РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: інформаційна система кондитерської фабрики.

Мета кваліфікаційної роботи: розробка інформаційної системи кондитерської фабрики для автоматизації управління бізнес процесами.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні додатка, що надає можливість електронного зберігання даних про товари, продажі, надходження, ведення бази даних, підбиття підсумків з продажу, отримання звітів і формування супутньої ділової документації.

Актуальність інформаційної системи визначається великим попитом на подібні розробки, що оптимізують та спрощують дії щодо ведення бази даних кондитерської фабрики, скорочують час на оформлення продаж; підвищують ефективність своєї діяльності шляхом електронного ведення документації з можливістю аналізу наявних даних і надання необхідних звітів і супутньої ділової документації.

Список ключових слів: СКЛАД, МАГАЗИН, КОНДИТЕРСЬКА ФАБРИКА, БАЗА ДАНИХ, МОДЕЛЮВАННЯ ПРОЦЕСІВ, АЛГОРИТМИ, ПРОГРАМУВАННЯ.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: information system of confectionery factory.

The purpose of the qualification work: development of an information system of a confectionery factory for automation of business process management.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes existing solutions, selects a platform for development, performs design and development of the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download of the program, describes the program .

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance lies in the creation of an application that provides the ability to electronically store data on goods, sales, receipts, database maintenance, summarizing sales, receiving reports and the formation of related business documentation.

The relevance of the information system is determined by the high demand for such developments, which optimize and simplify the maintenance of the database of the confectionery factory, reduce the time for registration of sales; increase the efficiency of their activities by electronic record keeping with the ability to analyze available data and provide the necessary reports and related business documentation.

List of keywords: WAREHOUSE, SHOP, CONFECTIONERY, DATABASE, PROCESS MODELING, ALGORITHMS, PROGRAMMING.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІС − інформаційна система;

БД − база даних;

ОС − операційна система;

ПЗ − програмне забезпечення;

ПК − персональний комп'ютер;

СКБД − система керування базою даних.

ЗМІСТ

ВСТУП

На сьогодні виробництво кондитерської продукції є однією з найрозвинутіших галузей харчової промисловості в Україні та світі загалом. Кондитерський ринок є високо конкурентним та насиченим. В кондитерській галузі на виробництві задіяне близько 170 тис. працюючих. Виробничі потужності галузі завантажені орієнтовно на 70%. Загальний обсяг виробництва становить понад 1 млн. т продукції на рік, що дає змогу не лише повністю забезпечити потреби внутрішнього ринку, а й експортувати її у значних обсягах за кордон.

Такий значний поштовх кондитерській сфері забезпечили багато факторів, але один з головних факторів − автоматизація підприємств за допомогою інформаційних систем.

На кондитерському підприємстві може одночасно зберігатися, готуватися до обробки і виготовлятися більше 100 найменувань кондитерських виробів, на декількох технологічних лініях, які відрізняються по конструкції і цільовому призначенню. Підвищення ефективності виробництва можливе за рахунок підвищення продуктивності технологічних процесів, поліпшення якості продукції, що випускається, зниження об'єму незавершеного виробництва і страхових запасів за рахунок локалізації і синхронізації регламентованого випуску товару. Це забезпечується за рахунок вдосконалення автоматизованих систем управління технологічними процесами (АСУ ТП), тому що дозволяє перевести виробничий процес на якісно новий ступінь розвитку, який характеризується вищим у порівнянні з попереднім, ступенем організації автоматизованого управління.

Метою кваліфікаційної роботи бакалавра є розробка інформаційної системи кондитерської фабрики для автоматизації управління бізнес процесами.

Для досягнення поставленої мети необхідно виконати наступні етапи:

– розробити структуру автоматизованої системи обробки даних для кондитерської фабрики;

– розробити базу даних системи;

– розробити алгоритми функціонування системи;

– створити комплекс інформаційного, програмного, технічного забезпечення для реалізації розроблених алгоритмів.

Основні задачі для вирішення:

1. Ведення списків продукції (за видами, типами);
2. Формування прайсу продукції;
3. Облік замовлень (з веденням їх архіву);
4. Формування звітності та статистики.

Практичне значення полягає у створенні додатка, що надає можливість електронного зберігання даних про товари, продажі, надходження, ведення бази даних, підбиття підсумків з продажу, отримання звітів і формування супутньої ділової документації.

Розроблена інформаційна система призначена для комерційного застосування в будь-якому з кондитерських підприємств.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Основним завдання кондитерської фабрики є виробництво та збут кондитерських виробів. Для цього фіксується дата виробництва и дата поставки продукту (кожен продукт має свій унікальний код), кількість товару, його тип.

Щоб точно фіксувати постачання продуктів магазин (споживач) має № накладної (унікальний), кількість надходження товару, ціну за одиницю товару, дату поставки, ПІБ директора.

Це дозволяє систематизувати роботу фабрики: фіксувати дату випуску и дату збуту продукції, його тип, назву; легко дізнатися дані магазину (споживача), якій виконав імпорт товару. Також за допомогою цих даних можна швидко і зручно використовувати інформацію про постачання.

Кондитерська галузь має передумови для успішного розвитку і високої конкурентоспроможності на внутрішньому і зовнішньому ринках. Провідні кондитерські фабрики вже провели модернізацію та автоматизацію своєї бази даних, встановили найсучасніші виробничі лінії, значно підвищили технологічність і наукомісткість підприємств. Це дає підставу стверджувати, що кондитерська сфера зараз та в майбутньому буде розвиватися неймовірними темпами, і цьому розвитку неодмінно сприяла автоматизація БД і всієї сфери в цілому.

База даних (англ. database) - сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

У сучасних інформаційних системах для забезпечення роботи з базами даних використовують системи керування базами даних (СКБД). Система керування базами даних - це система, заснована на програмних та технічних засобах, яка забезпечує визначення, створення, маніпулювання, контроль, керування та використання баз даних. Програми для роботи з базою даних можуть бути частиною СКБД або автономними.

Інформаційна система (англ. Information system) – сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів.

Конкурентоспроможність кондитерської продукції у великій мірі залежить від рівня автоматизації технологічних процесів (ТП), що враховує традиції і накопичений досвід фахівців кондитерів, які розробляють рецептуру. АСУ ТП повинна бути реалізована на основі відповідних методів адаптації з інтелектуальним наповненням знаннями і може називатися системою управління на основі експертних оцінок.

1.2. Призначення розробки та галузь застосування

Розроблена інформаційна система кондитерської фабрики оптимізує та спрощує дії щодо ведення бази даних, скорочує час на оформлення продаж; підвищує ефективність своєї діяльності шляхом електронного ведення документації з можливістю аналізу наявних даних і надання необхідних звітів і супутньої ділової документації.

Створений додаток надає можливість електронного зберігання даних про товари, продажі, надходження, ведення бази даних, підбиття підсумків з продажу, отримання звітів і формування супутньої ділової документації.

Розроблена інформаційна система призначена для комерційного застосування в будь-якому з кондитерських підприємств.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка автоматизованої системи обробки даних для кондитерської фабрики засобами C++ Builder6» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Завданням даної кваліфікаційної роботи є розробка інформаційної системи, що включає в себе базу даних, яка буде надавати інформацію про продукцію, поставки, прайс-лист, замовлення, клієнтів та інтерфейс для забезпечення доступу до цих даних, а також можливість отримання звітності про роботу фабрики за відповідними запитами.

Метою програми є розробка інформаційної системи кондитерської фабрики для автоматизації управління бізнес процесами, а також зручне оформлення замовлень клієнтів компанії.

Для досягнення поставленої мети необхідно вирішити наступні основні задачі:

– розробити структуру автоматизованої системи обробки даних для кондитерської фабрики;

– розробити базу даних системи;

– розробити алгоритми функціонування системи;

– створити комплекс інформаційного, програмного, технічного забезпечення для реалізації розроблених алгоритмів.

Основні задачі для вирішення:

1. Ведення списків продукції (за видами, типами);
2. Формування прайсу продукції;
3. Облік замовлень (з веденням їх архіву);
4. Формування звітності та статистики.

Вхідна інформація даної системи - діяльність та внутрішня структура кондитерської фабрики. Вихідна інформація - звіти та результати запитів.

Вхідними даними для розробки даної системи є інформація про фабрику та її функціонування, клієнти, продукцію, яку вона випускає та постачальників цієї продукції.

Інформаційна система повинна забезпечувати:

– можливість додавання, змін та видалення даних;

– можливість обробки даних: пошуку, підбору даних по запитам, систематизацію інформації та фільтрацію даних по певній умові;

– зручний користувальницький інтерфейс, який реалізує різні рівні доступу, розраховані на різні категорії користувачів;

– містити необхідні запити, форми і звіти для обробки збереженої інформації;

Результатом виконання даної роботи повинна бути розроблена база даних, що служить для автоматизації обробки інформації про кондитерську фабрику, що дозволяє отримувати, змінювати, шукати інформацію, обмежувати доступ до неї, мати стійкість до різних апаратним і програмним збоїв.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

В основі інформаційної системи обробки даних для кондитерської фабрики є проектування бази даних та організація зручного доступу до неї. База даних створюється для вирішення наступних завдань:

– структурувати інформацію про постачальників;

– структурувати інформацію про готової продукції;

– структурувати інформацію про сировину;

– структурувати інформацію про покупців.

База даних повинна бути спроектована таким чином, щоб можна було отримати наступну інформацію:

– списки покупців;

– списки постачальників;

– списки поставленого сировини;

– списки витрат на виготовлення продукції;

– інформацію про продаж продукції;

– інформацію про кількість виготовленої продукції;

Також на основі даних з БД необхідно:

– сформувати вихідний документ «Продажі»;

– розрахунковим шляхом отримувати ціну товару;

– ведення списків продукції за видами та типами;

– формування прайс-листа;

– формування та ведення обліку замовлень;

– формування та ведення списку клієнтів;

– формування звітності та статистики.

Розроблена інформаційна система повинна мати дворівневий доступ: адміністратор та користувач (працівник).

При запуску програми користувач підприємства повинен мати можливість зареєструватися, або авторизуватися.

Функції адміністратора:

– можливість створювати замовлення на будь-яку продукцію, яка є у наявності;

– перегляд інформації по виробам кондитерської фабрики (назви виробів, їх наявність, вага, ціна, кількість та їх склад);

– перегляд історії замовлень (звіти на контракти та замовлення);

– формування прайс-листа;

– формування та ведення обліку замовлень;

– формування та ведення списку клієнтів;

– формування звітності та статистики.

Функції працівника:

– перегляд таблиць бази даних, які розбиті на категорії: Продукція,

Сировина, Замовлення, Клієнти;

– можливість управління даних (перегляд, редагування та додавання даних) у таблицях бази даних за допомогою спеціальних форм.

– пошук потрібних даних у таблицях та їх друк.

1.5.2. Вимоги до інформаційної безпеки

Для забезпечення інформаційної безпеки розроблена база даних повинна:

– забезпечувати збереження і надання на вимогу даних;

– забезпечувати можливість додавання, змін та видалення даних;

– забезпечувати захист даних від несанкціонованого доступу;

– контролювати цілісність, несуперечність, збереження та достовірність інформації;

– передбачати архівацію і відновлення даних.

1.5.3. Вимоги до складу та параметрів технічних засобів

Мінімальні системні вимоги для комп'ютера, на якому планується впровадження даної ІС та які дозволять швидко та стабільно працювати з програмою:

– операційна система Microsoft Windows 7;

– процесор з частотою не менше 3 ГГц;

– не менше 3 Гб оперативної пам'яті;

– від 48 Гб на жорсткому диску;

– графічна карта сумісна з DIRECTX 10;

– пристрій для читання дисків;

– звукова карта 7.1;

– необхідний вільний простір на диску - 2Гб.

1.5.4. Вимоги до інформаційної та програмної сумісності

ІС має бути створено засобами C++ Builder6. Програма має функціонувати у таких операційних системах: Windows 7, Windows 8, Windows 10 та Linux UBUNTU.

Програма повинна являти собою самостійний виконуваний модуль і бути структурована і закоментованою.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Створений додаток надає можливість електронного зберігання даних про товари, продажі, надходження, ведення бази даних, підбиття підсумків з продажу, отримання звітів і формування супутньої ділової документації.

Основні задачі, що вирішує дана розробка:

1. Ведення списків продукції (за видами, типами);
2. Формування прайсу продукції;
3. Облік замовлень (з веденням їх архіву);
4. Формування звітності та статистики.

Програма має дворівневий доступ: адміністратор та користувач (працівник). При запуску програми користувач авторизується за допомогою діалогового вікна з полями - логін та пароль. При невдалій спробі їх заповнення програма сповістить про помилку. Етап авторизації користувача програми надає йому унікальні права та розмежовує їх за рівнем доступу.

Кожному рівню доступні свої унікальні функції.

Функції працівника:

– перегляд інформації по виробам кондитерської фабрики;

– перегляд адресів магазинів, в яких присутня продукція фабрики;

– склад виробів кондитерської фабрики;

– можливість управління даних (перегляд, редагування та додавання даних) у таблицях бази даних за допомогою спеціальних форм.

– пошук потрібних даних у таблицях та їх друк.

Функції адміністратора бази даних:

– можливість створювати замовлення на будь-яку продукцію, яка є у наявності;

– перегляд інформації по виробам кондитерської фабрики (назви

виробів, їх наявність, вага, ціна, кількість та їх склад);

- перегляд історії замовлень (звіти на контракти та замовлення);
- формування прайс-листа;
- формування та ведення обліку замовлень;
- формування та ведення списку клієнтів;
- формування звітності та статистики.

2.2. Опис застосованих математичних методів

Під час проектування та розробки даної інформаційної системи математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

Для розробки та проектування дипломного проекту використано мови програмування SQL та засоби C++ Builder6. Програма функціонує у таких операційних системах: Windows 7, Windows 8, Windows 10 та Linux UBUNTU.

Windows 7 - назва операційної системи Windows, яка вийшла 22 жовтня 2009 року, менше, ніж через три роки після випуску попередньої операційної системи, комерційно невдалої Windows Vista. Розробка Windows 7 під різними кодовими назвами велась з 2000 року. Над нею працювали 2500 осіб.

Усього Microsoft пропонує 6 різних варіантів постачання, проте через роздрібну мережу поширюються переважно версії Home Premium і Professional. Інші варіанти поширюються обмежено, наприклад, тільки для корпоративних клієнтів або тільки в країнах, що розвиваються. Всі варіанти Windows 7, крім найбільш спрощених, працюють як на 32-бітній, так і на 64-бітній платформах. Для приватних користувачів передбачається спеціальна сімейна версія: умови ліцензії дозволяють встановити її не на одному, а на двох або трьох комп'ютерах.

Початкова редакція (Windows 7 Starter) поширюється виключно з новими комп'ютерами, вона не включає функціональної частини для програвання H.264,

AAC, Mpeg-2. Домашня базова - призначена виключно для випуску в країнах, що розвиваються, в ній немає інтерфейсу Windows Aero з функціями Peek і Shake і передперегляду в панелі завдань, загального доступу до підключення в інтернет і деяких інших функцій. У професійній, корпоративній і максимальній є підтримка XP Mode.

Всі 32-бітові версії підтримують до 3 Гб ОЗП. 64-бітові редакції від 4 Гб (Домашня базова) до 192 Гб пам'яті у всіх останніх редакціях, за винятком домашньої розширеної, в ній обмеження - 16 Гб.

Windows 8 – операційна система (ОС) від компанії Microsoft для персональних комп'ютерів, включаючи домашні та офісні системи, ноутбуків, планшетів і домашніх театрів. Розробка Windows 8 почалася ще до випуску її попередниці, Windows 7, у 2009. Система була анонсована на CES 2011, потім були випущені три підготовчі версії з вересня 2011 по травень 2012. На друк операційна система була направлена 1 серпня 2012, і остаточно випущена для широкого загалу 26 жовтня 2012.

Windows 8 ввела значні зміни у платформу операційної системи та користувацький інтерфейс для покращення роботи з планшетами, де Windows зараз конкурує з такими мобільними операційними системами як Android та Apple iOS. Зокрема, ці зміни включають оптимізовану під сенсорні екрани оболонку Microsoft "Metro", стартовий екран (що показує програми і динамічно оновлює вміст решітки плиток), нову платформу для розробки застосунків з акцентом на сенсорному вводі, інтеграцію онлайн служб (включаючи можливість синхронізувати застосунки і налаштування між пристроями), і онлайновий магазин для завантаження та купівлі нових програм. Windows 8 також додав підтримку таких нових технологій як USB 3.0, Advanced Format твердих дисків, NFC, і хмарні обчислення. Були додані нові додаткові можливості безпеки, такі як вбудовані антивіруси, інтеграція з фільтрацією Microsoft SmartScreen і підтримка безпечного завантаження із системою UEFI для уникнення враження шкідливими програмами під час завантаження ОС.

Випуск Windows 8 отримав неоднозначні відгуки. Поряд з відзначенням

покращеної продуктивності, поліпшеної безпеки і підтримки сенсорних екранів, нові користувачі критикували новий графічний інтерфейс через його незвичність, заплутаність і важкість у освоєнні (особливо при використанні клавіатури і миші).

Windows 10 – операційна система від компанії Microsoft для персональних комп'ютерів, ноутбуків, планшетів, лептопів-трансформерів і смартфонів. У компанії цю версію операційної системи називають останньою, позаяк надалі вона надаватиметься за моделлю «програмне забезпечення як послуга» і періодично оновлюватиметься.

Microsoft представила попередню версію Windows 10 у Сан-Франциско 30 вересня 2014 року. Реліз Windows 10 відбувся влітку 2015 року, а саме 29 липня в 190 країнах і 111-ма мовами. Протягом першого року після виходу системи користувачі мали змогу безкоштовно оновитися до Windows 10 на будь-якому пристрої під керуванням офіційних версій Windows 7, Windows 8.1 і Windows Phone 8.1, що відповідають певним вимогам.

Windows 10 – перша операційна система Microsoft, яка офіційно поширюється не тільки з серверів постачальника, але і з комп'ютерів її користувачів. За таким же принципом поширюються і поновлення Windows 10. Також є можливість залишити обмін оновленнями тільки між комп'ютерами локальної мережі [6].

Програма написана мовою C ++. Вона широко використовується для розробки програмного забезпечення, будучи одним з найпопулярніших мов програмування. Область його застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для вбудованих систем, високопродуктивних серверів, а також ігор. Існує безліч реалізацій мови C ++, як безкоштовних, так і комерційних і для різних платформ. Наприклад, на платформі x86 це GCC, Visual C ++, Intel C ++ Compiler, Embarcadero (Borland) C ++ Builder і інші. C ++ зробив величезний вплив на інші мови програмування, в першу чергу на Java і C #.

Мова виникла на початку 1980-х років, коли співробітник фірми Bell Labs Бйорн Страуструп придумав ряд удосконалень до мови С під власні потреби. Коли в кінці 1970-х років Страуструп почав працювати в Bell Labs над завданнями теорії черг (в додатку до моделювання телефонних викликів), він виявив, що спроби застосування існуючих в той час мов моделювання виявляються неефективними, а застосування високоефективних машинних мов занадто складно через їх обмеженою виразності.

