МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Квашук Ольги Олександрівни
(прізвище, ім'я, по батькові здобувача)

на тему
Розробка програмного забезпечення продажу жіночих прикрас
(повна назва теми)

за матеріалами праць провідних спеціалістів з розробки ПЗ та проектування БД
(повна назва бази дослідження)

науковий керівник к.е.н., доцент _____ Лисенко В.С.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 11.06.2025 р. № 12
Завідувач кафедри _____
(підпис)

д.т.н., професор Зеленський О.С.
Наук. ступень, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Зеленський О.С.
(підпис) (Прізвище, ініціали)
« 11 » червня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

1. Тема роботи Розробка програмного забезпечення продажу жіночих прикрас

Керівник роботи к.е.н., доцент Лисенко В.С.
затверджені наказом закладу вищої освіти від «04» березня 2025 р. № 222-ст

2. Строк подання здобувачем роботи до «09» червня 2025 р.

3. Зміст кваліфікаційної роботи, об’єкт, предмет та мета дослідження:

Розділ 1. Постановка задачі

Розділ 2. Розробка алгоритму розв’язання задачі

Розділ 3. Організація інформаційного забезпечення

Розділ 4. Розробка програмного забезпечення задачі

Об’єкт дослідження: продаж жіночих прикрас

Предмет дослідження: створення програм для продажу жіночих прикрас

Мета кваліфікаційної роботи: Розробка програмного забезпечення продажу жіночих прикрас

5. Дата видачі завдання «04» березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МДР	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	04.04.2025-13.04.2025	
2	Підготовка розділу 2	14.04.2025-26.04.2025	
3.	Підготовка розділу 3	27.04.2025-15.05.2025	
4	Підготовка розділу 4	16.05.2025-08.06.2025	
5	Реєстрація завершеної кваліфікаційної роботи	09.06.2025	Реєстраційний №___ «09»червня 2025 р.
6	Отримання відгуку від наукового керівника	10.06.2025	
7	Подання кваліфікаційної роботи на перегляд завідувачу кафедри	11.06.2025	
8	Отримання зовнішньої рецензії	12.06.2025	
9	Попередній захист кваліфікаційної роботи на кафедрі	13.06.2025	
10	Підготовка до захисту в ЕК	16.06.2025-21.06.2025	

Завдання підготував науковий керівник _____ __В.С. Лисенко__
(підпис) (прізвище та ініціали)

Завдання одержав _____ ___О.О. Квашук____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

на кваліфікаційну бакалаврську роботу

«Розробка програмного забезпечення продажу жіночих прикрас»

Квашук Ольги Олександрівни

Кваліфікаційна бакалаврська робота на здобуття освітньо-кваліфікаційного рівня бакалавра зі спеціальності 121 «Інженерія програмного забезпечення» – Державний університет економіки і технологій – Кривий Ріг, 2025.

Дана бакалаврська робота присвячена розробці програмного забезпечення для продажу жіночих прикрас з використанням технології Windows Forms на C#, 3D-графіки на мові програмування C++ з використанням бібліотеки OpenGL, а також бази даних MS Access.

Мета роботи – створення програмного забезпечення, яке дозволяє користувачам переглядати, вибирати та купувати жіночі прикраси. Для цього використовується Windows Forms на мові C#, щоб створити зручний та інтуїтивно зрозумілий інтерфейс для користувачів.

Візуалізація реалізується за допомогою 3D-графіки, яка відображає ціни на жіночі прикраси у вигляді гістограми.

Для зберігання та управління даними про доступні жіночі прикраси використовується база даних MS Access, що надає зручний та ефективний доступ до інформації про жіночи прикраси та клієнтів.

Результатом роботи є функціональне програмне забезпечення, яке може бути використано в ювелірних салонах або онлайн-магазинах для демонстрації та продажу жіночих прикрас. Розроблений додаток покращує процес продажу та задовольняє потреби клієнтів, надаючи їм зручні та реалістичні можливості для перегляду та вибору продуктів.

Ключові слова: C#, MS Access, MVC C++, БД, СУБД, OpenGL, ADO, ADO .NET.

ЗМІСТ

ВСТУП

Бізнес-процеси сучасного світу тісно зв'язані з інформаційними технологіями. Комп'ютеризація різних сфер діяльності людини призводить до економічного зростання і покращення фінансового стану підприємств. Людина займаючись підприємницькою діяльністю намагається отримати максимальний прибуток за найкоротший час і з мінімальними затратами. У цьому їй в пригоді стають як комп'ютерні системи так і різноманітне програмне забезпечення. В той час, необхідно розуміти, що головну роль відіграє правильна побудова бізнес-процесу підприємства. Якісно спроектувати майбутнє підприємство дозволяє системний аналіз, який дає змогу розібрати, зрозуміти та поліпшити окрему сферу діяльності людини, а також дозволяє передбачити її можливі шляхи розвитку.

Мета дипломної роботи – створення програмного забезпечення продажу жіночих прикрас на мові C#. Передбачено використання бібліотеки OpenGL для виведення статистики по цінам жіночих прикрас, атакож створення автоматизованих Java-скриптів у Web-програмування для виведення діаграм цін, а також універсальні функції експорту даних до html.

Основними завданнями даної роботи є:

– ведення категорії продукції жіночих прикрас;
– ведення інформації по клієнтам;
– організація купівлі жіночих прикрас;
– виведення 3D-графіки засобами OpenGL;
– створення Web-скриптів для організації діаграм та експорту даних.

Об'єктом дослідження виступають жіночі прикраси.

Предметом дослідження даної роботи є інформаційна підтримка процесу розробки програмного забезпечення жіночих прикрас.

Дана тема є надзвичайно актуальною в наш час, так як все більше з’являється магазині, постійно роблять нові прикраси. Росте асортимент прикрас, росте попит.

Програмний продукт буде розроблено за допомогою інструментальних засобів розробки мови програмування C#, розробка скриптів з використанням елементу Chart.js, розробка 3D-графіки на мові C++ (MFC) з використанням бібліотеки OpenGL. Для роботи з базами даних, використано дві технології – ADO для мови C++, ADO .NET для мови C#. У якості бази даних обрано MS Access.

На ринку програмного забезпечення призначеного для використання в магазинах є велика кількість систем, що автоматизують роботу продавця. Такі системи, взагалі, розробляються персонально для окремих магазинів не мають загальних стандартів, або ж автоматизують якусь одну, окрему, ділянку роботи продавця. Це ще раз підтверджує актуальність теми.

Практична цінність програмного забезпечення, що розроблятиметься, визначається її універсальністю і можливістю побудови, на її основі, реального програмного забезпечення для автоматизації роботи продавця.

Програмна система, що розроблятиметься, буде призначена для використання продавцем та полегшення і покращення організації його роботи.

Задачами, що слід виконати для досягнення мети роботи, є наступні:

- охарактеризувати вхідну та вихідну інформацію;
- розробити методику формування показників;
- розробити алгоритм роботи програми;
- побудувати графіки;
- розробити програмне забезпечення підрахунку максимальної покупки в залежності від замовлення покупця.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ

1.1. Характеристика задачі

Для того, щоб правильно сформулювати характеристику задач, необхідно зібрати матеріал з предметної області, і на його основі будувати програмне забезпечення, яке б відповідало меті дипломної роботи. Характеристика задач полягає у описанні призначення задач, підстав для їх рішення, предмету задач, а також визначення перших вимог до розробки програмного забезпечення, що стосуються її інтерфейсу, вхідної та вихідної інформації, звітування тощо. У роботі не тільки відбувається механічна заміна різних мов програмування, а дещо змінюється підхід до реалізації задачі тому і характеристики задач будуть різними.

На сьогоднішній день, у сфері послуг продажу жіночих прикрас займають одне з провідних місць. Магазини продажу приносять вагомий прибуток своїм власникам, та є надзвичайно рентабельними. Магазини працюють з величезними реєстрами даних, які потребують систематизації та спрощення доступу. Таким чином виникла потреба автоматизувати облік. Особливих ресурсів майбутнє програмне забезпечення не потребує, отже воно будує доступне для будь-якої компанії.

Останнім часом все помітнішою стає потреба у правильній організації процесу роботи такої структури. Здебільшого якість роботи програмного забезпечення має суто економічні характеристики або мультимедійного направлення. Але програми автоматизації мають порівняно широке коло аудиторії і тому мають більш специфічне призначення.

При розробці даного програмного забезпечення, було поставлено наступну задачу – спроектувати програмне забезпечення, яке можна було б використовувати для продажу жіночих прикрас.

Неможливо швидко і якісно обробляти великі обсяги однотипної інформації, представленої в текстовій формі. Таку інформацію набагато

зручніше обробляти за допомогою таблиць. Але сприйняття громіздких таблиць також виявляється скрутним для людини. За допомогою графіків і діаграм можна візуалізувати великі обсяги однотипної табличної інформації.

Під автоматизацією магазину, перегляд всіх даних по поточному виробу, графічне відображення замовлень.

Розроблювана програма буде вирішувати наступні функціональні задачі:

- ✓ Продажу ювелірних виробів;
- ✓ побудова графіків;
- ✓ побудова діаграм;
- ✓ додавання нових товарів до бази даних;
- ✓ швидкий доступ до будь яких даних;
- ✓ копіювання та видалення даних з бази;
- ✓ формування SQL запитів.

Таким чином необхідно створити програмне забезпечення, що дозволяло б зберігати та використовувати інформацію, яка описуватиме весь процес.

1.2. Вхідна інформація

Вхідна інформація – це сукупність показників, характеристик об’єкта дослідження або інших значень, над якими треба виконувати обчислення, здійснювати їх обробку, тощо. Результатом обробки вхідної інформації є вихідна інформація, детальніше про неї у пункті 1.3.

Ефективна діяльність магазину вимагає наявності первинних даних. Вхідна інформація представляє собою сукупність показників, характеристик об'єкта дослідження, або інших значень, над якими необхідно проводити обчислення, здійснювати їх обробку тощо.

Усі об’єкти, що знаходяться в обігу заносяться до бази даних. Опис системи вхідної, вихідної та довідкової інформації буде представлено у

вигляді визначення особливостей і змісту вхідного матеріалу для кожного із завдань автоматизації процесу, що розглядається в даній роботі:

а) надходження категорії до магазину, перелік полів представлено в табл. 1.1;

б) перелік продукції представлено в табл. 1.1.

Таблиця 1.1

Перелік і опис вхідних повідомлень товарів в магазині

№з/п	Назва вхідного повідомлення	Іденти-фікатор	Форма Представ-лення	Термін і частота надходження	Джерело
1	Номер товару	ID	Таблична	При реєстрації нового товару в магазині	Менеджер
2	Назва	Name	Таблична	При реєстрації нового товару в магазині	Менеджер
3	Каталог	Catalog	Таблична	При реєстрації нового товару в магазині	Менеджер

При надходженні товару до магазину в основній відомості представлені першочергово необхідні данні про товар. Цих даних достатньо для прийомки товару але ці данні не є вичерпними для надання повної характеристики покупцю.

Повний перелік всієї необхідної інформації для надання розгорнутих характеристик товару береться із документів: «Продукція». Цієї інформації достатньо для надання розгорнутих характеристик товару покупцеві в повному обсязі в табл. 1.2.

Таблиця 1.2

Перелік і опис вхідних повідомлень документа «Продукція»

№ з/п	Назва вхідного повідомлення	Іденти-фікатор	Форма представ-лення	Термін і частота надходження	Джерело
1	Номер каталогу	ID_cat	Таблична	При реєстрації нового товару в магазині	Менеджер

Продовження таблиці 1.2

№ з/п	Назва вхідного повідомлення	Іденти-фікатор	Форма представ-лення	Термін і частота надходження	Джерело
2	Назва продукту	Name	Таблична	При реєстрації нового товару в магазині	Менеджер
3	Зображення	Image	Таблична	При реєстрації нового товару в магазині	Менеджер
4	Опис продукту	Description	Таблична	При реєстрації нового товару в магазині	Менеджер
5	Ціна	Price	Таблична	При реєстрації нового товару в магазині	Механік

Дана таблиця демонструє інформацію представлену в документі «Продукція». Даний документ надається виробником і характеризує. Компетентний в цьому питанні покупець зможе адекватно оцінити виріб.

Головною умовою реалізації поставленої задачі – розробка програмного додатку на мові C# є наявність і правильне розподілення вхідної інформації. Вхідними даними виступають поля таблиць бази даних MS ACCESS «shop.mdb» (таблиці 1.3 – 1.5).

База даних «shop.mdb» складається з таблиць. Структура кожної таблиці наведена нижче. Таблиці бази даних містять наступну інформацію:

1) «Товари» в даній таблиці міститься першочергово необхідна інформація про товар (табл. 1.3);

2) «Замовлення прикрас покупцем» (табл. 1.4);

3) «Покупці» (табл. 1.5).

Таблиця 1.3

Перелік і опис вхідних повідомлень таблиці «Товари»

№ з/п	Назва вхідного повідомлення	Іденти-фікатор	Форма представ-лення	Термін і частота надходження	Джерело
1	Номер товару	ID_cat	Таблична	При реєстрації нового товару	Менеджер

Продовження таблиці 1.3

№ з/п	Назва вхідного повідомлення	Іденти-фікатор	Форма представ-лення	Термін і частота надходження	Джерело
2	Модель	Name	Таблична	При реєстрації нового товару	Менеджер
3	Відомості	Descri-ption	Таблична	При реєстрації нового товару	Менеджер
4	Ціна	Price	Таблична	При реєстрації нового товару	Менеджер

При надходженні прикрас до магазину в основній відомості представлені першочергово необхідні данні про них. Цих даних достатньо для прийому товарів, але ці данні не є вичерпними для надання повної характеристики покупцю.

Таблиця 1.4

Перелік і опис вхідних повідомлень таблиці «Замовлення прикраси клієнтом»

№з /п	Назва вхідного повідомлення	Іденти-фікатор	Форма представ-лення	Термін і частота надходження	Джерело
1	Клієнт	ID_klient	Таблична	При реєстрації нового товару в магазині	Менеджер
2	Номер продукту	ID_Prod	Таблична	При реєстрації нового товару в магазині	Менеджер
3	Дата замовлення товару	Data_zak	Таблична	При реєстрації нового товару в магазині	Менеджер
4	Ціна	Cena	Таблична	При реєстрації нового товару в магазині	Менеджер
5	Кількість виробів	Kol	Таблична	При реєстрації нового товару в магазині	Менеджер

Дана таблиця містить всі необхідні данні про заказану продукцію.

Таблиця 1.5

Перелік і опис вхідних повідомлень «Покупці»

№ з/п	Назва вхідного повідомлення	Іденти-фікатор	Форма Представ-лення	Термін і частота надходження	Джерело
1	ID клієнта	ID	Таблична	При реєстрації нового товару в магазині	Менеджер

Продовження таблиці 1.5

№ з/п	Назва вхідного повідомлення	Іденти-фікатор	Форма Представ-лення	Термін і частота надходження	Джерело
2	Прізвище	Fname	Таблична	При реєстрації нового товару в магазині	Менеджер
3	Ім’я	Sname	Таблична	При реєстрації нового товару в магазині	Менеджер
4	Адреса	Adress	Таблична	При реєстрації нового товару в магазині	Менеджер
5	Індекс	Post_index	Таблична	При реєстрації нового товару в магазині	Менеджер
6	Телефон	Telefon	Таблична	При реєстрації нового товару	Менеджер

Отже, в результаті проведеної роботи, отримуємо потужну основу для створення бази даних, котра задовольнятиме потреби користувачів будь-яких рівнів володіння комп’ютером.

1.3. Вихідна інформація

В загальному розумінні вихідна інформація представляє собою результат роботи з базою даних. Процес опрацювання даних, що знаходяться в базі або додаються до неї реалізується з використанням алгоритмів розробленої програми. Вихідна інформація у даній роботі визначена у вигляді вибраних користувачем даних. Також можна сказати, що програмне забезпечення використовується для полегшення пошуку інформації шляхом сортування та фільтрації. Описання вихідної інформації відбувається у наступній послідовності:

- визначення вхідної інформації для програми, необхідної для проведення опрацювання інформації в базі даних;
- створення структури бази даних, яка є об’єктом обробки для програми, що розробляється, як головне завдання дипломної роботи;
- створення програми для обробки даних запропонованої бази;
- процес розрахунку, тобто робота з базою даних з використанням функцій, що містить розроблена програма;

- отримання вихідної інформації, що є завершальною стадією даної послідовності дій.

Після здійснення вищевказаних процесів можна відзначити, що вихідна інформація у даній роботі визначена у табличному вигляді. Також можна сказати, що програмне забезпечення, створене для продажі ювелірних виробів (таблиця 1.6). Для зручності вводу ключові поля винесені на початок і всі записи є першочергово відсортовані.

