додатком. Це означає, що він дає засоби (рос.-средства) для виконання наступних операцій:

❶ переміщення компонентів;	❹ виводу даних;
❷ уводу даних;	❺ маніпулювання з даними;
❸ уявлення даних;	❻ допомоги користувачу (help).

У залежності від фізичної реалізації, інтерфейс користувача може бути частиною додатку, а може бути окремим клієнтським додатком. Крім того, він може бути й набором кодів, що містять сценарії Web-сторінок, які потім відображаються у браузере. Під «користувачем» додатка як правило розуміється людина, яка користується додатком. Тому під інтерфейсом користувача розуміються форми, елементи управління (меню, кнопки, вікна і т.д.), звіти (рос.-отчеты) та інші об'єкти, що даються кінцевому користувачу для взаємодії з додатком. Однак «користувач» додатку не обов'язково означає

людину: будь який додаток також можна розглядати, як користувача. В цьому випадку під інтерфейсом користувача розуміється засіб взаємодії поміж програмами, який забезпечує сумісну роботу додатків.

6.5.6.Розробка фізичного проекту. Після того, як була спроектована модель даних, створені концептуальний і логічний проекти, спроектовано інтерфейс користувача і служби додатка, треба виконати роботи, які відносяться до фізичного проекту рішення. Це потребує виконання наступних дій:

❶ оцінку фізичного проекту;

❷ проектування компонентів;

❸ розробку стратегії доступу до бази даних.

Фізичний проект визначає реалізацію додатку і використання для цього потрібних технологій. Як правило, є декілька шляхів реалізації, з котрих треба вибрати найбільш придатний (рос.-подходящий). Для оцінки фізичного проекту використовуються ті ж самі шість ключових факторів, що і для оцінки логічного проекту. Як і для логічного проекту, точні критерії оцінки фізичного проекту визначаються конкретним рішенням, але, як правило, для фізичного проекту найбільш складними обмеженнями є вимоги до досягнення оптимальних характеристик показників: ***(рос.-производительности, сопровождаемости, расширяемости, доступности, масштабируемости и безопасности).*** Спеціальної уваги потребують також і такі галузі, як ***проектування компонентів (component design) і доступ до баз даних (database access).*** Проектні рішення у цих галузях можуть сильно вплинути (рос.-повлиять) на загальне функціонування додатку.

6.5.7.Розгортання і супроводження рішення. Після того як додатки були реалізовані і відтестовані, починається розгортання системи. Коли система розгорнута, її потрібно супроводжувати. І розгортання, і супроводження є достатньо важливими елементами загального процесу створення додатка для його успішної подальшої роботи.

6.6.Які існують додатки?

Як правило, розробка додатка складається з проектування, моделювання, створення прототипу і у кінцевому підсумку його реалізації і тестування. На фазах проектування і моделювання розробляється ***архітектура додатку, яка частково залежить і від архітектури обчислювального середовища***. Як правило, більша частина додатків містить код уявлення (рос.-представления), код обробки даних і код звернення до сховищ (рос.-хранилищ) даних. Архітектура додатка визначає те, як буде організовано цей код. Аналіз і проектування є невід'ємною частиною процесу розробки додатка. Розробляючи архітектуру додатка, потрібно дбайливо (рос.-тщательно) обмірковувати наслідки, що витікають з проектних рішень. Перед тим як почати загальне проектування, має рацію визначитися з типом майбутнього додатку. Для опису характеристик або типу додатка використовується цілий ряд термінів. Доречи, додаток, може характеризуватися такими назвами (табл.. 6.2):

Таблиця 6.2.

Тип додатків у залежності від архітектури, у якій він виконується

Назви додатків	
❶ настільний (рос.-настольный); ❷ контейнер; ❸ розподілений (рос.-распределённый); ❹ клієнт-серверний; ❺ SDI; ❻ MDI; ❼ консольний; ❽ діалоговий (майстер, Wizard);	❾ одноярусний; ❶❼ двоярусний; ❶❶ багатоярусний (N-рівневий); ❶❷ Web-додаток; ❶❸ сумісно працюючі додатки; ❶❹ компонент; ❶❺ мережовий сервіс або Web-сервіс.

По змісту ці терміни перекривають сфери дії один одного, що зовсім не дивно. Як правило, реальні додатки уявляють собою комбінацію декількох вищевказаних типів. Охарактеризуємо коротко кожного з них.

Настільні (desktop) додатки — це додатки, котрі виконуються на одному комп'ютері і використовуються одним користувачем. Настільні додатки розміщуються на жорсткому диску комп'ютера користувача і працюють тільки з локальними ресурсами. Настільні додатки можуть бути основані на двоярусній архітектурі, яка припускає (рос.-подразумевает) звертання (рос.-обращение) до серверу за даними.

Контейнер (container). У розробленої Sun Microsystems компонентній архітектурі *JavaBeans* і у компонентній технології Microsoft *Component Object Model (COM),* ***контейнер*** є прикладною програмою або підсистемою, у якій виконується побудований або вбудований блок програми, який зветься компонентом (*component)*. Наприклад, компонент типу ***кнопка*** або інший елемент графічного інтерфейсу користувача, чи маленький калькулятор – всі вони мають виконуватися з використанням компонентної моделі JavaBeans, котра дозволяє виконати їх у контейнері Netscape, який є браузером або у контейнерах Microsoft, таких як MS Internet Explorer, Visual Basic, Excel або Word. Для трьох останніх наповненням контейнеру є скриптовий код мови Visual Basic for Application, котрий вони виконують (інтерпретують) самі знаходячись у ***оперативному запам'ятовуючому пристрої.***

Розподілені (distributed) додатки — це складені (рос.-составные) додатки, різні частини яких виконуються відокремленню (рос.-по отдельности), але роблять загальну справу. Розподілені додатки застосовують для рішення проблеми або виконання задачі декількома комп'ютерними системами.

Клієнт-серверні (client/server) додатки — це розподілені додатки, основані на моделі обчислень, у котрій клієнт запитує послуги у другої сутності — сервера. У типовому для бізнес-систем клієнт-серверному додатку клієнт виконується на персональному комп'ютері, а розташований на віддаленій машині сервер надає йому послуги по доступу до даних, які зберігаються на сервері. Клієнтська частина додатку як правило оптимізується для взаємодії з

користувачем, у той час як серверна частина надає функціональність, яка сумісно використовується багатьма користувачами.

В клієнт-серверних системах обробка даних діється і на клієнте, і на сервері. Цим вони відрізняються від систем, дані яких зберігаються на файл-сервері: у таких системах ніяких обчислень на файл-сервері не провадяться і усі компоненти додатку виконуються на клієнті. Більшість розподілених додатків мають клієнт-серверну архітектуру. Ця модель, у котрій споживач (рос.-потребитель) послуг (клієнт) запитує їх у постачальника (сервера). Клієнт-серверна архітектура звичайно реалізується у вигляді декількох додатків, які виконуються на різних системах, але це не є обов'язковим. Ці додатки можуть виконуватися і на однієї комп'ютерної системі. Справа тут ні у реалізації, а у взаємодії, при котрої один запитує послуги, які надаються іншим — саме воно визначає суть клієнт-серверної архітектури.

***SDI (однодокументний інтерфейс** – Single Document Interface)*. У Windows-додатках зустрічаються два основних стиля інтерфейсу користувача. Деякі з додатків, наприклад Блокнот (Notepad), дозволяють працювати тільки з одним документом. Щоб відкрити інший документ, потрібно закрити поточний (рос.-текущий). Додатки, подібні до Блокноту, які використовують одне головне і декілька додаткових вторинних вікон, звуться ***SDI-додатками** (Single Document Interface — **однодокументний інтерфейс**)*. Єдиний можливий спосіб працювати одночасно з декількома об'єктами у SDI-додатку — це відкрити декілька екземплярів цього додатка. Головні вікна SDI-додатка можна згортати (рос.-сворачивать) і розгортати незалежно один від одного. Якщо робиться спроба відкрити вже відкритий об'єкт, активізується існуюче вікно. У Windows 95 частіше усього зустрічаються саме SDI-додатки (рис 6.23.), оскільки в операційній системі зроблено акцент на поняття документу.

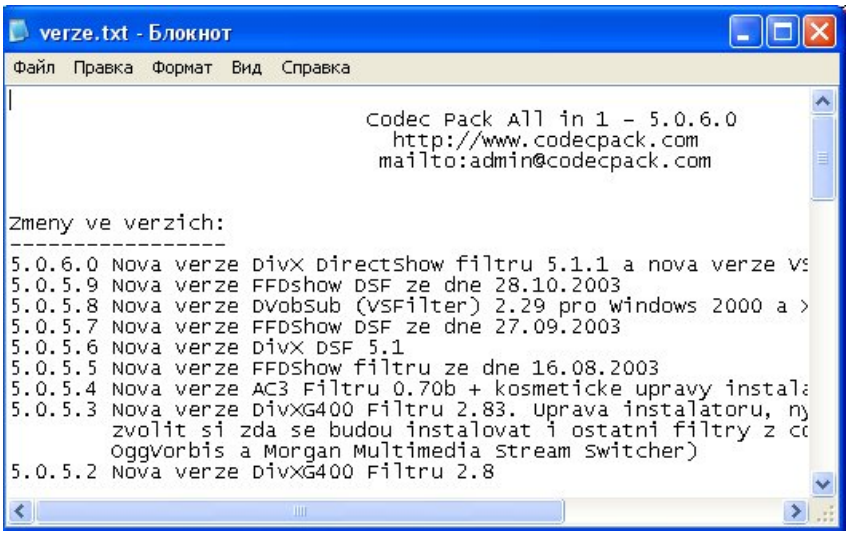


Рис. 6.23 . Додаток Блокнот з SDI–інтерфейсом

***MDI (багатодокументний інтерфейс** – Multiple Document Interface)*. Інші ж додатки, наприклад Microsoft Word у Windows 95, дозволяють працювати одночасно з декількома документами. Кожний документ відображається у

окремому вікні, переключатися поміж котрими дозволяє пункт меню ***Окно*** (***Window***) (рис. 6.24).

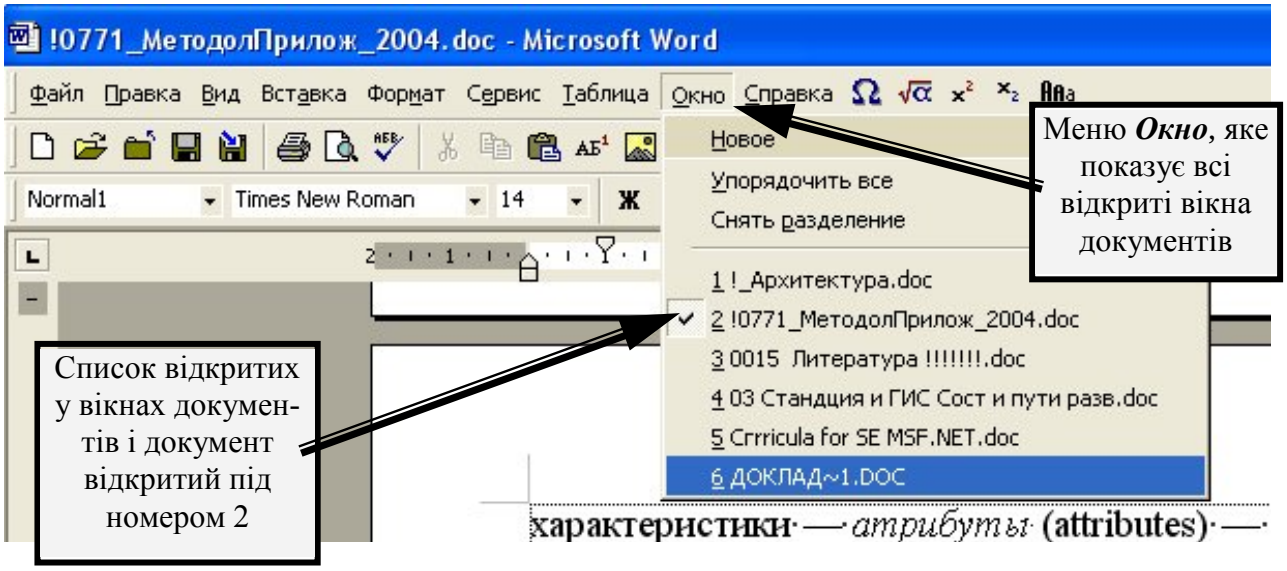


Рис. 6.24. MDI–інтерфейс у додатку Word

Кожне дочірнє вікно по своєї суті є головним, але воно завжди міститься всередині батьківського (рос.-родительского) вікна. Дочірні вікна MDI-додатків згортаються у межах батьківського вікна, і, вже згорнені, містять значок документа. Прикладами MDI-додатків служать також Microsoft Access і Microsoft Excel. Microsoft Excel, наприклад, дозволяє створювати і відібражати дочерні вікна декількох різних типів, таких як діаграми і електронні таблиці. При активізації таких вікон змінюється стрічка меню додатка.

Усі індивідуальні вікна заключені у межі батьківського вікна Excel. При згортанні Microsoft Excel згортаються усі дочерні вікна, а на панелі задач відображається тільки значок батьківського (рос.-родительского) вікна (рис. 6.25).

Рис. 6.25. Відображення значка батьківського (рос.-родительского) вікна додатку MS Excel на панелі задач (Task Bar) ОС Windows

Консольний додаток. У консольних додатків нема графічного інтерфейсу. Замість нього взаємодія користувача з консольним додатком провадиться за допомогою ***командного інтерфейсу (інтерфейс командної стрічки).***

Як правило у консольного додатка є набір команд, котрі можна використовувати для доступу до функціональності додатка. Однак запам'ятати, які команди за що відповідають і який у них синтаксис, бува часом важко. Більшість додатків для мейнфреймів і успадкованих додатків відносяться до категорії ***консольних***.

Ці додатки призначені для використання з алфавітно-цифровим терміналом. В середовищі Windows термінал часом заміняється програмою

емуляції вікна MS DOS – *Command Prompt* (***Командная строка***) (рис. 6.26).

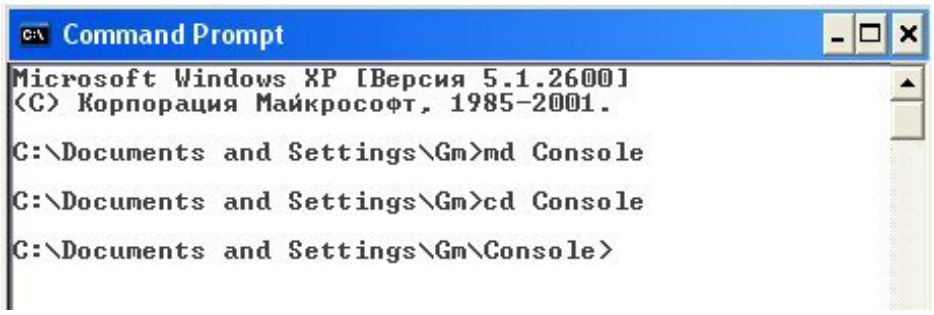


Рис. 6.26. Інтерфейс консольного Windows–додатка Command Prompt

Треба додати, що у цьому вікні функціонують і виконуються практично всі команди операційної системи MS DOS.

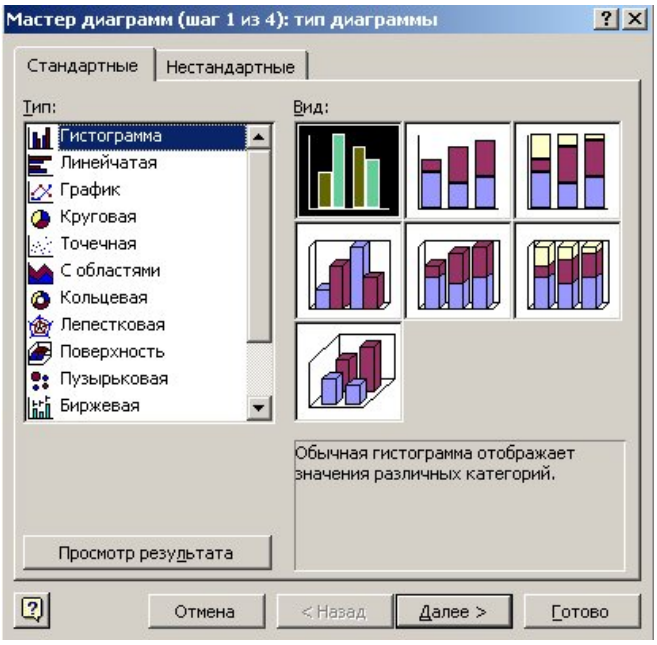


Рис. 6.27.
Майстер діаграм додатка MS Excel

Діалоговий додаток (майстер, Wizard) веде користувач через послідовність кроків до виконання визначеної (рос.-определенной) задачі. Діалогові додатки як правило активно взаємодіють з користувачем через набір екранів (або діалогових вікон), через котрі користувач робить свій вибір. Добрим прикладом діалогових додатків є досить поширені в середовищі MS Office і Windows ***майстри*** (Wizards) (рис. 6.27).

Одноярусним додатком (single-tier application) зветься додаток, який реалізований у одному фізичному ярусі. Це значить, що такий додаток виконується як одиничний (рос.-единичный) процес на однієї машині. Одноярусний додаток може бути заснован на різних логічних моделях. Це може бути багатоярусна логічна модель, у котрій служби різних типів відділені одна від одної. Або ж це може бути модель монолітного додатка, у котрому нема чіткого розподілу служб за типами. Для взаємодії поміж різними модулями одноярусного додатку використовується стандартний виклик процедур. Таким чином, у цих додатках немає надмірності (рос.-избыточности) взаємодії, яка притаманна (рос.-присуща) розподіленим додаткам. Однак продуктивність такого додатку обмежена можливостями апаратних засобів, на котрих він виконується. Одна з переваг одноярусних додатків міститься у легкості реалізації. З цієї причини одноярусні додатки достатньо широко розповсюджені.

Двоярусні додатки (two-tier application) є добрим прикладом розподіленого додатка, який використовує клієнт-серверну модель. Двоярусний додаток — це додаток, який реалізован у двох фізичних слоях. Більшість додатків, які потребують звернень до баз даних, реалізуються як двоярусні додатки. У двоярусній моделі додаток, який виконується на робочої станції (клієнт), звертається за даними до централізованої бази даних на віддаленній системі (сервері), який підтримує множину (рос.-множество) розподілених клієнтів. Така модель дозволяє сумісно використовувати обчислювальні ресурси і дає доступ до централізованої бази даних, однак супроводження і підтримка клієнтських додатків у такої моделі все ж таки складніше, ніж управління і підтримка централізованих систем минулих років. Однак реалізація клієнт-серверної моделі у реалізації Microsoft COM додає функціональності та інтероперабільності не тільки у взаємодії поміж додатками різних виробників, а також і поміж додатками, що працюють на різних платформах (рис. 6.28).

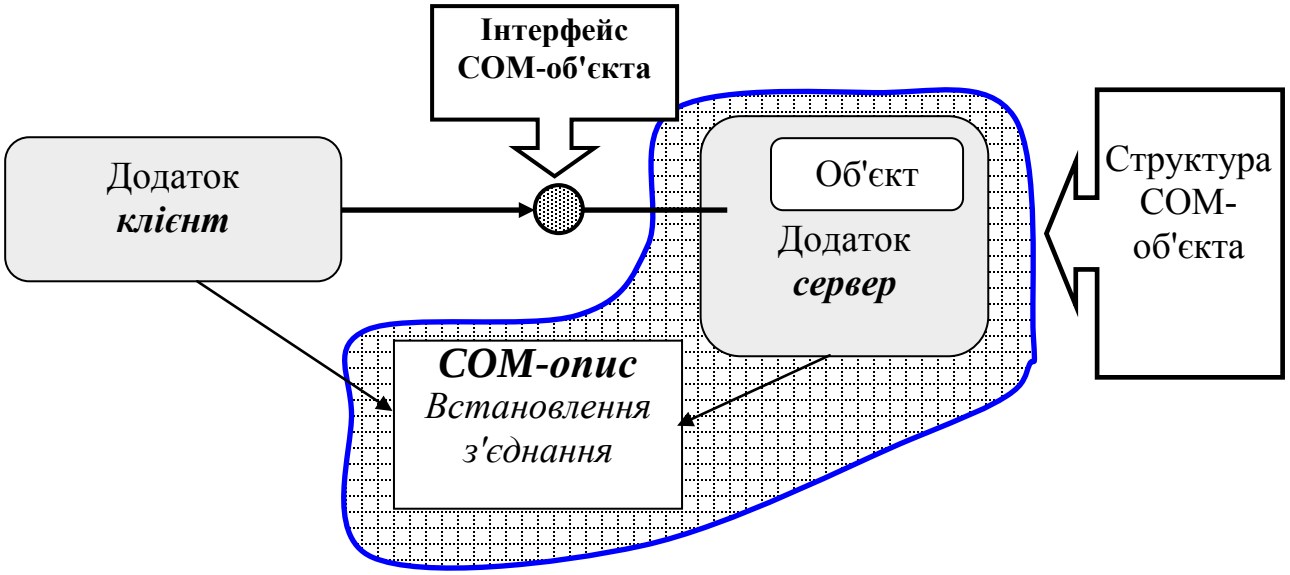


Рис. 6.28. Після встановлення зв'язку за допомогою COM поміж клієнтом і об'єктом-сервером, вони спілкуються без додаткових ресурсів ОС и комп'ютера

У двоярусних додатках існує знижена масштабованість, причиною чого як правило є обмеження на продуктивність сервера баз даних. У типовому двоярусному додатку клієнт напряму з'єднан з сервером баз даних. При цьому від сервера потребується по одному з'єднанню на кожного клієнта, котрий потім тримає з'єднання відкритим у впродовж усього часу життя додатка. Даний підхід, хоча і простий у реалізації, вкрай знижує продуктивність.

У основі архітектури ***багатоярусного додатку*** *(N-tier application[11])* лежить

[11] **Application Architecture: An N-Tier Approach - Part 1. (**By Robert Chartier)
http://www.15seconds.com/Issue/011023.htm?voteresult=5

ідея поділення додатка на окремі функціональні компоненти (рис. 6.29).

Як правило додаток проектується навкруг трьох ярусів служб — користувача, бізнесу і даних — так званий трьохярусний додаток. Багатоярусний додаток є логічним розширенням трьохярусного. У такому додатку один (або більше) з трьох початкових ярусів розбивається на додаткові яруси (рис. 6.29). Це додає додаткові рівні абстракції для опису моделі додатка.

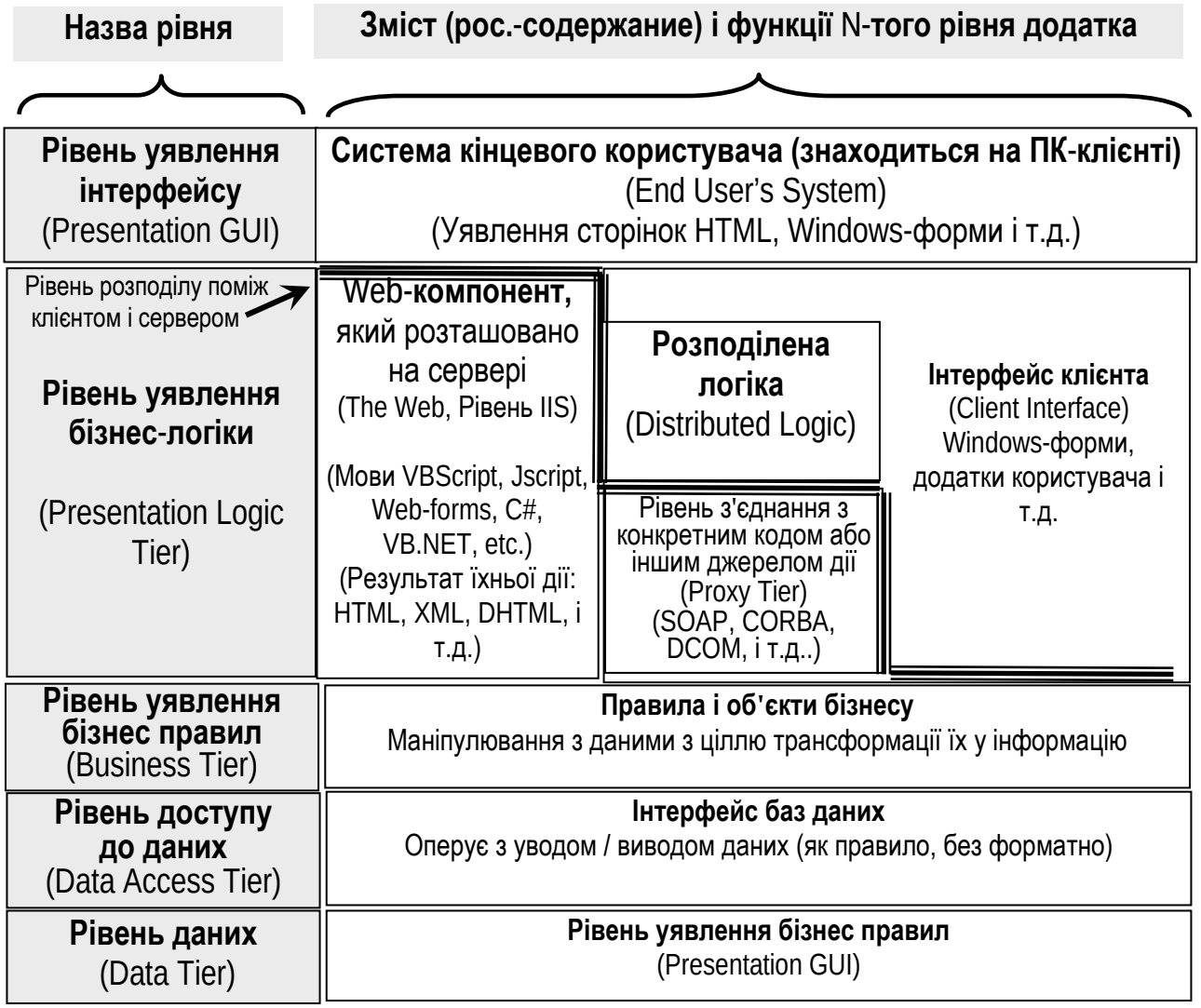


Рис. 6.29 Сучасне уявлення ***багаторівневого*** *(N-tire)* додатку

Web-додатки як правило уявляють собою набір Web-сторінок, які відображаються додатком-контейнером – браузером. З початку браузери могли відображати лише статичні Web-сторінки, але зараз багато які з них підтримують динамічні сторінки, що активно використовується у Web-додатках. Web-додатки об'єднують в собі Інтернет-технології і традиційні додатки і допускають їх локальне розміщення у інтранет (LAN) або глобальне в Інтернет. У свою чергу, Web-сторінки дають користувачеві інформацію про результати роботи додатку.

Більшість Web -додатків є мішаниною (рос.-смесью) HTML-сторінок і коду, який виконується при відображенні. Це може бути код сценарія, що міститься на сторинці або ж двоїчний код додатку або компоненту, які викликаються зі сторінки. Комбінувати HTML і програмний код вдається декількома різними

способами. Можна виконувати код тільки на сервері — така модель збільшує кількість браузерів, які працюють з додатком. Можна виконувати код і на клієнті. Така модель зменшує мережовий трафік і час відгук додатку на запит. Однак не всі браузери підтримують усі можливі моделі реалізації. I Active Server Pages (ASP), і Internet Server API (ISAPI) і Common Gateway Interface (CGI) є різними технологіями для створення Web-додатків, які виконують обробку даних на сервері.

Сумісно працюючі додатки (collaborative applications) — це розподілені додатки, котрі працюють разом. Частини сумісного додатка як правило самі уявляються повно функціональними додатками, котрі можна використовувати і незалежно один від одного. Наприклад, сумісно працюючий додаток може, складатися з додатків, які написані з застосуванням Visual Basic, Microsoft Word і Microsoft Excel, де додаток на мові Visual Basic використовує ***Automation*** для доступу до функціональності Word і Excel. Іншими прикладами сумісно працюючих додатків можна привести Lotus Notes і Microsoft Exchange.

Революційною технологією створення додатків є компоновка їх з компонентів. Компонент (component) - це фундаментальний будівельний блок для розробки розподілених багатоярусних додатків. Компоненти досить часто базуються на компонентній об'єктній моделі фірми Microsoft (Component Object Model, COM) або створюються у вигляді моделі незалежних програмних модулів повторного використання JavaBeans фірми Sun Microsystems і уявляють собою засіб формування функціональності додатків.

Схематично появу цієї нової концепції побудови програм з компонентів можна уявити як перехід до взаємозаміняємості, яка властива пристроям ПК, з яких він будується, мов із кубиків (рис. 6.30)

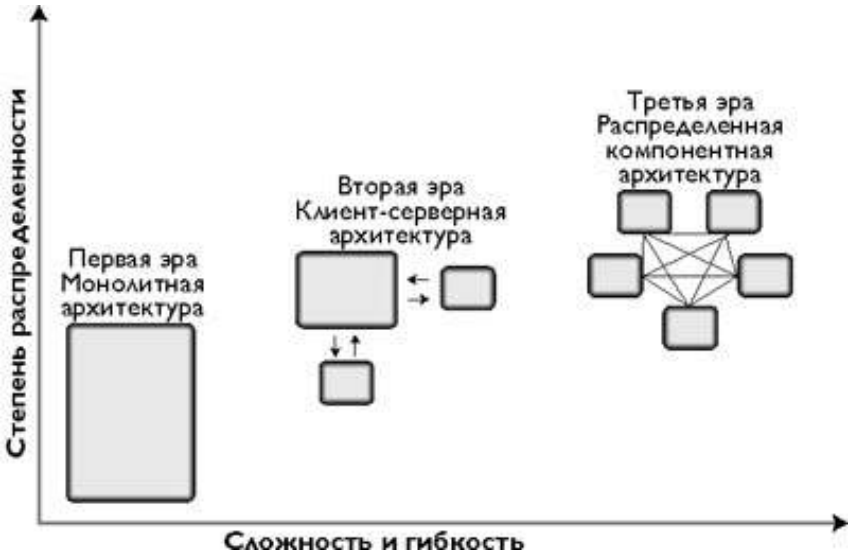


Рис. 6.30. Функціональні зв'язки поміж елементами різних додатків

Компоненти можна використовувати на усіх ярусах багатоярусної архітектури. Проектування компоненту звичайно уключає наступні кроки:

❶ визначення (рос.-определение) послуг, які надає компонент;

❷ виявлення об'єктів, логічним способом (рос.-образом) організуючих

функціональність компонента;

❸ визначення типу компонента: внутрипроцесного (in-process) або позапроцесного (out-of-process).

Так як уся функціональність схована (рос.-спрятана) всередині компонента, тільки компонент знає, як насправді надається ця функціональність, а додаток же ізольований від усіх деталей його реалізації. Таким чином, аби які зміни у внутрішній організації компонента не зачіпає (рос.-затрагивает) інші частини додатку. Послуги і атрибути компонента, які відкриті для зовнішнього світу, звуться, відповідно, ***методами (methods)*** і ***властивостями (рос.-свойствами) (properties) компонента.***

Компоненти призначені для повторного використання: грамотно спроектований компонент можна використовувати у багатьох додатках (рис.6.31). Тому, маючи у своєму розпорядженні набір придатних (рос.-подходящих) компонентів, можна достатньо швидко зібрати додаток з потрібною функціональністю. По технології Microsoft ***компонент*** структурно уявляє собою набір ***класів (classes) COM***, які організовані у вигляді окремих програмних одиниць, що виконуються, а також EXE- або DLL-файлів.

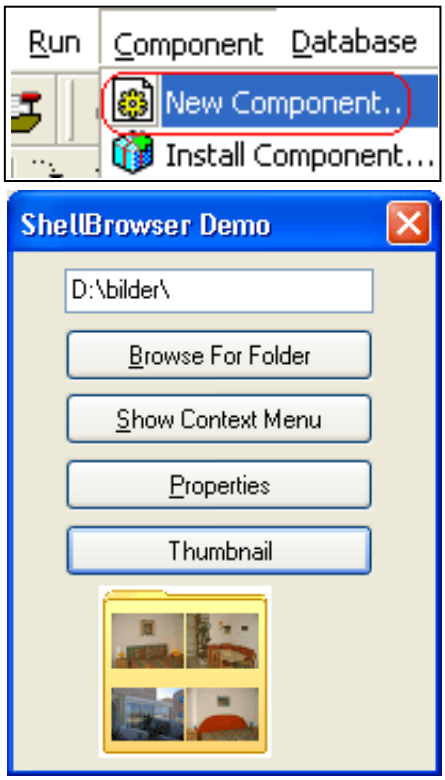


Рис. 6.31. Меню у середовищі Delphi для створення і підключення компонентів (зверху), а також інтерфейс типового компонента для огляду змісту папок (каталогів)

Компонентами нового покоління стали ***Web-сервіси***[12]. Споконвіку World Wide Web була мережею документів. Web-сервери спілкувалися з клієнтами за протоколом HTTP (Hypertext Transfer Protocol) і пересилали інформацію у формі гіпертекстових документів, які створені засобами мови HTML (Hypertext Markup Language). Такі документи як правило відображаються у браузерах і містять посилання (рос.-ссылки) на інші документи у WWW. Функції мультимедіа мови HTML також дозволяють авторам включати у свої Web-сторінки зображення, аплети (програми на мові Java, котрі автоматично завантажуються і виконуються на машині користувача), видеокліпи та інші документи у форматі HTML.

Удосконалення технологій привело до того, що багато комерційних сайтів будуються не на основі звичайних Web-серверів, а за допомогою багатоланцюгової (рос.-многозвенной) архітектури, яка припускає (рос.-подразумевает) використання серверів додатків.

Головною відміною звичайних Web-серверів від серверів додатків є те, що останні не просто повертають документ, а ще й можуть обробляти

[12] Web нового поколения — Web-сервисы. Алексей Федоров. http://www.compress.ru/Temp/1081/index.htm

запити користувачів і містять код, який реалізує бізнес-логіку. Як правило, сервери додатків генерують документи динамічно, у залежності від вказаних користувачем параметрів. Також слід відмітити, що застосування серверів додатків дозволяє створювати масштабовані рішення, які можуть одночасно обслуговувати велику кількість транзакцій і, відповідно, і користувачів.

Поява різноманітних мобільних пристроїв привела до того, що замість традиційних браузерів велика кількість комерційних Web-додатків тепер окрім протокола HTTP підтримують і протокол WAP (Wireless Access Protocol) і здатні (рос.-способны) повертати на запит інформацію не тільки у стандарті HTML, але й у стандарті, що задовільняє користувачів, які звертаються до сервісів за допомогою мобильних пристроїв — WML (Wireless Markup Language) (рис.6.32).

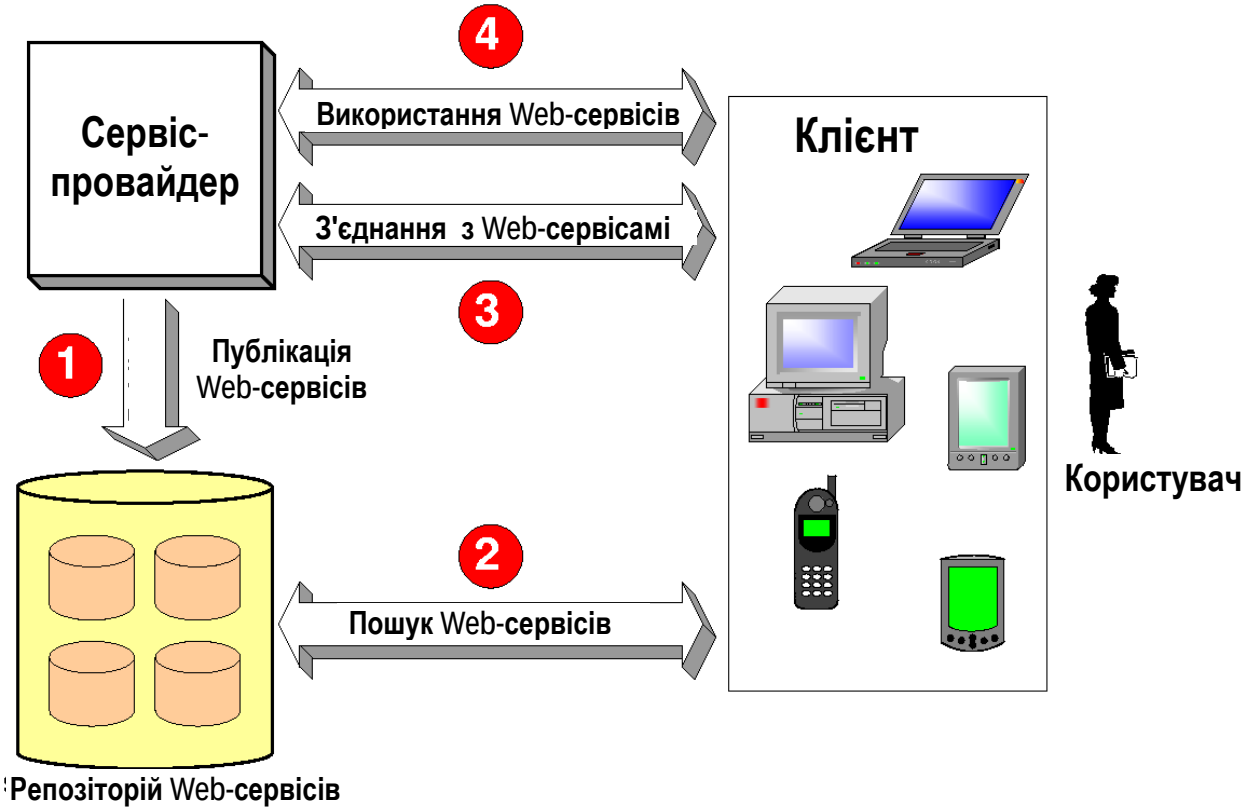


Рис. 6.32.
Процес роботи з Web-сервісами.
1) Публікація Web-сервісів у репозиторії на сервері. 2,3,4) Пошук, з'єднання і використання Web-сервісів мобільними і локальними пристроями.

Природно, електронна комерція не може обмежуватися простою обробкою транзакцій — наступним логічним кроком у розвитку Web (тобто WWW) стала інтеграція бізнес-процессів різних компаній. Таким чином і з'явився сервіс-орієнтований Web. У його основі лежать дві відносно нових технологій — SOAP (Simple Object Access Protocol) і XML (Extensible Markup Language). Згідно цьому сценарію Web складається з набору серверів додатків, які обмінюються інформацією у форматі XML за протоколом SOAP. Основою

сервіс-орієнтованого Web є Web-сервіс — набір логічно зв’язаних функцій, котрі можуть бути програмно викликани через Internet і технологію інтеграції Web-додатків. Інформація про те, які функції дає конкретний Web-сервіс, міститься в документі WSDL (Web Service Description Language), а для пошуку існуючих Web-сервісів використовуються спеціальні реестри, сумісні з специфікацією UDDI (Universal Description, Discovery and Integration) (рис 6.32).

Взагалі, Web-сервісами зветься активний контент, який реалізує деяку функціональність і містить дані, розташовані (рос.-расположенные) на Web-серверах, які підключаються для використання зовнішніми додатками. Web-сервіси повністю незалежні від мови і платформи реалізації. Зовнішні додатки працюють з сервісами за допомогою стандартних протоколів і форматів даних.

Звичайно, різноманітність пристроїв, протоколів обміну даними поміж ними, а також потрібних користувачам функцій постійно розширюють границі застосування і технологій створення і використання додатків, рішень і компонентів. У таблиці 6.3. приведені форми існування і деякі рішення у реалізації програмних компонентів двох головних платформ: Windows і Sun (J2EE). З таблиці слідує, що складність взаємодії між програмними компонентами збільшується і, відповідно, розвиваються моделі і механізми конструювання інформаційних зв'язків поміж ними.

6.7. Сучасні технології створення і використання компонентних додатків, Web-додатків і Web-сервісів

Одним з лозунгів фірми Sun Microsystems, є такий, що: "Мережа - це не просто комп'ютер, це мережовий комп'ютер". І всередині її (мережі) накопичено багато різних платформ, додатків, рішень и програмних систем, написаних на різних мовах програмування, успадкованих систем і багато чого іншого. Це висуває на перший план задачу об'єднання усіх ціх ресурсів у працездатний механізм вирішення корисних задач.

Флагманами у напрямі створення працездатних моделей використання усіх можливостей мережових структур є корпорація Microsoft і фірма Sun Microsystems. Вони довгі роки розробляють і розвивають потужні засоби обробки інформаційних ресурсів на усіх рівнях архітектур WWW та Інтернет. Останнім часом ними розроблені комплексні технології і архітектури Jini від Sun Microsystems і .NET від Microsoft.

Технологія Jini, яка заснована на мові програмування Java і компонентах JavaBeans, розроблялася з ціллю створення системи, котра б не викликала до себе багато уваги, поспівала б за постійними змінами і нарощуванням системи і забезпечувала постійну доступність сервісів при посередництві Інтернет 24 години на добу і 7 днів у тиждень (24 х 7). Безпека системи і конфіденціальність інформації, що передається у мережі досягається за рахунок розподіленої системи безпеки. Нарощування систем можливе за рахунок додавання нових, наслідування і зміни старих сервісів, також доступних при посередництві Інтернет.

Обидві технології спираються на модель Web-сервісів, які уявляють собою технологію інтеграції додатків, котра може використовуватися в Internet також і на базі платформи .Net.

Таблиця 6.3.

Найменування модульних додатків, реалізованих за компонентною технологією

№ пп	Назва компоненту	Сутність поняття (рос.-сущность понятия)
1	***Сервіс*** (service) або компонент	–1) Компонент, здібний (рос.-способний) виконати задачу користувача. –2) WSDL сервіс. Набір кінцевих точок. –3) Дивись Web-сервіс.
2	***Портлет*** (стандартний портальний компонент або DDB-компонент)	Реалізація деякого сервіса, що запускається портальним сервером, котра містить деякі дані, набір власних бізнес-функцій, а також стандартне уявлення на робочих панелях порталу. З точки зору користувача, портлет – це невелике вікно на сторінці портала у браузері, котре дає специфічні функції або інформацію, такі як календар, заголовки новин та ін. С точки зрения разработчика, портлети являются подключаемими модулями (фактически – отдельними додаткями), которие разрабативаются для работи внутри портлет-контейнера портала.
3	*Assembly* («пакет» або «комплект»)	–1) (в .NET) Нова модель розташування додатків, котра спрощує інсталяцию і відслідкування версій. Ключове поняття цієї моделі – асембл (assembly). –2) (в .NET) ***асембл (assembly)***, набір ресурсів і типів, а також метаданих, які описують типи і методи, реализовані assembly. Таким чином, assembly – це самоописаний компонент. Головна перевага таких компонентів у тому, що для їх використання не потрібні ніякі інші файли.
4	***Сервлет*** (servlet)	(В мові Java) Java програма, яка розширює функції Web-сервера, генерирує динамічний контент і взаємодіє з Web-клієнтом на основі парадігми запит/відповідь.
5	***Сервлет-контейнер*** (розподілений)	(В мові Java) Контейнер ***сервлета***, котрий може запускати Web-додаток, котрий скомпонован (зв'язан) як розподілений і може виконуватися у середовищі множини віртуальних машин Java (Java virtual machine), які виконуються на одному хості або на різних хостах.
6	***Аплет (applet)***	(В мові Java) Компонент, котрий як правило виконується у Web-браузері, але може також виконуватися і у інших різноманітних додатках і пристроях, котрі підтримують програмну модель взаємодії аплетів.
7	***Компонент*** (component)	–1) Складова частина розподіленого додатку. –2) Компонент є элементом архітектури з визначеними (заданими) границями. –3) (В мові Java) Програмний модуль рівня додатку, який підтримується контейнером. Компоненти конфігуруються у час розгортання. Платформа J2EE визначає чотири типи компонента: корпоративні (промислові) компоненти «зерна» (enterprise beans), Web–компоненти, аплети і додатки клієнти. –4) Компонент є об'єктом програмного забезпечення, який призначен для взаємодії з іншими компонентами. Він інкапсулює деяку функціональність або набір виконуваних функцій. Компонент має чітко визначаємий інтерфейс і підпорядковується (рос.-подчиняется) правилам поведінки, загальним для усіх компонентів у даній архитектурі. –5) Компонент є абстрактним набором програмних інструкцій (команд) і унутрішнього стану (рос.-состояния), котрий забезпечує перетворення даних через його інтерфейс.
8	***Веап (зерно)***	(В мові Java) Повторно використуємий програмний компонент (або компонент рівня ***підприємства***). Може служити складеной (рос.-составной) частиною при створенні додатків.

Продовження таблиці 6.3.

№ пп	Назва компоненту	Сутність поняття (рос.-сущность понятия)
9	*Java Beans*	(В мові Java) *Програмні компоненти* - незалежні програмні модулі, які повторно використовуються і котрі здібні (рос.-способны) взаємодіяти один з одним.
10	***Контейнер***	(В мові Java) Сутність, котра забезпечує життєвий цикл управління, безпеки, ррозгортання і сервіси при виконанні компонента. Кожний тип контейнеру (EJB–Enterprise Java Beans, Web, JSP–Java Server Pages, сервлет, аплет і додаток клієнт) також забезпечує компонентно-конкретизовані сервіси.
11	***Додаток***	(В мові Java) Зібраний у момент виконання з окремих компонентів, з'єднаних через мережу у окремому конкретному середовищі виконання, як правило розташоване (рос.-располагаемое) на різних платформах. Розподілені додатки підтримують моделі: двоярусну (клієнт/сервер), триярусну (клієнт/проміжне ПЗ (middleware)/сервер) і багатоярусну (клієнт/ множинне проміжне ПЗ/множина серверів).
12	***Модуль***	(В мові Java)Програмний модуль, котрий складається з одного або більше компонентів J2EE, які мають одинаковий тип контейнеру і дескриптору (признака) розгортання. Існує три типи модулей: EJB (Enterprise Java Beans), Web і додаток клієнт.
13	*COM, DCOM*	Відкрита архітектура для кросплатформенних розробок клієнт/серверних додатків, котра лежить в основі технологій ActiveX, DirectX і OLE 2.0. Специфікація, модель і технологія корпорації Microsoft, які призначені для побудови і розробки компонентів програмного забезпечення і їх інтерфейсів. COM встановлює абстракції і правила, необхідні для визначення об'єктів і їх интерфейсів, які реалізуються. У її склад входить також програмне забезпечення, яке реалізує ключові функції. Такі компоненти легко об'єднуються у програми або можуть бути додані (рос.-добавлены) до існуючих програм, щоб придати їм більшу функціональність. Компоненти пишутся на різних мовах (більш усього при цьому використується мова C++ або C#). COM сервер як правило є .DLL або .EXE файлом.
14	***Агент***	–1) Пристрій і/або програма, що встановлені у елементах комп'ютерної мережі для централізованого управління цими елементами і усією мережою. –2) Програма, яка діє від особи (рос.-лица) іншого суб'єкту, сутності або процесу.
15	*Web-**сервіс** (Web service)*	–1) Web-сервіси є новою Інтернет-парадигмою, незалежною від платформ і мов програмування. Web-сервіси реалізуються у вигляді автономних, модульних додатків, котрі можуть бути описані, надруковані (рос.-опубликованы), розміщені і бути викликані через електронну обчислювальну мережу для створення нових продуктів і сервісів. –2) Web-сервіс є програмною системой, яка розроблена для підтримки інтероперабельності міжмашинної взаємодії у комп'ютерній мережі. Він має інтерфейс, що описан у машинно-обробляємому форматі (як правило, WSDL). Інші системи взаємодіють з Web-сервісом заданим способом, відповідним опису, який використовує SOAP-повідомлення, типово передаваєме з використанням HTTP з XML серіалізацією у прив'язці до інших Web-орієнтованих стандартів.

Слід відмітити, що на відміну від технологій Java і Jini, наріжним (рос.-краеугольным) каменем програмної моделі Microsoft .NET є технологія Web-

сервісів, яка базується на середовищі .NET Framework, яка у свою чергу основана на наступних базових концепціях:

❶ незалежним від мови програмування середовищем виконання (Common Language Runtime);

❷ бібліотеці класів .NET (.NET Class Library);

❸ мові-посредниці Microsoft Intermediate Language (MSIL);

❹ мовах, які підтримують .NET.

Структура .NET Framework наведена на рис. 6.33. Як видно, мова йде фактично о единому середовищі виконання програм і підтримку процесів їх розробки. Власне тут зібрані базові класи для усіх мов програмування, які реалізоваі у вигляді бібліотеки ядра System, а також великої кількості (більш 20) спеціалізованих бібліотек з іменами System.Data, System.XML і т. д. Над ними розташований набір засобів формування для виконання модулів різного типу (доречи єдиний для різних мов модулів, що виконуються).

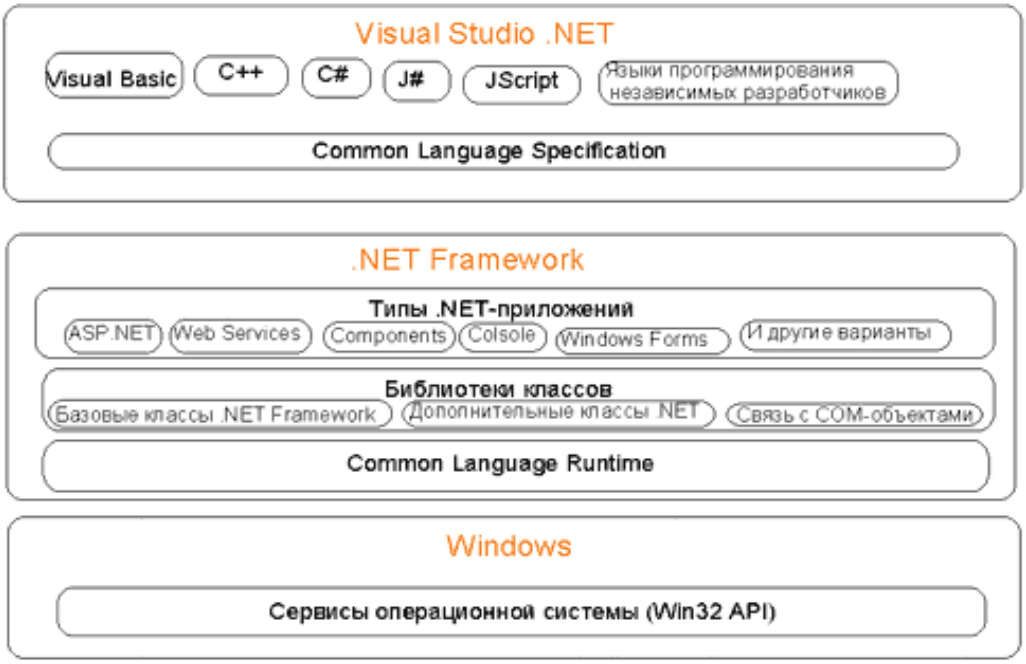


Рис. 6.33.

Структурна схема .NET Framework

Цікаво розглянути деякі риси вищевказаних технологій у порівнянні (див. табл.. 6.4.)

З однієї сторони, допуская мультимовність програмного коду, концепція .NET приваблює (рос.-привлекает) найкраще, що є у таких широко розповсюджених (рос.-распространенных) мовах програмування, як Perl, Cobol, Eiffel, Python і інші. Зіншого боку, це ускладнює роботу над одним проектом группі програмістів, які пишуть відрізки коду на різних мовах програмування. У Jini все простіше. Тут можна обійтися виключно мовою програмування Java.

В останній час з'являються повідомлення про розробку мов, котрі будуть наслідувати краще з Java і додавати свої функції, що запозичені (рос.-

позаимствованные) з інших мов. Серед таких можно назвать технології, що виникли на стику (рос.-стыке) двох мов. Це JPython, PERCobol, Tcl/Java project, Eiffel-to-JavaVM. Ці мови, можливо, сможуть підсилити і без того сильну позицію мови Java серед розробників.

Таблиця 6.4.

Порівняння основ технологій Jini та .Net[13]

Технології .NET	**Технології Jini**	**Різниці**
C#	Java	C# і Java є C-подібними мовами. При побудові деяких компонентів у C# використовувся пакет JavaBeans. Головною різницею є те, що C# запускається виключно під Windows, а Java працює на будь якій платформі, де є JVM. Обидві мови прекомпілюются 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+ буде використовувати код, який написано на Visual Basic, C# і на інших мовах програмування. JSP, у свою чергу, використовує код, який написаний, виключно, на Java.
Interface Language (IL) Common Language Runtime	Java Virtual Machine (JVM) и CORBA	Ядро технології .NET, Common Language Runtime, дозволяє запускати код, який написано на будь якій мові програмування, у середовищі Windows і інших операційних системах, а також використовувати сумісний набір компонентів и об'єктів. JVM дозволяє запускати, прекомпільований з Java, байт-код на будь якій платформі. CORBA дозволяє використовувати сумісний набір об'єктів, які написані на різних мовах програмування, на будь якій платформі.
ADO+ і SOAP Web Services	JDBC, EJB, JMS і Java XML Libraries (XML4J, JAXP)	ADO+ використовує модель обміну даних, що заснована на моделі XML і SOAP, котра дозволяє створювати надбудови (рос.-надстройки) над протоколом HTTP. У Jini ця модель реалізована по-іншому: технологія зоставляє можливість ррозробляти транзакції самим програмістам за допомогою бібліотек і компонентів мови Java.

Декілька слів потрібно сказати про середовище розробки компонентів і додатків Microsoft Visual Studio.NET. Це набір засобів швидкої розробки додатків RAD для створення наступного покоління веб-додатків і веб-сервісів на базі XML (рис.6.34).

Visual Studio.NET включає єдину IDE з підтримкою функцій RAD для побудови веб-додатків і компонентів проміжного рівня, які відповідають за

[13] Jini[tm].NET. *Александр Волоха (alex_frost@ukr.net),* www.ixbt.com/editorial/ jini-vs-net.shtml

бізнес-логіку, а також RAD XML-конструктори для роботи с даними. Microsoft пропонує п'ять мов "власного виготовлення": VC++, C#, VB.NET, Jscript і J#.

У саму пізню попередню версію VS.NET уключені перші три. Незалежні поставщики також можуть підключати свої засоби програмування до середовища VS.NET. До теперешнього часу відомо більш ніж 30 систем на базі різних мов (Cobol, Fortran, Perl, Python і т. д.), котрі підтримують створення .NET-додатків.

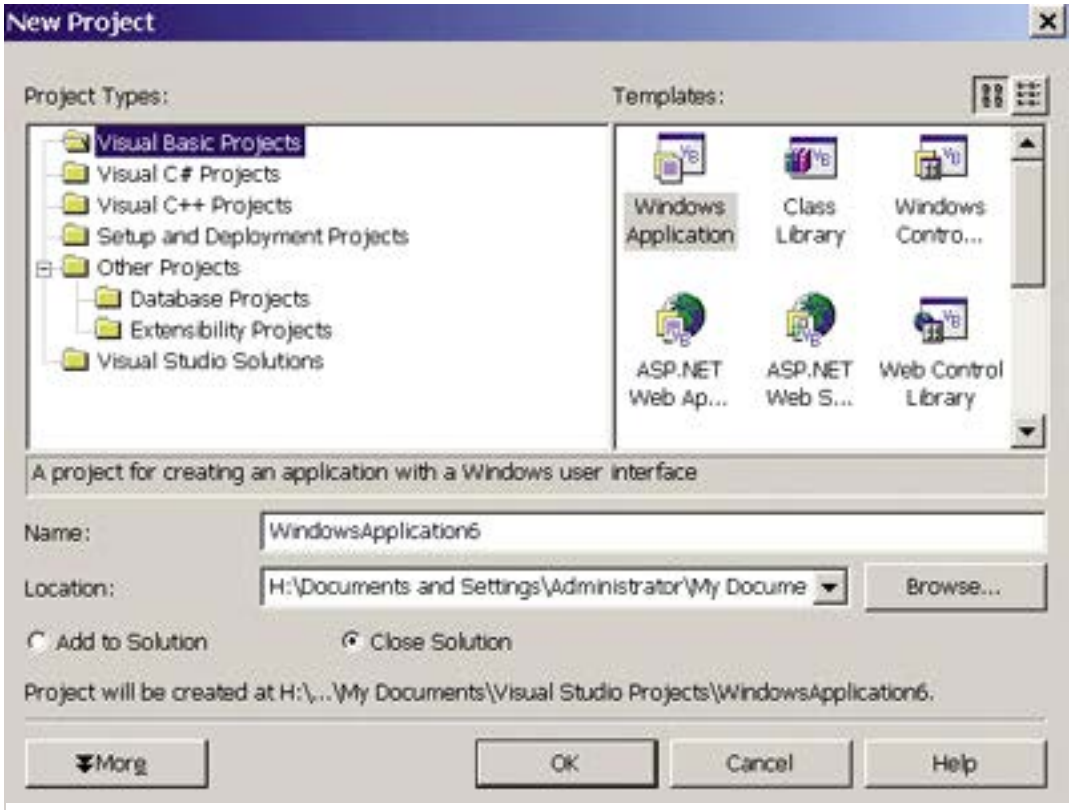


Рис. 6. 34.

Вибор мови і типу проекта у RAD IDE Visual Studio.NET

Visual Studio.NET дає можливість розробникам швидко проектувати веб-додатки широкого спектру застосування для будь яких пристроїв і будь якої платформи. Visual Studio.NET повністю інтегрується з .NET Framework, з забезпеченням підтримки множини мов програмування і автоматично вирішуючи багато які розповсюджені задачі програмування; завдяки чому розробники визволяються від рутинної роботи і можуть швидко створювати веб-додатки на тій мові, котрій віддають перевагу.

Запитання.

1.	Що собою уявляє програма?
2.	Яки назви мають послідовності операторів мов програмування у різному контексті?

3.	Скільки Ви можете згадати назв програм і їх різновидів ?
4.	Чим відрізняються поміж собою програма, підпрограма і функція?
5.	У яких середовищах пишуться програми?
6.	Для чого призначені API-інтерфейси?
7.	Які головні системні функції програмного інтерфейса (API)?
8.	Які переваги дають у використанні RAD-системи?
9.	У яких архітектурних середовищах виконуються програми?
10.	Чим архітектура комп'ютера відрізняється від архітектури мережі?
11.	Які головні складові архітектури Фон-Неймана?
12.	Назвіть головні архітектури, що існують у реальних обчислювальних системах?
13.	Які основні етапи розробки додатка або рішення?
14.	Які існують типи додатків, в залежності від архітектури, у якої вони працюють?
15.	У яких формах існують компонентні додатки, як вони звуться і від чого залежить така їх велика кількість?
16.	Які сучісні обчислювальні і програмні архітектури найбільш популярні?

Повідомлення – це передача інформації від одного об’єкта до іншого, у розрахунку на те, що за цим буде виконана деяка діяльність. Повідомленням може бути сигнал або виклик операції.

Мова UML.

7. МОВА UML ТА ЇЇ ЗАСТОСУВАННЯ

7.1. Причини появи об'єктно-орієнтованого підходу та мови UML

Програмування за всіх часів розвитку комп'ютерів було непростою справою. Особливо це стосувалося розробки великих програмних систем. Наприклад, Джозеф Фокс, один з керівників фірми IBM (у 1969-1977рр.), наводив приклад розробки інформаційної системи з управління польотами космічних кораблів типу Аполон/Скайлеб (рис. 7.1). Її загальна вартість складала 209 млн. дол.(!), а розробкою займалися 700 програмістів у впродовж 7-мі років.

Рис. 7.1. Пункт управління польотами кораблів Аполлон (зліва)
та момент стиковки космічних кораблів Аполлон і Союз (праворуч)

При цьому кількість машинних команд у кінці роботи над системою досягла 23 млн., а розробка велася на мовах Фортран і Assembler. Ті, хто хоча б раз запрограмував деякий алгоритм на мові Assembler’а зрозуміють, що значить відобразити командування комп'ютером у покроковому режимі. Доречи, у операційної системі Windows ″зразку″ 1998 року кількість команд була така ж сама! І не дивно, що нарікань на роботу цих систем завжди вистачало. По оцінкам експертів, від 40 до 60 відсотків крупних програмних систем у різних предметних галузях провалюються і не доводяться до свого кінця. І не даремно

у нової моделі розробки програмних рішень корпорації Microsoft MSF (див. підрозділ 6.5) однією з фаз її розробки є оцінка ризиків процесу їх створення[14].

Слід добре розуміти, що на складну, багатофакторну, складнопрогнозуєму працю великих колективів фахівців різних напрямів (аналітиків, управлінців, системотехніків, програмістів, системних адміністраторів, адміністраторів баз даних, технічних письменників (рос.-писателей), тестерів, складальників (рос.-сборщиков) компонент і деяких інших) дуже впливають фактори, які спеціалісти виділили у 5 рівнів ***складності програмування***:

❶ складність самої задачі, що вирішується;

❷ складність мови програмування або ще й комплексу тих мов, що задіяні у складному проекті;

❸ складність середовища виконання програми (архітектура апаратних та програмних засобів), яка зараз є суттєво мережною і розподіленою;

❹ організація технологічного процесу колективної розробки і створення програмного забезпечення (програмних продуктів);

❺прагнення (рос.-стремление) до універсальності та ефективності алгоритмів і даних (дані завжди розподілені і зберігаються у великих сховищах (рос.-хранилищах)).

Від властивостей (рос.-свойств) ***складності (рос.-сложности)*** не можна визволитися (рос.-избавиться), але можна змінити характеристики її проявів шляхом управління або організації.

По мірі розвитку дисципліни програмування, її методики управління складністю починали шлях від широко знаного фундаментального принципу управління складними системами, – *devide et impera* (розподіляй і пануй (рос.-разделяй и властвуй), лат.). Згідно першої частині цього принципу, при проектуванні складної програмної системи провадиться ***алгоритмічна декомпозиція*** задачі, що вирішується. Ціллю декомпозиції є уявлення системи, що розробляється, у вигляді декількох взаємодіючих підсистем (модулів або блоків), кожну з котрих легше налагодити (перевірити, випробувати) незалежно від інших. У цьому випадку потрібно тримати у голові інформацію про значно меншу кількість деталей. А при розробці великих програмних систем легше розподілити фрагменти головної програми поміж виконавцями і керувати цим процесом. Для подальшого підвищення продуктивності роботи програмістів і програмістських колективів були розроблені методи структуризації елементів коду програм, структури програм і програмних проектів.

У програмуванні структурний підхід вперше з'явився з виникненням перших підпрограм, процедур і функцій, які писалися у так званому

[14] Управління ризиками (risk management)– це одна з ключових дисциплін Microsoft Solutions Framework ® (MSF). MSF бачить у змінах та невизначеності (рос-неопределенности), що з-за них виникає, невід'ємну частину життєвого циклу інформаційних технологій. Дисципліна управління ризиками у MSF відстоює превентивний підхід до роботи з ризиками в умовах такої невизначеності, безперервне оцінювання ризиків і використання інформації о ризиках в рамках процесу прийняття рішень на протязі усього життєвого циклу проекту. Даний шестикроковий процес включає визначення (рос.-выявление) і аналіз ризиків; планування і реалізацію стратегій по їх профілактиці та пом'якшенню можливих наслідків; відслідковування (рос.-отслеживание) стану ризиків і добування (рос.-извлечение) уроків з придбаного досвіду.

процедурному (процедурно-орієнтованому) стилі. При цьому, виконавець просто визначав у програмі змінні і константи, котрі було потрібно зберігати у пам'яті комп'ютера і описував або використовував алгоритм їх обробки (рис. 7.2).

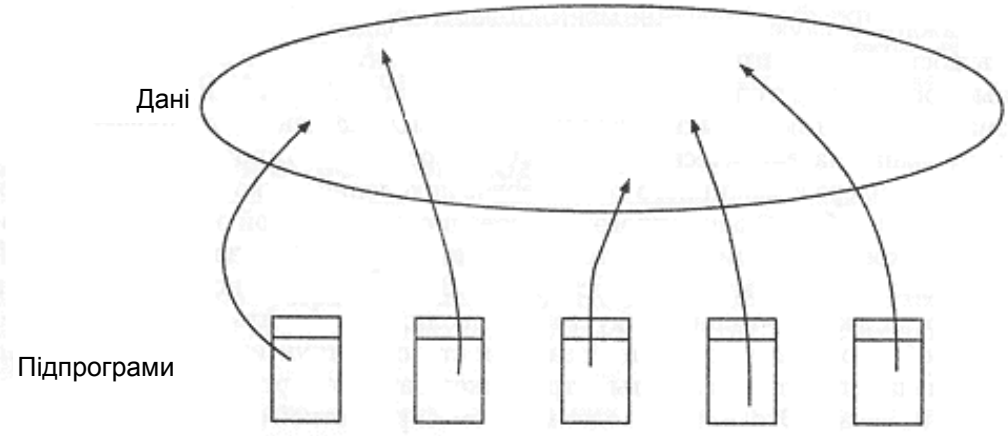


Рис. 7.2. Робота програм з даними у процедурно-орієнтованому стилі

Теоретичне обґрунтування структурний підхід отримав на початку 70-х років у роботах Е. Дейкстри, А.П. Єршова, Е. Йодана, Н. Вірта, Е. Брукса та інших теоретиків і практиків програмування. Подальший розвиток структурного підходу (по мірі збільшення програмних систем і кількості розробників, що одночасно приймали участь у виконанні одного проекту) призвело до початку використання модульного програмування. Воно передбачає декомпозицію прикладної задачі у вигляді ієрархії модулів. Спеціалізація модулів по видам обробки та наявність у них деяких даних визначених типів – це були ті властивості, котрі відображають генетичний зв'язок модульного програмування і об’єктно-орієнтованого програмування (рис.7.3).

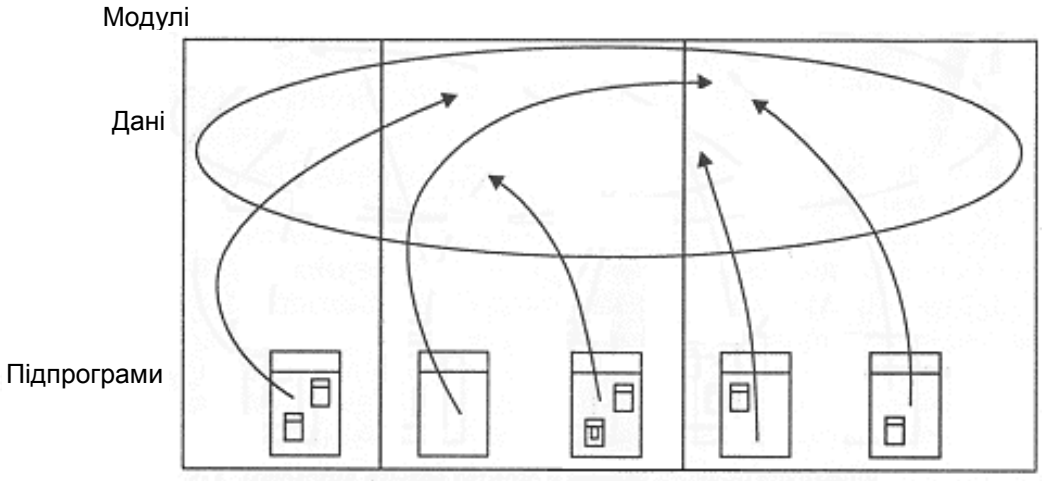


Рис. 7.3. Робота програм з даними у модульно-орієнтованому стилі

Зрозуміло, що найважливішими інструментами виробників програмного забезпечення, у яких знаходять відображення практично усі аспекти еволюції, є мови програмування. Мова програмування з самого початку орієнтована на комп'ютер і містить набір типізованих даних, операторів, операцій, функцій та інших операційних одиниць мови, котрі достатньо просто можуть бути переведені у інструкції (команди) по управлінню апаратним і програмним забезпеченням комп'ютера. Мова програмування, як правило, орієнтована на програміста і дає засоби для моделювання обраних об'єктів, їх властивостей і поведінки при рішенні прикладних задач у деякий предметній області у вигляді програми.

Розвиток мов програмування у напряму підвищення ефективності складання прикладних програм привів до розподілу мов на наступні рівні:

❶ Низький рівень (машинно-орієнтовані мови – мова асемблера)

❷ Високий рівень (процедурно-орієнтовані мови: FORTRAN, ALGOL, PL/1, Pascal. Пізніше деякі з них були дороблені до модульно –орієнтованого рівня у більш пізніх версіях). Сюди ж відносяться мови так званого третього покоління (3GL) (див. главу 5).

❸ Рівень задачі, що вирішується (проблемно-орієнтовані мови – GPS, SQL). Сюди ж відносяться мови четвертого покоління (4GL) (див. главу 5)

❹ Об'єктний рівень (об'єктно-орієнтовані мови – Smalltalk, C++, Delphi Object Pascal, Java і т.д.).

❺ Компонентний рівень (компонентно-орієнтовані мови – C#, Java і т.д.)

Результатом постійного розвитку мов, у яких узагальнення поняття «тип даних» стали класи об'єктів (наприклад, у мові C++) або об'єктні типи (мова Pascal), котрі можуть містити (рос.-содержать) у якості елементів не тільки дані визначеного типу, але й методи їх обробки (функції і процедури) (рис. 7.4).

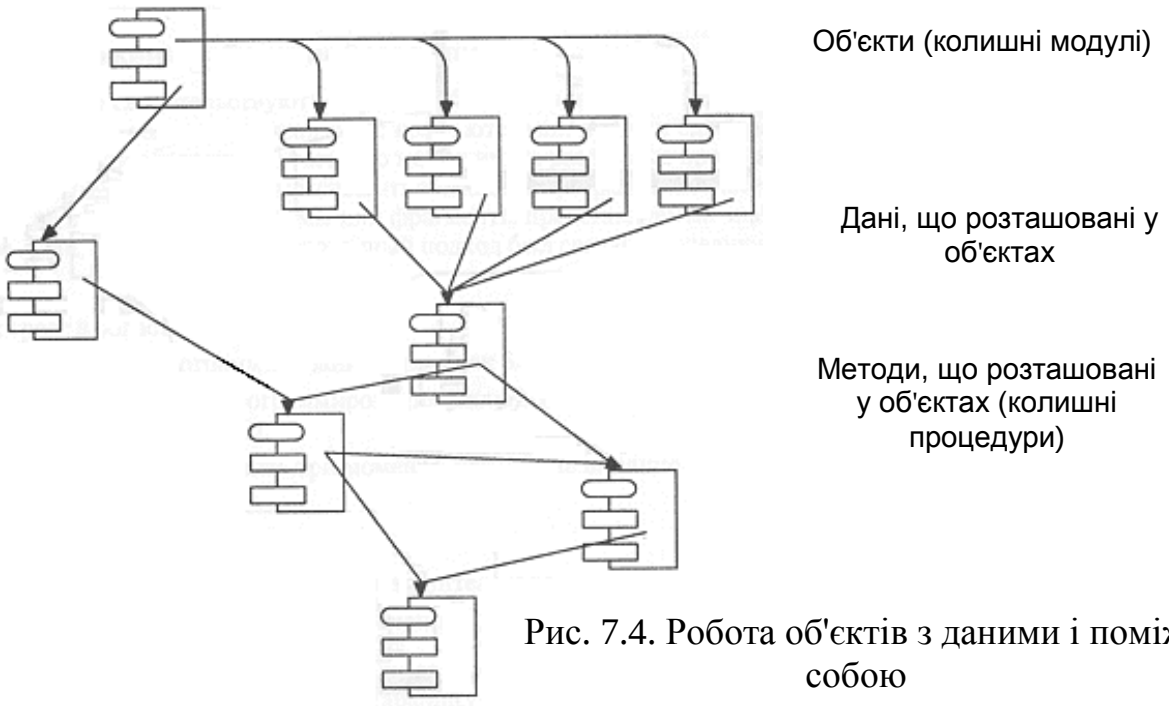


Рис. 7.4. Робота об'єктів з даними і поміж собою

Таким чином, по мірі розвитку технології програмування і у програмах, і в ***типах даних*** усе адекватніше почала відображатися структура прикладної задачі, що вирішується і здійснюватись відповідна інтеграція даних і програм у модулях. По мірі підвищення рівня абстракції у опису структур даних, при проектуванні нових додатків програмісти все далі відходили від фундаментальних (базових) ***вбудованих*** типів даних (тобто символьних, цілих, дійсних (реальних) та логічних). Одночасно з цим мови програмування поповнились засобами, необхідними для опису нових об'єктних структур. Розвиток ідей абстрагування і модульності призвело до появи у програмуванні ***об'єктно-орієнтованого підходу*** (ООП).