Назва «Сі++» була вигадана Ріком Масситті (Rick Mascitti) і вперше було використана в грудні 1983 року. Раніше, на етапі розробки, нова мова називалася «Сі з класами». Ім'я, що вийшло у результаті, походить від оператора Сі «++» (збільшення значення змінної на одиницю) і поширеному способу присвоєння нових імен комп'ютерним програмам, що полягає в додаванні до імені символу «+» для позначення поліпшень.

Стандарт С ++ складається з двох основних частин: опис ядра мови і опис стандартної бібліотеки.

Розвиток мови супроводив розвиток крос-компілятора cfront. Нововведення в мові відбивалися в зміні номера версії крос-компілятора. Ці номери версій крос-компілятора розповсюджувалися і на саму мову, але стосовно до теперішнього часу мова про версії мови С++ не ведуть. Лише в 1998 році мова стала стандартизованим.

Стандартна бібліотека С++ включає стандартну бібліотеку Сі з невеликими змінами, які роблять її більш відповідною для мови С ++. Інша велика частина бібліотеки С++ заснована на Стандартній Бібліотеці Шаблонів (STL). Вона надає такі важливі інструменти, як контейнери (наприклад, вектори і списки) і ітератори (узагальнені вказівники), що надають доступ до цих контейнерів як до масивів. Крім того, STL дозволяє схожим чином працювати і з іншими типами контейнерів, наприклад, асоціативними списками, стеками, чергами.

Доступ до можливостей стандартної бібліотеки С++ забезпечується за допомогою включення в програму (за допомогою директиви #include)

відповідних стандартних заголовків файлів. Всього в стандарті C++ 11 визначено 79 таких файлів. Засоби стандартної бібліотеки оголошуються які входять в простір імен std. Заголовки, імена яких відповідають шаблону «X», де X - ім'я заголовки стандартної бібліотеки C без розширення (cstdlib, cstring, cstdio та ін.). Містять оголошення, відповідні даної частини стандартної бібліотеки C. Стандартні функції бібліотеки C також знаходяться в просторі назв std.

C ++ містить засоби розробки програм контрольованої ефективності для широкого спектра задач, від низькорівневих утиліт і драйверів до вельми складних програмних комплексів. Зокрема висока сумісність з мовою Сі: код на Сі може бути з мінімальними переробками скомпільована компілятором C ++. Візуально-мовний інтерфейс є прозорим, так що бібліотеки на Сі можуть викликатися з C++ без додаткових витрат, і більш того при певних обмеженнях код на C ++ може експортуватися зовні не відрізнятися від коду на Сі.

До числа недоліків мови можна віднести:

– відсутність системи модулів C++ успадкував від Сі підключення заголовних файлів за допомогою препроцесора. Це змушує дублювати опису об'єктів, породжує неочевидні вимоги до коду і збільшує обсяг компілюємого тексту, а значить і час компіляції;

– успадковані від Сі небезпечні і провокують помилки можливості (макроозначення, адресна арифметика, неявне приведення типів, можливість прямого управління розподілом пам'яті).

– відсутність вбудованих механізмів статичної валідації часу життя об'єктів, що приводить до раптового краху програм через звернення до знищеної змінної, або через неправильну багатопотокової роботи з об'єктами.

– шаблони породжують об'ємний і не завжди оптимальний код. Часткове визначення шаблонів ускладнює як сама мова, так і програми, де воно використовується.

Єдиним прямим нащадком C ++ є мовою D, задуманий як переробка C ++

для усунення найбільш очевидних його проблем. Автори відмовилися від сумісності з Сі, зберігши синтаксис і багато базові принципи С ++ і ввівши в мову можливості, характерні для нових мов. У D немає препроцесора, заголовків файлів, множинного спадкоємства, але є система модулів, інтерфейси, асоціативні масиви, підтримка unicode в рядках, прибирання сміття (при збереженні можливості ручного управління пам'яттю) вбудована багатопоточність, висновок типів, явне оголошення чистих функцій і незмінних значень. Використання D вельми обмежена, вважати його реальним конкурентом С ++ можна [17].

Щоб здійснювати збереження і обробку даних в додатку використовується база даних (далі – БД). Вона є невід'ємною частиною додатку.

Сучасні бази даних зберігають великі об'єми інформації, тому обробляти її вручну, послідовно переглядаючи і редагуючи дані в таблицях, стає досить важко. Для підвищення ефективності користування базами даних застосовують запити, які дозволяють виконувати множинну обробку даних, тобто одночасно вводити, редагувати та видаляти велику кількість записів, а також вибирати дані з таблиць.

Бази даних становлять невід'ємну складову діяльності сучасних підприємств та організацій. Невпинне зростання обсягів інформації, що зберігається в базах даних, та розширення кола користувачів спонукають до використання систем керування базами даних (далі - СКБД). Для створення і реалізації бази даних було обрано СКБД Microsoft

SQL Server 2017. Microsoft SQL Server – система керування базами даних, розроблена корпорацією Microsoft [20].

Базовий код MS SQL Server (до версії 7.0) ґрунтувався на коді Sybase SQL Server.

Серед нових можливостей і удосконалень Microsoft SQL Server 2017 слід зазначити появу нових типів даних, а саме – для просторових даних, кращу сумісність з застосунками сторонніх розробників, наприклад Oracle, тіснішу

інтеграцію з Office, оптимізовані засоби шифрування даних, засоби управління на основі політик, а також покращені інструменти звітності і аналізу.

Основною використовуваною мовою запитів при роботі з базою даних є мова SQL. Вона є реалізацією стандарту ANSI / ISO по структуруванню мови запитів (SQL) з розширеннями.

SQL - декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних. Сама по собі SQL не є ані системою керування базами даних, ані окремим програмним продуктом. На відміну від дійсних мов програмування (C або Pascal), SQL може формувати інтерактивні запити або, будучи вбудованою в прикладні програми, виступати як інструкції для керування даними. Окрім цього, стандарт SQL містить функції для визначення зміни, перевірки та захисту даних.

SQL – це діалогова мова програмування для здійснення запиту і внесення змін до бази даних, а також керування базами даних. Багато баз даних підтримує SQL з розширеннями до стандартної мови. Ядро SQL формує командна мова, яка дозволяє здійснювати пошук, вставку, оновлення і вилучення даних за допомогою використання системи керування і адміністративних функцій. SQL також включає CLI (Call Level Interface) для доступу і керування базами даних дистанційно [2].

Структура SQL:

– DDL (Data Definition Language) - робота зі структурою бази;
– DML (Data Manipulation Language) - робота з рядками;
– DCL (Data Control Language) - робота з правами;
– TCL (Transaction Control Language) - робота з транзакціями.

Оператори Data Definition Language:

– CREATE - створення об'єкта (наприклад, таблиці);
– ALTER - зміна об'єкта (наприклад, додавання/зміна полів таблиці);
– DROP - видалення об'єкта (наприклад, таблиці).

Оператори Data Manipulation Language:

– INSERT - вставлення рядка;

– SELECT - вибірка (наприклад, даних з таблиці);

– UPDATE - зміна (наприкоад, таблиці);

– DELETE - видалення (наприклад, рядка з таблиці).

Оператори Data Control Language:

– GRANT - надання прав користувачу;

– DENY - явна заборона для користувача;

– REVOKE - відміна заборони/дозволу користувачу.

Оператори Transaction Control Language:

– BEGIN TRANSACTION - почати транзакцію;

– COMMIT - прийняти зміни прийняті в транзакції;

– ROLLBACK - відкат.

Як і з багатьма стандартами, що мають місце в IT-індустрії, з мовою SQL виникла проблема, пов'язана з рішенням багатьох розробників ПЗ з використанням SQL про недостатню функціональність версії стандарту (що, в принципі, для ранніх версій SQL було певною мірою справедливо) і їхнім бажанням її розширити. Це призвело до того, що у різних виробників СУБД в застосуванні різні діалекти SQL, причому, несумісні між собою у більшості випадків [4].

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Моделювання системи

Моделювання – це процес створення моделі, а також це дослідження об'єктів за допомогою побудови і вивчення їх моделей. В даному випадку використано UML-мову.

Визначимо основних користувачів бази даних.

Користувачами бази даних є адміністратор та будь-який користувач. Адміністратор має можливість ведення списку продукції, замовлень, клієнтів, а також формування прайс-листа та звітів (рис. 2.1).

Рис. 2.1. Діаграма варіантів використання (Use case) для адміністратора

Користувач, в свою чергу, має дозвіл лише на ведення кількості замовлень та розрахунок вартості замовлення (рис. 2.2).

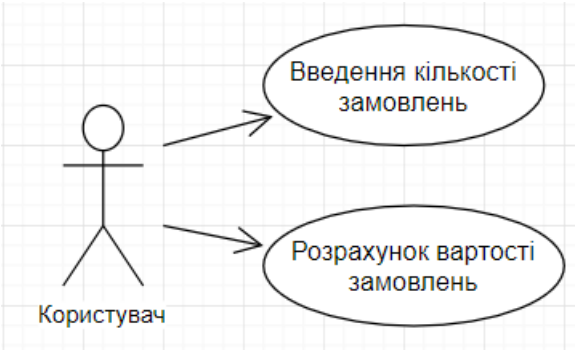

Рис. 2.2. Діаграма варіантів використання (Use case) для користувача

Функції адміністратора, що забезпечуються в програмі:

– можливість створювати замовлення на будь-яку продукцію, яка є у

наявності;

– перегляд інформації по виробам кондитерської фабрики (назви виробів, їх наявність, вага, ціна, кількість та їх склад);

– перегляд історії замовлень (звіти на контракти та замовлення);

– формування прайс-листа;

– формування та ведення обліку замовлень;

– формування та ведення списку клієнтів;

– формування звітності та статистики.

Функції працівника в даній програмі:

– перегляд таблиць бази даних, які розбиті на категорії: Продукція, Сировина, Замовлення, Клієнти;

– можливість управління даних (перегляд, редагування та додавання даних) у таблицях бази даних за допомогою спеціальних форм.

– пошук потрібних даних у таблицях та їх друк.

Основними сутностями розробленої бази даних є:

– список продукції за видами, типами та формою випуску продукції;

– прайс-лист;

– список замовлень;

– клієнт.

Опис типів сутностей приведено в таблиці 2.1.

Таблиця 2.1

Опис типів сутностей

Ім’я типу сутності	Опис
Вид продукції	Сутність, в якій містяться дані про види продукції
Тип продукції	Сутність, в якій містяться дані про типи продукції
Форма випуску	Сутність, в якій містяться дані про форму випуску продукції
Список замовлень	Сутність, в якій містяться дані про замовлення клієнтів
Прайс-лист	Сутність, в якій містяться дані про продукцію
Клієнт	Сутність, в якій містяться дані про клієнтів

Опис зв‘язків сутностей приведено в таблиці 2.2.

Таблиця 2.2

Опис зв‘язків сутностей

Ім’я сутності	Зв’язок	Ім’я сутності	Тип зв’язку
Вид продукції	Один до багатьох	Тип продукції	1:N
Вид продукції	Багато до одного	Список замовлень	N:1
Прайс-лист	Один до багатьох	Вид продукції	1:N
Прайс-лист	Один до багатьох	Форма випуску	1:N
Прайс-лист	Один до багатьох	Тип продукції	1:N
Клієнт	Один до багатьох	Список замовлень	1:N

Концептуальна схема моделі приведена на рис. 2.3.

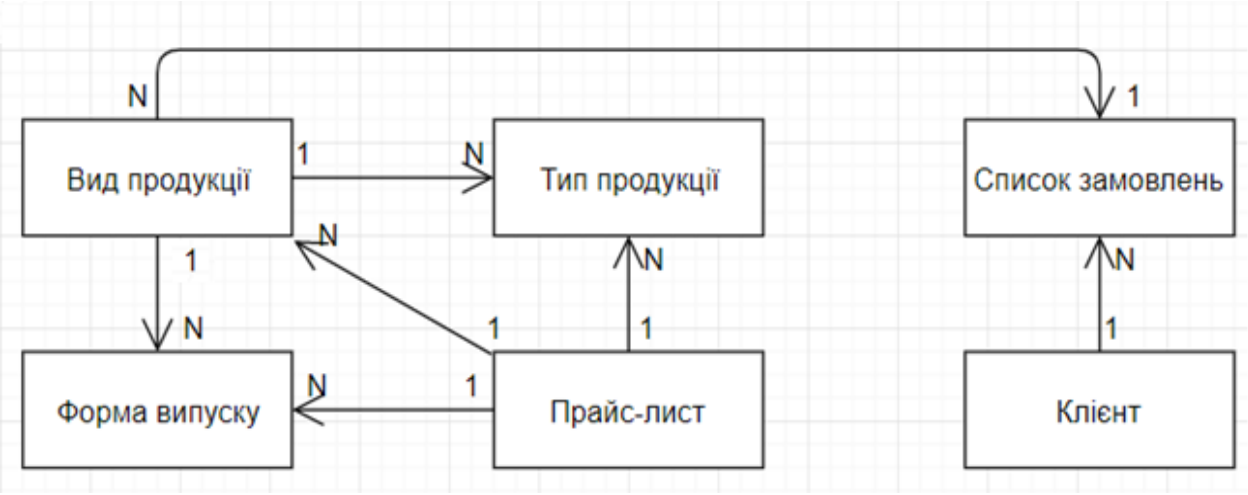

Рис. 2.3. Концептуальна схема даних

2.4.2. Розробка бази даних системи

Кожне реляційне відношення відповідає одній сутності і в нього вносяться всі атрибути сутності. Для кожного відношення необхідно визначити первинний ключ і зовнішні ключі.

ER-діаграма містить інформацію про сутності системи та способи їхньої взаємодії, включає ідентифікацію об'єктів, важливих для предметної області (сутностей), властивостей цих об'єктів (атрибутів) і їхніх відносин з іншими об'єктами (зв'язків), тобто модель база даних відображає її логічну структуру, показує поля кожної таблиці, ключі, зв’язки між таблицями.

Складовою частиною розроблюваної інформаційної системи є база даних програми, основною функцією якої є систематизація зберігання інформації, необхідної для функціонування програми і спрощення роботи з великим набором цієї інформації. Для швидкодії БД проведена її нормалізація.

На рис. 2.4. наведено логічну базу даних системи.

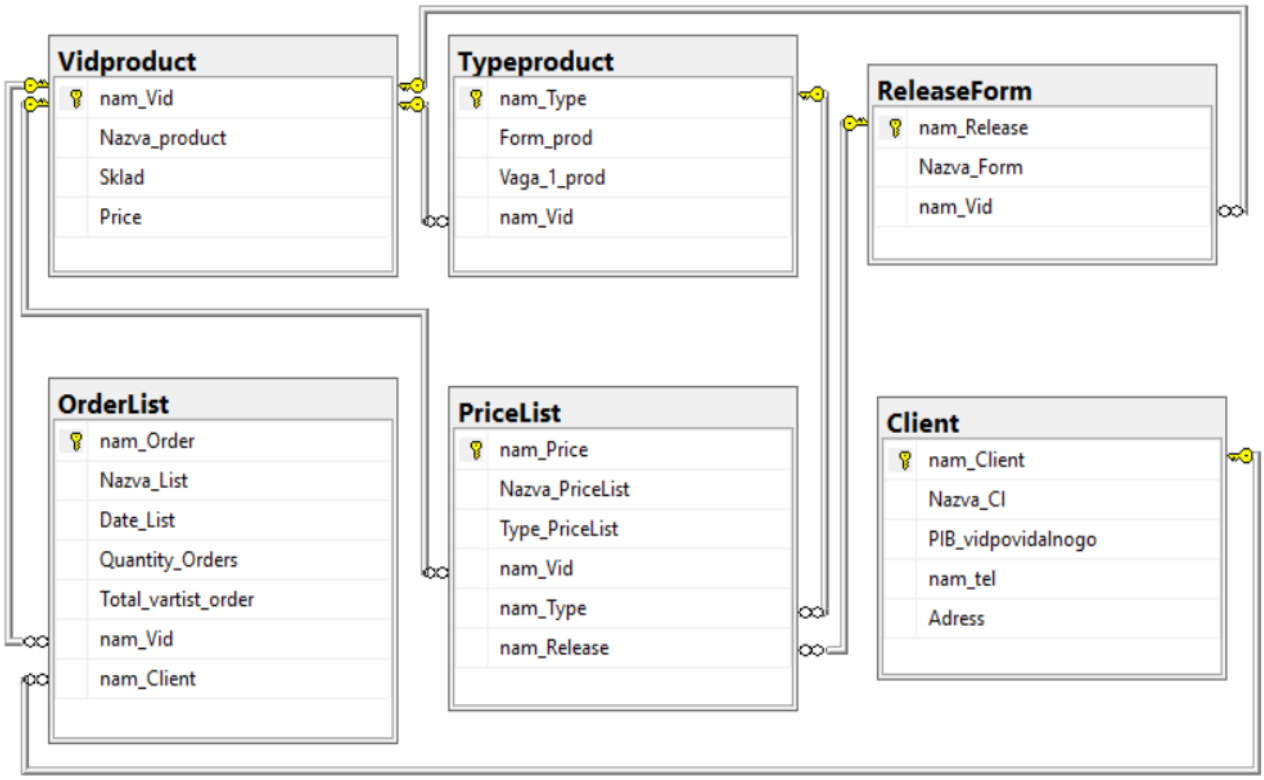

Рис. 2.4. Логічна схема даних

Після проведених перетворень остаточні схеми відносин бази даних наведені в табл. 2.3-2.8.

Таблиця 2.3

«Вид продукції» (VidProduct)

Призначення поля	Назва поля	Тип	Довжина	Ключ
ID продукції	Nam_Vid	INT		PK
Назва продукції	Nazva_product	NVARCHAR	100	
Склад	Sklad	NVARCHAR	60	
Ціна	Price	DECIMAL	10,2	

Таблиця 2.4

«Тип продукції» (TypeProduct)

Призначення поля	Назва поля	Тип	Довжина	Ключ
ID тип продукції	Nam_Type	INT		PK
ID вид продукції	Nam_Vid	INT		FK
Вага	Waga_1_prod	INT		
Форма продажу	Forn_prod	NVARCHAR	60	

Таблиця 2.5

«Форма випуску» (ReleasForm)

Призначення поля	Назва поля	Тип	Довжина	Ключ
ID форма випуску	Nam_Release	INT		PK
ID вид продукції	Nam_Vid	INT		FK
Назва форми випуску	Nazva_Form	NVARCHAR	60	

Таблиця 2.6

«Список замовлень» (OrderList)

Призначення поля	Назва поля	Тип	Довжина	Ключ
ID замовлення	Nam_Order	INT		PK
Назва замовлення	Nazva_List	NVARCHAR	100	
Дата замовлення	Date_list	DATE		
Кількість	Quantity_Order	INT		
Ціна замовлення	Total_vartist_order	DECIMAL	10,2	
ID вид продукції	Nam_Vid	INT		FK
ID client	Nam_Client	INT		FK

Таблиця 2.7

«Прайс-лист» (PriceList)

Призначення поля	Назва поля	Тип	Довжина	Ключ
ID прайс-лист	Nam_Price	INT		PK
ID вид продукції	Nam_Vid	INT		FK
ID тип продукції	Nam_Type	INT		FK
ID форма випуску	Nam_Release	INT		FK
Назва прайсу	Nazva_ PriceList	NVARCHAR	100	
Тип прайсу	Type_ PriceList	NVARCHAR	100	

Таблиця 2.8

«Клієнт» (Client)

Призначення поля	Назва поля	Тип	Довжина	Ключ
ID клієнта	Num_Client	INT		PK
Назва клієнта	Nazva_Cl	NVARCHAR	100	
ПІБ відповідального	PIB_vidppoidalnogo	NVARCHAR	100	
Номер телефону	Nam_tel	INT		
Адреса	Adres	NVARCHAR	100	

На рис. 2.5 приведена схема обробки даних з БД додатку.