Форма представлення незмінно таблична, але завдяки зручному інтерфейсу прискорює роботу, та зменшує кількість помилок завдяки контролю введення даних та вдало підібраним шрифту та розміру тексту, та спокійної гами кольорів.

Таблиця 1.6

Характеристика вихідних повідомлень

№ з/п	Назва вхідного повідомлення	Форма представлення	Термін і частота надходження	Джерело
1	Виконаний пошук по полю «Категорії»	Таблична (екранна)	При кожному звертанні	Користувач
2	Розгорнутий від продукції	Таблична (екранна)	При кожному звертанні	Користувач
3	Виконаний введений або вже існуючий запит	Таблична (екранна)	При кожному звертанні	Користувач

Характеристика вихідних повідомлень наведена у таблиці 1.7. Важливо зазначити, що форма різниться типом представлення, адже програмне забезпечення дає змогу зберігати звіт у файл або проглянути повний графічний звіт прямо на екрані комп’ютера.

Програмний продукт різниться та формою представлення. На головному вікні програми користувач може переглянути вироби в виді таблиці з інформацією у супроводі із зображенням обраного товару.

Розрахунки які містить діалогове вікно мають табличне та графічне відображення. Що дає змогу користувачеві наглядно розуміти сутність.

Отже, вся вихідна інформація зберігається у досить привабливому вигляді та легка для сприймання. Звіти мають велике функціональне навантаження, адже користувачу легше буде зробити свій вибір.

Таблиця 1.7

Перелік і опис вихідних повідомлень

№ з/п	Назва вхідного повідомлення	Форма представлення	Термін і частота надходження	Джерело
1	Перегляд товарів	Таблична (екранна)	При кожному звертанні	Користувач
2	Відсоток продажів по виробникам	Графічна (Стовпчаста діаграма)	При кожному звертанні	Користувач

Розроблене програмне забезпечення виконує функцію по виведенню необхідної інформації щодо жіночих прикрас.

У програмному додатку представлені як графічна так і таблична форма виведення інформації. Так в табличному режимі виводиться вся необхідна вихідна та розрахункова інформація про жіночі прикраси. Графічна форма представлення а саме галерея виконує допоміжну функцію та була створена для зручності та наочності.

Характеристика вихідних повідомлень наведена у таблиці 1.8.

Таблиця 1.8

Характеристика вихідних повідомлень

№ з/п	Назва вхідного повідомлення	Форма представлення	Термін і частота надходження	Джерело
1	Вибрано виріб за вказаною категорією	Таблична (екранна)	При кожному звертанні	Користувач
2	Вибрано детальний перегляд обраного виробу	Графічна (Галерея із зображеннями)	При кожному звертанні	Користувач

Висновки до розділу 1

Отже, у цьому розділі було успішно сформульовано характеристику задачі, мету її вирішення та цілі, які вона переслідує. Була визначена структура та зміст вхідної та вихідної інформації, основні вимоги до їх оформлення, джерела, які забезпечують задачу вхідною інформацією, та особи, які мають отримати вихідну інформацію. На основі цих даних можна переходити до наступного етапу розробки програмного забезпечення.

РОЗДІЛ 2
РОЗРОБКА АЛГОРИТМУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1. Розробка алгоритму вирішення задачі «Uvelir»

Так як задача на різних мовах програмування має і різний підхід алгоритми вирішення цих задач будуть кардинально відрізнятись. Усі три програмні комплекси будуть мати головне меню, за допомогою якого будуть реалізовуватись алгоритми вирішення задач та буде здійснюватись управління програмними комплексами.

Для реалізації такої задачі слід розробити алгоритм. Робота з програмою здійснюється на основі управління через пункти головного меню та панелі інструментів наведених на рис. 2.1 та описано у таблиці 2.1.

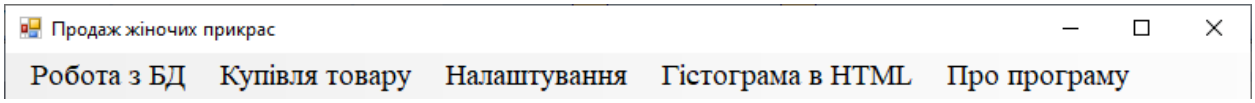


Рис. 2.1 Головне меню програми

Таблиця 2.1

Пункти головного меню

№ п/п	Найменування пункту меню	Призначення
1	«Робота з БД»	Створює підключення та запуск програми на C# для керування базою даних
2	«Купівля товару»	Викликає форму для організації купівлі товару
3	«Налаштування»	Вибір параметрів шрифту та кольору для виведення інформації
4	«Гістограма в HTML»	Виведення гістограми в HTML за ціною прикрас
5	«Про програму»	Виведення інформації про розробника програми

Алгоритм, за яким функціонує програма, розділено на дві схеми, які представлено на рис. 2.2. та рис. 2.3.

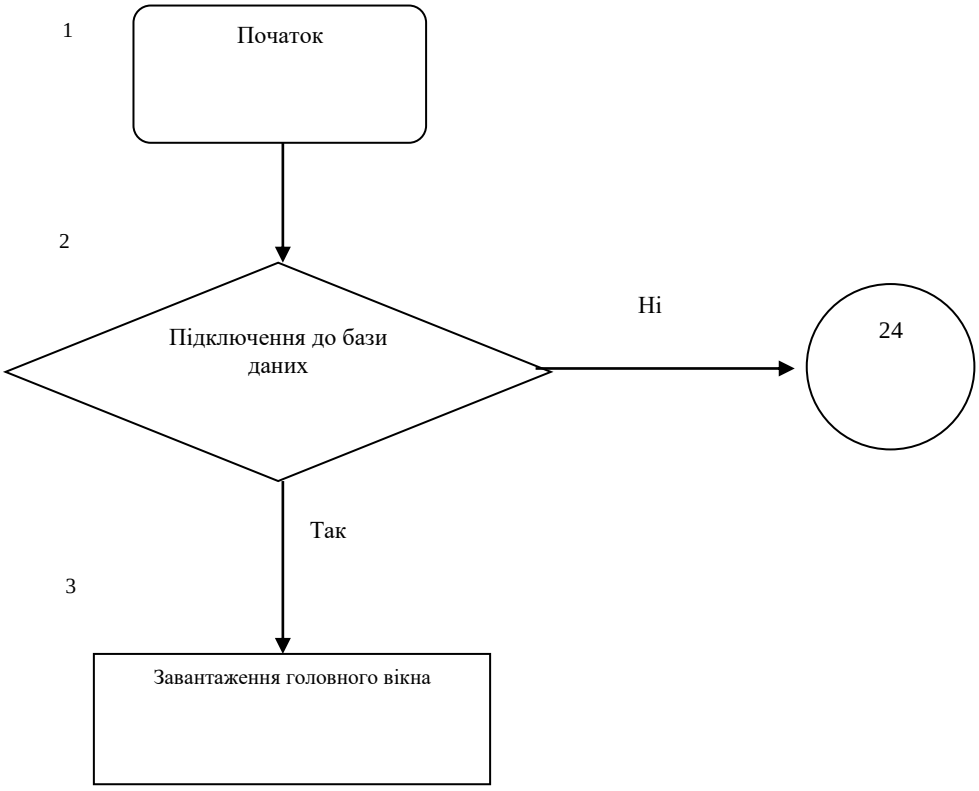


Рис. 2.2. Блок-схема роботи програми

Він включає наступні кроки:

1. Запуск програми та завантаження необхідних компонентів.
2. Відображення головного меню програми.
3. Очікування вибору користувача.
4. Обробка вибраного пункту меню.

Повторення кроків 3-4 до виходу з програми або вибору користувачем відповідного пункту меню для виконання інших дій.

Цей алгоритм забезпечує послідовне виконання різних функцій програми, включаючи роботу з базою даних, оформлення замовлень, налаштування візуальних параметрів та надання інформації про програму.

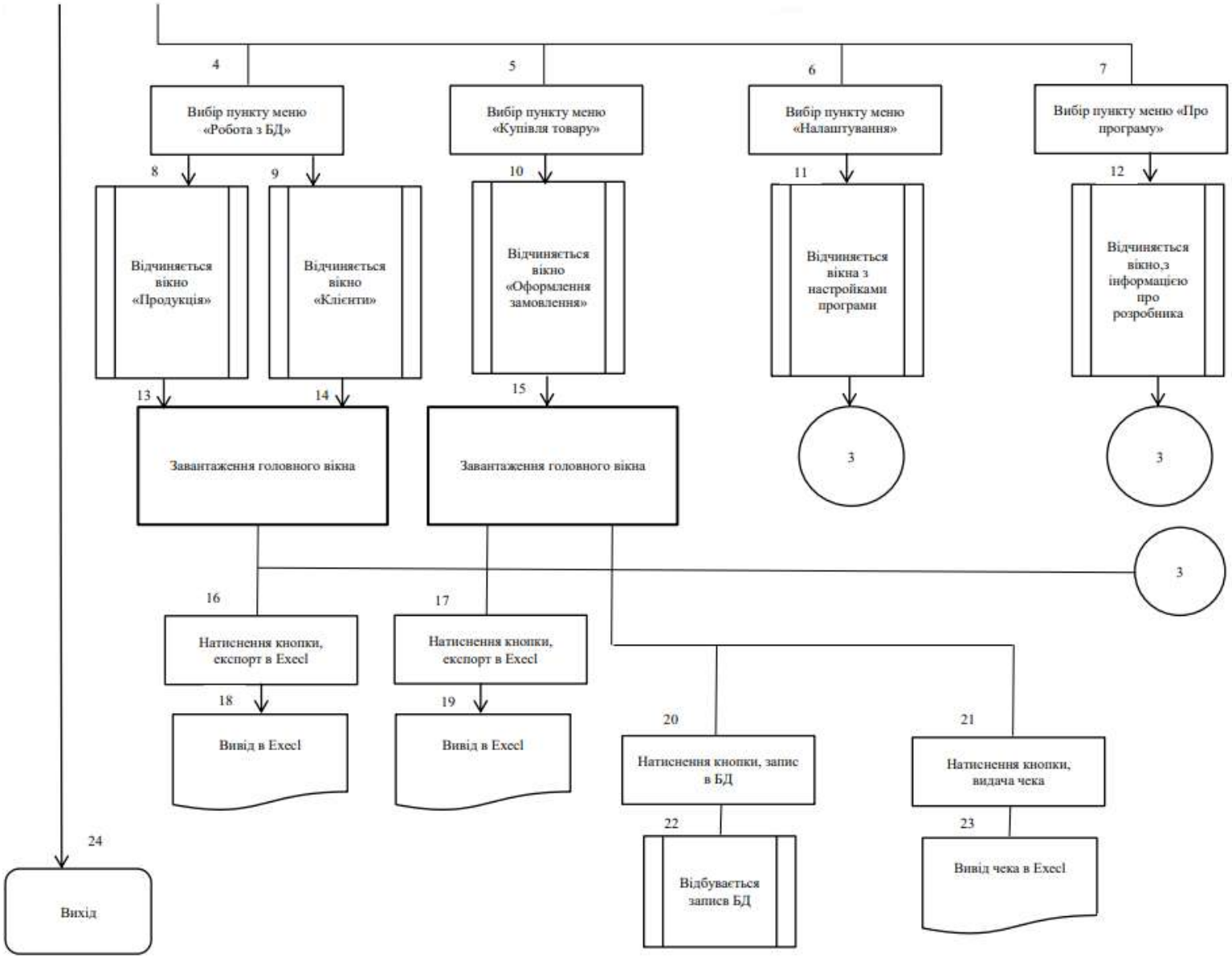


Рис. 2.3. Завершення алгоритму роботи програми

2.2. Розробка алгоритму вирішення задачі «Uvelir_3D»

Для успішної реалізації поставленої задачі необхідно розробити відповідний алгоритм.

Управління програмою «Uvelir_3D» здійснюється через головне меню, яке надає користувачу доступ до різних функціональних можливостей.

Структура меню та його пункти представлені на рис. 2.4.

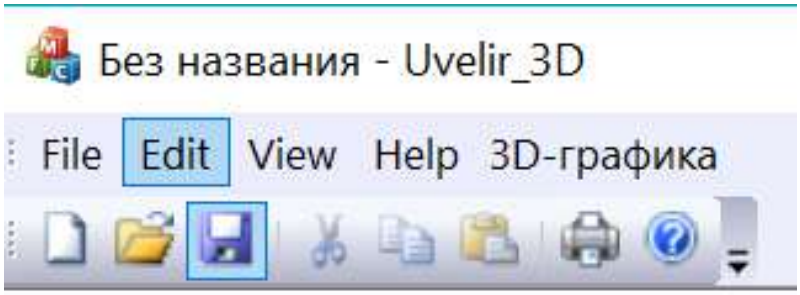


Рис. 2.4. Вигляд пунктів меню програми Uvelir_3D

Основна мета програми "Uvelir_3D" полягає у створенні 3D-графічної гістограми для візуалізації ювелірних прикрас.

Ця програма надає можливість створювати вражаючі інтерактивні візуалізації, що дозволяють зрозуміти розподіл прикрас за різними категоріями та параметрами.

Головний інтерфейс програми "Uvelir_3D" має інтуїтивно зрозуміле меню, яке надає користувачеві доступ до різних функцій.

В результаті використання програми "Uvelir_3D" користувач може отримати якісну та ефективну візуалізацію розподілу ювелірних прикрас за різними характеристиками.

Ця програма допомагає візуально представити дані та дозволяє зробити висновки про характеристики та тенденції в галузі ювелірного мистецтва.

Алгоритм наведено на рис. 2.5.

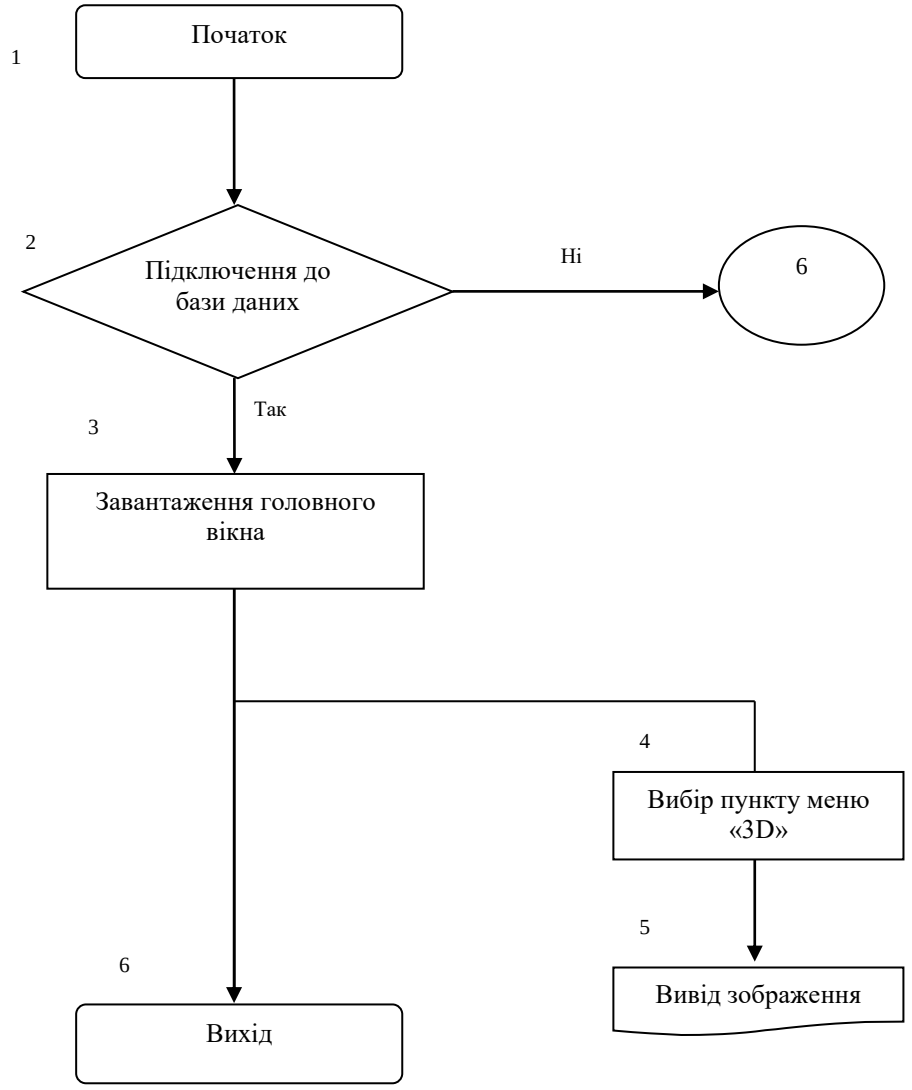


Рис. 2.5. Блок-схема алгоритму роботи програми «Uvelir_3D»

Цей алгоритм забезпечує користувачу можливість взаємодіяти з програмою «Uvelir_3D», вибирати потрібні функції та виконувати відповідні дії.