І це зрозуміло, адже людина мислить образами і об'єктами, він знає їх властивості і маніпулює ними, у відповідності з визначеними подіями.

До поняття ООП має відношення цілий набір наступних концепцій:

❶ моделювання подій реального світу;
❷ наявність типів даних, що визначає користувач;
❸ приховування (рос.-сокрытие) деталей реалізації;
❹ можливість багатократного використання коду завдяки наслідуванню;
❺ інтерпретація викликів функцій на етапі виконання.

Окрім того, у об'єктно-орієнтованому підході завжди присутні ***три типи даних*** і ***п'ять операційних характеристик***, які є ***найбільш важливими базовими елементами і операціями***. Типи даних уключають: ***об'єкт, клас*** і ***екземпляр.*** П'ять характеристик об'єктно-орієнтованої системи містить: ***абстракцію*** (abstraction), ***наслідування*** (inheritance), ***інкапсуляцію*** (encapsulation), ***поліморфізм*** (polymorphism) та ***динамічне зв'язування*** (dynamic binding). Ці елементи створюють потужне середовище для розробки додатків.

Згідно ООП, об'єкти, котрі ідентичні, за виключенням своїх станів (рос-состояний), під час виконання програми, групуються разом у "***класи об'єктів***". Так і прийшло ключове слово ***клас.*** Створення абстрактних типів даних (класів) – це основоположна концепція у об'єктно-орієнтованому програмуванні (рис. 7.5).

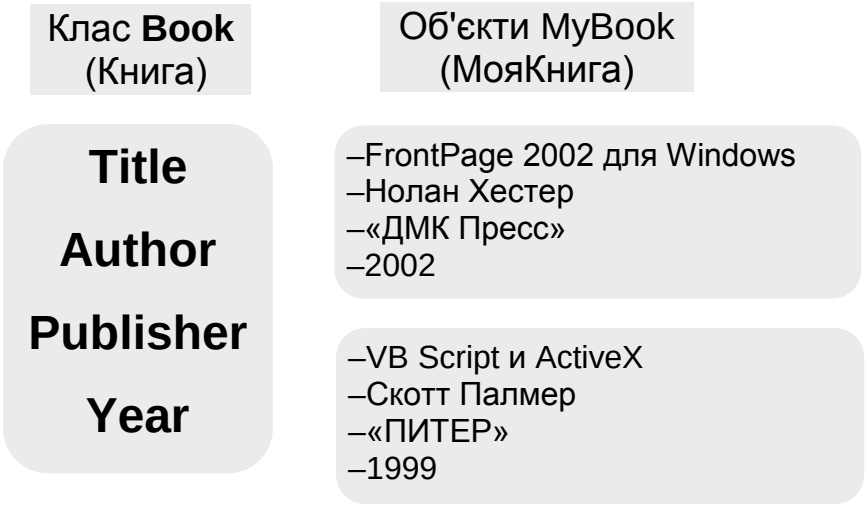


Рис. 7.5. Об'єкти MyBook класу Book

Абстрактні типи даних працюють майже так, як і вбудовані типи: ви можете створювати змінні цього типу (які звуться ***об'єктами*** або ***екземплярами***, якщо говорити об'єктно-орієнтованою мовою) і маніпулювати цими змінними (це зветься ***посилкою повідомлень (рос.-сообщений)*** або ***запитання (рос.-запрос)).*** Ви надсилаєте повідомлення і объєкт розглядає, що з ним можна зробити. Тому вони можуть бути уявлені як унікальні сутності (рос.-сущности) у комп'ютерній програмі.

Так як клас описує набір об'єктів, котрі мають ідентичні характеристики (елементи даних) і риси поведінки (рос.-черты поведения) (тобто функціональність), клас реально є типом даних, тому що, наприклад, число з плаваючою точкою також має набір характеристик і рис поведінки. Різниця полягає у тому, що програміст визначає клас виходячи кожного разу з нової реальної проблеми, щоб уявити відповідний блок для зберігання у комп'ютері. Таким чином програміст розширює мову програмування, додаючи специфікації нових типів даних, котрі йому потрібні.

Таким чином програма дозволяє адаптувати себе до мови проблеми шляхом додавання нових типів об'єктів, так що, коли Ви читаєте код, що описує рішення, то фактично читаєте слова, які описують проблему. Це більш гнучка і могутня абстракція мови, ніж в тих, що розроблялись раніш. ***Тому ООП дозволяє Вам описати проблему у термінах проблеми, а не у термінах комп'ютера, де працює рішення (програма).*** Кожний об'єкт повністю виглядає, як маленький комп'ютер (компонент), він має стан и він може працювати так, як Ви скажете (рис. 7.6).

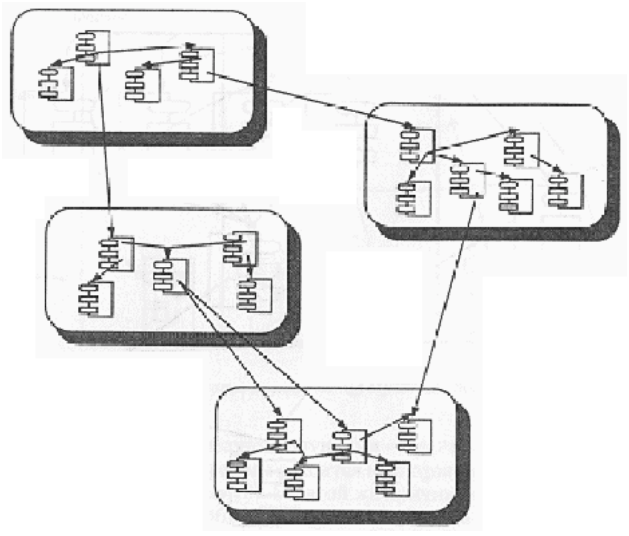


Рисунок 7.6. Взаємодія великих об'єктів (компонентів), що містять всередині себе інші об'єкти

Алан Кей, автор і розробник першої об'єктно-орієнтованої мови програмування SmallTalk (рис.7.7), підсумовує (рос.-суммирует) п'ять головних її характеристик.

❶ ***Все є об'єкт.*** Треба гадати про об'єкти, як про особливі змінні. Вони зберігають дані, але Ви можете "зробити запит" до такого об'єкту, попросив його самого виконати операцію. Теоретично Ви можете взяти аби який умоглядний (рос.-умозрительный) компонент у проблемі, котру Ви хочете вирішити (собак, дома, послугу і т.д.) і уявити його як об'єкт у вашої програмі.

❷ ***Програма - це низка (рос.-связка) об'єктів, які говорять один одному що робити, посилаючи повідомлення.*** Щоб зробити запит до об'єкта, Ви "надсилаєте повідомлення" цьому об'єкту. Правильніше Ви можете думати про

повідомлення, як про запит на виклик функції, котра належить визначеному (рос.-определенному) об'єкту.

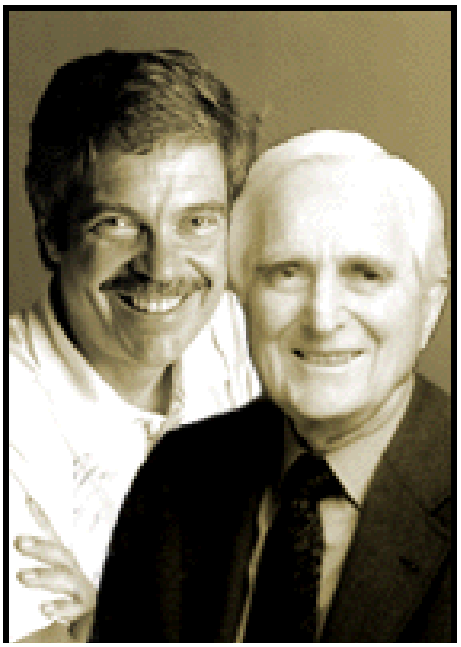


Рисунок 7.7. Автор першої ОО мови програмування Smalltalk Аллан Кей (позаду) і винахідник першого маніпулятора «миша» Дуг Энгельбарт (Doug EngelBart)

❸ ***Кожний об'єкт має свою власну пам'ять, відмінну від інших об'єктів.*** Говорячи іншими словами, Ви створюєте об'єкт нового виду, створюючи пакет, що містить існуючі об'єкти. Тому, Ви можете побудувати складні зв'язки, ховаючи їх позаду простих об'єктів.

❹ ***Кожний об'єкт має тип.*** Іншими словами, кожний об’єкт уявляється ***екземпляром класу***, де “***клас***” - це синонім “***типу***”. Більшість важливих відмінностей характеристик класу у тому, “***Які повідомлення Ви можете йому посилати***?”

❺ ***Усі об'єкти визначеного типу можуть приймати однакові повідомлення.*** Це дійсно важливе твердження. Так як об'єкт типу “круг” також є об'єктом типу “форма”, круг гарантовано буде приймати повідомлення форми. Це означає, що Ви можете писати код, котрий говорить формі і автоматично управляє усім, що відповідає опису форми. Це уявляється однією з найбільш корисних концепцій ООП.

Зрозуміло, що програмування було б дуже легкою справою, як би просте об'єднання об'єктів вирішувало аби яку проблему. Самі проблеми стають усе масштабнішими і складнішими. Повинна була з'явитися дисципліна не тільки ОО-програмування, але й ОО-проектування. Такою дисципліною і повинна була стати мова UML.

7.2.Моделювання складних інформаційних систем

Поміж тим, сучасний ринок програмного забезпечення диктує жорсткі вимоги до інформаційних систем, які розробляються. Системи повинні бути надійними, високопродуктивними, мати гнучкість до впровадження аби яких змін щодо виконання вимог, що до неї пред'являються, забезпечувати можливість масштабування і нарощування функціональності. Окрім того, розробка повинна вестись швидко, ефективно і з мінімальними видатками.

У зв’язку з цим, однією з важливіших задач розробки стає якісне проведення етапів аналізу і проектування, що забезпечує створення комплексної моделі (або декількох моделей) функціонування програмної системи, котрі можуть заложити твердий фундамент для подальшої її програмної реалізації з ціллю виконання вищенаведених вимог. Розробка моделі складної програмної системи перед безпосередньою її реалізацією зараз

є невід'ємною частиною усього проекту, так же як креслення (рос.-чертеж) є основою для побудови великої будови (рос.-здания). Створення моделі необхідно також і тому, що неможливо охопити з першого погляду, як всю складну систему у цілому, так і багато її окремих функціональних частин (рис. 7.8).

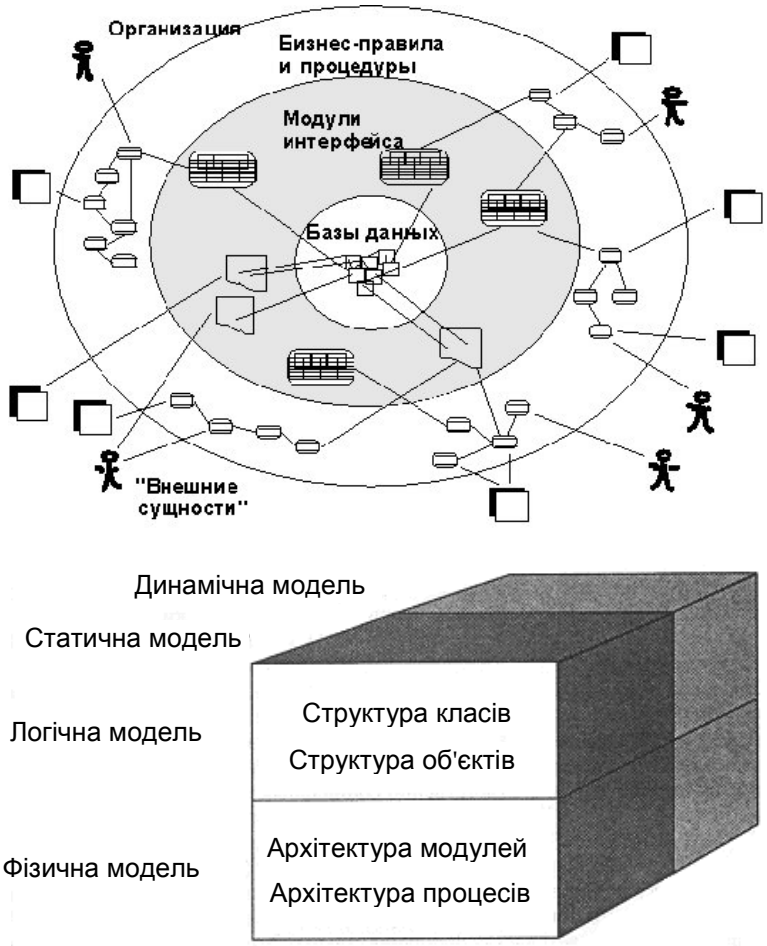


Рисунок 7.8. Фізична модель складної інформаційної системи і комплекс її об'єктно орієнтованих моделей у взаємозв'язку

Таким чином з розвитком об'єктно-орієнтованих (ОО) мов[2] і технологій ОО програмування стали розвиватися і загальні об'єктно-орієнтовані методи розробки ПЗ. Насправді, безглуздою виглядає ситуація, коли ПЗ спроектовано за допомогою структурного методу, а реалізовано у об'єктно-орієнтованому стилі. Адже головною задачею попередніх етапів розробки ПЗ, які головують безпосередньому програмуванню, є специфікація предметної області у термінах, зручних для подальшого застосування у процесі розробки. В цьому випадку, здійснюється переклад інформації з того вигляду, у якому вона існує у свідомості (рос.-сознании) фахівців предметної області (інженерів-телефоністів – при розробці ПЗ для телефонної станції, інженерів-гідрогеологів

[2] У 1972 році була створена перша ОО мова SmallTalk, а у 1980 – ОО мова C++.

– при розробці геоінформаційних систем розрахунку та уявлення підземних водних ресурсів, інженерів-залізничників – при створенні систем автоматизації залізничних перевезень і т.д.) на мову програмістів і програмних систем (рис. 7.9).

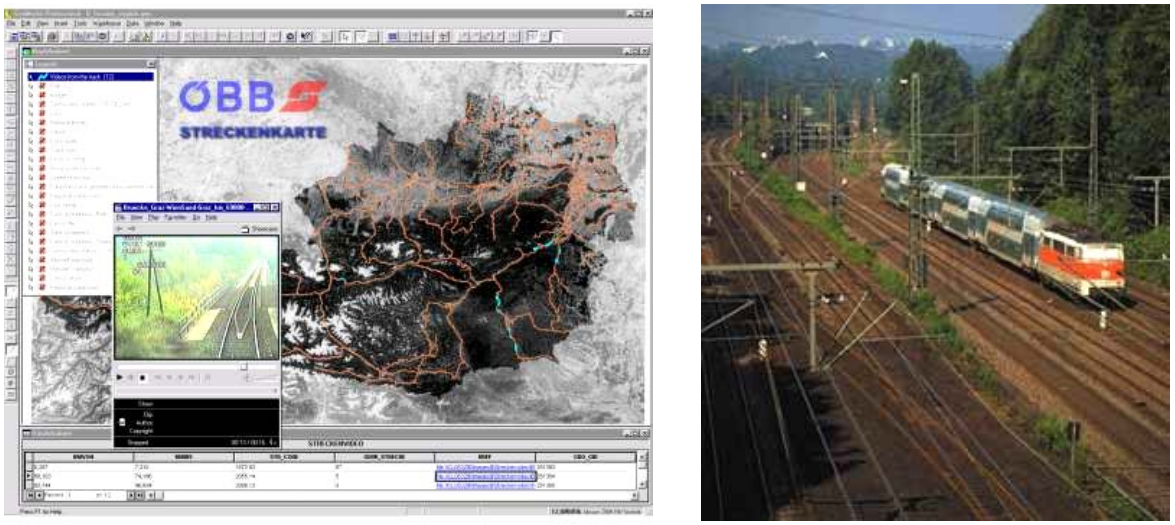


Рис. 7.9. Автоматизація залізничних перевезень за допомогою геоінформаційних систем (ГІС) у Австрії

Важливою ланкою (рос.-звеном) у розвитку ОО технологій аналізу, проектування і розробки програмних систем стало утворення в 1989 році некомерційного консорціуму «Група Управління Об'єктами» (Object Management Group – OMG[3]) (рис. 7.10), котрий об'єднав таких провідних виробників ПЗ, як DEC, HP, IBM, Microsoft, Oracle, Rational Software та ін. Це об'єднання стало могутнім каталізатором активізації продовження робіт з уніфікації методів та розробці індустріальних стандартів у області створення інтероперабельних[4] неоднорідних розподілених об'єктних середовищ у інформаційних технологіях.

Прийняття OMG, починаючи з 1991 року, ряду версій індустріального стандарту CORBA (Common Object Request Broker Architecture) і деяких інших, пов'язаних з його інфраструктурою, а також активне практичне впровадження (рос.-внедрение) самих CORBA-технологій, не могли не викликати великий інтерес у розробників і користувачів цього стандарту до методів ОО аналізу і проектування, до усвідомлення необхідності розробки стандарту мови

[3] Заснована у квітні 1989 року одинадцятью компаніями, Object Management Group™ (OMG™), є неприбутковою організацією, що вже у 2003 році об'єднувала більш ніж 800 организацій-членів. В рамках діяльності корпорації розробляються комерційно перспективні і незалежні від виробників специфікації для софтверної індустрії. OMG™ продвигає Архітектуру, Ведомую Моделлю (Model Driven Architecture ™) у якості «Архітектури Вибору для Зв'язаного (комунікаціями) Світу» ("Architecture of Choice for a Connected World" ™) на базі розробляємих нею стандартних всесвітньо ввідомих специфікацій: CORBA®, CORBA/IIOP™, UML™, XMI™, MOF™, Object Services, Internet Facilities і Domain Interface.

[4] Інтероперабельність – взаємна можливість/властивість інформаційних систем або комп'ютерів обмінюватися повідомленнями, виконувати програми або пересилати дані поміж їх різними функціональними блоками таким чином, щоб користувач при цьому практично ничого не повинен був би знати про особливості цих блоків. Підтримується засобами розвинутих багаторівневих інтерфейсів.

моделювання, котрий повинен був би стати основою просування (рос.-продвижения) вищевказаних технологій. У коло задач OMG входила також підтримка створення систем, які б опиралися на архітектуру CORBA, забезпечення інтероперабельності об'єктно-орієнтованих інструментальних середовищ, що використовували б такі технології та інтеграція накопиченого багаточисельними колективами досвіду у цій галузі.

Рис. 7.10. Логотип організації OMG

Разом з тим, різноманіття виникаючих у 80-х роках методик і технік ОО моделювання призвело до справжньої "***війни методів***", розгорнутою конкурентами. Це стало причиною того, що ідеї об'єктно-орієнтованого підходу при проектуванні систем стали втрачати своїх прихильників. Тим часом виробничий світ очікував появу концепції і реалізуючої її мови, здатних об'єднати кращі методи і навести (рос.-представить) загальну нотацію для ***єдиного уніфікованого уявлення ОО моделі аби якої інформаційної системи, що проектується.***

Але так, як к середині 90-х існувало вже більш ніж ***50-и різноманітних об'єктно-орієнтованих мов моделювання***, розробники і замовники утруднялися (рос.-затруднялись) при виборі метода проектування ІС, котрий, як правило, включав у себе і власну нотацію (тобто засоби уявлення систем и їх реалізацій). В той же час, у число передових вирвались покрашені версії деяких конкуруючих методологій, серед котрих особливо виділялися техніка Booch'90 (автор Граді Буч (Grady Booch)) (рис. 7.11), об'єктно-орієнтована методологія OMT (Object Modeling Technique) (автор Джим Рамбо (Jim Rumbaugh)) (рис. 7.12) і ОО підхід Fusion.

Рис. 7.11. Гради Буч

Ці та деякі інші методи почали змішувати техніки один одного, в результаті чого з'явилося декілька широко відомих методологій: *OOSE (Object Oriented System Engineering), OMT-**2 и Booch**'93*. Але, у цілому, кожен з вищевказаних методів володів (рос.-обладал) своїй специфікою і оставався відособленим від інших. Коротко особливості кожного з цих методів можна висловити (рос.-выразить) наступним чином.

OOSE уявляв собою UseCase-орієнтованим підходом, котрий забезпечував прекрасні рішення у області бізнес-інжинірінгу[5], а також при аналізі потреб до системи.

OMT-2 добре зарекомендував себе на стадії аналізу у системах, де вирішальну роль грала модель збереження (рос.-хранения) даних.

[5] Бізнес-інжиніринг –діяльність, спрямована на проектування і реалізацію бізнес-додатків, тобто програмних засобів для вирішення ділових і економічних задач.

Техніка Booch'93 була корисна на стадії проектування і грала велику роль при побудові систем зі складними алгоритмами функціонування.

Рис. 7.12.
Джим Рамбо

У результаті серії переговорів і сприянні OMG розробка ***Уніфікованої Мови Моделювання UML (Unified Modeling Language)*** була почата як уніфікація двох методів (Booch'93 і OMT) у жовтні 1994 року Граді Бучем (Grady Booch) і Джимом Рамбо (Jim Rumbaugh) у колективі і при підтримці великої софтверної фірми Rational Software Corporation. Перша версія Уніфікованого Методу (Unified Method 0.8) була оприлюднена у жовтні 1995. Восени 1995 до роботи приєднався Айвар Якобсон (Ivar Jacobson) (рис. 7.13), який уключив у загальний процес уніфікації ідеї свого методу OOSE. Таким чином, на першому концептуальному етапі UML мав трьох авторів: Буча, Рамбо і Якобсона, кожний з котрих був ідеологом свого ОО методу візуального моделювання.

У жовтні 1996 року була випущена редакція UML 0.91, у котрої були відображені багаточисельні пропозиції, які "три друга (three amigos)" отримали впродовж 1996 року. Каталізатором об'єднання зусиль з уніфікації UML став випуск консорціумом OMG "запиту на пропозиції (рус.-запроса на предложения)" з UML (RFP - Request for Proposal), так як випуск RFP є першим кроком процедури прийняття OMG того чи іншого стандарту.

Рис. 7.13.
Айвар Якобсон

Після цього Rational Software під своєю егідою об'єднала спеціалістів у організацію "Консорціум UML партнерів" ("UML Partners consortium") для вироблення формального визначення (рос.-определения) UML 1.0 у якості (рос.-качестве) єдиного стандарту. У роботі консорціуму прийняли участь представники таких відомих компаній як: Digital Equipment Corp., Hewlett-Packard, i-Logix, IntelliCorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI, Unisys та ін.

Кожен з партнерів у OMG вніс свій вагомий внесок у розробку загального проекту. Hewlett-Packard, наприклад, забезпечувала зв'язок поміж моделями UML і концепцією ***"повторного використання"***, IBM розробляла мову *OCL* (Object Constraint Language – Мова Об'єктних Обмежень (рос.-Ограничений)), яка описує семантику мови UML. Microsoft розвивала концепцію ***компонентного програмування*** у рамках моделі (COM – Component Object Model) і використання моделей і артефактів UML у ***репозиторії***[6]. Слід

[6] Репозиторій – сховище метаінформації (тобто описової інформації) про додатки, компоненти, сховища (рос.-хранилища) даних, бази даних і т.д., які розробляються,. Містить також, моделі процесів, що діються у системі,

відмітити, що вперше програмний продукт Microsoft Repository, заснований на технологіях COM почав розповсюджуватися у складі Microsoft Visual Basic 5.0. До 2000 року було продано більш 700 000 його копій і більш 65-и компаній розробили додатки, які використовували Microsoft Repository (рис. 7.14).

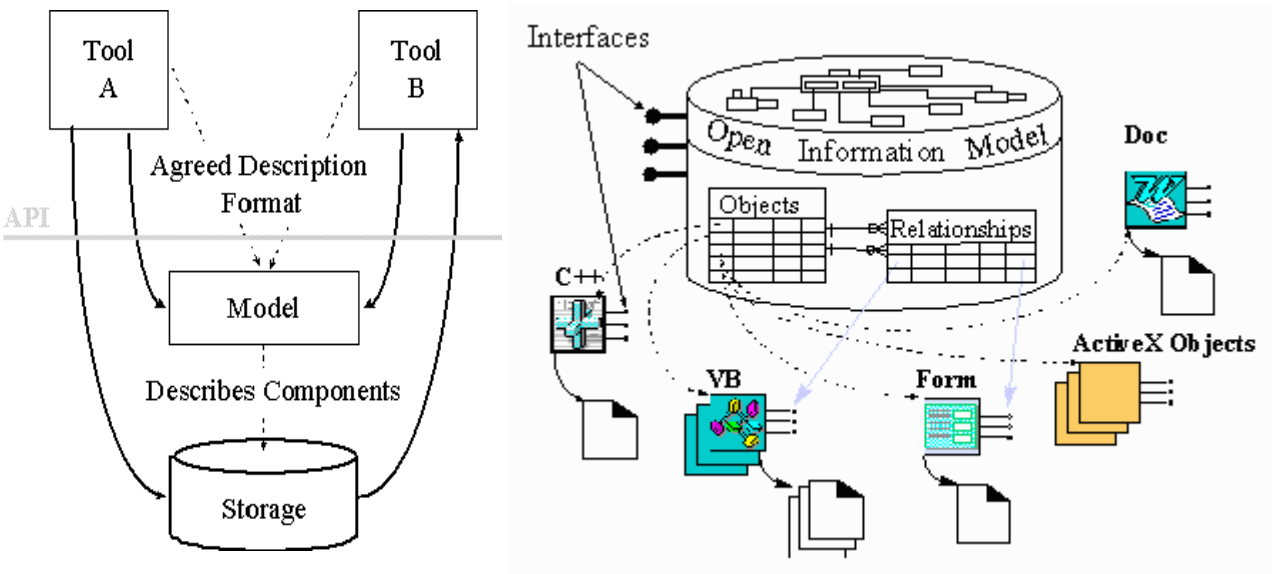


Рисунок 7.14. Модель уявлення репозиторія корпорації Microsoft

Oracle розширила рамки UML для моделювання бізнес-моделей у базах даних. Загальний список співвиконавців вийшов достатньо чисельним (рос.-обширным), так як кожна компанія у роботі вирішувала свої конкретні задачі, але погоджуючи їх з загальним напрямком робіт. З врахуванням зусиль всіх офіційних учасників у січні 1997 р. OMG затвердив варіант UML 1.0, а після розгляду і урахування відгуків представників громадськості і UML 1.1.

Саме цей варіант UML 1.1 у листопаді 1997 року було затверджено як стандарт IT галузі.

Тоді ж OMG прийняла UML як головну специфікацію для OMA (Object Modeling Architecture – Архітектури Моделювання Об'єктів), "***надаючи (рос.-предоставляя) розробникам, що працюють с аби якими мовами програмування і на аби яких платформах, загальну мову моделювання для побудови розподілених і бізнес додатків у інформаційних системах, що повинно вносити необхідну погодженість (рос.-согласованность) у розвиток цієї складної дисципліни***" (так наголосив Річард Солі, голова і виконавчій директор OMG).

UML не є чиєюсь власністю і його відчинено (рос.-открыт) для усіх. Хоч UML є строго визначеною мовою (рос.-определенным языком), він не ставить бар'єрів на шляху розвитку засобів моделювання. UML може бути розширено без перевизначення свого ядра. ***Передбачається використання UML як основи для створення програмних засобів візуального моделювання інформаційних систем з подальшою імітацією їх роботи у цьому середовищі.***

моделі даних, що використовуються системою, а також об'єктну модель з відповідним описом кожної з компонент. Як правило, є спеціалізованим великим програмним продуктом або частиною іншого ПЗ.

У рамках мови і концепції використання UML утворювачі цього ***уніфікованого засобу моделювання*** і їхні партнери проголосили наступний комплекс цілей, що було ***реалізовано***.

❶ Користувачам дається готова до використання, ***могутня мова моделювання***.

❷ Забезпечені для використання ***мовні механізми розширення і спеціалізації*** для різних предметних областей потрібних засобів моделювання, так як при подальшому розвитку UML небажано перевизначати його ядро кожного разу знов і знов. Тому концепції UML припускають її розширення для нестандартних ситуацій, а також містять засоби урахування спеціалізації задач у конкретних предметних областях (наприклад, системи реального часу, Web–технології, Web–сервіси, геоінформаційні системи і т.д.).

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, VBA (Excel, Access, u.a.), VBScript, vi, Visual Age, Visual Basic, 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. 7.15. Звісні сучасні мови програмування

❸ Забезпечена ***незалежність процесу розробки інформаційних систем від мов програмування і моделей цього процесу***, бо UML підтримує усі розумні мови програмування і моделі процесів розробки ПЗ, які прийняті при розробці програмного забезпечення на конкретному підприємстві (рис.7.15).

❹ Створене ***формальне обґрунтування мови моделювання***. Автори наголосили тезу, що формалізація мови повинна бути присутньою у розумних границях, щоб не заважати легкості сприйняття та простоті застосування колективами розробників. Правила та обмеження на конструювання валідних (тобто адекватних) моделей інформаційних процесів визначені ***англійською прозою***, а також і ***Мовою Об'єктних Обмежень*** (OCL). Семантика (тобто змістовність) моделей систем, що розробляються, також описана англійською прозою. Слід чітко розуміти, що англійська проза була використана тільки для того, щоб згідно з задумкою авторів не відштовхнути користувачів від надзвичайної складності створеної системи моделювання, бо разом з тим для усіх цих випадків розроблені принципові математичні докази і відповідні математичні уявлення.

❺ Зроблено важливий внесок у ***розширення ринку об'єктно-орієнтованих інструментів***, які виробляються. Окрім цього, UML повинен забезпечити інтероперабельність для усіх існуючих і нових об'єктно-орієнтованих засобів, що з'являються знов і знов.

Головним практичним виходом розробленої мови UML є то, що вона призначена для специфікації, візуалізації, конструювання та документування артефактів, які відчужуються від розробників та інших матеріалів, пов'язаних з елементами розробки програмних систем.

У концептуальному плані нова мова характеризується наступними відмітними ознаками.

З точки зору змісту UML наслідує принципи і механізми техніки Booch і методів OMT і OOSE. По-друге, він перевершує їх. По-третє, слід відмітити, що UML - це стандарт мови, а не стандарт процесів аналізу, проектування і розробки. UML є квінтесенцією концепцій моделювання, згідно яких було досягнуто консенсусу у середовищі провідних компаній-виробників ОО програмного забезпечення, а конкретно у питаннях:

❶ Погоджень про семантику і нотації для різноманітних аспектів сучасного моделювання у лаконічної формі.

❷ Визначення достатньої глибини семантики для майбутніх технологій, таких як: компонентне програмування, розподілені обчислення, геоінформаційні і безпровідні технології і т.д.

Потрібно також розуміти, що незважаючи на широкі можливості UML, це тільки мова моделювання, тобто засіб моделювання, котрим потрібно ***вміти*** користуватися, ***знати***, коли і де застосовувати ту чи іншу техніку, чітко уявляти собі порядок етапів проектування і структуру моделі, що розробляється.

У інфраструктурі UML важливе місце займають засоби проектування і реалізації. До них відносяться:

❶ ***мови програмування.*** UML – мова візуального моделювання, яка не призначена для візуального програмування, хоча інструментальний засіб, що підтримує нотації UML, може реалізовувати кодогенерацію у конкретну мову програмування (C++, Delphi, Java та ін.) на основі побудованих ***моделей;***

❷ ***інструментальні засоби.*** UML не є специфікацією для розробки інструментальних середовищ. В UML визначена тільки модель семантики, але не визначені моделі інтерфейсу, зберігання і підтримки часу виконання. Зараз вже є декілька CASE – засобів, виробники котрих підтримують моделювання засобами UML. Серед них можна виділити: Rational Rose, Select Enterprise, Platinum і Visual Modeler (рис. 7.16).

❸ ***моделі процесу.*** Деякі підприємства використовують UML як єдину мову при розробці усіх компонентів проекту, хотя конкретні етапи і сам процес розробки можуть бути реалізовані по-різному. При цьому спостерігається збіжність моделей процесів для різних організацій, але загальної згоди по цьому питанню поки ще не знайдено. UML не нав'язує свою модель процесу, але при розробці мови автори мали на увазі процес, що має наступні властивості: ***той, що керується прецедентами, ітеративний та інкрементний.***

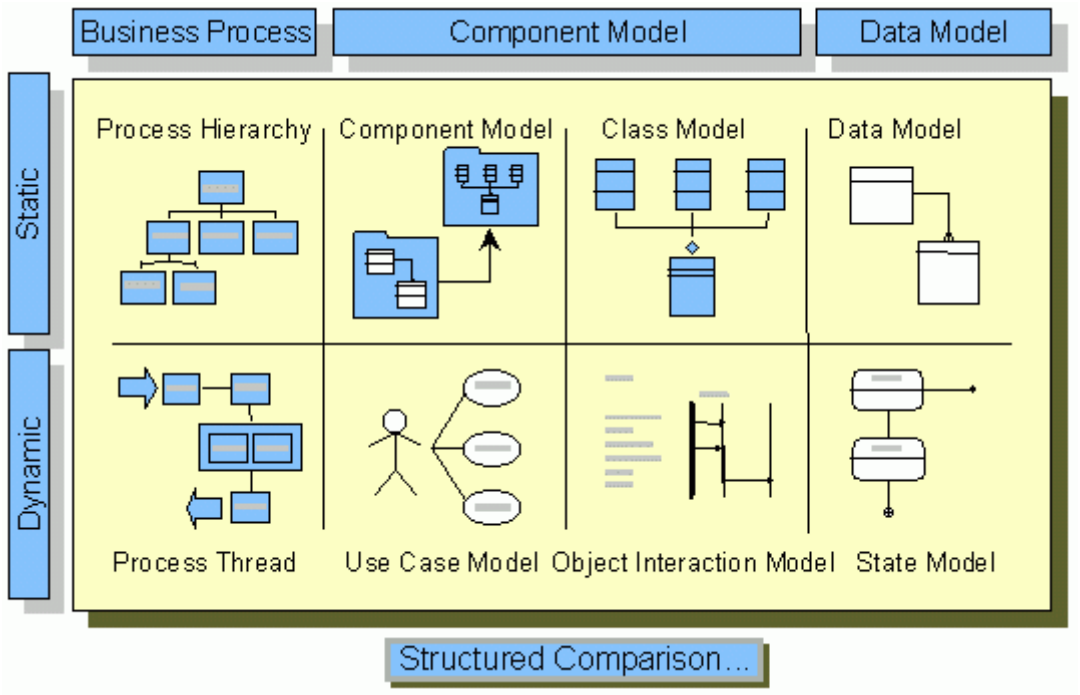


Рисунок 7.16. Застосування моделей UML при розробці бізнес додатків за допомогою CASE-засобів системи фірми Select Enterprise

7.3 Структура і склад мови UML

Розглянемо структуру мови з точки зору компонент, що складають його опис. Визначення (рос.-определение) UML, є самодостатнім і включає наступні документи (у версії 1.3 1999 року випуску):

❶ Семантика UML;
❷ Нотація UML;
❸ Стандартні шаблони;
❹ Визначення засобів CORBA-інтерфейсів;
❺ Специфікації UML XMI DTD (для обміну моделями за допомогою мови XML);
❻ Специфікації мови OCL (Object Constraint Language).

Опис семантики UML ***дається*** на рівні метамоделі і ***уявляється*** наступними погодженими способами та включеними трьома наступними елементами.

❶ Абстрактний синтаксис включає метамодель UML, яка розбита на 9 взаємопов'язаних пакетів, кожний з котрих описується 9-ю діаграмами класів UML (рис. 7.17). Метамодель вміщує у себе коло 50-ти класів, більш ніж 100 асоціацій та 1000 обмежень (рос.-ограничений).

❷ На кожному етапі моделювання можна використати правила спроможності (рос.-состоятельности), які описані англійською прозою з використанням мови обмежень OCL, яка, у свою чергу, є об’єктно-

орієнтованою мовою з рівністю (рос.-равенством) та обмеженими (рос.-ограниченными) кванторами.

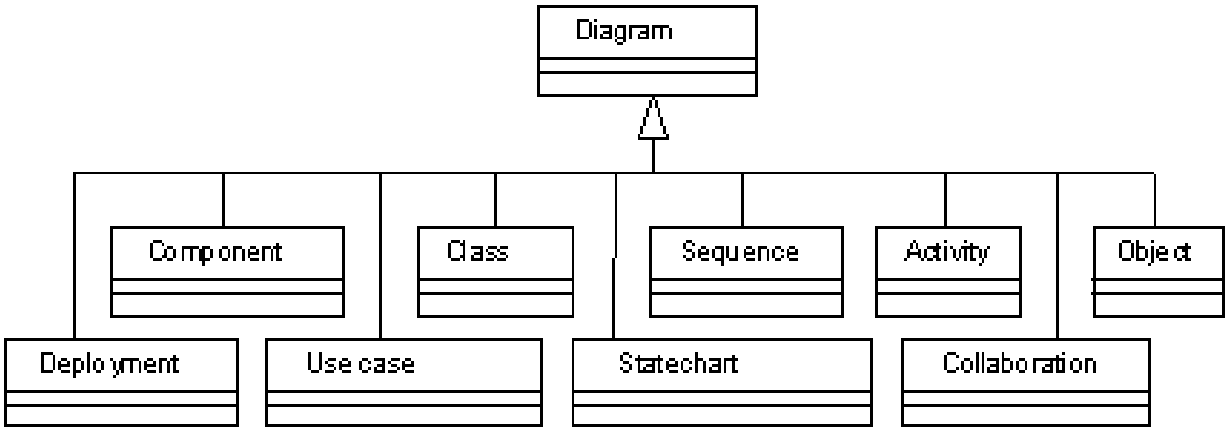


Рис. 7.17. Класи 9-ти діаграм UML,
які уявлені у термінах самої мови

❸ Присутній опис елементів нотації моделювання (семантики), котрий також виконано у англійській прозі.

Розглянуті три описи у сукупності утворюють (рос.-образуют) формальне визначення (рос-определение) UML. Більш формальне визначення, на думку авторів, потребувало б застосування складного математичного апарату, що у кінцевому підсумку утруднило б (рос.-затруднило) повсюдне використання мови, хоч би й додало б суворості (рос.-строгости) визначенням (рос.-определениям). Розширення, що вводяться користувачем у UML можливі завдяки наявності наступних вбудованих у мову механізмів:

❶ Стереотипи дозволяють визначити підкласи існуючих елементів моделювання, зберігаючи форму, але вкладаючи новий зміст та додаючи нові властивості.

❷ Теговані значення - дозволяють визначати нові властивості елементів моделювання, що дає можливість прив'язати до моделі цілком довільну (рос-произвольную) інформацію.

❸ Обмеження (рос.-ограничения) дозволяють визначити підмножину елементів моделювання, наділене визначеними властивостями (наприклад, упорядженість). Вказані умови можуть бути описані за допомогою мови OCL.

7.4. Типи діаграм UML та їх використання

Моделювання складної системи засобами UML приводить до її опису у різних проекціях (рис. 7.18). Кожна проекція описує деякий визначений аспект системи, що розробляється, а усі разом вони визначають систему з належною ступінню (рос.-степенью) повноти, правильності та адекватності. У UML для опису системи при аналізі і проектуванні передбачені наступні 9 графічних діаграм:

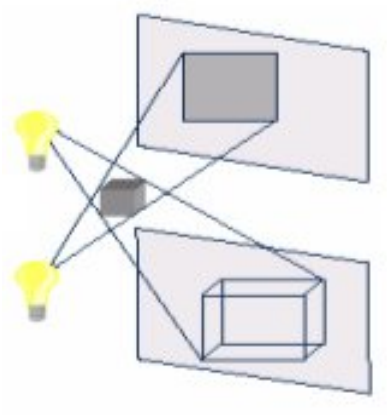


Рис. 7.18. Висвітлення різних проєкцій (аспектів) системи

Діаграми, які описують статичний стан системи:

❶ Діаграми прецедентів (варіантів використання (рос.-использования)) (Use case diagrams).

❷ Діаграми класів (Class diagrams).

❸ Діаграми об'єктів (Object diagrams).

Діаграми, які описують поведінку системи:

❹ Діаграми становищ (рос.-состояний) (State diagrams).

❺ Діаграми видів діяльностей (Activity diagrams).

❻ Діаграми послідовностей (Sequence diagrams).

❼ Діаграми співробітництва (рос.-сотрудничества) (кооперації, взаємодії) (Collaboration diagrams).

Діаграми, які описують фізичну реалізацію системи:

❽ Діаграми компонент (Component diagrams).

❾ Діаграми розгортання (рос.-развёртывания) (Deployment diagrams) (рис. 7.19).

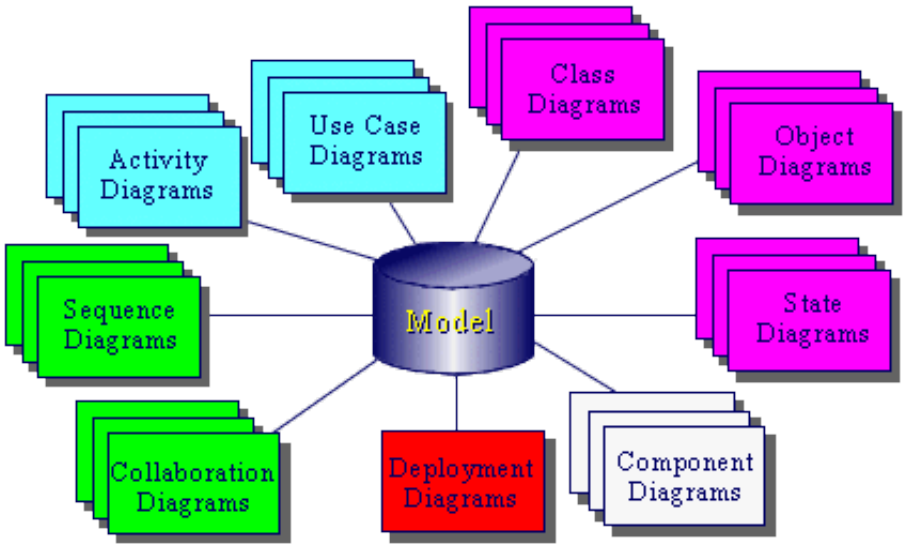


Рис. 7.19. Комплекс діаграм, які описують повну модель системи

Таким чином, мова UML дозволяє:

❶ формалізувати ***функціональні вимоги до системи*** за допомогою апарату ***use case діаграм***;

❷ деталізувати ***вимоги до системи*** шляхом побудови діаграм технологічних сценаріїв;

❸ розбити ***складну систему*** на складові частини з мінімумом взаємних зв'язків шляхом виділення ***пакетів***;

❹ виділити ***класи даних***, побудував концептуальну модель даних у вигляді ***діаграми класів***;

❺ виділити ***класи***, які описують ***інтерфейс користувача***, та створити схему навигації екранів;

❻ описати ***процеси взаємодії об'єктів*** при виконанні системних функцій;

❼ описати ***поведінку об'єктів*** у вигляді діаграм станів (рос.-состояний);

❽ показати ***програмні компоненти*** та їх взаємодію через інтерфейси;

❾ уявити (рос.-представить) ***фізичну архітектуру системи***.

У комплексі, діаграми дозволяють освітити ***п'ять найважливіших сторін системи***, з точки зору розробника (рис. 7.20).

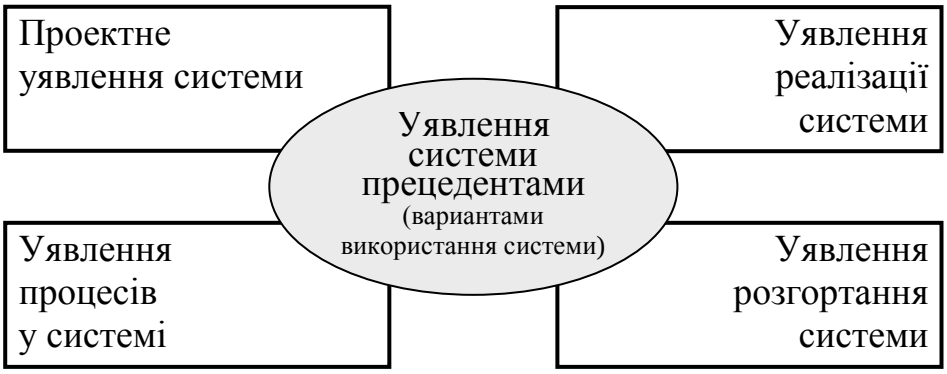


Рис. 7.20. П'ять різних поглядів на систему, які отримуються за допомогою дев'яти діаграм UML.

І так як мова UML велика (рос.-обширная) і складна у плані термінології і нотації, слід коротко описати діаграми кожного з вищенаведених типів.

Техніка ***Прецедентів*** (Варіантів Використання – UseCase) – була вперше запропонована Айваром Якобсоном у 1992 році і швидко завоювала загальне визнання за рахунок простоти і легкості сприйняття (рос.-восприятия) та застосування. Суть її полягає (рос.-состоит) у наступному. Система, що проектується уявляється у вигляді наборів ***актантів (акторів)***, які взаємодіють з системою за допомогою так званих ***прецедентів*** (рис. 7.21). Актантом є будь яка сутність, що взаємодіє з системою зовні.

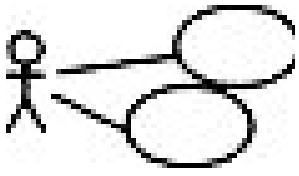


Рис. 7.21. Діаграми прецедентів (Use case)

Їм може бути людина, обладнання, інша система, тобто таким чином розробник визначає, ***що*** взаємодіє з системою. У свою чергу ***прецедент*** описує, ***що*** система предоставляє актанту - тобто визначає деякий набір транзакцій, що здійснює актант при діалогу з системою. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія. Детальний опис всіх цих процесів – уділ (рос.-удел) інших технік моделювання UML. ***Діаграма*** ж ***прецедентів*** несе у собі високий рівень абстракції, що дозволяє ще на ранніх етапах проекту визначати і

зафіксувати функціональні вимоги до системи, її майбутні інтерфейси та забезпечити гнучкий і ефективний механізм взаємодії поміж розробником і замовником проекту.

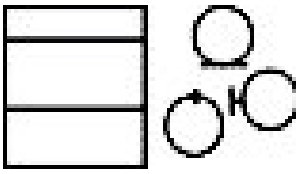


Рис. 7.22. Діаграми класів (classes)

Діаграма класів показує статичну структуру різних частин системи (рис. 7.22). Таким образом, складовими (елементами) даного типу діаграм є класи, об'єкти і відношення поміж ними. Нотація класів і об'єктів проста та інтуїтивно зрозуміла усім, хто колись мав досвід роботи з різного роду CASE-інструментаріями (рис. 7.22, 7.23). Клас представлено прямокутником з трьома розділами (рис.7.22), у котрих відповідно розміщуються: ***ім'я класу***, ***атрибути*** і ***операції***. Схожа нотація застосовується і для об'єктів - екземплярів класу, з тією різницею, що до імені класу додається ім'я об'єкту і весь надпис підкреслюється. Нотація UML надає широкі можливості для відображення додаткової (і як правило дуже важливої) інформації, до котрої можна віднести абстрактні операції і класи, стереотипи, загальні і приватні (рос.-частные) методи, інтерфейси, параметризовані класи і т.д. Важливішими елементами ***діаграм класів і об'єктів*** є ***асоціації***, тобто ***статичні зв'язки поміж класами***, які зображуються у вигляді пов'язаних ліній, на котрих може вказуватися потужність (рос.-мощность) асоціації, її направлення, назву, і можливе обмеження, що реалізує механізм розширення UML (рис. 7.24).

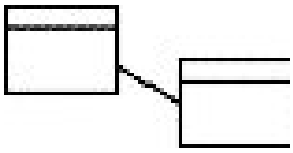


Рис. 7.23. Діаграми об'єктів (objects)

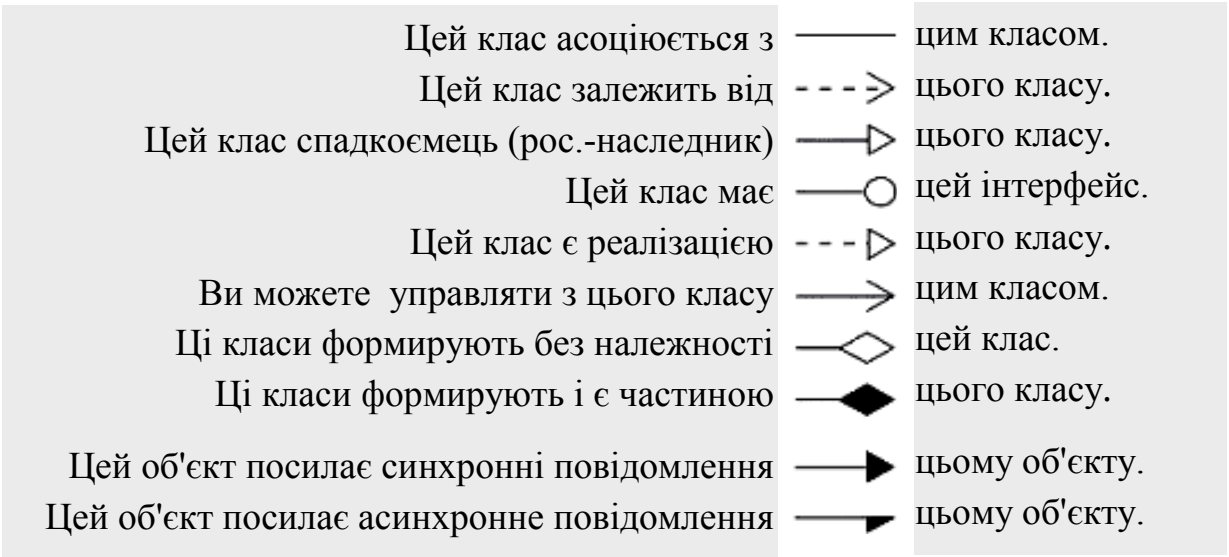


Рис. 7.24. Асоціації діаграм мови UML.

Існує можливість відобразити специфічні властивості асоціацій – наприклад, відношення агрегації, коли складовими частинами класу можуть виступати інші класи. Таке відношення зображується у вигляді ромбу, який

розташовано поруч з агрегуючим класом. Відношення ***узагальнення (рос.-обобщения)*** також має власну графічну нотацію у вигляді трикутника і з'єднуючої лінії, яка дозволяє уявити ієрархію наслідування – від суперкласу до підкласів (рис.7.24).

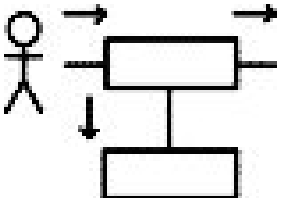


Рис. 7.25. Діаграми кооперації (collaboration)

Діаграми кооперації (взаємодії) описують взаємодію об’єктів системи, для цілей отримання (рос.-получения) деякого результату (рис.7.25). Під отриманням результату розуміється виконання закінченої дії, наприклад, опис у термінах взаємодіючих об’єктів змодельованого раніш прецеденту ***Використання системи*** або деякого ***Сервісу системи***, об'явленого як операція класу на відповідній діаграмі. Діаграми взаємодії уявляються у двох формах – Діаграма слідування і Діаграма кооперації. І перша, і друга описують потоки повідомлень (виклик методів або сигнали) поміж об’єктами, що приймають участь у взаємодії. Служачи, у цілому, однієї цілі, діаграми, по суті своєї, мають істотні різниці.

Діаграма слідування (рис.7.26), робить упирання на часову (рос.-временнŷю) послідовність повідомлень, що передаються. При цьому важливим є порядок, вигляд та ім'я повідомлення. На діаграмі зображуються виключно ті об’єкти, котрі безпосередньо приймають участь у взаємодії, і не показуються можливі статичні асоціації з іншими об'єктами. Таким чином, для ***Діаграм слідування*** ключовим моментом є динаміка взаємодії.

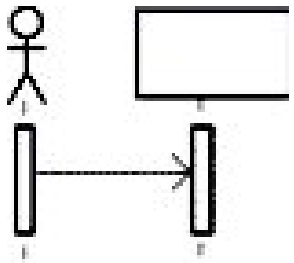


Рис. 7.26.

Діаграми слідування (sequence)

Треба також мати на увазі, що ***Діаграма слідування*** (послідовності) уявляє собою і має два виміри (рос.-измерения). Перше – зліва направо, у вигляді вертикальних ліній, які зображують об’єкти, які приймають участь у взаємодії. Верхня частина ліній доповнюється прямокутником, що вміщує ім'я класу об’єкта або ім’я екземпляру об’єкта. Друге вимірювання - вертикальна тимчасова (рос.-временная) вісь. Повідомлення, які посилаються одним об’єктом іншому, зображуються у вигляді стрілок з іменем повідомлення та упорядковуються по часу виникнення.

Для ***Діаграми кооперації*** головним є можливість відобразити не стільки послідовність взаємодії, скільки усе оточення об’єктів, які приймають участь у ньому. Тобто показані не тільки повідомлення, що посилаються і приймаються, але й побічні (рос.- косвенные) зв'язки поміж асоційованими об’єктами. Стверджують, що ***Діаграми кооперації*** описують повний контекст взаємодії і уявляють собою своєрідний тимчасовий (рос.-временной) "зріз" конфігурації мережі об’єктів, які взаємодіють для виконання визначеної бізнес-цілі програмної системи.

Діаграма кооперації зображує об’єкти, які приймають участь у взаємодії, у вигляді прямокутників, які містять ім’я об’єкту, його клас і, можливо, значення атрибутів.

Асоціації поміж об’єктами, як і на діаграмах класів, зображуються у вигляді сполучних (рос.-соединительных) ліній. Можлива вказівка імені асоціації і ролей, котрі грають об’єкти у даній асоціації. Динамічні зв'язки – потоки повідомлень, уявляються також у вигляді з'єднувальних ліній поміж об’єктами, зверху котрих розташовуються стрілка з указником напряму та ім'я повідомлення.

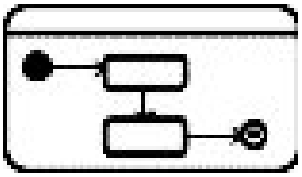


Рис. 7.27. Діаграми становищ (state)

Діаграми становищ описують поведінку об’єкта у часі (рос.-во времени), тобто моделює усі можливі зміни у стані об’єкта, які викликані зовнішніми впливами з боку інших об’єктів або зовні (рис. 7.27). ***Діаграми становищ*** застосовуються для опису поведінки об’єктів і для опису операцій класів. На відміну від ***діаграм взаємодії***, даний тип діаграм описує зміну стану тільки одного класу або об’єкту. Кожне становище об’єкту уявляється на ***Діаграмі становищ*** у вигляді прямокутника з закругленими кутами, містить ім’я становища і, можливо, значення атрибутів об’єкту у даний момент часу. Перехід здійснюється при наступі деякої події – отримання об’єктом повідомлення або прийманням сигналу, і зображується у вигляді стрілки, яка поєднує два сусідніх становищ. Вим’я події вказує на перехід. Окрім того, на переході можуть вказуватися дії, які виробляє об’єкт у відповідь на зовнішні події (при переході з одного стану в інший або при знаходженні у деякому визначеному (рос.-определенном) становищі). Треба відмітити, що ***Діаграма становищ*** описує, як правило, реакцію об’єкта на асинхронні зовнішні події, а для опису реакції на внутрішні події призначені ***Діаграми активності***. Спрацьовування переходу може залежати не тільки від наставання деякої події, але й від виконання визначеної умови, що зветься ***пусковою умовою***. Об’єкт перейде з одного стану у інший, тільки якщо виникла вказана подія і пускова умова прийняла значення "істина".

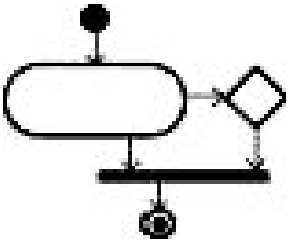


Рис. 7.28. Діаграми активності (activity)

Діаграми активності – окремий (рос.-частный) випадок ***Діаграм становищ*** (7.28). Кожний стан – це сутність виконання деякої операції. А перехід у наступне становище спрацьовує тільки при завершенні операції у початковому становищі. Таким чином, реалізується принцип процедурного, синхронного управління, який обумовлено завершенням внутрішніх дій. Становище, яке описується не має внутрішніх переходів і переходів по зовнішнім подіям.

Їх графічна нотація практично не відрізняється від нотації ***Діаграм становищ***, с той різницею, що на переходах відсутня сигнатура події і додається символ "синхронізації" переходів для реалізації паралельних алгоритмів.

Головним напрямком використання ***Діаграми активності*** є опис операцій класів, коли необхідно уявити алгоритм її реалізації, при цьому кожний крок є виконанням операції деякого класу.

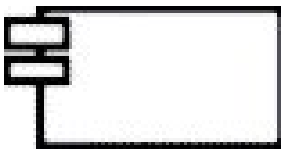


Рис. 7.29. Діаграма компонентів (component)

Діаграми компонентів і ***розгортання***, на відміну від раніш розглянутих діаграм прецедентів, класів, становищ і слідування, котрі є логічним уявленням системи у процесі її розробки, описують фізичне уявлення системи (рис. 7.29, 7.30). ***Діаграма компонентів*** дозволяє визначити архітектуру системи, яка розробляється, встановив залежності поміж програмними компонентами, у ролі котрих може виступати початковий, бінарний і код, що виконується. У багатьох середовищах розробки модуль (або компонент) відповідає (рос.-соответствует) файлу. Пунктирні стрілки, що поєднують модулі, показують відношення взаємозалежності (аналогічні тим, котрі мають місце при компіляції).

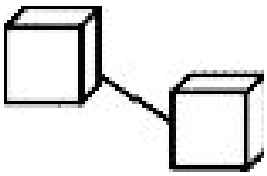


Рис. 7.30. Діаграма розгортання (deployment)

Діаграми розміщення використовуються для уявлення конфігурації компонентів, процесів і об’єктів, які присутні в системі на етапі виконання. Окрім того, ***Діаграми розгортання*** показують фізичну залежність апаратних пристроїв, що задіяні у реалізації системи, а також з'єднань поміж ними – маршрутів передачі інформації.

Питання

1.	Чому все більше ускладнюється процес розробки програм і програмних систем?
2.	Яки і чому існують 5 рівнів складності програмування?
3.	Які існують 5 рівнів у розвитку мов програмування?
4.	Чим викликана поява об’єктно-орієнтованому підходу?
5.	Який комплекс цілей було втілено і реалізовано у мові моделювання процесу розробки програм UML?
6.	Які існують засоби проектування і реалізації у інфраструктурі UML?
7.	Що складає структуру UML і що входить до його складу?
8.	Для чого призначені діаграми UML і як вони звуться?
9.	Які головні процеси розробки програмних систем спрощує мова UML?

Мова – природна (рос.-естественная) або штучна (рос.-искусственная) знакова система, яка призначена для передачі інформації. До природних знакових систем відносяться мови спільності (рос.-общности): українська, російська, англійська та ін., а до штучних, як правило мови програмування: C++, Java і т.д.

Енциклопедія.

8. ВСТУП ДО ТУРБО ПАСКАЛЮ

8.1. Початки (рос.-истоки) Турбо Паскалю

Інформаційні технології (IT), як сукупність засобів та методів обробки даних та інформації за допомогою персональних комп'ютерів, набули статусу головних діючих сил сучасного індустріального суспільства. Розуміння основ IT та оволодіння їхніми головними концепціями та навичками ефективного застосування розглядається зараз у світі як одна з фундаментальних основ освіти на рівні володіння читанням та письменністю. Взагалі, інформатика є галуззю науки, що вивчає структури та загальні якості інформації, а також питання, пов'язані зі збором, зберіганням, пошуком, переробкою, перетворенням, розповсюдженням та використовуванням її у різних сферах діяльності людини. Тому дуже важливо основний натиск у вивчанні мов програмування робити на модель "з орієнтацією на програмування" (programming-first introduction), де одною з найпопулярніших є мова Турбо Паскаль (рис.8.1).

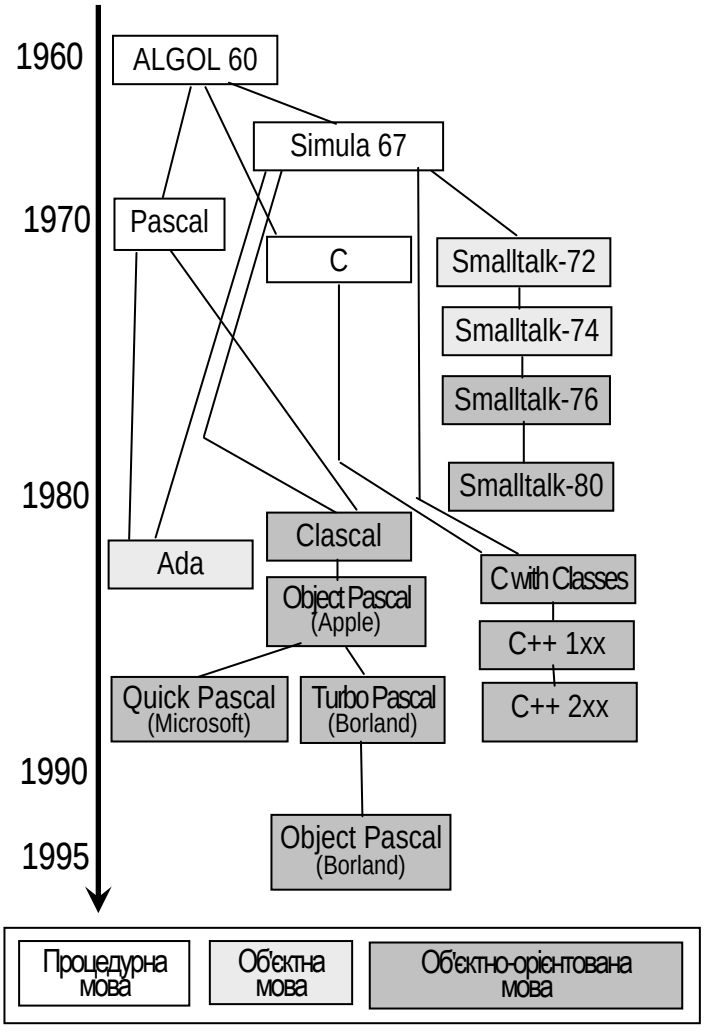


Рис. 8.1. Генеалогія мови Турбо Паскаль

При цьому треба додати по-перше, що ця мова є підмножиною об'єктно-орієнтованої мови Object Pascal – основи досить широко розповсюдженого середовища швидкої розробки додатків Delphi.

По-друге, мова Паскаль, яка лягла у основу мови ТП, розроблялася з наступними цілями:

❶ дозволяти систематично і точно виражати концепції і структури головних елементів галузі програмування;

❷ дозволяти здійснювати розробку програм систематично;

❸ показувати, що мова програмування з багатим набором гнучких структур даних (у протилежність мові С) і управляючих конструкцій може бути реалізована ефективно;

❹ включити у мову розвинуті засоби діагностики помилок, що є зручним засобом для навчання програмуванню і дослідженню програм.

Відповідно цього, у матеріал даного видання закладено завдання засобами мови Турбо Паскаль навести (рос.-представить) наступні головні елементи концепції мов програмування:

❶ моделі і структури даних; ❷ керуючі структури; ❸ порядок виконання керуючих структур;	❹ алгоритми й обчислення; ❺ концепцію розробки інтерфейсу програм й програмних систем; ❻ засоби дослідження розроблених програм.

Для читача цього видання ми повинні відзначити деякі ***дуже важливі обставини.***

❶ Так, як інтегроване середовище розробки (ІСР) Турбо Паскаль (ТП) розроблене для закордонного користувача (американців, англійців, французів та інших англомовних фахівців), то воно у своєї роботі у рядках програм розуміє ***тільки*** текст на англійській мові.

Рис. 8.2.
Блез Паскаль

❷ На екрані виводяться ті іншомовні тексти, які є строковими константами (тобто наборами символів, що середовищем Турбо Паскаль не аналізуються і виводяться у відповідності з їх кодом). Наприклад, український текст, що відокремлений (рос.- ограничен) з обох боків лапками (рос.-кавычками): "Українська мова" буде однозначно виведено на екран комп’ютера оператором процедури `WRITELN`('Українська мова') у такому ж вигляді (при наявності

відповідного україномовного драйвера[1]). При цьому треба розуміти, що ця процедура виводу даних на екран комп'ютера тільки відображує на екрані коди відповідних символів.

❸ У зв'язку з тим, що закордонні фірми ще не українізували програмні продукти (засоби) широкого походження (Windows, Word, Excel, Turbo Pascal та ін.), поширене походження поки мають російські терміни, які і будуть у подальшому використовуватися для широкого кола користувачів (у тому числі і закордонних студентів з СНГ).

Взагалі, навіть і сьогодні, не усім відомо, що інтегроване середовище розробки (IDE[2]) Турбо Паскаль (TURBO PASCAL) виробництва фірми Borland International (США) інтегрувало у собі вплив та внесок трьох талановитих європейських фахівців.

Блез Паскаль (Blaise Pascal) (1623 – 1662), француз за походженням, був видатним математиком, фізиком і філософом (рис. 8.2). У 1641 році він сконструював першу у світі машину, що суммувала числа. Це був перший цифровий калькулятор у світі, звався "Паскаліною" і був подібним до механічних калькуляторів, що використовувалися значно пізніше у 1940-х роках.

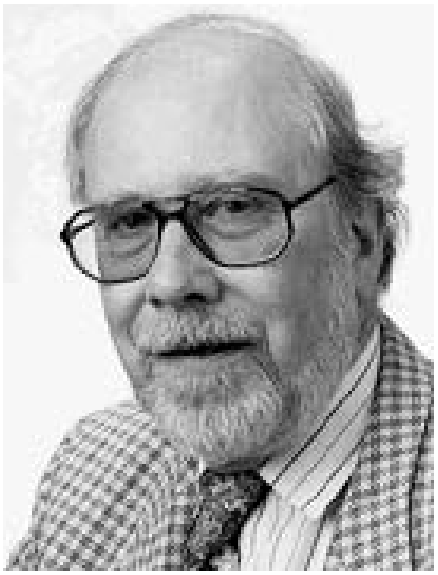


Рис. 8.3.
Ніклаус Вірт

Швейцарець Ніклаус Вірт (Niklaus Wirth), відомий усьому сучасному світові науковець, програміст і викладач(рис. 8.3), розробив нову алгоритмічну мову Паскаль у 1968 р. для цілей реалізації концепції підтримки виконання обчислень з чіткою типізацією і назвав її ім'ям Блеза Паскаля. Мова розроблялася Віртом спеціально з педагогічними цілями і з урахуванням простоти отримання відповідного машинного коду. Завдяки цьому створення такого компілятора не перевищує за трудомісткістю нормальну дипломну роботу випускника вищого навчального закладу. Окрім цього, сама мова вийшла дуже простою. Опис стандартного Паскалю займає біля 30-ти сторінок тексту.

З появою персональних комп'ютерів багато програмістів почали використовувати їх для

[1] Драйвер – програма, що управляє роботою зовнішнього пристрою (миша, клавіатура, принтер и т.і.); драйвер, як правило, є інтерфейсом поміж даним пристроєм и програмами уводу-виводу операційної системи. Найбільш характерним прикладом драйвера є програма KEYRUS.COM, котра кирилізує клавіатуру и монітор для забезпечення російськомовного інтерфейсу користувача з персональним комп'ютером .

[2] IDE – (Integrated Development Environment – Інтегроване середовище розробки (ICP)) – програмний засіб у якому співпрацюють разом редактор тексту, компілятор, завантажувач (рос. – загрузчик) та інщі компоненти роботи з програмою.

вирішення своїх прикладних задач. Але існуючі компілятори і транслятори[3] з мов ALGOL, FORTRAN та інших працювали досить повільно. Одному з французьких програмістів на ім'я Філіп Кан (Philippe Kahn), парижанину за походженням, з родини німецького інженера і французької кінорежисерки та датчанину Андерсу Хейльсбергу (Anders Hejlsberg) (рис. 8.4), це дуже заважало у роботі. Ф. Кан до того ж, свого часу вчився у університеті у "самого" Ніклауса Вірта. От вони і засіли поодинці за роботу і зробили у 1982 р. не тільки самий швидкий у світі компілятор з алгоритмічної мови Паскаль, а ще й дуже зручне середовище для роботи з ним, яке й назвали Turbo Pascal (Турбо, тобто "швидкий" Паскаль). Розповсюдження його було почато наприкінці 1983 року у США фірмою Ф. Кана Borland International по ціні US$49.95.


Рис. 8.4. Філіп Кан та Андерс Хейльсберг (1982 р.)

Низька ціна та висока якість зробила цей програмний продукт доступним мільйонам користувачів та професійних програмістів. Тільки за два роки було продано 300 000 екземплярів Турбо Паскалю, що перебільшило кількість усіх компіляторів з інших мов програмування для мікрокомп'ю-терів на той час! І зараз користуються повагою програмістів усього світу програмні продукти фірми Borland, такі, як Delphi останніх версій 5, 6 та 7, Paradox, Quattro Pro, Turbo Pascal, Borland C++, Sidekick, ObjectVision та інші. Більше ніж 50-ти мільйонів користувачів залишили свої електронні поштові адреси на сайті цієї фірми для сесійної розсилки новин та програмних апгрейдів. За 2002 рік фірмою було продано програмних продуктів на суму $US226 мільйонів, а 900 її штатних співробітників працюють у офісах країн США, Австралії, Франції, Англії, Німеччини, Італії, Швеції, Сінгапура та Данії.

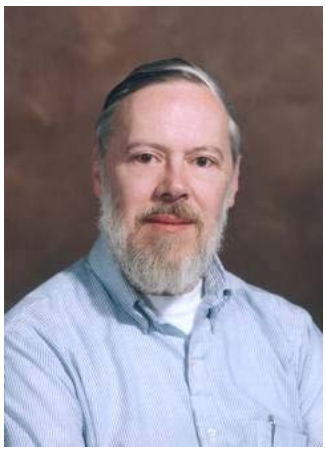


Рис. 8.5. Денніс Річі

Варто додати, що А. Хейльсберг є автором мови Delphi (1975 р.), а після запрошення до фірми Microsoft у 2000-му році він розробив мову C# (Сі шарп). За це та за особистий внесок у розвиток програмування у 2001 році його було нагороджено престижною премією відомого у світі видання "Dr. Dobb's Journal".

Цікаво, що компілятор мови Турбо Паскаль був написаний на мові С (Сі). А цю мову, розробив у 1972 році Денніс Річі (Dennis Ritchie), 31-річний американський спеціаліст з системного програмування

[3] Компілятор (транслятор) – програма яка переводить початковий текст програми, що написан програмістом на мові високого рівня у еквівалентну програму на машинній мові комп'ютера.

(рис. 8.5), котрий отримав ступінь бакалавра з прикладної математики у Гарвардському університеті і почав працювати у "Bell Telephone Laboratories" у 1968 році. Річі сподівався, що його нова мова стане в нагоді для програмування нової операційної системи UNIX. До речі, автор UNIX Кен Томсон (Ken Thompson), що був автором прародича С – мови В, вважався серед колег найкращім програмістом у світі!

Слід додати, що крім IDE Турбо Паскаль, співробітники фірми Borland розробили і однойменну алгоритмічну мову, яка є розширенням мови Паскаль Ніклауса Вірта. Це пов’язане з тим, що постійно спілкуючись з користувачами і слідкуючи за розвитком індустрії програмування вони додавали до стандарту мови усе нові і нові корисні можливості.

Тому Турбо Паскаль є зараз водночас і ***середовищем розробки програм*** і ***алгоритмічною мовою програмування***. У RAD Delphi подальші розширення кількості управляючих конструкцій, типів і структур даних призвели до появи мови ООП Object Pascal/Delphi, яка за своїми можливостями дорівнює таким відомим мовам, як C++, SmallTalk, Java, Python та ін.

8.2. Технологія роботи у середовищі Турбо Паскаль версії 7.0

Працюючи з комп'ютером користувач повинен постійно алгоритмізувати свої дії на рівні взаємодії з самим комп'ютером, мовами програмування і даними, які він повинен обробити.