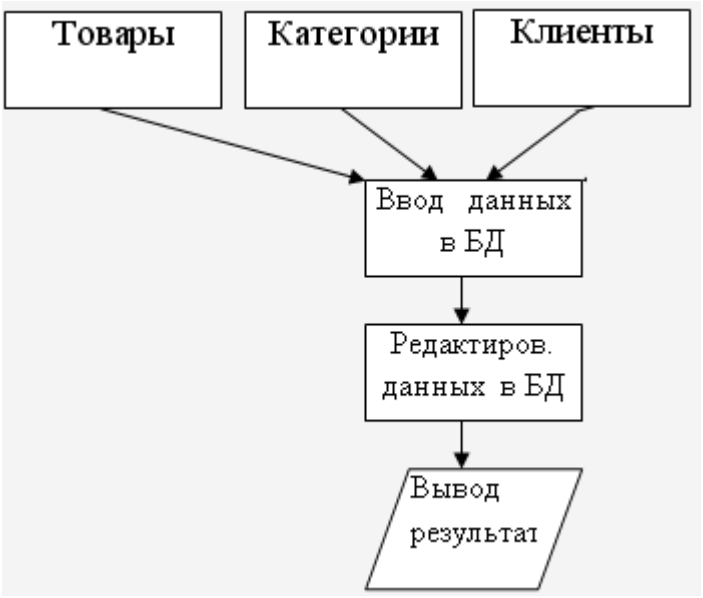

Рис. 2.5. Схема обробки даних

На рис. 2.6 представлена схема здійснення процесу обліку продажів в БД.

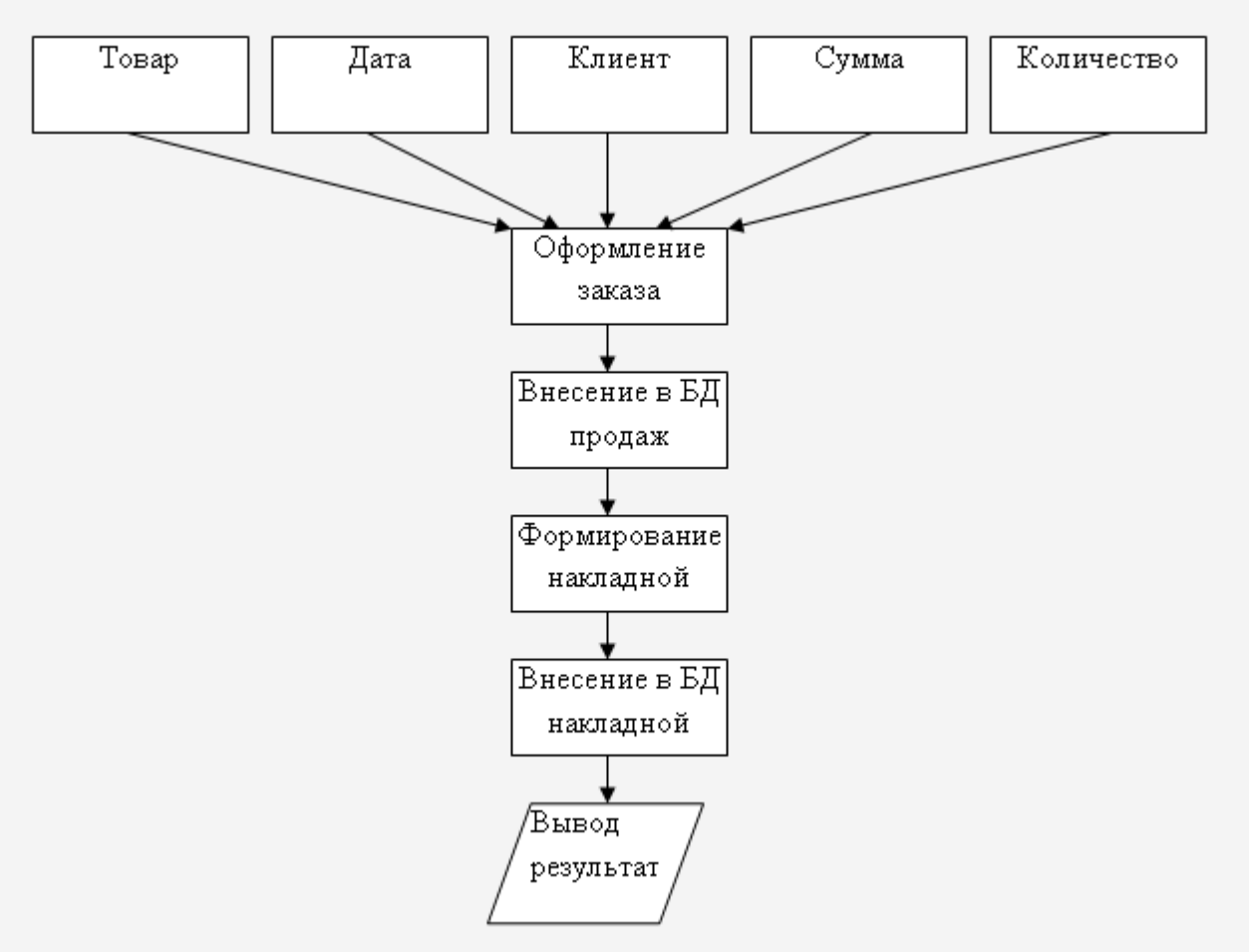

Рис. 2.6. Схема обліку продажів

2.4.3. Проектування користувальницького інтерфейсу

Автоматизована інформаційна система конфетної фабрики працює у меню-орієнтованому режимі. Схема інтерфейсу програми приведена на рис 2.7.

Рис. 2.7. Схема інтерфейсу програми

Для автоматизованої інформаційної системи конфетної фабрики була розроблена початкова модель користувацького інтерфейсу, яка приведена на рис.2.8.

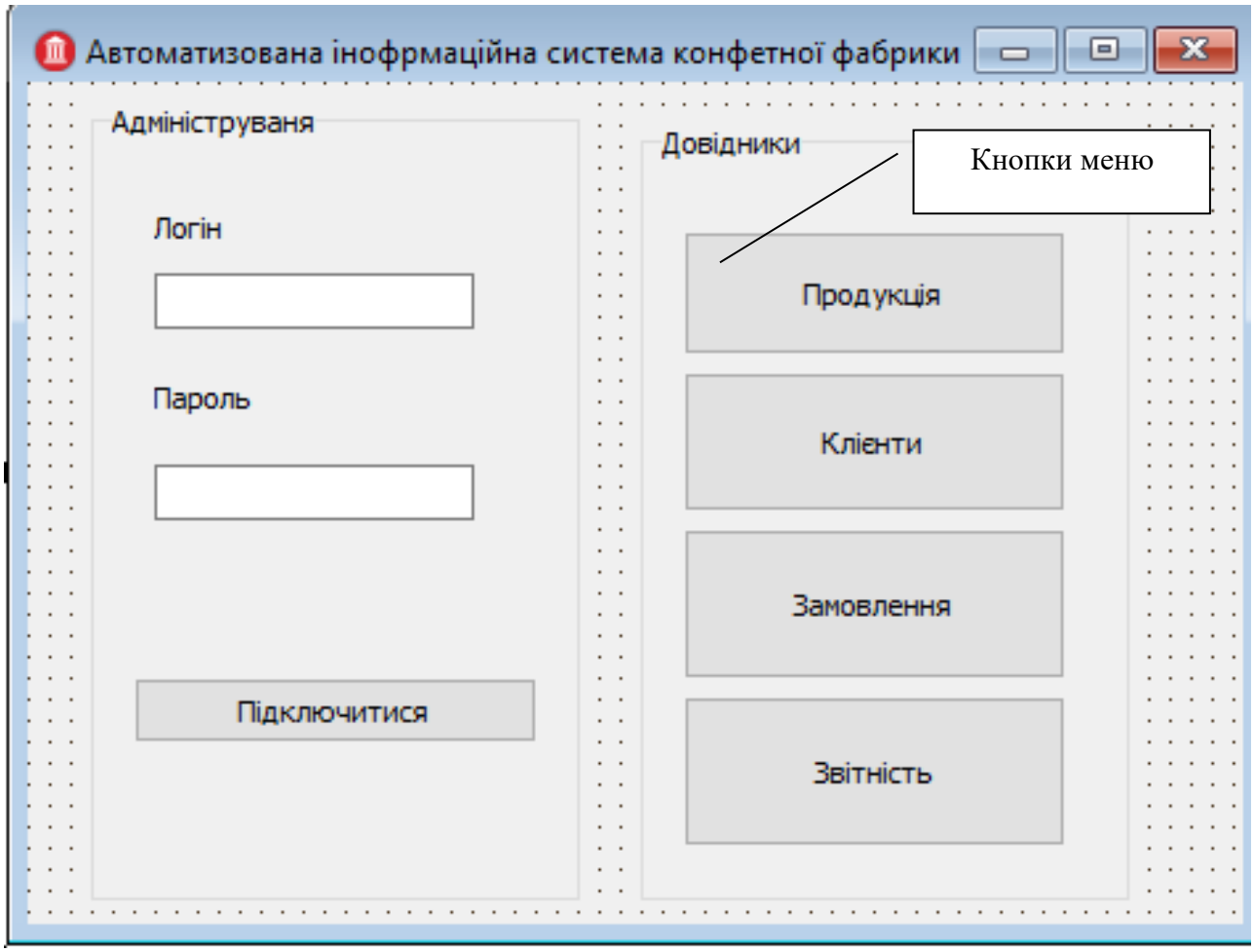

Рис. 2.8. Початкова модель користувацького інтерфейсу головного вікна

Меню головної форми програми складається з панелі авторизації користувача та кнопок доступу до головних складових доступу до БД системи: Продукція, Клієнти, Замовлення та Звітність. При натискання на обрані кнопки меню завантажуються відповідні другорядні форми (рис. 2.9), що забезпечують доступ до таблиць БД та дозволяють редагувати їх та виконувати відповідні запити (рис. 2.10).

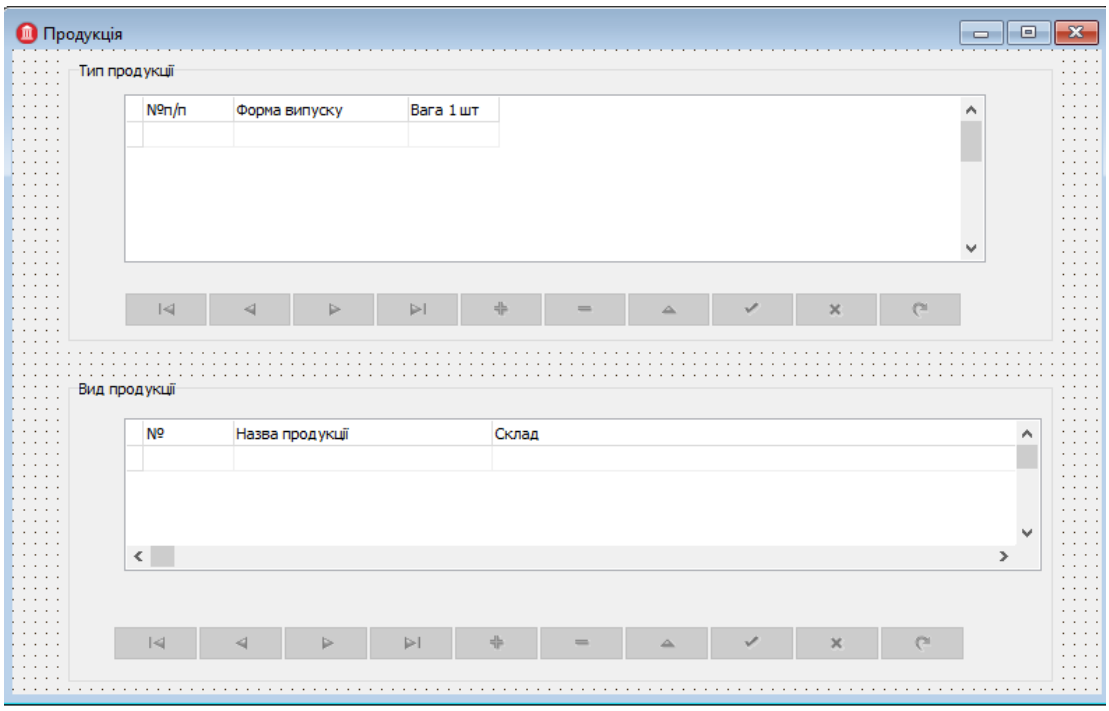

Рис. 2.9. Початкова модель користувацького інтерфейсу другорядних вікон

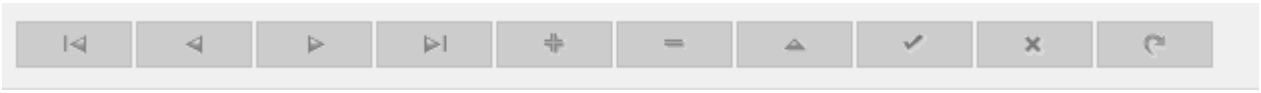

Рис. 2.10. Функціональні кнопки меню

2.4.4. Опис структури програми

Текст програми написаний мовою C++ та складається з чотирьох основних програмних модулів: MainForm.cpp, LoginForm.cpp, ClientsForm.cpp, GoodsForm.cpp, ProvidersForm.cpp.

MainForm.cpp – це основний модуль програми, що містить реалізацію класу TFormMain. Клас TFormMain містить обробники подій елементів інтерфейсу головного вікна та функції, що забезпечують підключення до бази даних.

LoginForm.cpp – модуль, що містить реалізацію класу TFormLogin. Клас TFormMain містить реалізацію елементів інтерфейсу вікна авторизації користувача та функції, що забезпечують передачу даних від таблиць бази даних у програму та обробку цих даних.

ClientsForm.cpp – модуль, що містить реалізацію класу TFormClients. Клас TFormClients містить реалізацію елементів інтерфейсу вікна роботи з базою клієнтів. Також містить функції, що забезпечують передачу даних від таблиці бази даних «Клієнти» у програму, обробку цих даних, та внесення змін до бази даних.

GoodsForm.cpp – модуль, що містить реалізацію класу TFormGoods. Клас TFormGoods містить реалізацію елементів інтерфейсу вікна роботи з базою товарів. Також містить функції, що забезпечують передачу даних від таблиці бази даних у програму, обробку цих даних, та внесення змін до бази даних.

Діаграма класів програми наведена на рис. 2.11.