Він дозволяє зручно та ефективно використовувати програму для розв'язання поставлених завдань.

Висновки до розділу 2

Отже, у цьому розділі було успішно розроблено алгоритми вирішення задач «Uvelir», «Uvelir_3D» шляхом побудови блок-схем алгоритмів цих задач. При створені алгоритми для вирішення цих задач було використано повний набір вхідних та вихідних даних задач.

РОЗДІЛ 3
ОРГАНІЗАЦІЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ЗАДАЧІ

Програмний комплекс «Uvelir» та «Uvelir_3D» буде використовувати одну базу даних «shop.mdb». В базі даних створюємо чотири таблиці: «Categories», «Products», «Klients», «Zakaz». Нижче наведена структура цих таблиць.

«Categories» – таблиця, котра містить категорії товарів жіночих прикрас.

Опис властивостей стовпчиків таблиці «Categories» наведено у табл. 3.1.

Таблиця 3.1

Опис властивостей стовпчиків таблиці «Categories» реляційної бази даних

№ з/п	Назва поля таблиці	Тип даних	Властивості (ключі та інші обмеження)
1	**ID**	Autonumber (Long Integer)	Первинний ключ
2	Name	Nchar(20)	Назва категорії
3	Catalog	Nchar(20)	Каталог для файлів зображень

«Products» – таблиця, в якій описана повна характеристика представлених жіночих прикрас для продажу.

Опис властивостей стовпчиків таблиці «Products» наведено у табл. 3.2.

«Klients» – ця таблиця клієнтів-покупців жіночих прикрас.

Опис властивостей стовпчиків таблиці «Klients» наведено у табл. 3.3.

Таблиця 3.2

Опис властивостей стовпчиків таблиці «Products» реляційної бази даних

№ п/п	Назва поля таблиці	Тип даних	Властивості (ключі та інші обмеження)
1	**ID**	Autonumber (Long Integer)	Первинний ключ
2	**ID_cat**	Long Integer	Зовнішній ключ (для зв’язку з таблицею Categories)
3	Name	Nchar(20)	Назва товару
4	Image	Nchar(50)	Ім’я файлу прикраси
5	Description	Nchar(50)	Додаткова інформація про товар
6	Price	Double	Ціна продажу товару у грн.

Таблиця 3.3

Опис властивостей стовпчиків таблиці «Klients» реляційної бази даних

№ п/п	Назва поля таблиці	Тип даних	Властивості (ключі та інші обмеження)
1	**ID**	Autonumber (Long Integer)	Первинний ключ
2	FName	Nchar(20)	Ім’я клієнта
3	SName	Nchar(20)	Прізвище клієнта
4	Adress	Nchar(50)	Адреса клієнта
5	Post_index	Nchar(15)	Поштовий індекс клієнта
6	Telefon	Nchar(15)	Телефон клієнта
7	Email	Nchar(20)	E-mail клієнта

«Zakaz» – ця таблиця дає нам інформацію, стосовно замовлень продажу жіночих прикрас. Опис властивостей стовпчиків таблиці «Prodaga» наведено у табл. 3.4.

Таблиця 3.4

Опис властивостей стовпчиків таблиці «Zakaz» реляційної бази даних

№ п/п	Назва поля таблиці	Тип даних	Властивості (ключі та інші обмеження)
1	**ID**	Autonumber (Long Integer)	Первинний ключ
2	**ID_klient**	Long Integer	Зовнішній ключ (для зв’язку з таблицею Klients)
3	**ID_prod**	Long Integer	Зовнішній ключ (для зв’язку з таблицею Zakaz)
4	Data_zak	Date/Time	Дата продажу товару
5	Cena	Double	Ціна продажу товару, грн.
6	Kol	Long Integer	Кількість одиниць товару

Коли вже всі необхідні таблиці мають свою назву, повністю описані всі їх поля, можна тепер навести схему даних, тобто наглядно представити встановлені зв’язки між таблицями (рис. 3.1).

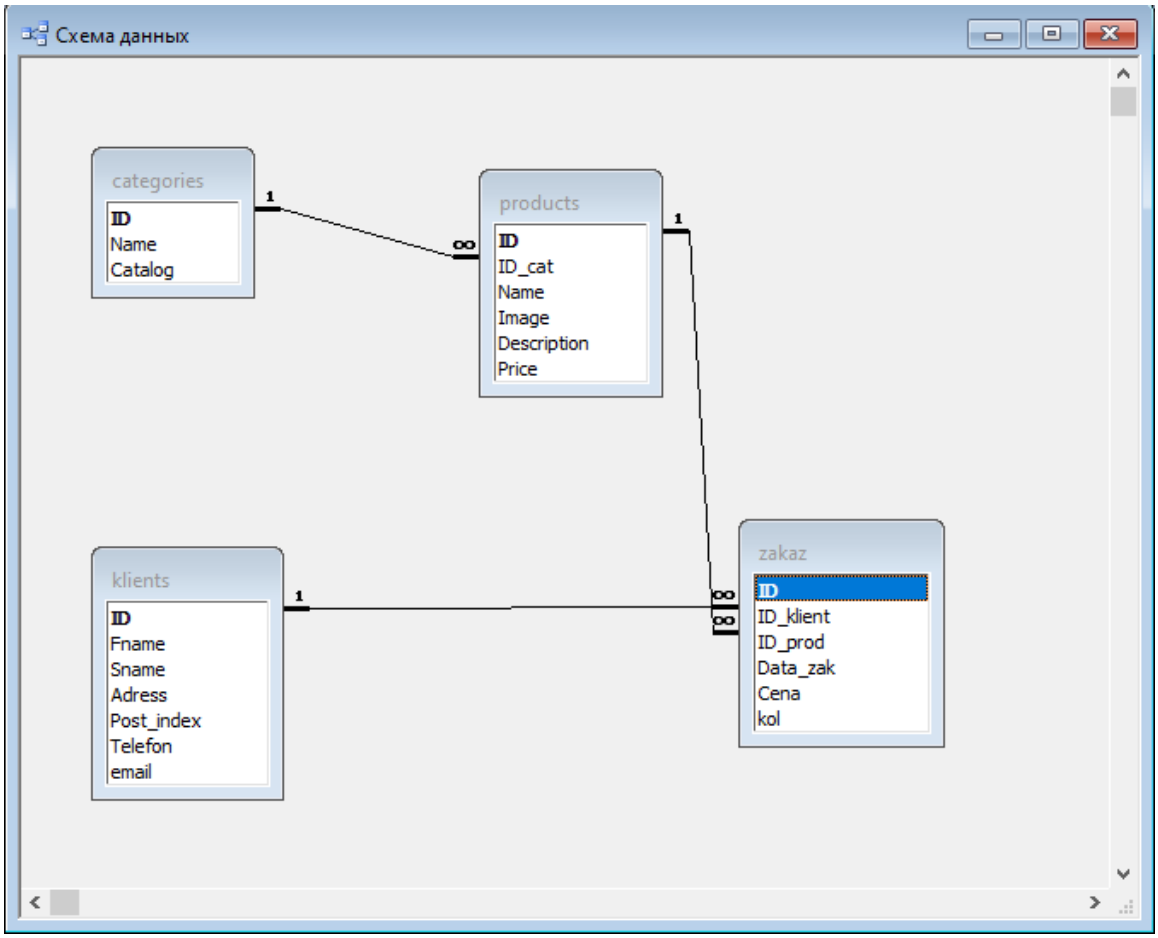


Рис 3.1. Схема бази даних «shop.mdb»

Висновки до розділу 3

В цьому розділі було розглянуто характеристику бази даних MS Access, яка дозволяє ефективно організувати продаж жіночих прикрас різних категорій. Наведена схема даних дозволяє ефективно вести довідники клієнтів, а також різні категорії жіночих прикрас, які призначено для продажу.

РОЗДІЛ 4
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАДАЧІ

4.1. Опис програми продажу жіночих прикрас «Uvelir»

Технологія обробки інформації для отримання кінцевого результату на мові програмування C# передбачає створення об’єктів інтерфейсу користувача та об’єктів обробки та відображення інформації, до яких слід навести: екранні форми для вводу, обробки та відображення інформації на екрані, запити до бази даних для обробки та відображення інформації та звіти для обробки та відображення інформації у режимі попереднього перегляду або на принтері.

За допомогою запитів існує можливість формувати підсумкову інформацію, виконувати різноманітні розрахунки над даними та відбирати записи бази даних, що задовольняють відповідним критеріям пошуку. Результатом виконання запитів до бази даних завжди є таблиця, яка може містити як первинну, так і результатну або проміжну інформацію. Тому таблицю-запит часто називають результуючою таблицею.

Для реалізації поставленої задачі було обрано середовище Microsoft Visual Studio 2019, мова програмування C#, так як вона надає найбільш широкі можливості для програмування додатків для ОС Windows з використанням сучасної потужної платформи .NET FrameWork [1-3].

Даний додаток працює на будь-якій сучасній операційній системі Windows, де має місце платформа .Net FrameWork версії 4.5.

Переваги мови C# в порівнянні з аналогічними програмними продуктами [5-8]:

1. Швидкість розробки додатків за допомогою Windows Forms;

2. Універсальність роботи додатку та обробка виключних ситуацій, використовуючи try…catch;

3. синхронізація з додатками, написаними на платформі .NET;

4. використання нової універсальної технології ADO .NET для обробки баз даних.

5. Створення нових компонентів, використовуючи переваги об’єктно-орієнтованого програмування.

Головна форма програми

Розробка програмного забезпечення реалізації задачі продажу жіночих прикрас виконана на сучасній мові програмування C# у середовищі Visual Studio 2019. Дане програмне забезпечення виконано з використанням Windows Forms. Всі візуальні класи програми похідні від класу Form. Інтерфейс головної форми програми приведено на рис. 4.1.

Рис. 4.1. Головна форма програми

У головній формі передбачено п’ять опцій меню [додаток А]:

1. Робота з БД.

2. Купівля товару.

3. Налаштування.

4. Гістограма в HTML.

5. Про програму.

Опція «Робота з БД» складається з двох пунктів меню:

1. «Продукція» – ведення інформації по жіночим прикрасам.

2. «Клієнти» – ведення довідника клієнтів.

Опція «Купівля товару» відповідає за оформлення продажу вибраної жіночої прикраси певним клієнтом з автоматизованим формуванням чеку та записом в базу даних.

Опція «Налаштування» складається з двох пунктів меню:

1. «Вибір шрифту» – вибір шрифту для виведення текстової звітної інформації.

2. «Вибір кольору» – вибір кольору відповідно для заголовка таблиці, заголовків стовпців таблиці та тексту таблиці.

Всі вибрані налаштування відображаються у вигляді звітної інформації у форматах *.xls та *.htm, а також при закритті програми записуються до файлу налаштувань «setup.ini».

При виконанні опції «Про програму» на екран виводиться інформація про розробника програмного забезпечення (рис. 4.2.).

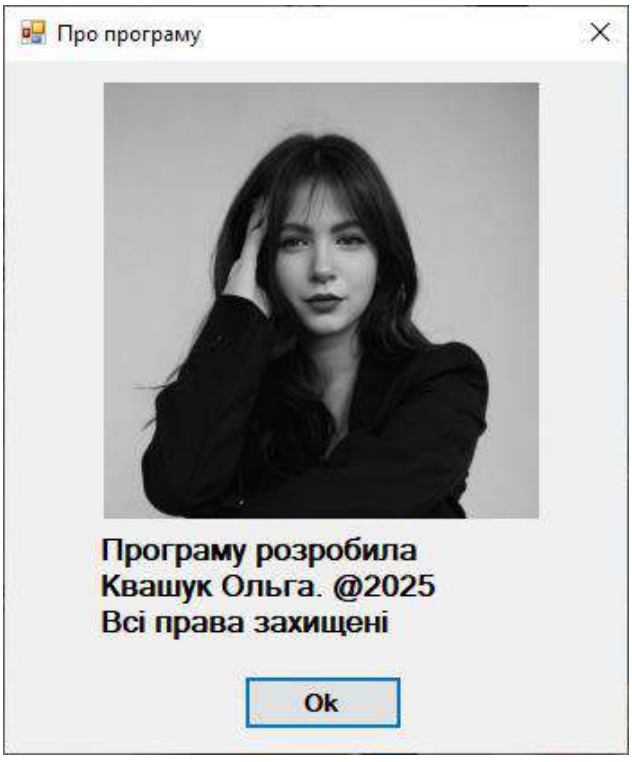


Рис. 4.2. Форма «Про програму»

У головній формі програмі відбувається підключення до бази даних «shop.mdb» з використанням об’єкту OleDBConnection. Потім даний об’єкт передається до інших форм програми.

Розробка програмного забезпечення та тестування задачі

Розглянемо клас Form3 для ведення довідник довідника клієнтів.

Проектування форми виглядає наступним чином (рис. 4.3).

Зверху форми знаходиться елемент напису Label, посередині елемент роботи з таблицями DataGridView, елемент знизу – рядок навігації даних (клас BindingNavigator). Слід зауважити, що до рядку навігації додано кнопку «Оновити» для оновлення даних в таблиці БД, а також кнопки із значком «xls» для експорту даних в MS Excel та кнопки із значком «html» для експорту даних в формат html.

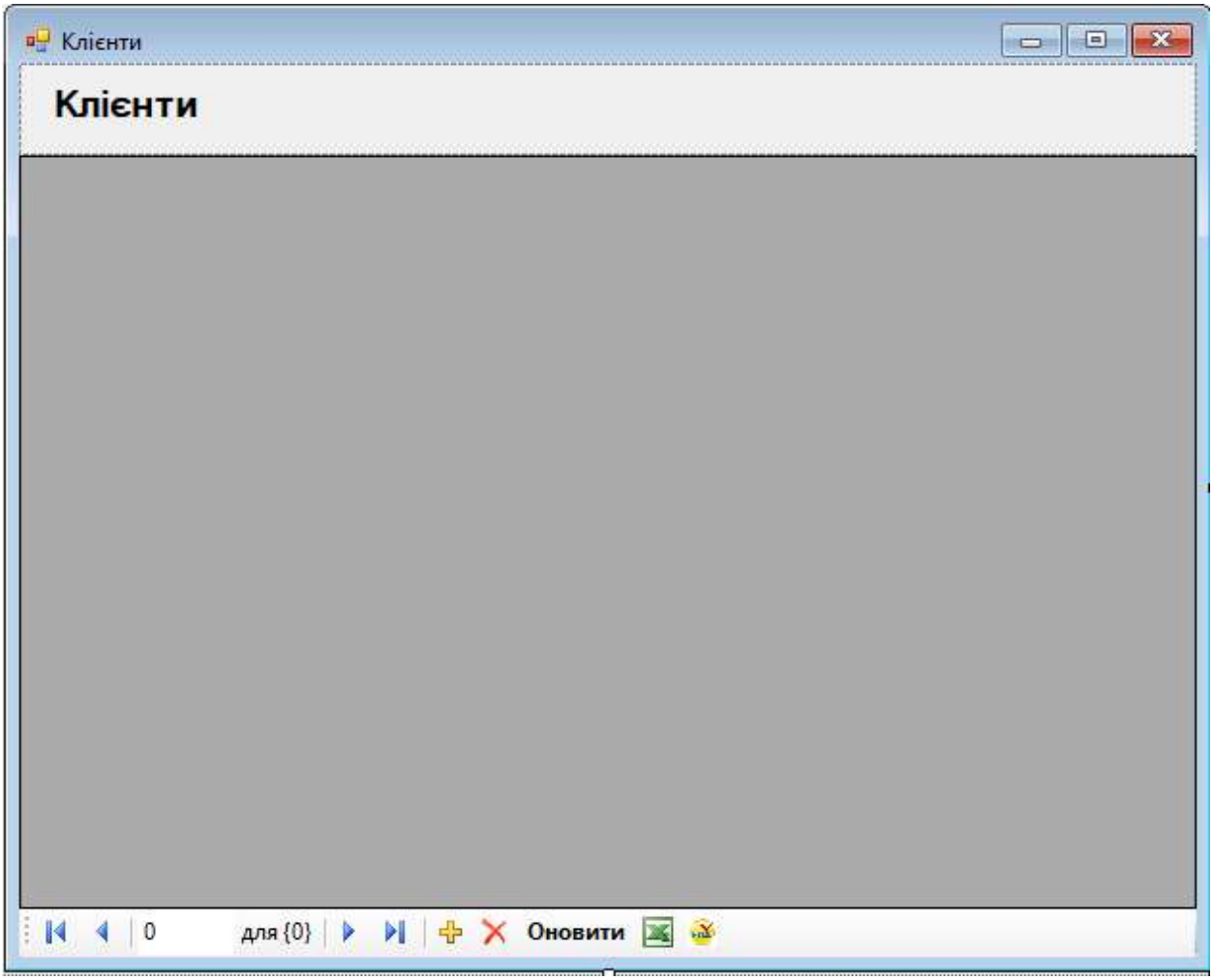


Рис. 4.3. Проектування форми Form3

Наведемо список та коротку характеристику основних функцій класу Form3.