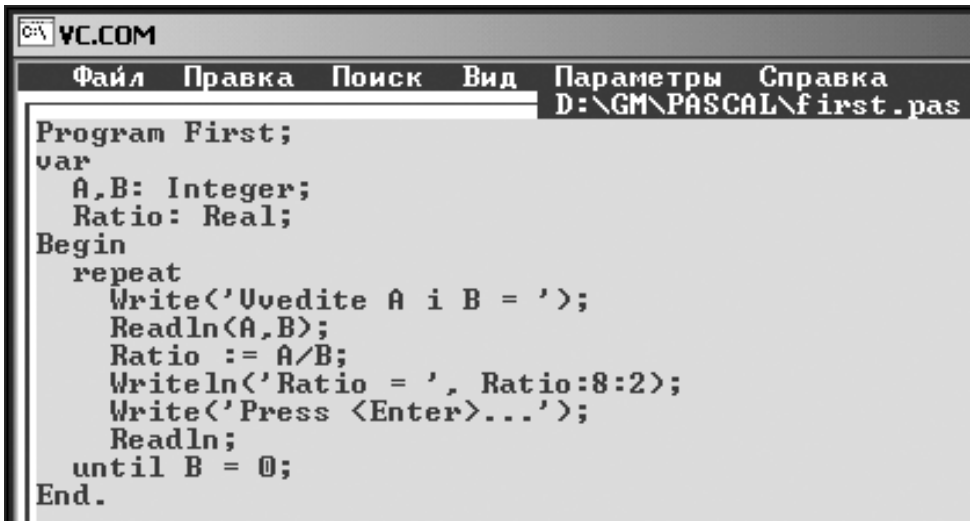


Рис. 8.6. Текст програми з іменем First, яка набиралася у файл-менеджері Volcov Commander у файл first.pas і шлях до якого має ім'я D:\gm\pascal\first.pas

З цієї точки зору, методично, робота майже аби якого програміста при реалізації програми на комп'ютері за допомогою мови програмування Турбо Паскаль складається з наступних декількох фаз, що виконуються постійно від програми до програми (рис. 8.6).

1. Вирішення задачі в її предметній області: алгебра, геометрія, електротехніка, гірнича справа і т.д. до одержання конкретних кінцевих формул результату (ТП за Вас цього не зробить!). Приклад: Обчислення факторіала *n!* без рекурсії.

2. Вибір вхідних і вихідних даних, які необхідні для рішення поставленої задачі.

3. Проходимо по розділі описів програми ТП, починаючи з модулів, що підключаються. Чи потрібний розділ Uses і які модулі ТП або користувача необхідно використовувати (а може необхідно їх додатково розробити).

4. Вибір необхідних структур даних мови ТП для забезпечення точності рішення задачі та алгоритмічної універсальності.

5. Розробка інтерфейсу програми (тобто організація уводу-виводу даних).

6. Вибір необхідних керуючих структур мови ТП (:=, `if, case, for, repeat, while`) і убудованих функцій (може потрібно розробити функцію користувача) для створення алгоритму рішення задачі.

7. Складання алгоритму[4] задачі.

8. Кодування алгоритму програми мовою ТП. Результатом цієї фази є готова програма, у якій програміст об'єднує алгоритм з обраними структурами даних. Недаремно Н. Вірт, свого часу, визначив, що "програма = структури даних + алгоритм".

9. Увід (рос. – ввод) тексту програми у комп'ютер на алгоритмічній мові Турбо Паскаль, що виконується за допомогою клавіатури у текстовий файл[5] з використанням будь якого тестового редактору (NotePad, Norton Commander, Turbo Pascal та ін.) (рис. 8.6). Файл повинен мати розширення .PAS.

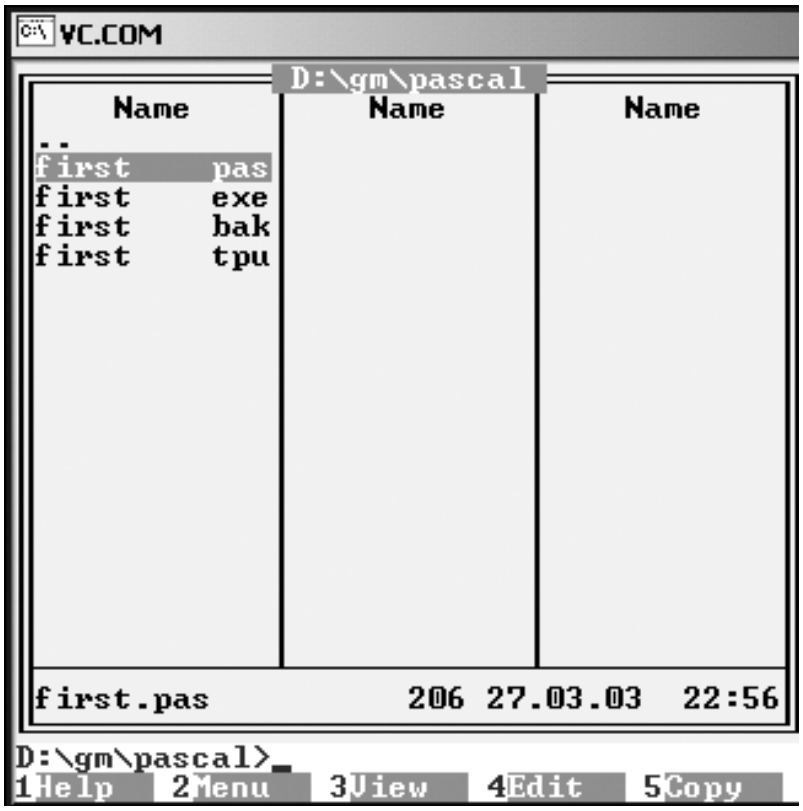


Рис. 8.7. Чотири форми існування програми, що міститься у файлах з ім'ям first і різними розширеннями у каталозі **`pascal`**:

❶ first.pas – початковий (рос. – исходный) файл (тобто текстовий);

❷ first.bak - резервна копія вихідного файлу;

❸ first.tpu – результат компіляції на диск вмісту first.pas – модуль Турбо Паскалю;

❹ first.exe - завантажувальний файл.

10. Виклик компілятора для компіляції тексту програми. Після його роботи ми отримуємо так званий об'єктний файл (з розширенням .TPU) (рис. 8.7). На етапі компіляції виконується перевірка правильності як побудови програми, так і використання конструкцій мови ТП.

11. Виклик компонувальника (рос. – компоновщика) для настройки адресних частин програмних об'єктів по місцях

[4] Алгоритм – послідовність дій чи кроків, виконання яких дозволяє вирішити конкретну задачу.
[5] Файл – пойменована область на диску, яка містить цілістну сукупність даних.

у оперативної пам'яті (ОП). Після його роботи ми отримуємо третій файл, що потім виконується (рос. – исполняется), з розши-ренням **.**EXE, який повністю готовий для завантаження у ОП і виконання. ***(Примітка: Усі чотири файли мають однакові імена але різні розширення і містяться на жорсткому диску)*** (рис. 8.7)).

12. Якщо програма написана і уведена правильно, а також не потребує подальшого відлагодження[6], EXE-файл завантажується програмою–завантажувачем (рос. - загрузчиком) у оперативну пам'ять для її виконання. Завантажена у ОП програма під час своєї роботи інтерактивно[7] спілкується з користувачем, відповідно до інтерфейсу[8], який був запрограмований розробником.

13. Дослідження програми засобами ІСР Турбо Паскаль.

10. Вивантаження програми по завершенні її роботи з оперативної пам'яті відповідними системними утилітами[9]. Результати її роботи залишаються або на екрані комп'ютера, або у відповідному файлі на жорсткому диску.

Філіп Кан, автор Турбо Паскалю, враховуючи такий багатоетапний цикл роботи програміста, вирішив у своєму інтегрованому середовищі розробки багато складних задач. Серед головних треба назвати такі.

Програму** MyProg **запускаємо натисненням клавіш* Ctrl+F9 *у вікні редактора Турбо Паскаль, де був набраний її текст. Результат роботи програми виводиться у вікні екрану DOS. Перехід до вікна DOS – натиснення клавіш* Alt+F5*, до вікна редактора – натиснення будь-якої клавіші (див. рис.).

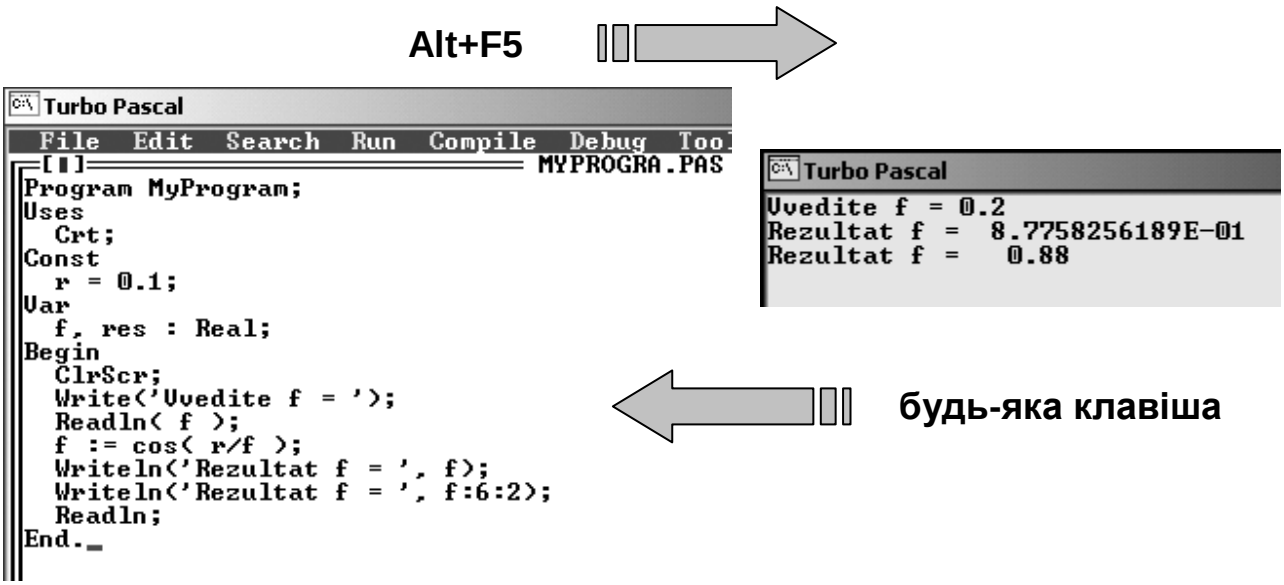


Рис. 8.8. Екрани, з якими працює користувач при виконанні своєї програми (у даному випадку – MyProgram)

[6] Відладка – пошук помилок у алгоритмі і тексті програми.

[7] Інтерактивний (діалоговий) – режим, у котрому користувач задає програмі команди і уводить данні поки вона працює. Такий режим передбачає обмін текстовими командами (запитами) і відповідями (запрошеннями).

[8] Інтерфейс – засіб сполучення (рос.-сопряжение) двох або більше різних систем, що забезпечує їх логічну або фізичну взаємодію.

[9] Утилітою зветься будь-яка системна спеціальна програма, яка спрямована на виконання визначеного завдання з обслуговування операційної системи.

На рисунку 8.8. на першому (синьому, див зліва на рис. 8.8) екрані розташована програма користувача, яку він запускає на виконання комп'ютером натисненням сполучення клавіш **Alt+F5**. На другому (чорному) екрані, здійснюється інтерактивне спілкування користувача з програмою відповідно до розробленого інтерфейсу, тобто увід даних у програму та вивід інформації від програми і комп'ютера.

Після виконання програми процес закінчується повертанням у (синій) екран програми, якщо у кінці не застосовано виклик процедури READLN без параметрів. У випадку застосування виклику процедури READLN, робота програми затримується у чорному екрані, де можна розглядати результати її роботи. Повернення до синього екрану здійснюється натисненням клавіші Enter (увід).

Процес підготовки програми до запуску, тобто увід її тексту у редакторі тексту, трансляція цього тексту у об'єктний файл (obj-файл), перетворення об'єктного файла на той, що буде виконуватися (exe–файл) та запуск цього файла на виконання – провадиться цілком послідовно і конвеєрно у рамках IDE Турбо Паскаль. Останні три етапи виконуються взагалі однією командою RUN або сумісним натисненням клавіш **Ctrl+F9.**

Процес налагодження і дослідження роботи програми в IDE Турбо Паскаль зроблено максимально зручним, що дозволяє установлювати контрольні точки, виконувати програму пооператорно, контролювати вміст (рос.–содержание) змінних (тобто які дані у них є і як вони змінюються у процесі роботи) та багато іншого. Меню IDE ТП та його команди описані у Додатку 1.

Перша версія IDE ТП була зроблена для використовування на персональних комп’ютерах з операційною системою[10] (ОС) MS DOS, яка була розрахована на одного користувача. Тому її виклик до роботи виконувався досить просто. Потрібно було лише знайти на жорсткому диску підкаталог BIN каталогу TP70, вказати ім'я TURBO або TURBO.EXE у командній стрічці та натиснути на клавішу Enter (рис. 8.9).

```
C:\>cd tp70
C:\TP70>cd bin
C:\TP70\BIN>turbo
```

Рис. 8.9. Команди ОС MS DOS для виклику середовища ТП

Зараз же, у комп’ютерних класах комп'ютери об’єднуються у мережі під управлінням ОС Windows NT або Windows 2000. Це накладає додаткові умови на процес роботи студентів з IDE ТП.

[10] Операційна система (ОС) - сукупність програмних засобів, що забезпечують управління апаратними ресурсами комп'ютера та взаємодію програмних процесів с апаратурою, іншими процесами і користувачем. ОС виконує слідуючи функції: управління пам'яттю, увідом-вивідом, файловою системою, взаємодією процесів, діспетчеризацію процесів, захист, облік використованих ресурсів, обробку командної мови.

Логічні[11] диски C: та D: для запису на них будь якої інформації та доступу користувачеві взагалі закриті і на них розташовані системні програми та офісні додатки[12]. Тому, для виклику ТП потрібно зробити наступні кроки.

1. Натиснути клавіші **Ctrl+Alt+Del**, для того, щоб увійти у виділений користувачу операційний простір на жорсткому диску, на підставі установок системного адміністратора мережі. Операційний простір – це ресурси системи, у тому числі і потрібні для роботи програмні засоби (файл-менеджери, системи програмування, редактори (графічні, текстові) та ін).

2. Після натиснення сукупності клавіш **Ctrl+Alt+Del** потрібно ввести ім'я користувача та його пароль у перші дві стрічки діалогового вікна операційної системи (рис. 8.10). Пароль (Password) відображується зірочками для того, щоб сторонній не зміг його впізнати. Перехід з вікна User Name до вікна Password можна робити або курсором миши, або клавішею Tab.

3. Після відкриття екрану з зображенням робочого столу ОС Windows потрібно навести курсор миші на кнопку Start (Пуск) і натиснути її. Потім розкрити меню Program (Програми), знайти стрічку з написом Turbo Pascal і клацнути по неї лівою кнопкою миші.

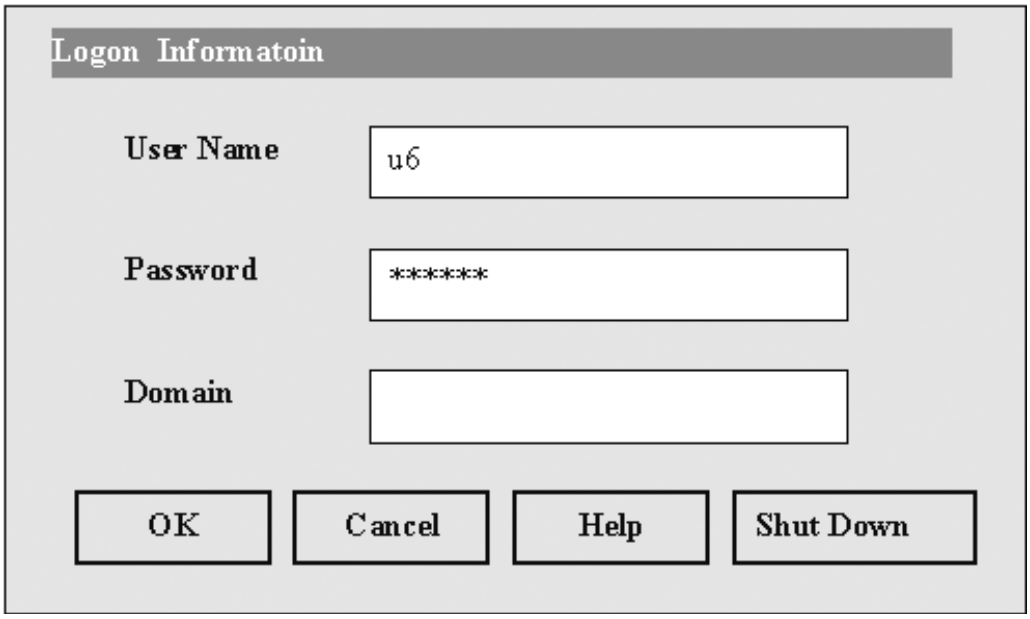


Рис. 8.10. Діалогове вікно для входу у ОС Windows NT

4. У розкритому вікні ТП у меню **File** (рис. 8.11) клацнути по команді **Change dir** (Змінити каталог). У діалоговому вікні, що відкриється, вибрати потрібний логічний диск (G:, Z: чи якийсь інший) і каталог у якому знаходяться програми ТП з розширеннями **.**PAS, що належать користувачу (рис. 8.12).

[11] Логічний диск – фізичний простір на диску, до якого користувач звертається по відомому операційній системі імені (A: – гнучкий диск, дискета, C: D:, E:, …, Z: – робочі ділянки на жорсткому диску).

[12] Додаток – программа чи група програм, яки зроблені для кінцевих користувачів. Сюди входять текстові процесори, електронні таблиці, програми баз даних та ін.

Після цього можна відкривати нове вікно для набору тексту програми (Команда **New** з меню **File**) або редагувати та запускати для виконання якусь з раніш уведених програм (Команда **Open** з меню **File**).

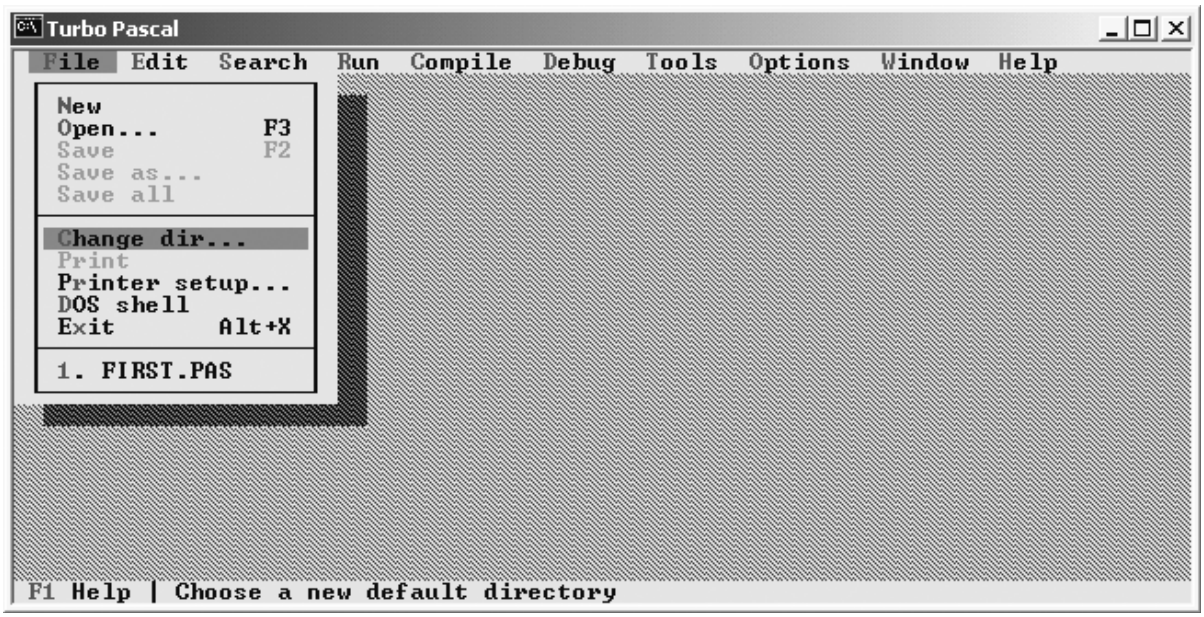


Рис. 8.11. Розкрите меню **File** з командами роботи з файлами та командою **Change dir**

Рис. 8.12. Діалогове вікно команди **Change dir,** у якому користувач вибирає потрібний йому каталог (директорій)

Найкоротшою програмою (яка щось діє !) у ТП є така:

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

Ця програма виведе на екран текст: "Hello, world!".

Вправи

1. Завантажити Турбо Паскаль, активізувати вікно (**Edit**) редагування програм і набрати довільний текст у вікні редагування. Набраний текст записати на диск у файл з ім'ям NAME1.PAS.
2. Очистити вікно редагування для роботи з новим файлом. Набрати довільний текст. Набраному тексту привласнити ім'я NAME2.PAS (перейменувати файл). Далі записати файл NAME2.PAS на диск.
3. Переглянути поточний каталог (диск) і знайти файл з ім'ям NAME1.PAS. Завантажити цей файл у вікно редагування. Відредагувати файл (щось змінити в тексті файлу). Далі очистити вікно редагування для роботи з новим файлом.
4. Завантажити у вікно редагування файл NAME2.PAS. Далі записати цей файл в інший каталог поточного диска. Перейменувати файл NAME2.PAS у файл NAME3.PAS і записати останній на диск.
5. Переглянути історію роботи з файлами. Знайти передостанній файл, з яким працювали за останнім часом і завантажити його у вікно для редагування.
6. Тимчасово вийти з Турбо Паскалю в MS-DOS і повернуться в систему Турбо Паскалю.
7. Завантажити у вікно редагування файл NAME1.PAS, призвести редагування тексту файлу і вийти із середовища Турбо Паскалю (тобто закінчити роботу).

8.3. Будівельні блоки (базові елементи) програм на мові ТП

Програма на мові ТП, як реалізація відповідного алгоритму, містить сукупність об'єктів (лексем[13]), що дозволяють комп'ютеру за допомогою транслятора з мови ТП зрозуміти, що потрібно діяти для виконання завдання програміста.

Програма, яка написана на мові ТП поділяється на ***лексеми*** і ***розподілювачі*** (рос. – разделители). До розподілювачів, котрі відділяють одну лексему від іншої, відносяться пробіли (рос. – пробелы), або символ "**;**" та коментарі (рис. 8.13). Таким чином, дві поряд стоячі лексеми повинні відділятися одним чи більшою кількістю розподілювачів.

Об'єкти програми (лексеми) розподіляються на групи. До першої групи відносяться елементи алфавіту мови ТП. Він містить наступні букви і цифри у восьмибітному коді ASCII[14] (табл.8.1) (див. Додаток 6):

Таблиця 8.1

Символи алфавіту мови ТП.

Великі букви англійської мови	Від A до Z
Малі букви англійської мови	Від a до z
Десятинні (арабські) цифри	Від 0 до 9
Шістнадцятирічні цифри	Від 0 до 9 та від A до F

Примітка. ТП не відрізняє великих букв від малих, тому він однаково зрозуміє слова **`BEGIN`** та **`begin`**.

Приклад (рис.8.13). Лексеми мови Турбо Паскаль:

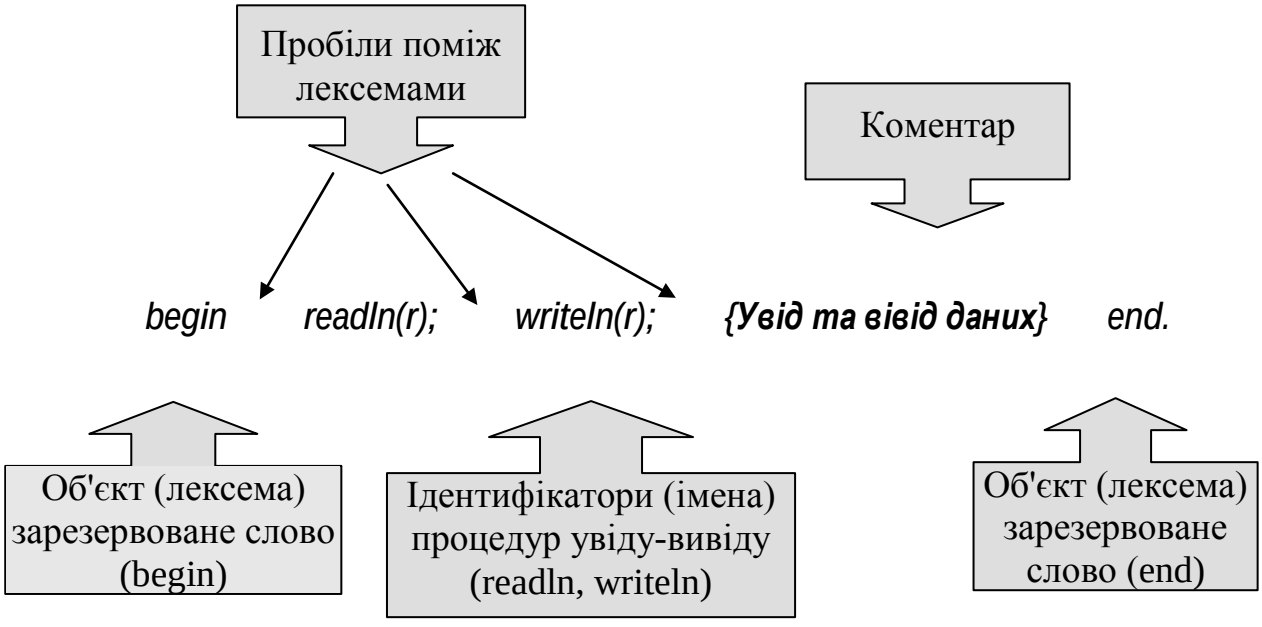


Рис. 8.13. Складаючи рядку кода (текста) програми ТП

[13] Лексема – мінімальні одиниці значень текста у програмі.

[14] ASCII (American Standard Code for Interchanging Information) – вісьмибітний код, якій відображає усі потрібні для передачі інформаційні у комп'ютері символи. Сюди входять букви, цифри, спеціальні символи та ін.

Другу групу складають спеціальні символи. Вони можуть бути простими та складеними (табл. 8.2, 8.3):

Таблиця 8.2

Спеціальні прості (одиночні) символи мови ТП

Прості одиночні символи		Прості одиночні символи	
Назва символу	Символ	Назва символу	Символ
Додавання	+	Відкриваюча кругла дужка	(
Віднімання	-	Закриваюча кругла дужка	)
Множення	*	Двокрапка	:
Ділення	/	Крапка з комою	;
Рівність	=	Спецсимвол	^
Менше	<		@
Більше	>	Відкриваюча фігурна дужка	{
Відкриваюча квадратна дужка	[	Закриваюча фігурна дужка	}
Закриваюча квадратна дужка	]	Долар	$
Крапка	.		#
Кома	,		

Таблиця 8.3

Спеціальні складені символи мови ТП

Складені символи	
Назва символу	Символ
Менше чи дорівнює	**<=**
Більше чи дорівнює	**>=**
Привласнити	**:=**
Діапазон	**..**
Відкриваюча дужка з зірочкою	**(***
Закриваюча дужка з зірочкою	***)**
Відкриваюча дужка з крапкою	**(.**
Закриваюча дужка з крапкою	**.)**

До третьої групи входять так звані "зарезервовані" чи службові слова. Ця назва вказує на те, що ці слова потрібно використовувати тільки у одному значенні, котре їм призначено раз і назавжди у мові ТП. Тому програміст не має дозволу використовувати їх по іншому призначенню. У таблиці 8.4 приведені всі зарезервовані слова мови ТП.

До четвертої групи входять так звані "ідентифікатори[15]", що уявляють собою імена[16] об'єктів програми, які уводить користувач.

Таблиця 8.4

Зарезервовані (службові) 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	

Ідентифікатори виступають іменами констант, типів даних, змінних (об'єктів для зберігання даних різних типів), процедур, модулів, програм і полів записів. Ідентифікатор може мати довільну довжину, але для ТП значення мають тільки перші 63 символи. Решта символів значення не має, але зберігається. Ідентифікатор повинен починатися з літери або знака підкреслення (_) і ***не може*** містити розподілювачів (пробілів, коментарів). Після першого символу ідентифікатора можна використовувати літери, цифри, символи підкреслення (доречи, у кодуванні ASCII значення ***коду цього символу:*** десятинне – 95, шістнадцятирічне – $5F, тобто у байті розміщуються слідуючи вісім бітів *0101 1111,* див. Додаток 6). І у зарезервованих словах, у ідентифікаторах можна використовувати як строкові, так і прописні літери (компілятор їх не розрізнює). Для уточнення деяких ідентифікаторів потрібне посилання на той ідентифікатор, від якого перший залежить. Наприклад, для уточнення ідентифікатора `Ident` за допомогою ідентифікатора `UnitName` потрібно писати `UnitName.Ident`[17] у рамках існуючого контексту[18].

Наведемо декілька прикладів ідентифікаторів:

```
Writeln          Dos_Exec          Int22String5
Exit             Crt.Windows       NameOfTheGame
Real2String      F1.txt            System.MemAvail
                 First_Name
```

[15] Ідентифікатор – неподільна (рос. – неделимая) послідовність символів алфавіту, яка утворює (рос. – образовывает) ім'я об'єкту, що використовується.

[16] Ім'я є символічним уявленням "сутності" (рос. – сущность). Сутність може являтися об'єктом або деякою дією, що виконується. Сутність може мати велику кількість найменувань. Кожне ім'я має сенс тільки в границях деякого іменного контексту.

[17] Зв'язування (рос. – связывание) є відображенням імені на відповідну сутність. В рамках контексту ім'я може мати не більше одного зв'язування. Можна посилатися за допомогою іншого імені на контекст. Структуроване або складене (рос. – составное) ім'я формується у вигляді комбінації імені контексту з ім'ям сутності у цьому контексті, які розподілені крапкою.

[18] Контекст – фрагмент документу, у межах якого можна уяснити значення окремого слова або об'єкту, які в нього входять. Тільки у контексті слово або об’єкт отримують конкретне значення.

Таким чином, при реалізації алгоритмів на мові ТП важливо пам'ятати наступне.

❶ Імена (ідентифікатори) не можуть починатися з цифри і містити у собі російські чи українські літери та символ пробілу.

❷ У іменах допускається використання великих (прописних) та малих англійських літер, цифр від 0 до 9 та символу підкреслення (_), що має ASCII-код 95 (Додаток 6).

❸ Кількість символів у ідентифікаторах (іменах) обмежується редактором ТП до 127, а сам компілятор ТП відрізняє один ідентифікатор від іншого по першим 63–м символам.

❹ Для підвищення змістовності ідентифікаторів у мові ТП прийнято конструювати їх за принципом, коли декілька слів пишуться одне за одним, але кожне з великої букви.

```
ZarazWyBachyteIdentificatorDlaZberiganniaRechovoyiZminnoi:=0.265;
ZarazWyBachyteIdentificatorDlaZberiganniaRechovoyiZminnoi1:=54.3;
```

Вищенаведені ідентифікатори розрізнюються компілятором ТП, оскільки перший містить 60 символів, а другий – 61, з яких перші 60 співпадають.

```
Zaraz_Wy_Bachyte_Identificator_Rechovoyi_Zminnoyi := 0.265;
Zaraz_Wy_Bachyte_Identificator_Rechovoyi_Zminnoyi_1 := 1.3;
```

Нагадуємо, що вищенаведені ідентифікатори містять у середині символи підкреслювання.

Вправи

1. Поясніть поняття «Базові елементи мови Турбо Паскаль». Навести приклади.

2. Поясніть поняття: алфавіт мови програмування, ключове слово, мінімальна значеннєва одиниця мови. Яке призначення цих понять у тексті програми? Навести приклади.

3. Поясніть поняття ідентифікатора. Сформулюйте правила написання ідентифікатора. Поясніть призначення ідентифікатора в програмі.

4. Що таке розподілювач (роздільник) у тексті програми, для чого служать роздільники? Навести приклади.

5. Поясніть поняття «коментар» та його призначення в тексті програми.

6. Чим прості символи відрізняються від складених символів?

7. Якими символами може починатися ідентифікатор, а якими не може?

8. Яка може бути мінімальна кількість символів у ідентифікаторі?

9. Яка може бути максимальна кількість символів у ідентифікаторі, що її розрізняє транслятор?

8.4. Константи, змінні та їх типи

Усі дані (числа, значення обчислених виразів, тексти, результати порівнянь та ін.) у програмі інтерпретуються як ***константи*** і ***змінні***.

Константи у процесі роботи програми ніколи не змінюють своє значення. Будь яке число, котре з'являється у програмі у звичайному вигляді (наприклад: 6.0 або 20) зветься ***константою***. Величина, котрій привласнюється найменування і котра може приймати у процесі обчислень різні числові значення, зветься ***змінною***. У відповідності до того, які дані розміщуються у змінних, ті можуть мати різні ***типи***, що визначає припустимі з ними операції (табл. 8.5).

Таблиця 8.5

Найбільш корисні типи даних і операції, що припустимі до них

Найменування типу	**Спосіб представлення (приклади даних відповідного типу)**	**Допустимі до цього типу даних операції**
`Real` (дійсні числа)	1.0 0.0012 1E-6 -2.639E4	+,–,/,*.
`Integer` (цілі числа)	1 -56 2892	+,–, *, DIV, MOD.
`Char` (символи)	A, 'A', 1, '1'	операції порівняння, зціплення (+) конкатенації
`String` (стрічки)	'Type mismutch!' 'Українська мова' 'Не забудьте ввести!'	операції порівняння, зціплення (+) конкатенації, перетворення
`Boolean` (логічні значення)	false, true (неправда), (правда)	операції порівняння, логічні операції
`Array` (масиви)	Послідовність значень вищевказаних чи інших типів	всі операції, що допустимі для відповідних типів

Примітка: 1) До операцій ***порівняння*** належать наступні: = - дорівнює; <> - не дорівнює; > - більше; < - менше; >= - більше або дорівнює, <= - менше або дорівнює.

2) ***Логічні*** операції включають: OR – логічне додавання, AND – логічне множення, NOT – операція заперечення (рос. – отрицания), XOR – виключне АБО (рос. – или).

Цілі і дійсні константи розрізняються наявність або відсутністю десятинної точки. Так, 3 – ціла константа, а 3.0 або 3.000000 – дійсні константи і спосіб

запису їх у пам'яті ПК та арифметичні операції, що з ними можна виконувати абсолютно різні.

Щоб спростити запис дуже великих та дуже малих чисел, використовується наступний спосіб. Після дійсної константи розташовується літера E та одно– або двозначне ціле число з відповідним знаком (+ чи –). Такий запис вказує на те, що початкова константа повинна бути помножена на 10 у вказаній степені.

Наприклад, константи:

+15.016 (можна записати у вигляді 1.5016E01 або 1.5016E+01);

5.0E+6 (можна записати у вигляді 5000000.0);

0.0000173 (можна записати у вигляді 1.73e–5).

Термін "змінна" використовується у мові ТП для позначення величини, звернення до якої призводиться через її найменування і котра може приймати різні значення, а не обмежена якимось одним.

Змінні свої значення і стан можуть змінювати багаторазово. Як відзначалося раніше вони мають ***ім'я*** і ***тип***. Ці типи (цілі, дійсні, логічні, символьні, текстові та ін.) користувач повинен описати на початку програми за допомогою спеціальних зарезервованих слів (див табл. 8.5).

Тип даних визначає:

1. Спосіб представлення елементів сукупності (цифровий, текстовий, логічний та ін.);
2. Множину (рос. – множество) допустимих значень;
3. Множину припустимих операцій, котрі використовуються до даного типу.

У таблиці 8.6 наведені приклади опису констант і змінних при використанні їх у програмах на мові Турбо Паскаль.

Таблиця 8.6

Опис змінних і констант у програмах ТП

Опис констант (перед знаком "=" вказується ім'я константи)	Опис змінних (після двокрапки позначений тип змінної)
`Const`	`Var`
`Rand = 5.2E-5;`	`R,t,p : real;`
`Norma = - 54;`	`I,j,k : integer`
`T = FALSE;`	`a : Boolean;`
`Symbol='R';`	`c : char;`
`My_Name = 'Alex';`	`S1 : string;`

Змінні і константи усіх типів використовуються у ***виразах*** (рос. – выражениях).

Вирази задають порядок виконання дій над елементами даних і складаються з операндів (констант, змінних, звернень до функцій), круглих дужок та знаків операцій. Операції, у свою чергу, визначають дії, котрі треба виконувати з операндами (дивись таблицю 8.7).

Таблиця 8.7

Приклади реальних виразів і їх запису на ТП

Тип виразу і змінних у ньому	**Математичний (чи реальний) вираз**	**Реалізація на Турбо Паскалі**
Дійсний	$6*\sqrt[4]{\frac{a^2\cdot(\frac{b}{c})}{\frac{b+c}{2}}}$	`6*sqrt(sqrt((a*a*(b/c))/ ((b+c)/2))`
Цілий	$\frac{a}{2}\cdot(x^2\cdot\frac{a+b}{c})$	`a div 2 * (sqr(x) *((a+b) div c)`
Логічний	$\neg(A\vee B)\wedge(C\vee\neg D)$	`NOT (A OR B) AND (C OR NOT D)`
Символьний	Д + А = ДА	‘Д’ + ‘А’ = ‘ДА’
Строковий	Сьогодні вівторок	‘Сьогодні вівторок’
Примітка: `sqrt(x)` – функція добування квадратного кореня зі значення `x`; `sqr(x)` – зведення (рос. – возведение) значення *x* у другу степінь; `div` – математична операція ділення цілковито одного цілого числа на інше; `NOT, OR, AND` – логічні операції (відповідно: заперечення (рос. - отрицания), логічне підсумовування та логічне множення).		

Операції, що застосовуються до операндів бувають унарні (одномісцеві) та бінарні (двомісцеві) (див. таблицю 8.8).

Таблиця 8.8

Одномісні та двомісні операції у ТП

Тип виразу і змінних у ньому	**Тип операції**	**Математичний (чи реальний) вираз**	**Реалізація на Турбо Паскалі**
Дійсний, цілий	Одномісна	– 1.35; –30	– 1.35; –30
	Двомісна	$5.1\cdot 25$	5.1 * 25
Логічний	Одномісна	$\neg D$	`NOT D`
	Двомісна	$A\vee B$	`A OR B`

Термін "вираз" використовується у мові ТП у тому ж значенні, що у звичайному математичному запису. Вираз може конструюватися з констант, змінних і функцій (вбудованих або створених користувачем), що об'єднуються поміж собою символами операцій, комами та дужками.

Дійсні і цілі числа можна "змішувати" у дійсних виразах і не можна у цілочисельних виразах., бо арифметика цих двох типів базується за абсолютно різними способами.

Для дійсних типів `Real` кількість цифр, що може зберігатися у пам'яті комп'ютера дорівнює 11-12-ти. Тому арифметичні операції, до того ж, з дробами кінцевої довжини не завжди точно відповідають звичайним правилам арифметики. Щоб це зрозуміти розглянемо наступний вираз:

0.40000000000 + 123456789778.0 – 123456789777.0

11 цифр 11 цифр 11 цифр

Оскільки операції будуть виконуватися зліва на право, то результат додавання з точністю 11 знаків буде дорівнювати 123456789778.0 і 0.4 безповоротно загублене. Після віднімання 123456789777.0 кінцевий результат буде дорівнювати 1.00000000000. Але, якщо вираз записати з дужками

0.40000000000 + (12345678978.0 – 12345678977.0)

то виконання операції віднімання попередньо у дужках , а потім тільки додавання першого числа дасть вірний результат 1.4.

Невірний порядок виконання дій може призвести не тільки до втрати точності, але взагалі до повної невдачі. Припустимо, необхідно обчислити вираз `A*B/C`, у якому величини `A`, `B` і `C` усі рівні 10^{30}. Спочатку буде виконано множення, яке дасть 10^{60}, що дуже велике для тієї величини, що може міститься у змінній типу `real`. Тому система ТП виведе попередження:

```
Error 205: Floating point overflow.
```

(Помилка 205: Переповнення числа з плаваючою точкою)

і вирішення задачі буде припинено. Але, якщо ті ж самі дані описати типом `extended`, який вміщує значно більші значення, обчислення будуть проведені коректно. На рисунку 8.14 показано як взаємодіють цілі числа з різними типами операцій у виразах, які привласнюються змінним різних типів.

```
var  x, y : real;
     i, j : integer;
Begin
  x := 5 / 2; {операція дійсного ділення дасть дійсне значення 2.5, а
оператор привласнення занесе його у дійсну змінну x}
  y := 5 div 2; {операція цілочисельного ділення за умови наявності
зліва від оператора привласнення дійсної змінної y, компілятором
замінюється оператором дійсного ділення та у змінну y заноситься дійсне
значення 2.5}
  i := 5 div 2; {операція цілочисельного ділення дасть ціле
значення 2, що буде занесено у цілочисельну змінну i}
  j := 5 / 2; {операція дійсного ділення не буде виконана і
компілятор видасть наступну діагностику }
```

```
Error 26: Type mismatch.
```

{що переводиться, як неспівпадання типів справа і зліва від оператора привласнення}

Рисунок 8.14. Взаємодія типів операцій з типами змінних

При діленні цілих чисел виникають свої особливі проблеми. Коли ціле число ділиться на ціле, то результат у загальному випадку буде дрібним (рос. – дробным) числом. Але тому що результат повинен бути цілим, число, що буде отримане, буде перетворено на ціле шляхом відкидання дробової частини.

5/3 = 1 (5 ділиться на 3 цілковито! У ТП це запишеться 5 div 3),

6/7 = 0 (6 ділиться на 7 цілковито! У ТП це запишеться 6 div 7) і т.д.

Однак потрібно добре розуміти особливості обчислень окремо з дійсними і окремо з цілими значеннями, а також їх взаємодію зі змінними, у які вони повинні бути розміщені. Краще за все це можна уяснити з приклада, наведеного на рис. 8.14.

Несподіванки можуть робитися і з дійсними числами. Розглянемо наступну суму:

1.0 / 3.0 + 1.0 / 3.0 + 1.0 / 3.0.

Результат ділення 1.0 / 3.0 дорівнює з точністю до 11 знаків 0.33333333333. Загальна сума буде дорівнювати 0.99999999999, а ніяк не 1.00000000000!

Таким чином потрібно дуже обережно конструювати вирази з даними і операціями з погляду на їх типи.

Вправи

1.Запишіть мовою ТП слідуючи вирази з початку з припущенням що усі змінні дійсні (real), а потім, що усі змінні цілі (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.Виконайте дії на сумісність типів:

а) X:=45; Y:=578; Z:=X+Y. Який може бути тип величин X, Y і Z?

б) X:=6546; Y: =5655.665; Z:=X-Y. Який може бути тип величин X, Y і Z?

в) X:=65; Y:=5; Z:=X/Y. Який може бути тип величин X, Y і Z?

г) X=567687687; Y=5.89; Z=X+Y. Який може бути тип величин X, Y і Z?

8.5. Загальна структура програм на Турбо Паскалі

Програма на мові ТП складається з заголовку блока[19] і закінчується точкою. Але мова ТП дозволяє максимально спрощувати запис необхідних дій, тому найкоротша програма (яка нічого не робить) виглядає у формі блока так.

```
Begin
End.
```

А якщо навести приклад програми у вигляді, що наводиться у всіх керівництвах по мовам програмування, де вона завжди привітає навколишній мир, то треба написати так.

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

Після натиснення клавіш **Ctrl+F9** у середовищі редактора ТП комп'ютер виконає програму і миттєво повернеться назад у середовище редактора. Щоб побачити результат роботи програми треба натиснути сукупність клавіш **Alt+F5**. Це дозволить побачити на екрані монітору цю вельми відому фразу. Натиснення будь-якої клавіші поверне Вас зворотно у редактор ТП.

Загальна (повна) структура програми на ТП приведена на рис. 8.15.

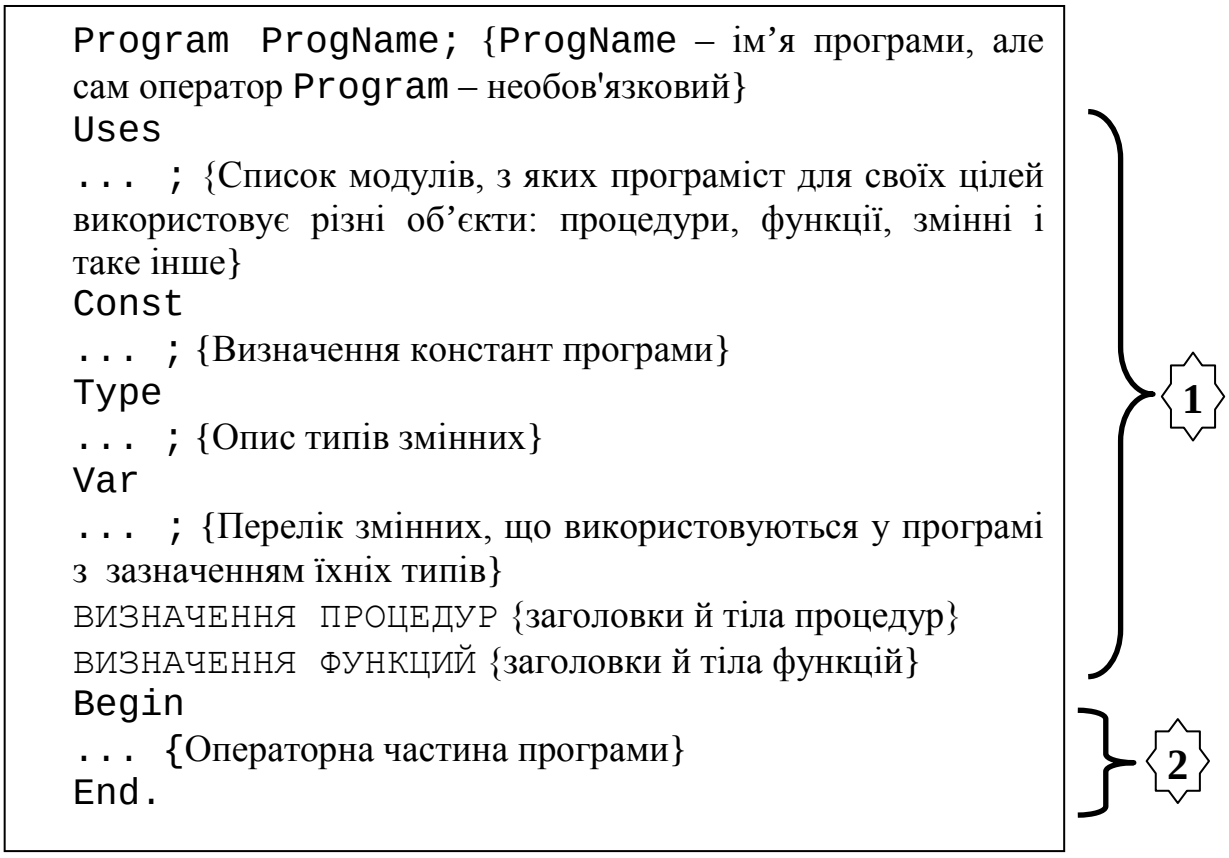


Рисунок 8.15. Загальна структура програми на мові ТП
(1 – розділ опису даних, 2 – операторна частина (розділ операторів)

[19] Блок – смислова частина програми, що завершується точкою. Уся інформація, що розташована після точки останнього зарезервованого слова end середовищем ТП ***до уваги не приймається!!!***

Таким чином, програма починається з заголовка `Program` і імені програми, а блок вміщує розділ описів і розділ операторів. Розділ операторів уявляє собою так званий складений оператор (операторні дужки `Begin End`), який вміщує послідовність операторів, що виконуються і розподілені точкою з комою (;).

Можна покращити програму `'Hello, World!'`, додав у її кінці оператор процедури `Readln` без параметрів! Він зупиняє роботу програми у екрані результатів і чекає натиснення клавіши Enter для завершення роботи програми:

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

Але, коли програміст реалізує складну програму вона повинна відповідати стандартам мови і включає дві головні частини: опис даних і операторну частину, яка описує послідовність дій для виконання алгоритму програми. Частина опису даних складається зі стандартних розділів, які Ви можете вводити за необхідністю (рис. 8.16).

```
Real A,B,C,D;
Begin
  B := 2.0;
    C := 7;
  A := B + C;
    D := b + c;
  Writeln('Resultat=', A, D)
End.
```

Рисунок 8.16. Програма обчислення виразів A=B+C та D=B+C. (B=2.0; C=7)

Вправи

1. Яка є повна структура програми на мові Турбо Паскаль?

2. Які розділи входять у програму?

3. Чи можна продовжувати операторну частину програми після кінця блоку (`End.`)?

4. Запишіть у вигляді програм ТП вирази з попереднього розділу та виконайте їх на комп'ютері.

8.6. Інтерфейс програми користувача. Процедури введення-виводу

Коли програма завантажена у комп'ютер і починає виконуватися, настає етап взаємодії трьох істот (рос. – сторон, существ), тобто спілкування наступних сторін:

❶ користувача програми, яку виконує комп'ютер;

❷ розробника програми, якою користується користувач;

❸ комп'ютера, який забезпечує діалог, запрограмований розробником.

При навчанні програмуванню найпростішим випадком є ситуація, коли студент програмує якийсь алгоритм, виступаючи при цьому одноразово і програмістом і користувачем. Тоді він повинен сам прогнозувати ситуації взаємодії користувача (тобто себе) з комп'ютером. Останній повинен організувати запроси на виконання деяких дій, а також їх виконувати (рис. 8.17).

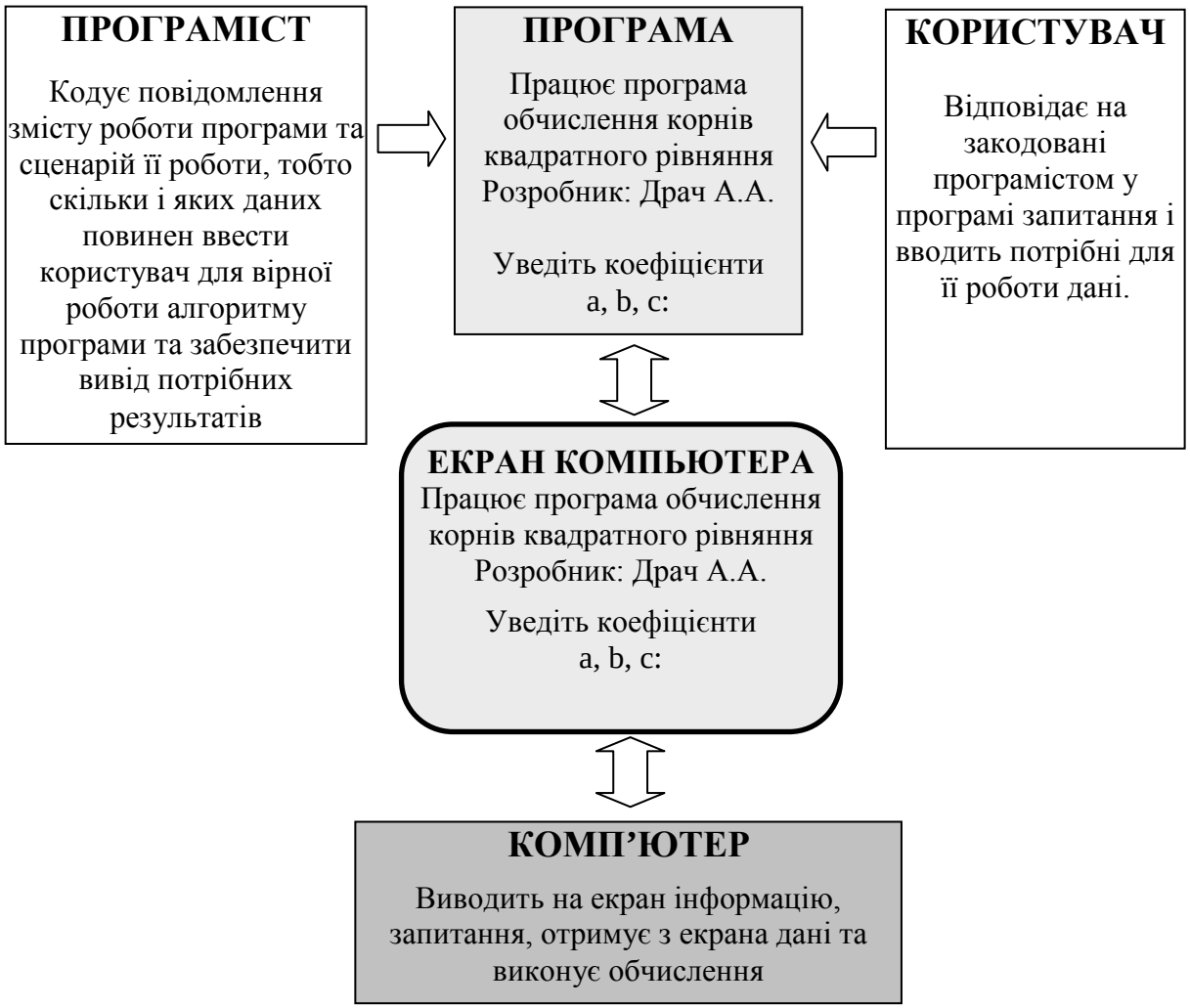


Рис. 8.17. Зміст інтерфейсу користувача

Найпростіша ситуація при виконанні програми полягає у тому, що коли користувачеві треба виконати якісь дії, то потрібно знати, а які ж вони повинні

бути. Наприклад, по ходу виконання програми, треба увести два числа дійсного типу, тобто числа, які мають крапку у своєму відображенні (2.357 та 42.9). Виникає питання: як попередити користувача, що комп'ютер чекає від нього тільки уводу (рос. – ввода) двох дійсних чисел (з крапками серед цифр при відображенні)?

Ось тут, розробник програми (для майбутнього користувача), повинен зробити попередження, що чекає у цей момент від нього комп'ютер. У Турбо Паскалі для цього існує оператор процедури з іменем `Writeln`, фактичними параметрами якого є стрічки (рос. - строки), що підказують користувачеві що робити далі.

Взагалі, для введення інформації у комп'ютер і виводу її з нього існують чотири процедури: `Read, ReadLn, Write` та `WriteLn`. Як відомо, ТП не відрізняє великих літер від малих то не буде помилки, якщо імена ціх же процедур будуть записані тільки малими символами: `read, readln, write` та `writeln.`

Процедура читання `Read` – забезпечує введення даних: чисел, символів, стрічок і таке інше. Нагадуємо, що як тільки у програмі зустрічається оператор `Read`, комп'ютер переходить у режим очікування поки користувач не введе потрібну кількість даних відповідного типу. І поки вони всі не будуть введені, робота комп'ютера не відновиться!

Формат виклику: `Read (x1, x2, x3..., xn);`

Де `x1, x2... , xn` – змінні типів, що описані в розділі програми `Var`, при розбіжності поміж даними що описані і які у них вводяться виникає помилка введення-виводу.

Процедура `ReadLn` при повторному виклику зчитує дані з нової стрічки.

Приклади:

```
Var
   A, B, C, D, Sum1, Sum2 : real;
Begin
   Read(A,B);          1.8   3.4  <Enter>
   Sum1:= A + B;
   Read(C,D);          2.87  0.7  <Enter>
                          ↑
        {Поміж числами повинно бути не менш одного пробілу}
   Sum2:= C + D;
   Readkey
End.
```

Використання процедури `ReadLn`

```
ReadLn (A,B);       1.8   3.4  <Enter>
Sum:= A + B;
ReadLn (C,D);       2.6   0.7  <Enter>
Sum2:= C + D;
```

У процедурах `Write и WriteLn` мається можливість форматування вихідних даних, тобто запису явно завданих стрічкових виразів біля імен змінних з виразами, що визначають ширину поля виводу:

Приклад: Вивід цілих значень, що мають опис:

```
Var
    I : Integer;
```

Значення, що міститься у змінній I	Вираз для виводу цього значення	Результат на екрані ← ліва крайня позиція поля виводу екрану
134	`Write(I);`	`134`
56789	`Write(I);`	`56789`
287	`Write(I,I,I);`	`287287287`
134	`Write(I:6);`	`  134`
1	`Write(I:10);`	`         1`

Якщо для цілого числа при виводі все дуже просто, тобто треба розмістити на екрані тільки знак і значення змінної, то для дійсної змінної все значно складніше. Це виникає з того, що взагалі для дійсних значень існує дві форми представлення:

❶ з фіксованою крапкою (–34.295);

❷ з плаваючою крапкою або у експоненціальній формі (тобто 182600.0 = $1.826*10^5$= +1.826E+05 = 18.26E4 і так далі).

При виводі числа з фіксованою крапкою потрібно відображати чотири складові:

❶ знак числа (знак “плюс” (+), як правило, не відображується);

❷ цілу частину;

❸ крапку, що розподіляє цілу та дробову частини числа;

❹ дробову частину.

У випадку виводу числа типу REAL з плаваючою крапкою для відображення його стандартно відводиться 18 позицій для восьми складових (рис. 8.18):

❶ пуста позиція (0);

❷ знак числа (пуста позиція якщо + або -) – займає одну позицію (1);

❸ ціле значення мантиси (2);

❹ розподіляюча крапка (3);

❺ дробова частина мантиси (4);

❻ буква E, що відзначає початок значення порядку числа (5);

❼ знак порядку: плюс (+) або мінус (-) (6);

❽ числове значення порядку (7).

Номери складових числа →	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Саме число →		**–**	**2**	**.**	**815239**	**E**	**+**	**02**

Рисунок 8.18. Складові числа з плаваючою крапкою

Примітка: У комп'ютерах типу Pentium III і вище відводиться ще більше знаків. А для зручності читання результатів можливо управляти кількістю знаків, що виводяться.

Приклад: Вивід дійсних значень, що описані:

```
Var
  R : real;
```

{У поле шириною 18 символів виводиться десятинне значення змінної R у форматі з плаваючою крапкою й з 2-ма пустими позиціями перед числом, з яких друга – виділяється під знак числа}

Значення, що міститься у змінній R	**Вираз для виводу цього значення**	**Результат на екрані ← ліва крайня позиція поля виводу екрану**
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`

Приклад програми з форматами для виводу даних:

```
Program MyFirst;
Uses Crt;
Var
  A, B : Integer;
  Ratio : Real;
Begin
  Write ('Vvedite dva chisla A, B: ');
  ReadLn(A,B);
  WriteLn('A= ', A:4, ' B= ', B:6);
  Ratio := A / B;
  WriteLn('Rezultat Ratio = ', Ratio:7:2);
  Write('Najmite <Enter>…');
  Readln;
End.
```

На рис. 8.19 приведена програма з великою кількість коментарів, яка має досить розвинутий інтерфейс користувача.

Приклад програми на мові Турбо Паскаль, яка реалізує обчислення згідно формул та інтерактивно взаємодіє з користувачем.

```
Program Second;  { Коментар: Типова програма на мові Турбо Паскаль}
Uses Crt;              {Підключення модуля Crt}
  const t=2.345;           {числова константа}
  const h : integer = 55;{типізована константа}

Var                      {оператор опису змінних }
  a,b,c,r, s,Res : Real; {дійсні змінні}
  i,j,k,n : integer; {цілочисельні змінні}
  str : string [20]; {рядкова змінна}
```

Розділ описів

```
Begin                  {Початок основного блоку програми}
  ClrScr;    {Очищення екрану процедурою ClrScr }
  Writeln('Vvedite dannye dlya programmy: b,c,r,s');
  Readln(b,c,r,s); {Вводимо з клавіатури дані для задачі}
```

$$a = \frac{\sqrt[4]{(b^3 + c^3)}}{\cos(r)^2 * \sin(s)^{\frac{2}{5}}}$$

{ Реалізуємо наведену праворуч формулу → }
{засобами функцій та операцій Турбо Паскалю }
{ й заносимо результат у змінну **a** }
{оператором присвоювання **:=**}
{Піднесення до ступеню реалізуємо у вигляді формули: $\mathbf{a^z = exp(z*ln(a))}$ }

```
  a:=exp(1/4*ln(b*b*b+c*c*c))/(cos (sqr(r))*
              sin(exp(2/5*ln(s))));

  Writeln('Rezultat   a=', a); {Виводимо на екран результат}
```

{Обчислюємо значення *t* згідно формули}

$$t = a * \frac{b^3}{c}$$

```
  Res:=a*(sqr(b)*b/c);  {та заносимо у змінну Res}

  Writeln('Rezultat Res= ', Res); {Виводимо на екран Res}
  Writeln('Na segodnya hvatit?'); Writeln;
  Write ('Vvedite Vashe imya: '); Readln(str); {Вводимо}
                                  {власне ім'я у змінну str}
  Writeln;
  Writeln('Zdravstvuyte,  ',str); Writeln;
  Writeln('  i do svidaniya  ' ,str);
  Readln;
End.
```

Кінець програми

Рис. 8.19. Приклад програми на мові Турбо Паскаль з коментарями

Після запуску на виконання такої програми, якщо відкрити її файл (`second.pas`) командою **Open** з меню **File** й натиснути сукупність клавіш **Ctrl+F9**, на екрані результатів (після вводу потрібної інформації) отримаємо наступні результати (рис.8.20).

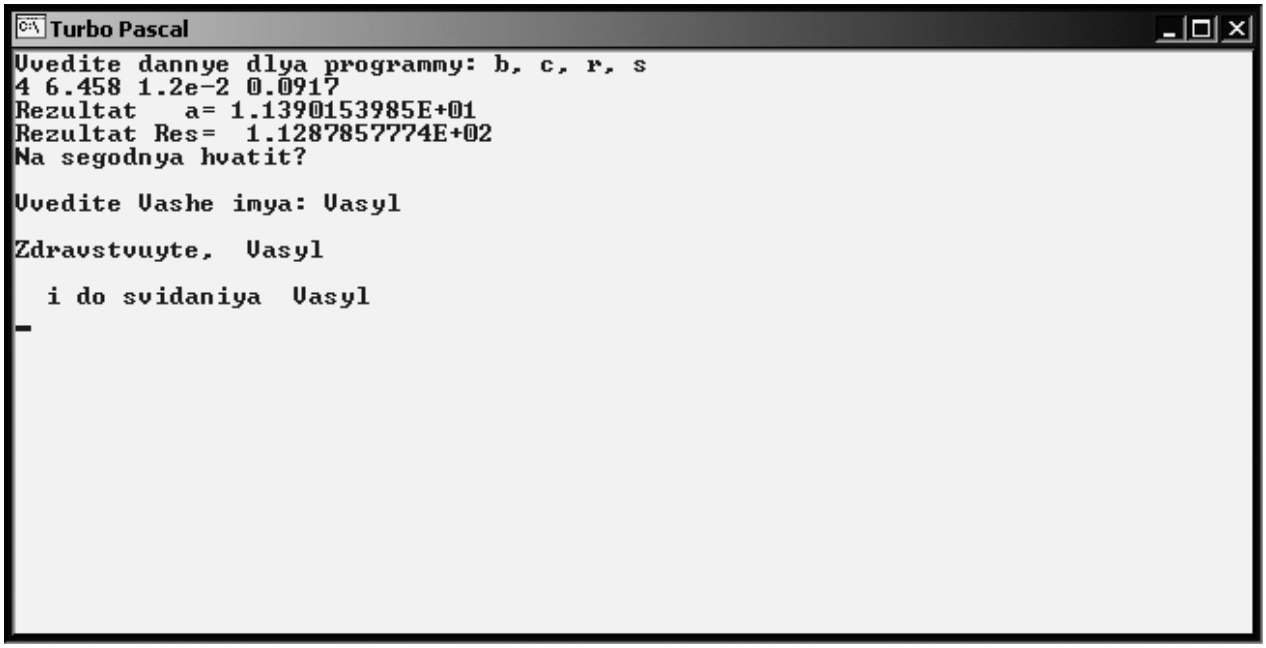


Рис. 8.20. Результат роботи програми з файлу `second.pas`

Треба звернути увагу на різницю поміж діями однотипних операторів уводу даних `Read, Readln` та виводу даних `Write, WriteLn`.

Єдина різниця поміж операторами процедур уводу даних `Read` та `Readln` полягає у тому, що після виконання оператору `Readln` курсор на чорному екрані DOS переводиться на початок наступної стрічки. А після виконання оператора `Read` курсор залишається на тієї ж стрічці. Тобто оператор `Read` працює з частиною стрічки, а оператор `Readln` з усією стрічкою.

Оператор `WriteLn` при виводі даних, що представлені у його списку фактичних параметрів (наприклад: `WriteLn(A,B,C);`), після виводу значень об'єктів `A, B` і `C` на екран переведе курсор на слідуючи стрічку. А після виконання оператора `Write(A,B,C);` курсор залишиться на тієї ж стрічці і буде чекати виводу іншої порції даних. Це буде означати, що вивід інших значень наступним оператором `Write` буде продовжено у ту ж саму стрічку, але до тих пір, поки не буде вичерпане місце, що відведено на екрані для однієї стрічки. Довжина стрічки символів при виводі на екран, до речі, дорівнює 80-ти символам. Таким чином, як тільки кількість виведених у одну стрічку символів перебільшує 80, усі наступні символи починають виводитися з початку наступної стрічки.

Це дуже добре видно на прикладі виводу 10-ти значень числа з фіксованою точкою 2.5676 за допомогою оператора `Write(2.5676)` та 10-ти значень

числа з фіксованою точкою –2.5676 за допомогою оператора `Write(-2.5676)` (рис. 8.21).

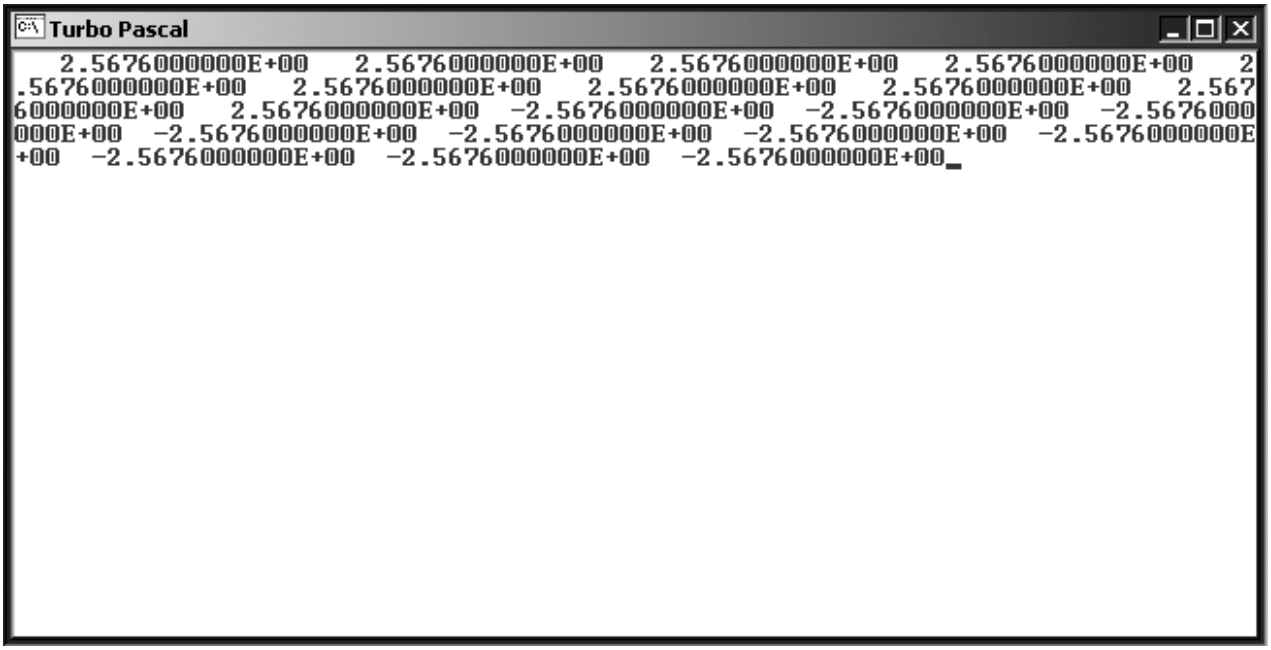


Рис. 8.21. Вивід 20-ти чисел оператором `Write`

Виводу 10-ти значень числа з фіксованою точкою 2.5676 за допомогою оператора `Writeln(2.5676)` та 10-ти значень числа з фіксованою точкою -2.5676 за допомогою оператора `Writeln(–2.5676)` наведено на рис. 8.22.

Рис. 8.22. Вивід 20-ти чисел оператором `Writeln`

Зверніть увагу на те, що при виводі даних операторами `Write` та `Writeln`, Турбо Паскаль виділяє попереду кожного дійсного значення об'єкта пробіл та позицію під знак "–"! Якщо знаку "–" немає, то виводяться два пробіли. Пам'ятайте, також, що при вводі даних з екрану операторами `Read` та `Readln` пробіли ігноруються. Це означає, що при запиту програми на увід даних, їх значення при наборі з клавіатури можна розділяти не тільки одним пробілом, а навіть і декількома.

Вправи

1. Як можна задати вихідні дані в програмі?
2. Як можна вивести інформацію на екран з програми?
3. Як можна увести інформацію з екрану у програму?
4. Поясните виконання операторів `Read` і `Readln` у програмі. У чому їхня відмінність ?
5. Поясните виконання операторів `Write` і `Writeln` у програмі. У чому їхня відмінність?
6. Скласти програму для рішення задачі: "Знайти площу прямокутника зі сторонами A и B. Вивести на екран значення площі з точністю до 3-х десятинних знаків у дробовій частині числа".
7. Скласти програму для обчислення середнього арифметичного чотирьох цілих чисел X, Y, Z і C.
8. Період коливань маятника довжиною L обчислюється по формулі: $t = 2\pi\sqrt{\frac{l}{g}}$, де g – прискорення вільного падіння (9.81м/с^2). Знайти період коливань маятника.
9. Сила притягання F між тілами масами m_1 і m_2, що знаходяться на відстані r друг від друга, дорівнює: $F = \gamma \cdot \frac{m_1 \cdot m_2}{r^2}$, де гравітаційна постійна γ=6.673•10^{11}м^3/(кг•з^2). Знайти силу притягання F.
10. Периметр p правильного n-косинця, описаного біля окружності радіусом r, дорівнює: $p = 2n \cdot r \cdot tg\left(\frac{\pi}{n}\right)$. Знайти периметр p.
11. Енергія E, випромінювана чорним тілом на хвилі довжини λ при температурі τ, дорівнює: $E = \frac{2\pi \cdot c \cdot h \cdot \lambda^{-5}}{e^{c \cdot h / \beta \cdot \lambda \cdot \tau^{-1}}}$, де c =2,997924•10 ^{8} – швидкість світла; h=6,626•10^{-34} Дж/с – постійна Планка; β=1,38• 10^{-23} Дж/град – постійна Больцмана. Знайти енергію E, випромінювану чорним тілом.

8.7. Вирази, операнди і операції

Вирази у програмуванні служать для визначення дій, котрі у математиці описуються формулами.

Вирази у мові Турбо Паскаль складаються з операцій, операндів та вбудованих (рос. – встроенных) стандартних функцій (`cos, sin, exp` та інш., див. таблицю у описі дійсних типів даних). По кількості операндів, які об'єднують операції, останні поділяються на унарні (одномісцеві) та бінарні (двомісцеві) (табл. 8.9).

Таблиця 8.9.

Приклади виконання унарних та бінарних операцій

Унарні операції		Бінарні операції	
Вираз	Результат	Вираз	Результат
– (+7)	– 7	2 + 6	8
– (–2)	2	(0.5 – 1)*2+10	9
`not False`	`True`	`False or True`	`True`

Послідовність виконання операцій визначається трьома факторами:

❶ пріоритетом операцій, що використовуються;

❷ порядком розташування операцій у виразах;

❸ використанням дужок (рос. – скобок).

У мові ТП усі 22 операції за пріоритетами поділяються на 4 групи (табл.8.10).