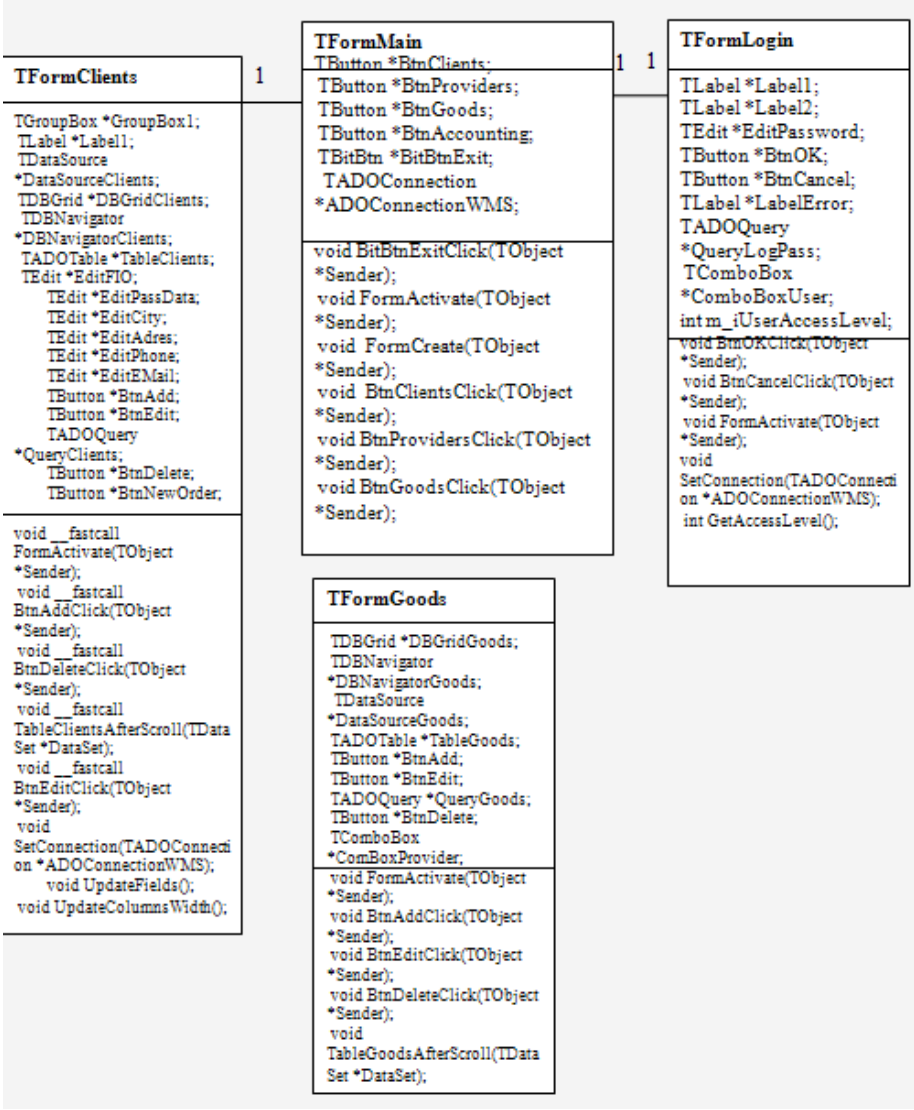

Рис. 2.11. Діаграма класів додатку

На рис. 2.12 наведена структура форми проекту DataModule

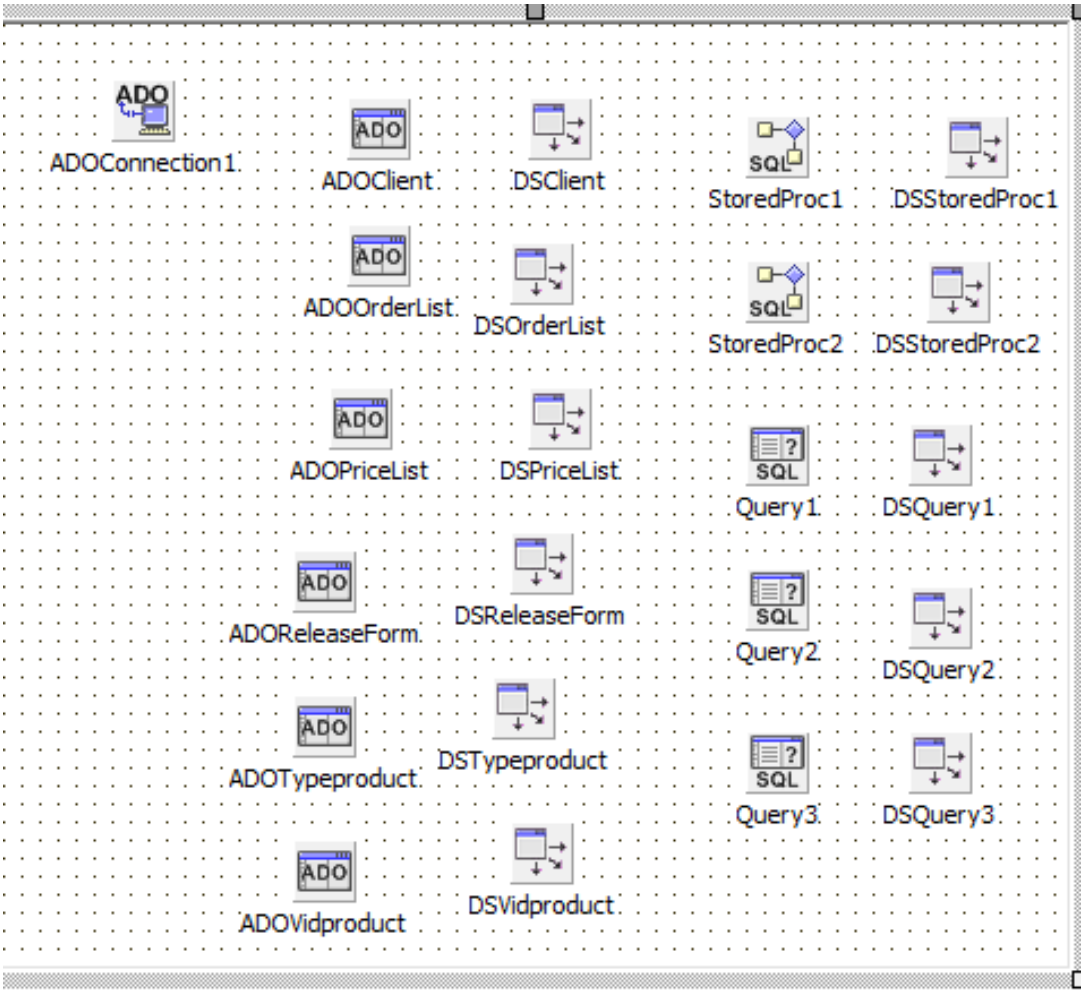

Рис. 2.12. Схема форми DataModule

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Складовою частиною розроблюваної інформаційної системи є база даних програми, основною функцією якої є систематизація зберігання інформації, необхідної для функціонування програми і спрощення роботи з великим набором цієї інформації. База даних включає в себе вводимо інформацію про: вид продукції, типи продукції, форму випуску продукції, список замовлень, дані про клієнтів та прайс-лист продукції. На основі цих даних формуються відповідні звіти за запитами та обраховується загальна вартість замовлень.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Рекомендовані системні вимоги для комп'ютера, які дозволять швидко та стабільно працювати з програмою:

– операційна система Microsoft Windows 10;
– процесор з частотою не менше 4 ГГц;
– не менше 8 Гб оперативної пам'яті;
– від 64 Гб на жорсткому диску;
– звукова карта 5.1;
– рекомендований вільний простір на диску - 4Гб.

2.6.2. Використані програмні засоби

Використані засоби – система керування базами даних SQL Server 2012 та інтегрована систему розробки програмного забезпечення C++ Builder6 в операційній системі Windows 10.

2.6.3. Виклик та завантаження програми

Для запуску програми потрібно відкрити виконуючий файл програми «Fabrika.exe».

Після запуску відобразиться головне вікно програми, яке містить кнопки меню та форму адміністрування (першочергове підключення до бази даних) (рис.2.13).

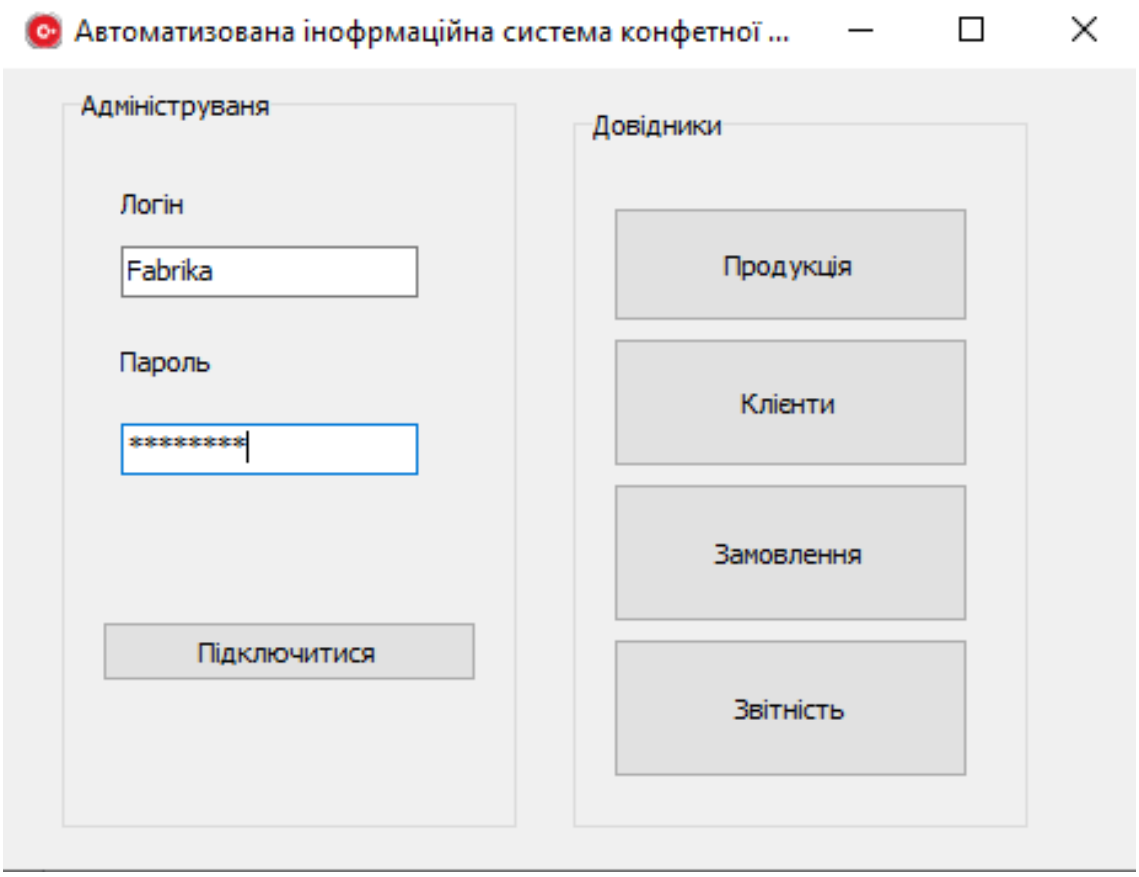

Рис. 2.13. Головне вікно програми

Якщо підключення не відбулося відкривається відповідне вікно повідомлення (рис.2.14).

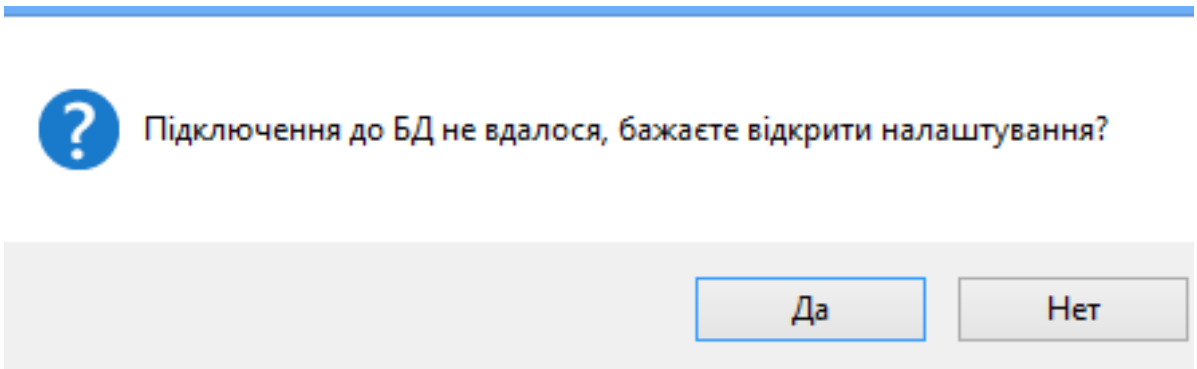

Рис. 2.14. Відеограма виводу повідомлення про невдале підключення до БД

2.6.4. Опис інтерфейсу користувача

Після успішної авторизації та підключення до БД системи, відобразиться головне вікно програми, яке містить кнопки доступу до головних складових БД системи: Продукція, Клієнти, Замовлення та Звітність (рис. 2.15).

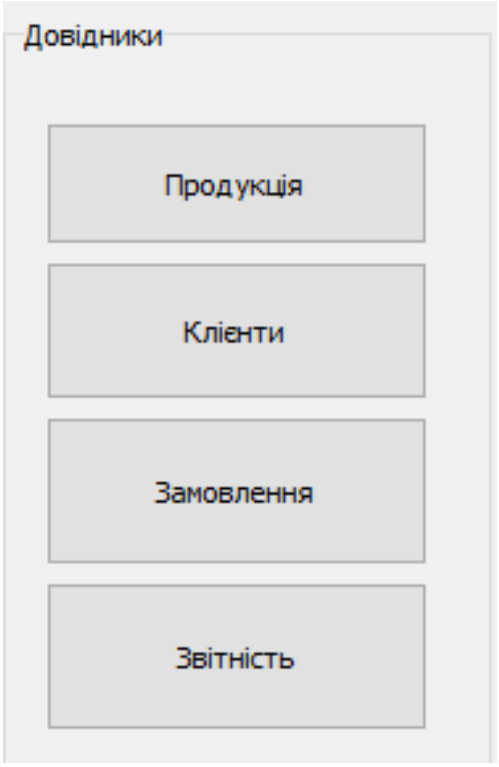

Рис. 2.15. Головне вікно програми

При натискання на обрані кнопки меню завантажуються відповідні форми, що забезпечують доступ до таблиць БД та дозволяють редагувати їх та виконувати відповідні запити.

При виборі відповідної кнопки відкривається вікно з табличним даними та кнопками для керування набором даних (рис.2.16 – 2.18). При формуванні замовлення, а також заповненні даних про товар, використовуються випадаючі списки (дати, клієнта, типу та виду продукції).

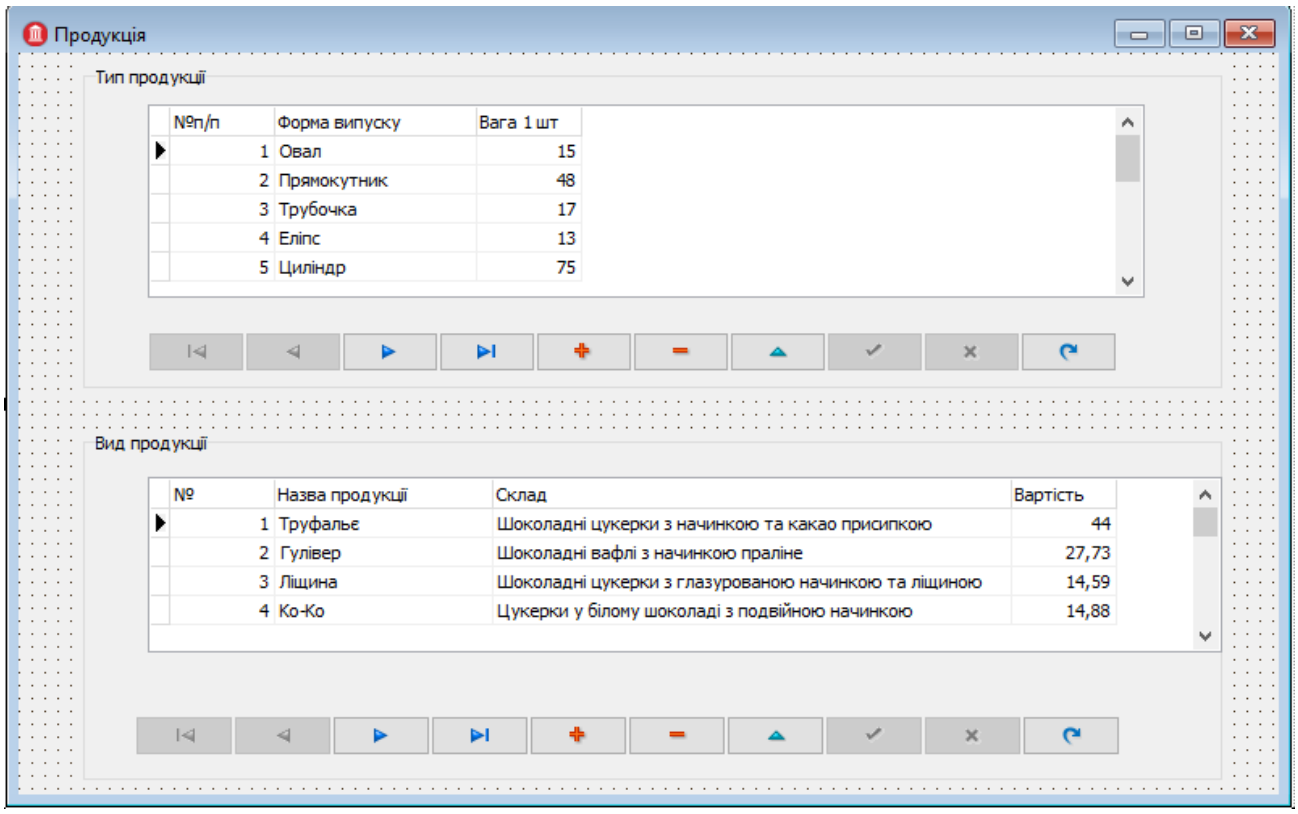

Рис. 2.16. Форма «Продукція»

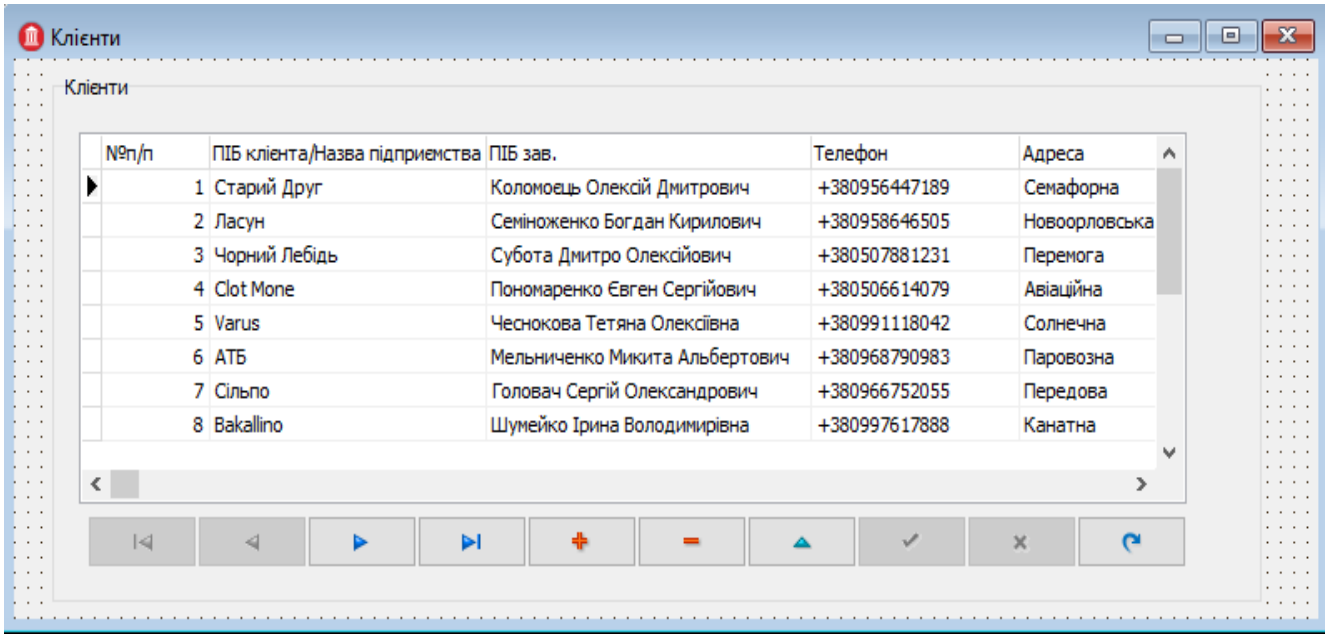

Рис. 2.17. Форма «Клієнти»

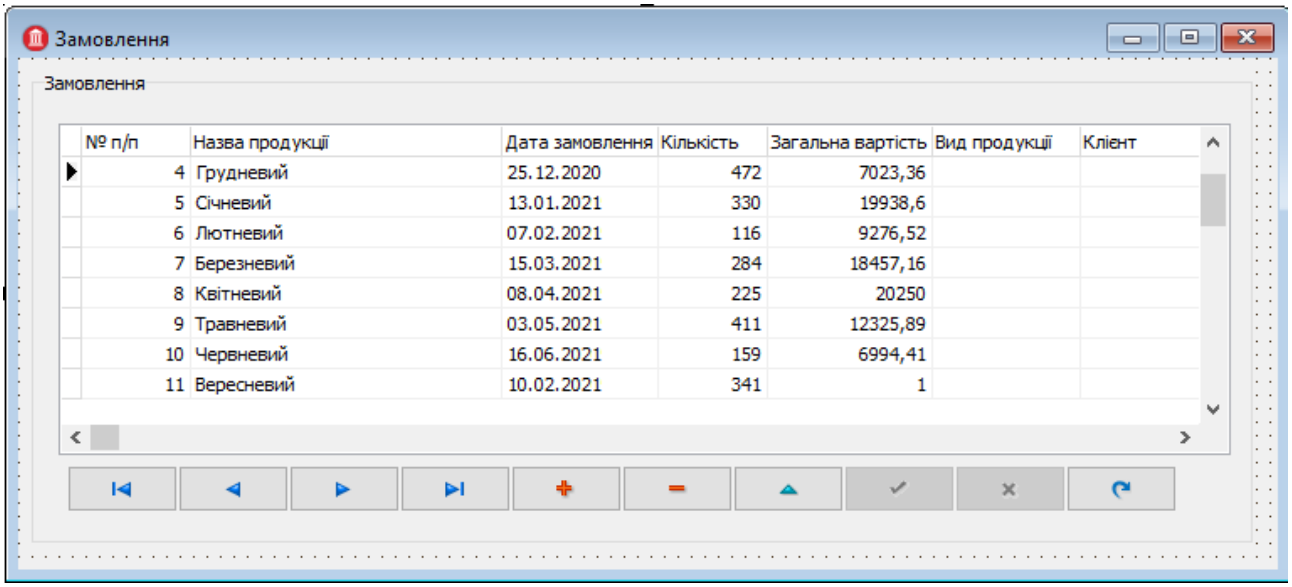

Рис. 2.18. Кнопка меню «Замовлення»

Всі статистичні звіти зведені в одному вікні, яке активізується при натиску на кнопку «Звітність». Відеограми статистичних даних приведені на рис. 2.19-2.23.