1. Form3_Load – функція завантаження форми.
2. toolStripButton1_Click – функція оновлення даних.
3. toolStripButton2_Click – функція експорту даних в MS Excel.
4. toolStripButton3_Click – функція експорту даних в HTML.

Приведемо код функції Form3_Load.

```
//Завантаження форми
        private void Form3_Load(object sender, EventArgs e)
        {
            try
            {

                if (da != null) da.Dispose();
                da = new OleDbDataAdapter("Select * From Klients", conn);

                OleDbCommandBuilder bulder = new OleDbCommandBuilder(da);

                if (ds != null) ds.Dispose();
                ds = new DataSet();
                da.Fill(ds);


                //Прив'язка dataGridView1 до об'єкту DataSet
                bindingSource1.DataSource = ds.Tables[0];
                bindingNavigator1.BindingSource = bindingSource1;
                dataGridView1.DataSource = bindingSource1;

                ////////////////////////////////////////////
                dataGridView1.Columns[0].Visible = false;

            }

            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }
```

У даній функції створюється об’єкт класу адаптеру da для взаємодії з таблицею «Klients» бази даних. З використанням цього об’єкту створюється набір записів, який прив’язується до елементу DataGridView. Після цього можна у повному обсязі редагувати дані таблиці – додавати, видаляти, модифікувати записи. Слід зауважити, що всі зміни відбуваються тільки в елементі DataGridView. Щоб відповідні зміни відбулися у базі даних – необхідно натиснути кнопку оновлення даних. За формування запитів на

оновлення (UPDATE), видалення (DELETE) та додавання (INSERT) даних відповідає клас OleDBCommandBuilder.

Наведемо код функції оновлення даних.

```
//Оновлення даних
        private void toolStripButton1_Click(object sender, EventArgs e)
        {
            try
            {
                da.Update(ds.Tables[0]);
            }

            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }
```

Після спрацювання даної функції буде внесено зміни до бази даних.

Наведемо код функцій відгуків експорту даних в MS Excel та HTML.

```
//Експорт у XLS
        private void toolStripButton2_Click(object sender, EventArgs e)
        {
            if (!Form1.ExportXLS(ds.Tables[0].DefaultView))
            {
                MessageBox.Show("Неможливий експорт у Excel!",
                     "Увага!", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }

        //Експорт у HTML
        private void toolStripButton3_Click(object sender, EventArgs e)
        {
            if (!Form1.Export_Html("Клієнти.htm", ds.Tables[0].DefaultView, "Клієнти",
                Form1.font_text, Color.Red, Form1.cvet_hapka, Form1.cvet_text))
            {
                MessageBox.Show("Неможливий експорт у HTML!",
                         "Увага!", MessageBoxButtons.OK, MessageBoxIcon.Information);
                return;
            }

            Form6 ff = new Form6();
            ff.webBrowser1.Navigate(Application.StartupPath + "\\" + "Клієнти.htm");
            ff.ShowDialog();
        }
```

На рис. 4.4. наведено результат виконання класу Form3 для ведення інформації клієнтам.

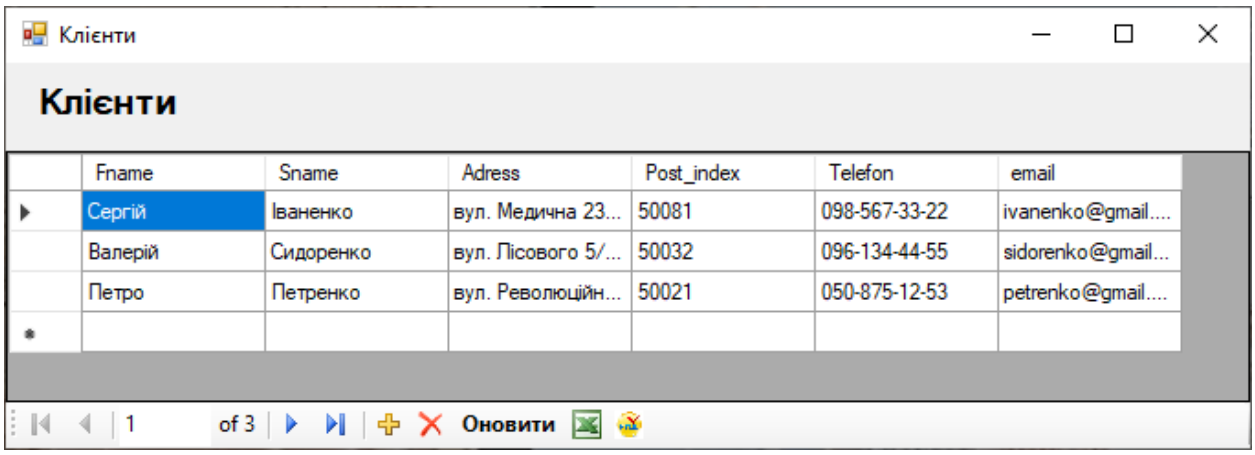


Рис. 4.4. Ведення довідника клієнтів

При натисненні на кнопку «xls» – інформацію буде виведено у MS Excel, звідки її можна роздрукувати у зручному для користувача вигляді (рис. 4.5).

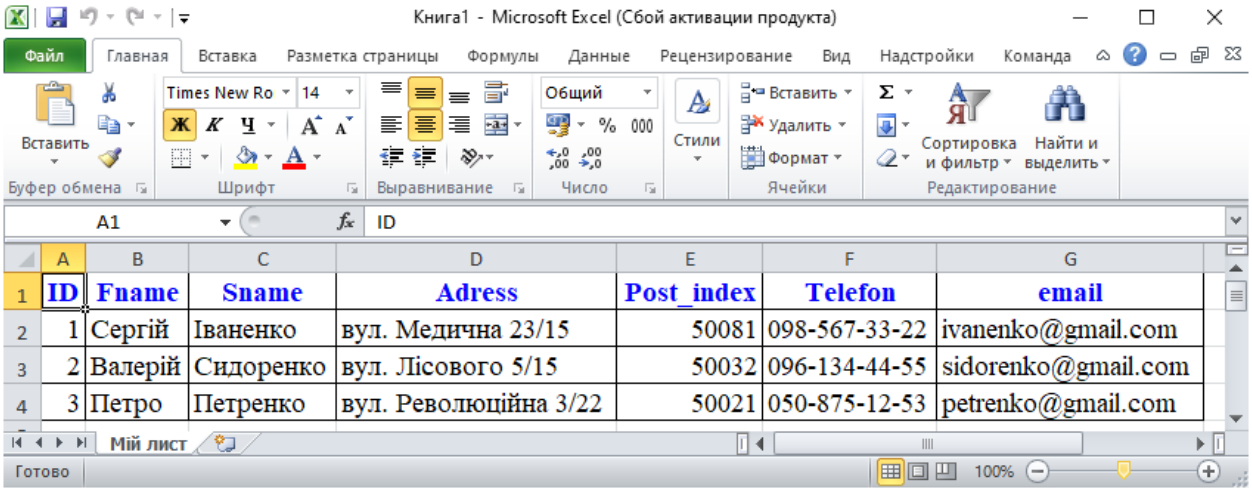


Рис. 4.5. Експорт даних до MS Excel

При натисненні на кнопку «html» – інформацію буде виведено у формат html (рис. 4.6). Код html-сторінки виводиться в розробленій програмі – у формі Form6. У якості елементу керування для перегляду web-сторінок використано елемент WebBrowser.

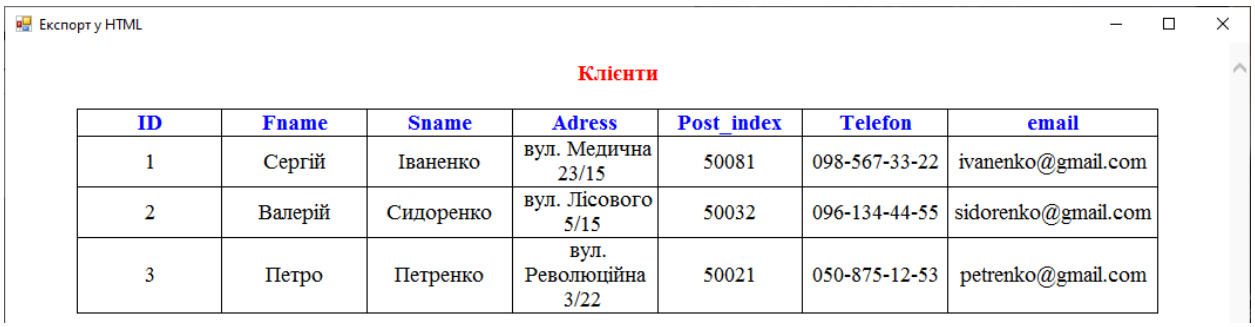


Рис. 4.6. Експорт даних в HTML

Наступним етапом є проектування класу Form2 де відбувається ведення інформації по продукції жіночих прикрас. Так як таблиці бази даних пов’язані відношенням один до багатьох, спочатку необхідно вибрати конкретну категорію, а потім по ній сформувати SQL-запит [додаток Б]. З цією метою спроектовано клас Name_List, який вміщує базову інформацію для будь-якого списку – строкове значення та значення ключового ідентифікатору – ItemData.

Наведемо код класу Name_List.

```
//Клас для заповнення списку
    public class Name_List
    {
        public string str;
        public int ItemData;

        public Name_List(string str, int ItemData)
        {
            this.str = str;
            this.ItemData = ItemData;
        }

        public override string ToString()
        {
            return str;
        }
    }
```

Спроектуємо форму Form2 для ведення даних інформації по продукції жіночих прикрас (рис. 4.7.).

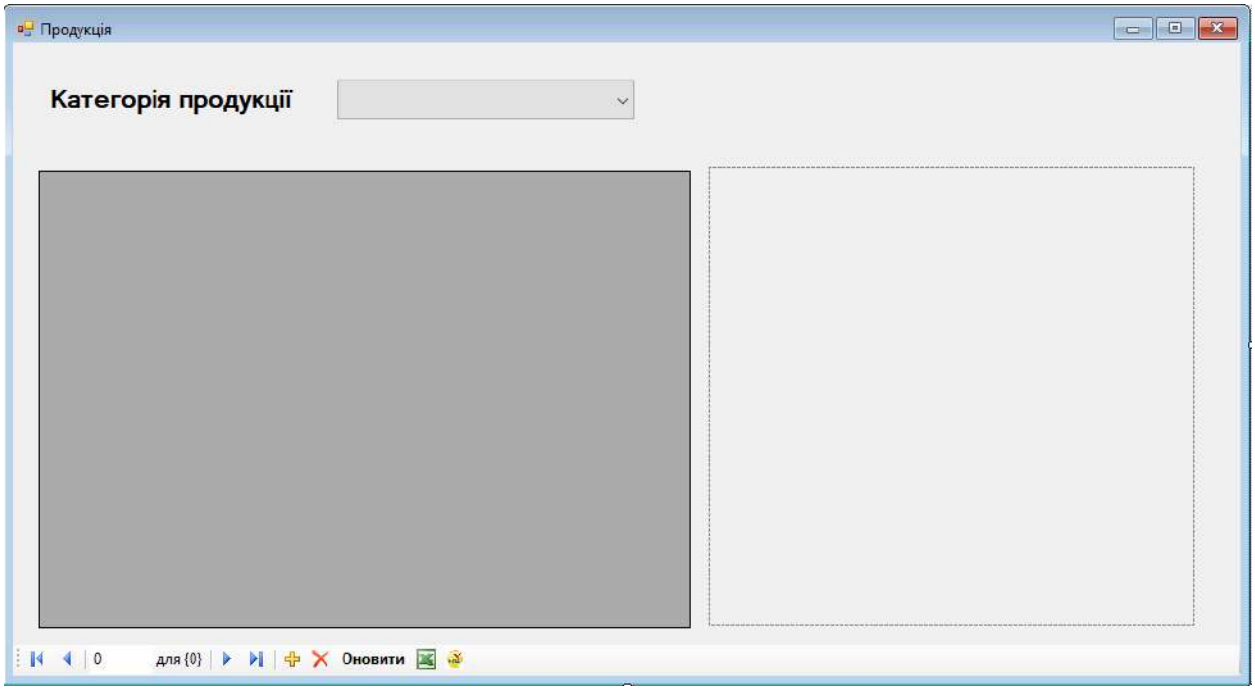


Рис. 4.7. Проектування форми Form2

Наведемо список та коротку характеристику основних функцій класу Form2.

1. Form2_Load – функція завантаження форми.
2. comboBox1_SelectedIndexChanged – вибір групи виконавця зі списку.
3. toolStripButton1_Click – функція оновлення даних.
4. toolStripButton2_Click – функція експорту даних в MS Excel.
5. toolStripButton3_Click – функція експорту даних в HTML.

Приведемо код функції Form2_Load.

```
private void Form2_Load(object sender, EventArgs e)
        {
            try
            {
                //Заповнення списку через DataReader
                OleDbCommand cmd = new OleDbCommand("Select * From Categories", conn);
                OleDbDataReader rd = cmd.ExecuteReader();
                while (rd.Read())
                    comboBox1.Items.Add(new
Name_List(rd[1].ToString(),Convert.ToInt32(rd[0])));

                comboBox1.SelectedIndex = 0;

            }

            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

У даній функції відбувається заповнення списку combobox1 інформацією по виконавцям з використанням класу Name_List. Для виконання запиту використовується об’єкт OleDBCommand. Результат виконання записується в об’єкт для читання OleDBDataReader, який застосовується для заповнення елементів керування та призначений тільки для читання записів.

Далі наведемо код функції comboBox1_SelectedIndexChanged, яка спрацьовує при виборі категорії продукції жіночих прикрас.

```
//Зміна категорії продукції
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            try
```

```
        {
            string str;
            int idCat = ((Name_List)comboBox1.SelectedItem).ItemData;
            str = "Select * From Products where ID_cat=" + idCat;

            if (da != null) da.Dispose();
            da = new OleDbDataAdapter(str, conn);

            OleDbCommandBuilder bulder = new OleDbCommandBuilder(da);

            bulder.QuotePrefix = "[";
            bulder.QuoteSuffix = "]";

            if (ds != null) ds.Dispose();
            ds = new DataSet();
            da.Fill(ds);
            ds.Tables[0].Columns[1].DefaultValue = idCat;

            //Прив'язка dataGridView1 до об'єкту DataSet
            bindingSource1.DataSource = ds.Tables[0];
            bindingNavigator1.BindingSource = bindingSource1;
            dataGridView1.DataSource = bindingSource1;

            ////////////////////////////////////////////
            dataGridView1.Columns[0].Visible = false;
            dataGridView1.Columns[1].Visible = false;

            ////////////////////////////////////////////
            bindingSource1.Position = 0;

        }

        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }

    }
```

При виборі категорії продукції зі списку формується запит по конкретному ідентифікатору. Результат відповідного запиту по категорії прив’язується до елементу DataGridView з подальшими можливостями редагування.

Слід звернути увагу на функцію зміни поточного запису bindingSource1_CurrentChanged, яка відповідає за зміну конкретного зображення продукції жіночих прикрас. Відповідне зображення завантажується у форматі «*.jpg» з папки «Image», яка знаходиться у поточному каталозі. У папці «Image» зображення розбиті по категоріям які знаходяться у таблиці «Categories» бази даних. Це наступні категорії: браслети, брелоки, брошки, буси, кільця, Намиста.

Наведемо код даної функції.