Таблиця 8.10

Пріоритети операцій у Турбо Паскалі

Групи пріоритетів	Операції	Категорія операцій
Перший (вищий)	+ – **not** **@**	Унарні операції
Другий	* / **div** **mod** **and** **shl** **shr**	Бінарні операції типу множення
Третій	+ – **or** **xor**	Бінарні операції типу додавання
Четвертий (нижчий)	= <> < > <= >= **in**	Бінарні операції відносин (рос.-отношений)

Найпершими обчислюються підвирази у дужках як самостійні операнди, а потім цей результат використовується для виконання операцій, що їх обрамовують (рос. – обрамляют). Операції першого (вищого) пріоритету виконуються у наступну чергу. Операції четвертого (нижчого) пріоритету

виконуються останніми. Операції рівного пріоритету (типу *, /) виконуються зліва направо послідовно. Це означає, що у виразу `A*B/C` значення змінної `A` буде по-перше помножене на значення змінної `B`, а потім результат цієї дії буде поділений на значення змінної `C`.

За характером операцій, що виконуються, їх можна поділити на такі групи (таблиця 8.11):

Таблиця 8.11

Типи операцій

№ п/п	Група операцій	Тип операцій	
		Унарні	Бінарні
1	Арифметичні операції	+, –	+, –, *, /, div, mod
2	Операції відносин	немає	=, <>, <, >, <=, >=
3	Булевські (логічні) операції	not	and, or, xor
4	Порозрядні логічні та пересувні (рос. – сдвиговые) операції	not	and, or, xor, shl, shr
5	Стрічкові (рос. – строковые) операції	немає	+ (конкатенація)
6	Операції над множинами	+, –	+, –, *, in, <=, >=
7	Операція узяття адреси	@	немає

Наведемо приклади виконання деяких операцій у виразах з різними типами даних (таблиця 8.12).

Таблиця 8.12

Приклади обчислення виразів різних типів.

Тип даних у виразу	Вираз	Послідовність виконання	
		Частковий вираз	Результат
1	3	4	5
Дійсний	Арифметичний вираз 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
Логічний	Логічний вираз (4>5) and not (2<=9)		
		4>5= 2<=9= not(true)= false and false = Результат:	false true false false false

Продовження таблиці 8.12.

1	3	4	5
Цілий	Арифметичний вираз 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 кодуванні програми програміст повинен знати і виконувати деякі важливі правила, щоб складена їм програма відповідала його намірам.

1. Два символи операцій не повинні стояти поруч. Тому вираз `A * - B` є безглуздим, але вираз `A * (- B)` є припустимим.

2. Дуже важливу роль відіграють у виразах дужки. Від їх положення залежить хід виконання операцій так як і математичних записах. Наприклад, вирази `A/B+C` і `A/( B+C)` дадуть різні результати у зв'язку з різницею у пріоритетах операцій що входять до них.

3. Якщо черга виконання операцій у виразах не повністю визначена дужками, то операції виконуються таким чином: по-перше виконуються піднесення (рус. – возведение) у потрібну степінь, потім усі операції множення та ділення, а потім операції сумування та віднімання. Тому наступні два вирази еквівалентні:

```
A*B+C/B-E*X
(A*B)+(C/B)-(E*X)
```

4. Всередині послідовностей множень та ділень, або сумувань і віднімань, у котрих порядок операцій не визначений дужками, дії виконуються підряд справа наліво.

Вправи

1. Які операції є унарними? Привести приклади.

2. Які операції є бінарними? Привести приклади.

3. Поясніть поняття пріоритету математичних операцій при обчисленні математичних виразів.

4. У якому порядку виконуються математичні операції?

5. Яке призначення дужок у математичних виразах?

6. Як записуються математичні вирази в Турбо Паскалі?

7. Як можна змінити природний порядок виконання математичних операцій у математичних виразах?

8. Обсяг циліндра з радіусом основи R і висотою H дорівнює: $V = \pi \cdot R^2 \cdot h$. Площа його бічної і повної поверхонь відповідно рівні: $S_{\text{б ок}} = 2 \cdot \pi \cdot R \cdot h$, $S_{\text{п}} = 2 \cdot \pi \cdot R \cdot h + 2 \cdot \pi \cdot R^2$. Знайти V, $S_{\text{б}}$ і $S_{\text{п}}$.

9. Знайти довжину окружності, площу кругу й обсяг кулі того самого радіуса R. При обчисленні використовувати формули: $l = 2 \cdot \pi \cdot R$, $S = \pi \cdot R^2$, $V = \frac{4}{3} \pi \cdot R^3$.

10. Визначити швидкість різання кругла шліфувального верстата: $V = d_1 \cdot z \cdot \pi \cdot \frac{d_3}{d_2}$, де d_1 - діаметр шківа двигуна, d_2 - діаметр робочого вала, d_3 - діаметр інструмента, що ріже, z - частота обертання.

11. Обчислити загальну поверхню й обсяг круглого конуса, що має радіус R і довжину утворюючої L. При обчисленні використовувати формули: $S = \pi \cdot R^2 + \pi \cdot R \cdot L$, $V = \frac{1}{3} \pi \cdot R^2 \cdot H$, де H - висота конуса, обумовлена по формулі: $H = \sqrt{L^2 - R^2}$.

12. Дано окружність радіуса r. Знайти площі сегмента і сектора. При обчисленні використовувати формули: $S_{\text{сек}} = \frac{\pi \cdot r^2 \cdot \alpha}{360^\circ}$, $S_{\text{сег}} = \frac{r^2}{2}\left(\frac{\pi \cdot \alpha}{180^\circ} - \sin\alpha\right)$, де α - центральний кут у градусах.

13. Дано гіпотенузу і катет прямокутного трикутника. Знайти другий катет і радіуси описаної й уписаної окружностей R і r. При обчисленні використовувати формули: $r = \frac{2 \cdot S}{a + b + c}$, $R = \frac{a \cdot b \cdot c}{4 \cdot S}$, де a, b, c - сторони трикутника, S - площа.

14. Обчислити відстань між двома точками з координатами (x_1, y_1) і (x_2, y_2). Для обчислень скористайтеся формулою: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

15. Знайдіть периметр і площу прямокутного трикутника, якщо відомі довжини двох катетів.

16. Дано сторону рівностороннього трикутника. Знайти його периметр і площу.

17. Знайти площу кільця і площу частини кільця з центральним кутом φ (у градусах). Для обчислень скористатися формулами: $S = \pi\left(R^2 - r^2\right)$, $S = \frac{\varphi \cdot \pi}{360}\left(R^2 - r^2\right)$.

8.8. Головні задачі комп’ютерних обчислень. Прості типи даних. Ініціалізація даних перед обчисленням виразів

Головною задачею, що вирішує комп'ютер, є виконання різного роду обчислень по заданим алгоритмам. При вирішенні математичних задач це, звичайно, робиться за формулами. Формули можуть бути простими чи складними (рис. 8.23), але підхід до подання їх у програмі завжди єдиний.

$$z = \sqrt{\frac{S-R}{D+Y}}\ , \quad t = (y+6)^5 + 2{,}705 \cdot \log_5 \left[\frac{1}{B} + \frac{1}{B^2} - \frac{1}{B^{\frac{3}{4}}} + B^3 \right] - \sin^6 y + \left| \sqrt{B} - 3 \right|$$

$$y\,(x) = \oiiint_D \frac{(x^2+1)\,dx}{\sqrt{x^3+7x^2-23}}$$

Рис. 8.23. Приклад математичних формул різної складності

За допомогою конструкцій мови Турбо Паскаль формується ***вираз*** (рос. – выражение), який повинен бути коректним з точки зору синтаксису і семантики мови ТП. Отриманий після виконання обчислень у виразу результат, як правило, потрібно зберегти для цілей подальшого використання, тобто:

❶ запису його на диск;
❷ виводу на екран комп’ютера;
❸ виводу на принтер,

і таке інше.

Зберігання цілих і дробових чисел, символів, невеликих текстів і других типів даних виконується у областях пам'яті комп’ютера, які звуться ***змінними*** (рос. – переменными) і мають такі властивості (Таблиця. 8.13):

❶ імена (так звані ідентифікатори);
❷ приписаний відповідний тип, який характеризує їх у виразах;
❸ різний розмір (у байтах), який залежить від типу.

Таблиця 8.13. Властивості змінних у виразах

Ім'я змінної (ідентифікатор)	Тип змінної	Значення	Кількість байтів, що відводиться у пам'яті
`Flag`	`Boolean` (логічний)	False	1
`Disc`	`Integer` (цілий)	234	2
`Tri`	`Real` (дійсний)	3.33333333333	6

Подальша підстановка цих змінних у вирази дозволяє Вам фактично використовувати значення, що у них розташовані.

У мові ТП типи даних розподіляються на ***прості*** (базові) та ***структурні*** (див. Додаток 7). Прості типи є базовими, бо на їх основі будуються більш складні структурні типи. Так, як дані простого типу, до якого відносяться константи, змінні, вирази і функції мають тільки одне значення, тому прості типи ще звуться скалярними. Скалярний тип визначає множину значень змінних, констант, виразів та функцій, що належать до даного типу, форму уявлення у комп'ютері значень цих величин та операції, що до них можуть бути застосовані.

Скалярний тип може бути стандартним (базовим) чи задаватися користувачем за допомогою оператора (зарезервованого слова) `type`.

До базових (стандартних) типів належать:

❶ дійсний (`real`);

❷ цілий (цілочисельний) (`integer`);

❸ логічний (`boolean`);

❹ символьний (`char`).

Таким чином, щоб обчислити результат за якоюсь формулою потрібно у першу чергу сконструювати адекватний їй вираз з даних відповідного типу. А для того, щоб було куди його (результат) зберегти – потрібно мати відповідну змінну (тобто місце у пам'яті комп'ютера з відповідним ім'ям і типом). Іншими словами, ім'я змінної фактично є її адресою у пам'яті ПК.

Найчастіше пересилку даних у змінну виконують за допомогою оператора привласнення (рос. – присваивания) (див. діаграму нижче, рис. 8.24).

:=

Його дію можна графічно представити наступним чином:

A ⬅ <значення>

Наприклад:

A ⬅ H/ 0.5

Рис. 8.24. Заміщення новим обчисленим значенням старого у змінній `A`

(Де <значення> – дане будь якого ПРОСТОГО типу (`real, integer`)).

Тобто, якщо у змінній `H` знаходилась константа 2.0, то ділення її на 0.5 дає значення 4.0, яке заноситься у змінну `A`, тобто за адресою, яка відповідає у пам'яті місцю, яке відповідає імені `A` (рис. 8.24). Найважливішим є те, що нове значення (4.0) ***ЗАМІЩУЄ***, тобто стирає у змінній `A` її ***ПОПЕРЕДНЄ ЗНАЧЕННЯ***. Наступним важливим моментом є те, що дії з числами та іншими значеннями виразів (символами, логічними значеннями та ін.), виконуються на суматорі[20].

[20] Суматор – електронний пристрій комп'ютера, який знаходиться у мікропроцесорі і призначений для виконання арифметичних і логічних дій з даними (числами). Інша назва – арифметично-логічний пристрій.

Для вищевказаного прикладу процес переміщення числових значень поміж пристроями ПК можна схематично уявити так (рис. 8.25).

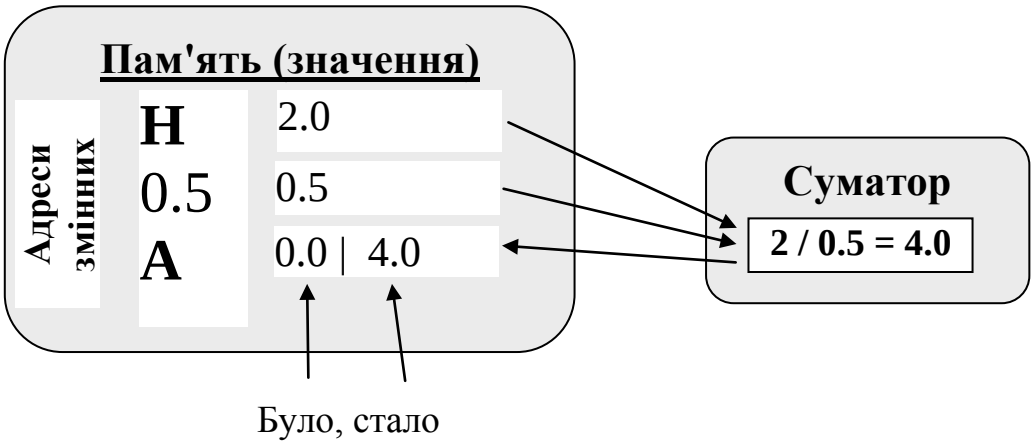


Рис. 8.25. Взаємодія чисел у процесі обчислень

Треба мати на увазі також те, що у математичному виразі X = 1, знак "=" має значення ***"дорівнює"***, а у виразах на алгоритмічній мові ТП у виразі `X` **:=** 1, знак "**:=**" має значення ***"засилається"***. Тобто значення 1 засилається у змінну `X` і заміщує там попереднє значення, що зберігалося.

Формули й більш складні послідовності алгоритмічних дій конструюються за допомогою трьох головних інструментів:

❶ виразів;
❷ функцій;
❸ процедур.

Наприклад, обчислення об'єму конусу, що має математичне відображення (8.1), реалізується таким чином (табл. 8.14)

$$V = \frac{1}{3}\pi R^2 H \qquad (8.1)$$

Таблиця 8.14

Реалізація формули (8.1) трьома різними засобами

№	**Тип реалізації**	Форма реалізації:	Занесення значення обчисленого об'єму у відповідні змінні: Vkon1, Vkon2, Vkon3
1	Вираз	`1/3*PI*(R*R)*H`	Vkon1 := 1/3*PI*(R*R)*H;

2	Функція (опис у програмі)	`Function V(PI,R,H :Real):Real;`
		`Begin`
		`  V:= 1/3*PI*(R*R)*H ;`
		`End;`
			Vkon2 := V (PI,R,H);

3	Іроцедура (опис у програмі)	`Procedure V(PI,R,H: Real;`
		`  var W:Real);`
		`Begin`
		`W:= 1/3*PI*(R*R)*H; End;`
			V (PI,R,H, Vkon3);

Після виконання усіх трьох дій, які представлені у стовпчику 3 таблиці (двох операторів привласнення і одного оператора процедури) у трьох змінних Vkon1, Vkon2, Vkon3 будемо мати однакові результати. З цих прикладів добре видно, що ***функція є числом***, а ***процедура – оператором***.

Якщо уважніше подивитися на конструкції у таблиці (8.14), то дуже добре видно, що і вираз, і функція і процедура – це пряме відображення послідовності проведення обчислень за розробленим користувачем алгоритмом. Але комп'ютер ніколи сам не зможе обчислити ці вирази, бо не знає, які ж саме дані треба підставити у разі чергового запуску програми на місце змінних, які відображують висоту конусу H і радіус його основи (рос.–основания) R. А число π дуже добре відоме усім і так.

Процес надання потрібних для обчислення виразів початкових значень змінним, що приймають участь у обчисленнях, зветься ***ініціалізацією***.

Ініціалізуватися можуть константи і змінні. Константи набувають значень у виді простих і типізованих констант (табл. 8.15). Типізована константа відрізняється від звичайної тим, що має визначений тип (Real, Integer та ін.) і значення якої може змінюватися користувачем у процесі подальшої роботи програми. Таким чином, для обчислення значення формули (8.1) у програмі висоту і радіус можна обрати у вигляді констант (а не змінних!) і значення ініціалізувати так, як показано у таблиці 8.15.

Таблиця 8.15

Приклади ініціалізації об'єктів у програмах у вигляді констант

Назва (вид) констант	Реалізація у програмі
Прості константи	`Const H = 1.34;` `R = 0.56; I = 2;`

Типізовані константи	`Const H : Real = 1.34;`
	`R : Real = 0.56;`
	`I : Integer = 2;`

Змінні, що попередньо описуються у операторі Var (який відводить відповідним об'єктам місце у пам’яті), можуть ініціалізуватися або за допомогою оператора привласнення, або за допомогою стандартних процедур з перемінною кількістю параметрів Read і ReadLn. Ці процедури дозволяють уводити дані у змінні користувачем з екрану комп'ютера (табл.. 8.16).

Таблиця 8.16

Ініціалізація змінних у програмах

Оператори ініціалізації	Реалізація ініціалізації

Привласнення (:=)	`Var`
	`R,H : Real; I : Integer;`
	`Begin R:=0.56; H := 1.37;`
	`I := 2;`
	`End.`

Процедури Read i Readln	`Var`
	`R,H : Real; I : Integer;`
	`Begin Readln(R, H, I); End.`

При виконанні ініціалізації за допомогою оператора привласнення комп'ютер сам занесе вказані користувачем дані у відповідні змінні `R,H` і `I`. При використанні процедур `Read i Readln`, коли виконання дій доходить до них, ПК відкриває користувачеві екран DOS і чекає уводу потрібних даних.

На першому екрані DOS (рис. 8.26) представлено становище, коли було зроблене перше переключення з вікна редактора до вікна DOS і комп'ютер чекає уводу користувачем чергового значення , яке потрапить до змінної R. Другий екран DOS демонструє стан, коли значення змінних уведені у відповідні змінні оператором процедури `readln` і виведені на екран оператором `writeln`. У передостанній стрічці останнього на рисунку 7.4 екрану DOS дані виведені безформатно, а у останній стрічці – з використанням формату.

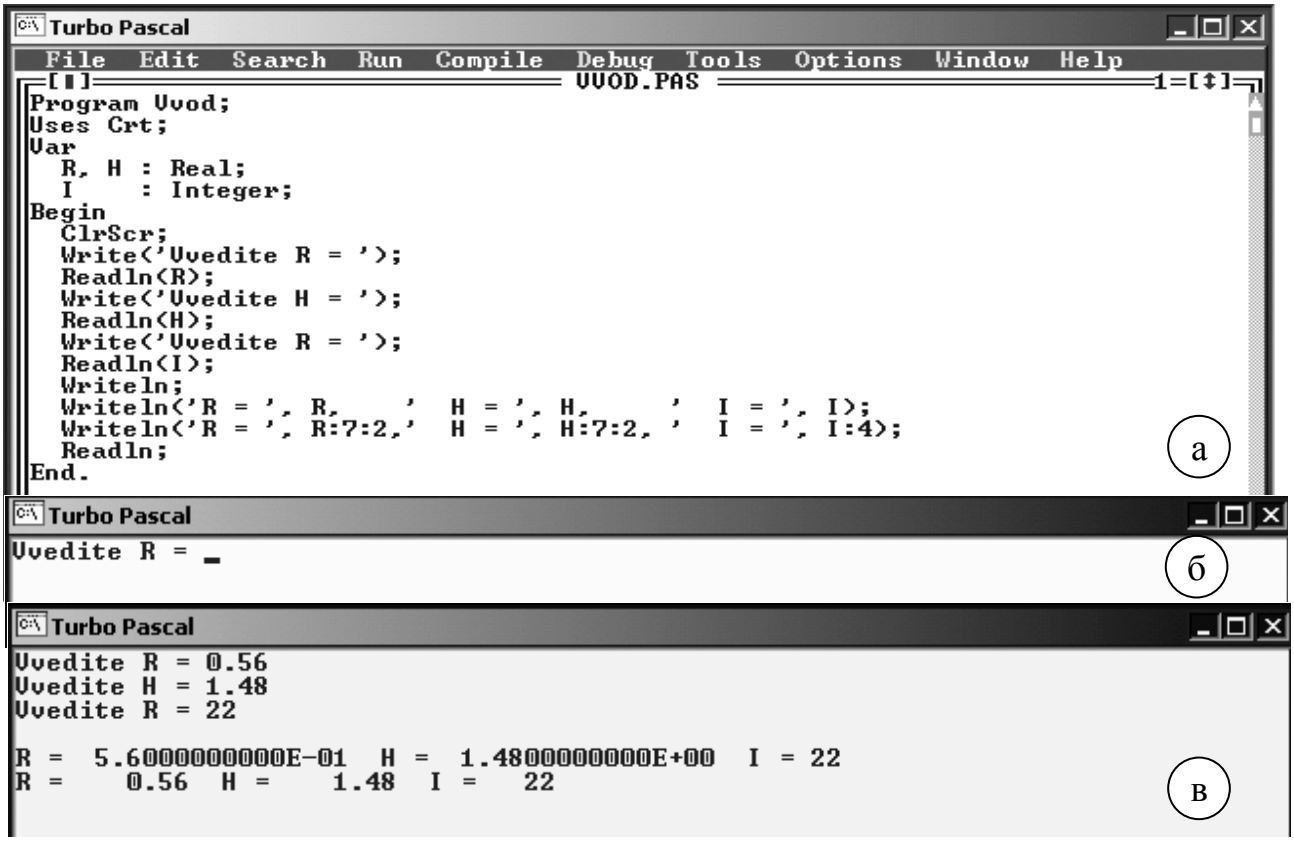


Рис. 8.26. Екран середовища ТП (а) та два екрани MS-DOS у процесі уводу даних у змінні програми процедурою `Readln` для подальших обчислень та виводу процедурами `Write` та `Writeln` без форматування (б) та з форматуванням (в).

Вправи

1. Для чого потрібна ініціалізація даних у програмах?
2. Як виконується у програмі операція привласнення?
3. Які типи даних у ТП є базовими? Чому вони так звуться?
4. Які властивості даних визначає стандартний тип?
5. Що потрібно для конструювання виразу?

8.9. Дійсні типи даних (Real). Операції і вбудовані функції роботи з ними.

Одним з найбільш розповсюджених типів даних для вирішення задач з раціональними (дробовими) числами є дійсний (рос. – вещественный), що у ТП має назву `Real`. Число, що описується таким типом, має максимально до 12 цифр при запису мантиси[21] (див. далі відповідні таблиці). Мінімальне число має порядок (-39), а максимальне – (+39). Наприклад, мінімальне число дорівнює

–9.9999999999E+39 (тобто $-9.9999999999*10^{+39}$),

а максимальне –

+9.9999999999E+39 (тобто $+9.9999999999*10^{+39}$).

При уведенні даних у програму для вирішення задач така велика точність практично не потрібна. Але вона має велике значення при проведенні складних обчислень.

Дійсні числа характеризуються тим, що мають точку у їхньому запису (Наприклад: 23.76). Точка відділяє цілу частину (23) від дробової (76) (рос.- дробной) частини. У виразах ці числа використовуються у вигляді різних об’єктів (див. табл. 8.17):

❶ констант;
❷ змінних;
❸ значень відповідних стандартних функцій.

Таблиця. 8.17

Форми уявлення и запису дійсних чисел

Вид уявлення даних	Уявляються	Приклади запису		
Константи	Числами	2.36	–40.256	1e-7
	Іменами	`Beta`	`Gamma`	`Teta`
	Текстом (стрінгами)	'61.912E-04'	'0.0000001'	'1.0e+5'
Змінні	Іменами	`Irez2`	`NextFunk`	`IOrect`
Функції	Зарезервованими іменами функцій з потрібними типами аргументів	`cos`(0.5)	`exp`(`Irez2`)	`sqrt`(2.0)
		`ln`(0.239)	`abs`(–5.1/0.3)	`sqr`(3E–2)

Константи, змінні і функції дійсних типів об'єднуються у вирази знаками арифметичних операцій (+, –, *, /). У виразах дійсного типу можуть бути присутні об’єкти, які уявляються цілими числами. Дійсні константи відображаються у двох формах: з фіксованою точкою та у експоненціальній формі (табл. 8.18). Експоненціальна форма уявляє число у вигляді цілої і дробової його частин, що помножені на основу 10 у цілій степені.

[21] Мантиса – дробова частина числа, що записується у експоненціальній формі (див. далі)

Таблиця 8.18

Різні форми уявлення дійсних чисел

Число з фіксованою точкою		Воно ж у експоненціальній формі (інші назви: напівлогарифмічна форма або з плаваючою точкою)
1.0	=	1.00000000000E+00
0.0000001	=	1e–7 = 1.0E–7
456.437	=	4.56437E+02 = 456.43E0
–0.0256	=	–2.56E–2

У мові Турбо Паскаль <u>не допускається</u> (!), щоб число дійсного типу:

– починалося з точки (.15 – невірно!);

– кінчалося точкою (29. – невірно!);

– не мало значення порядку після покажчика порядку літери E (6.11E - невірно!);

– мало би унарний знак перед покажчиком порядку літери E (-54.99+E - невірно!).

Дійсне число (константа) з фіксованою точкою містить наступні компоненти (рис. 8.27)

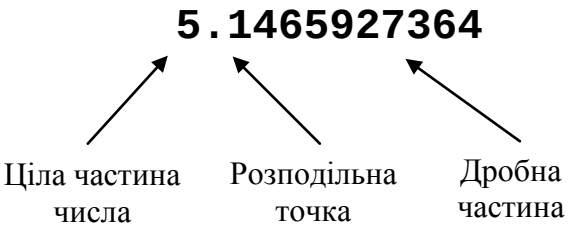


Рис. 8.27 Складові числа з фіксованою точкою

Для числа у експоненціальній формі, тобто з плаваючою точкою, складові числа такі (рис. 8.28):

Рис. 8.28. Складові частини числа дійсного типу з плаваючою точкою (так звана експоненціальна або логарифмічна форма запису)

Назва "число з плаваючою точкою" виникла тому, що її можна пересувати поміж цифрами у числі, змінюючи відповідно показник ступеню числа, тобто порядок. Таким чином, рівнозначні поміж собою такі форми запису числа:

5.1465927664 E–2 = 514.65927664 E–4 = 0.051465927664

Точність відображення числа дійсного типу залежить від місця у пам'яті ПК, що відводиться для його розташування (для його цифр, що мають значення) (табл. 8.19).

Таблиця 8.19

Границі змін значень дійсних чисел для різних описів

Довжина, байт	Назва типу	Кількість значущих цифр	Діапазон десятинного порядку
4	`Single`	7..8	$10^{-45} \ldots 10^{+38}$
6	`Real`	11..12	$-10^{+39} \ldots 10^{+38}$
8	`Double`	15..16	$-10^{+324} \ldots 10^{+308}$
10	`Extended`	19..20	$-10^{+4951} \ldots 10^{+4932}$
8	`Comp`	19..20	$-2 \cdot 10^{-63} + 1 \ldots 2 \cdot 10^{+63} - 1$

У виразах дійсного типу можливо використовувати наступні арифметичні функції, що вбудовані у мову ТП (таблиця 8.20). Треба звернути увагу на те, що у дужках вказані типи аргументів, а за дужками – тип результату роботи цих функцій.

Таблиця 8.20

Вбудовані арифметичні функції мови ТП

Найменування ф-ції	**Дії, що виконуються**
Abs(X:real):real;	– абсолютне значення аргументу (|x|)
ArcTan(X:real):real;	– арктангенс аргументу (arctg x)
Cos(X:real):real;	– косинус аргументу (cos x)
Exp(X:real):real;	– експонента аргументу, тобто e^X.
Frac(X:real):real;	– дробна частина аргументу
Int(X:real):real;	– ціла частина аргументу
Ln(X:real):real;	– натуральний логарифм (ln x)
Pi	– значення Pi=3.141592653…….
Sin(X:real):real;	– синус аргументу (sin x)
Sqr(X:real):real;	– квадрат аргументу (x^2)
Sqrt(X:real):real;	– корінь квадратний аргументу ($\sqrt{x}$)
Round(X:real):Integer;	– округлення дійсного числа X до найближчого цілого
Trunc(X:real):Integer;	– округлення дійсного числа X до цілого з відкиданням
Randomize	– ініціалізує випадкове значення вбудованого
Random(Range:Word);	– якщо параметр `Range` відсутній функція повертає дійсне число у діапазоні від 0 до 1; – якщо параметр `Range` вказаний – то ціле число у діапазоні від 0 до значення `(Range-1)`

Так як список функцій у Турбо Паскалі дещо обмежений, користувач може самостійно конструювати деякі важливі і часто потрібні у роботі функції. Наприклад, оскільки у ТП немає функцій *tg x* і *ctg x*, потрібно виразити їх за допомогою функцій *sin x* і *cos x*, наприклад так:

$$tg\ x = \frac{\sin x}{\cos x}, \quad ctg\ x = \frac{\cos x}{\sin x} = \frac{1}{tg\ x} \qquad (8.2)$$

А коли у програмі буде потрібно обчислити, припустимо, значення цих функцій від аргументу 0.6, на мові ТП ці вирази запишуться так:

```
.........................
zntg := sin(0.6)/cos(0.6);
  znctg := cos(0.6)/sin(0/6);
.........................
```

Тобто у змінні `zntg` і `znctg` будуть занесені значення відповідних тангенса і котангенса від константи 0.6.

Нижче наведені функції, що найбільш часто вживаються у програмах користувачів на мові ТП (таблиця 8.21). Зрозуміло, що їх повинен конструювати сам користувач.

Таблиця 8.21

Додаткові функції, що може описати користувач

Алгоритм реалізації ф-ції	**Дії, що виконуються**
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)
Arccos(X:real)=Pi/2-Arctan(x/sqrt(1-x*x))	– арккосинус x
PowerA(X:real)=exp(x*Ln(A))	– A^x (дійсна степінь числа A)
LogA(X:real)=Ln(x)/Ln(A)	– логарифм значення *x* з будь якої основи A – $\ln_A x$ (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, якщо x>0; 0, якщо x=0; -1, якщо x<0}	– функція знаку
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	– гіперболічний косинус x
Csc(X:real)=1/sin(x)	– косеканс x (x не дорівнює 0)
Sec(X:real)=1/cos(x)	– секанс x (x не дорівнює Pi/2)

У загальному вигляді взаємодія змінних, констант і функцій у виразі може бути представлена так (рис. 8. 29).

Рис. 8.29. Відповідність типів даних, констант, функцій і операцій у виразах до загального обраного дійсного типу даних (Real)

Вправи

1. Знайти периметр і площу паралелограма зі сторонами a, b і кутом між ними α. При обчисленні використовувати формулу: $S = a \cdot b \cdot \sin\alpha$.

2. Знайти внутрішній кут α і суму внутрішніх кутів правильного опуклого n-косинця. При обчисленні використовувати формули: $\alpha = \pi\frac{n-2}{n}$, $s = \pi(n-2)$.

3. Знайти обсяг і площу поверхні прямого паралелепіпеда зі сторонами a, b і c.

4. Знайти середню лінію і площу трапеції, якщо відомі її основа і висота.

5. Дано координати трьох вершин трикутника A(x_1,y_1), B(x_2,y_2) і З(x_3,y_3). Знайти середини його сторін. При обчисленні використовувати формули:

$x = \frac{(x_1 + x_2)}{2}$, $y = \frac{(y_1 + y_2)}{2}$, де M(x,y) - середина відрізка AB, заданого точками A(x_1,y_1) і B(x_2,y_2).

6. Дано координати трьох вершин трикутника A(x_1,y_1), B(x_2,y_2) і З(x_3,y_3). Обчислити периметр трикутника. Для обчислень скористайтеся формулою відстані між двома точками $A(x_1,y_1)$ і $B(x_2,y_2)$: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

7. На площині дана пряма рівнянням Ax+By+C=0 і точка M с координатами (x_1,y_1). Знайти відстань d від крапки до прямої: $d = \frac{A \cdot x_1 + B \cdot x_2 + C}{\sqrt{A^2 + B^2}}$.

8. Дано два вектори $\vec{a}$(x_1,y_1)і $\vec{b}$(x_2,y_2) і кут φ між ними (у градусах). Знайти скалярний добуток векторів по формулі: $\left(\vec{a},\vec{b}\right) = \left|\vec{a}\right| \cdot \left|\vec{b}\right| \cos\varphi$.

9. Дано два вектори $\vec{a}$(x_1,y_1)і $\vec{b}$(x_2,y_2). Знайти кут φ між ними. При обчисленні використовувати формули: $\left(\vec{a},\vec{b}\right) = \left|\vec{a}\right| \cdot \left|\vec{b}\right| \cos\varphi$, $\left(\vec{a},\vec{b}\right) = x_1 \cdot x_2 + y_1 \cdot y_2$, $|\vec{a}| = \sqrt{x_1^2 + y_1^2}$.

10. На площині дані дві прямі лінії: $y=k_1x+b_1$ і $y=k_2x+b_2$. Знайти кут φ між прямими, скориставшись формулою: $tg\varphi = \frac{k_2 - k_1}{1 + k_1 k_2}$.

11. Обчислити кути трикутника, сторони якого задані рівняннями прямих: $y=k_1x+b_1$, $y=k_2x+b_2$ і $y=k_3x+b_3$. Для обчислень скористатися формулою: $tg\varphi = \frac{k_2 - k_1}{1 + k_1 k_2}$, де k_1 і k_2 - коефіцієнти прямих, заданих рівняннями $y=k_1x+b_1$ і $y=k_2x+b_2$, а φ - кут між ними.

12. Написати програму для обчислення площі бічної поверхні $S_{\text{бок}} = 2\pi rH$ й обсягу $V = \pi r^2 H$ циліндра по заданим радіусі підстави r і висоті H. Відповідь вивести двічі: у стандартному виді і з заданою шириною поля висновку.

13. Написати програму для розрахунку часу $t = \frac{2V_0}{g}$ і дальності $l = V_0 t \cos\alpha$ польоту снаряда, що вилетів зі стовбура знаряддя з початковою швидкістю V_0 під кутом α до обрію. Прискорення вільного падіння $g = 9{,}8\frac{M}{c^2}$.

Відповідь вивести двічі: у стандартному вигляді і з заданою шириною поля висновку.

8.10. Цілочисельні типи даних (Integer). Операції та вбудовані функції роботи з ними.

На рівні з дійсними типами даних у програмах ТП використовуються цілочисельні (рос.–целочисленные) типи даних, які у ТП мають загальну назву "`Integer`". Цілі числа характеризуються тим, що на відміну від дійсних записуються без точки (Наприклад, число тридцять чотири запишеться так: 34).

Цілі числа віднесені у мові Турбо Паскаль до ***порядкових типів*** даних (див. Додаток 7).

До порядкових типів також віднесені такі типи (шо розглядаються далі):

❶ логічний (рос.-логический);
❷ символьний (рос.-символьный);
❸ перелічувальний (рос.-перечислимый);
❹ тип–діапазон.

До значень кожного з цих типів може застосовуватися функція `ord (x)`. У результаті для кожного з типів застосування функції `ord (x)` дає значення:

❶ цілий (значення цілого, тобто `ord (9)` = 9);
❷ логічний (0–1, тобто `ord(false)` = 0, а значення `ord(true)` = 1);
❸ символьний (0–255, тобто `ord ('M')` = 77, див. таблицю ASCII кодів у Додатку 6);
❹ перелічувальний (рос.-перечислимый);
❺ тип–діапазон (значення базового типу).

У виразах цілі числа, також як і дійсні, використовуються у вигляді:

❶ констант;
❷ змінних;
❸ значень відповідних стандартних функцій (див. табл. 8.22).

Треба мати на увазі, що деякі функції (`abs(x), sqr(x), random(x)` з цілим аргументом) використовуються водночас і для дійсних і для цілих типів даних. Це значить, що їхні значення при обох типах аргументів можна привласнювати дійсним змінним і використовувати у дійсних виразах. Але абсолютно неможливо використовувати дійсні результати у цілочисельних виразах і привласнювати дійсні значення цілочисельним змінним.

Наприклад:

```
Var
r, t : real;
i, j : integer;
begin
 .............
i := sqr (3); {i набуває ціле значення 9}
 r := sqr (3.0) {r набуває (дійсне!!!) значення 9.0 };
   j := i + r; {Так не можна!!!}
    t := i + r; {Так можна !!!}
...............
```

Ось чому потрібно уважно стежити за співпаданням типів у виразах ТП (табл. 8.22).

Таблиця. 8.22

Уявлення цілочисельних даних

Вид уявлення даних	Уявляються	Приклади запису		
Константи	Числами	1	–256	10000
	Текстом (стрінгами)	'64'	'–127'	'10000'
Змінні	Іменами	`Irez2`	`NextFunk`	`IOrect`
Функції	Зарезервованими іменами функцій з потрібними типами аргументів	`pred` (5)	`succ` (Irez2)	`sqr`(4)
		`trunc` (0.239)	`abs`(–75)	`ord` (32)

Константи, змінні і функції цілих типів об'єднуються у вирази знаками арифметичних операцій (+, –, *, div (цілочисельне ділення), mod (залишок від цілочисельного ділення)). У виразах у першу чергу виконуються операції у дужках (рос. – скобках), потім операції типу множення (*, div, mod), а потім (+, –). Операції (*, div, mod) виконуються послідовно (табл. 8.23).

Таблиця. 8.23

Виконання обчислень у цілочисельних виразах

Тип даних у виразу	Вираз	Послідовність виконання	
		Частковий вираз	Результат
Арифметичний цілий	–(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

Кількість цифр у відображенні числа цілого типу залежить від місця у пам'яті ПК, що відводиться для його розташування (для його цифр, що мають значення) . Так, якщо Ви бажаєте відобразити ціле число з кількістю цифр, що дорівнює шести, то потрібно обрати тип `Longint` бо інші типи не вмістять таку кількість знаків (див. табл. 8.24).

Таблиця. 8.24

Типи цілочисельних даних і їх величини

Довжина, байт	Назва типу	Діапазон значень 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

У виразах цілого типу можливо використовувати наступні вбудовані арифметичні процедури і функції (табл..8.25). Зверніть увагу на те, що у дужках вказані типи аргументів, а за дужками – тип результату.

Таблиця 8.25

Вбудовані функції для роботи з цілочисельними даними

Найменування функції чи процедури	Дії, що виконуються
Abs(X:integer): integer;	– абсолютне значення аргументу
Sqr(X: integer): integer;	– обчислює квадрат аргументу X
Round(X:real):integer;	– округлення дійсного числа X до найближчого цілого
Trunc(X:real):integer;	– округлення дійсного числа X до цілого з відкиданням дробової частини
Randomize	– ініціалізує випадкове значення вбудованого генератора випадкових чисел
Random(Range:Word);	– якщо вказаний параметр Range – повертає (обчислює) ціле число у діапазоні від 0 до значення (Range-1)
Ord(значення переліченого типу і у тому числі, цілого) : integer;	– повертає номер елемента у послідовності значень аргументу
Pred(значення переліченого типу і у тому числі, цілого):integer;	– повертає номер попереднього елемента у послідовності
Succ(значення переліченого типу і у тому числі, цілого):integer;	– повертає номер наступного елемента у послідовності
Inc(X [,k]); – процедура	– якщо присутній тільки аргумент X, то віднімає від нього одиницю. Якщо присутні X і k, віднімає від X значення k.
Dec(X [,k]); –процедура	– якщо присутній тільки аргумент X, то додає до нього одиницю. Якщо присутні X і k, додає до X значення k.

Для цілих чисел та інших порядкових типів справедливі такі співвідношення:

```
Ord(x)–1 = Ord(Pred(x)), Ord(x)+1 = Ord(Succ(x)).
```

Приклади:

```
Var
j, i, k, p, n, m : integer;
Begin
j := 23; i := 5;
.............................................
k := pred(23); {k набуває значення 22}
 p := succ(k) + ord(i); {p набуває значення 23+5=28}
  dec(j,i); {j набуває значення 23–5=18}
   ink(k); {k набуває значення 22+1=23}
    m := sqr(2)div pred(i); {m набуває значення 4/4=1}
.............................................
end.
```

Коли Ви починаєте вирішувати задачу, то повинні чітко розуміти, які типи даних будете використовувати, бо від цього залежать:

❶ типи даних, що обираються і приймають участь у обчисленнях (`integer, byte, word, real, extended, boolean` та ін.);

❷ припустимі операції над змінними у виразах з даними цих типів (+, –, *, /, div, mod, and, or та ін.);

❸ типи кінцевих значень виразів з обраними даними;

❹ результати обчислень, що повертають вбудовані функції. Бо деякі з них повертають результати, що залежать від типу аргументів.

Таким чином:

❶ Неприпустимо пересилати за допомогою оператора привласнення дані одного типу до змінної іншого. Наприклад, неможливо заслати логічне значення `True` у цілу чи дійсну змінну `Rez`.

❷ Неприпустимо у виразах зі змінними одного типу, застосовувати операції і функції, що належать до інших типів.

Наприклад, у цілу змінну `Irez` неможливо переслати результат дійсної операції (/):

```
Irez  := 2 / cos(0.5) ;
```

❸ Неприпустимо змішувати у виразах дані (константи і змінні) різних типів.

Наприклад, неможливо ділити цілочисельну константу на дійсну константу за допомогою оператора цілочисельного ділення:

```
Irez  := 40 div 4.0 ;
```

Але можливо переслати до логічної змінної результат операції порівняння, бо це є логічне значення.

```
Brez  :=  (5 >= 2) ;
```

Вправи

1. Які числа звуться цілими?

2. Які Ви знаєте цілочисельні вбудовані функції?

3. Чому не можна заносити (привласнювати) значення типу `real` до типу `integer`?

4. Виконайте наступні дії з цілочисельними даними. Перевірте свої обчислення на комп'ютері. Значення вхідних даних завдайте самі.

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;`

ж) `A div 1000;`

з) `A mod 1000;`

і) `A div 100`

к) `A mod 100;`

л) `A mod 2;`

м) `(A mod 10)*((A mod 10) mod 2).`

8.11. Логічні типи даних (Boolean). Операції і вбудовані функції роботи з ними. Конструювання логічних виразів для формування логіки роботи програм на основі п'яти рівнів абстракції

Одним з найбільш важливіших для конструювання логіки програми та організації вибору тих чи інших блоків обчислень у залежності від деяких обставин є логічний тип даних, що у ТП має назву `Boolean`. Він характеризується тільки двома конкретними значеннями:

0 (`False` – неправда);
1 (`True` – правда)

кожне з яких займає у пам'яті ПК лише один байт. Логічні змінні та їх значення розцінюються компілятором у програмах таким чином (у такому контексті):

❶ у логічних виразах, як значення `True` та `False`;

❷ у цілочисельних виразах та у результатах виконання вбудованих функцій роботи з порядковими типами (`ord, succ, pred`), як значення 0 або 1.

У виразах логічні дані використовуються у вигляді:

❶ констант;
❷ змінних;
❸ значень відповідних вбудованих функцій (див. табл. 8.26).

Таблиця 8.26

Вбудовані функції ТП для роботи з логічними змінними і значеннями

Найменування функції	Дії, що виконуються
Ord (значення порядкового типу і у тому числі, цілого) : integer;	– повертає номер елемента у послідовності значень аргументу `Ord(False) = 0, Ord(True ) = 1`
Pred (значення порядкового типу і у тому числі, логічного):integer;	– повертає номер попереднього елемента у послідовності: `Pred (True ) = False`
Succ (значення порядкового типу і у тому числі, логічного):integer;	– повертає номер наступного елемента у послідовності: `Succ (False) = True`

Odd (значення цілого типу): boolean;	– повертає значення True, якщо значення цілого непарне (рос. – нечётное), і значення False, якщо воно парне (рос. – чётное):
	`Odd (-2) = False, Odd (-1) = True,`
	`Odd (0) = False,`
	`Odd (1) = True, Odd (2) = False,`
	`Odd (3) = True, Odd (4) = False` і так далі.

Зверніть увагу на те, що ***логічні значення*** у програмі Ви можете ***отримувати*** у результаті порівняння характеристик і значень різних об'єктів.

Наприклад, небо синє, або ні; дощ йде або, ні; дощ сильний, або ні; 1 менше 5, або ні і так далі. Але для комп'ютера такі загальні характеристики невідомі (синій, сильний, смачний і т. ін.). Для ПК потрібно усі ці характеристики відображати чисельно. Наприклад, не існує логарифма негативного (рос. – отрицательного) числа. Тому, перед обчисленням логарифма у Вашій програмі, якщо Ви бажаєте щоб вона була універсальною, потрібно перевірити, від якого числа Ваша програма буде його обчислювати. Для цього потрібно порівняти вхідне числове значення з нульовим (значенням). Цей конкретний факт не обмежує безліч можливих ситуацій, які Ви повинні будете програмувати. Тому у мові ТП є спеціальні операції порівняння або відношення (табл. 8.27).

Таблиця 8.27

Операції порівняння мови ТП

Математичне позначення операцій порівняння	=	≠	<	>	≤	≥
Позначення цих операцій у мові Турбо Паскаль	=	<>	<	>	<=	>=

Операції порівняння поєднують об'єкти порівняння (константи, змінні та вирази) у ***прості і складні логічні вирази***, але одна змінна (константа, вираз) може поєднуватися лише однією операцією порівняння лише з однією змінною (константою, виразом)! Тобто не можна писати `0<x<9`. Але, вірними простими логічними виразами з точки зору конструкцій мови ТП можуть бути такі:

`5 < 1` {Завжди має значення `False`}

`x >= 4` {При $x \geq 4$ цей вираз має значення `True`}

`a` `<= 'f'` {При опису змінної *`a`* символьним типом (`var a : Char`;) і привласненні *`a`* значення літери d (a:='d';) вираз `a` $\leq$ `'f'` має значення `True`}

`(x*x – b) <> 0` {Має значення `True`, якщо значення виразу $(x^2\text{-}b) \neq 0$}

Але, якщо Вам потрібно сконструювати складний логічний вираз, то прості логічні вирази Ви повинні об'єднувати у складні за допомогою логічних операцій (таблиця 8.28).

Таблиця 8.28

Логічні операції

№ п/п	Назва операції	Математичне позначення	Позначення у ТП
1.	Заперечення (рос. – отрицание)	¬	**NOT**
2.	Логічне множення, І (рос. – логическое умножение, И)	∧	**AND**
3.	Логічне додавання, АБО (рос. – логическое сложение, ИЛИ)	∨	**OR**
4.	АБО, що виключає (рос. – исключающее ИЛИ)	⊕	**XOR**

Логічними операціями (NOT, AND, OR та XOR) звуться операції, котрі можуть виконуватися з логічними константами, змінними, виразами та значеннями (`True` або `False`).

Результати виконання вищевказаних логічних операцій з двома операндами (змінними) `A` та `B`, що мають у різних поєднаннях значення `True` і `False`, наведені у таблиці 8.29.

Таблиця 8.29

Результати виконання логічних операцій NOT, AND, OR та XOR

Логічні аргументи, що приймають різні значення True і False		Результат виконання відповідних логічних операцій з різними комбінаціями значень аргументів A і 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

У побітовій нотації (тобто при взаємодії окремих бітів), таблиця 8.29 приймає наступний вигляд (таблиця 8.30).

Таблиця 8.30

Побітове виконання операцій NOT, AND, OR та XOR

Значення аргументів A і B		Результати виконання логічних операцій з аргументами A і 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 виконанні логічних операцій найвищий пріоритет має унарна операція заперечення (`NOT`). Потім йде операція логічного множення (`AND`), а найнижчий пріоритет мають операції `OR` та `XOR`.

Складність розуміння взаємодії вищеописаних компонентів міститься у тому, що виконання арифметичних і текстових операцій з даними одного і того ж типу, як правило, дає результат такого ж типу, а порівняння значень тих же типів дає натомість логічні значення (табл. 8.31).

Якщо за основу абстрактного уявлення взаємодії різних сутностей (рос.-сущностей) прийняти логічне значення (яке існує тільки у комп'ютері!), то перший рівень абстракції міститься у порівняннях двох об'єктів (значень змінних або констант), тобто у атомарних ***логічних діях***. Вони дають у результаті ***логічні значення***, а з іншого боку, можуть об'єднуватися у вирази, які можуть містити досить багато таких малих елементарних порівнянь. Окрім того, для перевірки у програмах результатів виконання окремих блоків, логічні значення можуть змінюватися і зберігатися у логічних змінних. Ці об'єкти

також можуть приймати участь у різної ступені складності логічних виразах. І так далі.

Таблиця 8.31

Різні типи результатів виконання різних операцій з деякими типами даними

Тип даних, що взаємодіють	Вираз і результат	Тип результату	Порівняння тих же даних	Тип результату логічний!
Дійсний	3.1 + 6.91 = 10.1	Дійсний	3.1 < 6.91	`True`
Цілий	8 div 4 = 2	Цілий	8 <= 4	`False`
Стрінговий	'con' + 'tent' ='content'	Стрінговий	'con' > 'tent'	`False`

Примітка: Не можна порівнювати поміж собою дані різних типів!!!

Наведемо приклад логічного оператору зі складним логічним виразом для демонстрації п'яти рівнів логічних абстракцій (див. фрагмент програми):

```
…
Var
  A : Boolean;
  i, j : integer;
Begin
  i := 1;
  j := 10;
  A :=True;
If A AND (5 > i) OR NOT (4 > j) AND (-5 <> 4) Then
  j := 2;
End.
```

У складному (!) логічному виразі

```
A AND (5 > i) OR NOT (4 > j) AND (-5 <> 4)
```

можна виділити наступні п'ять рівнів абстракцій уявлення елементів порівняння і результатів порівняння у програмах:

Перший рівень	***Операції порівняння*** **(>,** <> та інш.)
Другий рівень	Логічні ***значення*** **(***True, False***)**
Третій рівень	Логічні ***змінні (що описуються: Var A : boolean; і зберігають логічні значення)***
Четвертий рівень	Логічні ***операції (AND, OR, NOT та ін.)***
П’ятий рівень	Логічні ***вирази*** **((**`5 > i), (A AND (5 > i))` **та ін.)**

Вправи

1. Які вирази є простими логічними виразами?
2. Які вирази є складеними логічними виразами?
3. Які операції відносини між величинами Ви знаєте?
4. Які логічні операції застосовуються при обчисленні логічних виразів?

5. Який порядок обчислення складених логічних виразів?

6. Обчислити значення логічних виразів і реалізувати рішення в програмі:

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. Поясніть помилки в наступних записах:

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. Вказати порядок виконання операцій при обчисленні виразів:

а) A and b or not c and d;

б) (X >= 0) or T and Odd(X) or (Y*Y <> 4).

9. Обчислити значення виразів і перевірити свої обчислення у програмах:

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. Записати у виді логічного виразу наступні умови:

а) Чотирикутник ABCD є квадратом.

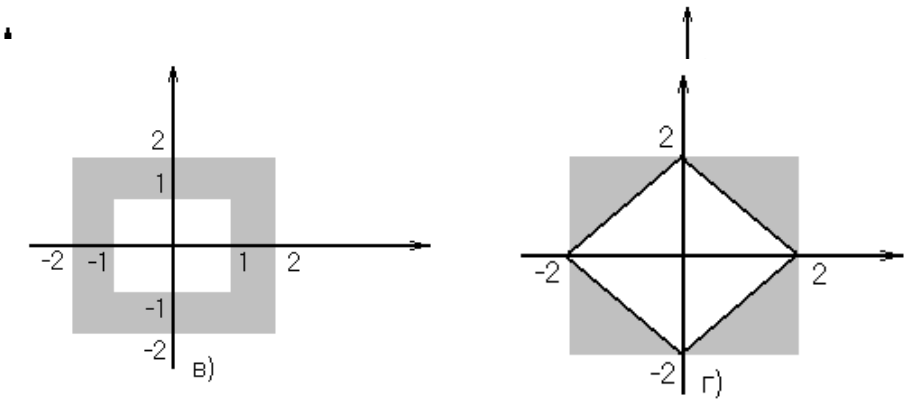
б) Чотирикутник ABCD є ромбом.

в) Чотирикутник ABCD є трапецією.

г) Трикутник ABC рівнобедрений.

д) Трикутник ABC рівносторонній.

11. Скласти програму, що визначає належить крапка A(x,y) заштрихованої області чи там її немає. Якщо належить, вивести на екран значення TRUE, інакше - FALSE.



8.12. Використання логічних операцій та операцій відношення для запису складних умовних виразів

Досить часто при обчисленні математичних виразів у програмах зустрічаються складні умови урахування розподілу аргументів функціональних залежностей на числовій осі. Тоді у логічних операторах потрібно відповідним чином конструювати логічні вирази, які відповідають кожному конкретному випадку.

Розглянемо записи таких логічних виразів для найважливіших прикладів.

1. Аргумент ***x*** належить до відрізку [0,4], що інакше (математично) записується так

$0 \le x \le 4$ (рис. 8.30):

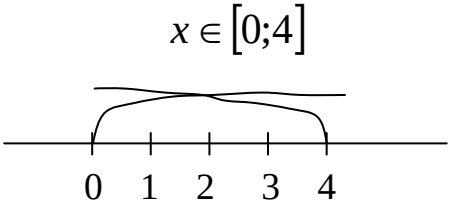


Рис. 8.30. Інтервал типу "відрізок"

Тобто, якщо у змінної ***x*** буде міститися значення –3 або +10, програма повинна "зрозуміти", що ці значення не належать до вказаного вище відрізку числової вісі (рос.-оси). Навпаки, якщо ***x*** має значення 2, то програма зрозуміє, що ми влучили у відрізок.

Певно Ви пам'ятаєте, що в мові ТП ми не в змозі цілком записати математичну умовну конструкцію `0<=x<=4`, яка б могла вирішити нашу проблему. Треба розкласти її на атомарні порівняння, а потім об'єднати у єдину формулу. І от тут нам на допомогу приходять логічні операції. Щоб вірно вибрати потрібну логічну операцію, розглянемо ймовірні варіації змінної ***x*** на інтервалі (-∞,+∞) з урахуванням умови $0 \le x \le 4$ поодинці та взаємодію логічних виразів (табл. 8.31).

Порівнюючи отримані результати з таблиці 8.31, можна побачити, що на мові ТП складний логічний вираз $0 \le x \le 4$ запишеться у такому вигляді:

```
(x >= 0) AND (x <= 4)        (8.3)
```

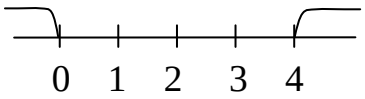
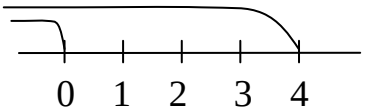
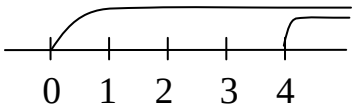
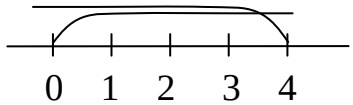
Наприклад, для різних значень, що можуть міститися у змінній x, значення логічного виразу (8.30) дорівнюють (при x = –3, 2, 6):

```
(-3 >= 0) AND (-3 <= 4) дорівнює 0;
 (2 >= 0) AND  (2 <= 4) дорівнює 1;
 (6 >= 0) AND  (6 <= 4) дорівнює 0.
```

Таким чином, комп'ютер розуміє, що коли вираз дорівнює 0 (False), чергове значення не належить відрізку, а коли дорівнює 1 (True) – значення належить відрізку.

Таблиця 8.31

Значення простих логічних виразів

№ п/п	Атомарні (прості) логічні вирази	Їхні логічні значення для аргументу ***x***, що належать відрізку [0,4], (при ***x*** = 2)	Взаємодія отриманих логічних значень	Графічне зображення
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		

Примітка: ❶ Нагадуємо, що взаємодія логічних значень, отриманих у результаті порівнянь, які об'єднані логічною операцією OR, дає зовсім інші результати:

(0 OR 0) = 0, (0 OR 1) = 1, (1 OR 0) = 1, (1 OR 1) = 1.

❷ У таблиці значення 0 еквівалентно FALSE, а 1 – TRUE.

2. Якщо ж Вам потрібно враховувати вимоги, щоб значення аргументу ***x*** знаходилися за межами відрізку [0,4], тобто 0>***x***>4, то використовуючи допоміжну таблицю отримаємо (див. табл.8.31).

Порівнюючи отримані результати з таблиці 8.31, можна побачити, що на мові ТП складний логічний вираз 0 > ***x*** > 4 запишеться у такому вигляді:

```
(x < 0) OR (x > 4)        (8.4)
```

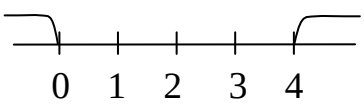
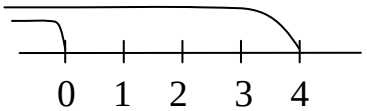
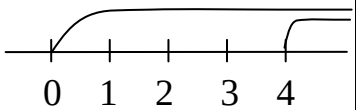
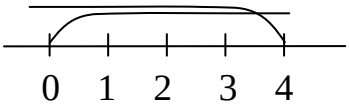
Для різних значень, що можуть міститися у змінній x, значення логічного виразу (8.4) дорівнюють (при x = –3, 2, 6):

```
(-3 < 0) OR (-3 > 4) дорівнює 1;
( 2 < 0) OR ( 2 > 4) дорівнює 0;
( 6 < 0) OR ( 6 > 4) дорівнює 1.
```

Таким чином, комп'ютер розуміє, що коли знання виразу дорівнює 0, чергове значення аргументу ***x*** належить відрізку, а коли дорівнює 1 – значення ***x*** знаходиться за межами відрізку і умова 0 > ***x*** > 4 виконується.

Таблиця 8.31

Значення простих логічних виразів

№ п/п	Атомарні (прості) логічні вирази	Їхні логічні значення для аргументу ***x***, що не належать відрізку [0,4], (***x*** = 6, або –6)	Взаємодія отриманих логічних значень	Графічне зображення
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. Розглянемо випадок, коли аргумент ***x*** повинен належати водночас двом інтервалам: [2,5] і [10,14] (рис. 8.31):

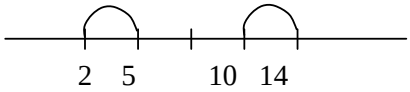


Рис. 8.31. Належність значень змінної одночасно двом відрізкам

Для цього випадку вираз на мові ТП має вигляд:

```
((x >= 2) AND (x <= 5)) OR ((x >= 10) AND (x <= 14))
```

4. Для випадку, коли аргумент ***x*** повинен знаходитись за межами інтервалів [2,5] та [10,14] можна записати такий складний логічний вираз:

```
((x < 2) OR (x > 5)) AND ((x < 10) OR (x > 14))
```

Зрозуміло, що використовуючи логічні операції та операції порівнянь можна записати умови довільної ступені складності.

Приклад: Записати логічний вираз на мові ТП для обчислення функції:

$$y(x) = \begin{cases} \sqrt{x^2+1}, & 1< x <6 \\ \dfrac{1}{x^3}, & 1\geq x \geq 6 \end{cases}$$

Такий вираз можна записати за допомогою двох різних форм, які дадуть однаковий результат (див. перший і другий логічні оператори на мові ТП):

```
Program VariousIf;
Var
  y, x : real;
Begin
  x := 2.0;
  if (x > 1.0) AND (x < 6.0) then {перший оператор}
    y := sqrt(x*x+1)
  else
      y := 1/(x*x*x);
  if (x <= 1) or (x >= 6) then    {другий оператор}
    y := 1/(x*x*x)
  else
    y := sqrt(x*x+1);
End.
```

Вправи

1. Дано координати вершин чотирикутника $A(x_1,y_1)$, $B(x_2,y_2)$, $C(x_3,y_3)$ і $D(x_4,y_4)$. Визначити, чи є даний чотирикутник ромбом. Для обчислень скористайтеся формулою відстані між двома точками $A(x_1,y_1)$ і $B(x_2,y_2)$: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

2. Дано три координати трьох вершин трикутника $A(x_1,y_1)$, $B(x_2,y_2)$ і $C(x_3,y_3)$. Визначити, чи є даний трикутник рівностороннім. Для обчислень скористайтеся формулою відстані між двома точками $A(x_1,y_1)$ і $B(x_2,y_2)$: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

3. Дано три числа. Знайти найменше з них і вивести його значення на екран.

4. Дано три числа. Знайти найбільше з них і вивести його значення на екран.

5. Скласти програму визначення номера найменшого елемента з трьох x_1, x_2, x_3.

6. Розташуйте три числа *a, b, c* у порядку збільшення їх значень і виведіть на екран.

7.Скласти програму знаходження множення двох найбільших з трьох чисел *x, y, z* .

8.13. Управляючі структури (оператори) мови ТП. Прості оператори.

Оператори мови ТП призначені (рос. - предназначены) для опису дій, котрі будуть виконані при реалізації розробленого заздалегідь алгоритму. Структурно, вони є елементами мови, яка служить для організації проведення операцій над даними (фактично вони є глаголами мови ТП і відповідають на питання “що зробити?”: привласнити, повторити і таке інше). Але потрібно постійно пам'ятати, що розробка алгоритму завжди провадиться на основі знання і умілого використання потрібних типів даних і операторів, що їх можуть допомогти обробити.

Оператори (згідно синтаксису) розділяються на дві групи:

❶ прості оператори;

❷ складні оператори.

Оператори відокремлюються один від одного "розподілювачем" – "**;**", тобто крапкою з комою. Наприклад:

```
... x:=0.324; r:=cos(x); t :=sqrt(sqr(x*x+r)+1; ...
```

Тому взагалі у складеному (рос.–составном) операторі (`begin end;`) перед ключовим словом `end` знак "**;**" можна не ставити. Але якщо він стоїть, то вважається що після розподілювача "**;**" розташований ***пустий оператор***.

```
... begin ... x:=0324; r:=cos(x);      end;
```

Пустий оператор

Прості оператори це оператори, які не містять у собі інших операторів (табл. 8.32).

Таблиця 8.32

Прості оператори мови ТП

Назва оператора	Форма запису оператора
Оператор привласнення (рос. – присваивания)	`A := B;`
Оператори виклику процедури	`writeln('Уведить w: ',w); readln;`
Оператор переходу	`goto` (Не розглядається далі, тому що не відповідає вимогам структурного програмування!)
Примітка. Зверніть увагу, що оператор процедури, на відміну від інших простих операторів (`:=`), що записуються завжди однаково (!), має тільки загальну ***форму*** запису: **<ім'я процедури і (можливо!) (список фактичних параметрів)> ;** а ***зміст*** – різний! Бо усі процедури, що виконують різні дії, мають: 1. різні імена; 2. різну кількість фактичних параметрів (або можуть зовсім їх не мати!). Причетність групи символів (які відображують ім’я та список фактичних параметрів у дужках) до операторів процедури освідчує символ (**;**) у їх кінці.	

Оператор привласнення (рос. – присваивания) **":="** служить для привласнення значень змінним. Іншою мовою, він дозволяє заносити обчислені за виразами результати у конкретні місця пам'яті комп'ютера, які для зручності пойменовані для подальшого використання значень, що у них містяться.

Точне значення оператора привласнення можна описати так: ***замінити (замістити) попереднє значення змінної, ім'я якої стоїть у лівій частині оператора, на значення виразу, що стоїть у правій частині оператору привласнення.*** Таким чином, оператор

```
A := B + C;
```

означає те, що потрібно обчислити суму значень змінних `B` та `C` та замінити цією сумою попереднє значення змінної `A`. Специфічні властивості оператора привласнення виявляються дуже яскраво у такої формі:

```
N := N + 1;
```

Сенс цього виразу полягає у тому, що змінній `N` потрібно привласнити її попереднє значення, збільшене на 1.

Оператор процедури. Виконання оператора процедури призводить до активізації дій, що описані у її тілі. Найчастіше у програмах використовуються процедури уводу даних у змінні програми з екрану (`read, readln`) і виводу даних на екран комп'ютера (`write, writeln`). Їх досить часто звуть операторами уводу–виводу:

```
... writeln('Уведить a,e: ');  readln(a,e); ...
```

Після виконання цієї групи операторів на чорному екрані DOS з'явиться стрічка тексту `'Уведіть a,e: '` і комп'ютер буде чекати від користувача уводу двох значень даних, що відповідають опису їх у програмі. Наприклад, якщо вони описані як дійсні змінні, то треба буде увести підряд, *але розподілені пробілом*, два дійсних значення, припустимо такі: 2.0 0.291.

Слід додати, що до складу програми можуть бути уключені коментарі – які уявляють собою стрічки тексту, що пояснюють зміст дій програміста, але не впливають на хід виконання самої програми. Добрим стилем програмування у софтверних[23] фірмах (тобто тих, які виробляють програмне забезпечення) вважається такий, коли 45% тексту програми складають тільки коментарі!

Коментарі містяться у спеціальних дужках, перша з яких є відкриваючою, а остання – закриваючою (табл. 8.33).

Таблиця 8.33

Приклади коментарів

	Відкриваюча дужка	Коментар	Закриваюча дужка
Три різних типа дужок для запису коментарів	**{**	Перший коментар	**}**
	/*	Другий коментар	***/**
	(*	Третій коментар	***)**

[23] Софтверний (від англійського слова "software") – той, що має відношення до виробництва програмного забезпечення.

Вправи

1. Які оператори ТП звуться простими?
2. Як відділяються оператори один від одного?
3. Який оператор зветься “пустим”?
4. Який зміст має оператор привласнення?
5. Чим оператор процедури відрізняється від інших простих операторів?
6. Що таке коментар і для чого він застосовується?

8.14. Складні (структурні) оператори управління виконанням алгоритмів. Складовий оператор `begin ... end`

Складні (структурні) оператори уключають у себе інші оператори і управляють послідовністю їх виконання (таблиця 8.34).

Таблиця 8.34

Складні оператори мови ТП

Назва оператора	Форма запису оператора
Складовий (рос. – составной) оператор або структурний оператор, котрий ще називають операторними дужками	`begin ... end;`
Умовний оператор	`if ... then ... else`
Оператор вибору	`case ... of ...else ... end`
Оператор циклу з параметром	`for ... to ... do`
Оператор циклу з передумовою	`while ... do`
Оператор циклу з післяумовою	`repeat ... until`

Іноді синтаксис мови ТП потребує, щоб у деякому місці програми знаходилося не більше одного оператора, а для вирішення задачі потрібно їх декілька. ***Складовий оператор (рос.-составной оператор)*** уключає до себе їх усіх, але вважається комп'ютером за один (рис.8.32).

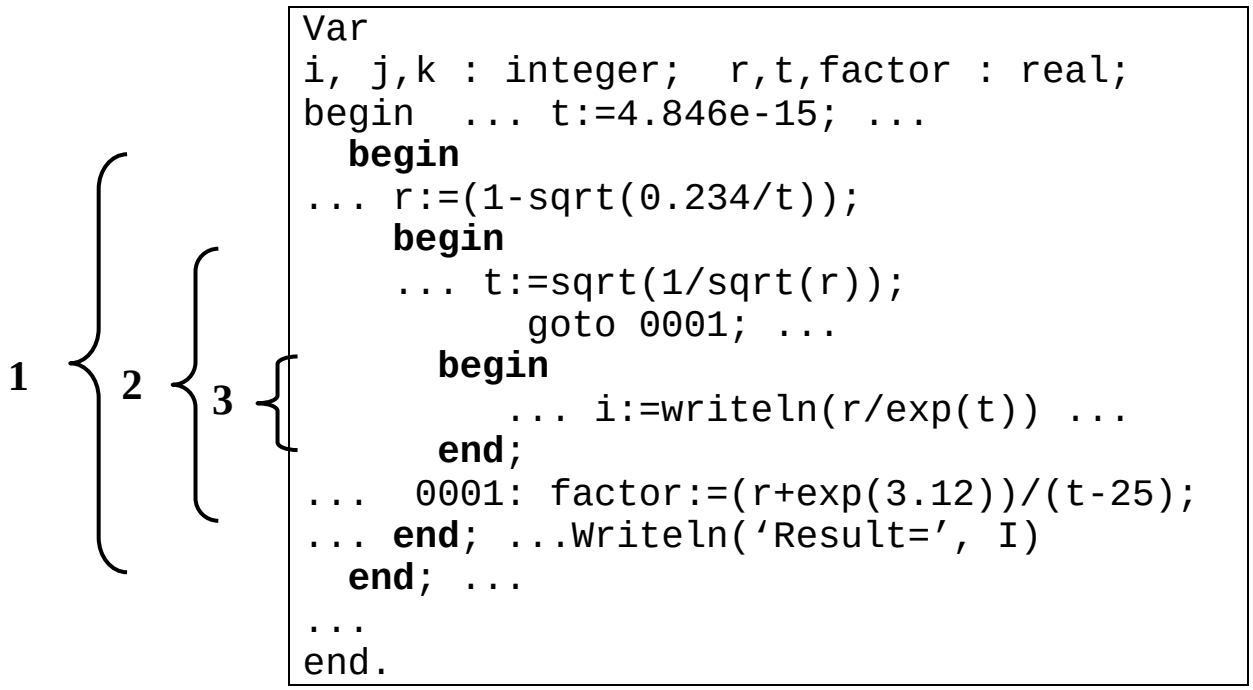


Рис. 8.32. Рівні (1, 2 і 3) включення операторів у операторні дужки `begin` та `end`

Тобто, складовий оператор об'єднує групу операторів у єдине ціле, після чого їх можна вважати за один оператор. Таким чином, складовий оператор складається з послідовності операторів, що об’єднуються і розташовані поміж ключовими словами (операторними дужками) `begin` та `end:`

❶ дужки, що відкриваються – `begin`;

❷ дужки, що закриваються – `end`.

Не слід плутати операторні дужки з аналогічними зарезервованими словами, якими починається і закінчується розділ операторів програми. Треба також уважно стежити, щоб кількість операторних дужок, що відкриваються дорівнювала кількості дужок, що закривається за відповідними групами операторів (рис. 8.32).

Наведемо список складних операторів яким потрібен складовий оператор для об’єднання простих операторів у групи, які вважаються за єдине ціле (табл. 8.34).

Таблиця 8.34

Складні оператори, яким потрібен складовий оператор

Назва оператора	Форма запису оператора

Умовний оператор	`if ... then`
	`begin <оператори> end`
	`else`
	`begin <оператори> end;`

Оператор вибору	`case ... of`
	`<МІТКА> : begin <оператори> end`
	`else`
	`begin <оператори> end`
	`end;`

Оператор циклу з параметром `for`	`for ... to ... do` `begin <оператори> end;`
Оператор циклу з передумовою `while`	`while ... do` `begin <оператори> end;`

Вправи

1. Які оператори звуться складними?
2. Для чого використовуються складові оператори?
3. Яким складним операторам потрібні складові оператори, а яким ні?
4. Перелічить усі складні оператори мови ТП.

8.15. Оператори розгалуження алгоритмів. Умовний оператор `if`. Оператор вибору `case`

Обчислювальний процес зветься розгалуженим (рос. – разветвляющимся), якщо у залежності від виконання визначених умов виконуються ті чи інші групи операторів. Кількість груп операторів і умови, щодо яких вони виконуються, конструює сам програміст. Кожна така група операторів зветься гілкою алгоритму обчислень. Вибір тієї чи іншої гілки здійснюється вже при виконанні програми у результаті перевірки запрограмованих умов, які залежать від властивостей вхідних даних і деяких проміжних (рос. – промежуточных) результатів.

В Турбо Паскаль включені фактично два умовні оператори `if` і `case`. Вони мають повну або неповну форму (див. далі), але перший зветься умовним оператором, а другий – оператором вибору.

Умовний оператор **`if`** призначений для виконання або невиконання різних груп операторів у залежності від виконання або невиконання умов, що задаються користувачем. Він має так звану неповну і повну форми. Їх синтаксис:

Неповна форма: `if` <логічний вираз умови> `then` <оператор P1>;
Повна форма: `if` <логічний вираз умови> `then` <оператор P1>
`else` <оператор P2>;

Логічні вирази конструюються за допомогою наступних елементів:

❶ дужок ();
❷ логічних значень (`false, true`);
❸ логічних змінних, що уявлені їх іменами (`Switch, tt1` та інш.);
❹ операцій порівняння (=, <, >, <>, <=, >=);
❺ логічних операцій (`not, or, and`).

Оператори P1 та P2 можуть бути:

❶ простими операторами;
❷ простими і складними операторами, що об'єднані операторними дужками `begin … end;`
❸ складними операторами і, у тому числі, іншими умовними операторами.

Оскільки оператори у мові ТП розділяються символом "**;**" (крапка з комою), в середині складного логічного оператора цей символ застосовувати НЕ МОЖНА!

помилка!

```
if A > B then C := A ; else C := B;
```

У цьому прикладі крапка з комою, що стоїть перед службовим словом `else` закінчує текст оператора `if`. А це призводить до синтаксичної помилки, оскільки оператора, що починається з службового слова `else` – немає.