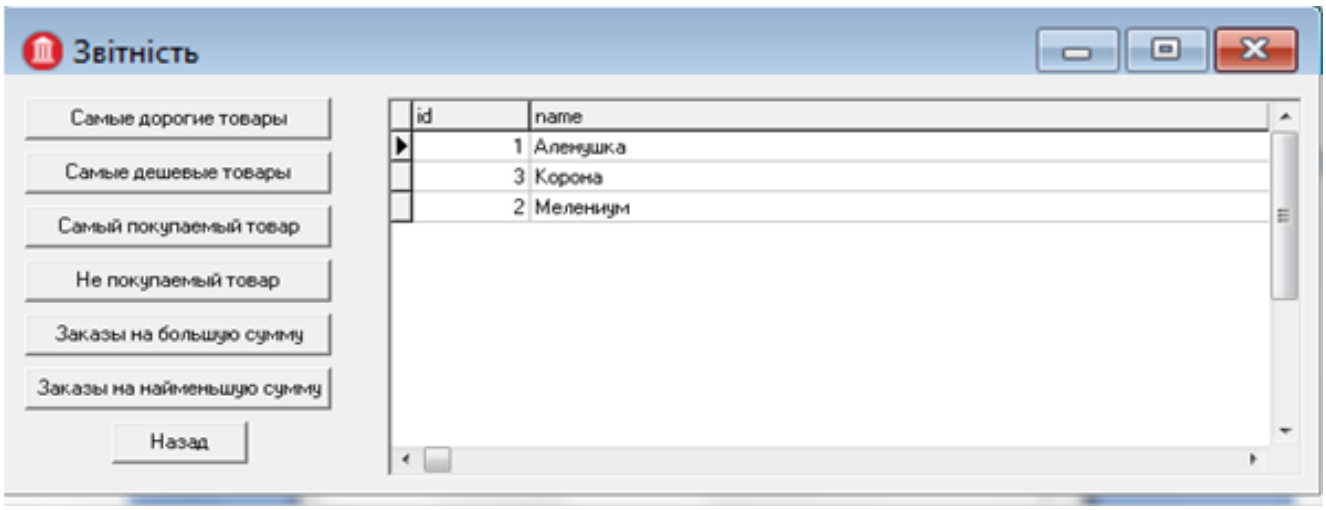

Рис. 2.19. Звітність

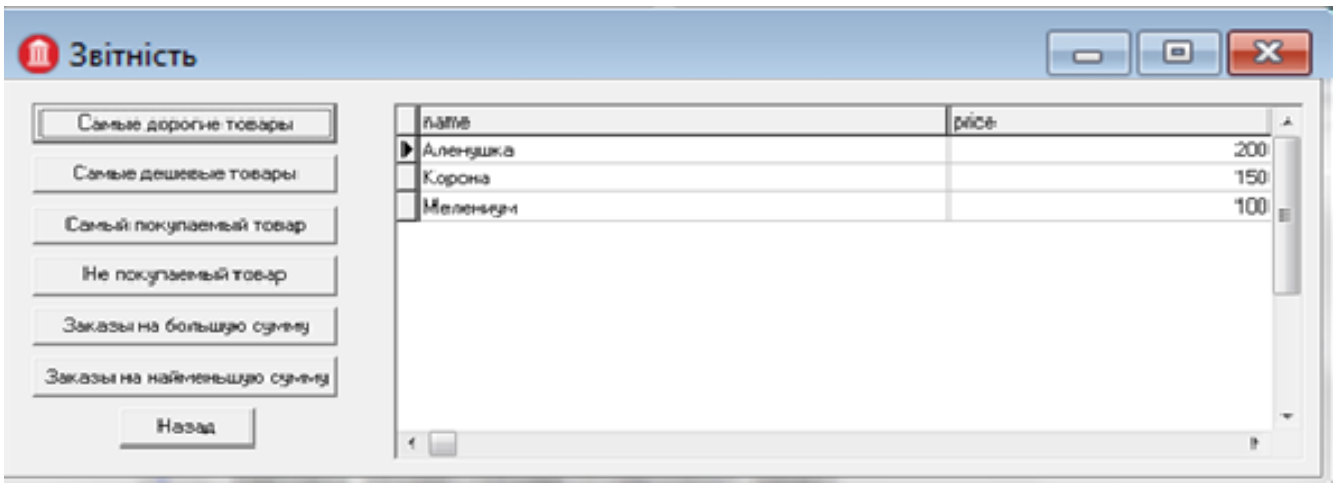

Рис. 2.20. Звітність. Кнопка меню «Дорога продукція» (по убуванню)

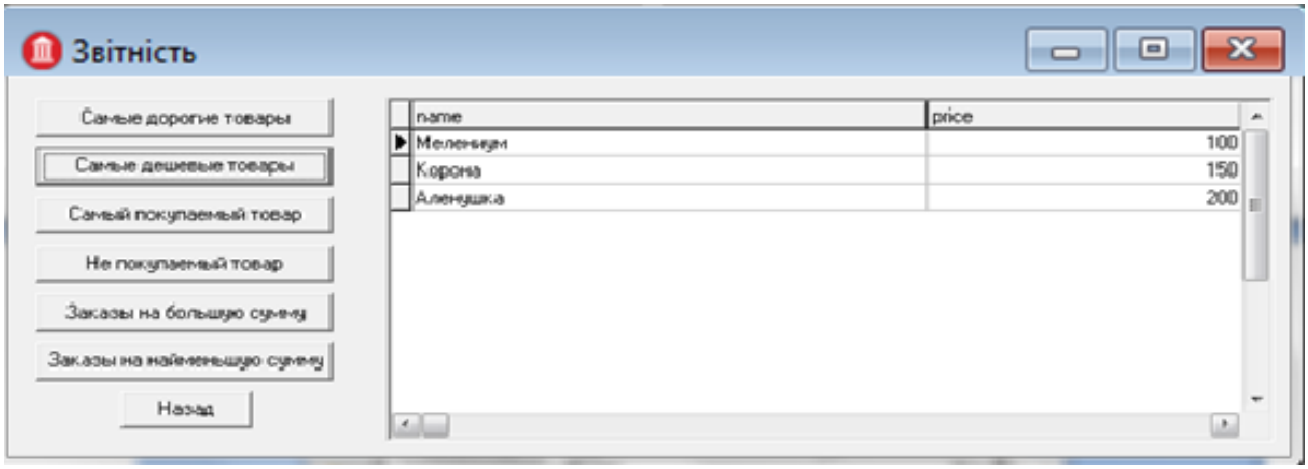

Рис. 2.21. Звітність. Кнопка меню «Дешева продукція» (по зростанню)

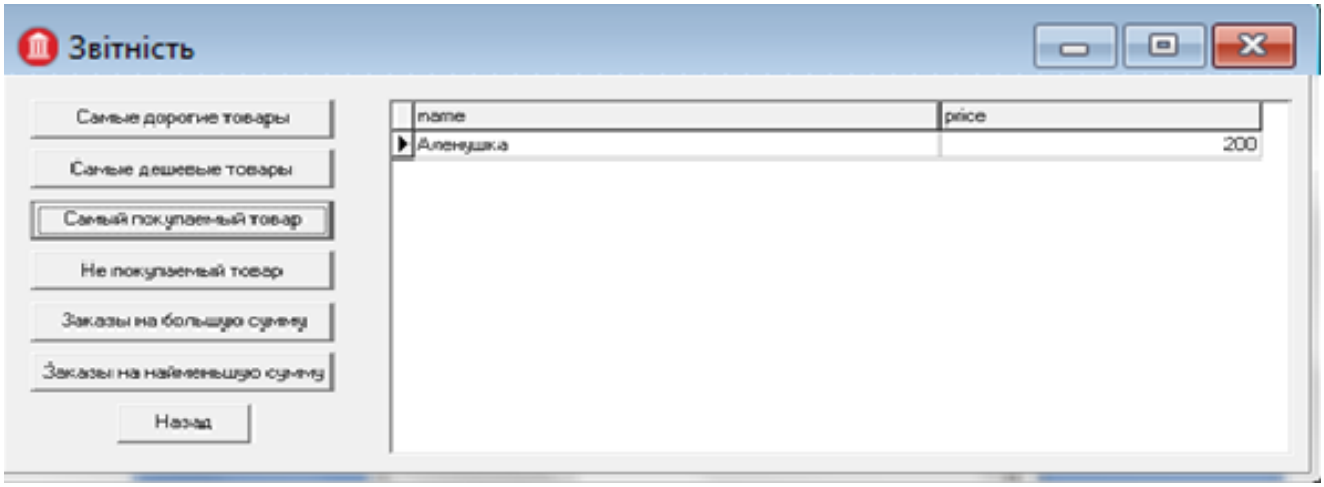

Рис. 2.22. Звітність. Кнопка меню «Попит на товар»

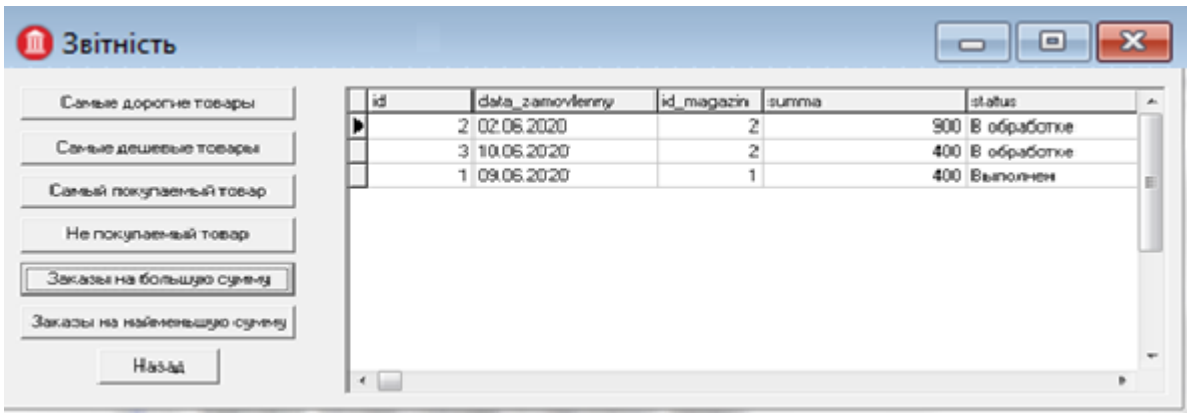

Рис. 2.23. Звітність. Кнопка меню «Замовлення в обробці»

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані розробки програмного забезпечення:

- передбачуване число операторів – 860;
- коефіцієнт складності програми – 1,25;
- коефіцієнт кореляції програми в ході її розробки – 0,1;
- середня годинна заробітна плата програміста, грн/год – 40;
- вартість машино-години ЕОМ, грн/год – 7.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_и + t_a + t_п + t_{отл} + t_д \text{, людино-годин,} \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

$t_и$ – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

$t_п$ – витрати праці на програмування по готовій блок-схемі,

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ,

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C(1 + p), \tag{3.2}$$

де q – передбачуване число операторів,

С– коефіцієнт складності програми,

р – коефіцієнт кореляції програми в ході її розробки.

$$Q = 860 \cdot 1{,}25 \cdot (1 + 0{,}1) = 1182;$$

Витрати праці на вивчення опису задачі $t_и$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85)K}, \text{ людино-годин}, \tag{3.3}$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, В=1.2 ... 1.5,

К – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. до 2 – 0,8.

$$t_u = \frac{1182 \cdot 1{,}2}{85 \cdot 0{,}8} = 22, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)K}; \tag{3.4}$$

$$t_a = \frac{1182}{20 \cdot 0{,}8} = 73, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K}; \quad (3.5)$$

$$t_n = \frac{1182}{25 \cdot 0{,}8} = 59, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K}; \quad (3.6)$$

$$t_{отл} = \frac{1182}{4 \cdot 0{,}8} = 369, \text{ людино-годин,}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1{,}2 \cdot t_{отл}; \quad (3.7)$$

$$t_{отл}^{к} = 1{,}2 \cdot 369 = 443, \text{ людино-годин}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}; \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15...20)K}; \quad (3.9)$$

$$t_{др} = \frac{1182}{15 \cdot 0{,}8} = 98, \text{ людино-годин.}$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др}; \quad (3.10)$$

$$t_{до} = 0{,}75 \cdot 98 = 73$$, людино-годин.

$$t_{д} = 98 + 73 = 172$$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 +22 + 73 +59 + 443 +172 = 821, людино-годин.

В результаті ми розрахували, що в загальній складності необхідно 821 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин,

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

$$З_{ЗП} = 821 \cdot 40 = 32843, \text{грн.}$$

$З_{МВ}$ – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М}, \text{ грн,} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 443 \cdot 7 = 3104 \text{, грн.}$$

$$К_{ПО} = 32843 + 3104 = 35947 \text{ , грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{, мес. ,} \qquad (3.14)$$

де B_k - число виконавців;

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

$$T = \frac{821}{1 \cdot 176} = 4{,}6 \text{ міс.}$$

Таким чином, очікувана тривалість розробки складе 4,6 місяця, а витрати на створення програмного забезпечення 35947 грн.

ВИСНОВКИ

Метою кваліфікаційної роботи бакалавра є розробка інформаційної системи кондитерської фабрики для автоматизації управління бізнес процесами.

Завдання полягає в створенні автоматизованої системи на основи баз даних для роботи з інформацією, пов'язаною з обліком реалізації товарів на малих підприємствах, які займаються реалізацією кондитерської продукції.

Створений додаток надає можливість електронного зберігання даних про товари, продажі, надходження, ведення бази даних, підбиття підсумків з продажу, отримання звітів і формування супутньої ділової документації.

Основні задачі, що вирішує дана розробка:

1. Ведення списків продукції (за видами, типами);
2. Формування прайсу продукції;
3. Облік замовлень (з веденням їх архіву);
4. Формування звітності та статистики.

Програма має дворівневий доступ: адміністратор та користувач (працівник). При запуску програми користувач авторизується за допомогою діалогового вікна з полями - логін та пароль. При невдалій спробі їх заповнення програма сповістить про помилку. Етап авторизації користувача програми надає йому унікальні права та розмежовує їх за рівнем доступу.

Кожному рівню доступні свої унікальні функції.

Розроблена ІС з наявною в ній бази даних дозволяє заносити, зберігати та модифікувати дані про товари підприємства, відправку товарів покупцям, обробляти вказану інформацію з метою отримання необхідних для роботи звітів.

Використані засоби для розробки даної системи – система керування базами даних SQL Server 2012 та інтегрована систему розробки програмного забезпечення C++ Builder6 в операційній системі Windows 10.

Результат використання роботи представлено у вигляді інформаційної системи, яка містить структуру спроектованих таблиць, схему даних зі зв’язками між таблицями, екранні форми для занесення вхідних даних та екранні форми для перегляду результатів їх обробки, запити для аналізу даних, звіти, головну форму.

Розроблена інформаційна система призначена для комерційного застосування в будь-якому з кондитерських підприємств.

В економічному розділі були проведені обчислення трудомісткості та витрат для розробки програмного забезпечення. Для створення даної інформаційної системи знадобиться 4,6 місяця, трудомісткість розробки ПЗ – 821 людино-годин, а витрати на її створення програмного забезпечення - 35947 грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Безбородько І.Г. Все про бази даних - Львів 2013, 2014 - 276с.
2. Благодаров А.В., Гринченко Н.Н., Овечкин Г.В. Клиент-серверные технологии баз данных. Методические указания к лабораторным работам. РГРТУ, Рязань, 2007.
3. Бейлі Л. Вивчаємо SQL. - Львів: Символ-Плюс, 2000. - 120 с.
4. Бенджамін П. Types and Programming Languages/П. Бенджамін. - MIT Press, 2002. - 432 с.
5. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).
6. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.03.2019.
7. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 15.01.2021.
8. Вставка формул в word URL: https://evileg.com/ru/forum/topic/596/ дата звернення: 5.04.2021.
9. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.
10. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

11. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

12. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп’ютерні науки» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

13. Методичні рекомендації щодо написання, оформлення та представлення учнівських науково-дослідницьких робіт учнів – членів Малої академії наук України / Г.Г. Півняк, Л.М. Коротенко, І.М. Удовик, Є.М. Головня – Д.: ДВНЗ «Національний гірничий університет», 2017. – 24 с.

14. Офіційний сайт середовища розробки Visual Studio Code /URL: документація - Режим доступу : https://code.visualstudio.com/docs. дата звернення: 5.04.2021.

15. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

16. Офіційний сайт середовища розробки Visual Studio Code URL: https://code.visualstudio.com/docs дата звернення: 15.01.2021.

17. Шуремов Е.Л., Умнова Э.А., Воропаева Т.В. Автоматизированные информационные системы бухгалтерского учета, анализа, аудита: Учебное пособие для вузов / М: Перспектива, 2001. 363 с.

18. Царев, В.А. Проектирование, анализ и программная реализация структур данных и алгоритмов: Учебное пособие / В.А. Царев, А.Ф. Дробанов. – Череповец., 2007. – 104 с.

19. Фаулер С. «UMLосновы» С. Фаулер, К. Скотт – СПб: Символ-плюс, 2002. – 167.

20. Фуртат Ю.О. «Про вплив адаптивних користувацьких інтерфейсів на надійність та ефективність функціонування автоматизованих систем». – URL: http://ntv.ifmo.ru/file/article/8397.pdf. дата звернення: 5.04.2021.