```
//Зміна поточного запису
        private void bindingSource1_CurrentChanged(object sender, EventArgs e)
        {
            //Завантаження зображення за маршрутом
            DataView dt = ds.Tables[0].DefaultView;
            string str = "Image\\" + comboBox1.SelectedItem.ToString() + "\\" +
dt[bindingSource1.Position][3];
            //MessageBox.Show(str);
            pictureBox1.Image = Image.FromFile(str);
        }
```

На рис. 4.8. наведемо результат виконання класу Form2.

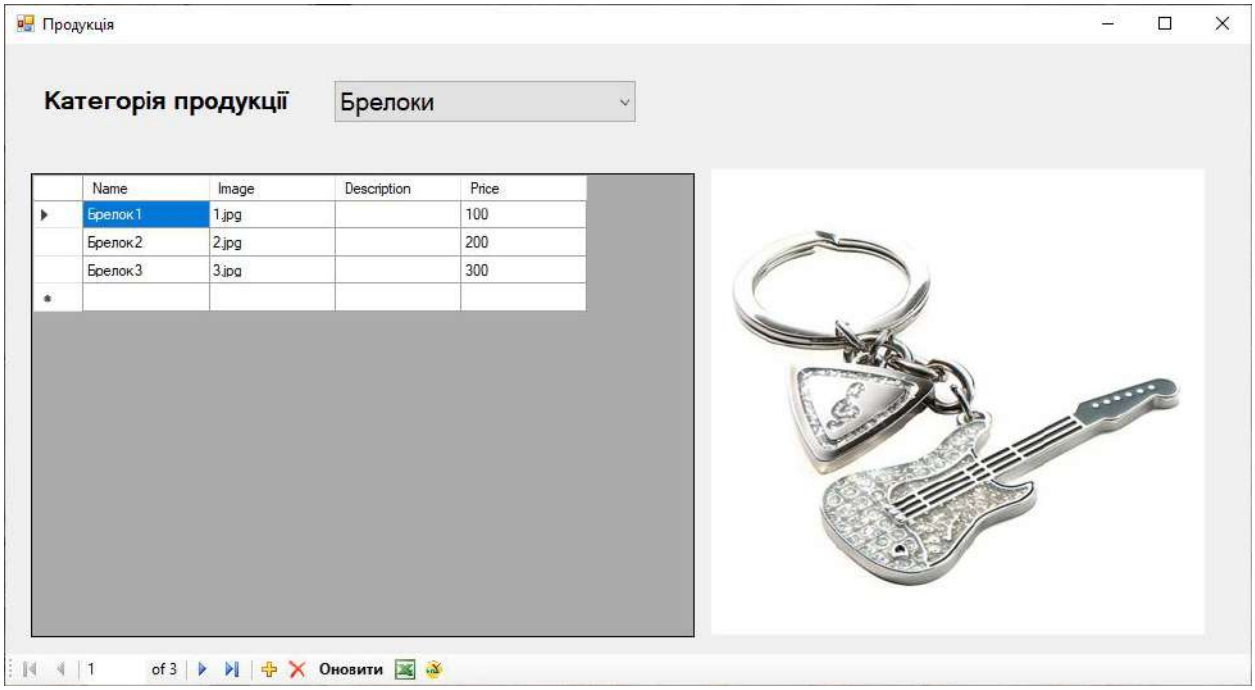


Рис. 4.8. Ведення інформації по продукції жіночих прикрас

Ведення бази даних по продукції жіночих прикрас є початковим етапом для оформлення продажу даних товарів з формуванням відповідної інформації.

Слід сказати, що окрім розроблених форм мають місце універсальні функції роботи з даними.

На наступному етапі розглянемо безпосередньо автоматизацію продажу жіночих прикрас. Спроектуємо форму для оформлення продажу жіночих прикрас Form4 (рис. 4.9).

Як видно з рис. 4.9. для оформлення продажу треба вибрати клієнта за

його ідентифікатором, та відповідний товар, який він хоче придбати. За допомогою елементу MonthCalendar можна вибрати дату оформлення продажу. За умовчанням буде відображатись поточна дата.

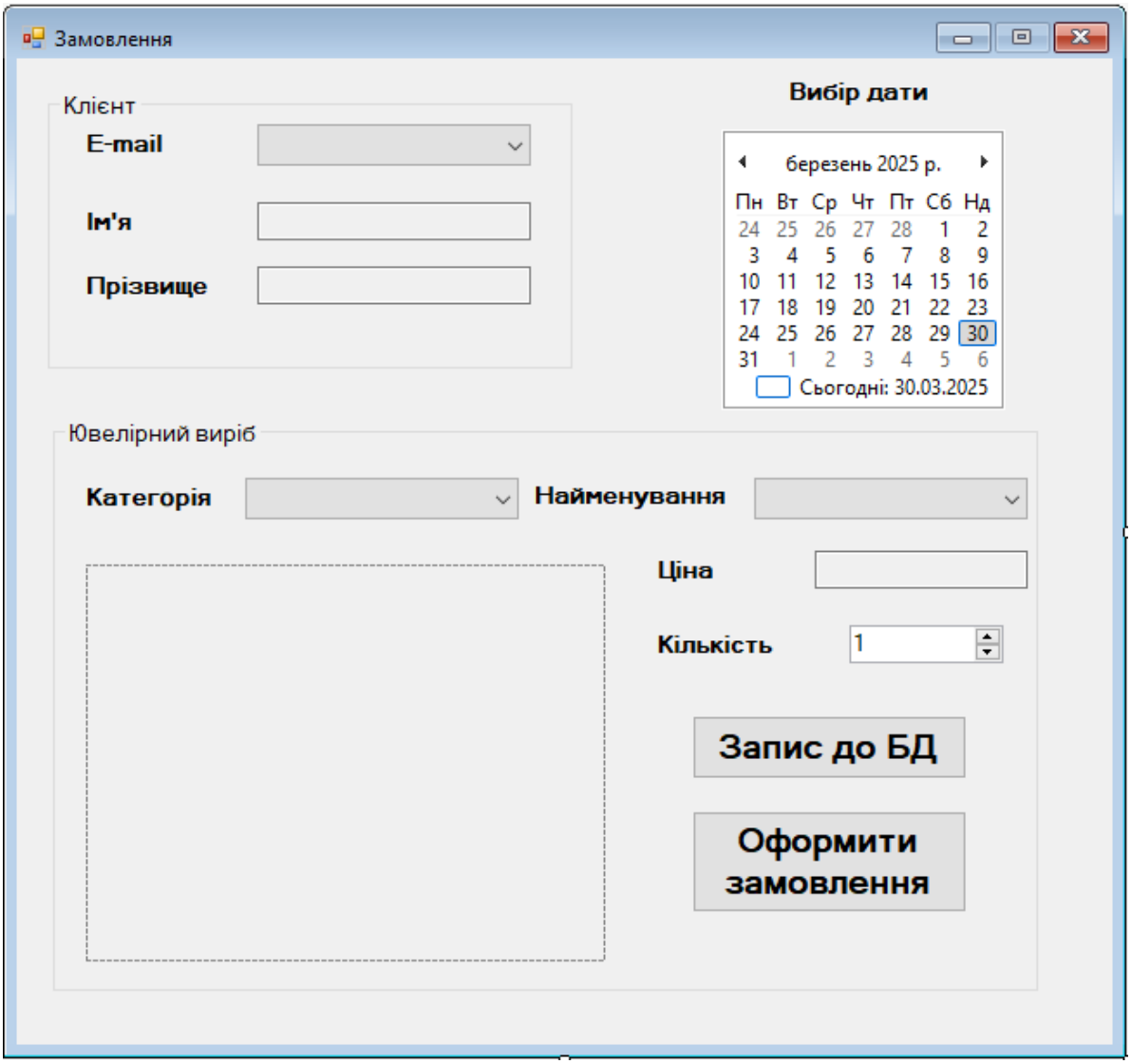
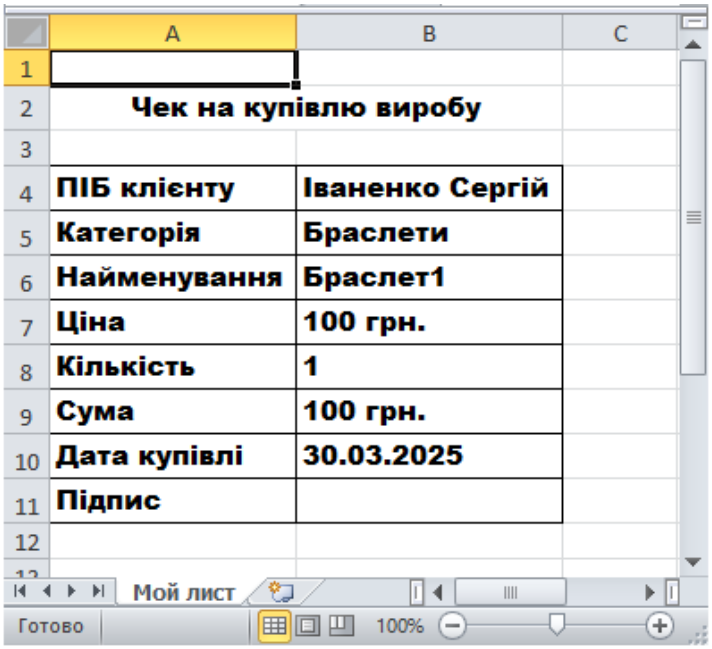


Рис. 4.9. Проектування форми Form4 «Оформлення продажу жіночих прикрас»

При натисканні на кнопку «Оформити замовлення» у середовищі MS Excel у автоматизованому режимі буде сформовано чек, який можна роздрукувати. При натисканні кнопки «Запис до БД» інформація щодо продажу вибраного диску буде записана до бази даних до таблиці «Zakaz».

На рис. 4.10. наведемо автоматизоване формування чеку при натисканні на кнопку «Оформити замовлення».



	A	B	C
1			
2	**Чек на купівлю виробу**		
3			
4	**ПІБ клієнту**	**Іваненко Сергій**	
5	**Категорія**	**Браслети**	
6	**Найменування**	**Браслет1**	
7	**Ціна**	**100 грн.**	
8	**Кількість**	**1**	
9	**Сума**	**100 грн.**	
10	**Дата купівлі**	**30.03.2025**	
11	**Підпис**		
12			

Рис. 4.10. Формування чеку на купівлю товару жіночих прикрас

Результат виконання форми «Оформлення продажу» приведено на рис. 4.11.

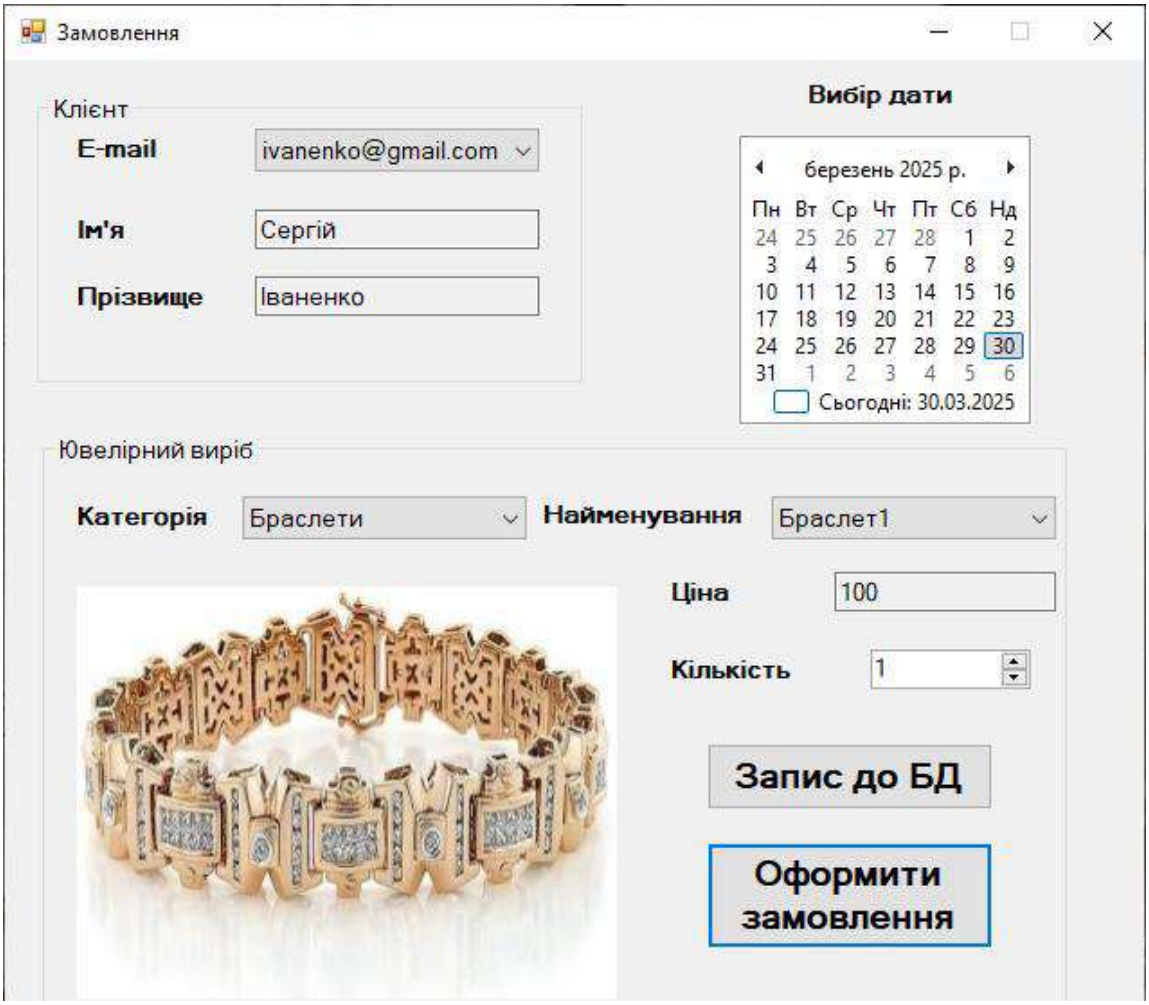


Рис. 4.11. Результат виконання форми «Оформлення продажу»

Так, у візуальному режимі клієнту надається повна інформація про диск та його виконавця з графічною ілюстрацією у вигляді файлу *.jpg (елемент керування PictureBox).

Змінити шрифт можливо натиснувши пункт меню «Вибір шрифту». У відкритому діалозі зміни можливо здійснити всі необхідні зміни, що спростили б використання запропонованої програми (рис. 4.12).

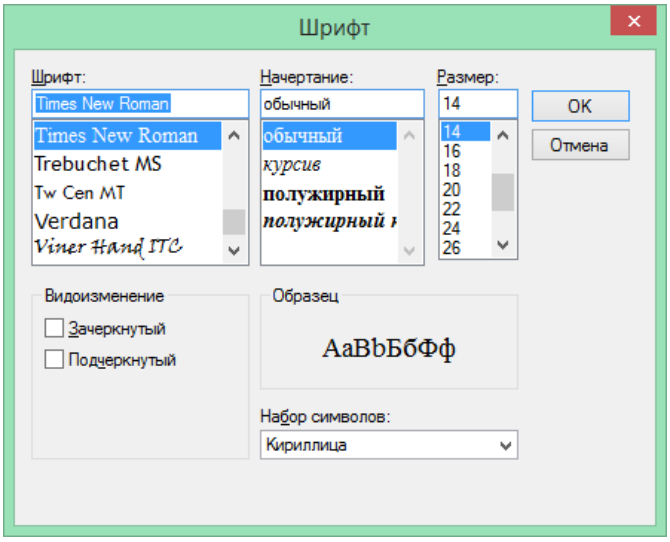


Рис. 4.12. Вікно вибору шрифту

Як видно з наданого малюнка, можливо змінити текст використовуючи курсив або виділити його жирним.

Для обрання необхідного кольору тексту, необхідно натиснути пункт меню «Вибір кольору» (рис. 4.13). Змінювати текст можливо також, використовуючи палітру, а не стандартні кольори.

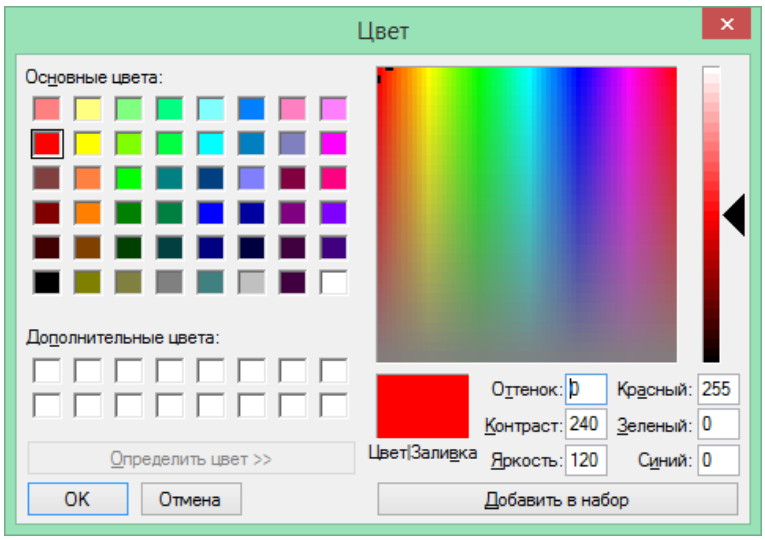


Рис. 4.13. Вікно вибору кольору

4.2. Опис програми продажу жіночих прикрас «Uvelir_3D»

За допомогою мови Visual C++ та використання бібліотек OpenGL було створене 3D зображення гістограми, для того щоб наочно проаналізувати ціни на жіночи прикраси, причому напроти кожного стовпчика з ціною виводяться безпосередньо зображення прикрас. Була передбачена робота зі світлом, виведення тіні стовпчиків, виведення тексту, накладення текстур на землю, градієнтна заливка [додаток В].

Наведемо фрагмент коду зчитування інформації з бази даних, використовуючи технологію ADO.