Для розгляду прикладу використання логічного оператора складемо програму обчислення наступного виразу (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 обчислення виразу (8.5) на мові ТП може виглядати так:

```
Program Sign_X;

Var
  x, y : real;
Begin
    writeln('Vvedite x');
       readln(x);
  if (x > 0) and (x < 1) then
                              y := sqr(x)
     else
  if (x > 1) and (x < 3) then
                              y := sqrt(x)
     else
        y := 1/sqrt(abs(x));

   writeln('Znachenie Y=', y:8:3);

end.
```

Після зарезервованих слів `then` и `else` повинен стояти тільки один (!) оператор. Тому, якщо після них потрібно виконати декілька операторів, треба їх об'єднати у один операторними дужками `begin` та `end.` Існують, так звані, вкладені (рос. – вложенные) оператори `if` (вкладені іфи). Це значить, що за зарезервованим словом `else` міститься наступний оператор `if.`

Наприклад, нехай потрібно обчислити значення функції `y(x) = sign(x)` згідно виразу (8.6).

$$y\,(x) = sign\,(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \qquad (8.6)$$

Алгоритм послідовності обчислень за цією формулою з використанням вкладеного оператору `if` може виглядати так.

```
  Label 10;
Var
   x, y : real;
Begin
   writeln('Vvedite x');
   readln(x);
  if (x < 0) then
      y := -1
  else
    if (x = 0) then
       y := 0
    else  y := 1;
  writeln('Znachenie Y=', y:8:3);
end.
```

При необхідності конструювання складних логічних умов виникають синтаксичні складності коректного використання оператора `if`, коли потрібно вкладати один такий оператор `if` у інший. Якщо його відокремлювати операторними дужками `begin ... end` великих проблем не виникає.

Наприклад:

```
if умова then
  begin
    оператор;
    if умова then
      оператор
    else
      оператор;
    оператор
  end
else
  begin
    оператор;
    if умова then
      оператор
  end;
```

Однак, якщо вкладений оператор `if` є єдиним оператором у гілці (рос. – ветви) альтернативи, то може виникнути неоднозначність: якому `if` відповідає гілка `else.`

Наприклад:

```
if умова then
  if умова  then
    оператор
    else
      оператор; {Невірне використання оператора if}
```

У таких випадках повинно виконуватися наступне правило.

Ключове слово `else` зв'язується з найближчим, що стоїть перед ним ключовим словом `if`, котре ще не було зв'язане з ключовим словом `else`.

Якщо у передуючому прикладі більш конкретніше позначати структуру вкладеності і записати `else` на одному рівні з `then`, якому воно відповідає, то можемо отримати такий фрагмент програми.

```
if умова then
  begin
    if умова then
      оператор
  end
else
  оператор;
```

Тому треба запам'ятати наступне обмеження. Оператор, що розташовується безпосередньо після службового слова **`then`** не може бути ***умовним***, у той час як оператор, що розташовується після **`else`** може бути ***будь-яким***, і, зокрема, умовним. Це обмеження виникає, тому що компілятору не зрозуміло до якої з умов ***відноситься*** `else` (перевірте на комп'ютері) (табл.8.35):

Таблиця 8.35

Приклади конструювання вкладених операторів `if`

НЕВІРНО	ВІРНО (другий оператор `if` входить до складеного оператора)	ВІРНО (оператор `if` використовується по праву сторону від 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 використанні умовного оператора `if` після `else` ніяких двозначностей не виникає.

Прикладом, коли логічний вираз у операторі `if` має більш складну структуру, може бути задача визначення, чи можливо побудувати трикутник з довжин відрізків, що задаються у змінних: x, y, z (x>0, y>0, z>0).

Такий умовний оператор має вигляд.

```
if (X+Y>Z) AND (X+Z>Y) AND (Y+Z>X) then
  Writeln('ТРИКУТНИК ПОБУДУВАТИ МОЖЛИВО')
else
  Writeln('ТРИКУТНИК ПОБУДУВАТИ НЕМОЖЛИВО');
```

Оператор вибору `case` служить для переключення на виконання груп операторів у залежності від значення деякого цілочисельного перемикача (рос. – переключателя) або виразу. Синтаксис повної його форми такий:

```
case <перемикач> of
<мітки M1> : <оператори P1>;
< мітки M2> : <оператори P2>;
….....……
< мітки MN> : <оператори PN>;
else < оператори PM>  {альтернатива}
end;
```

Де:

– перемикач – може бути цілочисельною змінною або цілочисельним виразом (`i, Iswitch, 2*i, abs(j)` та інш.).

– мітки M1 ... MN – можуть відображатися:

❶ Одним цілочисельним значенням (3).

❷ Діапазоном цілочисельних значень (1..27).

❸ Переліченням цілочисельних констант (2,4,10,75).

❹ Набором комбінацій елементів перших трьох пунктів (1,3..15,45,54..71).

– оператори P1…PN, PM – це оператори аби якої степені складності, але узяті у операторні дужки `begin ... end`.

Значення функції, що було обчислене за допомогою умовного оператору `if` у попередньому розділі за формулою (8.5) можна реалізувати за допомогою оператора `case` таким чином.

```
Var
   x, y : real; i : integer;
Begin
  writeln('Vvedite 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('Znachenie Y=', y:8:3);
end.
```

Вправи

1. Дани два числа. Замінити друге число нулем, якщо воно більше першого, і залишити його колишнім, якщо це не так.

2. Знайти найменше з трьох даних чисел.

3. Знайти найбільше з трьох даних чисел.

4. Дано три числа. Звести в квадрат ті з них, значення яких ненегативні. Негативні числа залишити без зміни.

5. Дано три числа a, b і c. З'ясувати, чи вірно, що a<b<c. Відповідь вивести на екран у текстовій формі: «Вірно» чи «Невірно».

6. Написати програму для обчислення різних значень функції y:

$$y=\begin{cases}\sin x, & \textit{при} \quad x\le 0\\ \operatorname{arctg} x, & \textit{при} \quad 0<x\le \pi/4\\ \log_2 x, & \textit{при} \quad \pi/4<x\le 32\\ 1/x, & \textit{у останніх випадках}\end{cases}$$

Протестувати програму при різних значеннях аргументів.

7. Написати програму для обчислення різних значень функції z:

$$z=\begin{cases}\ln(x(, & \textit{при} \quad x<-\pi\\ \sin x+\cos 2x, & \textit{при} \quad -\pi\le x<\pi\\ x^3+1, & \textit{при} \quad \pi\le x<10\\ \dfrac{x+1}{x^2+8}, & \textit{при} \quad 10\le x<100\\ \ln x, & \textit{у останніх випадках}\end{cases}$$

Протестувати програму при різних значеннях аргументів.

8. Написати програму для обчислення різних значень функції z:

$$z=\begin{cases}\operatorname{arctg}\dfrac{x}{y}, & \textit{при} \quad y\ne 0,(x(>(y(\\ \arcsin\dfrac{x}{y}, & \textit{при} \quad y\ne 0,(x(\le(y(\\ 0, & \textit{у останніх випадках}\end{cases}$$

Протестувати програму при різних значеннях аргументів.

9. Алгоритми програм 6, 7, 8 реалізувати за допомогою оператора вибору.

8.16. Циклічні обчислювальні процеси і оператори циклів. Цикли з параметром. Оператор циклу з параметром `for`

Коли обчислювальний процес містить багаторазові дії за одними і тими ж математичними залежностями, але для різних значень змінних, що до них належать, його звуть циклічним. Наприклад, якщо потрібно обчислити суму $\sum_{n=1}^{10} a^n$ (8.7), то вона включає такі складові (табл. 8.36).

Таблиця 8.36

Складові обчислення суми за формулою (8.7)

Зміни значень ступеню n	Елементи суми	Інший запис елементів суми і їхніх компонентів
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
Елементи додавання	$a^1 + a^2 + a^3 + a^4 + a^5 + a^6 + a^7 + a^8 + a^9 + a^{10}$	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
Підсумковий результат додавання	$\sum_{n=1}^{10} a^n$	$\sum_{n=1}^{10} a^n$

Зрозуміло, що якщо ми знаємо конкретну кількість цих елементів і як вони конструюються, процес можна запрограмувати досить легко. Тобто треба циклічно десять разів провести визначені операції множення і додавання. І для виконання таких багаторазових дій призначені декілька спеціальних операторів мови Турбо Паскаль.

Отже, групи операторів, що багаторазово повторюються, звуться *циклами*, а змінні, що міняють свої значення у циклі – *змінними* циклу.

Алгоритм циклічної структури у найбільш загальному вигляді повинен містити такі фрагменти дій:

❶ підготовку циклу: ініціалізація (завдання початкових значень) змінним циклу;

❷ тіло циклу: дії, що повторюються у циклі для різних значень змінних циклу;

❸ модифікацію (зміни) значень змінних циклу перед кожним новим його повторенням;

❹ управління циклом: перевірку вимог продовження (або закінчення) циклу і перехід на початок тіла циклу, або вихід з нього.

Одним з прикладів простого циклічного процесу є задача табулювання значень функції однієї змінної y = f(x).

Для її обчислення завдається початкове X_0 і кінцеве X_N значення аргументу X цієї функції, а також кількість K значень функції, які потрібно обчислити на усьому відрізку поміж X_0 та X_N при додатково обчислених на ділянці решти аргументів X_i. Вищенаведені дані дають змогу обчислити крок h, з яким змінюються значення аргументу X на обраній ділянці від X_0 до X_N.

Треба пам'ятати, що кількість K значень аргументів на одиницю більше за кількість відрізків N, на які, як правило, поділяють ділянку поміж X_0 та X_N. Тобто N = K – 1. Наприклад, у таблиці 8.37 задамо наступні вирази і значення для обчислення функції, що обчислює піднесення (рос. – возведения) аргументу X у ступень 1,2 (тобто одну цілу і дві десятих).

Таблиця 8.37

Складові процесу обчислення значень аргументів і функцій на відрізках ділянки $[X_0,X_N]$

Значення	Величина	Вираз/Значення
Функція	$Y = f(X)$	$X^{1,2}$
Початкове значення аргументу	X_0	1
Кінцеве значення аргументу	X_N	5
Кількість відрізків на ділянці	N	4
Кількість значень функції	K	5
Крок зміни аргументу функції	h	$(X_N – X_0)/N$ =1

У таблиці 8.38 наведені обчислені значення функції $Y(X)=X^{1,2}$ (X у ступеню 1.2), обчислені на ділянці [1,5], яка поділена на чотири частини з кроком h= (5–1)/4 = 1. У таблиці 8.38 також показано, що кожне наступне значення аргументу X можна обчислювати двома різними методами. Або додаючи до кожного попереднього значення аргументу X величину кроку h: $X_{i+1} = X_i + h$, або збільшуючи початкове значення аргументу X_0 на кількість кроків, що кратна кількості відрізків, передуючих значенню X_i: $X_i = X_0 + i*h$.

Таблиця 8.38

Значення аргументів X і значень функції $Y=f(X)=X^{1,2}$

Значення аргументів X	Значення функції $X^{1,2}$
$X_0 = 1$	1.0
$X_1 = X_0 + h = 1 + 1 = 2$	2.3
$X_2 = X_1 + h = 2 + 1 = 3$, або $X_2 = X_0 + 2h = 1 + 2 = 3$	3.7
$X_3 = X_2 + h = 3 + 1 = 4$	5.3

$X_4 = X_3 + h = 4 + 1 = 5$, або $X_4 = X_0 + 4h = 1 + 4 = 5$	6.9

Виходячи з цих значень аргументів можна обчислити практично аби яке значення функції однієї змінної (sin x, cos x, e^x, (a+bx), x^2, $\sqrt{x}$ та інш.).

На рисунку 8.33 наведена схема розташування вищенаведених об'єктів у декартових осях координат X та Y.

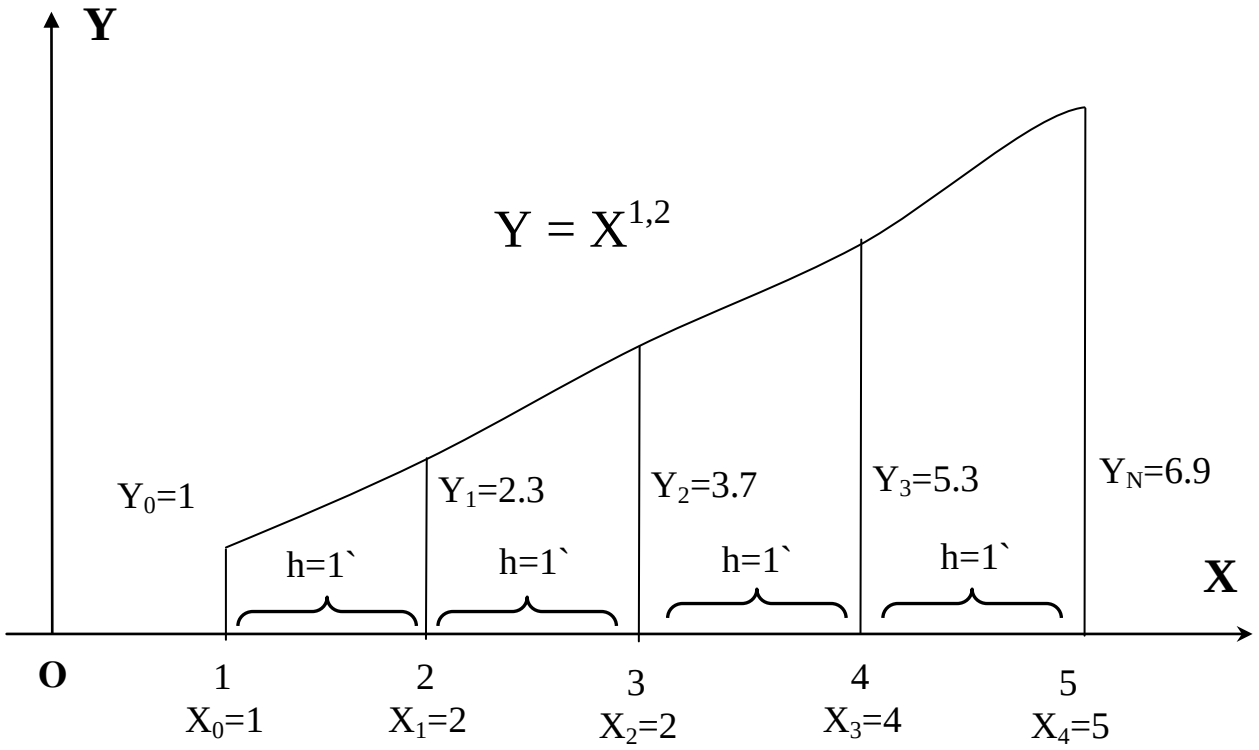


Рис. 8.33. Розташування та значення абсцис X та значень ординат Y для заданих значень і виразів

Таким чином цикли можна конструювати, якщо визначені:

❶ початкове X_0 та кінцеве X_N значення аргументу;

❷ крок h зміни аргументу;

❸ кількість N відрізків на ділянці поміж X_0 та X_N;

❹ кількість K=N+1 значень аргументів X.

Умовами виходу з циклу можуть бути:

❶ вичерпання кількості N обчислених значень функції Y=f(X) для усіх N значень аргументів $X_0, X_1, \dots, X_N$.

❷ перевищення поточним значенням аргументу X кінцевого значення аргументу X_N, який моделюється.

Програмно цикл можна реалізувати за допомогою:

❶ операторів привласнення (`:=`);

❷ умовного оператору (`if … then … else`);

❸ оператору безумовного переходу (`goto`).

Але у мові ТП для цієї цілі існують спеціальні оператори циклу, котрі виключають необхідність конструювання програм за допомогою інших операторів.

Окрім того, оператори циклу забезпечують більш компактну, прозору, простішу форму запису алгоритмів, а також дають змогу створювати більш ефективні програми.

Отже, Ви повинні дуже чітко розуміти, що існують ***два*** типи циклів:

❶ з ***наперед відомою*** кількістю повторень;

❷ з ***невідомою наперед*** кількістю повторень.

Цикли першого типу називають також циклами з лічильником (рос. – счётчиком) або з параметром, маючи на увазі, що цей параметр повинен змінюватися у ході обчислювань. Число повторень тіла циклу у цьому випадку підраховується за допомогою спеціальної змінної (лічильника), для якої відомі початкове і кінцеве (порогове) значення та значення шагу її зміни. Управління цим циклом здійснюється за допомогою порівняння поточного значення лічильника з заданим пороговим (максимальним або мінімальним) значенням. Змінну-лічильник часто звуть параметром циклу, а самий цикл – циклом з параметром.

Оператор циклу з параметром у мові Турбо Паскаль є управляючою структурою, яка задає повторення процесу виконання одного оператора чи групи операторів у межах зміни значення деякого параметра поміж двох цілих чисел, з яких одне є початковим, а інше – кінцевим. Параметр, що пробігає у своїх значеннях від початкового до кінцевого, зветься управляючим параметром циклу (УПЦ). Тип УПЦ завжди ***цілий*** (!), а зміна його провадиться з шагом тільки 1 (або –1).

Загальний вигляд оператора циклу з параметром такий:

`for … to … do` (або `for … downto … do`).

Сінтаксис цих двох форм даного оператора такий:

`for` <ім'я УПЦ := початкове значення> `to` <кінцеве значення> `do`
<оператор>;
`for` <ім'я УПЦ := кінцеве значення> `downto` <початкове значення> `do`
<оператор>;

При використовуванні циклу `for … to … do` потрібно мати на увазі наступне:

❶ у тілі циклу може стояти тільки один оператор, а при необхідності застосування більшої їх кількості, останні об'єднуються у складений операторними дужками `begin` та `end;`

❷ в середині тіла циклу не можна примусово (рос. – принудительно) змінювати значення УПЦ;

❸ у цикл можна входити тільки через його заголовок. Тобто, не можна передавати управління з поза його меж у його середину (рисунок 8.34).

❹ змінна циклу може бути тільки перелічувального типу.

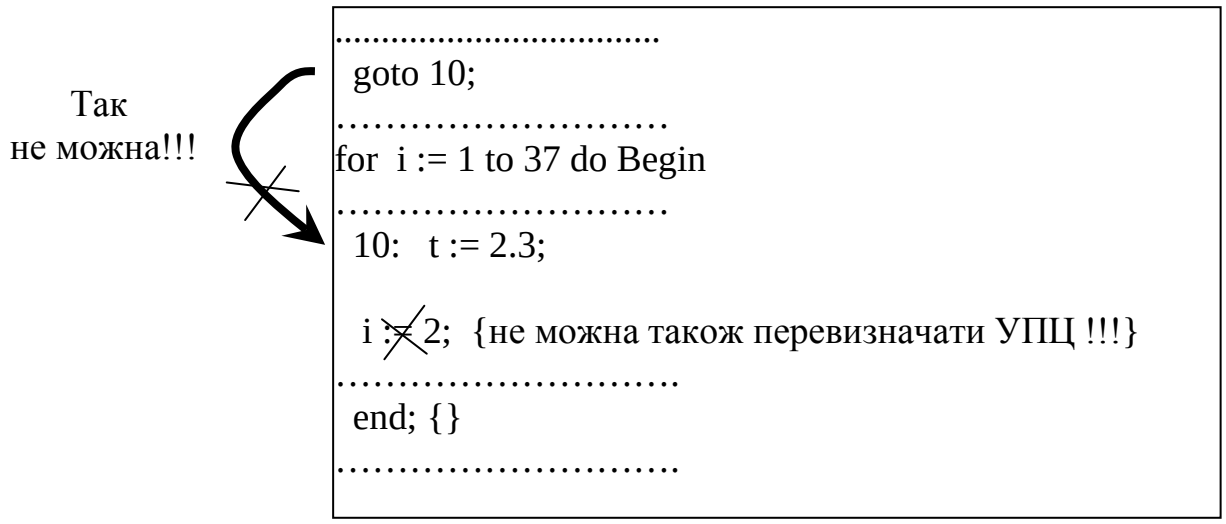


Рис. 8.34 Неможливість передачі управління в середину циклу з параметром

Наведемо приклад обчислення за допомогою циклу з параметром значення дійсного числа зведеного у цілу ступінь $y = a^n$. Для цього скористуємося формулою $a^n = \underbrace{a \cdot a \cdot a \cdot \ldots a}_{\text{(помножити } n \text{ разів)}}$. Додатково обчислимо суму $\sum_{i=1}^{n} a^i = \underbrace{a^i + a^i + \ldots + a^i}_{\text{(додавати } n \text{ разів)}}$.

```
{Обчислення ступеня дійсного числа a}
                                { і суми цих ступенів}
Program Stupen;
  Var
    A, Y, S : real;
    I, N : integer;
Begin
  Writeln('Введемо основу a та ступінь n');
  Readln(A, N);
  Y:=1;
  S:=0;
  for I:=1 to N do
    begin
       Y:=Y*A;
       S:=S+Y;
    end;
  Writeln(N, 'Ступінь числа', A);
  Writeln('Дорівнює= ',Y, 'а сума ступенів',S);
End.
```

Як правило, цикли з параметром використовуються з структурами даних типу масивів. Це пов'язано в першу чергу з тим, що цей тип є статичним і з початку програми повинен бути описаним на визначену заздалегідь розмірність. А параметр циклу при своєї зміні проходячи від початкового значення індексу масиву до кінцевого його значення, яке співпадає з пороговою величиною цього параметру, дозволяє легко опрацьовувати його (масиву) елементи.

Вправи

1. Що таке цикл?

2. Для чого потрібен параметр циклу?

3. Для чого служить порогове значення параметру циклу?

4. Поясніть логіку виконання циклів `for ... to … do` і `for ... downto ..do`.

5. Які змінні можуть використовуватися як параметр циклу For? Які змінні можуть використовуватися як границі діапазону для параметра циклу For?

6. Які фрагменти дій повинен містити алгоритм циклічної структури у найбільш загальному вигляді?

7. Побудувати таблицю значень абсцис і ординат для функції $f(x) = x - \sin(x)$ на відрізку [0,π/2] з числом розбивок відрізка m=10 та вивести дані на екран.

8. Написати програму для виводу на екран таблиці значень функції

y=tg x – sin 2x на проміжку [-π/4; π/4] із кроком π/16.

9. Написати програму для виводу на екран таблиці значень функції $y= x^2 + 7x - 14$ на проміжку [-3; 8] із кроком 0,55.

10. Написати програму для виводу на екран таблиці значень функції $y = \frac{\sin x + \sin 2x}{\cos x + \cos 2x}$ на проміжку [0; π] із кроком π/12.

11. Обчислити Sn (суму N перших значень) виразів *S*, якщо

$$S = 10 - 3\left(1 + \frac{2}{3}\right) + 7\left(2 - \frac{3}{9}\right)^2 - 11\left(3 + \frac{4}{27}\right)^3 + ...$$

12. Обчислити значення кінцевої суми S для виразу:.

$$S = \sum_{n=1}^{5} n^2 + \sum_{n=1}^{12} n^3 .$$

13. Обчислити значення кінцевої суми S для кінцевого ряду $\frac{1}{3} + \frac{1}{8} + \cdots + \frac{1}{n^2 - 1}$. Значення n увести з екрану.

14. Обчислити значення кінцевого добутку для виразу суми S $\left(1+\frac{1}{1(1+2)}\right)\cdot\left(1+\frac{1}{2(2+2)}\right)\cdots\left(1+\frac{1}{n(n+2)}\right)$. Значення n увести з екрану.

8.17. Оператор циклу з передумовою `while`. Оператор циклу з післяумовою `repeat`

Вище були розглянуті алгоритми з конструкцією циклічної структури типу:

```
for <УПЦ:=1> to <порогове значення N> do <оператор>,
```

яка працює з заздалегідь відомою кількістю повторень циклу `N`. Але, існують такі випадки, коли кількість повторень циклу взагалі невідома, але завдані деякі умови його завершення або продовження. Для програмної реалізації таких обчислювальних процесів у мові ТП існує два оператори:

❶ оператор циклу з передумовою (`while ... do`);

❷ оператор циклу з післяумовою (`repeat ... until`).

Оператор циклу з передумовою має таку загальну форму запису:

```
while <УМОВА> do <ОПЕРАТОР>;
```

Тут:

❶ слова `while` (поки) та `do` (робити) є службовими;

❷ УМОВА – логічний вираз будь якого ступеню складності;

❸ ОПЕРАТОР – будь який оператор мови ТП і у тому числі складений оператор. Зверніть увагу на те, що у цьому операторі може використовуватися тільки ***один(!)*** оператор (привласнення, умовний, вибору, і т. ін.), або сукупність операторів узята у операторні дужки `begin … end`.

Важливо мати на увазі, що цикл з передумовою виконується тоді і тільки тоді, коли логічний вираз <УМОВА> дорівнює логічному значенню TRUE. Як тільки він приймає значення FALSE – цикл завершується і управління передається оператору, що слідує за цим циклом.

Якщо <УМОВА> має значення FALSE з самого початку, цикл не виконається ані одного разу. Тобто, не виконається ні одного оператору з тіла циклу.

Припустимо, що Вам потрібно обчислити за допомогою оператора циклу з передумовою функцію X^2 для трьох різних випадків, що вказані у таблиці 8.39. Складність конструювання циклу витікає з тієї причини, що потрібно враховувати три логічних умови:

❶ змінна X повинна належати відрізку [-200, 200];

❷ значення ступеню функції X^2 повинно не перевищувати 10^3;

❸ у двох кінцевих варіантах завдання необхідно врахувати обчислення парних і непарних значень ступеню функції X^2.

Складність вирішення цих задач полягає у тому, що Ви повинні урахувати початкове значення аргументу X, його участь у трьох вищевказаних логічних виразах та участь логічного виразу у операторі циклу `while ... do`. Нагадуємо, що для об'єднання декількох простих логічних виразів у один складний потрібно використовувати логічні операції (and, or, not).

Таблиця 8.39

Варіанти вирішення комплексних завдань

Умови трьох завдань		
1	2	3
Обчислити значення X^2 для змінної X (–200≤X≥200), щоб максимальне значення степені X не перевищувало 10^3	Обчислити значення X^2 для парних (рос.– чётных) значень змінної X на відрізку (–200≤X≥200), щоб максимальне значення степені X не перевищувало 10^3	Обчислити значення X^2 для непарних (рос.– нечётных) значень змінної X (–200≤X≥200), щоб максимальне значення степені X не перевищувало 10^3
Варіанти вирішення завдань 1,2 та 3 з циклом `while ... do`		
1	2	3

const tens = 1e3;
var
x , st : Longint;
begin
writeln('Vvedite 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('Vvedite 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('Vvedite 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.

Зверніть увагу на те, що якщо будуть введені значення X < –200, або X > 200 цикл в усіх трьох програмах не буде виконано ні жодного разу.

Оператор циклу з післяумовою має наступний вигляд.

```
repeat
  <ОПЕРАТОР-1>;
  <ОПЕРАТОР-2>;
  ……..................
  <ОПЕРАТОР-N>
until <УМОВА>;
```

Де:

❶ `repeat` (повторяти), `until` (до тих пір, поки...) – службові слова;

❷ ОПЕРАТОР–i (i=1,2, ..., N) – будь який оператор мови ТП;

❸ УМОВА – логічний вираз, що побудований за правилами мови ТП.

Дія оператора `repeat ... until` подібна до роботи оператора `while ... do`, але є деякі важливі різниці.

1. Перевірка значення логічного виразу <УМОВА> реалізується після одноразового виконання тіла циклу. Тобто оператори ОПЕРАТОР<1 ... N> будуть виконані хоча б раз за усякі умови.

2. Оператори `repeat ... until` подібні до операторних скобок `begin ... end`, тому поміж ними можна розміщувати групи операторів, відділяючи їх поміж собою точкою з комою.

3. Цикл `repeat ... until` виконується, на відміну від циклу `while ... do`, до тих пір, поки значення логічного виразу <УМОВА> дорівнює FALSE і завершує свою роботу коли він приймає значення TRUE.

Наведемо приклад використання цього циклу (див. табл.8.40) для вирішення попередньої задачі таблиці 16.1 обчислення функції X^2.

Таблиця 8.40

Варіанти вирішення комплексних завдань

Завдання для вирішення		
1	2	3
Обчислити значення X^2 для змінної X (–200≤X≥200), щоб максимальне значення степені X не перевищувало 10^3	Обчислити значення X^2 для парних (рос.–чётных) значень змінної X (–200≤X≥200), щоб максимальне значення степені X не перевищувало 10^3	Обчислити значення X^2 для непарних (рос.–не-чётных) значень змінної X (–200≤X≥200), щоб максимальне значення степені X не перевищувало 10^3
Варіанти вирішення завдань 1,2 і 3 з циклом `repeat ... until`		
1	2	3

const tens = 1e3;
var
x , st : Longint;
begin
writeln('Vvedite 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('Vvedite 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('Vvedite 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.

Цікаво простежити як працюють усі три оператори циклів (“з параметром”, “з передумовою” та “з післяумовою”) при обчисленні одного й того ж виразу.

Припустимо, необхідно обчислити наступну суму $\sum_{n=1}^{10} a^{\frac{1}{n}}$. Вирішення задачі буде складатися з наступних дій.

❶ Попередньо потрібно увести значення постійної величини *a*.

❷ Змоделювати зміну значення виразу $\frac{1}{n}$ на кожному новому кроці.

❸ Змоделювати обчислення функції однієї змінної $a^{\frac{1}{n}}$. Це, до речі, можна зробити за допомогою тотожності (рос. – тождества) $a^k = e^{k \cdot \ln a}$, яке у мові ТП реалізується за допомогою вбудованих функцій `exp`(x) та `ln`(x) у вигляді $a^{k} = \exp(k \cdot \ln(a))$. У виразі $e^{k \cdot \ln a}$ значення k буде моделювати вираз $\frac{1}{n}$. Значення елемента суми для кожного значення змінної *n* будемо привласнювати змінній `elm`.

❹ Далі потрібно організувати циклічний процес обчислювання вищенаведеної суми з вищеописаних компонентів.

Приведемо алгоритм і його реалізацію на мові ТП, що збирає суми $\sum_{n=1}^{10} a^{\frac{1}{n}}$ у трьох різних циклах у змінних з іменами SFOR, SWHILE, SREPEAT.

```
var
  n : integer;
  a, elm, SFOR, SWHILE, SREPEAT, k : real;
Begin
  Writeln('Введемо 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; end;
{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('3 Суми =',SFOR:10:3,SWHILE:10:3,SREPEAT:10:3);
end.
```

Сформулюємо вищенаведену задачу іншим чином. Припустимо, потрібно обчислити суму $\sum_{n=1}^{\infty} a^{\frac{1}{n}}$ з точністю $\varepsilon = 0.01$.

Тоді цикл з параметром (`for … to … do`) для цього випадку використовувати вже неможливо, оскільки заздалегідь невідомо скільки разів потрібно додавати елементи суми до загальної суми, щоб задовольнити поставлені умови. Не зайве нагадати, що вирішення цієї задачі неможливо напряму і у додатку MS Excel. Адже і у такому потужному програмному продукті потрібно буде написати спеціальну процедуру на мові Visual Basic for Application з використанням подібних циклічних операторів цієї мови.

Отже, на мові ТП рішення нової задачі можна записати з використанням одного з двох операторів циклу з умовами:

`(while … do)` або `(repeat … until)`.

Для демонстрації їх можливостей використаємо їх обидва. Точність епсілон (ε) обчислення значення суми у програмі моделюємо константою з іменем `eps`, початкове значення якої завдаємо на рівні 0.01. Зверніть увагу, що головною вимогою завершення циклів повинно бути виконання умови, щоб значення чергового елементу суми `elm` було менше значення ε=0.01:

`elm < eps,`

Але, Ви пам'ятаєте, що умовою роботи циклу `while ... do` є значення TRUE умовного виразу його виконання. Зрозуміло, що значення першого елемента суми не менше `eps`. Тому ми можемо записати цю умову з використанням логічної операції інвертування (NOT) логічного значення.

```
Program Next_Cycles;
  const eps = 0.01;
var
  n : integer;
  k, a, SWHILE, SREPEAT : real;
Begin
  Writeln('vvedite 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; end;
{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('2 Summy=',SWHILE:10:3,SREPEAT:10:3);
end.
```

Так як оператор циклу `repeat ... until` виконує роль операторних дужок `begin ... end`, після останнього оператора тіла циклу перед службовим словом `until` ***розподільний знак*** **“;”** ставити не треба.

Оператор `while` використовується частіше за оператор `repeat`. Це пов’язано з тим, що у багатьох практичних випадках потрібно виконувати перевірку на закінчення циклу до його виконання і мати можливість при необхідності взагалі пропустити цей цикл.

Вправи

1. Чим відрізняються цикли з умовами від циклу з параметром?
2. Що собою уявляє умова у цих обох циклах?
3. Яке значення повинна мати умова у цикл `while`, щоб він виконувався?
4. Яке значення повинна мати умова у цикл `repeat`, щоб він виконувався?
5. Чим взагалі відрізняється цикл `while` від циклу `repeat`?
6. З використанням оператора циклу з передумовою `while` напишіть і виконайте програму табулювання функцій *y* з дійсним аргументом *x*:

$$a)\ y = \frac{\sin 3\pi - x}{x + \sqrt{\left|\frac{3\pi}{2} + x\right|}};$$

$$b)\ y = \frac{x + ctg\ (x+1)}{\sin^2\ (x+1)};$$

$$c)\ y = 10^{\log_{10}\left|x + \frac{a}{4}\right|};$$

$$d)\ y = \frac{\left|\sin^2 x^2 + \cos^2 x^2\right|}{1 + \dfrac{x^2 \cdot \sin^2 b \cdot x + 1}{x}}.$$

Границі зміни значень аргументу *x* узяти на відрізку [–1, 1].

7. Виконайте попередні вправи з використанням циклу `repeat`.
8. Знайдіть суму за формулою: $S = \sum_{n=1}^{\infty} \frac{1}{n^2}$.

8.18. Засоби дослідження виконання дій програми за допомогою дебаггера[24]

Якщо алгоритм програми занадто складний і Ви отримуєте незрозумілий результат або деякі операції у програмі зупиняють роботу комп'ютера (наприклад: ділення на нуль, переповнення розрядної сітки ПК і т.д.), то Ви повинні провести дослідження програми засобами ІСР ТП. Для цього у Турбо Паскалі є дуже розвинуті засоби перевірки стану змінних, результатів операцій з цими змінними і так далі за допомогою вбудованого дебаггера. ***Але пам'ятайте: дебаггер допоможе Вам тільки тоді, якщо у кожному рядку програми буде знаходитись тільки один оператор ТП !***

Для побудови простої дослідницької програми виконаємо у ІСР ТП команду **File/New**, яка відкриє чисте вікно редактора з назвою `NONAME00.PAS`. У цей чистий лист введемо текст наступної програми для її дослідження під назвою (іменем) `MyFirst`:

```
program MyFirst;
var
  A,B: Integer;
  Ratio: Real;
Begin
  repeat
    Write('Vvedite dva chisla A, B : ');
    Readln(A,B);
    Ratio := A/B;
    Writeln('Rezultat Ratio = ',Ratio:8:2);
    Write('Najmite <Enter>...');
    Readln;
  until B = 0;
End.
```

Для збереження тексту Вашої програми, тобто запису її на диск з відповідним ім'ям, увійдіть у меню **File**, виберіть **Save As** і наберіть у стрічці з назвою **Save file as** відповідного меню ім'я Вашого файлу `MyFirst`, а потім натисніть **Enter**. Розширення імені файлу `.PAS` додається при запису тексту програми на диск автоматично (рис. 8.35).

Компілюйте і запускайте свою програму, використовуючи **Ctrl-F9**. Турбо Паскаль автоматично відкомпілює Вашу дослідницьку програму перед запуском і виконає її.

Оскільки Ваші оператори були укладені в цикл <`repeat` … `until`>, це призведе до того, що усі оператори поміж `repeat` і `until` будуть виконуватися доти, поки умовний вираз, що впливає на кількість виконання циклів, не стане мати значення `True` (істина). Цей умовний вираз перевіряє,

[24] Дебаггер – програма, що допомагає знаходити та локалізувати помилки програмування.

дорівнює значення B нулю чи ні. Якщо змінна B має значення 0, цикл повинен завершитися.

Рис. 8.35. Вигляд програми MyFirst у синьому вікні редактора ТП

Взагалі, ця програма повинна працювати без зупинки, бо умовою виходу з циклу є фактично помилкове значення змінної *B* = 0, що зупиняє програму та викликає діагностику:

```
Error 200: Division by zero
(ошибка 200: деление на 0)
```

Цей код спроектований так спеціально, щоб показати Вам, як використовувати засіб під назвою дебаггер (тобто пошуковець помилок – "багів"), убудований у інтегроване середовище Турбо Паскаль і який дозволяє Вам пересуватися по своєму коду по рядках, вивчаючи роботу програми. У той же час, Ви можете переглядати значення усіх потрібних Вам змінних і досліджувати як вони змінюються у ході виконання програми.

Щоб почати сеанс дослідження, виберіть команду **Run/Trace Into** (чи натисніть **F7**). Якщо Ваша програма має потребу в перекомпіляції, Турбо Паскаль зробить це. Перший оператор у розділі операторів Вашої програми (у даному випадку – begin) буде висвітлено спеціальною смугою; з цього моменту ми будемо називати цю висвітлену смугу – смугою запуску (рис. 8.36). Наступні натиснення клавіші **F7**, пересувають курсор по стрічках Вашої програми, послідовно виконуючи їх.

Другим етапом дослідження, є Ваша вказівка дебаггеру про імена змінних, зміни значення яких Ви бажаєте побачити у цьому сеансі роботи програми, що досліджується. Для подання визначених програмістом вказівок у ІСР ТП використовується спеціальне діалогове вікно **Add Watch** (тобто додавання контрольних точок для стеження (рос. – наблюдения)).

Рис. 8.36. Початок роботи програми у режимі сеансу дослідження

Для завдання потрібних точок виконайте команду **Debug/Add Watch** (рис. 8.37) і коли з'явиться діалогове вікно з назвою **Add Watch**, наберіть ім'я першої потрібної для дослідження у програмі змінної, тобто **A**, і натисніть **Enter** (рис. 8.38). Нагадуємо, що вибір команд можна виконувати або за допомогою курсору миші, або натисненням управляючих клавіш: у даному випадку **Alt+D**, а потім **A**.

Рис. 8.37. Середовище ТП перед виконанням команди **Debug/Add Watch**

Виконання вищевказаних дій призведе до появи додаткового вікна з ім'ям **Watches** у нижній частині головного екрану, у якому з'явиться ім'я цієї змінної

та її поточне значення (рис. 8.39). ***Прийміть до уваги, що це додаткове вікно закривається командою*** **Window / Close** (або – Alt+F3)***, а смуга запуску вилучається командою*** **Run / Program reset** (або – Ctrl+F2)**.**

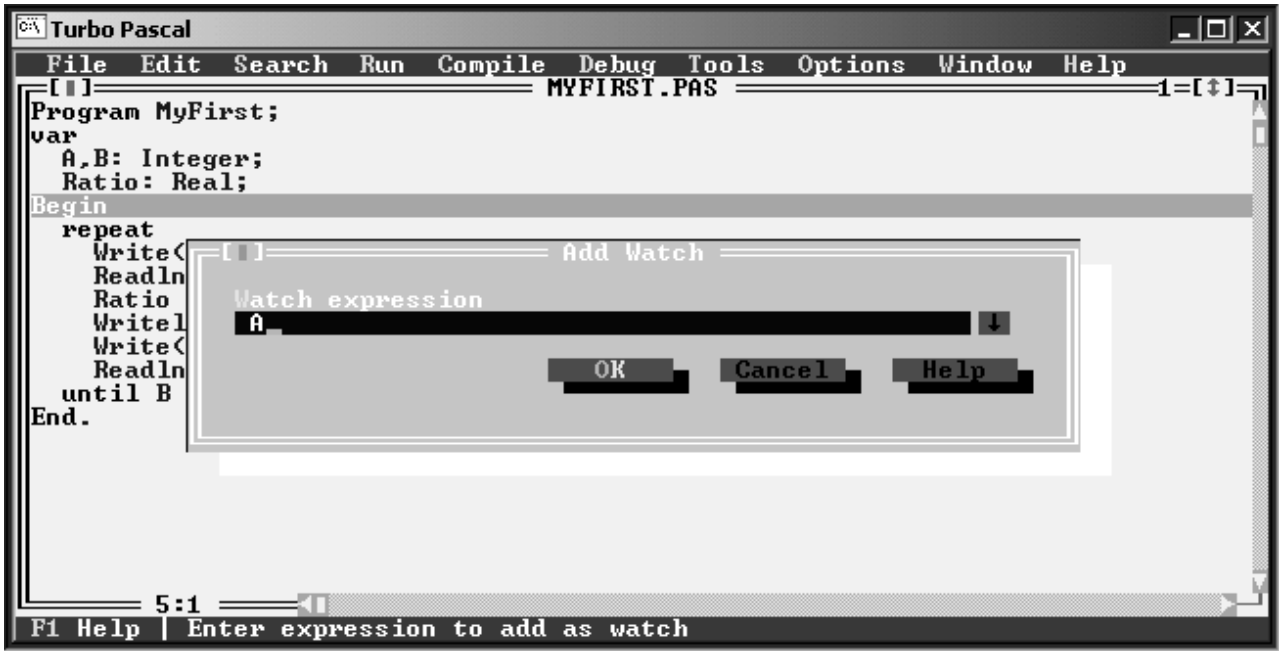


Рис. 8.38. Увід імені змінної A, значення якої досліджується

Важливо також пам’ятати, що для додавання наступних імен змінних (B та `Ratio`) у вікно **Watches** достатньо перед уводом їхніх імен попередньо натискувати клавішу **Insert**, що розташована справа над клавішею **Delete**.

Рис. 8.39. Висвітлення значень змінних у вікні
Watches (Спостереження)

Після уводу чергового імені у стрічку меню **Add Watch** і натиснення **Enter**, у вікні **Watches** знизу екрану додаються ці імена з їхніми поточними значеннями (рис. 8.39).

Тепер, коли усі змінні нами висвітлені, після чергового натиснення **F7** програму буде переведено до чорного екрану DOS (результатів), тому що оператор `Readln(A,B)` очікує введення двох Ваших чисел. Наберіть два цілих числа, розділені пробілом; переконайтеся, що друге число – не нуль (рис. 8.40).

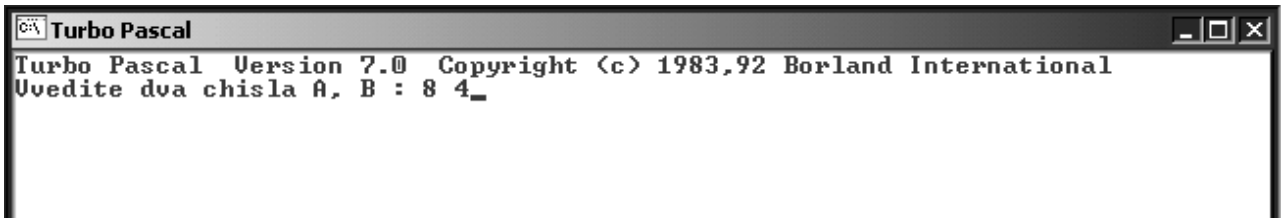


Рис. 8.40. Екран DOS при введенні даних `A` і `B`

Тепер натисніть **Enter** і Ви повернетеся назад у вікно редактора зі смугою запуску на операторі присвоювання в рядку 9 (`Ratio := A/B;`) програми `MyFirst` (див. рис. 8.36).

Натисніть **F7** і програма виконає цей оператор присвоювання. Тепер смуга запуску знаходиться на операторі `Writeln` у рядку 10. Але водночас значення змінної `Ratio` отримає нове значення 8/4 = 2, ініційоване уведеними даними, з'єднаними у виразі оператором ділення (/) (рис. 8.41). Натисніть **F7** двічі. Тепер ви повинні виконати `Readln` у рядку 12. Ще раз натисніть **F7**, подивитеся висновок своєї програми і потім натисніть **Enter**. Смуга запуску тепер знаходиться на операторі `until`. Натисніть **F7** кілька разів і програма повернеться до початку циклу `repeat`, тобто оператора `Write`, і т.д.

Рисунок 8.41. Значення змінної `Ratio` при введених даних `A` і `B`

Таким чином, могутній інструмент – дебаггер – дозволяє робити кроки по коду програми "один рядок за один раз" і водночас спостерігати за поведінкою Ваших змінних (таблиця 8.41).

Таблиця 8.41

Команди дебаггера та його дії

Назва дії	Команда ТП	Гарячі клавіші
Перехід до режиму дебаггера	**Run/Trace Into**	**F7**
Пересування по операторам програми	**–**	**F7**
Виклик вікна **Watches** для перегляду значень змінних та виразів у покроковому режимі	**Debug/Add Watch**	**Alt+D**, а потім **A**
Додавання у вікно **Watches** імен змінних та виразів	**Debug/Add Watch**	**Insert (клавіша)**
Закриття вікна **Watches**	**Window/Close**	**Alt+F3**
Вилучення смуги запуску	**Run/Program reset**	**Ctrl+F2**

Крім значень окремих змінних, у вікні **Watch** можна також спостерігати значення виразів. Щоб увести потрібний для дослідження вираз натисніть **Alt-D** для появи меню **Debug** (це альтернативна можливість виклику). Виберіть команду **Add Watch** з меню Watches (чи натисніть **Ctrl-F7**). Тепер можна набрать у вікні введення **Watch Expression** не тільки ім'я змінних `A`, `B` и `Ratio`, які з'являться у вікні **Watch** разом зі своїми поточними значеннями, а й також і вираз `A/B`.

Далі, виберіть **Run/Trace Into** (чи натисніть **F7**) для того, щоб зробити крок у своїй програмі. Цього разу, коли Ви повинні ввести два числа, уведіть 0 для другого числа. Коли Ви натиснете **Enter** і повернетеся в IDE Турбо Паскаль, подивитеся на вираз `A/B` у вікні **Watch** (для цього натисніть Alt і # (номер) вікна чи **Alt-W W**). Замість цього значення буде стояти фраза `"Invalid floating-point operation"` (неправильна операція над числами з крапкою, що плаває); це відбулося тому, що розподіл на нуль невизначено. Хоча помітимо, що наявність цього вираження у вікні **Watch** не приводить до зупинки програми з помилкою. Замість цього видається повідомлення про помилку, а відлагоджувач не виконує розподіл у вікні **Watch**.

Тепер натисніть **F7** знову, привласнюючи `A/B` перемінної `Ratio`. У цьому місці відбудеться аварійне завершення програми, і угорі вікна редактора знову з'явиться повідомлення про помилку `"Division by zero"`.

Тепер можливо Ви задумалися про те, що ж неправильно працює у вашій програмі: якщо Ви вводите значення 0 для другого числа (`B`), програма завершується з помилкою часу виконання.

Як зафіксувати її? Якщо `B` має значення `0`, не поділяєте `A` на `B`. Відредагуйте Вашу програму так, щоб вона виглядала в такий спосіб:

```
program MySecond;
  var
     A,B: Integer;  Ratio: Real;
Begin
  repeat
    Write('Enter two numbers: ');
    Readln(A,B);
    if B = 0 then
      Writeln('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.
```

Тепер запустіть свою програму (чи самі, чи використовуючи дебаггер). Якщо Ви використовуєте дебаггер, зауважуйте, як змінюються значення у вікні **Watch** у міру здійснення кроків у програмі. Коли Ви готові зупинитися, уведіть 0 у змінну `B`. Програма зупиниться після виведення повідомлення `"The ratio is undefined. Press <Enter>..."` ("Відношення не визначене. Натисніть <**Enter**>...").

Тепер Ви зрозуміли, яким могутнім засобом є дебаггер для дослідження роботи Вашої програми. Можна пересуватися в програмі рядок-за-рядком; можна роздивлятися у вікні **Watch** значення змінних і виразів Вашої програми й переглядати зміни значень по ходу виконання програми.

Вправи

1. Що таке дебаггер і для чого він призначений?

2. Які команди і гарячі клавіші послуговують для виклику дебаггера і переходу до режиму дебагування програми користувача?

3. Які команди і гарячі клавіші послуговують для пересування по операторам програми у режимі її дебагування?

4. Які команди і гарячі клавіші послуговують для виклику вікна **Watches** для перегляду значень змінних та виразів у покроковому режимі?

5. Які команди і гарячі клавіші послуговують для додавання у вікно **Watches** імен змінних та виразів?

6. Які команди і гарячі клавіші послуговують для закриття вікна **Watches**?

7. Які команди і гарячі клавіші послуговують для вилучення смуги запуску?

8. Обчислити за допомогою циклу з параметром `for` значення кінцевої суми S для виразу:

$$S = \sum_{n=1}^{5} n^2 + \sum_{n=1}^{12} n^3 .$$

Проведіть дослідження роботи програми за допомогою дебаггера.

9. З використанням оператора циклу з передумовою `while` напишіть і виконайте програму табулювання функції *y* з дійсним аргументом на відрізку [–2, 0]. Проведіть дослідження роботи програми за допомогою дебаггера.

$$y = \frac{\sin 3\pi - x}{x^{\frac{1}{3}} + \sqrt{\left|\frac{7\pi}{2} + x^2\right|}} .$$

10. З використанням оператора циклу з післяумовою `while` напишіть і виконайте програму табулювання функції *y* з дійсним аргументом на відрізку [–1, 1]. Проведіть дослідження роботи програми за допомогою дебаггера.

$$y = \frac{\sqrt{\left|x^3\right|}}{4 - x} .$$

8.19. Моделювання у циклічних обчисленнях деяких типових виразів

У циклічних процесах дуже часто використовуються деякі вирази, котрі мають стандартні алгоритмічні реалізації. До них можна віднести наступні:

$$(-1)^n \qquad (8.8)$$

$$n\,! \qquad (8.9)$$

$$(2\,n)\,! = 2n! \qquad (8.10)$$

Всі вони залежать від елементів натурального ряду чисел:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, ... , *n*

Як правило, скорочено зміну значень натурального ряду на відрізку [*1*,*n*] записують так:

1, 2, ... , *n*

У залежності від зміни значень елементів ряду *n*, значення вищевказаних виразів приймають відповідні значення (табл. 8.42). При цьому і ряд можна розглядати як вираз, значення якого змінюється у відповідності зі зміною номера елемента *n*.

Таблиця 8.42

Зміни значень виразів у відповідності зі зміною значень, що у них підставляються

Вираз	Зміна значень виразу				
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	…
$(2\,n)\,! = 2n!$	2!=2	4!=24	6!= 1·2·3·4·5·6= =720	8!=1·2·3·4·5·6·7·8= =40320	…

Значення виразу $(-1)^n$ можна моделювати двома різними способами:

❶ на кожному кроці приформовувати унарний знак "–" до передуючого значення змінної, що зберігає значення(–1) або (+1);

❷ на кожному кроці множити передуюче значення (–1) або (+1) на (–1).

Але треба пам'ятати, що для виконання приформування знаку чи множення на (–1) треба попередньо виймати число зі змінної на суматор, провадити потрібні дії, а потім засилати їх назад до змінної.

Перші п’ять рядків значень цих функцій, що будуть виведені на екран програмою, текст якої наведений на наступній сторінці, виглядають так:

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
```

```
{Програма моделювання значень функцій 8.8, 8.9, 8.10}
Var
     med1, med2, Factor1, Factor2 : Real;
         n, n2, I : Word;
Begin  {Ініціалізуємо початкові значення змінних}
  med1 := –1;  { Ім'я змінної, що приймає значення (–1)}
  med2 := –1; { Ім'я змінної, що приймає значення (–1)}
  Factor1n := 1; { Ім'я змінної, що приймає значення n!}
  Factor2n := 2; { Ім'я змінної, що приймає значення 2n!}

  n := 1; { Ім'я змінної, що приймає значення n}
  n2:= 2; { Ім'я змінної, що приймає значення 2n}
{    Фактично привласнюємо змінним значення виразів при n = 1 , яке }
{моделює значення змінної n. Змінну 2n моделює значення змінної n2. }
{Обмежуємо дію циклу з передумовою значенням n=40,}
{що еквівалентно максимальному значенню n1=40}.
{}
  Writeln ('Значення змінних=', 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   ('Значення   змінних=',   med1:8:1,
                med2:8:1,  n:3,  Factor1n:8:1,  n2:5,
                Factor2n:11:1);
    end;
End.
```

Дуже важливим моментом є процес відображення циклічного (тобто послідовного) додавання деяких даних до значення визначеної змінної. Припустимо, маємо обчислити вираз Y = A+5, при значенні A=2.

На мові ТП це виглядає так:

```
A := 2;
Y := A + 5; {Стає Y = 7}
```

Але, якщо треба додати до значення, що знаходиться у змінній Y ще якесь значення (припустимо 8), це буде виглядати так:

```
Y := Y + 8;  { Стає Y = 15}
Y := Y + 10; { Стає Y = 25}
```

…… {і так далі}.

Вираз `X := X + 10` для комп'ютера вказує, що потрібно:

❶ узяти попереднє значення, що знаходиться у змінній `X`;

❷ скласти його з постійною 10;

❸ заслати нове значення на старе місце у змінну `X`.

Інший приклад. Припустимо потрібно обчислити вираз:

$$f = \cos x + \cos y + \cos z.$$

Якщо це записувати одним виразом, то можна на мові ТП написати так:

```
f := cos (x) + cos (y) + cos (z).
```

Тоді вбудованою функцією `cos(x)` будуть обчислені три значення косинусів від відповідних аргументів *x, y, z,* а потім, після підсумовування цих трьох значень, сума буде заслана у змінну *f* і замінить там попереднє значення. Але, якщо цей же вираз обчислювати у циклі, то потрібно попередньо очистити змінну *f*, а потім провадити додавання на кожному кроці. Очищення змісту змінної *f* виконується засиланням до неї нульового значення. Відповідний код може виглядати так:

```
……
f := 0;
f := f + cos (x);
f := f + cos (y);
f := f + cos (z);
…… ….
```

У результаті виконання цієї послідовності команд отримаємо результат, що повинен дорівнювати обчисленому у вищенаведеному виразі.

Вправи

1. Обчислити за допомогою циклу з параметром `for` значення кінцевої суми S для виразу:

$$S = \sum_{n=1}^{5} (-1)^n n^2 + \sum_{n=1}^{12} (-1)^{3n} n^3 .$$

Проведіть дослідження роботи програми за допомогою дебаггера.

2. З використанням оператора циклу з передумовою `while` напишіть і виконайте програму табулювання функції *y* з дійсним аргументом x = 0.25 на відрізку зміни *n* [1, 5]. Проведіть дослідження роботи програми за допомогою дебаггера.

$$y = \frac{\sin 3\pi - x}{n!}.$$

3. З використанням оператора циклу з післяумовою `while` напишіть і виконайте програму табулювання функції *y* з дійсним аргументом на відрізку зміни *n* [1, 7] при *x*= –9. Проведіть дослідження роботи програми за допомогою дебаггера.

$$y = \frac{\sqrt{|x^3|}}{2n!}.$$

8.20. Особливості обчислення нескінченних (рос. – бесконечных) сум. Організація ітераційних процесів за допомогою циклів `while` та `repeat`

Одним з найбільш розповсюджених алгоритмічних процесів є математична дія додавання. У простих випадках сума виглядає звичайно так:

$$C = A + B.$$

Однак у більш складних випадках сума виглядає значно інакше, коли кількість елементів , що додаються, заздалегідь невідома, а результат прямує до границі (рос. – стремится к пределу) за деякою формулою. Типовим прикладом ітераційного циклічного процесу може бути задача обчислення суми нескінченного ряду. Поняття суми пов'язане з поняттям збіжності (рос. – сходимости) такого нескінченного ряду. Цей ряд значень $t_0, t_1, \ldots, t_n, \ldots$ зветься збіжним (рос. – сходящимся), якщо сума $s_n = t_0 + t_1 + \ldots t_n$ його перших (n+1) елементів, що складаються при нескінченному зростанні n прямує до деякої границі S, яка і зветься сумою ряду, тобто

$$\lim_{n\to\infty} \sum_{i=0}^{n} t_i = S$$

Загальний член t_n ряду, що збігається, при цьому прямує до нуля, тобто:

$$\lim_{n\to\infty} t_n = 0; \quad \lim_{n\to\infty} (s_n - s_{n-1}) = 0.$$

Таким чином, послідовність $s_1, s_2, \ldots, s_n, \ldots$ є послідовністю значень, яку ми шукаємо і визначає наступні умови закінчення додавання:

$$\left| s_n - s_{n-1} \right| \le \varepsilon \quad або \quad \left| t_n \right| \le \varepsilon.$$

Взагалі, сумою декількох елементів зветься додавання чергових значень до суми усіх попередніх елементів з цієї послідовності.

У математичному виді це записується за допомогою знаку додавання ∑:

$$\sum_{n=1}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1\cdot 2} + \frac{1}{1\cdot 2\cdot 3} + \ldots + \frac{1}{1\cdot 2\cdot 3\cdot \ldots \cdot n} \qquad (8.11)$$

$$\sum_{n=1}^{\infty} \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} \qquad (8.12)$$

Незважаючи на, начебто, просте відображення цього процесу, є деякі важливі обставини розглянути більш уважніше те, що діється при додаванні елементів нескінченних рядів.

Так, як операції додавання у математичному розумінні потрібно виконувати до нескінченності (∞), потрібно розуміти, що у комп'ютері такі

великі значення, звичайно ж, оброблятися не можуть. З іншого боку, як правило, потрібно обчислювати кінцеву суму з деякою точністю. Розглянемо, що мається на увазі у цьому випадку.

Коли виконуються обчислення на деякому *k*-му кроці вже мається накопичена сума (*k-1*) елементів і черговий *k*-й елемент додавання (рис. 8.42), що сумується до загальної купи.

$$\underbrace{(a_1 + a_2 + \ldots + a_{k-1})}_{\sum_{n=1}^{k-1}} + a_k$$

Рис. 8.42. Схема додавання у нескінченних сумах

З точки зору значення чергового елементу ряду, то він зробить тим менший внесок у загальну суму, чим меншим він буде сам. Якщо він стане менше якоїсь достатньо малої постійної величини (константи) (наприклад, наперед заданої величини $\varepsilon = 0.0000001$ чи деякої іншої), він повинен перестати вкладати відповідний внесок у загальну суму і впливати на ті цифри суми, які потрібні для точного її обчислення. Але, як свідчить досвід чисельних обчислень, є ще деякі фактори, котрі впливають на процес розробки алгоритму. Подивимось уважніше на результати обчислення сум, що відображаються формулами (8.11) та (8.12). Виявляється, що точність залежить ще і від того, як швидко збільшується знаменник дроби, що обчислюється. Так для формули (8.11) значення компонентів будуть змінюватися так (табл.8.43).

Таблиця 8.43

Зміна значень компонентів формули (8.11)

Значення *n*	Значення елемента $\frac{1}{n!}$	Значення суми *K* елементів $\sum_{n=1}^{K}\frac{1}{n!}$
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
Примітка: *K* – кількість підсумованих елементів		

Обчислимо формулу (8.12). Бачимо, що знаменник дроби, що має участь у виконанні потрібних дій росте дуже швидко і це можна бачити у наступній таблиці 8.44:

Таблиця 8.44

Значення елементів ряду (8.12)

Значення n	Значення $\frac{1}{n}$	Значення $\sum_{n=1}^{K}\frac{1}{n}$
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

Уявляється, що коли елемент, який додається, зменшується дуже повільно, передбачити його внесок у загальну суму досить важко. Тому Ви повинні мати на увазі декілька варіантів алгоритмів, що можуть дати змогу отримати потрібний результат у різних випадках чи використати дебаггер для контролю точності результату обчислень.

Подивимось уважніше на таблиці 8.43 та 8.44. Для успішного завершення сумування елементів першої суми, достатньо перевірити умову, щоб черговий елемент ряду був менший деякого малого числа ε (епсілон), яке завдає сам програміст (користувач). Величина ε і буде контролювати точність кінцевої суми. Іншою мовою, (згідно таблиці 8.45), якщо ми порівнюємо значення чергового елемента $\frac{1}{n!}$ з ε= 1e-6 (так записується у ТП число 0.000001), то отримуємо значення суми з точністю приблизно 6 знаків після крапки. При порівнянні значення тих же елементів з ε= 1e-7, точність збільшується, і так далі.

Таблиця 8.45

Залежність значення кінцевої суми від ε

Значення ε, що завдає користувач	Кінцеве значення параметра n	Значення останнього елемента $\frac{1}{n!}$ у суммі	Значення кінцевої суми $\sum_{n=1}^{K}\frac{1}{n!}$
.....			
1e-6	10	0.000000275	1.718281801
1e-7	11	0.000000025	1.718281826
1e-8	12	0.000000002	1.718281828
.....			

Реалізація алгоритму ітераційного обчислення ряду $\sum_{n=1}^{\infty} \frac{1}{n!}$ з таким підходом може виглядати так:

```
Uses Crt;
Const Eps = 1.e-7;  {Задаємо точність обчислення ε=0.0000001}
Var
  Sum, Elm, Fact  : Real; {Об'являємо змінні}
  n : Integer; { Об'являємо змінну для обчислення n}
Begin  ClrScr; {Очищаємо екран процедурою ClrScr}
  n := 1;      {Ініціалізуемо n}
  Sum := 0; {Обнуляємо змінну під суму}
  Elm := 1; {Ініціалізуємо Elm  для входу у цикл}
  Fact := 1; { Ініціалізуємо начальне значення факторіалу}
  while  Elm > Eps do
    begin
      Fact := Fact * n; {Моделюємо значення факторіалу n!}
      Elm := 1/ Fact; { Моделюємо значення елементу 1/n!}
      Sum := Sum + Elm; { Моделюємо суму ∑(n=1..∞) 1/n!}
      n := n + 1; { Моделюємо зміну значень індекса n}
    end;
  Writeln('Rezultat= ', Sum); {Виводимо на екран результат}
End.
```

Вправи

1. Обчислити наближене значення нескінченної суми S з точністю до ε=0.05. $S = \frac{1}{1\cdot 4}+\frac{1}{4\cdot 7}+\cdots+\frac{1}{(3n-2)(3n+1)}+\cdots$

2. Обчислити наближене значення нескінченної суми S з точністю до 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. Обчислити наближене значення нескінченної суми з точністю до ε=0.00001. $S = x-\frac{x^2}{2!}+\frac{x^4}{4!}-\cdots+(-1)^n\frac{x^{2n}}{(2n)!}\pm\cdots$

4. Обчислити наближене значення нескінченної суми S з точністю до ε=0.00000001. $S = \frac{x-1}{x}+\frac{(x-1)^2}{2x^2}+\cdots+\frac{(x-1)^n}{nx^n}+\cdots$

5. Обчислити суму з точністю до ε=0.00000001. $y=\sum_{n=1}^{\infty}\frac{(x+n)(x-n)}{2n!}$

6. Обчислити наступні суми з визначеною точністю:

$$\frac{1}{2}-\frac{\pi}{8}=\sum_{n=1}^{\infty}\frac{1}{(4n-1)(4n+1)}$$

$$\frac{1}{2}=\sum_{n=1}^{\infty}\frac{1}{(2n-1)(2n+1)}$$

$$\sin x=\sum_{n=1}^{\infty}(-1)^{n-1}\frac{x^{2n-1}}{(2n-1)!}$$

$$(1\pm x)^{-\frac{5}{2}}=1\mp\frac{5}{2}x+\frac{5\cdot 7}{2\cdot 4}x^2\mp\frac{5\cdot 7\cdot 9}{2\cdot 4\cdot 6}x^3+\frac{5\cdot 7\cdot 9\cdot 11}{2\cdot 4\cdot 6\cdot 8}x^4\mp\ldots$$

7. Обчислити наступні суми з визначеною точністю (при x=m, де m–номер в журналі).

$$(1-x)^{-m}=1+\sum_{n=1}^{\infty}\frac{m(m+1)\ldots(m+n-1)}{n!}x^n$$

$$(1+x)^{-m}=1+\sum_{n=1}^{\infty}(-1)^n\frac{m(m+1)\ldots(m+n-1)}{n!}x^n$$

8. Обчислити 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. Нескінченні множення та їх обчислення

Одним з достатньо розповсюджених, є також процес множення одного числа на інше.

$$C := A \cdot B;$$

Але коли кількість елементів, які приймають участь у процесі множення, починає зростати, обчислення такого результату стає складним.

Циклічним множенням зветься процес помноження деякої величини на те число, яке відображує результат множення усіх попередньо перемножених чисел.

У математичних виразах воно має позначення **П**, а його дію можна відобразити на графічній схемі таким чином (рис. 8.43).

$$\underbrace{(a_1 \cdot a_2 \cdot \ldots \cdot a_{k-1})}_{\prod_{n=1}^{k-1}} \cdot a_k$$

Рис. 8.43. Схема множення

Розглянемо одну з формул, що описує процес множення (8.13). Якщо почати підставляти у неї значення n = 1, 2, 3, 4, 5, … , ∞, то елементи цієї послідовності співмножників (рос. – сомножителей) будуть виглядати 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 результати обчислення цієї формули при зміні *n*, котре моделює *K*, наведено у таблиці 8.46.

Процес обчислення результату перемноження *n* елементів, що наведений у цієї таблиці, обмежений величиною ε= 1e-7. Тому він закінчився на значенні *K* = 3162. Треба особливо відмітити, що підхід щодо перевірки вимоги завершення вищенаведеного ітераційного процесу істотно відрізняється від того, котрий розглядався при обчисленні формул додаванні. Це пов'язано з тим, що значення чергового елементу напряму не впливає на кінцевий результат. Тому у алгоритмі був використаний підхід, що закінчує процес обчислення у випадку, ***коли різниця поміж попереднім і поточним значеннями результату множення стає менше ε = 1e-7*** (рис. 8.44). Тоді можна бути впевненим, що подальші дії не призведуть до зміни цифр числа у перших 7-ми позиціях (при ε = 1e-7) .

Таблиця 8.46

Обчислення результатів множення за формулою (8.13)

Значення n	Значення $\dfrac{1}{\left(\dfrac{1}{n}+1\right)}$	Значення формули $\prod_{n=1}^{K}\dfrac{1}{\left(\dfrac{1}{n}+1\right)}$
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

Треба відмітити, що елементи формули множення (стовпчик 2, табл.8.46), не зменшується, як у процесі додавання, а зростає.

$$\underbrace{(\mathbf{a_1 \cdot a_2 \cdot \ldots \cdot a_{k-1}})}_{\prod\limits_{n=1}^{k-1}} - \underbrace{(\mathbf{a_1 \cdot a_2 \cdot \ldots \cdot a_{k-1} \cdot a_k})}_{\prod\limits_{n=1}^{k}} < \varepsilon$$

Рис. 8.44. Схема порівняння елементів процесу множення для його своєчасного завершення

Алгоритм обчислення множення значень за формулою $\prod_{n=1}^{\infty}\frac{1}{\left(\frac{1}{n}+1\right)}$ можна записати так:

```
Uses Crt;
Const Eps = 1.e-7;  {Задаємо точність обчислення ε=0.0000001}
Var
  Mun, Mun_1, Mun_2, Elm, Chis, Znam : Real;{Об'являємо
                                                змінні}
  n : Integer; { Об'являємо змінну для обчислення n}
Begin      ClrScr; {Очищаємо екран процедурою ClrScr}
  n := 1;      {Ініціалізуемо n}
  Mun := 1; {Заносимо одиницю у змінну Mun для множення на
                                                неї}
  Mun_1 := 1; {Заносимо константу 1 для виконання вимоги
                                        циклу і входу у нього}
  Mun_2 := 0.5;{Значення першого елемента множення,
                                        обчислене користувачем}
  while  abs(Mun_1 - Mun_2) > Eps do
    begin
      Mun_1:=Mun_2; {Запам'ятовування передуючого значення функції,
       а потім попереднього значення добутку для порівняння з поточним}
      n := n + 1; { Моделюємо зміну значень індекса n}
      Chis := 1; { Моделюємо значення чисельника дроби}
      Znam := 1+1/n; { Моделюємо значення знаменника дроби}
      Elm := Chis/Znam; {Обчислюємо значення чергового
                                елементу за формулою 1/(1/n+1)}
      Mun := Mun * Elm; {Обчислюємо добуток }
      Mun_2 := Mun; {Запам'ятовуємо значення добутку}
    end;
  Writeln('Rezultat= ', Mun); {Виводимо на екран результат}
End.
```

Вправи

1. Обчислити наближене значення нескінченного множення з точністю до ε=0.0001 наступні вирази.

$$\prod_{n=2}^{\infty}\left(1-\frac{1}{n^2}\right) \qquad \prod_{n=1}^{\infty}\frac{4n^2}{4n^2-1}$$

8.22. Підпрограми: процедури і функції. Формальні і фактичні параметри. Передача параметрів “за значенням” та “за адресою”

Дуже часто у програмах користувача потрібно по декілька разів викликати деякі наперед визначені послідовності операторів для виконання обробки складних послідовностей даних. Це можна робити за допомогою трьох типів циклів, що описані у попередніх розділах цих вказівок.

Але надзвичайно широко розповсюджена інша форма повторювання послідовностей дій, що багаторазово виконуються. У алгоритмах такого роду у різних місцях програми зустрічається однакові фрагменти послідовностей операторів, які відрізняються тільки по значенням початкових (рос.–исходных) даних. При розробці програми за таким алгоритмом потрібно задавати одну і ту ж групу операторів, що відповідають кожному з фрагментів, які повторюються. Для підвищення ефективності програмування у мові ТП уведено поняття ***підпрограми***.

Група операторів, що буде повторно використовуватися, оформлюється у вигляді самостійної програмної одиниці – підпрограми. Вона записується одноразово у розділі описів головної програми, а у відповідних місцях просто ***викликається***. При виклику вказується ***ім'я підпрограми*** і супутні цьому виклику, так звані, ***фактичні параметри***.

Таким чином, ***підпрограмою*** зветься логічно завершена послідовність (група) операторів, що виконує визначену послідовність дій, котру можна викликати з різних місць програми, що зветься ***головною*** або ***викликаючою***. Механізм підпрограм може мати вкладеність достатньо великої кількості рівнів. Це означає, що підпрограма, яку викликає головна програма, може у свою чергу викликати інші підпрограми, які також викликають якісь потрібні підпрограми і так далі (рис. 8.45).

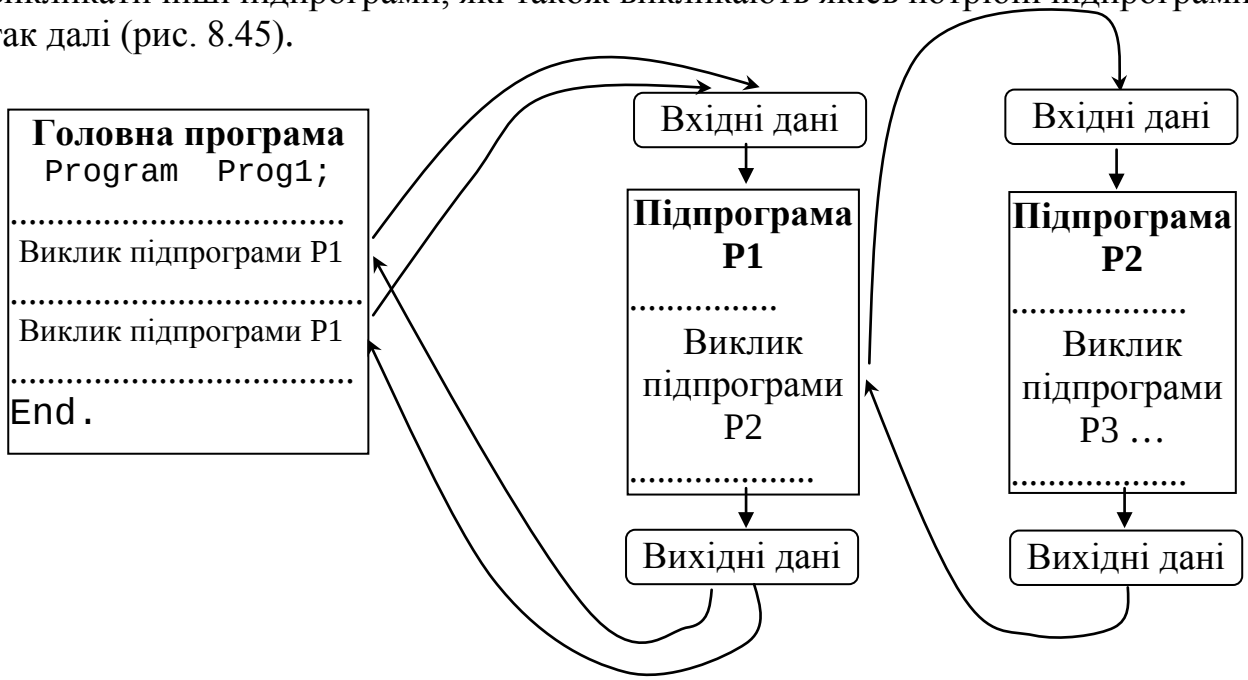


Рис. 8.45. Взаємодія головної програми з підпрограмами, що викликаються

Розрізняються два види підпрограм:

❶ ***Підпрограма–функція***, результатом роботи якої є одне значення будь-якого типу (дійсного, цілого, стрінгового, логічного та ін.).

❸ ***Підпрограма–процедура***, результатом роботи якої може бути декілька значень різних типів або послідовності наборів даних різних типів (дійсних, цілих, масивів, записів та ін.).

Тому розрізняються і форми їхніх викликів. Ім'я функції використовується у виразах у виді ***операнда, тобто елемента виразу***. Навпаки, виклик процедури завжди є ***оператором***. Але, так як загальні механізми їх роботи практично однакові, тут і далі під підпрограмами будемо розуміти і процедури і функції.

Процедури і функції описуються у спеціальному розділі описової частини головної програми слідом за описом її констант і змінних. Опис кожної з цих підпрограм починається з заголовка, який містить назву відповідного блоку операторів (`Procedure` або `Function`), ім'я цього блоку, список формальних параметрів, а функція ще додатково об'яву типу результату, який вона повертає у програму, що її викликає (рис. 8.46).