ДОДАТОК А

КОД ПРОГРАМИ

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "MainForm.h"
#include "LoginForm.h"
#include "ClientsForm.h"
#include "ProvidersForm.h"
#include "GoodsForm.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
#define USER_INVALID -1
#define USER_PURCHASING_DEP 0
#define USER_SALES_DEP 1
#define USER_ACCOUNTING_DEP 2
#define USER_CEO 3
TFormMain *FormMain;
__fastcall TFormMain::TFormMain(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TFormMain::BitBtnExitClick(TObject *Sender)
{
   Close();
}
void __fastcall TFormMain::FormActivate(TObject *Sender)
{
   TFormLogin *FormLogin = new TFormLogin(this);
   try
   {
      FormLogin->SetConnection(ADOConnectionWMS);
                                          FormLogin->ShowModal();
```

```
      int iUserAccessLevel = FormLogin->GetAccessLevel();
      //проверка уровня доступа пользователя
      switch (iUserAccessLevel)
      {
         case USER_PURCHASING_DEP: BtnClients->Enabled = false;
                                   BtnAccounting->Enabled = false;
                                   break;
         case USER_SALES_DEP:      BtnProviders->Enabled = false;
                                   BtnAccounting->Enabled = false;
                                   break;
         case USER_INVALID:        Close();
      }
   }
   __finally
   {
                                          delete FormLogin;
   }
}
void __fastcall TFormMain::FormCreate(TObject *Sender)
{
  ADOConnectionWMS->ConnectionString = WideString("Provider=Microsoft.Jet.OLEDB.4.0; ")
+                    WideString("Persist Security Info=False; Data Source=") +
  WideString(ExtractFilePath(Application->ExeName)+"warehouse.mdb");
}
void __fastcall TFormMain::BtnClientsClick(TObject *Sender)
{
   TFormClients *FormClients = new TFormClients(this);
   try
   {
      FormClients->SetConnection(ADOConnectionWMS);
FormClients->ShowModal();
   }
   __finally
   {
delete FormClients;
```

```
    }
}
void __fastcall TFormMain::BtnProvidersClick(TObject *Sender)
{
    TFormProviders *FormProviders = new TFormProviders(this);
    try
    {
        FormProviders->SetConnection(ADOConnectionWMS);
FormProviders->ShowModal();
    }
    __finally
    {
delete FormProviders;
    }
}
void __fastcall TFormMain::BtnGoodsClick(TObject *Sender)
{
    TFormGoods *FormGoods = new TFormGoods(this);
    try
    {
        FormGoods->SetConnection(ADOConnectionWMS);
FormGoods->ShowModal();
    }
    __finally
    {
delete FormGoods;
    }
}

#include <vcl.h>
#pragma hdrstop
#include "LoginForm.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TFormLogin *FormLogin;
```

```
__fastcall TFormLogin::TFormLogin(TComponent* Owner)
        : TForm(Owner)
{
   m_iUserAccessLevel = -1;
}
void __fastcall TFormLogin::BtnOKClick(TObject *Sender)
{
   try
   {
      //проверка пароля
      QueryLogPass->Close();
      QueryLogPass->SQL->Clear();
      QueryLogPass->SQL->Text="SELECT * FROM Пользователи WHERE Имя LIKE
'"+ComboBoxUser->Text+"'";
      QueryLogPass->Open();
      QueryLogPass->First();
      AnsiString sPassword = QueryLogPass->FieldByName("Пароль")->AsString;
      if (sPassword != EditPassword->Text)
      {
         LabelError->Caption = "Неправильный пароль! Попробуйте еще раз!";
         LabelError->Visible = true;
      }
      else
      {
         m_iUserAccessLevel = QueryLogPass->FieldByName("Доступ")->AsInteger;
         Close();
      }
   }
   catch(...){}
}
void __fastcall TFormLogin::BtnCancelClick(TObject *Sender)
{
   Close();
}
void TFormLogin::SetConnection(TADOConnection *ADOConnectionWMS)
```

```
{
   QueryLogPass->Connection = ADOConnectionWMS;
}
void __fastcall TFormLogin::FormActivate(TObject *Sender)
{
   //вывод списка пользователей из таблицы "Пользователи" в  ComboBox
   ComboBoxUser->Items->Clear();
   try
   {
      QueryLogPass->SQL->Clear();
      QueryLogPass->SQL->Add("SELECT * FROM Пользователи");
      QueryLogPass->Open();
      QueryLogPass->First();
      while(!QueryLogPass->Eof)
      {
         ComboBoxUser->Items->Add(QueryLogPass->FieldByName("Имя")->AsString);
         QueryLogPass->Next();
      }
      ComboBoxUser->ItemIndex = 0;
    }
    catch(...){}
}
int TFormLogin::GetAccessLevel()
{
  return m_iUserAccessLevel;

#include <vcl.h>
#pragma hdrstop
#include "ClientsForm.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TFormClients *FormClients;
__fastcall TFormClients::TFormClients(TComponent* Owner)
      : TForm(Owner)
{
```

```
}
void TFormClients::SetConnection(TADOConnection *ADOConnectionWMS)
{
   QueryClients->Connection = ADOConnectionWMS;
   TableClients->Connection = ADOConnectionWMS;
   TableClients->TableName = "Клиенты";
   TableClients->Active = true;
}
void __fastcall TFormClients::FormActivate(TObject *Sender)
{
    UpdateColumnsWidth();
    UpdateFields();
}
void TFormClients::UpdateFields()
{
    EditFIO->Text = TableClients->FieldByName("ФИО")->AsString;
    EditPassData->Text = TableClients->FieldByName("Паспорт")->AsString;
    EditCity->Text = TableClients->FieldByName("Город")->AsString;
    EditAdres->Text = TableClients->FieldByName("Адрес")->AsString;
    EditPhone->Text = TableClients->FieldByName("Телефон")->AsString;
    EditEMail->Text = TableClients->FieldByName("E-mail")->AsString;
}
void TFormClients::UpdateColumnsWidth()
{
   DBGridClients->Columns->Items[0]->Width = 30;
   for (int i = 0;i < DBGridClients->Columns->Count; i++)
      DBGridClients->Columns->Items[i]->Width = 80;
}
void __fastcall TFormClients::BtnAddClick(TObject *Sender)
{
  try
  {
     QueryClients->Close();
     QueryClients->SQL->Clear();
     QueryClients->SQL->Add("INSERT INTO Клиенты (ФИО, Телефон");
```

```
      QueryClients->SQL->Add("VALUES (:Param1, :Param2, :Param3, :Param4, :Param5,
:Param6)");
      QueryClients->Parameters->ParamByName("Param1")->Value = EditFIO->Text;
      QueryClients->Parameters->ParamByName("Param2")->Value = EditPassData->Text;
      QueryClients->Parameters->ParamByName("Param3")->Value = EditCity->Text;
      QueryClients->Parameters->ParamByName("Param4")->Value = EditAdres->Text;
      QueryClients->Parameters->ParamByName("Param5")->Value = EditPhone->Text;
      QueryClients->Parameters->ParamByName("Param6")->Value = EditEMail->Text;
      QueryClients->ExecSQL();
   }
   catch(...){}
   TableClients->Active = false;
   TableClients->Active = true;
   UpdateColumnsWidth();
   UpdateFields();
}
void __fastcall TFormClients::BtnDeleteClick(TObject *Sender)
{
    DBGridClients->DataSource->DataSet->Delete();
}
void __fastcall TFormClients::TableClientsAfterScroll(TDataSet *DataSet)
{
    UpdateFields();
}
void __fastcall TFormClients::BtnEditClick(TObject *Sender)
{
   try
   {
      QueryClients->Close();
      QueryClients->SQL->Clear();
      QueryClients->SQL->Add("WHERE (ФИО=:Param1) AND (Паспорт=:Param2) AND
(Город=:Param3) AND (Адрес=:Param4) AND (Телефон=:Param5) AND ([E-mail]=:Param6)");
      QueryClients->Parameters->ParamByName("newParam1")->Value = EditFIO->Text;
      QueryClients->Parameters->ParamByName("newParam2")->Value = EditPassData->Text;
      QueryClients->Parameters->ParamByName("newParam3")->Value = EditCity->Text;
```

```
      QueryClients->Parameters->ParamByName("newParam4")->Value = EditAdres->Text;
      QueryClients->Parameters->ParamByName("newParam5")->Value = EditPhone->Text;
      QueryClients->Parameters->ParamByName("newParam6")->Value = EditEMail->Text;
      QueryClients->Parameters->ParamByName("Param1")->Value = TableClients-
>FieldByName("ФИО")->AsString;
      QueryClients->Parameters->ParamByName("Param2")->Value = TableClients-
>FieldByName("")->AsString;
      QueryClients->Parameters->ParamByName("Param3")->Value = TableClients-
>FieldByName("")->AsString;
      QueryClients->Parameters->ParamByName("Param4")->Value = TableClients-
>FieldByName("Адрес")->AsString;
      QueryClients->Parameters->ParamByName("Param5")->Value = TableClients-
>FieldByName("Телефон")->AsString;
      QueryClients->Parameters->ParamByName("Param6")->Value = TableClients-
>FieldByName("E-mail")->AsString;
      QueryClients->ExecSQL();
   }
   catch(...){}
   TableClients->Active = false;
   TableClients->Active = true;
   UpdateColumnsWidth();
   UpdateFields();
}
void __fastcall TFormClients::BtnNewOrderClick(TObject *Sender)
{
    const int xlWBATWorksheet = -4167;
    Screen->Cursor = crHourGlass;
    DBGridClients->DataSource->DataSet->DisableControls();
    TBookmark bm = DBGridClients->DataSource->DataSet->GetBookmark();
    DBGridClients->DataSource->DataSet->First();
     // создаём объект Excel
    Variant XLApp = CreateOleObject("Excel.Application");
    Variant XLBook = XLApp.OlePropertyGet("Workbooks").OlePropertyGet("Add",
xlWBATWorksheet);
    Variant XLSheet = XLBook.OlePropertyGet("Worksheets", 1);
```

```
    XLSheet.OlePropertySet("Name", "Grid Data");
    mem->Clear();
     // добавляем информацию для имён колонок
    AnsiString sline = "";
    for(int col = 0; col < DBGridClients->FieldCount; ++col)
    {
       sline += DBGridClients->Fields[col]->DisplayLabel + (char)9;
    }
    mem->Lines->Add(sline);
     // получаем данные из memo
    for(int row = 0; row < DBGridClients->DataSource->DataSet->RecordCount; ++row)
    {
       sline = "";
       for(int col = 0; col < DBGridClients->FieldCount; ++col) {
          sline += DBGridClients->Fields[col]->AsString + (char)9;
       }
       mem->Lines->Add(sline);
       DBGridClients->DataSource->DataSet->Next();
    }
    // копируем данные в clipboard
    mem->SelectAll();
    mem->CopyToClipboard();
    XLSheet.OleProcedure("Paste");
    XLApp.OlePropertySet("Visible", true);
    DBGridClients->DataSource->DataSet->GotoBookmark(bm);
    DBGridClients->DataSource->DataSet->FreeBookmark(bm);
    DBGridClients->DataSource->DataSet->EnableControls();
    Screen->Cursor = crDefault;
}

#include <vcl.h>
#pragma hdrstop
#include "GoodsForm.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
```

```
TFormGoods *FormGoods;
__fastcall TFormGoods::TFormGoods(TComponent* Owner)       : TForm(Owner)
{
}
void TFormGoods::SetConnection(TADOConnection *ADOConnectionWMS)
{
   QueryGoods->Connection = ADOConnectionWMS;
   TableGoods->Connection = ADOConnectionWMS;
   TableGoods->TableName = "Товары";
   TableGoods->Active = true;
}
void TFormGoods::UpdateFields()
{
    EditName->Text = TableGoods->FieldByName("Название товара")->AsString;
    EditCount->Text = TableGoods->FieldByName("Количество товара")->AsString;
    EditPrice->Text = TableGoods->FieldByName("Цена за единицу")->AsString;
    EditLeft->Text = TableGoods->FieldByName("Остаток")->AsString;
    ComBoxProvider->Text = TableGoods->FieldByName("Поставщик")->AsString;
}
void TFormGoods::UpdateColumnsWidth()
{
   for (int i = 0;i < DBGridGoods->Columns->Count; i++)
      DBGridGoods->Columns->Items[i]->Width = 80;
}
void __fastcall TFormGoods::FormActivate(TObject *Sender)
{
   ComBoxProvider->Items->Clear();
   try
   {
      QueryGoods->SQL->Clear();
      QueryGoods->SQL->Add("SELECT * FROM Поставщики");
      QueryGoods->Open();
      QueryGoods->First();
      while(!QueryGoods->Eof)
      {
```

```
            ComBoxProvider->Items->Add(QueryGoods->FieldByName("Название")->AsString);
            QueryGoods->Next();
        }
        ComBoxProvider->ItemIndex = 0;
    }
    catch(...){}
    UpdateColumnsWidth();
    UpdateFields();
}
void __fastcall TFormGoods::BtnAddClick(TObject *Sender)
{
  try
  {
      QueryGoods->Close();
      QueryGoods->SQL->Clear();
      QueryGoods->SQL->Add("INSERT INTO Товары ([Название товара], [Количество
товара], [Цена за единицу], Остаток, Поставщик)");
      QueryGoods->SQL->Add("VALUES (:Param1, :Param2, :Param3, :Param4, :Param5)");
      QueryGoods->Parameters->ParamByName("Param1")->Value = EditName->Text;
      QueryGoods->Parameters->ParamByName("Param2")->Value = EditCount->Text;
      QueryGoods->Parameters->ParamByName("Param3")->Value = EditPrice->Text;
      QueryGoods->Parameters->ParamByName("Param4")->Value = EditLeft->Text;
      QueryGoods->Parameters->ParamByName("Param5")->Value = ComBoxProvider->Text;
      QueryGoods->ExecSQL();
  }
  catch(...){}
  TableGoods->Active = false;
  TableGoods->Active = true;
  UpdateColumnsWidth();
  UpdateFields();
}
void __fastcall TFormGoods::BtnEditClick(TObject *Sender)
{
  try
  {
```

```
      QueryGoods->Close();
      QueryGoods->SQL->Clear();
      QueryGoods->SQL->Add("UPDATE Товары SET [Название товара]=:newParam1,
[Количество товара]=:newParam2, [Цена за единицу]=:newParam3, Остаток=:newParam4,
Поставщик=:newParam5");
      QueryGoods->SQL->Add("WHERE ([Название товара]=:Param1) AND ([Количество
товара]=:Param2) AND ([Цена за единицу]=:Param3) AND (Остаток=:Param4) AND
(Поставщик=:Param5)");
      QueryGoods->Parameters->ParamByName("newParam1")->Value = EditName->Text;
      QueryGoods->Parameters->ParamByName("newParam2")->Value = EditCount->Text;
      QueryGoods->Parameters->ParamByName("newParam3")->Value = EditPrice->Text;
      QueryGoods->Parameters->ParamByName("newParam4")->Value = EditLeft->Text;
      QueryGoods->Parameters->ParamByName("newParam5")->Value = ComBoxProvider->Text;
      QueryGoods->Parameters->ParamByName("Param1")->Value = TableGoods-
>FieldByName("Название товара")->AsString;
      QueryGoods->Parameters->ParamByName("Param2")->Value = TableGoods-
>FieldByName("Количество товара")->AsString;
      QueryGoods->Parameters->ParamByName("Param3")->Value = TableGoods-
>FieldByName("Цена за единицу")->AsString;
      QueryGoods->Parameters->ParamByName("Param4")->Value = TableGoods-
>FieldByName("Остаток")->AsString;
      QueryGoods->Parameters->ParamByName("Param5")->Value = TableGoods-
>FieldByName("Поставщик")->AsString;
      QueryGoods->ExecSQL();
   }
   catch(...){}
   TableGoods->Active = false;
   TableGoods->Active = true;
   UpdateColumnsWidth();
   UpdateFields();
}
void __fastcall TFormGoods::BtnDeleteClick(TObject *Sender)
{
    DBGridGoods->DataSource->DataSet->Delete();
}
```

```
void __fastcall TFormGoods::TableGoodsAfterScroll(TDataSet *DataSet)
{
   UpdateFields();
}
void __fastcall TFormGoods::BtnNewOrderClick(TObject *Sender)
{
   const int xlWBATWorksheet = -4167;
   Screen->Cursor = crHourGlass;
   DBGridGoods->DataSource->DataSet->DisableControls();
   TBookmark bm = DBGridGoods->DataSource->DataSet->GetBookmark();
   DBGridGoods->DataSource->DataSet->First();
   Variant XLApp = CreateOleObject("Excel.Application");
   Variant XLBook = XLApp.OlePropertyGet("Workbooks").OlePropertyGet("Add",
xlWBATWorksheet);
   Variant XLSheet = XLBook.OlePropertyGet("Worksheets", 1);
   XLSheet.OlePropertySet("Name", "Grid Data");
   mem->Clear();
    // добавляем информацию для имён колонок
   AnsiString sline = "";
   for(int col = 0; col < DBGridGoods->FieldCount; ++col)
   {
      sline += DBGridGoods->Fields[col]->DisplayLabel + (char)9;
   }
   mem->Lines->Add(sline);
    // получаем данные из memo
   for(int row = 0; row < DBGridGoods->DataSource->DataSet->RecordCount; ++row)
   {
      sline = "";
      for(int col = 0; col < DBGridGoods->FieldCount; ++col) {
         sline += DBGridGoods->Fields[col]->AsString + (char)9;
      }
      mem->Lines->Add(sline);
      DBGridGoods->DataSource->DataSet->Next();
   }
   // копируем данные в clipboard
```

```
    mem->SelectAll();
    mem->CopyToClipboard();
    XLSheet.OleProcedure("Paste");
    XLApp.OlePropertySet("Visible", true);
    DBGridGoods->DataSource->DataSet->GotoBookmark(bm);
    DBGridGoods->DataSource->DataSet->FreeBookmark(bm);
    DBGridGoods->DataSource->DataSet->EnableControls();
    Screen->Cursor = crDefault;
}

#include <vcl.h>
#pragma hdrstop
#include "ProvidersForm.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TFormProviders *FormProviders;
__fastcall TFormProviders::TFormProviders(TComponent* Owner)  : TForm(Owner)
{
}
void TFormProviders::SetConnection(TADOConnection *ADOConnectionWMS)
{
    QueryProviders->Connection = ADOConnectionWMS;
    TableProviders->Connection = ADOConnectionWMS;
    TableProviders->TableName = "Поставщики";
    TableProviders->Active = true;
}
void __fastcall TFormProviders::FormActivate(TObject *Sender)
{
     UpdateColumnsWidth();
     UpdateFields();
}
void TFormProviders::UpdateFields()
{
     EditAdres->Text = TableProviders->FieldByName("Юридический адрес")->AsString;
     EditName->Text = TableProviders->FieldByName("Название")->AsString;
```

```
    EditAccount->Text = TableProviders->FieldByName("№ счета")->AsString;
    EditPhone->Text = TableProviders->FieldByName("Телефон")->AsString;
    EditEMail->Text = TableProviders->FieldByName("E-mail")->AsString;
}
void TFormProviders::UpdateColumnsWidth()
{
   DBGridProviders->Columns->Items[0]->Width = 30;
   for (int i = 1;i < DBGridProviders->Columns->Count; i++)
      DBGridProviders->Columns->Items[i]->Width = 100;
}
void __fastcall TFormProviders::BtnAddClick(TObject *Sender)
{
  try
  {
     QueryProviders->Close();
     QueryProviders->SQL->Clear();
     QueryProviders->SQL->Add("VALUES (:Param1, :Param2, :Param3, :Param4, :Param5)");
     QueryProviders->Parameters->ParamByName("Param1")->Value = EditAdres->Text;
     QueryProviders->Parameters->ParamByName("Param2")->Value = EditName->Text;
     QueryProviders->Parameters->ParamByName("Param3")->Value = EditAccount->Text;
     QueryProviders->Parameters->ParamByName("Param4")->Value = EditPhone->Text;
     QueryProviders->Parameters->ParamByName("Param5")->Value = EditEMail->Text;
     QueryProviders->ExecSQL();
  }
  catch(...){}
  TableProviders->Active = false;
  TableProviders->Active = true;
  UpdateColumnsWidth();
  UpdateFields();
}
void __fastcall TFormProviders::BtnEditClick(TObject *Sender)
{
  try
  {
     QueryProviders->Close();
```