```
CString str_con=(L"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=shop.mdb");

	try
	{

	 pConn->Open((_bstr_t)str_con,"","",0);

	 //Заполнение массивов и закрытие набора данных

	 CString str = L"Select [Categories.catalog], [Products.name], [Products.image],
[Products.price] From [Categories] Inner Join [Products] On Categories.ID =
Products.Id_cat";

	 pRecordset = pConn->Execute((_bstr_t)str,0,adCmdText);
	 //return;

	 FieldsPtr fields = pRecordset->Fields;
	 pRecordset->MoveFirst();
	 long i=0;
	 while(!pRecordset->ADO_EOF)
	 {
		name[i] = (char*)(_bstr_t)fields->GetItem(0L)->GetValue();
		name[i]+=(char*)"\\";
		name[i]+= (char*)(_bstr_t)fields->GetItem(2L)->GetValue();
		mas[i] =  fields->GetItem(3L)->GetValue();
		pRecordset->MoveNext();
		i++;
	 }


	 pRecordset->Close();
	 pRecordset.Release();
	 pConn->Close();
	 pConn.Release();

	 n = i;
	}
	catch(_com_error &ce)
	{
		ErrMessage(ce);
		if(pConn->GetState())pConn->Close();
		return;
	}
```

На рис. 4.14. виведено гістограму по цінам жіночих прикрас.

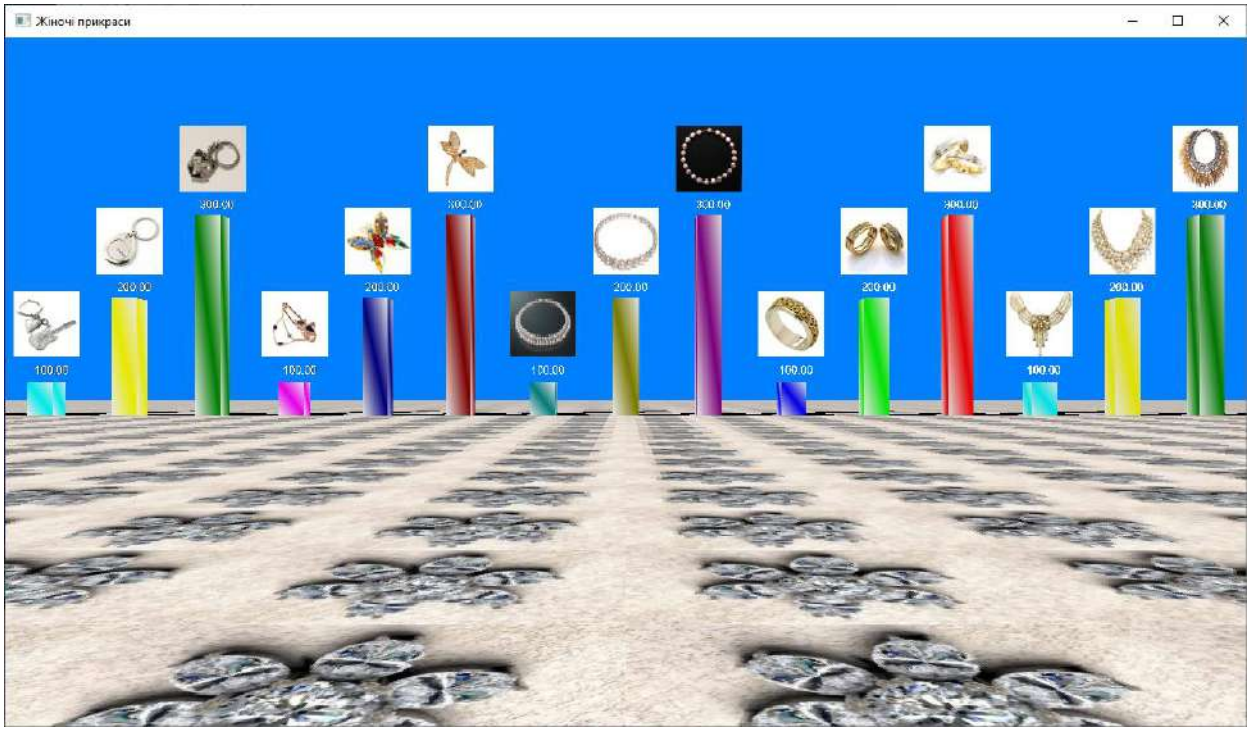


Рис. 4.14. Виведення 3D-гістограми по цінам жіночих прикрас

Програмний код, завдяки якому реалізовано виведення діаграм з використанням бібліотеки OpenGL:

```
void GR_3D::OnPaint()
{
	CPaintDC dc(this); // device context for painting
	//
	CRect clientRect;
	GetClientRect(&clientRect);
	glViewport(0, 0, clientRect.right, clientRect.bottom);

	glPushMatrix();//сохранение текущей матрицы
	glClearColor(1, 1, 1, 1); // фон
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);// буфер
цвета и глубину
	glPushAttrib(GL_ALL_ATTRIB_BITS);//сохранение текущих
настроек цвета
	//-----------------------------------------------------2
	//определение координат
	n1=n2=n;
	if (clientRect.right > clientRect.bottom)    n1=n *
(float)clientRect.right  / clientRect.bottom;
	else                                                n2=n *
(float)clientRect.bottom  / clientRect.right;
	glOrtho (      //етот режим лучше стандартного -- более
гибкий -- любые системы координат
```

```
            -n1,  n1,
            -n2,  n2,
            -n*2, n*2);//
      //=====================================================2
      glRotatef(xRot, 1.0f, 0.0f, 0.0f);
      glRotatef(yRot, 0.0f, 1.0f, 0.0f);

      (this->*delegate1)();

      glPopMatrix();//восстановление матрицы
      glPopAttrib();//восстановление настроек цвета
      glFinish();
      SwapBuffers(wglGetCurrentDC()); // Вывод в контекст
}
```

Спочатку функції виведення відбувається очищення буферів глибини та кольору. Потім визначається співвідношення розмірів вікна для малювання, щоб діаграма поміщувалась повністю в вікні для виведення. Варто звернути у вагу на те, що в приведеному лістингу використовується вказівка на функцію (`this->*delegate1)`, котра зберігає адресу функції для виведення потрібної діаграми. Завдяки цьому в програмі досягається скорочення коду до одного рядку замість декількох конструкцій для переключення між функціями «if () else». І нарешті в кінці функції відбувається виведення в контекст (тобто наше вікно) намальованої графіки.

Для ініціалізації бібліотеки OpenGL та формування списку шрифтів для виведення тексту використовується наступна функція:

```
bool GR_3D::Init_3d()
{
      HGLRC           hrc;  // дескриптор OPENGL
      int pixelformat;

      if ((pixelformat = ChoosePixelFormat(GetDC()->GetSafeHdc(),
&pfd)) == 0 ||
            SetPixelFormat(GetDC()->GetSafeHdc(), pixelformat,
&pfd) == FALSE)
            return false;

      ::DescribePixelFormat(GetDC()->GetSafeHdc(), pixelformat,
sizeof(pfd), &pfd);
      if (pfd.cColorBits<16)return false;

      hrc = wglCreateContext(GetDC()->GetSafeHdc());// создание
контекста
```

```
     wglMakeCurrent(GetDC()->GetSafeHdc(), hrc);    // установка
контекста

    //-----------------------------------------------------2
     // Формирование списка шрифтов в OpenGL
     CFont font;
     font.CreatePointFont(10,_T("Arial"));
     SelectObject(wglGetCurrentDC(),font.m_hObject);

    wglUseFontOutlinesA(wglGetCurrentDC(), 0, 255,
    GLF_START_LIST, 0,0, WGL_FONT_POLYGONS, 0);
    //=====================================================2

     glEnable(GL_DEPTH_TEST); // для 3 мерки
     return true;
}
```

Також виведені діаграми можливо обертати за допомогою миші. Така функція виявилась можливою завдяки взаємодії двох функцій-відгуків на події:

а) натиснення лівої клавіші миші:

```
void GR_3D::OnLButtonDown(UINT nFlags, CPoint point)
{
     // TODO: добавьте свой код обработчика сообщений или вызов
стандартного
     anchor=point;
     HCURSOR hCur = AfxGetApp()->LoadStandardCursor(IDC_SIZEALL);
     SetCursor(hCur);

     CDialogEx::OnLButtonDown(nFlags, point);
}
```

Функція OnLButtonDown запамятовує в змінну anchor координати положення миші і таким чином ми отримуємо координати точки відліку.

б) руху миші:

```
void D2_DIAGRAM::OnMouseMove(UINT nFlags, CPoint point)
{
     // TODO: добавьте свой код обработчика сообщений или вызов
стандартного
     if (nFlags & MK_LBUTTON)
     {
          HCURSOR hCur = AfxGetApp()-
>LoadStandardCursor(IDC_SIZEALL);
          SetCursor(hCur);

          int
               x = point.x-anchor.x,
               y = point.y-anchor.y;
          float angle=0.1f;
          xRot += angle * y;
```

```
            yRot += angle * x;

            Invalidate(false);

            anchor=point;
        }

        CDialogEx::OnMouseMove(nFlags, point);
}
```

Функція OnMouseMove під час руху миші порівнює координати початкової точки- відліку і змінює змінні xRot та yRot пропорційно переміщенню курсора миші. Потім виклик функції Invalidate(false) дає команду для оновлення зображення у вікні програми і функція void **GR_3D**::OnPaint() виконує оновлення зображення у вікні програми із застосуванням змінних xRot та yRot, котрі являються кутами поворотів навколо осей X та У. Таким чином не витрачається час на очищення вікна від старого зображення завдяки нульовому false – аргументу функції Invalidate, а одразу завантажується зображення, сформоване бібліотекою OpenGL. Таким чином досягається висока частота оновлення зображення на моніторі, що дозволяє користувачеві сприймати зображення вікна приблизно в такій же якості як і при перегляді кінофільмів.

Слід зазначити, що використані процедури та функції є універсальними та їх легко пристосовувати до будь-яких змін в програмному коді. Вищевказана функція розрахунку використовується також у створенні графічного звіту, що спрощує передачу даних та обчислення щодо графічного виведення інформації.

```
//Функция инициализации текстур
bool GR_3D::Init_Textur()
{
if(_access("BMP",0))
{
 MessageBox(L"Отсутствует папка BMP,\nгде должны находится файлы *.bmp\nдля организации
текстур \n",L"Внимание!",MB_ICONINFORMATION);
 return 0;
}
CString str;
if(_access("BMP\\textur.bmp",0))str=L"textur.bmp\n";
if(_access("BMP\\textur2.bmp",0))str+=L"textur2.bmp\n";
if(str.GetLength ())
{
 str = L"В папке BMP отсутствуют следующие файлы:\n" + str;
 AfxMessageBox(str,MB_ICONINFORMATION);
 return 0; }
```

Слід зазначити, що використані процедури та функції є універсальними та їх легко пристосовувати до будь-яких змін в програмному коді. Вищевказана функція розрахунку використовується також у створенні графічного звіту, що спрощує передачу даних та обчислення щодо графічного виведення інформації.

Технологія обробки інформації для отримання кінцевого результату в Visual C# передбачає створення об’єктів інтерфейсу користувача та об’єктів обробки та відображення інформації, до яких слід навести: екранні форми для вводу, обробки та відображення інформації на екрані, запити до бази даних для обробки та відображення інформації та звіти для обробки та відображення інформації у режимі попереднього перегляду або на принтері.

За допомогою запитів існує можливість формувати підсумкову інформацію, виконувати різноманітні розрахунки над даними та відбирати записи бази даних, що задовольняють відповідним критеріям пошуку. Результатом виконання запитів до бази даних завжди є таблиця, яка може містити як первинну, так і результатну або проміжну інформацію. Тому таблицю-запит часто називають результуючою таблицею.

Створений програмний продукт виступає додатком до створеної програми на мові C#, тобто метою його створення є швидке та багатофункціональне адміністрування інформації в базі даних. Слід зазначити, що створена програма завдяки своїй простоті, зручності та контролю введення даних гарантує швидку взаємодію користувача із таблицями бази даних.

4.3. Опис скриптів для формування діаграм у форматі HTML

У якості web-частини даного програмного забезпечення вибрано автоматичне формування скриптів на основі елементу керування chart.js. З цією метою розроблено універсальну функцію, до якою треба передати два масиви вхідних параметрів – числові параметри по яким будується діаграма,

а також рядкові параметри для виведення підказок та легенди стовпчиків.

Наведемо код даної функції:

```
//Універсальна функція скрипта для побудови діаграм в Web
        public static void SaveHtmlFile(string fileName, string title, string chartLabel,
string[] countries, double[] population)
        {
            // Формуємо HTML-контент
            string htmlContent = $@"
<!DOCTYPE html>
<html lang='uk'>
<head>
    <meta charset='UTF-8'>
    <meta name='viewport' content='width=device-width, initial-scale=1.0'>
    <title>{title}</title>
    <script src='https://cdn.jsdelivr.net/npm/chart.js'></script>
    <style>
        body {{
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color: #f4f4f4;
        }}
        canvas {{
            width: 100vw !important;
            height: 100vh !important;
        }}
    </style>
</head>
<body>
    <canvas id='populationChart'></canvas>
    <script>
        const ctx = document.getElementById('populationChart').getContext('2d');
        new Chart(ctx, {{
            type: 'bar',
            data: {{
                labels: {GenerateArrayForJS(countries)},
                datasets: [{{
                    label: '{chartLabel}',
                    data: {GenerateArrayForJS(population)},
                    backgroundColor: 'rgba(75, 192, 192, 0.6)',
                    borderColor: 'rgba(75, 192, 192, 1)',
                    borderWidth: 1
                }}]
            }},
            options: {{
                responsive: true,
                maintainAspectRatio: false,
                scales: {{
                    y: {{
                        beginAtZero: true
                    }}
                }}
            }}
        }});
    </script>
</body>
</html>";
            // Записуємо HTML в файл
            File.WriteAllText(fileName, htmlContent);         }
```

Дана функція є універсальною і дозволяє автоматично формувати Java-скрипти по даним які беруться з бази даних та відповідна інформація відразу відкривається з браузеру. Це дозволяє користуватися статистикою цін на жіночі прикраси не тільки з дек стоп-додатку, а і з мобільних пристроїв.

Гістограма в HTML статистики ціни на жіночі прикраси продемонстрована на рис. 4.15.

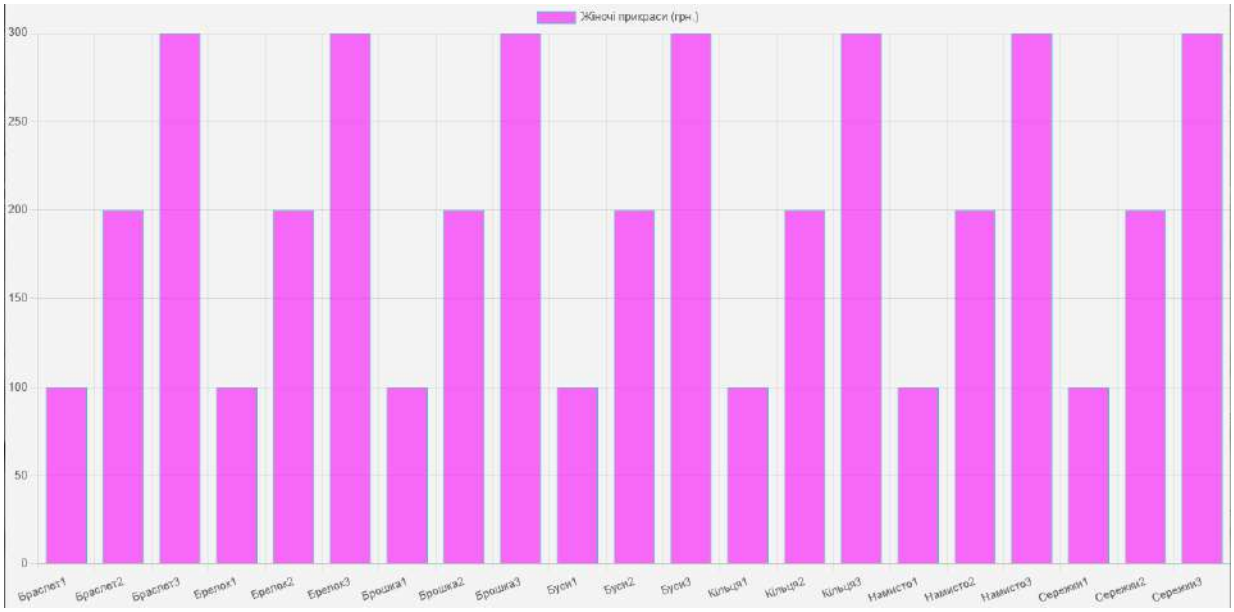


Рис. 4.15. Гістограма в HTML зміни цін на жіночі прикраси

Висновки до розділу 4

Отже у четвертому розділі продемонстроване розроблене програмне забезпечення продажу жіночих прикрас, яке складається з трьох частин: програма «Uvelir» розроблена на C#; програма «Uvelir_3D» виводить статистику цін по жіночим прикрасам, використовуючи гістограму та завантажене зображення товарів у за допомогою текстур; у якості web-частини застосовується автоматичне формування Java-скриптів для формування статистичної інформації по ціні жіночих прикрас.

ВИСНОВКИ

У результаті виконання дипломної роботи було створене програмне забезпечення продажу жіночих прикрас.

Було сформульовано характеристику задачі, мету її вирішення та цілі, які вона переслідує. Була визначена структура та зміст вхідної та вихідної інформації, основні вимоги до їх оформлення. Джерела, які забезпечують задачу вхідною інформацією, та особи, які мають отримати вихідну інформацію.

В процесі роботи було обрано методи розв'язання задачі - методи теорії алгоритмів, методи об'єктно-орієнтованого проектування.

Для реалізації задачі було створено базу даних, призначену для зберігання вхідної інформації. Для розв’язання задачі підрахунку максимальної швидкості автомобіля та побудови 3D-графіки бездоганно підійшла база даних Access. Відповідно до розробленої моделі, був визначений алгоритм побудови програми. Для реалізації алгоритму задачі на було обрано мову програмування C#, C++. Та підключення незалежної від середовища графічної бібліотеки **OpenGL** (Open Graphics Library).

Дана розробка вирішує такі задачі:

- продажа жіночих прикрас;
- графічне представлення;
- швидкість та зручність роботи з інформаційною базою даних;
- підрахунок та графічне представлення.

Головним результатом роботи стала розробка програмного продукту, який полегшить роботу и закази магазину ювелірних виробів.

Також важливо зазначити, що кожен із представлених інформаційних масивів використовується як довідкова інформація. А вибірка точних значень та подальше її використання відбувається методом поблочного запису.

Складовими кожної з таблиць є точно підібрані поля, котрі за собою тягнуть ряд значень. Кожне із значень розширює інформаційне

представлення по об’єкт не залежно від стану та інформаційного спрямування.

Створена структура даних дає змогу зберігати інформацію і яка б максимально описувала предметне поле, але при цьому займала мало цифрового місця. При створенні бази даних максимально використано правила оптимізації та групування даних. Такий підхід дав змогу звільнити досить багато як системної пам’яті, котру використовує програмне забезпечення, так і об’єм самої бази даних. Проведена оптимізація дала змогу зробити таблиці більш інформативні та доступні для будь-якого користувача.

До переваг програми потрібно віднести можливості швидкого та зручного введення нової інформації, отримання потрібних табличних результатів.

Програмний комплекс «Uvelir_3D», розроблений на мові C++ з використанням відкритої графічної бібліотеки OpenGL, дає можливість наочного подання інформації, у вигляді гістограми з використанням ефектів світла та накладення текстур. За допомогою цих засобів надається можливість наочно аналізувати інформацію про зміни цін на жіночі прикраси тих чи інших категорій.

Застосування універсальних Java-скриптів на основі елементу керування Chart.js дають можливість гнучко передавати дані з бази до створюваного html-файлу та користуватися ним в будь-яких браузерах. Це дозволило зробити гістограми в html для цін жіночих прикрас різних категорій.

На основі проведеної роботи, можна з успіхом розвивати та вдосконалювати розроблену програмну систему та надавати їй нової функціональності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Агуров, Павел C#. Сборник рецептов / Павел Агуров. - М.: "БХВ-Петербург", 2012. - 432 с.

2. Албахари, Джозеф C# 3.0. Справочник / Джозеф Албахари, Бен Албахари. - М.: БХВ-Петербург, 2012. - 944 с.

3. Албахари, Джозеф C# 3.0. Справочник / Джозеф Албахари, Бен Албахари. - М.: БХВ-Петербург, 2013. - 944 с.

4. Альфред, В. Ахо Компиляторы. Принципы, технологии и инструментарий / Альфред В. Ахо и др. - М.: Вильямс, 2015. - 266 с.

5. Бишоп, Дж. C# в кратком изложении / Дж. Бишоп, Н. Хорспул. - М.: Бином. Лаборатория знаний, 2013. - 472 с.

6. Вагнер, Билл C# Эффективное программирование / Билл Вагнер. - М.: ЛОРИ, 2013. - 320 с.

7. Зиборов, В.В. Visual C# 2012 на примерах / В.В. Зиборов. - М.: БХВ-Петербург, 2013. - 480 с.

8. Зиборов, Виктор Visual C# 2010 на примерах / Виктор Зиборов. - М.: "БХВ-Петербург", 2011. - 432 с.

9. Ишкова, Э. А. Самоучитель C#. Начала программирования / Э.А. Ишкова. - М.: Наука и техника, 2013. - 496 с.

10. Касаткин, А. И. Профессиональное программирование на языке си. Управление ресурсами / А.И. Касаткин. - М.: Высшая школа, 2012. - 432 с.

11. Лотка, Рокфорд C# и CSLA .NET Framework. Разработка бизнес-объектов / Рокфорд Лотка. - М.: Вильямс, 2010. - 816 с.

12. Мак-Дональд, Мэтью Silverlight 5 с примерами на C# для профессионалов / Мэтью Мак-Дональд. - М.: Вильямс, 2013. - 848 с.

13. Марченко, А. Л. Основы программирования на C# 2.0 / А.Л. Марченко. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2011. - 552 с.

14. Подбельский, В. В. Язык C#. Базовый курс / В.В. Подбельский. - М.: Финансы и статистика, Инфра-М, 2011. - 384 с.

15. Прайс, Джейсон Visual C# 2.0. Полное руководство / Джейсон Прайс, Майк Гандэрлой. - М.: Век +, Корона-Век, Энтроп, 2010. - 736 с.

16. Рихтер, Джеффри CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C# / Джеффри Рихтер. - М.: Питер, 2013. - 928 с.

17. Смоленцев, Н. К. MATLAB. Программирование на Visual C#, Borland JBuilder, VBA (+ CD-ROM) / Н.К. Смоленцев. - М.: ДМК Пресс, 2011. - 456 с.

18. Троелсен, Эндрю Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. - 486 с.

19. Троелсен, Эндрю Язык программирования C# 2008 и платформа .NET 3.5 / Эндрю Троелсен. - М.: Вильямс, 2010. - 370 с.

20. Фримен, Адам ASP.NET MVC 3 Framework с примерами на C# для профессионалов / Адам Фримен , Стивен Сандерсон. - М.: Вильямс, 2011. - 672 с.

21. Дж.Рихтер. CLR via C# . Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд./Дж.Рихтер – Питер, 2013. - 896с.

22. Дж.Рихтер. CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#, 2-е изд/ Дж.Рихтер - Питер, 2007. - 656 с.

23. Зеленков Ю. Введение в базы данных / Ю. Зеленков – СПб.: Питер, 2003

24. Кузнецов С.Д. Основы современных баз данных. 2-е издание испр. / С.Д.Кузнецов,– М.:Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний– 2007. – 484с.

ДОДАТКИ

Додаток А

Лістинг коду головної форми програми Form1.cs

```
namespace Uvelir
{
    public partial class Form1 : Form
    {
        //Объект Connection с драйвером ODBC
        OdbcConnection conn;

        public static Color cvet_hapka;
        public static Color cvet_text;
        public static Font font_text;

        int kod_zap = 0;

        public Form1()
        {
            InitializeComponent();

            //Инициализация и открытие БД MySQL
            //Формирование объекта Connection
            try
            {
                string source = "DRIVER=MySQL ODBC 3.51 Driver;UID =
root;SERVER=localhost";
                conn = new OdbcConnection(source);


                conn.Open();

                OdbcCommand cmd = new OdbcCommand("USE Shop1", conn);
                cmd.ExecuteNonQuery();

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                return;
            }

            //Считывание настроек
            try
            {
                BinaryReader bb = new BinaryReader(new FileStream("nastr.ini",
FileMode.Open));
                cvet_hapka = Color.FromArgb(bb.ReadByte(), bb.ReadByte(), bb.ReadByte());
                cvet_text = Color.FromArgb(bb.ReadByte(), bb.ReadByte(), bb.ReadByte());
                font_text = new Font(bb.ReadString(), bb.ReadSingle(),
(FontStyle)bb.ReadInt32());
                bb.Close();
                return;
            }
            catch
            {
                MessageBox.Show("Файл nastr.ini не найден!\nБудут приняты значения по
умолчанию");
            }

            cvet_hapka = Color.Blue;
            cvet_text = Color.Black;
            font_text = new Font("Times New Roman", 14, FontStyle.Regular);
            kod_zap = 1;
        }
```

```
        //Универсальная функция экспорта в Excel произвольных данных
        public static bool ExportXLS(DataView dt)
        {
            try
            {
                Excel.Application ExcelApp = new Excel.Application();

                Excel.Workbook book = ExcelApp.Workbooks.Add(Type.Missing);
                ExcelApp.SheetsInNewWorkbook = 1;
                Excel.Worksheet sheet = (Excel.Worksheet)ExcelApp.Sheets.get_Item(1);
                sheet.Name = "Мой лист";

                Excel.Range range, range1;

                range = sheet.Cells;

                string str = (char)('A' + dt.Table.Columns.Count - 1) + "";

                //A1 - D1
                range1 = range.get_Range("A1", ((char)('A' + dt.Table.Columns.Count - 1))
+ "1");

                range1.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;
                range1.Font.Bold = true;

                int row = 1, col;
                string v;

                Excel.Border border;
                border = range1.Borders.get_Item(Excel.XlBordersIndex.xlEdgeTop);

                //Рисование шапки
                for (col = 0; col < dt.Table.Columns.Count; col++)
                {
                    v = dt.Table.Columns[col].ColumnName;
                    if (v == "") continue;
                    range.set_Item(row, col + 1, v);
                }


                row++;

                foreach (DataRowView rr in dt)
                {
                    for (col = 0; col < dt.Table.Columns.Count; col++)
                    {
                        v = rr[col].ToString();
                        if (v == "") continue;
                        double dd;

                        if (dt.Table.Columns[col].DataType.ToString() ==
"System.DateTime")
                            v = Convert.ToDateTime(rr[col]).ToShortDateString();

                        if (double.TryParse(v, out dd) == true)
                            range.set_Item(row, col + 1, dd);
                        else
                            range.set_Item(row, col + 1, v);
                    }

                    row++;
                }
```

```
                range1 = range.get_Range("A1", ((char)('A' + dt.Table.Columns.Count - 1))
+ (dt.Count + 1).ToString());
                range1.VerticalAlignment = Excel.XlVAlign.xlVAlignCenter;
                range1.Borders.get_Item(Excel.XlBordersIndex.xlEdgeBottom).LineStyle = 1;
                range1.Borders.get_Item(Excel.XlBordersIndex.xlEdgeLeft).LineStyle = 1;
                range1.Borders.get_Item(Excel.XlBordersIndex.xlEdgeRight).LineStyle = 1;
                range1.Borders.get_Item(Excel.XlBordersIndex.xlEdgeTop).LineStyle = 1;

range1.Borders.get_Item(Excel.XlBordersIndex.xlInsideHorizontal).LineStyle = 1;
                range1.Borders.get_Item(Excel.XlBordersIndex.xlInsideVertical).LineStyle
= 1;

                range1.Font.Name = Form1.font_text.Name;
                range1.Font.Size = Form1.font_text.SizeInPoints;
                range1.Font.Color = Form1.cvet_text;

                range1 = range.get_Range("A1", ((char)('A' + dt.Table.Columns.Count - 1))
+ "1");
                range1.Font.Color = Form1.cvet_hapka;

                sheet.Columns.AutoFit();
                ExcelApp.Visible = true;

            }

            catch (Exception)
            {
                return false;
            }

            return true;
        }


        //Отклик на продукцию...
        private void продукцияToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form2 ff = new Form2();
            ff.conn = conn;
            ff.ShowDialog();
        }

        //Отклик на клиенты
        private void клиентыToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form3 ff = new Form3();
            ff.conn = conn;
            ff.ShowDialog();
        }

        //Оформление покупки
        private void оформитьПокупкуToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form4 ff = new Form4();
            ff.conn = conn;
            ff.ShowDialog();
        }


        //Выбор шрифта
        private void выборШрифтаToolStripMenuItem_Click(object sender, EventArgs e)
        {
            fontDialog1.Font = font_text;
```

```
            if (fontDialog1.ShowDialog() == DialogResult.OK)
            {
                font_text = fontDialog1.Font;
                kod_zap = 1;
            }
        }

        //Цвет шапки таблицы
        private void цветШапкиТаблицыToolStripMenuItem_Click(object sender, EventArgs e)
        {
            colorDialog1.Color = cvet_hapka;
            if (colorDialog1.ShowDialog() == DialogResult.OK)
            {
                cvet_hapka = colorDialog1.Color;
                kod_zap = 1;
            }
        }

        //Цвет текста таблицы
        private void цветТекстаТаблицыToolStripMenuItem_Click(object sender, EventArgs e)
        {
            colorDialog1.Color = cvet_text;
            if (colorDialog1.ShowDialog() == DialogResult.OK)
            {
                cvet_text = colorDialog1.Color;
                kod_zap = 1;
            }
        }

        //Закрытие формы
        protected override void OnClosing(CancelEventArgs e)
        {

            if (kod_zap == 1)
            {
                DialogResult dr = MessageBox.Show("Настройки были изменены!\nСохранить
изменения на диск?", "Сохранение",
                MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);
                if (dr == DialogResult.Yes)
                {
                    BinaryWriter bb = new BinaryWriter(new FileStream("nastr.ini",
FileMode.OpenOrCreate));

                    bb.Write(cvet_hapka.R);
                    bb.Write(cvet_hapka.G);
                    bb.Write(cvet_hapka.B);
                    bb.Write(cvet_text.R);
                    bb.Write(cvet_text.G);
                    bb.Write(cvet_text.B);
                    bb.Write(font_text.Name);
                    bb.Write(font_text.SizeInPoints);
                    bb.Write((int)font_text.Style);
                    bb.Close();
                    return;
                }
                else if (dr == DialogResult.Cancel)
                    e.Cancel = true;
            }
            base.OnClosing(e);
        }

    }
}
```

Додаток Б

Лістинг коду форми Form2.cs

```
namespace Uvelir
{
    public partial class Form2 : Form
    {

        public OdbcConnection conn;
        OdbcDataAdapter da;
        DataSet ds;


        public Form2()
        {
            InitializeComponent();
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            try
            {
                //Заполнение списка через DataReader
                OdbcCommand cmd = new OdbcCommand("Select * From Categories", conn);
                OdbcDataReader rd = cmd.ExecuteReader();
                while (rd.Read())
                    comboBox1.Items.Add(new
Name_List(rd[1].ToString(),Convert.ToInt32(rd[0])));

                comboBox1.SelectedIndex = 0;

            }

            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        //Изменение категории продукции
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            try
            {
                string str;
                str = "Select * From Products where ID_cat=" +
((Name_List)comboBox1.SelectedItem).ItemData;

                if (da != null) da.Dispose();
                da = new OdbcDataAdapter(str, conn);

                OdbcCommandBuilder bulder = new OdbcCommandBuilder(da);

                if (ds != null) ds.Dispose();
                ds = new DataSet();
                da.Fill(ds);


                //Привязка dataGridView1 к объекту DataSet
                bindingSource1.DataSource = ds.Tables[0];
                bindingNavigator1.BindingSource = bindingSource1;
                dataGridView1.DataSource = bindingSource1;
                ////////////////////////////////////////////
                dataGridView1.Columns[0].Visible = false;
                dataGridView1.Columns[1].Visible = false;
```

Продовження додатку Б

```
                ////////////////////////////////////////////
                bindingSource1.Position = 0;

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        //Изменение текущей записи
        private void bindingSource1_CurrentChanged(object sender, EventArgs e)
        {
            //Загрузка картинки по маршруту
            DataView dt = ds.Tables[0].DefaultView;
            string str = "Image\\" + comboBox1.SelectedItem.ToString() + "\\" +
dt[bindingSource1.Position][3];
            pictureBox1.Image = Image.FromFile(str);
        }

        //Обновление данных
        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                da.Update(ds.Tables[0]);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        //Экспорт в XLS
        private void button2_Click(object sender, EventArgs e)
        {
            if (!Form1.ExportXLS(ds.Tables[0].DefaultView))
            {
                MessageBox.Show("Невозможен экспорт в Excel!");
            }
        }
    }


    //Класс для заполнения списка
    public class Name_List
    {
        public string str;
        public int ItemData;
        public Name_List(string str, int ItemData)
        {
            this.str = str;
            this.ItemData = ItemData;
        }
        public override string ToString()
        {
            return str;
        }
    }
}
```

Додаток В

Лістинг коду програми Uvelir_3D.

```
//Функция инициализации текстур
bool GR_3D::Init_Textur()
{


if(_access("BMP",0))
{
 MessageBox(L"Отсутствует папка BMP,\nгде должны находится файлы *.bmp\nдля организации
текстур \n",L"Внимание!",MB_ICONINFORMATION);
 return 0;
}

CString str;

if(_access("BMP\\textur.bmp",0))str=L"textur.bmp\n";
if(_access("BMP\\textur2.bmp",0))str+=L"textur2.bmp\n";

if(str.GetLength ())
{
 str = L"В папке BMP отсутствуют следующие файлы:\n" + str;
 AfxMessageBox(str,MB_ICONINFORMATION);
 return 0;
}


//Генерирование текстур (дисплейные списки текстур)
glEnable(GL_TEXTURE_2D);

glGenTextures(1, &text1);
glGenTextures(1, &text2);

CUvelir_3DDoc* pDoc = (CUvelir_3DDoc*)((CFrameWnd*)AfxGetMainWnd())->GetActiveDocument();
glGenTextures(pDoc->n, txt);

int w,h;
unsigned char*data1;
if(!LoadTextur(L"BMP\\textur.bmp", &w,&h,data1))
return false;

glBindTexture(GL_TEXTURE_2D, text1);

gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGBA,
                  w,
                  h,
                  GL_RGB, GL_UNSIGNED_BYTE,
                  data1);
delete []data1;

if(!LoadTextur(L"BMP\\textur2.bmp", &w,&h,data1))
return false;

glBindTexture(GL_TEXTURE_2D, text2);

gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGBA,
                  w,
                  h,
                  GL_RGB, GL_UNSIGNED_BYTE,
                  data1);
delete []data1;
```

```
for(int i=0;i<pDoc->n;i++)
{
	if(!LoadTextur(L"Image\\"+pDoc->name[i], &w,&h,data1))
	return false;
	glBindTexture(GL_TEXTURE_2D, txt[i]);

	gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGBA,
					w,
					h,
					GL_RGB, GL_UNSIGNED_BYTE,
					data1);
	delete []data1;

}


glDisable(GL_TEXTURE_2D);

return 1;
}

// Функция для работы с текстурами формата bmp (универсальная - для любого количества цветов)
// возврат: true - текстура загружена, false - несоответствие формата или файл не найден
// w - ширина текстуры, h - высота, dd - выходной массив RGB
//-----------------------------------------------------------------------//
bool GR_3D::LoadTextur(CString str, int*w,int*h,unsigned char*&dd)
{
CBitmap cc;
BITMAP bt;
BITMAPINFO bb;
int i,kol_zap;
unsigned char temp;

CImage* im = new CImage;
im->Load(str);
if((*im)==0)return false;
cc.Attach(*im);

cc.GetBitmap(&bt);

int x,y;

x = bt.bmWidth;
y = bt.bmHeight;

//x = y = 128;

bb.bmiHeader.biSize = sizeof(BITMAPINFOHEADER);
bb.bmiHeader.biBitCount = 24;
bb.bmiHeader.biWidth = x;
bb.bmiHeader.biHeight = y;
bb.bmiHeader.biPlanes = 1;
bb.bmiHeader.biCompression = BI_RGB;
kol_zap = (x+1)*(y+1);
unsigned char *data = new unsigned char[3*kol_zap+5];
GetDIBits(GetDC()->m_hDC,*im,0,y,data,&bb,DIB_RGB_COLORS);
for(i=0;i<kol_zap;i++)
{
 temp = data[3*i];
 data[3*i] = data[3*i+2];
 data[3*i+2] = temp;
}
```

```
*w = x;
*h = y;

dd = data;
DeleteObject(*im);
return true;

}
//-----------------------------------------------------------------------//

GR_3D::~GR_3D()
{
    //Удаление, которое касается OpenGl
   if(text_nal)
    glDeleteTextures(1,&text1);

    HGLRC hrc = ::wglGetCurrentContext();
    ::wglMakeCurrent(NULL,  NULL);

    if (hrc)
      ::wglDeleteContext(hrc);

       ((CMainFrame*)AfxGetMainWnd())->graf = 0;
}


BEGIN_MESSAGE_MAP(GR_3D, CFrameWnd)
       ON_WM_CREATE()
       ON_WM_PAINT()
       ON_WM_SIZE()
       ON_WM_KEYDOWN()
       ON_WM_RBUTTONUP()
END_MESSAGE_MAP()


// GR_3D message handlers

//Функция инициализации 3D-графики
bool GR_3D::Init_3d()
{
       HGLRC        hrc;  // дескриптор OPENGL
    int pixelformat;

       CMainFrame* pp = (CMainFrame*)AfxGetMainWnd();

  if ((pixelformat = ChoosePixelFormat(GetDC()->GetSafeHdc(), &(pp->pfd))) == 0 ||
        SetPixelFormat(GetDC()->GetSafeHdc(), pixelformat, &pp->pfd) == FALSE)
                    return false;

       ::DescribePixelFormat(GetDC()->GetSafeHdc(), pixelformat, sizeof(pp->pfd), &(pp-
>pfd));
    if (pp->pfd.cColorBits<16)return false;

       hrc = wglCreateContext(GetDC()->GetSafeHdc());// создание контекста
    wglMakeCurrent(GetDC()->GetSafeHdc(), hrc);   // установка контекста

       // Формирование списка шрифтов в OpenGL
          CFont font;
          font.CreatePointFont(50,_T("Arial"));

       SelectObject(wglGetCurrentDC(),font.m_hObject);
       wglUseFontOutlinesA(wglGetCurrentDC(), 0, 255,
       GLF_START_LIST, 0,0.15, WGL_FONT_POLYGONS,0);
```

```
	glEnable(GL_DEPTH_TEST); // активизация буфера глубины
    return true;
}


int GR_3D::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
		return -1;

	if(!Init_3d())
	 {
	   MessageBox(L"Ошибка инициализации графики!!!",L"Внимание!",MB_ICONINFORMATION);
	   return -1;
	 }

	text_nal = Init_Textur();
	if(!text_nal)
		MessageBox(L"Отсутствует один из файлов текстуры\nили нарушена структура
файлов!\nПри работе с 3D-графикой используются только цветовые
настройки!",L"Внимание!!!",MB_ICONSTOP);

	SetWindowText(L"Ювелирные изделия");

	return 0;
}

//Вывод текста
void GR_3D::OutText(char *text)
{
	glListBase(GLF_START_LIST);
	glCallLists(strlen(text), GL_UNSIGNED_BYTE, text);
}


void GR_3D::OnPaint()
{
	CPaintDC dc(this); // device context for painting

	//glClearColor(0.25, 0.25, 0.25, 1);
	glClearColor(0, 0.5, 1, 1);
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);// буфер цвета и глубины

	glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

	glPushMatrix();//сохранение текущей матрицы
	glPushAttrib(GL_ALL_ATTRIB_BITS);//сохранение текущих настроек

	Graf();

	glPopMatrix();//восстановление матрицы
    glPopAttrib();//восстановление настроек цвета

    glFinish();

	SwapBuffers(wglGetCurrentDC()); // Вывод в контекст
}

void GR_3D::OnSize(UINT nType, int cx, int cy)
{
	CFrameWnd::OnSize(nType, cx, cy);

	glViewport(0,0,cx,cy);
```

```
       //Установка перспективы в матрице проекции

       glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

       gluPerspective(35.0f, 1.0*cx/cy, 1.0f,   50.0f);

       //Загркзка матрицы модели
       glMatrixMode(GL_MODELVIEW);
       glLoadIdentity();

       //Установка световых параметров
       Ust_lighting();
       Invalidate();
}


//Функция рисования призмы относительно y
void GR_3D::truba_y(double z1, double x1, double y1, double y2, double prir,
                                  long c1,long c2)
{
   // построение призмы вокруг прямой
   double z_prir[8] = {1, -1,-1,-1, -1, 1,1,1};
   double y_prir[8] = {1, 1, 1, -1, -1,-1,-1,1};
   double x,z;

   int r1,g1,b1;
   int r2,g2,b2;

       r1 = (c1 & 0x000000FF);
       g1 = (c1 & 0x0000FF00) >> 8;
       b1 = (c1 & 0x00FF0000) >> 16;

       r2 = (c2 & 0x000000FF);
       g2 = (c2 & 0x0000FF00) >> 8;
       b2 = (c2 & 0x00FF0000) >> 16;

        for(int j = 0; j < 8; j++)
                {

                   z = z1 + prir*z_prir[j];
                   x = x1 + prir*y_prir[j];
                   if ((j%2))
                   {
                       glColor3ub(r1,g1,b1);
                       glTexCoord2f(0,0);
                                    glVertex3d(x,y1,z);
                                    glColor3ub(r2,g2,b2);
                       glTexCoord2f(0,5);
                                    glVertex3d(x,y2,z);
                       continue;
                   }
                       glColor3ub(r1,g1,b1);
                                    glTexCoord2f(2,5);
                       glVertex3d(x,y2,z);
                                    glColor3ub(r2,g2,b2);
                                    glTexCoord2f(2,0);
                       glVertex3d(x,y1,z);
                }
}
```

Продовження додатку В

```
//Функция рисования сцены с графикой
void GR_3D::Graf()
       {
              gltApplyCameraTransform(&frameCamera);

              glColor3f(0, 1, 0);
              DrawGround();

        glDisable(GL_DEPTH_TEST);
        glDisable(GL_LIGHTING);
        glPushMatrix();
            glMultMatrixf(mShadowMatrix);
            DrawInhabitants(1);
        glPopMatrix();
        glEnable(GL_LIGHTING);
        glEnable(GL_DEPTH_TEST);
                    DrawInhabitants(0);
       }


//Установка световых параметров
void GR_3D::Ust_lighting()
 {
              GLfloat fLightPos[4]   = { -100.0f, 100.0f, 50.0f, 1.0f };  // Point source
              GLfloat fNoLight[] = { 0.0f, 0.0f, 0.0f, 0.0f };
              GLfloat fLowLight[] = { 0.25f, 0.25f, 0.25f, 1.0f };
              GLfloat fBrightLight[] = { 1.0f, 1.0f, 1.0f, 1.0f };

              GLTVector3 vPoints[3] = {{ 0.0f, -0.4f, 0.0f },
                                                    { 10.0f, -0.4f, 0.0f },
                                                    { 5, -0.4f, -5.0f }};

              glLightModelfv(GL_LIGHT_MODEL_AMBIENT, fNoLight);
              glLightfv(GL_LIGHT0, GL_AMBIENT,  fLowLight);
              glLightfv(GL_LIGHT0, GL_DIFFUSE,  fBrightLight);
              glLightfv(GL_LIGHT0, GL_SPECULAR, fBrightLight);


        glLightfv(GL_LIGHT0, GL_POSITION, fLightPos);

              gltMakeShadowMatrix(vPoints, fLightPos, mShadowMatrix);
              glEnable(GL_COLOR_MATERIAL);
              glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
              glMateriali(GL_FRONT, GL_SHININESS, 128);

              gltInitFrame(&frameCamera);  // Initialize the camera

              CUvelir_3DDoc* pDoc = (CUvelir_3DDoc*)((CFrameWnd*)AfxGetMainWnd())-
>GetActiveDocument();

              int n = pDoc->n;

              for(int iprizma = 0; iprizma < n; iprizma++)
              {
                     gltInitFrame(&prizma[iprizma]);    // Initialize the frame
              prizma[iprizma].vLocation[0] = (float)(-(n-1)/2. + iprizma);
                     prizma[iprizma].vLocation[1] = 0.0f;
                     prizma[iprizma].vLocation[2] = -9.0f;
               }
              glEnable(GL_LIGHTING);
                 glEnable(GL_LIGHT0);

 }
```

```
//Функция рисования сцены (земли)
void GR_3D::DrawGround()
{
    GLfloat fExtent = 20.0f;
    GLfloat fStep = 1.0f;
    GLfloat y = -0.4f;

   glEnable(GL_TEXTURE_2D);
   glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,GL_DECAL);
   glBindTexture(GL_TEXTURE_2D, text2);

    glNormal3f(0.0f, 1.0f, 0.0f);
       glBegin(GL_QUADS);
        glTexCoord2f(0,0);
              glVertex3f(-fExtent, y, -fExtent);
        glTexCoord2f(0,50);
              glVertex3f(-fExtent, y, fExtent);
        glTexCoord2f(50,50);
              glVertex3f(fExtent, y, fExtent);
        glTexCoord2f(50,0);
              glVertex3f(fExtent, y, -fExtent);
       glEnd();
   glDisable(GL_TEXTURE_2D);

}


//Функция вывода диаграммы и тени к ней
void GR_3D::DrawInhabitants(GLint nShadow)
    {
 // включение режима работы с текстурами


       if(text_nal&&nShadow==0&&kod_text!=3)
   {
       glEnable(GL_TEXTURE_2D);
   // Вывод текстуры с учетом цвета объекта
   if(kod_text==1)glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,GL_MODULATE);
   // Вывод текстуры без учета цвета объекта
   if(kod_text==2)glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,GL_DECAL);
   glBindTexture(GL_TEXTURE_2D, text1);
   }


       //Указатель на документ
       CUvelir_3DDoc* pDoc = (CUvelir_3DDoc*)((CFrameWnd*)AfxGetMainWnd())-
>GetActiveDocument();

       long c1,c2;
       double y,max,min;
       int i,n = pDoc->n;
       double* mas = pDoc->mas;
       char str[20];

       //Находи максимум массива
       min = mas[0];
       max = mas[0];
       for(i=1;i<n;i++)
       {
              if (mas[i]<min)min = mas[i];
              if (mas[i]>max)max = mas[i];
       }
```

Продовження додатку В

```
    for(i = 0; i < n; i++)
        {
                    glPushMatrix();
                    gltApplyActorTransform(&prizma[i]);

                    if(nShadow == 0)
                    {
                           c1 = colorArray[i%12+2];
                           c2 = colorArray[1];
                    }

                    else c1 = c2 = 0;

                    glBegin(GL_QUADS);
                           y = (mas[i]-min)/(max-min)*2;
                           truba_y(0, 0, -0.4, y, 0.15,c1,c2);
                    glEnd();

                    //Вывод текста в 3D

                    if(nShadow == 0)
                    {
                           glEnable(GL_TEXTURE_2D);
                           glBindTexture(GL_TEXTURE_2D, txt[i]);

                           glColor3f(1,0,0);
                            glBegin(GL_QUADS);
                                  glTexCoord2d(0,0);
                                  glVertex3d(-0.4,y+0.3,0);
                                  glTexCoord2d(0,1);
                                  glVertex3d(-0.4,y+1.1,0);
                                  glTexCoord2d(1,1);
                                  glVertex3d(0.4,y+1.1,0);
                                  glTexCoord2d(1,0);
                                  glVertex3d(0.4,y+0.3,0);
                            glEnd();

                            glDisable(GL_TEXTURE_2D);

                            glPushAttrib(GL_ALL_ATTRIB_BITS);//сохранение текущих
настроек
                                   glTranslatef(0.15,y+0.1,0);
                                   glRotatef(180,0,1,0);
                                   glScalef(0.15f,0.15f,0.15f);
                                   sprintf_s(str,"%0.2f",mas[i]);
                                   glColor3f(1,1,1);
                                   glDisable(GL_TEXTURE_2D);
                                   OutText(str);
                                   glEnable(GL_TEXTURE_2D);
                            glPopAttrib();


                    }


                    glPopMatrix();
        }

   glDisable(GL_TEXTURE_2D);

}
```