```
Procedure S1(a, b, c, …: real, i, j, k, … : integer;
                                var S, T, …:real);
begin
    <Оператори>
    S := 3.141592653 * a * b;
    T := 2 * S;
{Процедура повертає значення виразів через параметри S і T}
     <Оператори>
end;

 Function S2(a, b, c, …:real, i, j, k, … : integer):
real;
Begin
      <Оператори>
 S2 := 3.141592653 * R * R;
{Функція повертає результат через ім'я S2}
      <Оператори>
end;
```

Рис. 8.46. Схематичне уявлення описів процедури і функції

Заголовки підпрограм складаються з:

❶ Визначених зарезервованих слів (`Procedure` або `Function`).

❷ Імен цих підпрограм, що визначаються у відповідності з загальними правилами побудови ідентифікаторів.

❸ Параметрів, що уявляють собою перелік імен об'єктів для позначення початкових даних і результатів їхньої роботи з обов'язковим описом типів цих даних. Ці параметри звуться ***формальними.***

Заголовок підпрограми-функції завершується описом типу результату, який вона повертає у програму, що її викликає.

У мові ТП можливо описувати підпрограми, що не містять перелік формальних параметрів (рис. 8.47).

```
Procedure S1;
begin
.....
          <Оператори>
end;
```

```
Function S2: real;
begin
.....
          <Оператори>
end;
```

Рис. 8.47. Опис процедур і функцій,
які не містять формальних параметрів

Функція відрізняється від процедури також і тим, що у її кінці обов'язково повинен знаходитися оператор привласнення обчисленого результату до імені цієї функції (рис.8.48, стрічки 4 і 6).

```
Program  Sfunc;
Var
     x, y, z : real;
Function  Sqrt_11(elm :Real; {Ф-ція обчислює корінь другого
                                                ступеня}
  Begin
     Sqrt_11 := sqrt(elm);  {Тут elm – формальний параметр}
  End;
Begin
   X := 24.68;
  Y := Sqrt_11(X); {Тут X – фактичний параметр, тобто той, що має
                                значення}
   Writeln('Значення Y=', Y);
End.
```

Рис. 8.48. Привласнення значення дії функції до її імені (**Sqrt_11**)

Змістовна частина підпрограм уявляє собою ***блок*** і складається з:

❶ Розділу описів (міток, констант, типів, змінних, процедур, функцій).

❷ Розділу операторів, що уявляє собою складовий (рос. – составной) оператор (`begin … end`).

❸ Закінчується блок підпрограм точкою з комою.

Практично завжди відносно прості вирази можна реалізувати і за допомогою процедур, і за допомогою функцій. Компоновка використання змінних може бути різноманітною. Це можна відслідкувати на реалізації демонстраційної програми GLAVN, у якій показана взаємодія формальних, фактичних, глобальних і локальних параметрів (рис. 8.49).

```
Program GLAVN;
Const GLOB1 = 5.0;
Var
      GLOB2, FACT1, FACT2, V,S : Real;
Procedure Demo1(FORM1 : Real; Var FORM2 : real);{Перша
                                                  процедура}
   Const Lock1 = 3.141592653; {У локальній константі Lock1–
                                  значення π}
  Var
      Lock2, Lock3 : real;
Begin
   {Процедура Demo1 обчислює значення площі круга S = π·R², а потім об'єм}
   {циліндра за формулою V = H * S = H * π * R². Значення (аргумент)   }
   { R подається через формальний параметр FORM1, Обчислене значення }
   { V повертається через фактичний параметр FACT, який підставляється на}
    {місце формального параметру FORM2, а значення площі S передаємо}
     {через глобальну змінну GLOB2. Висота циліндра h знаходиться у }
      {глобальній константі GLOB1. Lock2, Lock3 – локальні змінні.}
LOCK2 := LOCK1 * SQR(FORM1); {S засилаємо у локальну змінну LOCK2}
   GLOB2 := LOCK2;
     FORM2 := GLOB1 * LOCK2;
end; {Proc Demo1}
{-------------------------------------------------------------------------------}
Procedure Demo2;{Друга процедура, яка працює без формальних параметрів}
Begin S := PI * SQR(FACT1);
      v := GLOB1 * S; end; {Proc Demo2}
Begin {Операторна частина програми}
  Writeln('Vvedite radius osnovanija cilindra u zminnu FACT1:');
   Readln(FACT1); {Значення радіуса R уведено у змінну FACT1}
    DEMO1(FACT1, FACT2);
     Writeln('Znach.R',R,FACT1,'Znach.S=',GLOB2,
'Znach.V=', V);
      DEMO2;
  Writeln('Znach.R',R,FACT1,'Znach.S=',GLOB2,'Znach.V=', V);
end.
```

Рис. 8.49. Приклад використання процедур з параметрами і без параметрів, а також різних типів параметрів і змінних

Слід розподіляти змінні і константи що описані у головній програмі і у підпрограмі. Ті змінні і константи, що описані у головній програмі звуться ***глобальними*** і їх так звана ***"область видимості"***, тобто простір, де їх можна використовувати, поширюється на усі процедури і функції, що описані у цій програмі. Окрім цього, їхні значення можна використовувати, передаючи через фактичні параметри, або вживаючи у виразах підпрограм безпосередньо.

Змінні і константи, що описані у підпрограмах можна вживати тільки у них самих, тому звуться вони ***локальними***.

Треба знати і розуміти, що імена глобальних змінних і констант можуть співпадати з іменами:

❶ Формальних параметрів підпрограм.

❷ Локальних параметрів у підпрограмах.

Але механізми мови ТП і компілятор слідкують за цими ситуаціями і роблять так, щоб у програмах усі ці об'єкти розглядалися і оброблялися як абсолютно різні. Тобто їх значення не перехрещуються. Але тоді глобальні параметри зі схожими іменами використовувати у підпрограмах з такими же іменами вживати ***неможливо*** (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
                    FACT1 := GLOB1 * SQR(FORM1);
                    GLOB2 := FACT1 * GLOB1;
    end;
Begin
……..
End.
```

Рис. 8.50. Перехрещення глобальних і локальних змінних

Згідно цього фрагмента програми, змінні **GLOB1** і **FACT1,** що описані як локальні параметри процедури Demo1 мають зовсім інші значення, ніж глобальні змінні з такими ж іменами **GLOB1** і **FACT1** у головній програмі!

А при передачі значень у підпрограми через фактичні параметри існує два головних механізми їх подальшого використання: ***“за значенням”*** та ***“за адресою”.***

Передача параметрів "за значенням" використовується у тих випадках, коли повертати у головну програму за їх допомогою ніяких даних не потрібно.

Взагалі, при написанні підпрограм, треба мати на увазі, що для них, формальні параметри (наприклад **FORM1** чи **FORM2** на рис. 8.51) уявляються просто "пустушками", які потім заміщуються при виклику конкретними значеннями, котрі можна потім використовувати у потрібних діях в операторах головної програми (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 типи передачі формальних параметрів:
–"за значенням" – формальний параметр **FORM1**
–"за адресою" – формальний параметр **FORM2**

Отже об’єкт або так звана “пустушка” FORM1 зветься формальним параметром підпрограми Demo1. А змінна FACT1, яка передає конкретне значення (2.3) і підставляється при виклику процедури Demo1 на місце формального параметра FORM1 зветься фактичним параметром.

FORM1 забезпечує при викликах процедури передачу фактичних параметрів "за значенням". Це означає, що значення аргументу (тобто фактичного параметру) просто призначається відповідному формальному параметру. Іншою мовою, перед початком роботи процедури обчислюється

конкретне значення фактичного параметра і він може передаватися у підпрограму таким чином:

❶ У вигляді значення глобальної змінної (наприклад – GLOB1), яке передається у процедуру PP2 при її виклику через формальний 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. Фактичний параметр – глобальна змінна

❷ У вигляді деякого значення (числової константи) (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. Фактичний параметр – константа

❸ У вигляді виразу, який обчислюється до передачі у підпрограми (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. Фактичний параметр – вираз

В усіх трьох випадках після обробки виклику процедури отримане значення копіюється у відповідний формальний параметр, що належить процедурі. Як тільки починається виконання процедури, вже ніякі зміни значення формального параметра не оказують впливу на значення відповідного аргументу. Це значить, що по закінченні роботи процедури аргумент буде мати точно таке ж значення, яким воно було до початку роботи процедури, незалежно від того, що діялось з цим формальним параметром (рис. 8.55)

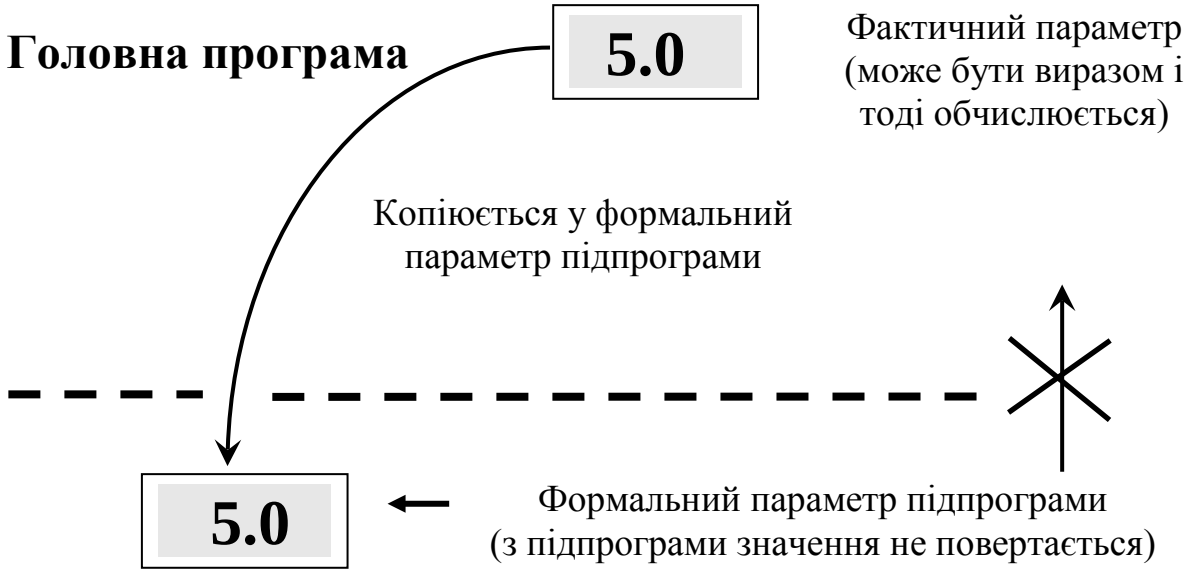


Рис. 8.55. Механізм передачі параметрів “за значенням”

Іншим чином виконується ***передача параметрів "за адресою", яка ще зветься "передачею за посиланням (рос. – по ссылке)" або передачею за var–параметром або параметром–змінною.***

За такий спосіб передачі параметра у підпрограму (процедуру або функцію) пересилається вже не ***значення аргументу***, а ***адреса його місцезнаходження*** (адреса у пам'яті комп'ютера). Якщо формальний параметр має атрибут `var` у його опису у голові підпрограми, а відповідний фактичний аргумент є змінною, то будь–які зміни значення формального параметра у тілі підпрограми будуть замінені його значенням за адресою аргументу, що підставлений. Це пов'язано з тим, що у цьому випадку формальний і фактичний параметри займають одне і те ж місце у пам'яті (рис. 8.56)

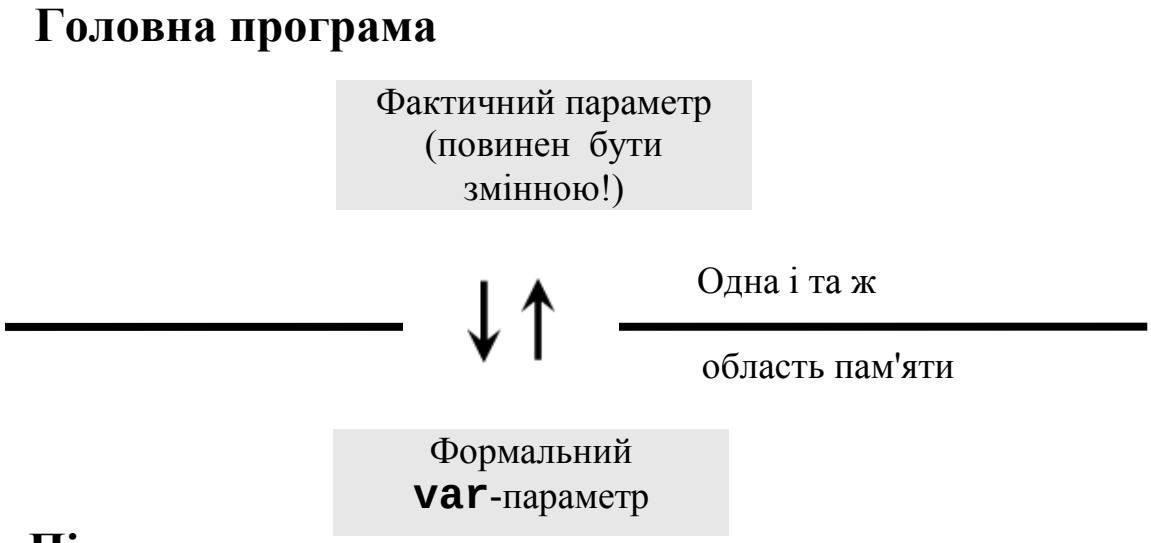


Рис. 8.56. Механізм передачі параметрів “за посиланням” (рос. – по ссылке) (або за адресою)

Тому, щоб повідомити компілятор про намір користувача передавати деякий параметр за адресою (за посиланням), у заголовку підпрограми потрібно указати перед його ім'ям службове слово "`var`".

Таким чином, все що буде відбуватися з цим параметром у підпрограмі, буде змінювати його значення у головній програмі. Наведемо приклад взаємодії значень формальних параметрів (рис. 8.57).

Як вже з'ясувалося, у тілі підпрограми використання глобальних змінних і `var`–параметрів практично не відрізняється. Важливо інше.

По-перше, якщо глобальна змінна фігурує у виразах і підпрограмах, її ім'я замінити вже ніяк неможливо, але можна змінити значення до виклику відповідної підпрограми. Тоді, якщо її ім'я фігурує у правій частині оператора привласнення у підпрограмі, зміна її значення у головній програмі відіб'ється на значенні усього виразу (рис. 8.58).

```
Var
    GLOB1, GLOB2 : Real;
Procedure PP5 (FORM1 : Real; var FORM2 : real);
   Var
                Lock1 : Real;
 Begin
   FORM2 := FORM1+ 2.5; {Значення GLOB2 у головній програмі }
    { зміниться на (5.0 + 2.5) = 7.5, бо FORM2 var-параметр!}
    FORM1 := FORM1 – 3.32; {Так взагалі писати не можна (!!!),}
        {бо FORM1 – значення, якому нічого привласнити не можна!}
 end;
……….
Begin
  GLOB1 := 5.0;
   PP5 (GLOB1,GLOB2); {Виклик процедури PP5}
  Writeln('GLOB1=', GLOB1:8:2, 'GLOB2=', GLOB2:8:2);
End.
```

Рис. 8.57. Приклад формування виклику процедури

При завершенні роботи програми (рис.21.13) на екран буде виведено:

```
GLOB1=    5.00    GLOB2=    7.50,
```

Тобто значення GLOB1 (що підставлялося на місце параметру-значення) – не змінилося, а помінялося тільки значення у змінній GLOB2, яка підставлялася на місце параметру-змінної.

```
Var
    GLOB1, GLOB2 : Real;
Procedure PP6 (FORM1 : Real);
 Begin
      GLOB2:= GLOB1 + FORM1;
 end;
Begin
   GLOB1 := 5.0;
     PP6 (5.0);      {Стає GLOB2 =10 }
      GLOB1 := 10;
       PP6 (10);  {Стає GLOB2 =20 }
End.
```

Рис. 8.58. Приклад впливу зміни значень глобальної змінної GLOB1 на результат обчислення процедурою PP6

Фундаментальним уявленням є те, що результат дії процедури може повертатися у головну програму лише або через ***глобальний параметр*** (наприклад – GLOB2) або через var–***параметр***.

І тут треба розуміти, що взагалі необхідно розробляти підпрограму таким чином, щоб результат її дії повертався ***тільки*** через var–параметр. Це пов'язано з тим, що у мові ТП є змога компілювати підпрограми відокремлено від головної програми у виді так-званих ***модулів***[25]. У цьому випадку зв'язок з підпрограмами повинен забезпечуватися тільки через формальні (прості і var) параметри (рис. 8.59).

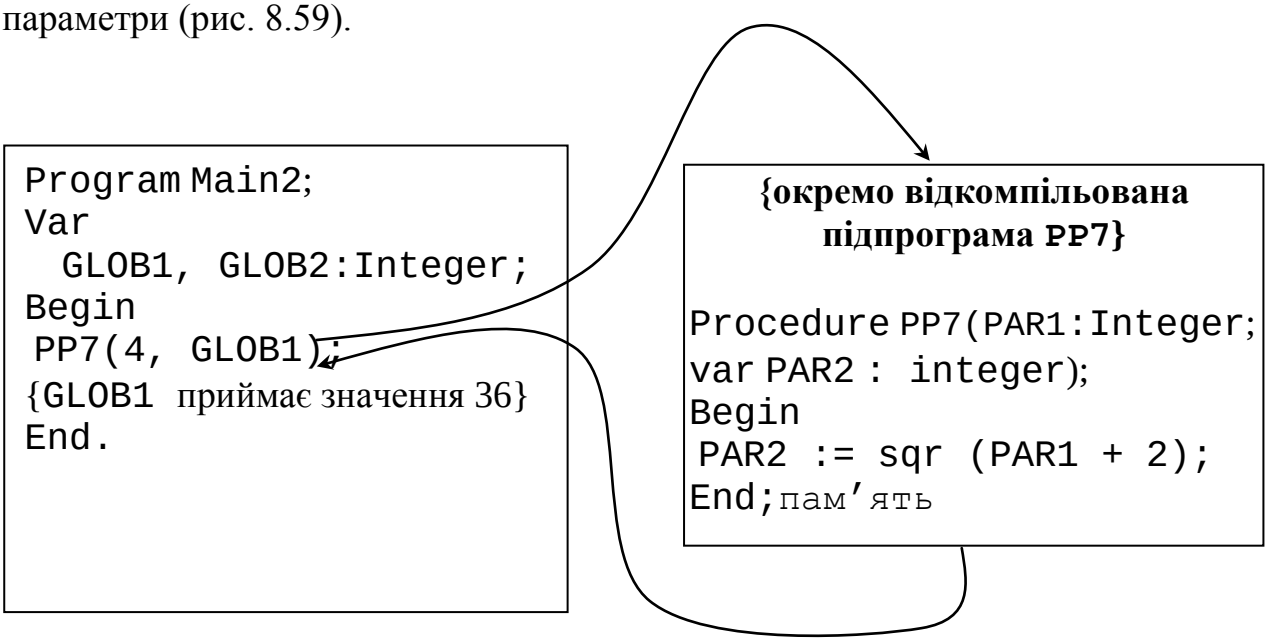


Рис. 8.59. Виклик процедури, що откомпільована окремо і розташована у модулі

Розглянемо приклад взаємодії об'єктів у програмах, процедурах та функціях при різних комбінаціях передачі до підпрограм даних у вигляді глобальних і фактичних параметрів. Для цього розглянемо таку задачу.

Реалізувати у вигляді процедури і функції обчислення за наступною формулою: $Z = f\ (x, y) = \dfrac{x + y^2}{a + b}$ (8.14).

Зрозуміло, що значення постійних величин *a* та *b* будуть визначені у програмі у вигляді констант. А от значення змінних *x* та *y* необхідно уводити у процесі рішення, тобто на етапі виконання програми. Проектування та кодування опису процедур і функцій припускає досить багато варіантів різноманітних дій, так як дані у підпрограми можемо передавати і через глобальні змінні, і через фактичні параметри. Тому розглянемо деякі можливі приклади, які витікають з того, що дані у підпрограми можуть передаватися двома головними способами: по значенню (параметр-значення) або за адресою.

[25] Модуль – програма, яка розглядається як окреме ціле у процесах збереження, трансляції та об'єднання з іншими програмними модулями при їх завантаженні у оперативну пам'ять комп'ютера

(параметр-змінна). У наступному прикладі обчислені за допомогою різних підпрограм значення вищенаведеної формули заносяться у змінні Z1, Z2, ... Z6. Зверніть увагу на процедуру P2 та функцію F5 що сконструйовані взагалі без формальних параметрів (рис. 8.60).

```
Program FXY;
    const a=3.4, b=-7.302;
 var
    x, y,
     Z1, Z2, Z3, Z4, Z5, Z6 : real;

{ Процедура P1 спроектована з параметрами. Усі дані уводяться}
{як параметри-значення, а результат виводиться через параметр-змінну yy}
Procedure P1(x1, y1, a1, b1 : real; var yy :real);
   begin yy:=(x1+sqr(y1))/(a+b) end;

{ Процедура P2 без параметрів. Використовуються глобальні змінні}
{програми FXY – x,y,a,b, а результат повертається через змінну Z2}
Procedure P2; begin Z2:=(x+sqr(y)/(a+b)); end;

{Процедура P3 з одним параметром-змінною zz, через яку у головну
програму FXY повертається обчислене за формулою значення }
Procedure P3 (var zz : real);
   begin zz := (x+y*y)/(a+b); end;

{Функція F4 з чотирма параметрами-значеннями. Результат обчислення за
формулою повертається через ім'я функції F4 }
Function F4(x2,y2,a2,b2 : real):real;
   begin F4 := (x2+sqr(y2))/(a2 +b2); end;

{Функція F5 без параметрів. Використовуються глобальні змінні програми.
А через які саме? }
Function F5:real; begin F5:=(x+sqr(y))/(a+b); end;

{Функція F6 з двома параметрами-значеннями, через котрі до неї
передаються значення змінних x та y }

Function F6(xx, yy :real):real;
   begin F6:=(xx+yy*yy)/(a+b); end;
```

Рис. 8.60. Програма опису функцій по різному

{ продовження програми рис. 8.60 на наступному листі }

```
      {}
Begin
      P1(4.1,2.56,-3.1,4.6,Z1);  {А}
        Z3:=F4(3,7,2,61);        {Б}

      Writeln('Vvedite x,y:');
        readln(x,y);
  P2;
    P3(Z4);
      Z5:=F5;
        Z6:=F6(Z4-2.5*Z5, ln(Z1/Z2));
  Writeln(Z1, Z2, Z3, Z4, Z5, Z6);
    Readln;
End.
```

Рисунок 8.60. Приклади реалізації обчислення формули (8.14) за допомогою різних варіантів конструювання підпрограм

Програма на рисунку 8.60 дозволяє відповісти на дуже цікаве запитання: Чи можливо на початку виконання програми **FXY** визвати будь-яку процедуру чи функцію з вищеописаних, якщо змінні a, b, x, y, що визначають обчислення за формулою (8.14) ще не ініціалізовані (тобто у них не уведені початкові значення)?

Стрічки **А** та **Б** на другій частині рисунку 8.60 дають позитивну відповідь на це запитання. Так, це можливо зробити, якщо звертатися до деяких процедур і функцій фактичними параметрами за значеннями, які є константами (дивись звертання до процедури P1 та функції F4 у стрічках А та Б).

Вправи

1. Написати функцію для обчислення знака числа по формулі: $sign(x) = \begin{cases} 1, x > 0 \\ 0, x = 0 \\ -1, x < 0 \end{cases}$.

2. Обчислити наступні вирази: sign(a*b), sign(a/b), sign(-b), sign(-a), скориставшись функцією, що була написана для попереднього прикладу.

3. Написати функцію для обчислення суми цифр тризначного натурального числа. Обчислити суму цифр для чисел від 100 до 120.

4. Написати функцію для обчислення факторіалу n!=1•2•3•...•n. Обчислити факторіали чисел від 1 до 7.

5. Написати функцію для зведення числа x у ступінь m (m -натуральне число). Обчислити x^3, x^4, x^5.

6. Написати функцію для обчислення арксинуса. Обчислити arcsin(0.9), arcsin(0.1), arcsin(--0.9), arcsin(0.99). Для обчислень скористатися формулою: $\arcsin(x) = arctg\left(\frac{x}{\sqrt{1-x^2}}\right)$.

7. Для пунктів 1-6 написати ті ж алгоритми, тільки у виді процедур.

8. Написати програму обчислення визначених інтегралів за формулою трапеції

$$\int_a^b y\,dx = \frac{b-a}{n}\cdot\left(\frac{y_0+y_n}{2}+y_1+y_2+\ldots y_{n-1}\right),$$

для наступних виразів:

a) $\frac{1}{1+x}$ *b*) $\frac{\ln(1+x)}{1+x^2}$

c) $\frac{\sin x}{x}$ *d*) e^{-x^2}

9. Написати підпрограми обчислення визначених інтегралів за допомогою формули трапеції для функцій *a-d* з точністю близько $\varepsilon = 0.00001$.

8.23. Робота з масивами. Приклади багатовимірних масивів

У лінійній алгебрі та інших розділах математики існує велика кількість задач, пов'язаних з використанням ***матриць***. Як Ви знаєте, матрицею називається сукупність чисел a_{ij}, розташованих у вигляді прямокутної

$$\begin{pmatrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & a_{1m} \\ a_{21} & a_{22} & \cdot & \cdot & \cdot & a_{2m} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & \cdot & \cdot & \cdot & a_{nm} \end{pmatrix}$$

таблиці[26]:

де a_{ij} – елементи матриці.

Індекси i та j означають, що елемент a_{ij} розташований на перетинанні (рос. – пересечении) i-го рядку та j-го стовпця матриці. Якщо матриця має n рядків та m стовпців, то вона називається матрицею ***розміру*** $n \times m$. Матриця називається ***квадратною*** матрицею порядку n , якщо $n = m$. При $n \neq m$ матриця називається ***прямокутною***. Прямокутна матриця розміру $n \times 1$, яка складається з одного стовпця

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_n \end{pmatrix}$$

називається ***вектор-стовпцем***. А прямокутна матриця розміру $1 \times m$, яка складається з одного рядка

$$\begin{pmatrix} \beta_1 & \beta_2 & \cdots \beta_m \end{pmatrix}$$

називається ***вектор-рядком***. Для роботи з матрицями у Турбо Паскалі існує структура даних під назвою ***масив***.

8.23.1. Опис масивів у програмі

Масив – це регулярна структура (послідовність) ***однотипних*** даних, що оголошується спеціальною конструкцією мови ТП:

[26] Воеводин В.В., Кузнецов Ю.А. Матрицы и вычисления. – М.: Наука. Главная редакция физико-математической литературы, 1984. – 320 с.

```
Array [<ДіапазониІндексів>] of <ТипКомпонентів>;
```

Доступ до аби якої компоненти масиву забезпечується простою вказівкою її порядкового номеру. У двовимірних масивах компоненти (елементи) розміщуються у пам'яті ПК послідовно, але так, що скоріше змінюється самий правий індекс. Тобто можна сказати, що вони зберігаються по рядках (рос.-построчно).

Найбільш часто масиви використовують для збереження вектор-стовпців та вектор-рядків (одновимірні масиви) та конструювання двовимірних масивів тобто матриць, тривимірних масивів і т.д.:

```
Var
   Vector : Array [1..3] of Real; {одномірний масив Vector,
                                   аналог вектор-стовпців та
                                   вектор-рядків}
   Matrix : Array [1..3,1..3] of Real; {двовимірний масив
                                        Matrix, аналог
                                        матриці}
```

Таким чином, ми оголошуємо дві структури: `Vector` – з трьох значень типу Real, проіндексованих заданим діапазоном цілих чисел:

```
Vector[1], Vector[2], Vector[3]
```

та `Matrix` – з дев'яти значень типу Real, проіндексованих заданим діапазоном цілих чисел (так вони і зберігаються у пам'яті комп'ютера):

```
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],
```

що відповідає матриці виду:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Окрім того, що Вам зрозуміло тепер, як отримати доступ до кожного з елементів масивів, Ви можете також помітити схожість індексування елементів масиву `Matrix` з позначеннями елементів матриць, як сукупності дійсних або інших чисел a_{ij} , які розташовані у вигляді прямокутної таблиці.

Цей факт дозволяє використовувати масиви для проведення обчислень з використанням матриць і не тільки. Але Ви повинні знати деякі тонкощі роботи з масивами у ТП.

Якщо індексація елементів задається числовим діапазоном, то необхідно дотримуватись лише двох вимог:

– <ДіапазониІндексів> не може мати тип LongInt. Тобто він повинен "уміщатися" максимум у типі Word і таким чином мати значення не більше 65535;

– добуток кількості компонентів масиву на розмір компонентів у байтах не може перевищувати 65520 байт ≈ 64Кбайт.

У рамках можливостей опису структур даних ТП можна використати такі конструкції:

```
Var
  A : Array [9..99] of Char; {Масив A з 91-го символьного елемента}
  B : Array [1..100] of Boolean; {Масив B зі 100-а елементів
                                  булівського типу}
  C : Array [-10..10] of LongInt; {Масив C з 21-го числового
                                   елемента, тому ж не вимагає
                                   приведення індексів до діапазону
                                   1..N (як у мові Фортран) або 0..N (як у
                                   мові C)}
```

Більш того, аби який зі ***структурованих типів*** (до яких у ТП відносяться: масиви, записи, множини (рос.-множества) і файли) може бути описаний за допомогою службового слова Type. Наприклад, попередній опис трьох масивів за його допомогою може бути виконано 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; {Масив A з 91-го символьного елемента}
  B : S; {Масив B зі 100-а елементів булівського типу}
  C : T; {Масив C з 21-го числового елемента}
  D : T of S; {Двовимірний масив D з 21 x 21 числових елементів}
  E : Array [-10..10] of Array [1..100] of Array [-
  10..10] of Real; {Тривимірний масив E (самі уявіть собі його структуру!)
```

У цьому опису у змінних (R, S, T і U) уявляються ″форми″ завдання існування тих чи інших структур даних (тобто набори послідовностей

компонентів різних типів). А потім у операторі Var вони використовуються для визначення ідентифікаторів масивів для обробки їх у програмі. Треба чітко розуміти різницю поміж цими двома службовими словами (Type і Var). Отже, Type завдає ***тільки форму*** уявлення послідовностей компонентів різних типів даних у майбутніх структурах масивів. А службове слово Var призводить до ***розміщення*** описаних у ньому масивів даних у ***пам'яті комп'ютера.*** Це значить, що після обробки середовищем ТП першого рядку (Var A : R;) у пам'яті ПК буде виділено рівно 90 байтів для розміщення 90 компонентів (змінних) типу Char (тобто символів).

8.23.2. Уведення-вивід масивів

Тепер, коли Ви вже знаєте, як описувати масиви й організовувати доступ до кожного з їх елементів за допомогою циклу з параметром, Ви в змозі організувати у своїй програмі уведення й вивід масивів. Майте на увазі, що у Турбо Паскалі масиви у оперативній пам'яті розташовуються статично, що не дозволяє без спеціальних засобів (посилання, динамічні змінні) змінювати їх границі. Вихід з цієї незручної ситуації полягає у тому, що Ви повинні у своїй програмі описати масиви на максимальну довжину, а потрібні розміри векторів та матриць уводити у програмі. Також зверніть увагу на те, що уведення й вивід масивів виконуються по елементах (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; {Масив A векторного типу з 50 елементів}
  B: MatrixType; {Масив B матричного типу з 50x50 елементів}
  i, j : Integer; {Параметри циклу для обробки векторів та
                   матриць}
  n, m : Integer; {Кількість елементів масивів відповідно у
                   стрічці та у стовпчику}
Begin
ClrScr;
   { == Ввод-вектору A ==}
WriteLn('Введіть кількість елементів вектора A, які
         обробляються (не більше 50)');
ReadLn(n);
WriteLn('Введіть n елементів вектору A:');
for i:=1 to n do {Організуємо цикл по кількості n елементів}
  begin
    Write (' A[',i,'] = '); {Зручно бачити індекси елементів,
                             які вводяться}
```

Рис. 8.61. Ввід та вивід масивів на мові Турбо Паскаль

```
    ReadLn (A[i])
  end;
WriteLn; {Встановлюємо курсор на нову стрічку}

   { == Вивід-вектора A ==}
WriteLn ('Введені значення вектора A: ');
 for  i:=  1  to  n  do {Використовуємо оператор Write з
                        форматуван-}
  Write (' A[',i,'] = ',A[i] : 9);{ням, щоб економити
                                   простір екрану}
WriteLn; {Встановлюємо курсор на нову стрічку}

   {== Ввід матриці B ==}
WriteLn ('Введіть кількість елементів матриці B, які
          обробляються - n, m (не більше 50x50)');
ReadLn(n, m);
WriteLn('Введіть nxm елементів матриці B:');
for i:= 1 to n do { Ввід матриці B потребує }
  for j:= 1 to m do { двох циклів з параметром}
    begin
      Write (' B[',i,','j,'] = ');
      Read (B[i,j]); {Читання элементу матриці}
    end;
WriteLn; {Установка курсору після читання останнього елементу
масиву}

WriteLn('Матриця B з', n,' стрічок та',m,'
         стовпчиків:');
for i := 1 to n do { Вивід матриці B потребує }
  for j:= 1 to m do { двох циклів з параметром}
    Write(' B[',i,','j,'] = ',B[i,j] : 9);
     WriteLn; {Встановлюємо курсор на нову стрічку}
ReadLn
End.
```

Рис. 8.61. Ввід та вивід масивів на мові Турбо Паскаль (продовження)

8.23.3. Стандартні прийоми роботи з векторами і матрицями. Підсумовування елементів масиву

При роботі з матрицями, як правило, приходиться мати справу з набором стандартних прийомів, та їх композицією. Тому Вам потрібно знати цей набір, щоб у майбутньому конструювати з нього фрагменти алгоритмів програм, які можуть виникнути з постановки конкретної задачі.

Самою простою задачею, яка виникає при роботі з векторами (матрицями) є підсумовування їх елементів. Для цього у програмі описуємо змінну Sum того ж типу, що й масив і у циклі по черзі додаємо до неї елементи масиву:

```
... {Сумування елементів одновимірного масиву – вектору}
. . .
Type
  VectorType = Array[1..20] of Real;
Var
  A: VectorType;
  i, n : Integer; {Параметр циклу та розмір вектора}
  Sum  : Real; {Змінна Sum для підсумовування}
. . .
  Sum := 0; {Підготовка змінної Sum для підсумовування!!!}
  for i: =1 to n do
    Sum := Sum + A[i];
. . .
```

Для двовимірного масиву треба додати ще один цикл з параметром для забезпечення перебору усіх індексів двовимірного масиву:

```
.   .   .
   { Сумування елементів двовимірного масиву}
Type
  MatrixType = Array[1..20, 1..25] of Real;
Var
  B: MatrixType;
  i, j, n, m : Integer; {Параметри циклів та границі масиву}
  Sum : Real; {Змінна Sum для підсумовування}
.   .   .
  Sum := 0; {Підготовка змінної Sum для підсумовування}
  for i:= 1 to n do {Рухаємось уздовж рядку (рос.-строки)}
    for j:= 1 to m do {Рухаємось уздовж стовпця}
      Sum := Sum + B[i, j]; {Підсумовуємо у Sum кожен
                             елемент матриці}
{Для обчислення середнього арифметичного додаємо оператор}
  Sum := Sum / (n*m);}
.   .   .
```

8.23.4. Використання лічильника (рос.-счётчика)

Ще однією задачею, яка виникає при роботі з масивами, є підраховування елементів матриці з потрібними характеристиками, наприклад, позитивних елементів. Це потребує уведення у тіло циклу умовного оператора:

```
. . .
{Визначимо кількість позитивних елементів Count типу Integer}
  Count := 0; {Підготовка змінної Count для підсумовування!!!}
  for i:= 1 to n do
    if A[i] > 0 then{Визначення позитивного елементу}
       Count := Count + 1; {Збільшення кількості позитивних
                                    елементів}
. . .
```

Тепер для Вас не виникне труднощів щоб скласти алгоритм для визначення позитивних елементів двовимірного масиву тобто матриці. Окрім цього, тепер Ви можете використати цей алгоритм для визначення:

1. числа негативних, нульових елементів;
2. числа елементів, рівних заданому числу;
3. Числа елементів великих чи менших заданого числа;
4. Числа елементів заданих значень, що знаходяться в інтервалі і т.д.

Тепер давайте використаємо наші знання для конструювання алгоритму для знаходження середнього арифметичного позитивних елементів двовимірного масиву. Для цього треба обчислити суму позитивних елементів матриці та їх кількість, тобто застосувати водночас вже відомі нам алгоритми підсумовування та «лічильника»:

```
. . .
{Композиція: «підсумовування» + «лічильник»}
{Знаходимо середнє арифметичне позитивних елементів }
Type
  MatrixType = Array[1..20, 1..25] of Real;
Var
  B: MatrixType;
  i, j, n, m : Integer; {Параметри циклів та границі масиву}
  Sum : Real; {Змінна Sum для підсумовування}
.  .  .
  Sum := 0; {Підготовка змінної Sum для підсумовування}

  Sum := 0;
  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;  {Середнє арифметичне позитивних елементів }
   .   .   .
```

8.23.5. Визначення max/min елемента масиву

Дуже часто програмування алгоритмів роботи з матрицями потребує визначення максимального або мінімального елементу. Для цього існує дуже простий алгоритм, який Ви повинні знати:

```
   .   .   .
   Max := A[1]; {Вважаємо, що перший елемент – максимальний }
   for i:= 2 to n do {Тому починаємо з другого елементу }
     if A[i] > Max {Порівнюємо кожний наступний елемент з поточним
                    максимальним значенням }
       then Max:= A[i];{Поновлюємо Max новим більшим значенням}
   {Після виконання циклу у Max знаходиться максимальне значення з A }
   . . .
```

На базі цього алгоритму можемо сконструювати алгоритм визначення мінімального елемента двовимірного масиву та його індексів:

```
. . .
   Var
     IndI, IndJ : Integer; {Змінні для збереження індексів}
   . . .
   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 Робота з парними/непарними елементами масиву

Ще одним досить корисним алгоритмом, який може Вам знадобиться при роботі з масивами, є пошук парних/непарних елементів цілих масивів та їх підсумовування. Для цього у Турбо Паскалі існує спеціальна вбудована функція `Odd(x)` (укр. – непарний), яка при цілому аргументі `x` повертає значення `True`, якщо `x` непарне число, та `False` у випадку парного числа:

```
   .   .   .
```

{Знаходимо суму непарних елементів з використанням функції `Odd(i)`}

```
Sum := 0;
for i:=1 to n do
  if Odd(i) then {для парних умова – not Odd(i)}
     Sum := Sum + A[i];
.   .   .
```

8.23.7. Вирішення практичної задачі з масивами

Тепер, коли Ви вже володієте основними прийомами роботи з матрицями, спробуємо вирішити конкретну задачу згідно сформульованої у п.1 технології роботи у середовищі ТП. Наприклад, маємо таку задачу:

Дано два вектори $\overline{X} = \left(x_1, \quad x_2, \quad x_3 \right)$ та $\overline{Y} = \left(y_1, \quad y_2, \quad y_3 \right)$. Знайти кут між ними.

1. Першим кроком, Вам необхідно вирішити задачу у її предметній області, тобто довести її до вигляду, коли Ви можете уявити собі всі дії по створенню алгоритму вирішення задачі. Як можна встановити з довідника з математики[27], ця задача походить з області векторного числення, з розділу "Геометричні додатки векторної алгебри", де наведено формулу обчислення куту поміж трьохвимірними векторами $\overline{X}$ та $\overline{Y}$:

$$\cos\varphi = \frac{\overline{X} \quad \overline{Y}}{\left|\overline{X}\right| \quad \left|\overline{Y}\right|} \tag{8.15}$$

де $\overline{X} \quad \overline{Y} = x_x y_x + x_y y_y + x_z y_z$ – скалярний добуток векторів $\overline{X}$ та $\overline{Y}$;

$\left|\overline{X}\right|$ – довжина вектора $\overline{X}$:

$$\left|\overline{X}\right| = \sqrt{X^2} = \sqrt{x_x^2 + x_y^2 + x_z^2}; \tag{8.16}$$

$\left|\overline{Y}\right|$ – довжина вектора $\overline{Y}$.

З урахуванням цих залежностей, остаточна формула має вигляд:

$$\cos\varphi = \frac{x_x y_x + x_y y_y + x_z y_z}{\sqrt{x_x^2 + x_y^2 + x_z^2} \quad \sqrt{y_x^2 + y_y^2 + y_z^2}} \tag{8.17}$$

2. Другим кроком Ви повинні вибрати вхідні й вихідні дані. Вхідними даними у нас будуть два вектори $\overline{X} = \left(x_1, \quad x_2, \quad x_3 \right)$ та $\overline{Y} = \left(y_1, \quad y_2, \quad y_3 \right)$, кожен з котрих складається з трьох компонент. Вихідними даними буде значення косинусу куту поміж ними $\cos\varphi$.

3. Проходимо по розділам описів. Потрібен модуль `Crt` для очистки екрану.

4. Четвертим кроком буде вибір структур даних:

```
Const
  n = 3;
Type
  VectorType = Array[1..n] of Real;
```

[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.

```
Var
  X, Y : VectorType; {Масиви для векторів}
  XY,                {Змінна для скалярного добутку }
  SumX,  {Змінна для підсумовування квадратів компонент X̄ }
  SumY,  {Змінна для підсумовування квадратів компонент Ȳ }
  CosFi : Real; {Змінна для обчислення результату}
  i : Integer; {Змінна-параметр циклу}
```

Оскільки Ви вже маєте деякий досвід у програмуванні, то можете передбачити, що буде потрібно окремо підсумовувати скалярний добуток у чисельнику формули (8.17), та суми квадратів компонент векторів у знаменнику, для чого потрібні окремі змінні.

5. Розробка інтерфейсу не представляє нічого складного. Увід векторів `A` вже розглянуто у розділі 8.23.2, а вивід результату й зовсім простий.

6. Для роботи з масивами нам буде потрібна керуюча структура `for`, у яку ми й помістимо усі обчислення з векторами.

7. Тепер Ви повинні зібрати все до купи і це й буде кінцева програма обчислення (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;
  {Тут має бути заповнення масивів для векторів}
  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. Обчислення куту поміж двома трьохвимірними векторами

Вправи

1. Дано два вектори $\overline{X} = (x_1, \ x_2, \ldots x_n)$ та $Y = (y_1, \ y_2, \ldots y_n)$. Знайти кут між ними.

2. В масиві $D=(d_1, d_2, ..., d_{12})$ визначити середнє арифметичне *S* негативних елементів та їх кількість *K*.

3. В масиві $D=(d_1, d_2, ..., d_{12})$ визначити добуток *P* позитивних елементів, кількість *K* нульових елементів та суму *S* негативних елементів.

4. В масиві $B=(b_1, b_2, ..., b_{10})$ визначити *M* максимальний елемент та його індекс. Максимальний елемент замінити на “0”.

5. Дано матрицю *Q(3,6)*. Визначити мінімальний і максимальний елементи матриці та їх добуток.

6. Дано матрицю *A(4,6)*. Визначити суму *S(6)* елементів, яки перевищують по модулю одиницю, в кожному парному стовпці і кількість таких елементів.

7. Дано матрицю *X(6,6)*. Визначити максимальний елемент на головній діагоналі та суму елементів головної діагоналі.

8. Дано матрицю *C(7,7)*. Визначити добуток позитивних елементів матриці, розташованих на перехресті парних рядків і непарних стовпців.

8.24. Модулі та робота з ними

На протязі роботи з цим навчальним посібником Ви вже неодноразово використовували модулі у своїх програмах (згадайте підключення бібліотечного модулю Турбо Паскалю `Crt` для використання процедури очищення екрану `ClrScr,` яка визначена у ньому). Всі системні бібліотеки Турбо Паскалю реалізовані як модулі, і щоб використовувати їх Ви повинні помістити у програмі стрічку:

```
Uses Crt, Graph; {Або інші потрібні модулі}
```

Взагалі ***модулі*** призначені для підтримки принципів модульного програмування при розробці програм. Відповідно до цього принципу взаємовплив логічно незалежних фрагментів програми повинен бути зведений до мінімуму. Але найважливішими корисними рисами модулів є можливість зберігати в них оформлені у вигляді процедур і функцій алгоритми, які Ви розробляєте впродовж Вашої роботи і бажаєте ***повторно використовувати.*** Окрім цього, Ваша програма, у якій описані модулі, "бачить" усі дані модулів (визначення типів, констант, змінних), тобто їх не треба описувати у основній програмі.

Отже, ***модуль*** розділяється на чотири частини:

1. Заголовок модуля (`Unit Name;`)
2. Розділ оголошень чи інтерфейс (`Interface`)
3. Розділ реалізації (`Implementation`)
4. Розділ ініціалізації (між `Begin` і `End`)

Усі блоки, що складають ці розділи, є необов'язковими і можуть або бути відсутніми, або з'являтися неодноразово. Найменший за розміром, безплідний але правильний з точки зору мови ТП модуль має вигляд:

```
Unit Empty;
Interface
Implementation
End.
```

Зверніть увагу, що крапки з комою після ключових слів відсутні. Якщо немає розділу ініціалізації, то `Begin` не ставиться.

Взагалі структура корисного ***модуля*** подібна до паскаль-програми:

```
Unit UnitName; {Заголовок модуля. Рекомендуємо
                UnitName <= 8 символів}
Interface {Розділ оголошень, які будуть використовуватись у цьому
           модулі та у програмах, які його будуть використовувати!}
Uses ...; {Список модулів, які використовуються у цьому модулі}
Const ...; {Оголошення бібліотечних констант}
Type ...; {Блок оголошень бібліотечних типів}
Var ...; {Блок оголошень бібліотечних перемінних}
```

```
Procedure ProcName(FormalParametersList); {У цьому розділі записуються тільки заголовки процедур з формальними параметрами!}
Function FuncName(FormalParametersList):Type; {У цьому розділі записуються тільки заголовки функцій з формальними параметрами!}

Implementation {Розділ реалізації}
Uses ...; {Використовувані при реалізації модулі}
Const ...; { Описи у розділі Implementation }
Type ...;  {     "бачить" тільки цей          }
Var ...;   {          модуль                  }
Procedure ProcName; {У цьому розділі записуються тільки заголовки процедур без формальних параметрів та їх тіла!}
Function FuncName; {У цьому розділі записуються тільки заголовки функцій без формальних параметрів та їх тіла!}
                    Begin
  {Блок ініціалізації модуля}
End.
```

Тепер ми можемо вирішити вправу 1 з розділу 22 з використанням модуля.

Приклад: Дано два вектори $\overline{X} = (x_1, \ x_2, \ldots x_n)$ та $Y = (y_1, \ y_2, \ldots y_n)$. Знайти кут між ними. Для обчислення кутів між векторами використати модуль, який повинен містити описи потрібних структур даних, процедуру вводу векторів з наперед заданою кількістю компонент та функцію обчислення куту між двома векторами з *n* компонентами.

Текст модулю `MyUnit` наведено на рисунку 8.63.

```
Unit MyUnit;
Interface
Uses
  Crt; {У цьому модулі використовуємо бібліотечний модуль Crt}
Type
  VectorType = Array[1..50] of Real;
Var
  XX, YY : VectorType; {Масиви, які використовуються у модулі,}
                       {а також у програмі, де він описаний}
  j : Integer;

Procedure VectInput(Var Vect : VectorType;
                        n : Integer);
{ VvodVect – процедура вводу вектора з n елементів}
```

Рис. 8.63. Структура модуля, для обчислення кутів поміж векторами

```
Function CosFi (Var X, Y : VectorType; n : Integer) :
                Real;
{ CosFi – функція для обчислення куту поміж векторами}
{ Z1, Z2 – одновимірні масиви з компонентами векторів}
{ n – кількість компонент кожного з векторів}
Implementation
Procedure VectInput; {Списку параметрів немає!}
Var
  i : Integer;
Begin
  for  i:=1 to n do
    begin
      Write('Vect[', i, '] = ');
      ReadLn(Vect[i]);
    end
End; {Proc VvodVect}
Function CosFi; {Списку параметрів немає!}
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 {Масиви XX та YY ініціалізуємо для використання у програмі}
  ClrScr;
  WriteLn('Введіть по 3 компоненти масивів XX і Y:');
  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 обчислення кутів між векторами (продовження)

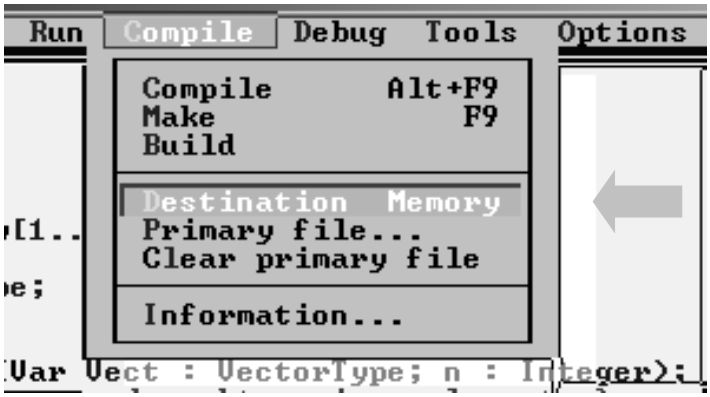


Рис. 8.64. Стандартна установка опції **Compile → Destination**

Рис. 8.65. Установка опції **Compile → Destination → Disk** для зберігання модуля на жорсткому диску

Рис. 8.66. Меню опції **Options**

Тепер, коли ми маємо свій особистий модуль, треба зробити так, щоб цим модулем можна було користуватися у Вашій програмі. Мабуть до цього моменту Ви не звертали увагу на те, що Ваші програми при натисканні клавіш **Alt+F9** компілюються у пам'яті ПК (рис. 8.64). Для модулів зручніше, щоб вони зберігалися десь на жорсткому диску. Щоб установити такий режим компілювання, треба вибрати опцію **Compile → Destination** й натиснути клавішу **Enter**. Якщо після цього Ви відкриєте меню опції **Compile,** то побачите наступні зміни (рис. 8.65).

Але не поспішайте компілювати свій модуль, бо існує ще декілька тонкощів...

Річ у тому, у Турбо Паскалі треба указувати каталоги (директорії): куди при компіляції зберігати **EXE** і **TPU** файли (**EXE & TPU Directories**) та відкіля брати модулі, які підключаються до Вашої програми (**Unit Directories**).

Для початку натисканням клавіш **Alt+O** відкрийте меню опції **Options**. А в ній виберіть команду **Directories** (рис. 8.66). При цьому, з'явиться меню **Directories** (рис. 8.67). У рядку **EXE & TPU Directories** укажіть, куди Ви бажаєте зберегти відкомпільований модуль. Пам'ятайте, що вихідний (рос. – исходный) файл ***повинен мати теж саме ім'я, що й назва модулю*** з розширенням **PAS** (**MYUNIT.PAS**). Після компіляції у вказаний Вами каталог, цей файл буде мати теж саме ім'я, але розширення **TPU** – Turbo Pascal Unit (модуль Турбо Паскалю).

Також треба повідомити компілятор ТП, з якого каталогу треба вибирати відкомпільований модуль. Для бібліотечних

модулів цей маршрут відомий – каталог **C:\BP\Unit**, куди при інсталяції Турбо Паскалю встановлюються всі бібліотечні модулі. Ваші особисті відкомпільовані модулі краще зберігати у окремому каталозі, який треба указати через крапку з комою у рядку **Unit Directories** меню **Directories** (рис. 8.67).

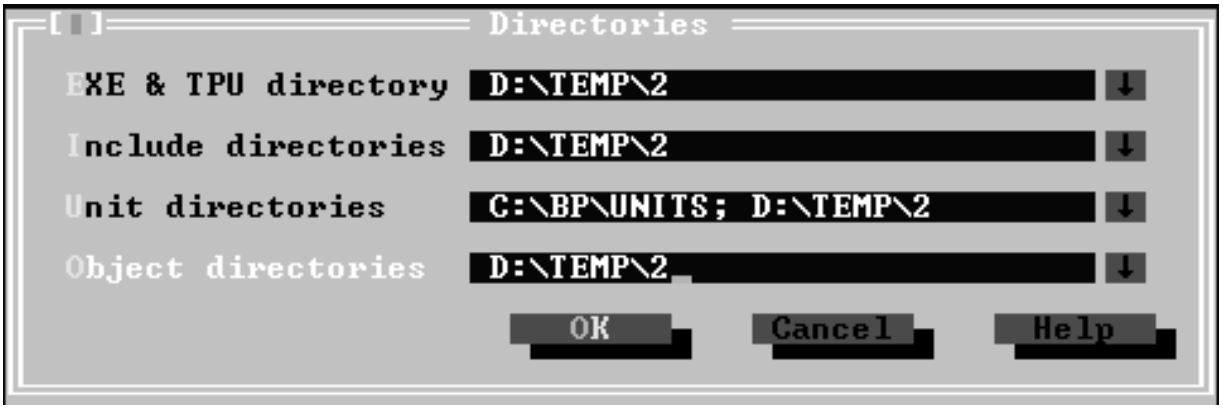


Рис. 8.67. Меню опції **Directories**

Сформулюємо етапи Ваших дій по створенню особистого ***модулю*** (**Unit**):

1. Наберіть текст Вашого модуля (**Unit MyUnit;**) та збережіть його у належний каталог (**D:\TEMP\2)** ***під назвою, яка співпадає з ім'ям модуля***, й з розширенням .**PAS** (**MYUNIT.PAS**).

2. Натисканням клавіш **Alt+C** відкрийте меню опції **Compile,** виберіть опцію **Compile → Destination** й натисніть клавішу **Enter**, для компіляції на диск (рис. 8.64).

3. Натисканням клавіш **Alt+O** відкрийте меню опції **Options**, а в ній виберіть команду **Directories** (рис. 8.66). При цьому, з'явиться меню **Directories** (рис. 8.56). У рядку **EXE & TPU Directories** укажіть, куди Ви бажаєте зберегти відкомпільований модуль.

4. У рядку **Unit Directories** меню **Directories** (рис. 8.67) укажіть через кому з крапкою свій каталог, де розташований Ваш модуль **(; D:\TEMP\2).**

5. Підключіть свій модуль до програми у стрічці

```
Uses Crt, MyUnit; {Тільки після бібліотечних модулів ТП!}
```

Тепер настав час зайнятися самою програмою. Гадаємо, що для Вас це не буде дуже складною справою (рис. 8.68).

```
Program MyUnitProg;
Uses
  Crt, MyUnit; {Особистий модуль описуємо після бібліотечного ТП}
Var
  X1, Y1 : VectorType; {Тип VectorType використовуємо з модуля
                        MyUnit}
  m      : Integer;
```

Рис. 8.68. Програма, яка використовує модуль `MyUnit`

```
Begin
  ClrScr;
  WriteLn('Кут між векторами X i Y',
          ' з трьома компонентами = ',
          CosFi (XX, YY, 3)); { Використовуємо ініціалізовані}
                              { у модулі MyUnit масиви XX і YY,}
                              { а також функцію CosFi}

  WriteLn('Vvedite 4 komponenty massiva X1:');
  VectInput(X1, 4); { Використовуємо процедуру VectInput,}
                    { яка описана у модулі MyUnit}

  WriteLn('Vvedite 4 komponenty massiva Y1:');
  VectInput(Y1, 4);

  WriteLn('Кут між векторами X1 i Y1',
          's chetyrmya comp. = ',
          CosFi (X1, Y1, 4)); { Використовуємо масиви X1 і Y1,}
                              { які описані у програмі}
  ReadLn
End.
```

Рис. 8.68. Програма, яка використовує модуль `MyUnit` (продовження)

Обов'язково виконайте самостійно цю програму з модулем `MyUnit`, який потрібно підготувати заздалегідь.

Вправи

1. Розробіть програмний модуль ТП, який повинен включати:
   – процедуру вводу вектора з *n* елементів;
   – процедуру обчислення вектору, який дорівнює сумі двох векторів;
   – функцію, яка знаходить найбільший компонент вектора;
   – процедуру виводу векторів на екран ПК.
2. Розробіть програмний модуль ТП, який повинен включати:
   – процедуру вводу вектора з *n* елементів;
   – процедуру обчислення вектору, який дорівнює добутку вектора на скаляр;
   – функцію, яка знаходить найменший компонент вектора;
   – процедуру виводу векторів на екран ПК.
3. Розробіть програмний модуль ТП, який повинен включати операції з комплексними числами, та програму, яка його використовує.

8.25. Обробка символів та строк

У сімействі персональних комп'ютерів IBM PC використовуються 256 різноманітних символи. Вони мають свої числові коди, значення котрих лежать у діапазоні від 0_{10} до 255_{10}, тобто загальна кількість символів дорівнює 256-ти.

Коли Ви натискаєте клавішу на клавіатурі, то це приводить до того, що у комп'ютер надсилається сигнал у виді двоїчного числа, яке ставиться у відповідність до ***кодової таблиці***, тобто внутрішнього представлення символів у комп'ютері. В усьому світі як стандарт прийнята ***таблиця ASCII*** *(American Standard Code for Information Interchange –* ***Американський стандартний код обміну інформацією)***. Вона указує на відповідність між зображеннями або умовними позначеннями символів та їх внутрішніми числовими кодами.

Для збереження двоїчного коду одного символу у цьому стандарті виділений 1 байт, який дорівнює 8 біт. З огляду на те, що кожен біт приймає значення "0" чи "1", кількість їхніх можливих сполучень у байті і дорівнює 256.

У комп'ютерах IBM PC існують особливі комбінації клавіш (табл. 8.47).

Таблиця 8.47

Особливі комбінації клавіш

Комбінації клавіш	Дії, які виробляються
Ctrl+Alt+Del	Перезапуск комп'ютера
Ctrl+NumLock	Припинення роботи програми
Ctrl+Break	Переривання (закінчення) роботи програми
Shift+PrtSc	Копіювання зображення з екрану на принтер

Кодова таблиця припускає представлення наступних груп символів:

❶ управляючі символи;
❷ знаки арифметичних операцій, знаки перетинання та цифри;
❸ букви латинського алфавіту;
❹ букви національних алфавітів;
❺ символи псевдографіки;
❻ математичні символи.

Управляючі символи використовуються для спеціальних цілей управління друкуючими пристроями, як маркери й обмежники (рос. – ограничители) при запису, читанні та передачі інформації. Найбільш цікаві з них у рамках даної книги – це Line Feed (LR – код 10) Carriage Return (CR – код 13). Вони зустрінуться нам при роботі з текстовими файлами у гл. ***.

Мова Турбо Паскаль підтримує стандартний символьний тип Char розміром 1 байт. Його значення Ви можете задавати як символ, який уміщується у лапках:

`'A', 'a', '1'` і так далі.

Зверніть увагу на те, що ‘1’ – це символ одиниці з кодом 49_{10}. Також цей символ (та інші ***ASCII-символи)*** можна представити за допомогою його ***ASCII-коду*** та спеціального префікса # (“шарп”)(див. Додаток 6):

```
   #49  ⇨ Char(49) ⇨  ‘1’ {Рівноцінне представлення ‘1’}
Write( #49 ); ⇨ Write( Char(49) ); ⇨ Write( ‘1’ ); {Вивід на
                                                екран символу ‘1’}
```

Для роботи з ***ASCII-символами*** використовуються корисні функції (табл. 8.48).

Таблиця 8.48

Функції роботи з ***ASCII-символами***

Функція : Тип	Призначення
Chr(X : Byte) : Char	Повертає символ ***ASCII-коду*** X
Ord(C : Char) : Byte	Повертає ***ASCII-код*** символу C

Тобто:

```
Ord(Char(49)) ⇨ 49.
```

Скільки б Ви не читали про символи, краще за все самостійно засвідчитись, як виглядає таблиця ***ASCII-символів*** на Вашому комп’ютері. Для цього напишемо й запустимо на виконання наступну програму (рис. 8.69).

```
Program ASCIIChar;
Uses Crt;
Var
  i : Integer; {Змінна циклу}
  StartCode, EndCode, {Початковий та кінцевий коди символів}
  XStart, YStart : Byte; {Позиції виводу на екран}

Procedure ShowChar(FirstCharCode, {Початковий код символу}
                   LastCharCode, {Кінцевий код символів}
                   X, Y : Byte); {Позиції виводу на екран}
Begin
  while FirstCharCode<=LastCharCode do
    begin
       GotoXY(X,Y); {Установимо курсор у позицію X,Y}
       Write((FirstCharCode):4); {Виведемо код символу}
       GotoXY(X,Y+1);{Установимо курсор у позицію X,Y+1}
       Write('  ', Chr(FirstCharCode), ' '); {Виведемо
                                                символ}
       Inc(FirstCharCode); {Збільшимо код символа на 1}
       Inc(X,4); {Збільшимо координату стовпця на 4}
       Delay(3000) {Затримаємо виконання програми на 3 сек.}
     end;
End; {Proc ShowChar}
```

Рис. 8.69. Вивід ***ASCII-символів*** на Вашому комп’ютері

```
Begin {Основна програма}
  ClrScr; {Очистимо екран}
  StartCode:=33; {Пропускаємо управляючі символи}
  EndCode:=52; {Виводимо по 19 символів}
  XStart:=1; {Починаємо з першої позиції екрану X}
  YStart:=1; {Починаємо з першої позиції екрану Y}
  for i:=1 to 10 do
    begin
      ShowChar(StartCode,EndCode,XStart,YStart);
      Inc(StartCode,20); {Переходимо до наступної порції
                          символів}
      Inc(EndCode,20); {Переходимо до наступної порції символів}
      Inc(YStart,2); {Зрушуємо курсор на два рядки нижче}
    end;
  ReadLn {Затримуємо екран до натиснення клавіши Enter}
End.
```

Рис. 8.69. Вивід *ASCII-символів* на Вашому комп'ютері (продовження)

Після цього на екрані ПК можна побачити наступну картину (рис. 8.70):

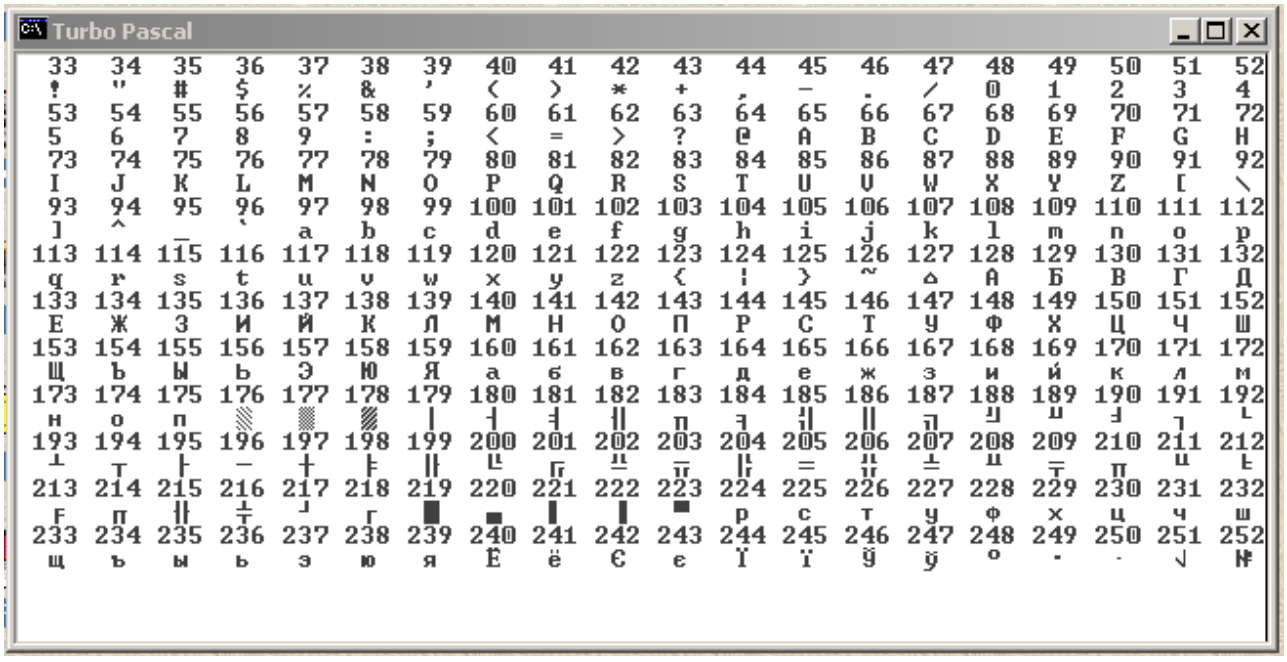


Рис. 8.70. Вивід на екран *ASCII-символів* та їх кодів

На екрані Ви можете побачити символи російського та українського алфавіту. Ці символи заносяться у кодову таблицю спеціальними програмами, які називаються драйверами. Найбільш поширена з них програма KeyRus.com, яку розробив Д. Гуртяк з м. Донецьк, Україна.

Повернемося до програми (рис. 8.69). При її розробці ми використовували ряд функцій Турбо Паскалю, про які потрібно навести подробиці, щодо їх використання (табл. 8.49).

Таблиця 8.49

Стандартні функції та процедури ТП

Функція : Тип	Призначення
`Inc(Var X:Integer)`	Збільшує значення `X` на 1
`Dec(Var X:Integer)`	Зменшує значення `X` на 1
`Inc(Var X:Integer; N:Integer)`	Збільшує значення `X` на `N`
`Dec(Var X:Integer; N:Integer)`	Зменшує значення `X` на `N`
`GotoXY(X,Y : Byte)`	Встановлює курсор у стовпець `X`, рядок `Y` (модуль `Crt`)
`Delay(ms : Word)`	Затримує процес (пауза) на `ms` мілісекунд (модуль `Crt`)

Зверніть увагу на те, що після того, як процедура `GotoXY` встановлює курсор у стовпець `X`, рядок `Y`, наступна операція виводу тексту на дисплей розмістить перший символ рядку, який виводиться, у позиції (`X`, `Y`).

Процедура `GotoXY` використовує систему координат поточного текстового вікна, тобто координати (1,1) відповідають лівому верхньому куту вікна (рис. 8.71).

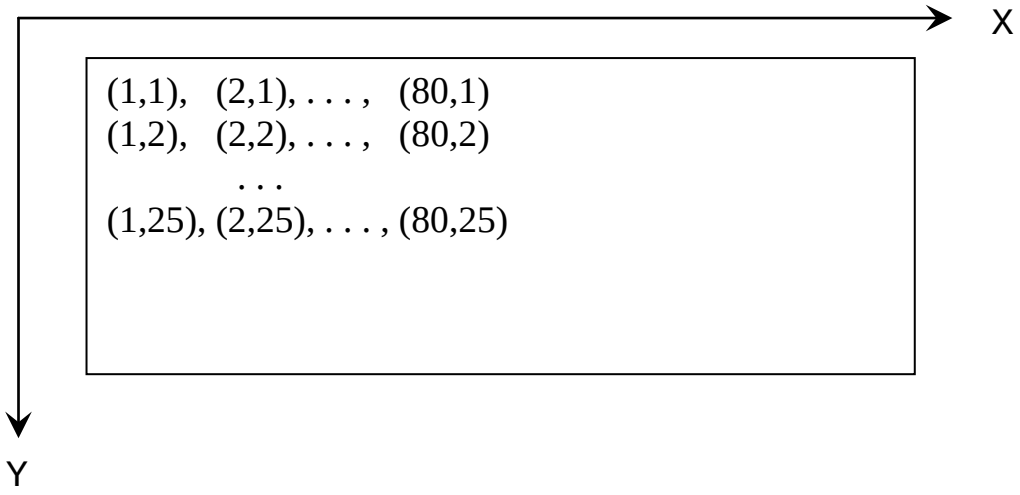


Рис. 8.71. Розподіл координат вікна при використанні процедури `GotoXY`

Якщо аргументи процедури `GotoXY` виявляться зовні поточного вікна, то її виклик не буде мати ніякої дії. Нижні дозволені значення для `X` та завжди дорівнюють 1, а верхні – за умовчанням – відповідно `X=80` та `Y=25.`

З точки зору малювання різноманітних прямокутних рамок за допомогою одинарних та подвійних ліній, найбільш корисними символами є символи з кодами 179-218 (рис. 8.72). Побачити кожен з цих символів можна у такий спосіб:

❶ натисніть клавішу Alt;

❷ наберіть відповідний код символу псевдографіки на додатковій (сірій) цифровій клавіатурі;

❸ відпустіть клавішу Alt, тоді символ псевдографіки з'явиться на екрані.

Оскільки комбінування символів псевдографіки між собою відносно складна справа, то Вам буде корисною інформація, яка наведена на рис. 8.61.

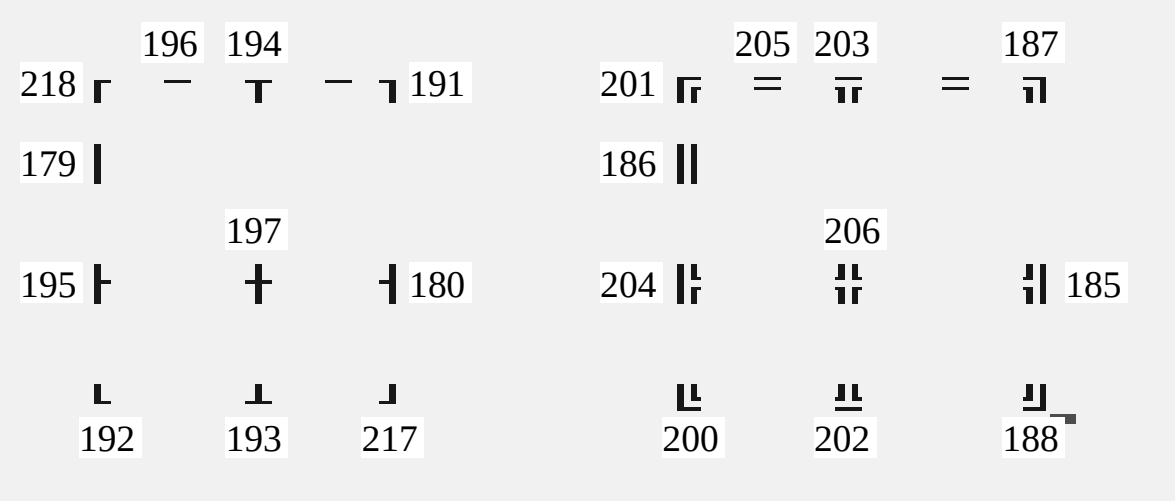


Рис. 8.72. Символи для малювання рамок

Цей малюнок має збудити Вашу уяву, щодо конструювання різноманітних рамок. Окрім того, рекомендуємо Вам уважно розглянути побудову інтерфейсу самого Турбо Паскалю з точки зору використання подібних рамок...

8.25.1. Операції над символами

```
Program Comparison;
Var
  Char1, Char2 : Char;
Begin
  Char1:='A'; {Код символу 'A' дорівнює 65 (рис. 24.2)}
  Char2:='a'; {Код символу 'a' дорівнює 117 (рис. 24.2)}
  if Char1 < Char2 then {Порівнюємо символи}
    WriteLn('First Symbol Less than Second');
  ReadLn
End.
```

Рис. 8.73. Порівняння двох символів

Символи можна тільки присвоювати і порівнювати одне з одним. При цьому вони вважаються рівними, якщо рівні їх ***ASCII-коди.*** Один символ вважається більше іншого, якщо має більший ***ASCII-код*** (рис. 8.73).

8.25.2. Опитування клавіатури

Щоб організувати опитування клавіатури нам знадобляться дві процедури. Перши з них "очищує" буфер клавіатури (рис.8.74).

```
{Файл ClrKey.inc}
Procedure ClrKeyBuf;
Var
  ch : Char;

Begin
  while KeyPressed do
    ch:=ReadKey
End; {Proc ClrKeyBuf}
```

Рис. 8.74. Процедура "очищення" буферу клавіатури

У цій процедурі функція `KeyPressed` повертає логічне значення `True`, якщо у буфері вводу з клавіатури знаходиться хоча б один символ, і `False`, якщо буфер пустий.

Коли програма стартує, буфер звичайно пустий. Але будь-яке натискання символьної клавіші (окрім клавіш регістрів **Ctrl, Shift, Alt** і перемикачів типу **NumLock, CapsLock** і т.д.) занесе її код у буфер. Коди у буфері будуть зберігатися до тих пір, доки вони не будуть зчитані, або буфер не буде очищений самою програмою.

Функція `ReadKey` начебто "виймає" послідовно уведені у нього символи по одному за кожне звертання (рос. – обращение). Ця функція завжди повертає один символ, тобто одне значення типу `Char`. Але у неї є дві важливі особливості:

❶ символи, які отримані, ніколи не відображаються на дисплеї, тобто ввід символу завжди відбувається "наосліп" (рос. – вслепую);

❷ режим роботи `ReadKey` залежить від стану буферу вводу: містить він символи або пуст. Якщо у буфері щось є, то `ReadKey` поверне перший символ у буфері (той, що був введений раніш за усіх) й видалить цей символ з буферу. Але якщо буфер пустий, то функція `ReadKey` пристановляє роботу програми й очікує, поки не буде натиснута яка-небудь клавіша, яка генерує символьний код.

Ці її особливості можна використовувати для побудови декілька корисних конструкцій (змінна `ch` повинна бути типу `Char`):

```
while KeyPressed do
    ch:=ReadKey; {Очистка буферу вводу}
repeat until
  KeyPressed; {Очікування натискання будь-якої клавіші}
```

Останній цикл завершиться, коли у буфер попаде будь-який символ. Запам'ятайте, що Ваша програма повинна у кінці завжди очищати буфер, інакше усе, що накопичилось у буфері, буде виведено у строку **MS-DOS** або у редактор середовища програмування.

З урахуванням усього вищенаведеного, представимо другу процедуру, яка очікує натискання клавіші (рис. 8.75).

```
{Файл Wait.inc}
Procedure Wait;
Begin
  repeat until
    KeyPressed
End; {Proc Wait}
```

Рис. 8.75. Процедура, яка очікує натискання клавіші

Після цього можна визначити програму, яка приймає символ натиснутої клавіші (рис. 8.76).

```
Program UntilKeyPressed;
Uses Crt;
     {$I clrkey.inc} {Підключаємо файл clrkey.inc}
     {$I wait.inc}   {Підключаємо файл wait.inc}
Var
  c : Char;
Begin
  ClrScr;
  WriteLn('Натисніть будь-яку символьну клавішу');
  ClrKeyBuf;          {Очищаємо буфер клавіатури}
  c:=ReadKey;         {Чекаємо натиснення клавіші}
  WriteLn('Була натиснута клавіша з символом ', c);
  WriteLn;
  WriteLn('Програма тримає паузу до першого ',
          'натискання будь-якої клавіші... ');
  Wait;               {Чекаємо натиснення клавіші}
  ClrKeyBuf           {Убираємо "сміття" з буферу}
End.
```

Рис. 8.76. Програма, яка приймає символ натиснутої клавіші

8.25.3. Строкові типи (String)

Строковий тип даних визначає множину символьних ланцюжків довільної довжини (від 1 до 255 символів). Його можна уявити себе як масив символів:

```
CharArray : Array[1..255] of Char;
```

Для визначення строкового типу використовується службове слово String, або це ж слово, поза котрим у квадратних скобках вказується максимальна довжина строки, яка повинна не перевищувати 255:

```
Type
   Line = String; {Строка довжиною 255 символів}
   Line30 = String[30]; {Строка довжиною 30 символів}
Var
   MyLine : Line;
   MyLine30 : Line30;
. . .
```

Для описаної строкової змінної довжиною N символів у ТП відводиться N+1 байтів пам'яті, з котрих перший байт за номером "0" містить значення поточної довжини цієї строки, а наступні N байтів призначені для зберігання символів строки, наприклад, 'I know Turbo Pascal' (рис. 8.77). Перевірте цей приклад, маючи на увазі, що код символу "пробіл" дорівнює 32. Будь ласка, перевірте наш приклад за допомогою рис. 8.76, щоб посвідчитись, що все гаразд.

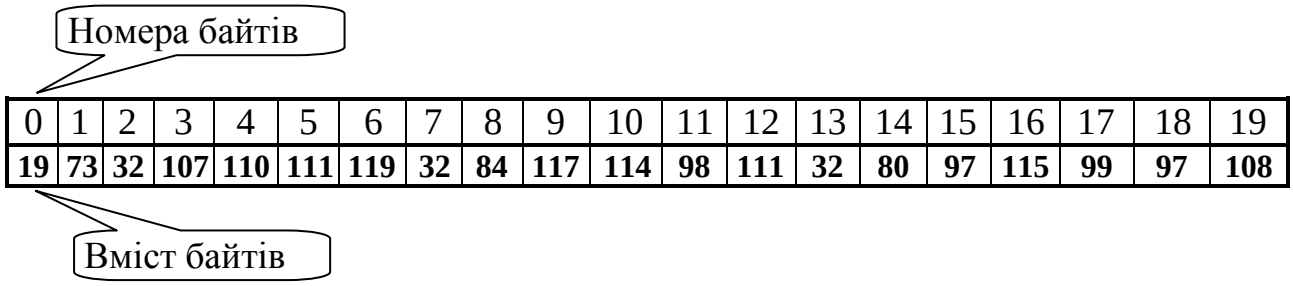


Рис. 8.77. Зберігання символі строки у змінній типу `String`

При такому способі зберігання у змінній типу `String` , виникає одна тонкість, яку Ви повинні собі дуже ясно уявляти (рис. 8.78).

```
Program StrigNumber;
Uses Crt;
Var
  St : String;
  Ch : Char;
  By : Byte;
```

Рис. 8.78. Перевіримо, що у нульовому байті...

```
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. Перевіримо, що у нульовому байті... (продовження)

При запуску програми рис.8.78 отримуємо наступний результат (рис. 8.79):

Рис. 8.79. Вміст нульового байту типу `String`

Ви маєте зробити висновок, що кількість байтів у типі `String` трактується, як ***код символу!*** З цього можна зробити висновок, як треба звертатися до цієї величини. Ще один висновок може бути таким: ТП буде бачити стільки символів строки, яку кількість Ви занесете у нульовий байт.

Таким чином, елементи строки нумеруються цілими числами від 1 до 255. Доступ до кожного елементу строки: S[i]:

```
MyLine := 'ABCD' {MyLine : String}
WriteLn (Ord(MyLine[0])); {Буде надруковано "4"}
MyLine[0] := #2;
WriteLn(MyLine); {Буде надруковано "AB"}.
```

Строки можна присвоювати, зливати (конкатенація), порівнювати. А також вводити операціями Read, ReadLn:

```
Var
  S1, S2, S3 : String;
Begin
  S1 := 'Вам ';
  S2 := 'привіт';
  S3 := S1 + S2; {S3 = 'Вам привіт'}
  S3 := S3 + '!'; {S3 = 'Вам привіт!'}
End.
```

Якщо довжина строки в правій частині перебільшує ліву, то надлишок відтинається.

Порівняння строк виконується за правилами:

1. Більш коротка строка завжди менше ніж більш довга;

2. При рівних довжинах відбувається заелементне порівняння символів цих строк с урахуванням лексикографічної упорядкованості значень стандартного символьного типу Char:

```
'abcd' = 'abcd'   ⇨ True
'abcd' <> 'abcde' ⇨ True
'abcd' <> 'abcd'  ⇨ True
'abcd' > 'abcD'   ⇨ True   {'d' < 'D'}
'abcd' > 'abc'    ⇨ True
'aBcd' < 'ab'     ⇨ True   {'B' < 'b'}
```

Строки можна вводити з клавіатури. При цьому Ви повинні відслідковувати відповідність введених символів довжині описаних строк (рис. 8.80).

```
Program StringInput;
Var
  S1 : String[5]; {Довжина строки S1 – п'ять символів}
Begin
  ReadLn(S1);     {Вводимо 6 символів: 'Привіт'}
  WriteLn(S1);    {Надлишок відтинається. Виводиться: 'Приві'}
  ReadLn
End.
```

Рис. 8.80. Відповідність введених символів довжині строки

Для роботи зі строками міститься велика кількість процедур та функцій, які наведені у таблиці 8.50.

Таблиця 8.50.

Процедури та функції роботи зі строками

Процедури та функції	Призначення
Редагування строк	
`Length(S: String) : Byte`	Видає поточну довжину строки
`Concat(S1, S2, . . . ,Sn ): String`	Повертає конкатенацію тобто злиття строк S1..Sn
`Copy (S: String; Start, Len: Integer) : String`	Повертає підстроку довжини `Len`, яка починається з позиції `Start` строки `S`
`Delete (Var S: String; Start, Len: Integer)`	Видаляє з `S` підстроку довжини `Len`, яка починається з позиції `Start` строки `S`
`Insert (Var S: String; SubS: String; Start: Integer)`	Вставляє в `S` підстроку `SubS`, починаючи з позиції `Start`

Продовження таблиці 8.50

`Pos (SubS, S: String;): Byte`	Шукає входження підстроки SubS у строці S й повертає номер першого символу SubS в S або 0, якщо S не містить SubS
Процедури перетворення	
`Str (x[:F[:n]]; Var S: String)`	Перетворює числове значення X у строкове S. Можливо задавати формат для X
`Val (S: String; Var x; Var ErrCode: Integer)`	Перетворює строку S цифр у значення числової змінної X

Вправи

1. Які особливі комбінації клавіш існують у комп'ютерах IBM PC?

2. Які корисні функції для роботи з ***ASCII-символами*** існують у Турбо Паскалі?

3. За допомогою програми рис. 8.58 виведіть на екран комп'ютера символи, ***ASCII-коди*** яких перевищують 240.

4. Створіть на екрані комп'ютера рамки з одинарною та подвійною обвідкою відповідно до рис. 8.61.

5. З використанням програми рис. 8.65 розробіть свою програму, котра опитує клавіатуру й очікує натискання декількох клавіш, відповідно до яких робить конкретні дії.

6. Як розміщується інформація у строкових змінних?

7. Розробіть програму, яка зчитує строкову інформацію з клавіатури та додає її до строкової змінної.

8.26. Рекурсія, множини та текстові файли

8.26.1. Рекурсія

Рекурсія – це традиційна перевага мови Паскаль над іншими мовами програмування. Турбо Паскаль повною мірою дозволяє будувати рекурсивні алгоритми. Під рекурсією розуміється виклик функції (процедури) з тіла цієї ж самої функції (процедури).

Рекурсивність часто використовується в математиці, оскільки багато визначень математичних формул рекурсивні. Як приклад можна привести формулу обчислення факторіала:

$$n! = \begin{cases} 1, & якщо\ n = 0; \\ n! = n * (n-1)!, & якщо\ n > 0. \end{cases}$$

а також – цілого ступеня числа:

$$x^n = \begin{cases} 1, & якщо\ n = 0; \\ x * x^{n-1}, & якщо\ n > 0. \end{cases}$$

У приведених формулах для обчислення кожного наступного значення потрібно знати попереднє. У Турбо Паскалі рекурсія записується так само, як і у формулах:

```
Function Fact(n : Word) : LongInt;
Begin
  If n=0 Then
    Fact:=1
  Else
    Fact := n*Fact(n -1)
End;
```

А також – ступеня числа *x*:

```
Function IntPower(x : Real; n : Word) : Real;
Begin
  If n=0 Then
    IntPower:=1
  Else
    IntPower :=x*IntPower(x, n-1)
End;
```

Якщо у функцію передаються *n>0*, то відбувається наступне: запам'ятовуються відомі значення членів виразу у гілці `Else` (для факторіала це *n*, для ступеня – *x)*, а для обчислення невідомих викликаються ті ж функції, але з «попередніми» аргументами. При цьому знову запам'ятовуються (але в

іншому місці пам'яті – стеці!) відомі значення членів і відбуваються чергові виклики. Так відбувається доти, поки вираження не стане цілком визначеним (у наших прикладах – це присвоювання гілки `Then`), після чого алгоритм починає «розкручуватися» у зворотну сторону, вилучаючи з пам'яті «відкладені» обчислені заздалегідь значення. Оскільки при цьому на кожному черговому кроці всі члени виразів вже будуть відомі, через *n* таких «зворотних» кроків ми одержимо кінцевий результат.

Незважаючи на наочність рекурсії, у багатьох випадках ті ж задачі більш ефективно вирішуються ітераційними методами (при порівнянній швидкості обчислень, але з економією пам'яті). Однак у граматичному розборі символьних конструкцій рекурсія доречна й ефективна.

8.26.2. Множини

Тип “множина” (рос.-множество) задається або перерахуванням значень (перелічимий тип), або відрізком типу, або ім'ям скалярного типу. Наприклад:

```
Type
  Letter  = Set Of ‘A’..’Z’;
  All     = Set Of 11..88;
  SubColr = Set Of (Red, Green);
  AllLet  = Set Of Char;
```

При визначенні множин необхідно враховувати наступні обмеження:

– базисний тип – повинен бути будь-яким скалярним типом (пам’ятайте, що тип `Real` не є скалярним!);

– максимальне число елементів – не більш 256-ти;

– значення елементів з базисного типу повинне належати множині 0..255.

Множини можуть обчислюватися за посередництвом виразів над множинами. Ці вирази складаються з констант, змінних і операцій.

Константи з множин – це список підмножин, що складають множину:

```
['A'..'Z', 'a'..'z', '0'..'9']
```

Змінні типу множина при використанні підкоряються спеціальному синтаксису – укладаються у квадратні дужки:

```
Sbyte := [1, 2, 3, 4, 10, 20, 30];
Schar := [‘а’, ‘б', ‘в'];
Sdiap := [1..4]; {то ж, что і [1, 2, 3, 4]}
Schar := [‘а..п',’р..9’];
Empty := []; {Так записується порожня множина}
```

8.26.3. Операції застосовні до множин

Над множинами визначені наступні операції:

`S1 = S2` {Рівність множин. Дорівнює `True` якщо `S1` і `S2` складаються з тих самих значень незалежно від порядку}

`S1 <> S2` {Нерівність множин. Дорівнює `True` якщо хоч один елемент відрізняється}

`S1 <= S2` {Входження `S1` у множину `S2`. Перевірка на підмножину. Дорівнює `True`, якщо всі елементи `S1` містяться в `S2`, незалежно від порядку проходження}

`S1 >= S2` {Включення `S2` у множину `S1`. Перевірка на надмножину. Дорівнює `True`, якщо всі елементи `S2` містяться в `S1`, незалежно від порядку проходження}

`E in S1` або `E in [..]` {Перевірка входження (приналежності) елемента `E` у множину `S1` або приналежність базовому типу `[..]`}

`S1 + S2` {Об'єднання множин. Нова множина містить всі елементи з `S1`, `S2` крім тих, що дублюються}

`S1 – S2` {Різниця множин. Нова множина містить елементи `S1` без елементів `S2`}

`S1 * S2` {Перетинання множин. Нова множина містить тільки елементи, що містяться одночасно в `S1` і `S2`}

Наприклад:

```
[1, 2, 3] = [1, 3, 2]        {Дасть True}
[1, 2] <> [1], 5 in [0..5]   {Дасть True}
[1, 2] + [3, 4]              {Дасть [1, 2, 3, 4]}
[1, 2, 3, 4] – [3, 4]        {Дасть [1, 2]}
[1, 2, 3, 4] * [3, 4, 5, 6]  {Дасть [3, 4]}
```

Наведемо приклад програми, що використовує множини при перевірці натискання клавіш клавіатури (рис. 8.81).

8.26.4. Ввід-вивід даних і файлова система MS-DOS

Будь-який обмін даними припускає наявність: ***джерела інформації, каналу зв'язку*** та її ***приймача*** (рис. 8.82).

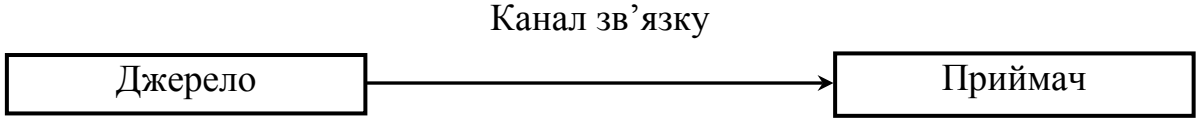


Рис. 8.82. Загальний вид процессу обміну даними

У випадку обміну даними між програмою і периферійними пристроями ПК, одним кінцем каналу обміну даними завжди є оперативна пам'ять ПК (ОЗП). Інший кінець цього каналу у Турбо Паскалі визначений як ***файл*** (рис. 8.83).

```
Program ChekSymbol;
Uses Crt;
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; {Об'єднання множин}
  Write('Продовжити?(Д/Н)');
  Repeat
    Ch:=ReadKey; {Функція ReadKey повертає код нажатого
                  символу}
  Until Ch In YesNo; {Перевірка на належність Ch
                      множині припустимих значень
                      відповідей користувача}
  If Ch In No Then
    Halt;
  WriteLn(Ch); {Виводимо символ, який введений
користувачем}
  WriteLn('Для закінчення натисніть будь-яку клавішу ');
  Repeat Until KeyPressed;
End.
```

Рис. 8.82. Використання множин при опитуванні клавіатури

Всі операції обміну підтримуються операційною системою, яка у системній області виділяє спеціальний буфер для операцій обміну. Це робиться для зменшення кількості звертань до зовнішніх пристроїв. Розмір буфера дорівнює 128 байт, але може змінюватися користувачем убік збільшення. При запису у файл вся інформація спочатку направляється в буфер і там накопичується доти, поки весь обсяг буфера не буде заповнений. Тільки після цього або після спеціальної команди скидання буфера відбувається передача даних.

Аналогічно при читанні з файлу зчитується не стільки, скільки запитується, а скільки уміститься в буфер. ***Наприклад, якщо з файлу на диску зчитується 4 числа з загального числа 64, то наступні 60 читаються вже з буферу (!)***.

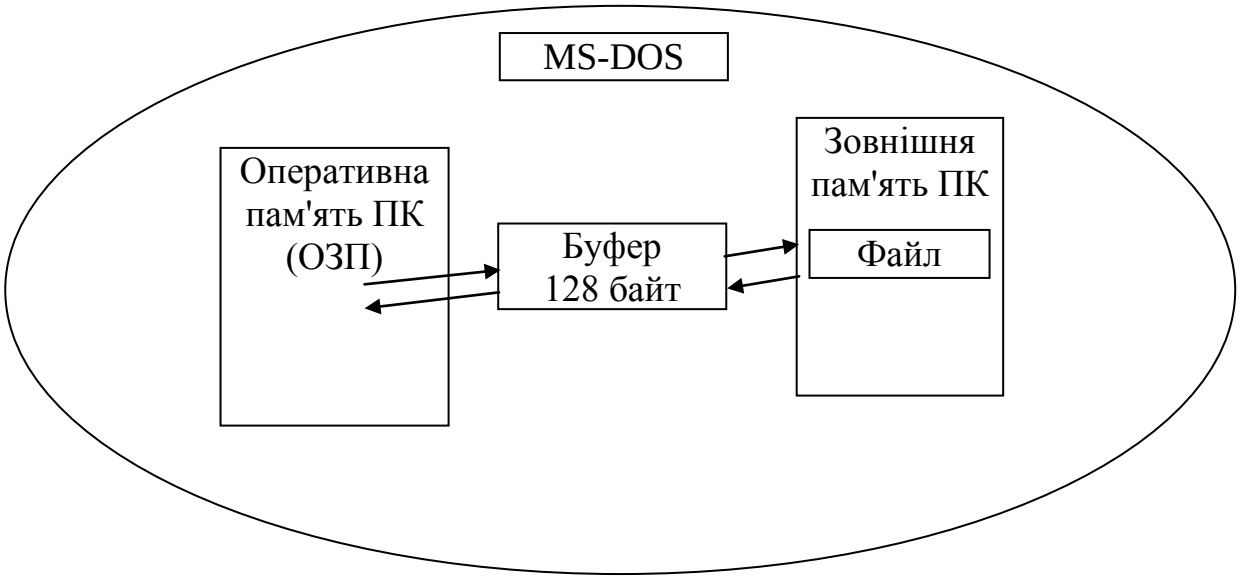


Рис. 8.83. Канали вводу-виводу в ТП

Операція виводу даних означає пересилання даних з робочої пам'яті (ОЗП) у файл, а операція введення – заповнення ячеєк пам'яті даними, отриманими з файлу.

Файлова система в ТП складається з двох рівнів: ***логічних файлів*** і ***фізичних файлів***.

8.26.5. Поняття логічного файлу

Логічний файл описується в програмі ТП як змінна файлового типу. Після цього вона зв'язується з ***фізичним файлом*** MS-DOS і може використовуватися для операцій уведення-висновку. Так, якщо ми хочемо працювати з текстовим файлом `'A:\TEXT.DAT'` , то в програмі повинні бути поміщені такі рядки:

```
Var
  f : Text; {Оголошуємо логічну файлову змінну f для роботи з
              текстовим файлом}
Begin
  Assign( f, 'A:\TEXT.DAT'); {Пов'язуємо фізичний файл
                              'A:\TEXT.DAT' на диску A: з
                              логічним файлом f}
. . .
End.
```

Після цього звертання до файлу на диску будуть робитися через файлову змінну `f`. Введення ***логічного файлу*** дозволяє звільнити програміста від рішення технічних проблем організації обміну даними у програмі.

8.26.6. Фізичні файли в MS-DOS

В операційній системі MS-DOS містяться два види ***фізичних файлів***:
– файли на магнітних носіях (флоппі-диск, вінчестер, віртуальний диск);
– зовнішні пристрої MS-DOS (файлові моделі цих пристроїв).

У ТП імена файлів першого типу можуть бути строковими константами або зберігатися у строкових змінних. Наприклад:

```
'C:\PAS\TESTFILE.PAS'
'A:\TEST.TXT'
'..\PRIMER.DAT'
```

Імена пристроїв MS-DOS мають фіксовані імена і використовуються як звичайні файли (табл.. 51).

Фізичні файли-пристрої в MS-DOS організовуються як ***текстові файли*** і для їх нормальної роботи треба зв'язувати їх імена з текстовими логічними файлами. Імена фізичних файлів-пристроїв записуються також як у простих текстових файлів: `'CON', 'PRN'` і т.д.

Таблиця 8.51

Стандартні імена файлів-пристроїв MS-DOS

Ім'я	Файл-пристрій	Примітка
`CON`	Консоль (клавіатура й екран)	Ввід з `CON` – це читання з клавіатури, а вивід у `CON` – це запис на екран

`LPT1`
`LPT2`
`LPT3`
	Паралельні порти виводу	Через ці імена файлів відбувається вивід даних на принтер або інші пристрої

`PRN`	Принтер. Синонім імені `LPT1`	Ім'я для звертання до принтера, підключеного до порту `LPT1`
`COM1` `COM2`	Послідовні порти	Імена файлів-пристроїв для вводу-виводу даних через серійні порти комунікації
`NUL`	Фіктивний пристрій	Це бездонний файл, що приймає що завгодно, але завжди порожній

Не визначена така структура даних, як файл у пам'яті ПК! Будь-який оголошений ***логічний файл*** має сенс тільки після позв'язування з зовнішнім ***фізичним файлом*** або ***файлом-пристроєм***.

8.26.7. Файлові типи ТП

Турбо Паскаль підтримує три файлових типи (тобто способів організації зберігання різних типів даних на диску):

– ***текстові файли*** (типу `Text`);

– ***компонентні файли*** (типу `File Of . . . [наприклад, Real]`);

– ***безтипові файли*** (типу `File`).

Текстові файли – це файли, що складаються з ASCII-кодів, включаючи розширені і управляючі коди. Текстові файли організуються по рядках (рос.-строках). Кінцем рядка є спеціальний символ **EOL** (**End Of Line** – кінець рядка), що складається з двох ASCII-кодів CR = #13 **(Carriage Return** – повернення каретки) і LF = #10 **(Line Feed** – переведення рядка), які ***не друкуються на екрані при вводі!***. Будь-яку інформацію (числову, символьну чи строкову) текстовий файл зберігає у виді послідовностей текстових (тип `Char` ТП) символів, що її зображують.

Компонентні файли на відміну від текстових складаються з машинних представлень чисел і побудованих з цих уявлень символів і інших структур даних. Такі файли зберігають усі дані що записуються в такому ж вигляді, як і пам'ять ПК. Тому за допомогою компонентних файлів можна здійснювати обмін даними тільки між дисками і робочою пам'яттю програми, але не можна, наприклад, прямо вивести з диску дані на безпосередньо на екран.

Безтипові файли теж складаються з машинних представлень даних. Їхня відмінність від компонентних файлів полягає в тому, що вони містять довільні набори байтів. Усі без винятку файли обов'язково містять у своєму кінці спеціальний код **End Of File** – **EOF**, називаний кінцем файлу.

Для всіх типів файлів мінімальною одиницею збереження інформації в них є байт. Принципи роботи з усіма типами файлів єдині, хоча і мають деякі відмінності.

8.26.8. Текстові файли

Текстовий файл можна розглядати як послідовність символів, розбиту на рядки спеціальним маркером. На практиці такий маркер уявляє собою послідовність із двох символів ASCII: повернення каретки (рос.-возврат каретки) `chr(код 13)` (**CR** – Carriage Return) і переклад рядка (рос.-перевод строки) `chr(код 10)` (**LF** – Line Feed) (див. додаток 6). Ці два символи задають стандартні дії по управлінню текстовими файлами. Зокрема, редактор ТП виводить у своєму вікні рядки тексту на основі аналізу появи пари кодів (**CR**/**LF**). Тобто, наприкінці кожного рядка завжди присутня пара цих управляючих символів. Вони вставляються в текстовому редакторі ТП наприкінці кожної рядка, ***ввід якого закінчується натисканням клавіші*** **Enter**. При натисканні наприкінці будь-якого рядка на клавішу **Del** поточний рядок і наступний поєднуються (тобто ці два символи разом вилучаються).

Для роботи з текстовими файлами використовуються спеціальні процедури:

`Assign(f; Name:String)` – зв'язує файлову змінну `f` з повним зовнішнім ім'ям файлу на диску, включаючи маршрут до нього.

`AssignCRT(Var f:Text)` – зв'язує файлову змінну з екраном монітора. Визначена в `Unit Crt`. Ця процедура необхідна у тому випадку, коли приходиться виводити інформацію на екран через файлову змінну і потрібно використовувати поточний установлений колір символів. Якщо просто використовувати процедуру `Assign`, то текст буде виводитися білим кольором на чорному тлі.

`Append(Var f:Text)` – відкриває існуючий файл для додавання рядків тексту. ***Якщо файл відсутній на диску, то виникає помилка вводу-виводу.***

`Rewrite(f)` – створює новий текстовий файл, до якого можна лише додавати рядки. ***Якщо файл з таким ім'ям вже існує на диску, то він видаляється і створюється новий.***

`Reset(f)` – використовується лише до існуючого файлу, після чого з цього файлу можна тільки послідовно читати. Коли новий текстовий файл закривається, до нього автоматично додається маркер кінця файлу `EOF` (**Ctrl+Z**).

`Read(Var f:text, W1[,W2, ...Wn])` – розширення стандартної процедури читання `Read`, що дозволяє працювати зі значеннями символьного типу, читаючи інформацію з файлу (якщо він заданий, інакше з клавіатури) у задані змінні. Здійснює читання з файлу `f`, де `W1[,W2, ...Wn]` – змінні стандартного паскалевського типу, у які і містяться або символи, або числа, отримані інтерпретацією символів цифр із файлу `f`. Тобто, рядок із двох символів «4» і «6» інтерпретується як число 46 і т.д. Роздільниками у рядку в цьому випадку є символ пробілу, а поміж рядками – маркер кінця рядка (**CR**/**LF**).

`ReadLn(Var f:text, W1[,W2, ...Wn])` – діє аналогічно `Read`. Відмінність полягає в тому, що після прочитання даних у змінні, пропускаються всі символи, що залишилися у даному рядку і маркер кінця рядка. Якщо в процедурі відсутній список змінних, то відбувається перехід до наступного рядку.

`Write(f, W1[,W2, ...Wn])` – переводить числа `W1[,W2, ...Wn]` із заданих змінних у послідовність відповідних символів і записує їх у файл `f`, не розділяючи пробілами. ***Тому, при необхідності, пробіли і маркери кінця рядка необхідно вставляти самостійно***. Для символьних, арифметичних і строкових змінних у файл виводяться їхні значення, а для `Boolean` виводиться рядок `True` чи `False` у залежності від його значення. Якщо поле задає більше

позицій, чим потрібно для виводу, то воно зліва доповнюється пробілами, інакше виводяться самі праві символи з представлення змінної (рис. 8.84).

У наступному прикладі (рис. 8.85) на диску **D** у кореневому каталозі формується (записується) текстовий файл. Дані з цього файлу, у свою чергу, можуть бути у майбутньому зчитані для використання в програмі користувача.

8.26.9. Функції для роботи з файлами

Для обробки текстових файлів мається ще і ряд корисних функцій:

`EoF[(f:Text)]:Boolean` – повертає `True`, якщо наступний за останнім прочитаним символом є маркер кінця файлу. Якщо файлова змінна не зазначена, то передбачається стандартний файл вводу.

```
Program WriteFile;
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('!',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. Вивід інформації оператором `Write`

`Eoln[(f:Text)]:Boolean` – повертає `True`, якщо наступний за останнім прочитаним символом є маркер кінця рядка. Якщо файлова змінна не зазначена, то передбачається стандартний файл вводу. Якщо `EoF(f)` – істина, то і `Eoln(f)` – істина.

`SeekEoln[(f:Text)]:Boolean` – аналогічна `Eoln`, однак до перевірки на маркер кінця рядка видаляє всі наступні пробіли і символи горизонтальної табуляції.

`SeekEoF[(f:Text)]:Boolean` – аналогічна `Eoln`, однак до перевірки на маркер кінця файлу видаляє всі наступні пробіли, символи горизонтальної табуляції і **CR**/**LF**.

```
Program FileWorks;
Uses Crt;
Const
  Bar : Char = ' '; {Пробіл – #32}
  lf  : Char = #10; {Line Feed – переведення рядку}
  cr  : Char = #13; {Carriage Return – повернення каретки}
Var
  f    : Text; {Оголошуємо логічну змінну f текстового типу}
  n, i : integer; {Цілі змінні для запису в файл}
 Begin
  ClrScr;
  Assign (f, 'd:\ee.dat'); {Зв'язуємо логічну змінну f з фізичним
                            файлом на диску для вводу (запису)
                            тексту}
  ReWrite(f); {Створюємо новий текстовий файл, до якого можна тільки
               додавати строки}
  i:=1; {Присвоюємо значення 1 цілій змінній i}
  n:=2; {Присвоюємо значення 2 цілій змінній n}
  Write(f, i, Bar); {Записуємо в буфер 1 і пробіл}
  Write(f, n, cr, lf); {Додаємо 2 й маркер кінця строки CR/LF}
  Write(f, n); {Додаємо 2 у наступну строку}
  Close(f); {Закриваємо файл. При цьому інформація з буфера
             записується у файл на диску, автоматично додається маркер
             кінця файлу EOF, для файла записуються дані: час запису,
             довжина у байтах, тип (архівний, прихований і т.д.). Тепер
             ним можна користуватися}
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

Приклад: програма, яка запитує у діалозі вихідний пристрій (це може бути консоль, принтер чи дисковий файл) і виводить на нього файл, ім'я якого теж визначає користувач. Програма закінчує роботу, коли замість букви, що визначає пристрій (`C`, `P` чи `D`), користувач натисне клавішу **Esc**. При завданні імені вхідного файлу програма перевіряє його наявність на диску. При виводу

файлу можна призупинити роботу програми, натиснувши клавіші **Ctrl+S**. Для продовження виводу – натиснути будь-яку клавішу (рис. 8.86).

```
Program SelectFile;
Uses Crt;
Var
  F : Text;
  S : String;
Function SelectDevice(Var F : Text) : Boolean;
Var
  Ch    : Char;
  Fname : String;
Begin
  HighVideo; {Включення яскравості кольору символів}
  Write(‘ C’);
  LowVideo; {Виключення яскравості кольору символів}
  WriteLn(‘. Вивід файлу на екран’);
  HighVideo;
  Write(‘ P’);
  LowVideo;
  WriteLn(‘. Вивід файлу на принтер’);
  HighVideo;
  Write(‘ D’);
  LowVideo;
  WriteLn(‘.Вивід файлу на диск’);
  HighVideo;
  Write(‘ Esc’);
  LowVideo;
  WriteLn(‘ – закінчити роботу.’);
  WriteLn;
  Write(‘ Ваш вибір : ’);
  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;
```

Рис. 8.86. Вивід файлу на різноманітні пристрої

```
  Case Ch Of
    'C' : Assign(F, 'Con');
    'P' : Assign(F, 'Prn');
    'D' : Begin
            Write('Ім'я вихідного файлу :');
            ReadLn(FName);
            Assign(F,FName);
          End;
  End; {Case}

  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 Ж');
    HighVideo;
    ReadLn(FName);
    LowVideo;
    Assign(F, FName);
    {$I-} {Відкл. аварійне завершення при виникненні помилки I/O}
    Reset(F);
    {$I+} {Відновлюємо режим}
    IOCode:=IOResult;
    If IOCode>0 Then
      WriteLn(^G^G' Файл ',Fname,' не найден!');
  Until IOCode=0;
End; {Proc GetInputName}
```

Рис. 8.86. Вивід файлу на різноманітні пристрої (продовження)

```
Begin {Початок Proc PrintFile}
  GetInputName(FIn);
  While Not Eof(FIn) Do
    Begin
      If KeyPressed Then {Натиснуто клавішу}
         Begin
           Ch:=ReadKey;
           If Ch=^S Then {Потрібна пауза}
              Ch:=ReadKey; {Продовжимо по будь-якій клавіші}
         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. Вивід файлу на різноманітні пристрої (закінчення)

8.26.10. Рішення задачі з використанням рекурсії, множин і текстових файлів

Тепер, коли ми з Вами розібралися з поняттями рекурсії, множин і текстових файлів, вирішимо комплексну задачу на використання цих понять.

Отже, поставлена наступна задача[1]. У комп'ютерній мережі використовується деяка множина різних протоколів. Кожен вузол (комп'ютер) підтримує декілька протоколів. Усі вузли між собою пов'язані. Пакет даних може бути переданий з вузла на вузол тільки у випадку, якщо у цих двох вузлів мається загальний протокол. Спочатку пакет був посланий першому вузлу. Знайти усі вузли, до яких дійде цей пакет, і вивести їхню кількість. (Усього вузлів може бути не більш 100, а протоколів не більше 50). Приклад мережі наведений на малюнку рис. 8.87.

Вхідні дані. Файл `e.dat`, що містить текстові дані наступного формату:
– перший рядок – кількість вузлів ***n***;
– кожен з наступних ***n*** рядків – кількість підтримуваних протоколів і номера цих протоколів.

[1] **Бобак И. Думай рекурсивно // Мой компьютер. – 2001. – № 23. – С. 36-39. (ibobak@torba.com)**

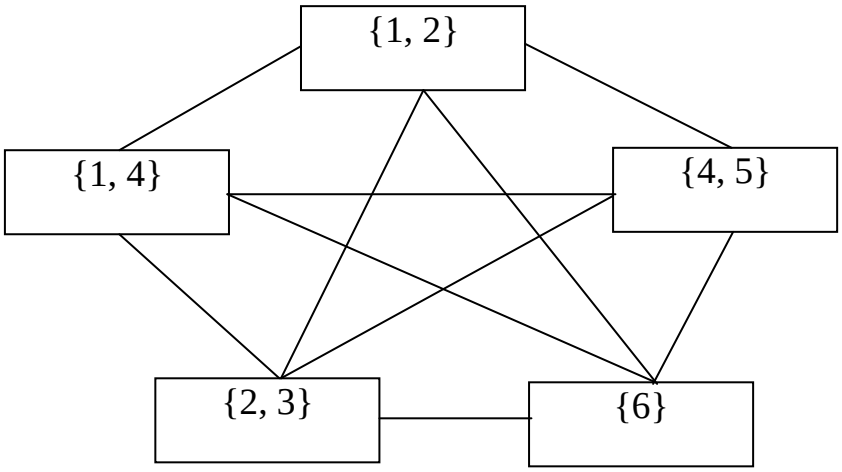


Рисунок 8.87. Схема мережі

Приклад даних для вводу:

```
5
2 1 2
2 1 4
2 2 3
1 6
2 4 5
```

Вихідні дані. Кількість вузлів, що можуть одержати пакет, посланий спочатку першому вузлу. Приклад (для наведених вхідних даних):

```
4
```

Нижче приведений алгоритм рішення поставленої задачі.

```
Program Recursion;
Uses Crt;
Type
  TProtocolSet = Set Of 1..50; {TProtocolSet – тип-безліч}
Var
  f : Text; {Описуємо текстовий файл}
  A : Array[1..100] Of TProtocolSet;{Масив, елементами
                    якого є множина TprotocolSet, що містить
                    номера протоколів для кожного вузла}
   was : array[1..100] of Boolean; {Статус обробки вузла. На
                    початку він повинний містити значення
                    False. Вузли обходимо, починаючи з
                    першого. Спочатку ввійдемо в перший вузол.
                    Далі з нього увійдемо в усі вузли, що мають з
                    ним загальний протокол. Потім для кожного з
```

них проробимо то ж саме. Ті вузли, де ми уже побували, будемо відзначати в булевому масиві значенням True.}

```
  n   : integer; {Кількість вузлів}
  cnt : integer; {Результат – кількість вузлів, до яких дійде пакет}

Procedure Rec(const p : Integer);
 {=*= Рекурсивна процедура, що визначає кількість вузлів cnt (глобальна
                змінна), до яких дійде пакет =*=}
{Суть процедури Rec складається в поширенні пакета. Зробивши виклик
 Rec(1), ми викличемо її рекурсивно для вузлів, до яких пакет може
 прийти з вихідного вузла. Далі, для останніх буде зроблений рекурсивний
 виклик процедури Rec з номерами вузлів, у які пакет може надійти в
 другу чергу і т.д. Для нашого випадку буде: 1, 2→1, 4; 1, 2→2, 3; 1, 4→4,
 5}

var
  i : Integer; {Параметр циклу for}
Begin
  if was[p] then
    exit; {Процедура exit завершує роботу програмного блоку}
  was[p] := True; {Відзначаємо вузол, який відвідано}
  inc(cnt); {Убудована процедура inc збільшує на 1 значення
               аргументу}
  for i:=1 to n do {Цикл обходу вузлів (n – глобальна змінна)}
  if A[i]*A[p] <> [] then { A[i]*A[p] – перетинання множин.
                Результатом є множина, що складається тільки з тих
                елементів двох множин, що містяться одночасно в
                кожній з них. [] – порожня безліч. У нашому
                випадку – це відсутність загального протоколу даних
                у двох вузлів}
    Rec(i); {Якщо вузли p і i мають загальний протокол, значить
             дані дійдуть до вузла i. Тоді рекурсивно викликаємо
             процедуру Rec}
End; {Proc Rec}

Procedure Init;
{=*= Процедура зчитування даних =*=}
Var
  i, j, sc, p : Integer; {i, j – параметри циклів; sc –
                          кількість протоколів на вузлі; p –
                          номер протоколу}
Begin
```

```
  FillChar(A, SizeOf(A), 0); {Убудована процедура заповнення
                              масиву (A) числом (SizeOf(A), де
                              SizeOf – убудована функція, що
                              повертає розмір типу з ім'ям A )
                              значенням (0)}
  cnt := 0; {Ініціалізація cnt}
  for i:=1 to n do {Цикл по числу n вузлів мережі, прочитаних в
                    основній програмі з файлу e.dat}
    begin
      Read(f, sc); {Читаємо з файлу e.dat кількість протоколів
                    sc на вузлі i}
      for j:=1 to sc do {Цикл по числу sc протоколів}
        begin
          read(f, p); {Читаємо з файлу e.dat номера
                       протоколів}
          include(A[i], p); {Включаємо убудованою
                             процедурою include номера
                             протоколів у i-ту множину
                             TProtocolSet}
        end;
    end
End; {Proc Init}

Begin {Початок основної програми}
  ClrScr; {Очищаємо екран}
  Assign (f, 'e.dat'); {Установлюємо зв'язок між логічним
                        файлом f і фізичним файлом e.dat}
  Reset(f); {Відкриваємо логічний файл f для читання}
  Read(f, N); {Читаємо з файлу e.dat кількість вузлів мережі N}
  Init; {Викликаємо процедуру зчитування даних про протоколи з
         файлу e.dat}
  Rec(1); {Передаємо пакет у перший вузол. Рекурсивно він
           поширюється в усі інші, якщо це можливо}
  WriteLn(cnt); {Друкуємо результат – кількість вузлів}
  Close(f); {Закриваємо фізичний файл e.dat}
End.
```

При застосуванні рекурсії варто піклуватися про те, щоб можна з неї було вийти. Рекурсія в цьому прикладі не нескінченна, вихід з неї передбачений: ми перевіряємо факт відвідування (обробки) вузла оператором:

```
if was[p] then exit;
```

і якщо вузол вже був оброблений, виходимо з рекурсії.

Вправи

1. Створіть рекурсивну програму «Ханойська Вежа (рос.-башня)». Умова наступна: мається дошка з трьома кілочками (рис. 8.88). На першому з них нанизано кілька дисків зростаючого до низу діаметра. Потрібно розташувати ці диски в такому ж порядку на третьому кілочку. Диски можна перекладати тільки по одному, а класти більший на менший не можна.

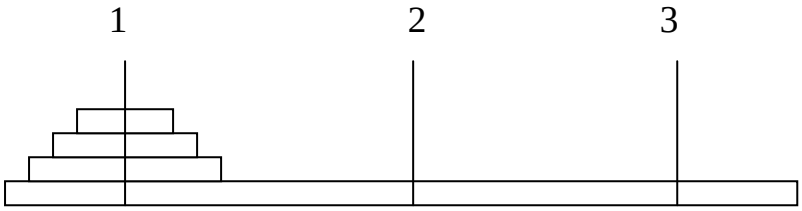


Рис. 8.88. Вигляд гри «Ханойська Вежа»

Алгоритм рішення головоломки наступний:

1.1. Перемістити верхні n-1 дисків на 2-ий кілочок.

1.2. Нижній диск із першого кілочка перемістити на третій.

1.3. Перемістити n-1 дисків на третій кілочок, використовуючи перший у якості допоміжного.

1.4. Повторювати, поки на третьому кілочку не буде сформована нова піраміда.

Вихідна задача зводиться до двох про перенос n-1 диска й одній задачі про перенос одного диска. Для n=1 потрібно одне переміщення.

2. Розробіть програму з застосуванням множин для рішення наступної задачі: користувачу пропонується ввести одну з букв A, L, P чи T, причому байдуже в якому регістрі – прописному чи рядковому. Якщо те, що введе користувач, буде відмінно від кожної з зазначених літер, програма повинна надрукувати повідомлення, що запрошує до повторного введення.

3. У текстовому файлі містяться довжини сторін 20 прямокутних паралелепіпедів. Обчислити обсяги цих паралелепіпедів. Результати розрахунку записати у файл у виді таблиці, що містить стовпчики: довжина першої сторони паралелепіпеда, довжина другої сторони паралелепіпеда, довжина третьої сторони паралелепіпеда, об’єм паралелепіпеда. Подбайте про відповідні заголовки для стовпчиків.

Файл із вихідними даними сформуйте за допомогою текстового редактора.

4. Трансформуйте текст програми про доставку пакетів у мережі (мал. 76) так, щоб у процедури глобальні параметри передавалися через формальні параметри.

8.27. Записи, Посилання, динамічні змінні й структури

8.27.1. Тип “запис” (record) й оператор приєднання with

Для компактного представлення комбінацій різнотипних даних у Турбо Паскалі їх можна об'єднувати в структурі-записи. Кожен запис складається з оголошеного числа полів і визначається конструкцією:

```
Var
  InfRec : Record;
              Поле1 : Тип_поля1;
              Поле2 : Тип_поля2;
                . . .
              ПолеN : Тип_поляN
           End;
```

Де Тип_поляN, це деякий тип даних ТП. Зверніть увагу, що після службового слова Var перед описом типу даних Record ставиться знак (:), а після Type – знак (=) (дивись наступний приклад).

Якщо типи декількох полів збігаються, то імена полів можуть бути просто перераховані (рос.-перечислены), наприклад – x,y :

```
Type
PointRecType = record x,y : Integer end;
Var
Point : PointRecType;
Px, Py : Integer;
. . .
{Звертання до полів запису:}
Px := Point.x;
Py := Point.y;
. . .
```

Оскільки імена полів «сховані» у середині типу, то вони можуть дублювати «зовнішні» змінні і поля в інших описах записів:

```
Type
  PointRecType = record x, y : integer end;
  ColorPointType = record
                     x, y : integer;
                     color : word
                   end;
Var
  X, Y : integer;
  Point : pointRecType;
  ColorPoint : ColorPointRecType;
. . .
```

Зверніть увагу також на те, що у цьому прикладі `x, Point.x` і `ColorPoint.x` – зовсім різні значення.

Наприклад, Ви можете розробити свої користувальницькі типи даних (рис. 8.89). Зв'язок значення у операторі варіантної частини запису з певними значеннями компілятором ніяк не відстежується. Запис з варіантами може використовуватися, наприклад, для читання полів різних типів з файлів бази даних у одну й ту ж область пам'яті.

```
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. Приклад конструювання типів “запис”

Змінна типу "запис" може брати участь тільки в операціях присвоювання. Але поле запису може брати участь у всіх операціях, які відносяться до типу цього поля. Для доступу до полів запису застосовується кваліфікаційне ім'я (рис. 8.89):

```
Users[3].Processor
```

Для полегшення роботи з полями записів вводиться оператор приєднання (рос. – присоединения) `With`:

`With` <Ім'яЗмінноїЗапису> `do` <Оператор>;

Усередині оператора `With` (він може бути складеним) звертання до полів запису виробляється прямо (рис. 8.90). Усередині області дії оператора приєднання `With` можуть вказуватися і змінні, що не мають відношення до запису, але необхідно дотримуватись наступних правил (рис. 8.91).

У випадку, якщо одне з полів запису саме є записом і знову містить поля запису, оператор приєднання можна поширити на декілька полів усередину, перелічивши їх через кому. Але усередині тіла оператора можна звертатися тільки до останніх полів:

```
With Им'яЗапису.Поле_Запис do
Begin
   {Звертання до полів Поле_Запис, тобто до тих, яким передувала
     конструкція “Им'яЗапису.Поле_Запис”}
End; {With}
```

```
{Два еквівалентних способу звертання до полів запису 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 еквівалентних способу звертання до полів запису

```
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 «розв'язки» співпадаючих
                               ідентифікаторів. Для Z  не потрібно}
    End; {With}
  . . .
End.
```

Рис. 8.91. Змінні, що не мають відношення до запису у операторі `With`

8.27.2. Система адресації MS-DOS

Адресуємий простір пам'яті в MS-DOS організовано сегментами: послідовними блоками пам'яті по 64Кб кожний. Якщо Вам відомий сегмент, то подальше уточнення адреси відбувається по його зсуву (рос – смещению), тобто номеру байта від початку сегмента. Таким чином, будь-яка ячейка адресуємого простору пам’яті визначається парою чисел:

<СЕГМЕНТ> : <ЗСУВ>,

який займає чотири байти: 2 байти <СЕГМЕНТ> і 2 байти <ЗСУВ>. Такий спосіб дозволяє адресувати більший простір оперативної пам’яті меншими числами. Щоб Вам простіше було це представити, наведемо такий приклад. У деякому районі побудовано 20 будинків по 99 квартир. Потрібно знайти спосіб адресації, щоб не виходити за межи двозначних чисел, бо наскрізна нумерація всіх квартир потребує можливості представляти максимальне число:

$$20 \times 99 = 1980,$$

яке займає чотири розряди. Вихід з такої ситуації полягає у подвійній адресації:

<БУДИНОК> : <КВАРТИРА>

Тоді будь-яку квартиру Ви можете знайти по номеру будинку та номеру квартири у цьому будинку. Наприклад, будинок 10, квартира 67:

<10> : <67>,

тоді як при наскрізній нумерації було б потрібно три розряди: $10 \times 67 = 670$.

8.27.3. Тип Pointer

Основним механізмом для організації динамічних даних у ТП є виділення в спеціальній області пам'яті, називаною «купою» (рос. – куча), ділянки (блоку) необхідного розміру і збереження адреси початку цієї ділянки в спеціальній змінній у форматі <СЕГМЕНТ> **:** <ЗСУВ>.

Умовимося називати надалі ***указником*** (рос.-указателем) змінні, котрі мають узагальнений тип `Pointer` – ***указник***. Тобто можна оголошувати змінні, значеннями котрих будуть адреси ячеєк пам’яті:

```
Var
  P : Pointer; {Змінна-указник}
```

Значення цього типу займають 4 байти пам’яті і містять адресу початку будь-якого об’єкту ТП. Адреса зберігається як два слова, тобто дві змінні типу `Word` (кожна з них займає у пам’яті 2 байти): одне з них задає ***сегмент***, а інше – ***зсув***. Значення ***указника*** не може бути виведене на екран та на друк. Його треба попередньо розшифровувати.

Майте на увазі, що компілятор ТП завжди повинен знати, який об'єкт адресується, щоб коректно його обробляти. Згадайте, що змінна типу `Integer` займає у пам’яті 2 байти, змінна типу `LongInt` – 4 байтb, змінна типу `Real` – 6 байтів, змінна типу `Byte` займає у пам’яті 1 байтів і т.д (див додаток 7).

Ще одна з тонкощів роботи у ТП полягає в тому, що коли Ви пишете листа до близької людини, то відступаєте абзаци, грамотно формулюєте фрази, де потрібно ставите коми, тире, крапки і т.д. Так от, програма, яку Ви пишете до транслятора, теж повинна бути грамотно написана, щоб він зміг її зрозуміти...

8.27.4. Засоби роботи з адресами

Для початку розглянемо функції для роботи з адресами різних об’єктів ТП (табл.. 52).

Таблиця 52

Функції для роботи з адресами різних об’єктів ТП

Функція : Тип	Значення, яке повертається
`Addr(X) : Pointer`	Посилання на початок об’єкту `X` у пам’яті
`Seg(X) : Word`	Сегмент, у якому зберігається об'єкт `X`
`Ofs(X) : Word`	Зсув у сегменті для об’єкту `X`
`Ptr(S, 0 : Word) : Pointer`	Посилання на місце у пам’яті яке задане значеннями зсуву 0 і сегменту `S`
`SizeOf(X) : Word`	Розмір об’єкту `X` у байтах
Операція `@X : Pointer`	Повертає посилання на початок об’єкту `X` у пам’яті (аналог функції `Addr`)

Функції `Addr(X)`, `Seg(X)` та `Ofs(X)`, а також оператор `@` повертають адресу об’єкту `X` або компоненти цієї адреси. Під змінною `X` можна розуміти будь-який об'єкт: змінні вбудованих типів ТП, користувацькі типи, об'єкти, процедури і функції (але не константи).

Функція `Addr(X)` та оператор `@` повертають значення типу `Pointer` – адресу об’єкту `X`. Їх дія однакова:

```
Var
  X : String;
p, q : Pointer;
 . . .
    p:=Addr(X);
    q:=@X; {Тепер p=q , тобто їх адреси рівні: вони вказують на один
            й той же об'єкт}
```

Як Ви вже знаєте, значення типу `Pointer` не може бути виведено на екран. Але оскільки цей тип складається з двох слів (`Word`), які зберігають сегмент та зсув, їх можна вивести поодинці, використовуючи функції `Seg` та `Ofs` (обидві типу `Word`):

```
WriteLn('Сегмент ', Seg(p), ' смещение ', Ofs(p));
```

Функція `Ptr(Seg, Ofs : Word)` виконує протилежну функції `Addr(X)` роботу: вона організовує посилання на місце у пам'яті яке визначене заданим сегментом та зсувом. Необхідність у такій функції може виникати, коли потрібно наложити динамічну структуру на системну область пам'яті. Так, наприклад, відомо, що образ текстового екрану починається з адреси $B000:$0000 й займає 4 000 байт (кольоровий та чорно-білий режими, 80 стовпців на 25 строк), то можна "наложити" на нього якусь структуру, наприклад, масив, використовуючи посилання на такий масив та функцію `Ptr` (рис. 8.92):

```
Type
  VideoArray = Array[0..3999] of Byte;
Var
  V : ^VideoArray; {Посилання на структуру}
Begin
  V:=Ptr($B000, 0); {Далі V^[i] звертається безпосередньо до ячеєк
                     відеопамяті у текстовому режимі}
 . . .
End.
```

Рис. 8.92. Використання функції `Ptr(Seg,Ofs : Word)`

Функція `SizeOf(X) : Word` повертає об'єм у байтах, який займає об'єкт `X`. Причому `X` може бути не тільки змінною, але й ідентифікатором типу (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:

```
Xtype  =   100   L =    4    X =   256
```

Рис. 8.93. Використання функції `SizeOf(X) : Word`

Значення SizeOf(строка) завжди дає максимальне значення довжини строки. Реальне значення дає функція Legth.

8.27.5. Посилальні змінні

Усі посилальні змінні мають однаковий розмір, рівний 4-м байтам, і містять адреса початку ділянки оперативної пам'яті, у якому розміщена конкретна динамічна структура даних (Integer, Array, Record і т.ін.), наприклад:

```
 . . .
Var
  J : ^Integer; {Посилальна змінна, що указує на ціле значення}
  I : Integer; {Ціла змінна}
 . . .
   J := 200;
   I := 200;
 . . .
```

Різниця між цілою змінною I (рис. 8.94, а) та посилальною змінною J на цілу змінну полягає у наступному (рис. 8.94, б):

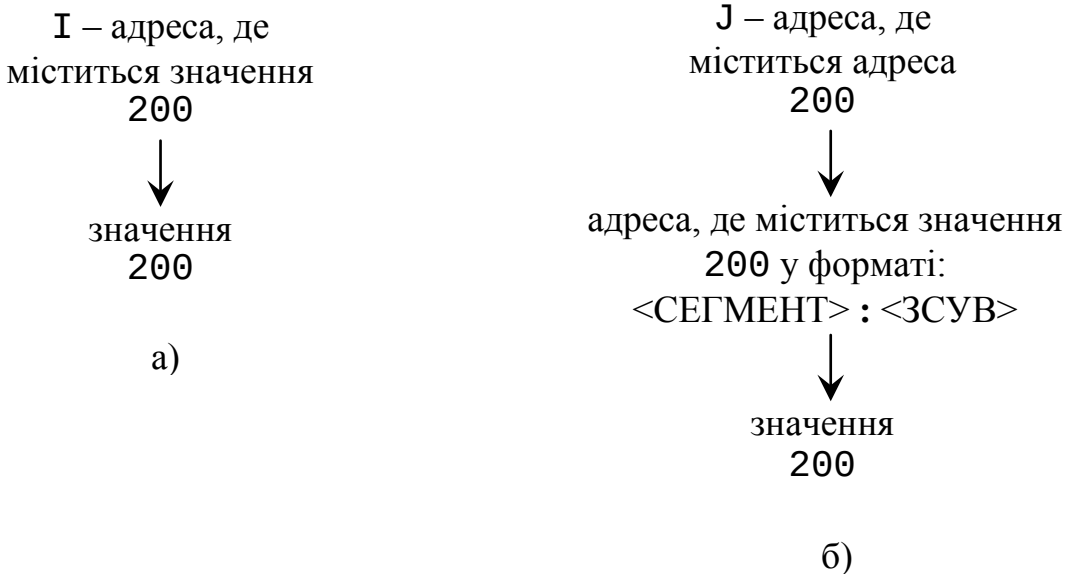


Рис. 8.94. Зміст даних різного типу I та J

Щоб посилання ні на що не вказувало, йому привласнюється значення Nil:

```
J := nil;
```

Це визначена константа типу Pointer, що відповідає адресі 0000:0000.

Підсумовуючи усе вищесказане можемо зробити висновок, що для визначення посилальної змінної у ТП потрібно описати її як посилальний тип:

```
Type
   Ім'яПосилальногоТипу = ^Ім'яБазовогоТипу;
    {Де Ім'яБазовогоТипу – будь-який ідентифікатор типу (Real, Integer і
      т.д.), це потрібно компілятору ТП для організації роботи програми}
```

У результаті цього визначення створювані потім посилальні змінні будуть указувати на об'єкти базового типу, визначаючи тим самим динамічні змінні базового типу:

```
Type {БАЗОВІ ТИПИ}
  DimType = Array [1..10000] of Real; {Масив}
  RecType = record ... end; {Запис}
   { ПОСИЛАЛЬНІ ТИПИ }
  IntPtr = ^Integer; {Посилання на ціле значення}
  Dim Ptr = ^DimType; {Посилання на масив даних}
  RecPtr = ^RecType; {Посилання на запис}
  XXXPtr = Pointer; {Посилання «узагалі» - указник}
```

8.27.6. Операція розіменування

Суть цієї операції складається в переході від посилальної змінної через адресу, на яку вона посилається до значення, на яке вона вказує. При цьому слідом за посилальною перемінною указується символ "^":

```
 . . .
Var
  I, J : ^Integer;
 . . .
I:=2;
J:=4;
J^:=I^; {Копіюємо вміст I у J. Зараз  указує на 2  (рис. 8.95, а)}
J:=I; {Або можна так: I  і  J  тепер теж  указують на  2  (рис. 8.95, б)}
```

Але ці операції виконуються компілятором ТП по різному (рис. 8.95, а, б).

Ячейка зі значенням 4 перетворюється у «сміття», оскільки до неї тепер немає доступу.

Посилальні змінні і указники сумісні між собою по типу, тобто немає помилки в присвоюванні

```
DimPtr := RecPtr;
```

Але після розіменування починається контроль типів об'єктів по зазначених адресах:

```
DimPtr^ := RecPtr^; {Дає помилку!!!}
```

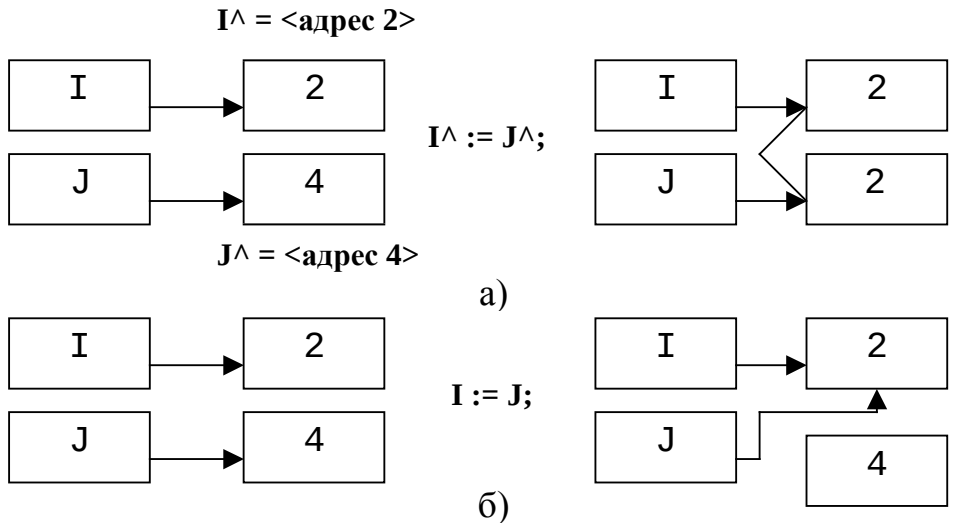


Рис. 8.95. Операції розіменування

Розіменовані посилання на структури (масиви) індексуються або розділяються на поля запису звичайним образом:

DimPtr^[i] – доступ до елемента I динамічного масиву
RecPtr^.Поле – доступ до поля динамічного запису

8.27.7. Розміщення динамічних змінних. Процедури `New` і `GetMem`

Розміщення динамічних змінних у ТП виконується процедурами `New(Var P: Pointer)` і `GetMem(Var P: Pointer)` (табл. 52).

Таблиця 52.

Процедури розміщення динамічних змінних

Процедури і функції	Призначення
`New(Var P : Pointer)` або `New(ТипПосилання) : Pointer`	Відводить місце для збереження динамічної змінної `P^` і привласнює її адресу посиланню `P`
`GetMem(Var P: Pointer; Size : Word)`	Відводить місце в `Size` байт у купі (Heap), присвоюючи адресу його початку указнику (посиланню) P

Як Ви вже знаєте, у ТП мається спеціальний не типізований указник `Pointer`. Він оголошується стандартним образом:

```
  . . .
Var
  X : Pointer;
  . . .
```

Не типізованому указнику може бути призначене значення будь-якого типізованого указника або навпаки. Він виконує роль своєрідного буфера для збереження адреси динамічної змінний будь-якого типу. Нехай маємо:

```
Type
   IntPtr = ^Integer; {Посилання на ціле значення}
Var
  P  :  IntPtr; {Посилальна змінна типу IntPtr, що визначає
                 посилання на ціле число}
```

Тоді при виклику

```
New(P);
```

Або

```
P:=New(IntPtr);
```

У купі виділяється блок пам'яті розміром `SizeOf (Integer)` тобто 2 байти й адреса першого байта цього блоку запишеться у `P`. Після виконання `New` можна вже посилатися на динамічну змінну `P^`.

Окрім того, можна виклик `New (P)` замінити на `GetMem (P, SizeOf (Ім'яБазовогоТипу_P))`, де `SiZeOf (X)` – стандартна функція ТП, що повертає розмір у байтах базового типу.

Процедура `GetMem` призначена для виділення пам'яті указникам:

```
Var
  P : Pointer; {Оголошуємо указник}
Begin
  GetMem (P, 4*1024); {Тепер P указує на блок пам'яті в купі
                       розміром 4Кб. P^ не має типу, але
                       містить 4096 байт}
 . . .
End.
```

8.27.8. Звільнення динамічних змінних. Процедури Dispose і FreeMem

Для звільнення пам’яті, яку займає динамічна змінна `P`, використовується процедура `Dispose (Var P: Pointer)`. Ця процедура працює тільки з типізованими посилальними змінними (табл.. 53). Для коректної роботи Ви завжди повинні робити виклики `Dispose` парними викликам `New`. Після виклику процедури значення посилання `P` не визначене, як і значення розіменування `P^`.

Таблиця 53.

Процедури звільнення динамічних змінних у ТП

Процедури і функції	Призначення
`Dispose (Var P: Pointer)`	Знищує зв'язок, створений раніше `New`, між посиланням P і значенням, на яке вона посилається
`FreeMem (Var P: Pointer; Size : Word)`	Звільняє `Size` байт у купі, починаючи з адреси, записаної в указнику (посиланні `P`)

Для звільнення безупинних ділянок пам'яті заданого розміру потрібно використовувати процедуру

```
FreeMem (Var P : Pointer; Size : Word)
```

Виклики `FreeMem`, як і `Dispose`, повинні бути парні викликам `GetMem`. Значення посилальної змінної `P` після виклику `FreeMem` вважаються невизначеними.

8.27.9. Поєднуємо разом поняття Record і динамічних змінних: рішення задачі по створенню динамічних структур типу "черга"

Найбільш важливим аспектом програмування з використанням динамічних структур даних є стекові структури, однозв'язкові і двозв’язкові списки, черги, двоїчні дерева і т.д. Для програмної підтримки всіх цих структур можна застосовувати масиви, але при ближчому розгляді стає очевидним, що або необхідно описувати досить великі масиви з відомою надмірністю, або, у випадку визначення свідомо малих масивів, область прикладного застосування програми буде обмежена. Альтернативою масивам служать указники у сполученні зі структурою Record, у яку включають інформаційну частину й указник на наступну структуру Record. Їхня перевага полягає як у тім, що вони дозволяють створювати динамічні структури необхідної розмірності, так і в тім, що ми одержуємо можливість оперувати великими об'єктами за допомогою усього лише 4-байтних указників.

8.27.10. Логічна структура черги FIFO

Чергою *FIFO* (First In – First Out – "першим прийшов – першим виключається") називається такий послідовний список зі змінною довжиною, у якому включення елементів виконується тільки з однієї сторони списку (цю сторону часто називають ***кінцем*** чи ***хвостом черги***), а виключення – з іншої сторони (називаної ***початком*** чи ***головою черги***). Ті самі черги, що виникають до прилавків і кас, є типовим побутовим прикладом черги FIFO.

Основні операції над ***чергою*** – це включення і виключення елементів, визначення розміру черги, очищення, не руйнуюче читання.

Черги будуються на базі ***лінійних списків***. Представлення ***черги*** за допомогою ***лінійного списку*** дозволяє досить просто забезпечити будь-які бажані операції обслуговування черги. Особливо це зручно, коли число елементів у черзі важко передбачити. Лінійні списки також знаходять широке застосування в додатках, де непередбачені вимоги до розміру пам'яті, необхідної для збереження даних; мається велике число складних операцій над даними, особливо включень і виключень.

8.27.11. Зв'язні лінійні списки

Списком називається упорядкована безліч, що складається з перемінного числа елементів, до яких застосовні операції ***включення*** і ***виключення***. ***Список,*** що відбиває відносини сусідства між елементами, називається ***лінійним***. Якщо обмеження на довжину списку не допускаються, то список представляється в пам'яті у виді динамічної зв'язної структури. ***Лінійні зв'язні списки*** є найпростішими динамічними структурами даних.

Графічно зв'язку в списках зручно зображувати за допомогою стрілок. Якщо компонент не зв'язаний ні з яким іншим, то у полі указника записують значення, що не вказує ні на який елемент. Таке посилання позначається спеціальним ім'ям – **nil.**

На рис. 8.96 приведена спрощена структура ***однозв'язного списку***. На ньому поле **INF** – інформаційне поле, яке може включати різноманітні дані, як вбудованих, так і користувацьких типів Турбо Паскалю дані, **NEXT** – указник на наступний елемент списку. Кожен список повинний мати особливий елемент, називаний ***указником початку списку*** чи ***головою списку***, що звичайно по форматі відмінний від інших елементів. У поле указника останнього елемента списку знаходиться спеціальна ознака **nil**, що свідчить про кінець списку.

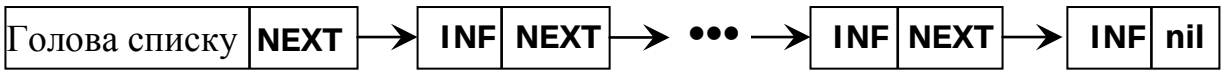


Рис 8.96. Логічна структура однозв'язного списку

8.27.12. Реалізація операцій над зв'язними лінійними списками

Для розгляду програмних прикладів визначимо наступні типи даних:

```
Type
  data = ...; { Будь-яка структура інформаційної частини списку }

Type  { Елемент однозв'язного списку (sll – single linked list):}
```

```
sllptr = ^slltype; { указник в однозв'язному списку }
slltype = record { елемент однозв'язного списку }
            inf : data;  { інформаційна частина }
            next : sllptr; { указник на наступний елемент }
          end;
```

На всіх рисунках, що демонструють операції над зв'язаними лінійними списками, суцільними лініями показані зв'язки, що малися до виконання і збереглися після виконання операції. Пунктиром показані зв'язки, установлені при виконанні операції. Значком "↯" відзначені зв'язки, розірвані при виконанні операції. У всіх операціях надзвичайно важлива послідовність зміни указників, що забезпечує коректні операції зі списком, які не торкаються інших елементів. При неправильному порядку зміни легко втратити будь-яку частину списку. Тому на рисунках поруч із установлюваними зв'язками в дужках показані кроки, на яких ці зв'язки встановлюються.

Словесні описи алгоритмів дані у виді коментарів до програмних прикладів.

8.27.13. Перебір елементів списку.

Цю операцію, Ви будете частіше інших виконувати над лінійними списками. При її виконанні здійснюється послідовний доступ до елементів списку – або до усіх до кінця списку, або до визначення шуканого елемента.

```
    { Перебір 1-зв'язного списку }
Procedure LookSll (head : sllptr);
  { head - указник на початок списку }
Var
  cur : sllptr; { адреса поточного елемента }
Begin
  cur:=head; { 1-й елемент списку призначається поточним }
   while cur <> nil do
     begin
       < обробка cur^.inf>
    {обробляється інформаційна частина того елементу, на який указує
    cur. Обробка може включати: друк вмісту інформаційної частини;
    модифікацію полів інформаційних частин; порівняння полів
    інформаційних частин зі зразком при пошуку по ключу; підрахунки
    ітерацій циклу при пошуку по номеру і т.д.}
        cur:=cur^.next;  {з поточного елементу вибирається
    указник на наступний елемент і для наступної ітерації наступний
    елемент стає поточним; якщо поточний елемент був останнім, то його
    поле NEXT містить порожній указник і тому у cur запишеться nil, що
    приведе до виходу з циклу при перевірці умови while}
     end; {while}
End; { Proc LookSll}
```

8.27.14. Вставка елемента у список.

Вставка елемента у середину однозв'язного списку показана у прикладі нижче та на рис.8.97.

```
 { Вставка елемента у середину однозв'язного списку }
Procedure InsertSll(prev : sllptr; inf : data);
 {prev – адреса попереднього елементу; inf – дані нового елементу}
Var
  cur : sllptr; {Адреса нового елементу}
Begin
  New(cur); {Виділення пам'яті для нового елементу}
  cur^.inf:=inf; {Запис у інформаційну частину елементу}
  cur^.next := prev^.next; {Елемент, який йшов за попереднім
                            тепер буде йти за новим}
  prev^.next := cur; { Новий елемент йде за попереднім}
End; {Proc InsertSll}
```

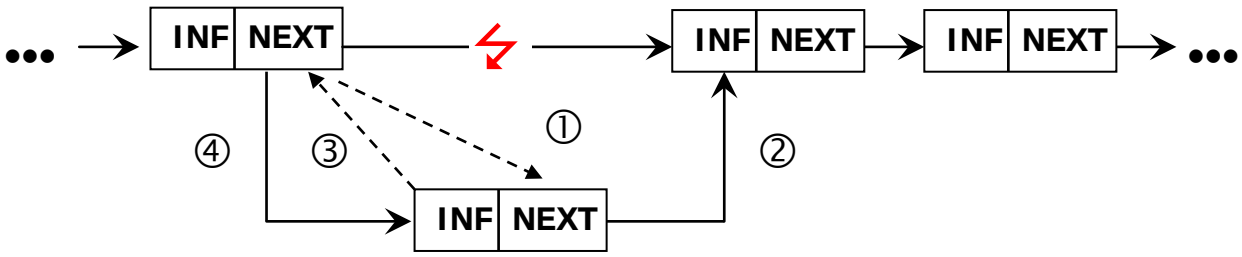


Рис. 8.97. Вставка елемента у середину списку

Ви повинні добре себе уявляти, що писля того, як Ви за допомогою процедури `New` отримали адресу нового елементу списку `cur` (current – поточний) та заповнили його інформаційну частину `inf`, Ви повинні на першому кроці ① (рис. 86) перенести у його поле **NEXT** адресу з попереднього елементу. Тоді він буде указувати на наступний елемент (крок ②). Після цього Ви повинні занести адресу `cur` нового елемента у поле **NEXT** попереднього елементу (крок ③). Тоді, після цього, попередній елемент буде указувати на новий елемент (крок ④).

У такий спосіб Ви можете зробити вставку у середину списку, але не зможете зробити вставку у його початок. При цьому повинний модифікуватися указник на початок списку, як показано на рис. 8.98.

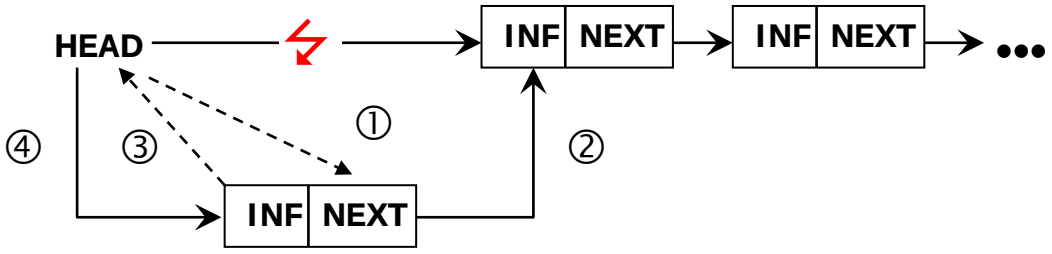


Рис. 8.98. Вставка елемента у голову списку

Як Ви бачите, операція дуже схожа на попередню (рис. 86), але тонкощі полягають у засобі зберігання указника **HEAD**. Тому, давайте розробимо процедуру, яка буде виконувати вставку елемента в будь-яке місце однозв'язного списку:

{ Вставка елемента в будь-яке місце однозв'язного списку }

```
Procedure InsertSll (Var head : sllptr; { Указник на початок
                                          списку, що може змінитися у процедурі. Якщо
                                          head=nil – список порожній }
                     prev : sllptr; {Указник на елемент, після
                                     якого робиться вставка. Якщо prev=nil – вставка
                                     перед першим елементом }
                     inf : data; {Дані нового елементу }
                     Var cur : sllptr) {Адреса нового елементу }
Begin
  New(cur);  {Виділення пам'яті для нового елемента й запис у його
              інформаційну частину}
  cur^.inf:=inf;
  if prev <> nil then
    begin {Якщо є попередній елемент – вставка у середину списку, див.
           рис.87}
      cur^.next:=prev^.next;
      prev^.next:=cur;
    end
  else
    begin {Вставка у початок списку }
      cur^.next:=head; {Новий елемент указує на колишній 1-й елемент
                        списку; якщо head=nil, то новий елемент
                        буде й останнім елементом списку}
      head:=cur; { Новий елемент стає 1-им у списку, указник на початок
                   тепер указує на нього}
   end {if}
End; {Proc InsertSll}
```

8.27.15. Видалення елемента зі списку.

Порядок видалення елемента з однозв'язного списку з середини списку та з початку декілька відрізняється. Тому, давайте почнемо з видалення елемента з середини списку (рис. 8.99).

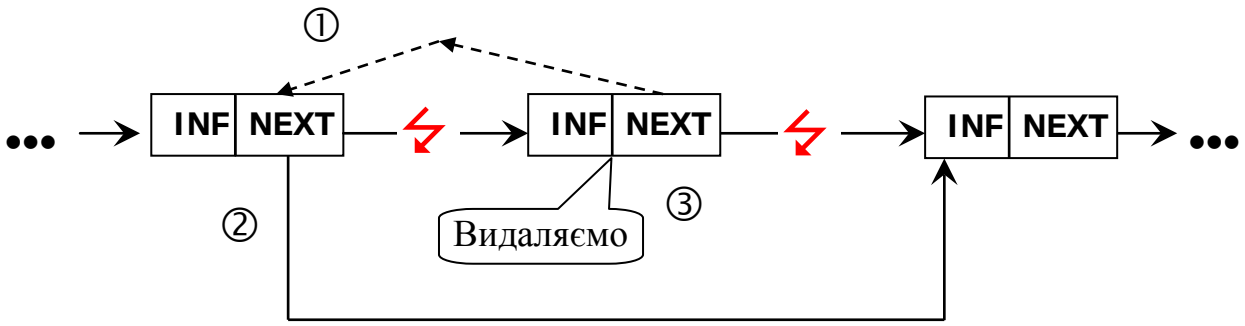


Рис. 8.99. Видалення елементу з середини списку

Тут, спершу Ви повинні указник на наступний елемент з елементу, який видаляється, перенести в попередній (крок ①), тоді на кроці ② указник обминає елемент, який видаляється. І на третьому кроці ③ Ви повинні видалити сам елемент, щоб звільнити від нього пам'ять.

У випадку видалення елементу з голови списку Ви повинні указник **HEAD** переадресувати на наступний елемент (рис. 8.100).

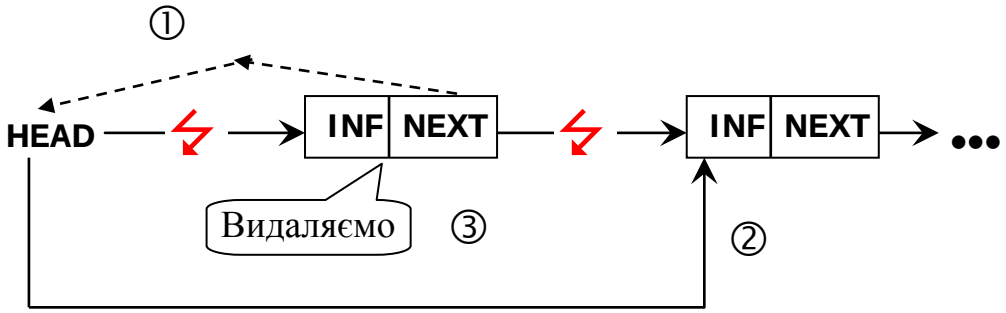


Рисунок 8.100. Видалення елементу з голови списку

Тут, спершу Ви повинні указник на наступний елемент з елементу, який видаляється, перенести в указник **HEAD** (крок ①), тоді на кроці ② указник **HEAD** обминає елемент, який видаляється. А на третьому кроці ③ Ви повинні видалити сам елемент, щоб звільнити від нього пам'ять.

Очевидно, що процедуру видалення легко розробити, якщо відома адреса елемента, що передує тому, що видаляється. Однак, більш універсальною буде процедура для випадку, коли елемент, що видаляється, задається своєю

адресою del . Процедура забезпечує видалення як із середини, так і з початку списку.

```
 {Видалення елемента з будь-якого місця однозв'язного списку}
Procedure DeleteSll(var head : sllptr; {Указник на початок
                                        списку, може змінитися в
                                        процедурі}
                    del : sllptr {Указник на елемент, який
                                  видаляється} );
var prev : sllptr; {Адреса попереднього елементу }
Begin
   if head=nil then
     begin {Спроба видалення з порожнього списку розцінюється як
            помилка (у наступних прикладах цей випадок враховуватися на
            буде)}
       Writeln('Помилка!');
       Halt;
     end;
  if del=head then {Якщо елемент, що видаляється, 1-й у списку, то
                    наступний за ним стає першим }
    head:=del^.next
  else
    begin { При видаленні із середини списку приходиться шукати елемент,
            що передує тому, що видаляється; пошук робиться перебором
            списку із самого його початку, поки не буде знайдено елемент,
            поле next якого збігається з адресою елемента, що
            видаляється}
      prev:=head^.next;
      while (prev^.next<>del) and (prev^.next<>nil) do
        prev:=prev^.next;
        if prev^.next=nil then
          begin {Це випадок, коли перебраний весь список, але елемент
                 не знайдений, він відсутній у списку; це розцінюється
                 як помилка (у наступних прикладах цей випадок
                 враховуватися на буде)}
            Writeln('Помилка!');
            Halt;
          end;
      prev^.next:=del^.next; {Попередній елемент тепер указує на
                              наступний за тим, що видаляється}
    end;
  Dispose(del); {Елемент виключений зі списку – тепер можна звільнити
                 займану їм пам'ять}
End; {Proc DeleteSll}
```

8.27.16. Перестановка елементів списку.

Мінливість динамічних структур даних припускає не тільки зміни розміру структури, але і зміни зв'язків між елементами. Для зв'язних структур зміна зв'язків не вимагає пересилання даних у пам'яті, а тільки потребує зміни указників в елементах зв'язної структури.

Давайте спробуємо зробити перестановку двох сусідніх елементів списку. Однак, по-перше спробуємо розібратись, як проходить цей процес (рис. 8.101).

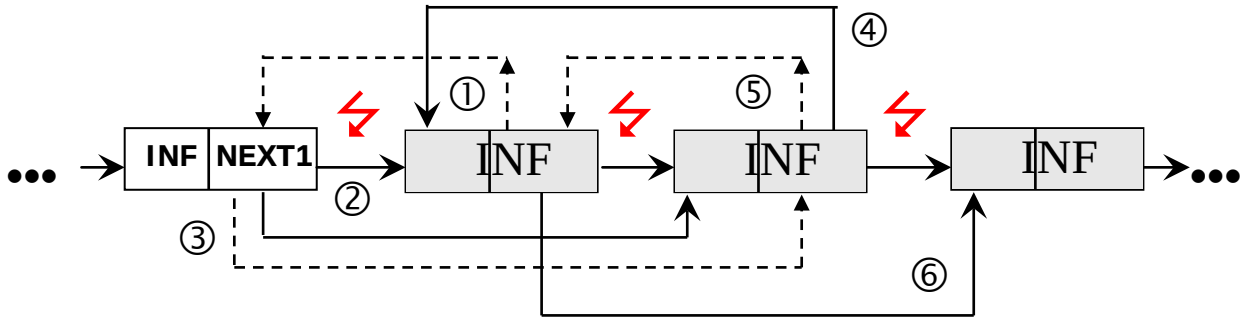


Рис. 8.101. Перестановка двох сусідніх елементів у середині списку

В алгоритмі перестановки в однозв'язному списку (рис. 8.101) виходимо з того, що відомо адресу елемента, що передує парі, у якій виробляється перестановка. У приведеному алгоритмі також не враховується випадок перестановки першого і другого елементів.

{ Перестановка двох сусідніх елементів в однозв'язному списку }

```
Procedure ExchangeSll(prev : sllptr {Указник на елемент, що
                                    передує парі, що переставляється} );

Var
  p1, p2 : sllptr; {Указники на елементи пари}
Begin
  p1:=prev^.next;   {Указник на 1-й елемент пари}
  p2:=p1^.next;     {Указник на 2-й елемент пари}
  p1^.next:=p2^.next; {1-й елемент пари тепер указує на наступний за
                       парою}
  p2^.next:=p1;     {1-й елемент пари тепер випливає за 2-им }
  prev^.next:=p2;   {2-й елемент пари тепер стає 1-им }
End; {Proc ExchangeSll}
```

8.27.17. Рішення задачі по створенню черги і реалізація операцій з нею

Тепер Ви добре себе уявляєте, що лінійні списки можна застосовувати у додатках, де непередбачені вимоги на розмір пам'яті, необхідної для збереження даних, а також велике число складних операцій над даними, особливо включень і виключень.

Стек – це такий послідовний список з перемінною довжиною, включення і виключення елементів у який виконуються тільки з однієї сторони списку, називаного вершиною стека. Існують й інші назви стека – магазин, а також черга, що функціонує за принципом LIFO (Last-In-First-Out – "останнім прийшов – першим виключається"). Прикладом стека може бути обойма у пістолеті.

Основні операції над стеком – це включення нового елемента (англійська назва push – заштовхувати) і виключення елемента зі стека (англ. pop – вискакувати).

Відомо, що стек може бути представлений як лінійний список, у якому включення елементів завжди провадиться з початки списку, а виключення – також з початку. Для представлення його Вам досить мати один показник – top, що завжди вказує на останній записаний у стек елемент. У вихідному стані (при порожньому стеці) покажчик top – порожній. Головні процедури StackPush і StackPop зводяться відповідно до включення елемента у початок списку і виключення з початку ж списку.

Зверніть увагу на те, що при включенні елемента для нього виділяється пам'ять, а при виключенні – звільняється. Перед включенням елемента перевіряється доступний обсяг пам'яті, і якщо він не дозволяє виділити пам'ять для нового елемента, стек вважається заповненим. При очищенні стека послідовно проглядається весь список і знищуються його елементи. При списковому представленні стека виявляється непросто визначити розмір стека. Ця операція могла б зажадати від Вас перебору всього списку для підрахунку числа елементів. Щоб уникнути послідовного перебору всього списку Ви повинні ввести додаткову перемінну stsize, що відбиває поточне число елементів у стеці і коректується при кожнім включенні/виключенні:

```
{Стек на 1-зв'язному лінійному списку}
unit Stack;
Interface
type data = ...; {Елементи можуть мати будь-як тип}
Procedure StackInit;
Procedure StackClr;
Function StackPush(a : data) : boolean;
Function StackPop(Var a : data) : boolean;
Function StackSize : integer;

Implementation
type stptr = ^stit; {Показник на елемент списку}
```

```
     stit = record {Елемент списку}
       inf : data; {Дані}
       next: stptr; {Покажчик на наступний елемент}
       end;
Var top : stptr; {Покажчик на вершину стека}
    stsize : longint; {Розмір стека}
{** Ініціалізація – список порожній }
Procedure StackInit;
  begin   top:=nil; stsize:=0;  end; {Proc StackInit}
{**Очищення – звільнення всієї пам'яті}
Procedure StackClr;
 var x : stptr;
  begin {Перебір елементів до кінця списку і їхнє знищення}
  while top<>nil do
    begin  x:=top; top:=top^.next; Dispose(x);  end;
  stsize:=0;
end; {Proc StackClr}
Function StackPush(a: data) : boolean; {Занесення в стек}
 var x : stptr;
  begin   {Якщо немає більше вільної пам'яті – відмовлення}
  if MaxAvail < SizeOf(stit) then StackPush:=false
  else {Виділення пам'яті для елементу і заповнення інф. частини}
    begin  New(x); x^.inf:=a;
                {Новий елемент міститься у голові списку}
      x^.next:=top; top:=x;
      stsize:=stsize+1; {Корекція розміру}
      StackPush:=true;
    end;
end; {Func StackPush}
Function StackPop(var a: data) : boolean; {Вибірка зі стека}
 var x : stptr;
  begin
 {Список порожній – стек порожній}
  if top=nil then StackPop:=false
  else begin
    a:=top^.inf; {Вибірка інформації з 1-го елементу списку}
    {1-й елемент виключається зі списку, звільняється пам'ять}
    x:=top; top:=top^.next; Dispose(top);
    stsize:=stsize-1; {Корекція розміру}
    StackPop:=true;
end;  end; { StackPop }
Function StackSize : integer; {Визначення розміру стека}
```

```
  begin   StackSize:=stsize;   end; {Func StackSize}
END.
```

Програмний приклад для організація на однозв'язному лінійному списку черги **FIFO** розробіть самостійно. Для лінійного списку, що представляє чергу, Вам необхідно буде зберігати: `top` – указник на перший елемент списку, і `bottom` – указник на останній елемент.

Вправи

1. Для яких цілей використовується оператор `With`?

2. У часовій майстерні всі замовлення, що надходять, реєструються в комп'ютері. Про кожне замовлення зберігається наступна інформація:
   – номер замовлення;
   – прізвище замовника;
   – тип годинника (механічні, електронно-механічні, електронні);
   – марка годинника;
   – термін виконання і вартість замовлення.

Годинники різних типів ремонтуються різними майстрами. Видати завдання кожному майстру в порядку терміновості замовлень, а також указати загальну суму замовлень.

3. Написати програму для обробки інформації про товари, що зберігаються на складі. Інформація містить у собі:
   – найменування товару.
   – вартість товару.
   – країна-виробник товару.
   – кінцевий термін реалізації товару.
   – кількість товару, що мається у наявності.

Програма повинна дозволяти користувачу зчитувати дані з клавіатури і заносити їх у необхідний запис, а також зчитувати дані із заданого запису й виводити на екран список товарів, з терміном реалізації, що минає (менше місяця).

4. Створити модуль для роботи з комплексними числами і програму, що його викликає, для перевірки правильності роботи модуля.

5. Додати в кінець не порожньої черги всі її елементи, розташовуючи їх у зворотному порядку.

6. Інформаційне поле елемента черги – рядок. Підрахувати кількість слів черги, що збігаються з останнім рядком.

7. Інформаційне поле елемента черги – числове. Підрахувати і вивести на екран елементи черги, не рівні нулю.

Додаток 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 приСпосібиться 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

Додаток 2

Уведення у позиційні системи числення

Д.2.1. Позиційні системи числення

Ми з Вами добре знайомі й звикли працювати з десятковою системою числення. Однак, при реалізації обчислень на комп'ютері фахівці зіштовхнулися з проблемою неможливості її технічної реалізації. Адже для відображення десяти стійких станів 0..9 вимагаються відповідні технічні рішення. Їх поки в природі не існує! Тому в комп'ютері повсюдно використовується тільки двійкова (рус. – двоичная) система числення. Це й зрозуміло, рішення лежать на поверхні: дві цифри “0” і “1” відобразити значно легше. Електричне реле може бути включено чи виключено, конденсатор – заряджений чи не заряджений, магнітний домен – поляризований чи ні, тригер знаходиться в стані “0” чи “1” і т.д. До речі, тригер – це основа конструкції всіх комп'ютерів.

Тригер – це послідовна електронна схема з двома станами, кожен з яких, за певних умов на входах схеми, підтримується постійним (тобто стабільним). Кожному з цих станів ставиться у відповідність логічне значення, що «зберігає» тригер. Таким чином, сукупний стан послідовної схеми, запам'ятовуючі властивості якої реалізовані на тригерах, являє собою просто комбінацію станів цих тригерів.

Тому, при введенні даних у комп’ютер у виді десяткових чисел вони автоматично перетворюються в їхній двійковий еквівалент, щоб їх було зручно представляти за допомогою тригерів. Разом із тим, користувачам, яким приходиться працювати з мовами програмування, а особливо з мовою Асемблера, необхідно добре уявляти собі процеси перетворення чисел з одних систем числення в іншу. Хоча двійкова система й забезпечує точне представлення чисел у пам'яті, проте з послідовністю тільки нулів та одиниць працювати дуже важко. Окрім того, зростає ймовірність робити помилки, оскільки при наборі числа виду “10110101” надзвичайно важко відстежити помилку при введенні.

Багато років тому програмісти переконалися, що у більшості випадків їм приходилося працювати не з окремими бітами, а з групами бітів. Перші мікропроцесори були 4-бітовими пристроями (вони обробляли по 4 біти за один прийом). Тому логічною альтернативою двійкової системі виявилася система, що оперувала четвірками бітів.

Як Вам відомо, чотирма бітами можна представити двійкове значення, від 0000 до 1111 (що еквівалентно десятковим значенням від 0 до 15), тобто усього 16 можливих комбінацій. Якщо в системі числення повинні бути позначені всі ці комбінації, то вона повинна мати 16 цифр. Така система числення називається шістнадцяткова (рос. – шестнадцатиричная). Із шістнадцяти цифр цієї системи числення перші 10 одержали позначення від 0 до 9 (десяткові значення від 0 до 9), а останні шість – від A до F (десяткові значення від 10 до 15).

Необхідно уявляти собі, що звичайна десяткова система – це лише одна з багатьох ***позиційних систем числення***. У цих системах використовується кінцевий набір різних символів. Кожен символ називається ***цифрою*** і позначає визначену кількість одиниць. Число різних символів у наборі називається ***основою*** системи числення. Якщо значення, що виражається, більше значення максимальної цифри в системі числення, то для його вираження використовується послідовний запис цифр цієї системи, який і ***утворює число***. У залежності від конкретної позиції, яку кожна з цифр займає в числі, їй ставиться у відповідність ваговий множник, який дорівнює основі системи.

Розглянемо звичну для нас десяткову систему. У ній тільки 10 різних цифрових символів: 0, 1, ... 9. Тоді, наприклад, число 536,4 з урахуванням позиційності розташування цифр, виражається наступним поліномом:

$$5\times10^{2}+3\times10^{1}+6\times10^{0}+4\times10^{-1}.$$

Тут цифра 5 входить з вагою 100, цифра 3 – з вагою 10, цифра 6 – з вагою 1, а цифра 4 – з вагою 0,1.

Тоді в узагальненому виді, для будь-якої позиційної системи числення по деякій основі число може бути записано:

$$N=d_{n-1}d_{n-2}\ldots d_1d_0d_{-1}d_{-2}\ldots d_{-m}\ . \qquad (2.1)$$

Виразу (1) буде відповідати поліном:

$$\boldsymbol{N=d_{n-1}b^{n-1}+d_{n-2}b^{n-2}+\ldots+d_{-m}b^{-m}\ .} \qquad (2.2)$$

У цій загальній формі d_i – це цифри, які лежать у діапазоні $0\le d_i<b$; n – число цифр лівіше ***розділової***, чи ***позиційної***, крапки (у деяких випадках замість крапки використовується кома); m – число цифр правіше крапки, а b – основа системи числення. У таблиці 2.1 перелічені системи числення, що найбільш використовуються. Як правило, у системах з основою, меншою 10, як цифрові символи використовуються відповідні перші цифри десяткової системи; для систем же з основою, яка більше 10, використовуються десяткові цифри з додаванням перших букв латинського алфавіту. У таблиці 2.1 зазначена також відносна упорядкованість цифр у системах.

Таблиця 2.1

Системи числення

Основа	Система числення	Цифрові символи
2	двійкова	0,1
3	трійкова	0,1,2
4	чотвіркова	0,1,2,3
5	п'ятіркова	0,1,2,3,4
8	вісімкова	0,1,2,3,4,5,6,7
10	десяткова	0,1,2,3,4,5,6,7,8,9
12	дванадцяткова	0,1,2,3,4,5,6,7,8,9,A,B
16	шістнадцяткова	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Зі згаданих систем особливий інтерес при вивченні обчислювальної техніки становлять саме три системи – це ***двійкова***, ***вісімкова*** (рос. –

восьмеричная) і ***шістнадцяткова.*** Запис деяких дробових чисел у цих системах виглядає у такий спосіб:

$$\mathbf{1011{,}101_2}$$
$$372{,}46_8$$
$$C65F{,}B3_{16}$$

За згодою десятковий індекс, що супроводжує число, указує ***основу*** системи числення. Індекс опускається, коли значення основи ясно з контексту.

Як і в десятковій системі, число представлене сукупністю виписаних поруч цифр (табл. 2.2). Дробова і ціла частини розташовуються відповідно праворуч і ліворуч від розділової коми. У випадку двійкової системи, цифри 0 і 1 називають ***бітами*** як скорочення від *binary digits* (двійкові цифри).

Таблиця 2.2

Перші 32 числа у двійковій, вісімковій і шістнадцятковій системах та їхні десяткові еквіваленти

Десяткові	Двійкові	Вісімкові	Шістнадцяткові
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. Перетворення чисел з однієї системи числення в іншу

Будь-яке число може бути інтерпретовано відповідним поліномом по основі обраної системи числення (двійковій, вісімковій або шістнадцятковій і т.д. системах). При роботі з комп'ютерами часто доводиться переводити числа з однієї системи числення в іншу. Тоді вихідне число і переведене в нову систему числення – можна вважати еквівалентними представленнями однієї й тієї ж величини в різних системах числення.

Д.2.2.1. Переклад у десяткову систему чисел з не десяткової системи

Розглянемо етапи перетворення числа з не десяткової позиційної системи в еквівалентну десяткову форму. Таке перетворення легко виходить простим обчисленням значення полінома, що відповідає числу. Це обчислення можна, наприклад, виконати у такий спосіб:

1. Записуємо число у виді полінома

$$d_{n-1}\,d_{n-2}\ldots d_0 d_{-1}\ldots d_{-m}=d_{n-1}b^{n-1}+d_{n-2}b^{n-2}+\ldots+d_0b^0+d_{-1}b^{-1}+\ldots+d_{-m}b^{-m},$$

де b – основа системи, виражена в десятковій формі, а d – цифри вихідної системи числення. Для тих систем, де цифри представляються буквами, останні при обчисленні заміняються на десяткові еквіваленти, наприклад A=10, B=11, C=12 і т.д.

2. Обчислюємо значення полінома, користуючись десятковою системою числення.

Для ілюстрації перекладу з двійкової системи в десяткову систему розглянемо двійкове число 1110,12. Записуючи його у виді полінома по ступіням основи 2, одержимо:

$$\begin{aligned}1110{,}1_2&=1\times2^3+1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}\\&=1\times8+1\times4+1\times2+0\times1+1\times0{,}5\\&=8+4+2+0+0{,}5\\&=14{,}5_{10}\end{aligned}$$

Таким чином, 14,5 є десятковим еквівалентом двійкового числа 1110,1.

Як інший приклад, перетворимо у десяткову систему шістнадцяткове 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. Переклад з десяткової системи в будь-яку позиційну систему

Переклад десяткового числа в еквівалентну форму в іншій системі числення більш складний. У процесі перетворення приходиться порізно трансформувати цілу і дробову частини даного числа.

Розглянемо спочатку перетворення цілого десяткового N_I у систему числення з основою b (b – ціле позитивне число). Оскільки число в системі з основою b можна записати у виді полінома по ступенях b, з відповідними коефіцієнтами, ми одержуємо:

$$\begin{aligned} N_I &= d_{n-1}b^{n-1} + d_{n-2}b^{n-2} + \ldots + d_1 b^1 + d_0 b^0 \\ &= d_{n-1}b^{n-1} + d_{n-2}b^{n-2} + \ldots + d_1 b^1 + d_0 \end{aligned} \qquad (2.3)$$

Тепер потрібно з (2.3) знайти цифри d_{n-1} ,…,d_1, d_0 , які задовольнять записаному рівнянню. Для цього розділимо обидві частини виразу на b. Одержимо цілу частку (2.4):

$$N' = d_{n-1}b^{n-2} + \ldots d_2 b^1 + d_1 b^0 \qquad (2.4)$$

і залишок:

$$\text{Залишок}\left(\frac{N_I}{b}\right) = d_0 \qquad (2.5)$$

Таким чином, залишок дорівнює молодшій цифрі числа в системі числення з основою b, тобто d_0 . У результаті ділення, в залишку може виявитися більш однієї десяткової цифри, якщо b більше 10. Однак, оскільки залишок завжди менше b, то його значення буде відповідати цифрі d_0.

Якщо процес ділення повторити для цілої частки N', ми одержимо знову цілу частку

$$N'' = d_{n-1}b^{n-3} + \ldots + d_2 b^0 \qquad (2.6)$$

і залишок:

$$\text{Залишок}\left(\frac{N'_I}{b}\right) = d_1 \qquad (2.7)$$

У цьому випадку залишок відповідає наступній праворуч цифрі числа з основою системи b. Легко бачити, що, повторюючи описаний процес аж до нульової частки, ми одержимо всі цифри d_i рівняння (2.3) для N_I. Зверніть увагу на те, що залишок слідує щораз представляти цифрою в системі числення з основою b. Очевидно, що процес завершиться після кінцевого числа кроків.

Розберемо описану процедуру на прикладі перекладу десяткового числа 52 в еквівалентну двійкову форму. Обчислення проводяться багаторазовим діленням на 2:

Залишок

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

Отже, $52_{10}= d_5\ d_4\ d_3\ d_2\ d_1\ d_0 =110100_2$.

Як другий приклад, розглянемо переклад десяткового числа 58 506 у шістнадцяткову систему. Послідовні ділення на 16 дають:

Ділення	Залишок	Значення залишку в шістнадцятковій системі числення
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		

Отже, $58\ 506_{10}= d_3\ d_2\ d_1\ d_0 = E48A_{16}$.

Процедура перекладу правильного десяткового дробу в систему числення з основою b повинна бути декілька іншою. Позначимо через N_F десятковий дріб, що відповідає поліному (2.8):

$$d_{-1}b^{-1} +d_{-2}b^{-2}+\ldots+d_{-m}b^{-m} \qquad (2.8)$$

де d_{-1} , d_{-2} , …, d_{-m} – цифри, які потрібно визначити. Оскільки поліном і N_F позначають ту саму величину, то має місце рівність (2.9):

$$N_F = d_{-1}b^{-1} +d_{-2}b^{-2}+\ldots+d_{-m}b^{-m} \qquad (2.9)$$

Перемножуючи обидві частини рівності (2.9) на b, одержимо (2.10):

$$\begin{aligned} b\,N_F &= d_{-1}b^{0} +d_{-2}b^{-1}+\ldots+d_{-m}b^{-m+1} \\ &= d_{-1} +d_{-2}b^{-1}+\ldots+d_{-m}b^{-m+1} \\ &= d_{-1}+N'_F \end{aligned} \qquad (2.10)$$

Добуток складається з цілої частини d_{-1} і дробової частини N'. Ціла частина еквівалентна старшій цифрі вихідного дробу в системі числення з основою b. Як і раніше, легко бачити, що ціла частина, яка відповідає d_{-1i}, лежить у діапазоні від 0 до $b-1$, і, отже, для систем з основою, більшою 10, ми повинні у відповідних випадках як цифри брати букви.

Якщо провести ті ж дії над дробовою частиною результату $b\ N_F$, тобто помножити його на b, то можна буде визначити наступну цифру розкладання дробу N_F.(2.11). А саме, оскільки

$$
\begin{aligned}
b\,N'_F &= d_{-2}b^0 + d_{-3}b^{-1} + \ldots + d_{-m}b^{-m+2} \\
&= d_{-2} + d_{-3}b^{-1} + \ldots + d_{-m}b^{-m+2} \\
&= d_{-2} + N''_F
\end{aligned}
\qquad (2.11)
$$

то ціла частина добутку відповідає d_{-2}. Очевидно, що неодноразово повторюючи описаний процес, ми зможемо визначити наступні цифри числа в системі по основі *b*. Процес припиняється, коли виходить нульова дробова частина. Однак на відміну від процесу перетворення цілих чисел, що завжди закінчується через кінцеве число кроків, процес перетворення десяткового дробу може бути нескінченним. Іншими словами, представлення десяткового дробу з кінцевим числом цифр може мати нескінченне число цифр у системі числення з іншою підставою. Тому, у будь-якому випадку, процес перетворення зупиняють при досягненні необхідної точності.

Приведемо два приклади перетворення десяткових дробів. Спочатку розглянемо переклад числа 0,6875 у двійкову форму:

Добуток	Значення розряду
$2 \times 0{,}6875 = 1{,}3750$	$d_{-1} = 1$
$2 \times 0{,}375 = 0{,}750$	$d_{-2} = 0$
$2 \times 0{,}75 = 1{,}50$	$d_{-3} = 1$
$2 \times 0{,}5 = 1{,}0$	$d_{-4} = 1$

У результаті ми одержуємо $0{,}6875_{10} = 0.d_{-1}\,d_{-2}\,d_{-3}\,d_{-4} = 0{,}1011_2$.

Тепер переведемо десятковий дріб 0,8435 у шістнадцяткову систему. Потрібно виконати наступний ланцюжок множень:

Добуток	Значення розряду
$16 \times 0{,}8435 = 13{,}496$	$d_{-1} = D$
$16 \times 0{,}496 = 7{,}936$	$d_{-2} = 7$
$16 \times 0{,}936 = 14{,}976$	$d_{-3} = E$
$16 \times 0{,}976 = 15{,}616$	$d_{-4} = F$

Зупинивши процес на цьому кроці, ми одержимо $0{,}8435_{10} = 0.d_{-1}\,d_{-2}\,d_{-3}\,d_{-4} = 0{,}D7EF\ldots_{16}$...

На цьому прикладі видно, що процес перетворення цього дробу – нескінченний, оскільки третя цифра дробової частини на всіх кроках дорівнює 6.

Для змішаних десяткових чисел ціла і дробова частини обробляються порізно. Ціла частина перетвориться послідовними розподілами, а дробова – послідовними множеннями. Змішане число, що виходить у результаті, записується у виді цих двох частин розділених крапкою.

Д.2.3. Виконання операцій у двійковій системі числення

Двійкова система числення є основною системою представлення інформації в пам'яті комп'ютера. У цій системі числення використовуються цифри: 0, 1. Над числами у двійковій системі числення можна виконувати всі арифметичні дії.

При цьому використовуються наступні операції (табл.. 2.3):

Таблиця 2.3

Операції у двійковій системі числення

Додавання	Вирахування	Множення
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

Двійкова система числення має такі ж властивості, що і десяткова, тільки для представлення чисел використовується не 10 цифр, а усього дві.

Для кодування числа, що бере участь у обчисленнях, використовується спеціальна система правил перекладу з десяткової системи числення в двійкову. У результаті число буде записано двійковим кодом, тобто представлено різним сполученням усього двох цифр – “0” і “1”.

Для порівняння розглянемо число 45 для двох варіантів кодування. При використанні в тексті, це число потребує для свого представлення 2 байти, тому що кожна цифра буде представлена своїм кодом відповідно до таблиці ASCII. У шістнадцятковій системі код для цифри 4 буде 43, у двійковій системі – 01000011, відповідно для цифри 5 – 53 і 01010011.

Таблиця 2.4

Представлення чисел у позиційних системах

Десяткова	Шістнадцяткова	Двійкова
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. Спосіби кодування інформації

На уроках фізики при розгляді якого-небудь фізичного явища ви використовуєте формули. ***Формула*** – це свого роду математичний код.

Розглянуті приклади говорять про те, що для представлення інформації можуть використовуватися різні коди і, відповідно, треба знати визначені правила – закони запису цих кодів, тобто вміти ***кодувати*** та ***перекодовувати***.

Код – це набір умовних позначок для представлення інформації. Кодування – це процес представлення інформації у виді коду.

Як джерела інформації може виступати: людина, технічний пристрій, предмети, об'єкти неживої і живої природи. Одержувачів повідомлень може бути один чи декілька.

Д.2.4.1. Використання двійкової системи для кодування текстової інформації в ПК

Натискання клавіші на клавіатурі приводить до того, що сигнал надсилається у комп'ютер у виді двійкового числа, яке зберігається у ***кодовій таблиці***.

Кодова таблиця – це внутрішнє представлення символів у комп'ютері. В усьому світі як стандарт прийнята ***таблиця*** *ASCII (American Standard Code for Information Interchange –* ***Американський стандартний код обміну інформацією)***.

Для збереження двійкового коду одного символу у цьому стандарті виділений 1 байт, який дорівнює 8 біт. З огляду на те, що кожен біт приймає значення “0” чи “1”, кількість їхніх можливих сполучень у байті дорівнює 256.

Виходить, що за допомогою одного байта можна одержати 256 різних двійкових кодових комбінацій й відобразити з їх допомогою 256 різних символів. Ці комбінації й складають таблицю кодів ASCII.

Наприклад, ви натискаєте на клавіатурі латинську букву “S”. У цьому випадку в пам'ять комп'ютера записується код 01010011. Для виводу букви S на екран комп'ютера відбувається його декодування – тобто по цьому двійковому коду будується на екрані комп’ютера зображення цього символу.

Д.2.4.2. Кодування графічної інформації

Створювати і зберігати графічні об'єкти в комп'ютері можна двома Спосібами – як растрове чи як векторне зображення. Для кожного типу цих зображень використовується свій спосіб кодування.

Векторне зображення являє собою графічний об'єкт, що складається з елементарних відрізків і дуг. Положення цих елементарних об'єктів визначається координатами крапок на екрані комп’ютера і довжиною радіуса.

Для кожної лінії вказується її тип (суцільна, пунктирна, штрих-пунктирна), товщина і колір. Інформація про векторне зображення кодується як звичайна буквено-цифрова й обробляється спеціальними програмами.

Растрове зображення являє собою сукупність крапок, використовуваних для його відображення на екрані монітора. Обсяг растрового зображення визначається множенням кількості крапок на інформаційний обсяг однієї крапки, яка залежить від кількості можливих кольорів. Для чорно-білого зображення інформаційний обсяг однієї крапки дорівнює 1 биту, тому що вона може бути або чорною, або білою, що можна закодувати двома цифрами – “0” чи “1”. Розглянемо, скільки буде потрібно біт для відображення кольорової крапки: для 8 кольорів – 3 біти; для 16 кольорів – 4 біти; для 256 кольорів – 8 бітів (це 1 байт). У таблиці 2.5 показано кодування колірної палітри з 16 кольорів. Різні кольори та їхні відтінки виходять за рахунок наявності чи відсутності трьох основних кольорів (червоного, синього, зеленого) та їхньої яскравості. Кожна крапка на екрані кодується за допомогою 4 бітів.

Таблиця 2.5

Кодування 16-кольорової палітри.

Колір	Яскравість	Червоний	Зелений	Синій	Значення у системі числення	
					10-кова	16-кова
Чорний	0	0	0	0	0	0
Синій	0	0	0	1	1	1
Зелений	0	0	1	0	2	2
Блакитний	0	0	1	1	3	3
Червоний	0	1	0	0	4	4
Фіолетовий	0	1	0	1	5	5
Коричневий	0	1	1	0	6	6
Білий	0	1	1	1	7	7
Сірий	1	0	0	0	8	8
Світло-синій	1	0	0	1	9	9
Світло-зелений	1	0	1	0	10	A
Світло-блакитний	1	0	1	1	11	B
Світло-червоний	1	1	0	0	12	C
Світло-фіолетовий	1	1	0	1	13	D
Жовтий	1	1	1	0	14	E
Яскраво-білий	1	1	1	1	15	F

Додаток 3

Характеристики мов програмування

Назва мови программування (МП)	Рік	Тип мови	Автор (и) розробки	Географія	Організація	Стандарт
Фортран (FORTRAN)	1954	A	Джон Бекус	Америка	IBM	ISO 1539:1997
Ліспп (LISP)	1958	F	Джон Маккарті	Америка	MIT	–
Алгол-60 (Algol 60)	1960	A	Пітер Наур+	Міжнар.	IFIP	–
Кобол (COBOL)	1960	A	+	Міжнар.	CODASYL Committee	ISO 1989:1985
Сімула (Simula)	1962	B	Крістен Нігаард+	Європа	–	–
Бейсік (BASIC)	1963	A	Томас Курт і Джон Кемені+	Америка	Dartmouth College	ISO 10279:1991
ПЛ/1 (PL/I)	1964	A	Джордж Радін	Америка	IBM	ISO 6160:1979
Алгол-68 (Algol 68)	1968	A	Аад ван Вайнгартен+	Міжнар.	IFIP	–
Паскаль (Pascal)	**1970**	**C**	**Ніклаус Вірт**	**Європа**	**ETH**	**ISO 7185:1990**
Форт (FORTH)	1970	A*	Чарльз Мур	Америка	Mohasco Industries	ISO 15145:1997
Сі (C)	1972	C*	Денніс Рітчі	Америка	AT&T Bell Labs	ISO 9899:1999
Smalltalk	1972	B/OO	Алан Кей	Америка	Xerox PARC	–
Пролог (Prolog)	1973	E	Алан Кольмеро+	Європа	Univ. of Aix-Marseille	ISO 13211:1995
Ада (Ada)	1980	H*	Джин Ішбіа+	Америка	CII Honeywell	ISO 8652:1995
Си++	1980/ 1984	H* –/OO	Бьорн Страуструп	Америка	AT&T Bell Labs	ISO 14882:1998
Turbo Pascal	**1983**	**–/–**	**Філіпп Кан, Андерс Хейльсберг (Anders Hejlsberg)**	**Америка**	**Borland International**	**–**
Perl	1986	ЯС/OO	Ларрі Вол (Larry Wall)	Міжнар.	–	–
HTML	1989	I	Тім Бернерс-Лі	Европа	CERN	–
Python	1990 (1999 v 2.0)	ЯС/OO	Гвідо ван Россум (Guido van Rossum)	Голландія/ Америка	–	–
Visual Basic	1991	ЯС/OO	–	Америка	Microsoft	–
Delphi/Object Pascal	**1995**	**–/OO**	**Андерс Хейльсберг**	**Америка**	**Borland International**	**–**
Java	1995	H ЯС/OO	Патрік Нотон і Джеймс Гослінг	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# (Сі Шарп)	**2000**	**H* – /OO/D**	**Андерс Хейльсберг+**	**Америка**	**Microsoft**	**–**

ЕКСПЕРИМЕНТАЛЬНІ ТА ПРОМИСЛОВІ МОВИ

Назва мови програмування (МП)	Рік	Тип мови	Автор (и) розробки	Геогра-фія	Організація	Стандарт
АПЛ (APL)	1957	I	Кеннет Айверсон	Америка	Harvard Univ.	ISO 8485:1989
Снобол (Snobol)	1962	I	Ральф Грісуолд	Америка	AT&T Bell Labs	–
Сетл (SETL)	1969	I	Джек Шварц	Америка	IBM	–
Паралельний Паскаль (Concurrent Pascal)	**1974**	**G**	**Пер Брінч Хансен**	**Америка**	**CIT**	**–**
CLU	1974	D	Барбара Лісков	Америка	MIT	–
Scheme	1975	F	Гай Стіл+	Америка	MIT	–
Mesa	1976	D*	Дж. Мічел+	Америка	Xerox PARC	–
Icon	1977	I	Ральф Грісуолд	Америка	AT&T Bell Labs	–
Модула-2 (Modula-2)	**1979**	**D***	**Ніклаус Вірт**	**Європа**	**ETH**	**ISO 10514:1996**
Оккам (Occam)	1982	G*	Дэвід Мэй+	Європа	Inmos	–
Cedar	1983	H*	Батлер Лэмпсон+	Америка	Xerox PARC	–
Common Lisp	1984	F	Гай Стіл+	Америка	MIT	–
Objective C	1986	H*	Брэд Кокс	Америка	Productivity Products	–
Эйфель (Eiffel)	1986	D*	Бертран Мейер	Європа	ISE	–
Оберон (Oberon)	**1988**	**D***	**Ніклаус Вірт**	**Європа**	**ETH**	**–**
Модула-3 (Modula-3)	1988	H*	Білл Калсов+	Америка	DEC SRC	–
Оберон-2 (Oberon-2)	**1991**	**D***	**Ханспетер Мёссенбёк+**	**Європа**	**ETH**	**–**
Limbo	1996	D*	Денніс Рітчі	Америка	Bell Labs (Lucent)	–
Component Pascal	**1997**	**D***	**Куно Пфістер+**	**Європа**	**Oberon microsystems**	**–**

УМОВНІ ПОЗНАЧЕННЯ

Выділені мови Паскаль-сімейства

Види (парадігми)

A - процедурне програмування;
B – об'єктно-орієнтоване програмування;
C - структурне програмування;
D - модульне (компонентне) програмування;
E - логічне (реляційне) програмування;
F - функціональне програмування;
G - рівнобіжне програмування;
H - гібрид (суміш парадигм; B+C+D+G);
I - спеціалізовані мови;
ЯС – мова скриптів;
ОО – об'єктно-орієнтована мова.

Скорочення

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
* Підтримка системного програмування.
+ Декілька авторів**.**

Додаток 4

Рівні розвитку мереж у інформаційно-комп'ютерних технологіях

Мережа

Перш, ніж приступити до розгляду міжмережевої взаємодії, уточнимо, що розуміється під терміном "***мережа***". Цей термін може вживатися у широкому змісті (мережа – це сукупність пов'язаних між собою комп'ютерів) і у вузькому змісті (мережа – це сукупність комп'ютерів, з'єднаних між собою відповідно до однієї зі стандартних типових топологій (шина, зірка, кільце) й пакетів, що використовують для передачі (пакет – це один із протоколів канального рівня, визначений для однієї з топологій).

Кожна мережа має свій IP-номер, що використовується на мережному рівні при виконанні маршрутизації. Коли дві чи більш мережі організують спільну транспортну службу, то такий режим взаємодії звичайно називають ***міжмережевою взаємодією (internetworking)***. Для позначення складеної (рос.-составной) мережі в англомовній літературі часто також використовуються терміни ***інтермережа*** (Internetwork чи Internet). Інтермережа забезпечує тільки передачу пакетів і не займається розглядом їхнього змісту. *Internetwork* є об'єднанням окремих мереж, з'єднаних проміжними мережними пристроями і яке працює як єдина велика мережа (рис. П.4.1).

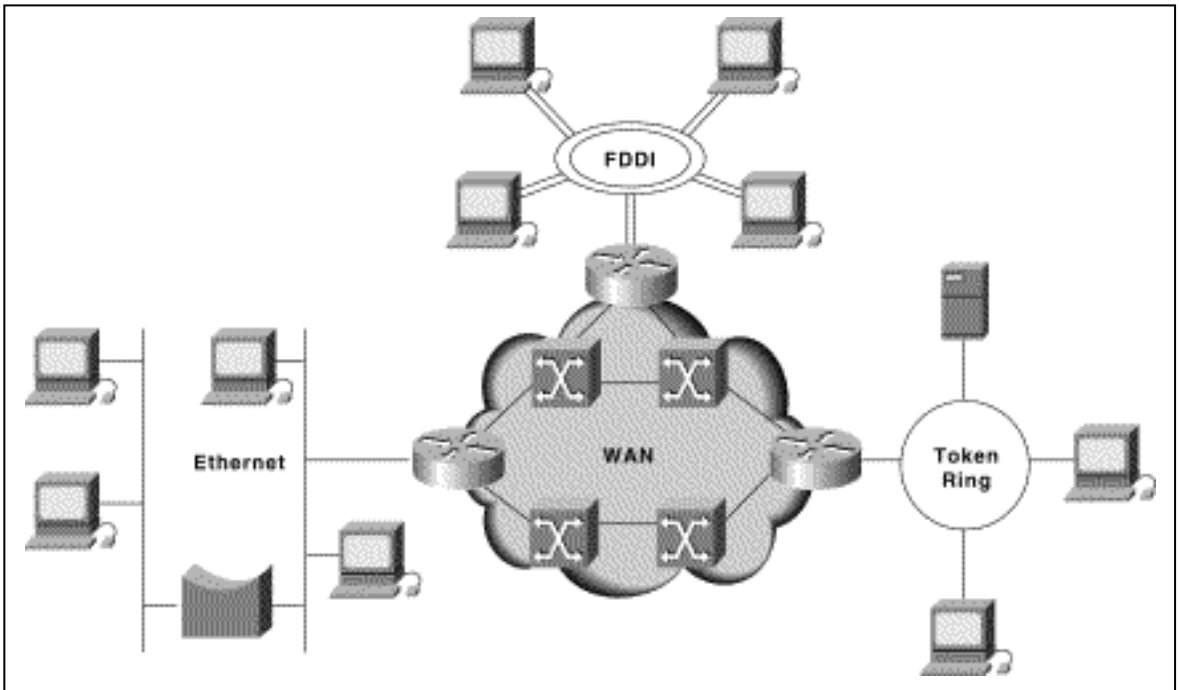


Рис. Д.4.1. Приклад взаємодії різних за складом мереж
у єдиному комплексі інтермережі

Об'єднання мереж (*Internetworking*) відноситься до галузей, продуктів й процедур, що відповідають вимогам створення й адміністрування мережних комплексів.

Виникнення інтермереж відбувалося по своєї більшості випадковим (рос.-. случайным) образом. При цьому, придбані комп'ютери і ОС, відповідали індивідуальним потребам груп користувачів. Мережі відділів будувалися для рішення конкретних задач груп співробітників. Наприклад, інженерний відділ міг обрати робочі станції SPARC фірми Sun Microsystems, що були з'єднані мережею Ethernet, тому що їм були потрібні додатки (рос.-приложения), які б працювали тільки у середовищі UNIX. Розподіл файлів при цьому реалізовувався за допомогою протоколів TCP/IP і NFS. У відділі продаж тієї ж самої організації вже могли бути придбані комп'ютери PS/2, встановлена мережа Token Ring і операційна система NetWare для рішення їх власних задач: ведення бази даних про клієнтів, підготовки листів, розробки комерційних пропозицій (рос.-предложений). Потім у рекламному відділі були обрані комп'ютери Macintosh, оскільки вони найкращім образом підходять для створення презентаційних матеріалів. Macintosh'и з'єднувалися за допомогою LocalTalk, а файли і принтери розподілялися з використанням AppleTalk. Відділ, що відповідає за автоматизацію підприємства, повинен ***інтегрувати*** усі ці цілком несумісні системи у єдиний прозорий організм.

Гетерогенність.

Таким чином, тільки невелика кількість мереж має ***однорідність (гомогенність)*** програмного й апаратного забезпечення. Однорідними частіше є мережі, що складаються з невеликої кількості компонентів від одного виробника.

Нормою сьогоднішнього дня є мережі ***неоднорідні (гетерогенні)***, які складаються з різних робочих станцій (Intel, Sun, IBM, Hewlett-Packard, Dell, COMPAQ), операційних систем (Windows, Unix, Linux, Sun, Macintosh), та різноманітних додатків, а для реалізації взаємодії між комп'ютерами використовують різні протоколи. Розмаїтість усіх компонентів, з яких будується мережа, породжує ще більшу розмаїтість структур мереж, що виходять з цих компонентів. Структурні і організаційні об'єднання комп'ютерів у єдиний комплекс мають наступні назви.

Local Area Network (LAN) – локальна мережа, з'єднаних між собою робочих станцій, які спільно використовують ресурси чи процесора сервера в межах невеликого географічного простору. Призначена для спільного використання ресурсів файлів-серверів для обміну файлами і повідомленнями, а також мережних принтерів. Може обслуговувати від декількох до декількох тисяч користувачів.

Wide Area Network (WAN) – фізична комунікаційна мережа, що зв'язує географічно вилучені друг від друга комп'ютери і мережні сегменти. Характеризує більш широку телекомунікаційну структуру, чим LAN. Може складатися з мереж приватних компаній, а також включати державні мережі. Обов'язково включає всі засоби передачі.

Интранет (Intranet) – є мережею, що розташована в межах підприємства. Може складатися з багатьох зв'язаних між собою локальних мереж (LAN), а

також використовувати орендовані лінії – WAN. Також може включати чи не включати з'єднання через один чи декілька шлюзів із зовнішнім Інтернетом. Основним призначенням інтранет є об'єднання інформації й обчислювальних потужностей (засобів) підприємства, забезпечення ними його працівників та накопичення стратегічного інформаційного ресурсу компанії. Інтранет може також використовуватися для забезпечення групової роботи і проведення телеконференцій.

Екстранет (Extranet) – об'єднана мережа, що використовує Інтернет-технології для з'єднання фірм і підприємств із їх постачальниками, чи клієнтами іншими фірм, пов'язаними загальними цілями. Екстранет можна представити у виді частини інтранет-мережі компанії, що зроблена доступною для інших компаній або вже є власністю декількох компаній. Загальна для них інформація доступна тільки для учасників чи комплексу і може відкриватися для доступу за особливими угодами.

Інтернет (Internet) – глобальна мережа, що з'єднує багато мільйонів комп'ютерів. Більш 100 країн поєднуються нею для обміну даними, новинами й іншою інформацією. Є децентралізованою мережею (не має одного головного комп'ютера). Кожен Інтернет комп'ютер називається ***хостом (host)*** і абсолютно незалежний від інших. Інтернет включає всі матеріальні складові, включаючи комунікації. Слід зазначити, що Інтернет не є синонімом World Wide Web (світова павутина) – це тільки один з його сервісів.

World Wide Web (WWW – ***світова павутина)*** – ***сервіс Інтерне***т, який підтримує збереження й доставку по всій мережі спеціально форматованих документів. Документи форматуются засобами скриптової мови *HTML (HyperText Markup Language),* яка підтримує посилання на інші документи (текст, графіка, аудіо-, відео і т.ін.), розташовані у будь-якій частині світу на будь-якому комп'ютері-сервері – потужному комп'ютері з програмою-сервером (цей комп'ютер повинен бути включеним у даний момент часу для забезпечення доступу, тобто функціонувати). Сервера, на яких розташовуються документи, можуть також містити інші різноманітні й численні дані, як правило в заархівованому вигляді. Ці дані можна "скачувати" на комп'ютер користувача для подальшої роботи. Документи проглядаються на комп'ютерах користувачів за допомогою додатків, які називаються браузерами. ***Браузер*** – це програма-клієнт, яка дозволяє користувачам читати HTML-документи з локального диску та Internet за допомогою сервіса WWW, а також здійснювати навігацію поміж ними. Прикладами програм-браузерів можна навести такі: MS Internet Explorer, Netscape Navigator, Opera та інші.

Слід ще раз особливо зазначити, що WWW не є синонімом Інтернет, а тільки входить до складу його ***сервісів***. З інших сервісів Інтернет можна назвати *E-mail* ***(електронна пошта)***, *FTP* ***(передача файлів)***, *Telnet* ***(телеконференції)*** та ін.

Додаток 5

Команди інтегрованого середовища розробки Turbo Pascal 7.0

Управляючі команди (“гарячі клавиші”)	
F1	-отримати справку / допомогу
F2	-зберігти поточний файл, що редагується, на диск
F3	-читати файл з диску
F4	-виконати програму до стрічки, де розташован курсор
F5	-(вкл/откл) суміщення вікон редактора і відлагоджувальника
F6	-циклічна зміна вікон (переход з вікна до вікна)
F7	-трассировка підпрограми
F8	-пооператорне виконання програми
F9	-компилювати програму
F10	-перехоід до верхнього (головного меню)
Alt–X	-вихід у DOS (кінець роботи)
Alt–0	-показати список активних вікон
Alt–F1	-показати останній екран підказки
Alt–F3	закрити (вилучити) поточне вікно
Alt–F5	-показати результати виконання програми
Alt–F9	-компілювати поточний файл
Alt–[перша буква назви меню]–викликає відповідне меню: (File, Run, …)	
Ctrl–F1	-підказка по слову
Ctrl–F2	-закінчити процес відлагодження
Ctrl–F3	-показати стек
Ctrl–F4	-обчислити вираз або зменити значення змінної
Ctrl–F5	-переміщення вікна і зміна його розмірів
Ctrl–F9	-запуск задачі на виконання
Esc	-закриття діалогового вікна
PrintScre	-друкування копії екрана на принтері
Shift–F6	-переключення активних вікон
Ctrl–Break	-переривання програми, що виконувалася
Alt–R–Enter	-запуск задачі на виконання
Shift– "клавиши стрелки"	-виділить фрагмент программы
Alt–BackSpace	-вдміна останьнього редагування

Команди роботи з блоком	
Ctrl–K B	-помітити початок блока
Ctrl–K K	-помітити кінець блока
Ctrl–K T	-помітити слово зліва від курсору
Ctrl–K P	-надрукувати блок
Ctrl–K C	-копіювати блок в позицію курсора
Ctrl–K V	-перемістити блок
Ctrl–K H	-(убрать/вновь выделить) пометку блока
Ctrl–K Y	-вилучити блок
Ctrl–K R	-читати блок з дискового файла
Ctrl–K W	-записати блок на диск
Ctrl–K I	-змістити блок вправо
Ctrl–K U	-змістити блок уліво

Команди вилучення/вставки	
Shift–Del	-перемістити блок, що виділен, у буфер Clipboard
Ctrl–Ins	-скопіювати блок, що виділен, у буфер Clipboard
Ctrl–Del	-вилучити блок, що виділено
Shift–Ins	-вставити блок з буфера Clipboard у позицію курсора
Ctrl–V	-вкл/викл. режим вставки
Ctrl–N	-вставити строку
Ctrl–Y	-вилучити строку
Ctrl–H	-стерти символ слева від курсора
Ctrl–G	- стерти символ над курсором
Ctrl–T	- стерти слово справа від курсора

Додаток 6

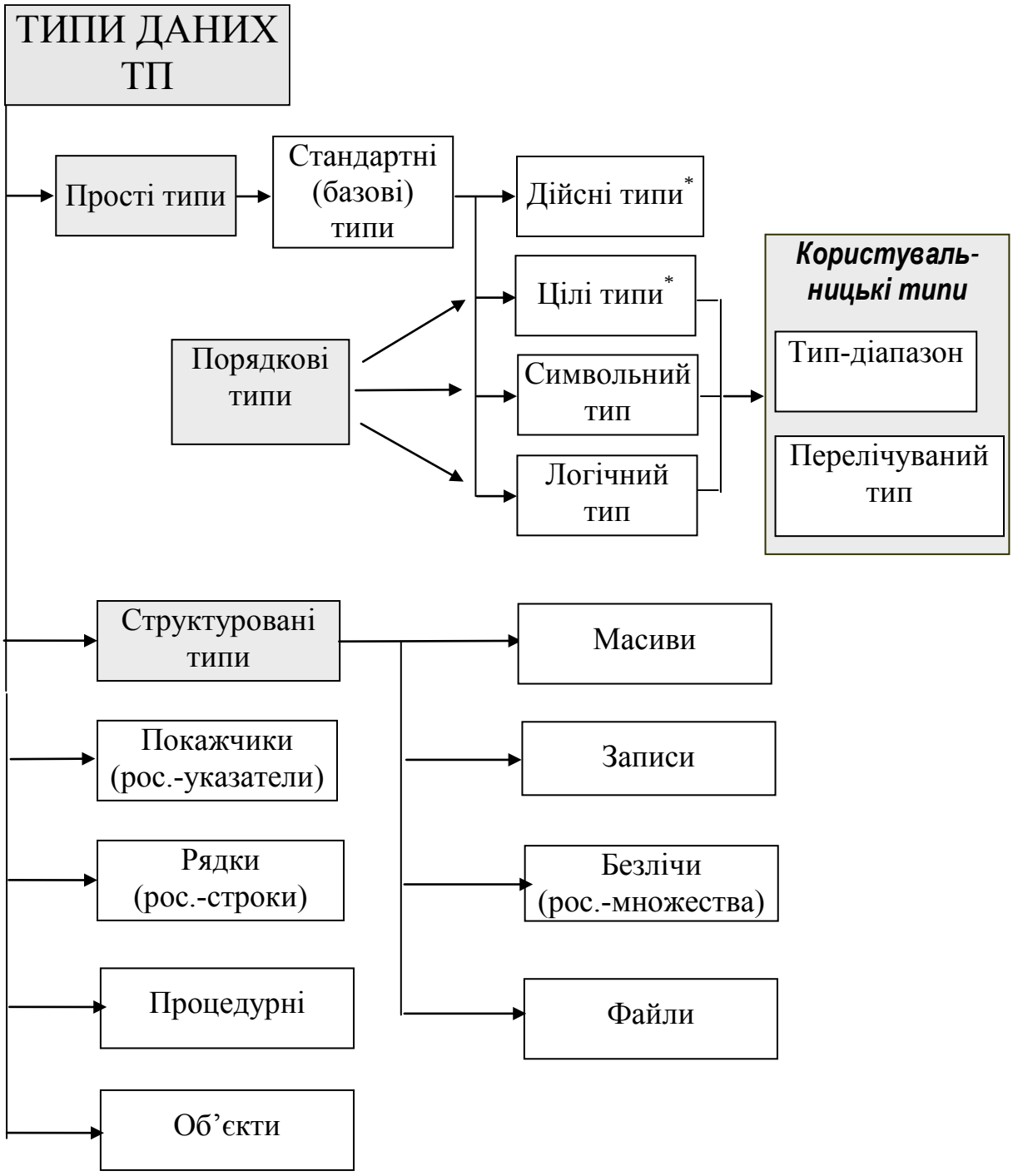
Коди ASCII (0-127)
(American Standard Code for Information Interchange)

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

Примітка:У таблиці: DEC – десятинний номер символу; CHAR – символи, які кодуються. Наприклад, символ ″4″ у таблиці має десятинний номер 52, що у двійковому еквіваленті (див. Додаток 2, підрозділ Д.2.2.2) дорівнює: 110100, а у восьмибітному еквіваленті – 00110100 (зліва додано два нулі до 8-ми бітів).

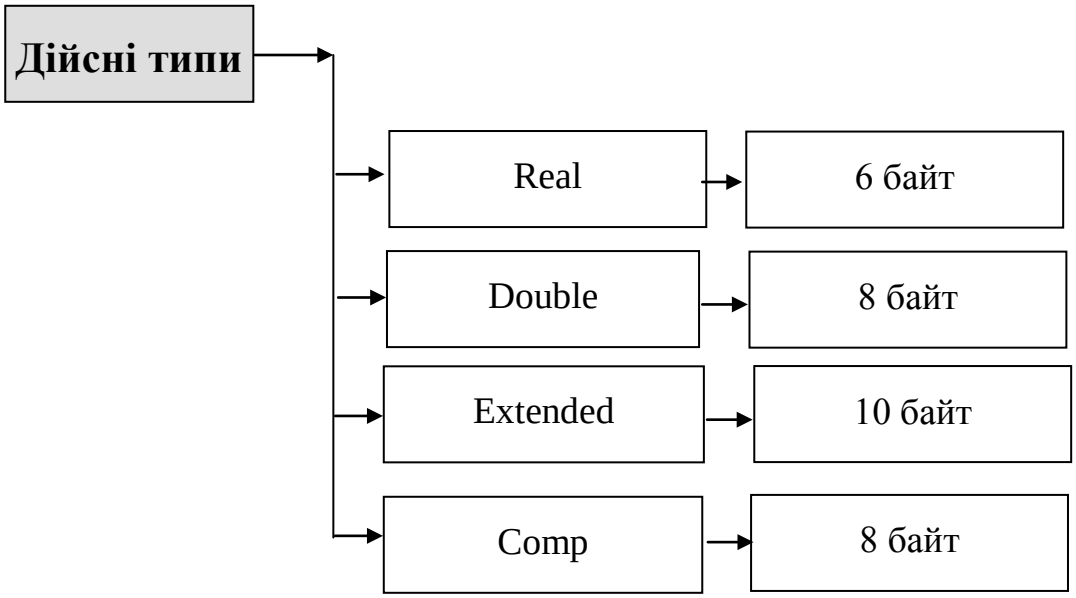
Додаток 7

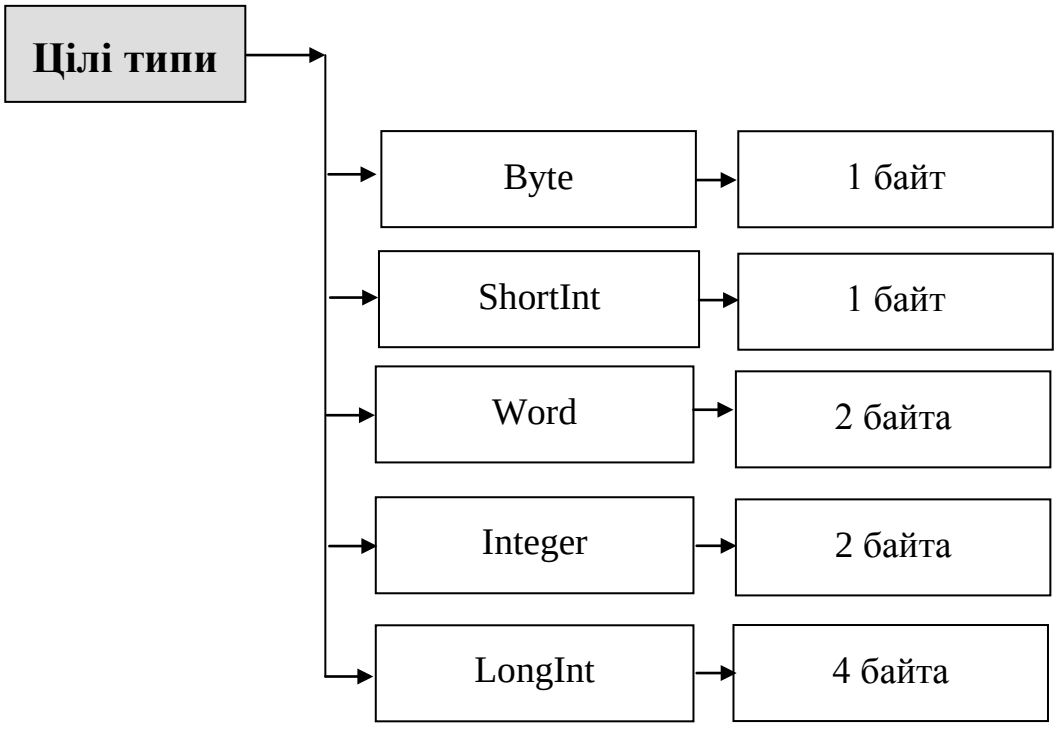
Головні типи даних Турбо Паскаля



Головні типи даних мови Турбо Паскаль. До головних, тобто базових типів відносяться чотири: Дійсні, Цілі, Символьний і Логічний. Перші два вміщують різновиди, що відрізняються величиною змінних та місцем, що даний тип займає у пам'яті (див. Діаграми нижче). Три з них (Цілі, Символьний і Логічний) є порядковими, тобто об'єднують дані, кожне з котрих завжди має фіксоване місце у послідовності (Наприклад: 1,2, ... і т.д.). З цих трьох типів будуються два користувальницькі типи Тип-диапазон та Перелічуваний тип.

Різновиди Дійсного (Real) і Цілого (Integer) типів
та кількість байтів, що вони займають у пам'яті комп'ютера.





Додаток 8

ПЕРЕЛІК ТИПОВИХ ЛАБОРАТОРНИХ РОБІТ

1. Технологія роботи у середовищі Турбо Паскаль (ТП) версії 7.0. Виконання програми з базовими типами даних (констант і змінних).

2. Вивчення загальної структури програми. Розробка простого інтерфейсу користувача програми.

3. Вирішення задач, що базуються на використанні виразів, операндів і операцій мови ТП.

4. Ініціалізація даних перед обчисленням виразів. Дійсні (`Real`) і цілі (`Integer`) типи даних. Операції і вбудовані функції роботи з цими типами даних.

5. Логічні типи даних (`Boolean`). Операції і вбудовані функції роботи з ними. Конструювання логічних виразів для формування логіки роботи програм. Використання логічних операцій та операцій відношення для запису складних умовних виразів.

6. Управляючі структури (оператори) мови ТП. Прості оператори. Складні (структурні) оператори управління виконанням алгоритмів. Складовий оператор. Оператори розгалуження алгоритмів. Умовний оператор `if`. Оператор вибору `case`.

7. Циклічні обчислювальні процеси і оператори циклів. Цикли з параметром. Оператор циклу з параметром `for`. Оператор циклу з передумовою `while`. Оператор циклу з післяумовою `repeat`. Засоби дослідження виконання дій Вашої програми за допомогою дебаггера.

8.Моделювання у циклічних обчисленнях деяких типових виразів. Особливості обчислення нескінченних (рос. – бесконечных) сум.

9. Організація ітераційних процесів за допомогою циклів `while` та `repeat`. Нескінченні множення та їх обчислення.

10. Підпрограми: процедури і функції. Формальні і фактичні параметри. Передача параметрів “за значенням” та “за адресою”.

11.Програмування задач за алгоритмами чисельних методів, що реалізуються за допомогою опису у головній програмі відповідних процедур і функцій:

– обчислення визначених інтегралів методом трапецій і Сімпсона;

– обчислення трансцендентних рівнянь методом діхотомії;

– вирішення задач інтерполяції та екстраполяції і т. д.

Література

До глави 1. Тривалий шлях до персонального комп'ютера

1. Fortune 500. Rank Company Revenues. WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL: http://www.fortune.com
2. Glossary of Terms. WEB-сайт (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: URL: http://www.kbsu.ru/%7Ebook/theory/chapter1/1_1_3.html
5. История создания Windows. WEB-сайт (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: URL: http://mii-corp.newmail.ru/ W_E_S_PC/ microsoft.htm
11. Работа и история персонального компьютера. WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL: http://mii-corp.newmail.ru/W_E_S_PC/apple.htm
12. Работа и история персонального компьютера. WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL: http://mii-corp.newmail.ru/W_E_S_PC/intel.htm
13. Работа и история персонального компьютера. WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL: http://mii-corp.newmail.ru/W_E_S_PC/ibm.htm

До глави 2. Витонченість процесу уключення ПК

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.

До глави 3. **Командна основа роботи комп'ютера**

1. Component Object Model, Part II: Programming Interface Version 0.9 (Draft) October 24, 1995 Microsoft Corporation and Digital Equipment Corporation. WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL: www.graphcomp.com/info/specs/com/comch02.htm

2. Dave Anderson. WELCOME TO THE PC TECHNOLOGY GUIDE! WEB-сайт (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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.

До глави 4. **Концепції інтерфейсу**

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 (Електронн. ресурс) / Спосіб доступу: URL: http://www.journals.ru/users/webdev (http://progs-maker.narod.ru/other.html)

9. Головач Влад. В. 5 ПРАВИЛ хорошего интерфейса. WEB-сайт (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: URL: http://www.ostu.ru/nit/staff/ag/public1.htm

15. Хэйес Фрэнк. Distributed Component Object Model. WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL: http://www.citforum.ru/programming/application/dcobjmod.shtml

16. Эндрю вэн Дам. Пользовательские интерфейсы нового поколения. WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL: http://javaworld.osp.ru/os/1997/06/34.htm

До глави 5. Еволюція мов програмування

1. Albahari Ben. A Comparative Overview of C#. WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL: http://genamics.com/developer/csharp_comparative.htm

2. Anders Hejlsberg. From Wikipedia, the free encyclopedia. WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL: http://www.wikipedia.org/wiki/Anders_Hejlsberg

3. Kalev Danny. Интервью с Бьерном Страуструпом. Будущее за мультипарадигматическим программированием. WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL:http://www.softcraft.ru/paradigm/common/siw.shtml

4. Mauny M.. Functional Programmig using Caml Light. January 1995. WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL: http://cristal.inria.fr/tutorial/index.html

5. MS Office 2000 для разработчиков. WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL:http://visual.2000.ru/develop/vb/source/mod2000.htm

6. Report on the Programming Language Haskell 98. WEB-сайт (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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.

До глави 6. Зміна методології створення програм

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 (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: URL: http://www.theserverside.com/resources/article.jsp?l=J2EE-vs-DOTNET

7. Web Services Development Tools – J2EE & .NET. WEB-сайт (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: URL: http://www.pcweek.ru/?ID=184308

32. Колесов Андрей. Visial Basic.NET — страсти накаляются // PC Week/RE, № 18. 2001. –С. 36. (WEB-сайт (Електронн. ресурс) / Спосіб доступу: URL: http://visual.2000.ru/kolesov/pcweek/2001/10424vb7.htm

33. Колесов Андрей. Преобразование объектов COM в Web-сервисы. WEB-сайт (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: 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 (Електронн. ресурс) / Спосіб доступу: URL: http://www.interface.ru/microsoft/Net/7.gif

55. Федоров Алексей. Web нового поколения — Web-сервисы. WEB-сайт (Електронн. ресурс) / Спосіб доступу: 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.

До глави 7. Мова UML та її застосування

1. Object-Oriented Programming. WEB-сайт (Електронн. ресурс) / Спосіб доступу: 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.

До глави 8. **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.

Загальнометодична література

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 e–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 (e-Образование) (см. *e-Learning*)

e-Learning (e-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 *Hyper Text 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.

❷Ячейка 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.

- B -

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.).

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:

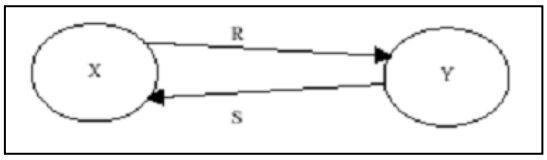


Рисунок 1. Интероперабельность двух систем

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 гипертектстовой 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 опередеенное 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)*** Запоминающее 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 ATM (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 ***f(x)***.

В этом случае говорят, что на множестве X задана функция f (а также – что переменная y есть функция переменной x, или что y зависит от x) и пишут $f : X \to Y$.

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 p-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 «e», *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 РС. –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. –250р. 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

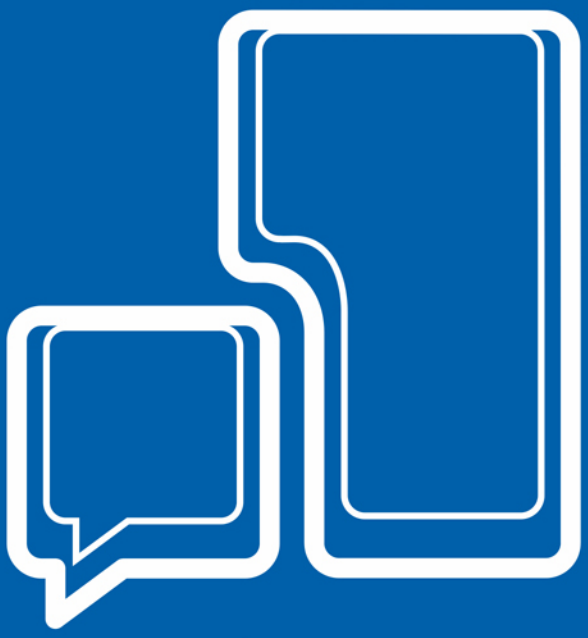
49. Экономический глоссарий. WEB-сайт (Электронн. ресурс) / Способ доступа: URL: http://www.ndc.ru/seminars/piter/10.htm

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

Програмного забезпечення комп'ютерних систем

www.programmer.dp.ua