```
      QueryProviders->SQL->Clear();
      QueryProviders->SQL->Add("WHERE ([Юридический адрес]=:Param1) AND
(Название=:Param2) AND ([№ счета]=:Param3) AND (Телефон=:Param4) AND ([E-
mail]=:Param5)");
      QueryProviders->Parameters->ParamByName("newParam1")->Value = EditAdres->Text;
      QueryProviders->Parameters->ParamByName("newParam2")->Value = EditName->Text;
      QueryProviders->Parameters->ParamByName("newParam3")->Value = EditAccount->Text;
      QueryProviders->Parameters->ParamByName("newParam4")->Value = EditPhone->Text;
      QueryProviders->Parameters->ParamByName("newParam5")->Value = EditEMail->Text;
      QueryProviders->Parameters->ParamByName("Param1")->Value = TableProviders-
>FieldByName("Юридический адрес")->AsString;
      QueryProviders->Parameters->ParamByName("Param2")->Value = TableProviders-
>FieldByName("Название")->AsString;
      QueryProviders->Parameters->ParamByName("Param3")->Value = TableProviders-
>FieldByName("№ счета")->AsString;
      QueryProviders->Parameters->ParamByName("Param4")->Value = TableProviders-
>FieldByName("Телефон")->AsString;
      QueryProviders->Parameters->ParamByName("Param5")->Value = TableProviders-
>FieldByName("E-mail")->AsString;
      QueryProviders->ExecSQL();
   }
   catch(...){}
   TableProviders->Active = false;
   TableProviders->Active = true;
   UpdateColumnsWidth();
   UpdateFields();
}
void __fastcall TFormProviders::BtnDeleteClick(TObject *Sender)
{
    DBGridProviders->DataSource->DataSet->Delete();
}
void __fastcall TFormProviders::TableProvidersAfterScroll(
      TDataSet *DataSet)
{
    UpdateFields();
```

```
}
void __fastcall TFormProviders::BtnNewOrderClick(TObject *Sender)
{
   const int xlWBATWorksheet = -4167;
   Screen->Cursor = crHourGlass;
   DBGridProviders->DataSource->DataSet->DisableControls();
   TBookmark bm = DBGridProviders->DataSource->DataSet->GetBookmark();
   DBGridProviders->DataSource->DataSet->First();
   Variant XLApp = CreateOleObject("Excel.Application");
   Variant XLBook = XLApp.OlePropertyGet("Workbooks").OlePropertyGet("Add",
xlWBATWorksheet);
   Variant XLSheet = XLBook.OlePropertyGet("Worksheets", 1);
   XLSheet.OlePropertySet("Name", "Grid Data");
   mem->Clear();
    // добавляем информацию для имён колонок
   AnsiString sline = "";
   for(int col = 0; col < DBGridProviders->FieldCount; ++col)
   {
      sline += DBGridProviders->Fields[col]->DisplayLabel + (char)9;
   }
   mem->Lines->Add(sline);
    // получаем данные из memo
   for(int row = 0; row < DBGridProviders->DataSource->DataSet->RecordCount; ++row)
   {
      sline = "";
      for(int col = 0; col < DBGridProviders->FieldCount; ++col) {
         sline += DBGridProviders->Fields[col]->AsString + (char)9;
      }
      mem->Lines->Add(sline);
      DBGridProviders->DataSource->DataSet->Next();
   }
   // копируем данные в clipboard
   mem->SelectAll();
   mem->CopyToClipboard();
   XLSheet.OleProcedure("Paste");
```

```
    XLApp.OlePropertySet("Visible", true);
    DBGridProviders->DataSource->DataSet->GotoBookmark(bm);
    DBGridProviders->DataSource->DataSet->FreeBookmark(bm);
    DBGridProviders->DataSource->DataSet->EnableControls();
    Screen->Cursor = crDefault;
}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
namespace Ui {class MainWindow;}
class MainWindow : public QMainWindow
{    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private slots:
    void on_pushButton_clicked();
    void on_pushButton_2_clicked();
    void on_pushButton_3_clicked();
    void on_pushButton_4_clicked();
    void on_pushButton_5_clicked();
    void on_pushButton_6_clicked();
    void on_action_2_triggered();
    void on_action_triggered();
private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "basa_prod.h"
```

```
#include "QSqlQuery"
#include "imt.h"
#include "debq.h"
#include "global.h"
#include "text.h"
MainWindow::MainWindow(QWidget *parent) :
   QMainWindow(parent),
   ui(new Ui::MainWindow)
{
   ui->setupUi(this);
   QSqlDatabase db;
      db = QSqlDatabase::addDatabase("QSQLITE");
      db.setDatabaseName("\Basa.db");
      // C:/text/
      db.open();
      setWindowTitle( tr("Меню ") );
      sost=0;
}
MainWindow::~MainWindow()
{
   delete ui;
}
void MainWindow::on_pushButton_clicked()
{    basa_prod*bp=new  basa_prod();
   bp->showFullScreen();
   bp->show();
   sost=1;
   this->close();
#include <vcl.h>
#pragma hdrstop
USEFORM("MainForm.cpp", fmMain);
USEFORM("ParamForm.cpp", fmParam);
USEFORM("KeyBoardForm.cpp", fmKeyBoard);
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
```

```
	try
	{
		Application->Initialize();
		Application->CreateForm(__classid(TfmMain), &fmMain);
		Application->CreateForm(__classid(TfmParam), &fmParam);
		Application->CreateForm(__classid(TfmKeyBoard), &fmKeyBoard);
		Application->Run();
	}
	catch (Exception &exception)
	{
		Application->ShowException(&exception);
	}
	catch (...)
	{
		try
		{
			throw Exception("");
		}
		catch (Exception &exception)
		{
			Application->ShowException(&exception);
		}
	}
	return 0;
}

#ifndef KeyBoardFormH
#define KeyBoardFormH
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
class TfmKeyBoard : public TForm
{
```

```
__published:   // IDE-managed Components
        TButton *Q;
        TButton *Button1;
        TButton *Button2;
        TButton *Button3;
        TButton *Button4;
        TButton *Button5;
        TButton *Button6;
        TButton *Button7;
        TButton *Button8;
        TButton *Button9;
        TButton *Button10;
        TButton *Button11;
        TButton *Button12;
        TButton *Button13;
        TButton *Button14;
        TButton *Button15;

        TPopupMenu *PopupMenu1;
        TMenuItem *N1;
        TMenuItem *N2;
        void __fastcall QClick(TObject *Sender);
        void __fastcall FormShow(TObject *Sender);
        void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
        void __fastcall Button30Click(TObject *Sender);
        void __fastcall Button31Click(TObject *Sender);
        void __fastcall Button29Click(TObject *Sender);
        void __fastcall Button28Click(TObject *Sender);
        void __fastcall FormMouseUp(TObject *Sender, TMouseButton Button,
          TShiftState Shift, int X, int Y);
        void __fastcall N2Click(TObject *Sender);
        void __fastcall N1Click(TObject *Sender);
private:          // User declarations
public:           // User declarations
        __fastcall TfmKeyBoard(TComponent* Owner);
```

```
        void AddChar(char, TEdit*);
};
extern PACKAGE TfmKeyBoard *fmKeyBoard;
#endif

#ifndef MainFormH
#define MainFormH
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <stdio.h>
#include <Menus.hpp>
#include <Dialogs.hpp>
#include <ExtCtrls.hpp>
#include <Buttons.hpp>
#include <Graphics.hpp>
#include <jpeg.hpp>
struct TTask{  //задание
   AnsiString mTask;
   AnsiString mPromt;
   AnsiString mAnswer[5];
   AnsiString mPromtTranslate;
   AnsiString mAnswerTranslate;
   int mLevel;
   int mStep;
   int mNumAnswer;
   TTask () ;
   void   Clear();
   void ReadTask(AnsiString);
   void ReadPromt(AnsiString);
   void ReadAnswer(AnsiString);
   void ReadPromtTranslate(AnsiString);
   void ReadAnswerTranslate(AnsiString);
```

```
    AnsiString Symbol(AnsiString);
    bool CheckAnswer(AnsiString);
    TTask& operator=( TTask& );
    bool operator==( TTask& );
    void Save(FILE*);
 };
class TfmMain : public TForm
{
__published:   // IDE-managed Components
        TPageControl *PageControl1;
        TTabSheet *TabSheet1;
        TTabSheet *TabSheet2;
        TTabSheet *TabSheet3;
        TMainMenu *MainMenu1;
        TMenuItem *N1;
        TMenuItem *N2;
        TMenuItem *N3;
        TMenuItem *N4;
        TOpenDialog *OpenDialog1;
        TMenuItem *N5;
        TMemo *Memo1;
        TPanel *Panel1;
        TBitBtn *BitBtn1;
        TComboBox *ComboBox1;
        TBitBtn *BitBtn2;
        TBitBtn *BitBtn3;
        TBitBtn *BitBtn4;
        TBitBtn *BitBtn5;
        TBitBtn *BitBtn6;
        TScrollBox *ScrollBox1;
        TGroupBox *GroupBox4;
        TGroupBox *GroupBox3;
        TEdit *ebPromt;
        TGroupBox *GroupBox2;
        TLabel *Label2;
```

```
TLabel *Label3;
TLabel *Label4;
TLabel *Label5;
TLabel *Label6;
TBitBtn *BitBtn7;
TMenuItem *N6;
TSaveDialog *SaveDialog1;
TMenuItem *N7;
TMenuItem *N8;
TMenuItem *N9;
TMenuItem *N10;
TBitBtn *BitBtn8;
TStaticText *StaticText1;
TStaticText *StaticText2;
TEdit *ebkAnswer3;
TMenuItem *N11;
TMenuItem *N12;
TStatusBar *StatusBar1;
TPopupMenu *PopupMenu1;
TMenuItem *N13;
TMenuItem *N14;
TPanel *Panel2;
TButton *Button18;
TButton *Button29;
TButton *Q;
TButton *Button4;
TImage *Image1;
TEdit *ebAnswer;
TBitBtn *Button3;
TBitBtn *Button2;
TBitBtn *Button1;
TImage *Image2;
TLabel *lbTask2;
void __fastcall N4Click(TObject *Sender);
void __fastcall Button1Click(TObject *Sender);
```

```
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall N5Click(TObject *Sender);
        void __fastcall BitBtn1Click(TObject *Sender);
        void __fastcall BitBtn3Click(TObject *Sender);
        void __fastcall BitBtn4Click(TObject *Sender);
        void __fastcall BitBtn2Click(TObject *Sender);
        void __fastcall ComboBox1Change(TObject *Sender);
        void __fastcall BitBtn5Click(TObject *Sender);
        void __fastcall BitBtn6Click(TObject *Sender);
        void __fastcall BitBtn7Click(TObject *Sender);
        void __fastcall FormPaint(TObject *Sender);
        void __fastcall FormCanResize(TObject *Sender, int &NewWidth,
          int &NewHeight, bool &Resize);
        void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
        void __fastcall N6Click(TObject *Sender);
        void __fastcall N7Click(TObject *Sender);
        void __fastcall N8Click(TObject *Sender);
        void __fastcall N9Click(TObject *Sender);
        void __fastcall N10Click(TObject *Sender);
        void __fastcall BitBtn8Click(TObject *Sender);
        void __fastcall N11Click(TObject *Sender);
private:        // User declarations
        TTask mCurTask;  //текущий
        TTask maTask[2000];
        int mTasksCount;   //все
        int mTaskNumber;
        int mBadAnswer;
        int mTaskMode;
        int mTaskProbability;
        int mMainHeapSize;
        int mStepSize;
        int mnRightAnswer;
        int mnWrongAnswer;
        int mEditTask;
```

```
        int mPromtCounter;
        bool mRightAnswer;
        bool mSave;
        bool mSSFont;
        AnsiString mFileName;
        int GetTaskFromFullHeap();
        int GetTaskNumber();
        int MainHeapSize(bool _s=false);
        int FullHeapSize();
        int FindCurTask();
        int HeapSize(){return mMainHeapSize*10+10;}
        int StepSize(){return mStepSize*5+5;}
          void SetText(AnsiString, int);
        void Read(AnsiString);
        void ShowTask(int);
        void SaveTasks(AnsiString);
        void ShowTaskMode(void);
        void NextTask(void);
        void TaskEstimate(void);
        void CheckAnsver(void);
        void Prompt(void);
public:             // User declarations
        __fastcall TfmMain(TComponent* Owner);

        void StepLeft();
        void StepRight();
        void DeleteChar();
        void BackSpace();
        void InsertChar(char);
};
extern PACKAGE TfmMain *fmMain;
#endif

#ifndef ParamFormH
#define ParamFormH
```

```
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
class TfmParam : public TForm
{
__published:   // IDE-managed Components
        TLabel *Label1;
        TLabel *Label5;
        TLabel *Label6;
        TComboBox *ComboBox1;
        TComboBox *ComboBox2;
        TComboBox *ComboBox3;
        TButton *Button1;
        TButton *Button2;
        TLabel *Label2;
        TComboBox *ComboBox4;
        TLabel *Label3;
        TComboBox *ComboBox5;
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall ComboBox4Change(TObject *Sender);
private:          // User declarations
public:           // User declarations
        bool mRet;
        bool Execute(void);
        __fastcall TfmParam(TComponent* Owner);
};
extern PACKAGE TfmParam *fmParam;
#endif
#include <stdio.h>
#include <vcl.h>
#pragma hdrstop
#include "ParamForm.h"
```

```
#include "KeyBoardForm.h"
#include "MainForm.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TfmMain *fmMain;
AnsiString fnHead(AnsiString _l) ;
AnsiString fnTail(AnsiString _l)   // убирает до пробела / табуляции
{
 _l=_l.Trim();
 AnsiString h = fnHead(_l);
 int n = h.Length();
 int l =  _l.Length();
 AnsiString ret = _l.Delete(1,n);
 return ret.Trim();
}
AnsiString fnTail(AnsiString _l,int _k)  // убирает  с 1 задания цифры
{
AnsiString ret = fnTail(_l);
if (ret.IsEmpty()) return ret;
if (_k<1) return ret;
 return fnTail(ret,_k-1);
}
AnsiString fnHead(AnsiString _l)     // убирает  после пробела / табуляции
{
 AnsiString str = "";
 _l=_l.Trim();
 int l = _l.Length();
 for(int i=1;i<=l;i++){
  if ((_l[i] == ' ') || (_l[i] == '\t' )) return str;
  str=str+_l[i];
 }
 return str ;
}
AnsiString fnSymbol(AnsiString k , AnsiString  _str){
 _str=_str.UpperCase();
```

```
  int r = _str.Length();
int t = k.Length();
AnsiString s ;
for(int j =1,i =1 ;j <=t;j++,i++){
        s += k[j];
        if(i>r) return s;
        if(_str[i]!=k[j]){
        return s;}
}
 return s;
}
AnsiString fnReplaseSimb(AnsiString _str, AnsiString _old, AnsiString _new)//замена # на __
{
 int n= _str.Pos(_old);
 if(!n) return _str;
 AnsiString s1 = _str.SubString(1,n-1);
 int l1 = _str.Length();
 int l2 = _old.Length();
 AnsiString s2 = _str.SubString(n+1,l1-n+1-l2);
 return s1+_new+s2;
}
void  TTask ::Save(FILE* _out) {
 fprintf(_out," 1 %d %d %s \n ",mStep,mLevel,  mTask.c_str());
  fprintf(_out," 2 %s \n ",  mPromt.c_str());
   for (int i =0; i<5;i++){if (mAnswer[i]!="")
   fprintf(_out," %d %s \n ",3,  mAnswer[i].c_str()); }
   fprintf(_out," %d %s \n ",4,  mPromtTranslate.c_str());
   fprintf(_out," %d %s \n ",5,  mAnswerTranslate.c_str());
   fprintf(_out,"\n");
}
//---------------------------------------------------------------------------
__fastcall TfmMain::TfmMain(TComponent* Owner) // конструктор(при создании
:TForm(Owner)
{
 mTaskProbability=8;
```

```
 mTaskMode=0;
 mBadAnswer = 5;
 mMainHeapSize=3;
 mStepSize=3;
 mnWrongAnswer =0;
 mnRightAnswer = 0;
 mTaskNumber=-1;
 mTasksCount=0;
 mEditTask =0;
 mPromtCounter =0;
 ForceCurrentDirectory=true;
 mSave=false;
 randomize();
 mSSFont=false;
 for(int i=0;i<Screen->Fonts->Count;i++){
  if(Screen->Fonts->Strings[i] ==AnsiString ("MS Sans Serif")) mSSFont=true;
 }
 if(!mSSFont){
  AddFontResource("Microsoft Sans Serif.ttf");
 }
}
void __fastcall TfmMain::N4Click(TObject *Sender)
{
 Close();
}
void TfmMain::TaskEstimate()
{
if(mRightAnswer) mnRightAnswer++; else mnWrongAnswer++;
int t = FindCurTask();
int k,l,p;
for(int i =0; i<mTasksCount; i++){
maTask[i].mStep= maTask[i].mStep?maTask[i].mStep-1:0;
}
if (t<0) return;
p = MainHeapSize();
```

```
maTask[t].mStep = StepSize()-1;
if(mRightAnswer) {
k = maTask [t].mLevel;
if(k){
if(k>0){
maTask [t].mLevel--;
}
else{
if(p<HeapSize()){
maTask [t].mLevel= -(maTask [t].mLevel+1);
}else{
maTask [t].mLevel++;
}
}
}
}else {
if(p<HeapSize()) {
maTask [t].mLevel = mBadAnswer;}
else {
maTask [t].mLevel = maTask [t].mLevel<0? -mBadAnswer:mBadAnswer;
}
}
 l=0;
 for(int i =0; i<mTasksCount; i++){ l+= !maTask[i].mLevel;}
 l= 100*l/mTasksCount;
 AnsiString s;
 s= "Оцінка - ВІДМІННО" ;
 if(l<90) s = "Оцінка - ДОБРЕ" ;
 if(l<75) s = "Оцінка - ЗАДОВІЛЬНО" ;
 if(l<60) s = "Оцінка - ПОГАНО" ;
 StatusBar1->Panels->Items[4]->Text=s;
}
void TfmMain::SetText(AnsiString _txt, int _s)
{
 int s=0;
```

```
 AnsiString hd, str="";
 for(;;){
  hd=fnHead(_txt);
  s+=(hd.Length()+1);
  if(s>_s) break;
  str=str+hd+AnsiString(" ");
  _txt=fnTail(_txt);
  if(_txt.IsEmpty()) break;
 }
 lbTask1->Caption=str;
 lbTask2->Caption=_txt;
}
void __fastcall TfmMain::Button1Click(TObject *Sender)
{
 NextTask();
}
void TfmMain::NextTask()
{
static bool m=true;
 if(m){
  Width=900;
  Height=480;
  m=false;
  WindowState=wsMaximized;
 }
mSave=true;
mTaskNumber=GetTaskNumber();
mCurTask = maTask[mTaskNumber];
mRightAnswer = true;
SetText(fnReplaseSimb(mCurTask.mTask,"#","_____"), 120);
lbPromt->Caption = mCurTask.mPromt;
ebAnswer->Text ="";
lbTask1->Visible=true;
lbTask2->Visible=true;
lbPromt->Visible=true;
```

```
if ( mCurTask.mLevel > 0)lbPromt->Font->Color= clRed;
 if ( mCurTask.mLevel < 0)lbPromt->Font->Color= clBlue;
 if ( mCurTask.mLevel == 0)lbPromt->Font->Color= clGreen;
ebAnswer->Visible=true;
Button2->Visible=true;
Button3->Visible=true;
Button1->Visible = false;
Image1 ->Visible=true;
Button1->Caption="Продовжити";
Label1->Visible=false;
lbTranslate1->Visible=false;
lbTranslate2->Visible=false;
lbTranslate1->Caption= mCurTask.mPromt+ " : "+ mCurTask.mPromtTranslate;
lbTranslate2->Caption=  mCurTask.mAnswer[0]+ " : "+ mCurTask.mAnswerTranslate;
ebAnswer->SetFocus();
}
void TfmMain::CheckAnsver(void)
{
if(ebAnswer->Text.IsEmpty()) return;
AnsiString str = ebAnswer->Text;
bool k =   mCurTask.CheckAnswer(str);
Label1->Visible=true;
String t = "Правильно!";
String f = "Помилка!";
if(k & mPromtCounter>0){
   AnsiString a = fnReplaseSimb(mCurTask.mTask,"#","_____") ;
    AnsiString b = mCurTask.mAnswer[0];
     AnsiString c = UpperCase(ebAnswer->Text);
    Memo1->Lines->Add (a);
}
if(k) {
  TaskEstimate();
  SetText(fnReplaseSimb(mCurTask.mTask,"#",str.UpperCase()), 120);
  Label1->Caption= t.UpperCase();
  lbTranslate1->Visible=true;
```

```
 lbTranslate2->Visible=true;
 Button1->Visible = true;
 Button3->Visible = false;
 Button1->SetFocus();
 }
 else {
 mRightAnswer= false;
 AnsiString a = fnReplaseSimb(mCurTask.mTask,"#","_____") ;
 AnsiString b = mCurTask.mAnswer[0];
  AnsiString c = UpperCase(ebAnswer->Text);
 Memo1->Lines->Add (a);
 Label1->Caption=f.UpperCase();
 }
}
void __fastcall TfmMain::Button3Click(TObject *Sender)
{
 CheckAnsver();
}
void TfmMain::Prompt(void)
{
 AnsiString str = ebAnswer->Text;
 bool k =   mCurTask.CheckAnswer(str);
 if(!k){mPromtCounter++;
  AnsiString s = mCurTask.Symbol(str);
  ebAnswer->Text = s.UpperCase();
  mRightAnswer = false;   }
  if(k & mPromtCounter>0){
  AnsiString a = fnReplaseSimb(mCurTask.mTask,"#","_____") ;
   AnsiString b = mCurTask.mAnswer[0];
    AnsiString c = UpperCase(ebAnswer->Text);
   Memo1->Lines->Add (a);
   mPromtCounter =0;
 }
}
void __fastcall TfmMain::Button2Click(TObject *Sender)
```

```
{
 Prompt();
 ebAnswer->SetFocus();
ebAnswer->SelLength=0;
ebAnswer->SelStart=ebAnswer->Text.Length();
}
int TfmMain:: GetTaskNumber(){
 int b;
 if(mTaskMode==1){
  return GetTaskFromFullHeap();
 }
 if(mTaskMode==2){
  b=random(100);
  if(b>(mTaskProbability+1)*10)
    return GetTaskFromFullHeap();
 }
 int k=MainHeapSize(),sum(0);
 if(k==HeapSize()){
 k=MainHeapSize(true);
 b= random(k);
 for(int i=0; i<mTasksCount;i++){
     if(maTask[i].mLevel>0){
        if(maTask[i].mStep<=0){
           if(sum==b) return i;         sum++;
          }
     }
 }
 }else{
  return GetTaskFromFullHeap();
 }
return 0;
}
int TfmMain:: GetTaskFromFullHeap()
{
   int sum (0);
```

```
  int k=FullHeapSize();
  int b= random(k);
  for(int i=0; i<mTasksCount;i++){
      if(maTask[i].mStep<=0){
        if(sum==b) return i;        sum++;
       }
  }
 return 0;
}
int TfmMain:: FindCurTask(){
 if(mTaskNumber <    mTasksCount)
  if (mCurTask == maTask[mTaskNumber]) return mTaskNumber;
 for(int i =0;i<mTasksCount;i++){
    if (mCurTask == maTask[i]) return i;
 }
 return -1;
}

int TfmMain:: FullHeapSize(){
 int sum=0;
 for(int i=0; i<mTasksCount;i++){
    if(maTask[i].mStep<=0)
        sum++;
 }
 return sum;
}
int TfmMain:: MainHeapSize(bool _s){
 int sum=0;
 for(int i=0; i<mTasksCount;i++){
    if(maTask[i].mLevel>0){
      if(_s){
       if(maTask[i].mStep<=0)
         sum++;
      }
      else{
```

```
      sum++;
      }
     }
  }
 return sum;
}
void TfmMain::Read(AnsiString _fn){
  int kl, kt, km(0);
  FILE * ptrFile = fopen(_fn.c_str(), "r");
  char str[240];
  int cd;
  mTasksCount=0;
  while(!feof (ptrFile)) {
    if (fgets(str, 240, ptrFile));
    cd = 0;
    sscanf(str,"%d",&cd);
    switch(cd){
    case 1 :
    maTask[mTasksCount].Clear();
    maTask[mTasksCount]. ReadTask(str);
    mTasksCount++;
    break;
    case 2 :
    maTask[mTasksCount-1]. ReadPromt(str);
    break;
    case 3 :
    maTask[mTasksCount-1]. ReadAnswer(str);
    case 4 :
    maTask[mTasksCount-1]. ReadPromtTranslate(str);
    break;
    case 5 :
    maTask[mTasksCount-1]. ReadAnswerTranslate(str);
    break;
    case 6:
```

```
scanf(str,"%d%d%d%d%d%d",&cd,&mBadAnswer,&mMainHeapSize,&mStepSize,&mTaskMode
,&mTaskProbability);
   break;
   case 7:
   sscanf(str,"%d%d%d%d",&cd,&km,&kl,&kt);
   break;
   }
 }
ComboBox1->Clear();
mEditTask =0 ;
for(int i =1; i<= mTasksCount;i++)ComboBox1->Items->Add(i);
ShowTask(mEditTask);
fclose (ptrFile);
fmKeyBoard->Close();
switch(km){
 case 1:
 Panel2->Left=kl;
 Panel2->Top=kt;
 Panel2->Visible=true;
 N12->Visible=true;
 N11->Visible=false;
 break;
 case 2:
 Panel2->Visible=false;
 fmKeyBoard->Left=kl;
 fmKeyBoard->Top=kt;
 fmKeyBoard->Show();
 N12->Visible=true;
 N11->Visible=false;
 break;
 default:
 N11->Visible=true;
 N12->Visible=false;
 break;
}
```

```
}
void TfmMain::ShowTask(int _s) {
  ComboBox1->ItemIndex=_s;
  ebkAnswer1->Text="";
   ebkAnswer2->Text="";
   ebkAnswer3->Text="";
   ebkAnswer4->Text="";
   ebkAnswer5->Text="";
   ebTransPromt->Text="";
   ebTransAnswer->Text="";
   ebTask->Text=maTask[_s].mTask;
   ebPromt->Text=maTask[_s].mPromt;
   ebkAnswer1->Text=maTask[_s].mAnswer[0];
   ebkAnswer2->Text=maTask[_s].mAnswer[1];
   ebkAnswer3->Text=maTask[_s].mAnswer[2];
   ebkAnswer4->Text=maTask[_s].mAnswer[3];
   ebkAnswer5->Text=maTask[_s].mAnswer[4];
   ebTransPromt->Text=maTask[_s].mPromtTranslate;
   ebTransAnswer->Text=maTask[_s].mAnswerTranslate;
}
void __fastcall TfmMain::BitBtn1Click(TObject *Sender)
{ if (!mTasksCount) return;
 mEditTask =0 ;
 ShowTask(mEditTask);
}
void __fastcall TfmMain::BitBtn3Click(TObject *Sender)
{
  mEditTask++;
  if(mEditTask>mTasksCount-1) mEditTask=mTasksCount-1;
  ShowTask(mEditTask);
}
void __fastcall TfmMain::BitBtn4Click(TObject *Sender)
{
 mEditTask = mTasksCount-1;
 ShowTask(mEditTask);
```

```
}
void __fastcall TfmMain::BitBtn2Click(TObject *Sender)
{
  mEditTask--;
  if(mEditTask<0) mEditTask=0;
  ShowTask(mEditTask);
}
void __fastcall TfmMain::ComboBox1Change(TObject *Sender)
{
 mEditTask=ComboBox1->ItemIndex;
 ShowTask(mEditTask);
}
void __fastcall TfmMain::BitBtn5Click(TObject *Sender)
{   mSave = true;
mEditTask=ComboBox1->ItemIndex;
 if (ebTask->Text.IsEmpty()) {
    }
  maTask[mEditTask].mTask= ebTask->Text;
   maTask[mEditTask].mPromt=ebPromt->Text;
   maTask[mEditTask].mAnswer[0]=ebkAnswer1->Text;
   maTask[mEditTask].mAnswer[1]=ebkAnswer2->Text;
   maTask[mEditTask].mAnswer[2]=ebkAnswer3->Text;
   maTask[mEditTask].mAnswer[3]=ebkAnswer4->Text;
   maTask[mEditTask].mAnswer[4]=ebkAnswer5->Text;
   maTask[mEditTask].mPromtTranslate=ebTransPromt->Text;
   maTask[mEditTask].mAnswerTranslate = ebTransAnswer->Text;
}
void __fastcall TfmMain::BitBtn6Click(TObject *Sender)
{ mEditTask=ComboBox1->ItemIndex;
   maTask[mEditTask]=  maTask[mTasksCount-1];
   mTasksCount--;
   for(int i =1; i<= mTasksCount;i++)ComboBox1->Items->Add(i);
   ComboBox1->ItemIndex=mEditTask;
   ShowTask(mEditTask);
}
```

```
void __fastcall TfmMain::BitBtn7Click(TObject *Sender)
{       ebTask->Text="";
   ebPromt->Text="";
   ebkAnswer1->Text="";
   ebkAnswer2->Text="";
   ebkAnswer3->Text="";
   ebkAnswer4->Text="";
   ebkAnswer5->Text="";
   ebTransPromt->Text="";
   ebTransAnswer->Text="";
mEditTask =  mTasksCount;
mTasksCount++;
   ComboBox1->Items->Add(mTasksCount);
   ComboBox1->ItemIndex =mEditTask;
}
void __fastcall TfmMain::FormPaint(TObject *Sender)
{
 static bool a=true;
 if(a){
  Graphics::TBitmap*   gBitmap = new Graphics::TBitmap;
  gBitmap->LoadFromFile("1.bmp");
  Canvas ->Draw(10,10,gBitmap)  ;
  delete gBitmap;
  a=false;
 }
}

void __fastcall TfmMain::FormCanResize(TObject *Sender, int &NewWidth,
      int &NewHeight, bool &Resize)
{
  if(mTasksCount) return;
  NwWidth=610;
  NewHeight=450;
}
```

```
void__fastcall TfmMain::FormClose(TObject *Sender, TCloseAction &Action)
{
 int k;

 if(!mSSFont){
  k=RemoveFontResource("Microsoft Sans Serif.ttf");
 }
 if(mSave){
   k=Application-> MessageBox ("", MB_YESNOCANCEL);
   switch(k){
    case IDCANCEL: Action =caNone; break;
    case IDYES: N6Click(NULL); break;
   } }
}
void __fastcall TfmMain::N6Click(TObject *Sender)
{
 if(mFileName.IsEmpty()){
    N7Click(Sender);
    return;
  }
 SaveTasks(mFileName);
}
void  TfmMain::SaveTasks(AnsiString _F){
FILE* out   = fopen(_F.c_str(),"w");
//FILE* out   = fopen("fff.txt","w");
fprintf(out,"6 %d %d %d %d
%d\n",mBadAnswer,mMainHeapSize,mStepSize,mTaskMode,mTaskProbability);
if(Panel2->Visible){
 fprintf(out,"7 1 %d %d\n",Panel2->Left,Panel2->Top);
}
if(fmKeyBoard->Visible){
 fprintf(out,"7 2 %d %d\n",fmKeyBoard->Left,fmKeyBoard->Top);
}
for (int i = 0; i<mTasksCount; i++  ){
maTask[i].Save(out);
```

```
}
mSave=false;
fclose (out);
}
void __fastcall TfmMain::N7Click(TObject *Sender)
{
 if(!SaveDialog1->Execute()) return;
 mFileName = SaveDialog1->FileName + ".txt";
 SaveTasks(mFileName);
}
void __fastcall TfmMain::N8Click(TObject *Sender)
{
 fmParam->ComboBox1->ItemIndex = mBadAnswer-2;
 fmParam->ComboBox2->ItemIndex = mMainHeapSize;
 fmParam->ComboBox3->ItemIndex = mStepSize;
 fmParam->ComboBox4->ItemIndex = mTaskMode;
 fmParam->ComboBox5->ItemIndex = mTaskProbability;
 if(!fmParam->Execute()) return;
 mBadAnswer= fmParam->ComboBox1->ItemIndex+2;
 mMainHeapSize=fmParam->ComboBox2->ItemIndex;
 int n = 0;
 for(int i =0; i< mTasksCount; i++){
  maTask[i].mStep = 0;
 if(maTask[i].mLevel>0){
 n++;
 if(n>HeapSize()) maTask[i].mLevel = - maTask[i].mLevel;
 }
 }
 mStepSize=fmParam->ComboBox3->ItemIndex;
 mTaskMode=fmParam->ComboBox4->ItemIndex;
 mTaskProbability=fmParam->ComboBox5->ItemIndex;
 if( HeapSize() < StepSize()) mStepSize = mMainHeapSize*2;
 mSave=true;
 ShowTaskMode();
}
```

```
void __fastcall TfmMain::N9Click(TObject *Sender)
{
 ShellExecute(Handle, "open", "help.pdf",0, 0, SW_SHOWNORMAL);
}
void __fastcall TfmMain::N10Click(TObject *Sender)
{for(int i =0; i< mTasksCount; i++){
maTask[i].mLevel = -mBadAnswer;
}
}
void __fastcall TfmMain::BitBtn8Click(TObject *Sender)
{  mEditTask  = mTaskNumber;
 ShowTask( mEditTask);
}
void __fastcall TfmMain::N11Click(TObject *Sender)
{
 fmKeyBoard->Show();
 fmMain->Show();
}

void TfmMain::StepLeft()
{
  int a=ebAnswer->SelStart;
  a--;
  a=a>0?a:0;
  ebAnswer->SetFocus();
  ebAnswer->SelStart=a;
  ebAnswer->SelLength=0;
}
void TfmMain::StepRight()
{
 int a=ebAnswer->SelStart;
 int b=ebAnswer->SelLength;
 int l=ebAnswer->Text.Length();
 a=a+b+1;
 ebAnswer->SetFocus();
```

```
 ebAnswer->SelStart=a;
 ebAnswer->SelLength=0;
}
void TfmMain::DeleteChar()
{
 int a=ebAnswer->SelStart;
 int b=ebAnswer->SelLength;
 int l=ebAnswer->Text.Length();
 AnsiString s1=ebAnswer->Text.SubString(1,a);
 AnsiString s2=ebAnswer->Text.SubString(a+b+1,l-a-b);
 if(!b){
  l=s2.Length();
  s2=s2.SubString(2,l-1);
 }
 ebAnswer->SetFocus();
 ebAnswer->Text=s1+s2;
 ebAnswer->SelStart=a;
}

void TfmMain::BackSpace()
{
 int a=ebAnswer->SelStart;
 int b=ebAnswer->SelLength;
 int l=ebAnswer->Text.Length();
 AnsiString s1=ebAnswer->Text.SubString(1,a);
 AnsiString s2=ebAnswer->Text.SubString(a+b+1,l-a-b);
 if(!b){
  l=s1.Length();
  s1=s1.SubString(1,l-1);
  a--;
  a=a>0?a:0;
 }
 ebAnswer->SetFocus();
 ebAnswer->Text=s1+s2;
 ebAnswer->SelStart=a;
```

```
}
void TfmMain::InsertChar(char _c)
{
 int a=ebAnswer->SelStart;
 int b=ebAnswer->SelLength;
 int l=ebAnswer->Text.Length();
 AnsiString s1=ebAnswer->Text.SubString(1,a);
 AnsiString s2=ebAnswer->Text.SubString(a+b+1,l-a-b);
 ebAnswer->Text=s1+AnsiString(_c)+s2;
 ebAnswer->SetFocus();
 ebAnswer->SelStart=a+1;
}
void __fastcall TfmMain::QClick(TObject *Sender)
{
 char c=((TButton*)Sender)->Caption[1];
 InsertChar(c);
}
void __fastcall TfmMain::Button33Click(TObject *Sender)
{
 BackSpace();
}

void __fastcall TfmMain::Button29Click(TObject *Sender)
{
 DeleteChar();
}
void __fastcall TfmMain::Button30Click(TObject *Sender)
{
 StepLeft();
 ebAnswer->SetFocus();
}
void __fastcall TfmMain::Button31Click(TObject *Sender)
{
 StepRight();
}
```

```
void __fastcall TfmMain::ebAnswerKeyPress(TObject *Sender, char &Key)
{
 switch(Key){
  case 13:
   if(ebAnswer->Text.IsEmpty()) return;
   Button3Click(Sender);
   break;
  case 27:
   Button2Click(Sender);
   break;
 }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows.Input;
using MySql.Data.MySqlClient;
using System.Windows;
using System.Collections.ObjectModel;
using System.Windows.Interactivity;
namespace
{
        public abstract class All_People
        {
                string Type_People;
                string Last_Name;
                string First_Name;
                string Father_Name;
                string Post;
                string Nomber;
```

```
                string Nick_Name;
                int Id;
        public int id { get { return Id; }
                        set { Id = value; } }
        public string type_people { get { return Type_People; }  set { Type_People = value; } }
        public string last_name { get { return Last_Name; }
 set { Last_Name = value; } }
        public string first_name { get { return First_Name; }
      set { First_Name = value;  } }
        public string father_name { get { return Father_Name; }
  set { Father_Name = value; } }
        public string post { get { return Post; }                                   set { Post = value;  } }
        public string nomber { get { return Nomber; }                               set { Nomber = value;  }
}
    public string nick_name { get { return Nick_Name; }            set { Nick_Name = value; } }
                } // общая класс для сущности клиент
    public abstract class All_Goods_And_Product
    {
        public All_Goods_And_Product(string name, float cost)
        {
                this.name = name;
                this.cost = cost;
        }
        string Name;
        float Cost;
        int Id;
        public int id { get { return Id; } set { Id = value; } }
        public string name { get { return Name; } set { Name = value; } }
        public float cost { get { return Cost; } set { Cost = value; } }
    } // общий класс для сущности продукт и товар
    public class BoolToVisibilityConverter : System.Windows.Data.IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
```

```
            return (bool)value ? Visibility.Visible : Visibility.Collapsed;
        }
        public object ConvertBack(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    } // конвертор для видимости элементов
    public class RelayCommand : ICommand
    {
        private Action<object> execute;
        private Func<object, bool> canExecute;
        public event EventHandler CanExecuteChanged
        {
            add { CommandManager.RequerySuggested += value; }
            remove { CommandManager.RequerySuggested -= value; }
        }
        public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)
        {
            this.execute = execute;
            this.canExecute = canExecute;
        }
        public bool CanExecute(object parameter)
        {
            return this.canExecute == null || this.canExecute(parameter);
        }
        public void Execute(object parameter)
        {
            this.execute(parameter);
        }
    } // класс для создания команд
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи