РЕФЕРАТ

Звіт про курсову роботу: 36 с., 25 рис., 4 додатки, 9 джерел.

Об’єкт дослідження – технології Visual Studio зі створення багатомодульних проектів та технології опрацювання даних, які зберігаються у файлах.

Мета роботи – вивчення засобів Visual C++ для створення програм, що працюють з файлами даних, включаючи їхнє програмне редагування (додавання нової інформації, зміну наявних записів або їх видалення), а також формування практичних умінь розробляти програмні застосунки з графічним інтерфейсом. Такі програми можуть містити меню, діалогові вікна, кілька пов’язаних між собою форм та інші елементи взаємодії.

Предмет дослідження – програмні засоби Visual C++ для створення файлів різних форматів, записування у них інформації про продуктові магазини, виведення вмісту цих файлів, вибір даних за умовою, можливості редагування файлу і видалення певних даних з файлу, створення нових файлів (звітних документів) з певними даними з основного файлу, засоби для створення програмних проектів з декількома динамічними формами та взаємодії між ними.

Отримані результати – розроблено та протестовано програмний проект опрацювання інформації про продуктові магазини, що зберігається у файлах.

БІНАРНИЙ ФАЙЛ, ТЕКСТОВИЙ ФАЙЛ, СОРТУВАННЯ, ВІДБІР ДАНИХ, НАЙМЕНУВАННЯ ГРУПИ ТОВАРІВ, НАЗВА ТОВАРУ, ОДИНИЦЯ ВИМІРУ, ЦІНА ОДИНИЦІ, ДАТА ОСТАННЬОГО ЗАВОЗУ, ВІДМІТКА ПРО УЦІНКУ

ЗМІСТ

ВСТУП

У сучасних програмних системах значну роль відіграють застосунки з графічним інтерфейсом, які забезпечують зручну взаємодію користувача з програмою та її функціями. Windows Forms є платформою для створення таких програм і надає можливість розробляти як прості, так і більш комплексні застосунки, зокрема й веборієнтовані. У цій роботі продемонстровано використання Visual C++ для побудови програм, що складаються з кількох модулів і виконують роботу з файлами. У процесі виконання практичних завдань часто постає необхідність отримувати й обробляти дані, що зберігаються на носіях, тому застосунок має забезпечувати відкриття файлів, зчитування інформації, її змінення та створення нових файлів за потреби.

Під час роботи з даними часто обирають двійковий формат, оскільки він забезпечує компактне зберігання інформації та не містить зайвих символів, що дозволяє економити місце і пришвидшувати обробку. Водночас для підсумкових результатів зазвичай створюють текстові файли, адже їх легко відкривати, читати та редагувати в більшості програм.

Програма, яка опрацьовує дані про продуктові магазини. У середовищі Visual Studio за допомогою Windows Forms можна створювати інтерфейс для введення та відображення даних, зокрема, дані про продуктові магазини, в якій користувач зможе додавати та переглядати інформацію про найменування групи товарів, назва товару, одиниця виміру, ціна одиниці, дата останнього завозу, відмітка про уцінку, сортувати товари однієї групи в алфавітному порядку, фільтрувати товари з молочного відділу, відображаючи лише ті товари, які з молочного відділу, видаляти товари які уцінили, зберігати дані у бінарному файлі та формувати текстовий файл, відповідно до заданої умови, а також редагувати вже створений бінарний файл.

1 ПОСТАНОВКА ЗАВДАННЯ

Розробити програмний проєкт у Visual C++, призначений для створення й опрацювання бінарних і текстових файлів, що містять інформацію про товари в продуктовому магазині: найменування групи товарів, назва товару, одиниця виміру, ціна одиниці, дата останнього завозу та відмітка про уцінку.

При створенні програмного проекту передбачити виконання таких завдань:

– програмне створення бінарного файлу і заповнення його наступними даними: найменування групи товарів, назва товару, одиниця виміру, ціна за одиницю, дата останнього завозу, наявність уцінки

– програмний перегляд даних створеного бінарного файлу;

– можливість редагування та зберігання відредагованого файлу;

– впорядкування товари однієї групи в алфавітному порядку;

– відбір даних з файлу з виведенням результатів на екран : товари з молочного відділу;

– відбір даних зі створенням текстового файлу за умовою: товари завезені за вказаною датою;

– можливість програмного відбору та вилучення даних за умовою: товари які уцінили;

– створення додаткової форми, яка містить інформацію про автора;

– створення додаткової форми, що містить текст завдання на курсову роботу;

– створення форми заставки.

– додатково реалізувати у програмі функцію яка буде сортувати товари за ціною, від найменшого до набільшого.

2 ІНСТРУМЕНТАРІЙ ДЛЯ РОЗВ’ЯЗУВАННЯ ЗАДАЧ

2.1 Інформація про файли

Файл – це впорядкована сукупність даних, розміщена на носії інформації, наприклад жорсткому диску або флеш-пам’яті, і записана під окремою назвою. У такому об’єкті можуть міститися різні типи інформації: текстові матеріали, зображення, аудіо, відео, програмні компоненти та інші дані. Завдяки унікальному імені файл легко відшукати й відкрити для подальшого використання. Файли поділяються на:

1. Текстові файли – це файли, у яких зберігаються символи, сформовані у рядки, тобто інформація, що легко читається людиною.
2. Бінарні файли – це файли які складаються з послідовностей байтів і використовуються для зберігання даних, вони не мають внутрішнього поділу на символи чи рядки. У бінарних файлах дані записуються в такому вигляді, в якому вони зберігаються у пам'яті комп’ютера, тому їхній вміст не завжди можна переглянути у зрозумілому вигляді за допомогою звичайних текстових редакторів [1].

Файл окрема одиниця пам'яті, тому видалення одного файлу не впливає на інші. Користувач може створювати файли самостійно або працювати з уже створеними файлами.

Файли можна створювати різними способами: вручну через контекстне меню, за допомогою середовища розробки або через командний рядок. У межах цієї курсової роботи всі операції з файлами виконуються у Visual Studio на мові C++. Для роботи з файловою системою використовується простір імен `System::IO`, яке підключається за допомогою директиви `using namespace System::IO` у верхній частині коду. Це простір імен містить набір інструментів для створення, модифікації та зчитування файлів і каталогів [2].

Клас `File` надає методи для базових операцій із файлами: створення нових файлів, видалення, копіювання, переміщення, а також відкриття для

читання чи запису. Це дозволяє працювати як із текстовим, так і з бінарним вмістом. Клас `Directory` відповідає за роботу з папками: створення, перенесення, отримання списку вкладених файлів і каталогів. `Path` містить методи для обробки шляхів: виділення каталогу, назви файлу, розширення або складання повного шляху з кількох частин.

Для опрацювання текстових даних використовуються `StreamReader` і `StreamWriter`. Перший дозволяє читати текст по символах або рядках, другий – записувати текстову інформацію у файли, що зручно для роботи зі звичайними текстовими структурами даних. Для опрацювання бінарних файлів застосовуються `BinaryReader` і `BinaryWriter`. Вони забезпечують швидку та компактну роботу з числовими типами, логічними значеннями та рядками у двійковому форматі, без зайвих службових символів. Основою для більшості потокових операцій є `FileStream`, який створює канал між програмою та файлом і дозволяє виконувати низькорівневі операції читання та запису [3].

Текстові файли являють собою послідовність символів, які програмне забезпечення інтерпретує відповідно до обраної кодування. Бінарні файли – це набір байтів, що точно відтворюють внутрішнє представлення даних, тому вони зберігаються та обробляються швидше й компактніше, ніж текстові аналоги.

2.2 Опрацювання файлів у C++

Для опрацювання файлів у C++ широко застосовуються класи `FileStream`, `StreamReader`, `StreamWriter`, `BinaryReader` та `BinaryWriter`. Ці інструменти забезпечують можливість працювати як із текстовими файлами, так і з бінарними структурами даних, дозволяючи відкривати файли для читання, дописування, повного перезапису або вибіркового редагування [6]. Приклад такої роботи наведено у лістингу 2.1.

Лістинг 2.1 – Код видалення файлу перед створенням нового

```
FileStream^ file = gcnew FileStream(b_name, FileMode::Delete);
BinaryWriter^ fb = gcnew BinaryWriter(file);
```

У цьому прикладі застосовано режим `FileMode::Delete`, який видаляє файл із диска. Завдяки такому підходу можна гарантувати, що під час подальшого запису даних програма працюватиме з новим, чистим файлом. Після видалення файлу інструмент BinaryWriter використовується для створення нового бінарного файлу і запису в нього оновленої інформації.

Метод `FileMode` має такі режими доступу, як:

- `Create` – створює новий файл, а якщо він вже створений, то перезаписує його;
- `CreateNew` – створює новий файл. Якщо файл із зазначеним іменем уже існує, буде згенеровано виняток
- `Open` – відкриває файл, якщо він створений;
- `OpenOrCreate` – відкриває або створює новий файл;
- `CreateNow` – створює файл лише в тому випадку, якщо він ще не існує;

Клас `BinaryWriter` використовується для запису даних у файл у двійковому форматі. Такий підхід корисний, коли потрібно записувати дані в ефективному та компактному вигляді, перетворюючи їх у байти для зберігання у файлі. Це особливо підходить для збереження великих обсягів інформації або спеціалізованих даних.

При роботі з файлами, особливо при записуванні текстових даних, важливо враховувати вибір правильного кодування для збереження символів. Це дозволяє забезпечити коректне відображення і читання тексту при його зчитуванні з файлу. За замовчуванням клас `BinaryWriter` використовує кодування UTF-8 для збереження текстових даних, однак у разі потреби можна налаштувати інше кодування, що підходить для специфічних вимог або різних мовних стандартів [7].

3 СТВОРЕННЯ ОСНОВНОГО МОДУЛЯ

3.1 Організація бінарного файлу

Для створення проекту опрацювання інформації про товари в магазині використовують компоненту `tabControl1`. Додати її можна скориставшись командою *Додати вкладку*, яку вибирають з контекстного меню або натиснувши на стрілочку у верхньому правому кутку елемента.

На першій вкладці створюємо надпис *Заповнення бінарного файлу*, вписавши його у властивості Text. Для другої, третьої та четвертої вкладок створюємо надписи: *Виведення вмісту файлу, Відбір даних за умовою та Формування текстового файлу відповідно*. Для заповнення першої вкладки використовуємо компоненту `button1` з надписом *Записати дані у бінарний файл* та компоненту `dataGridView`. На останній позачергово створюємо вісім стовпців, і при їх створенні відразу задаємо тип даних та заголовки стовпців. Вигляд вікна додавання стовпців наведено на рисунку 3.1.

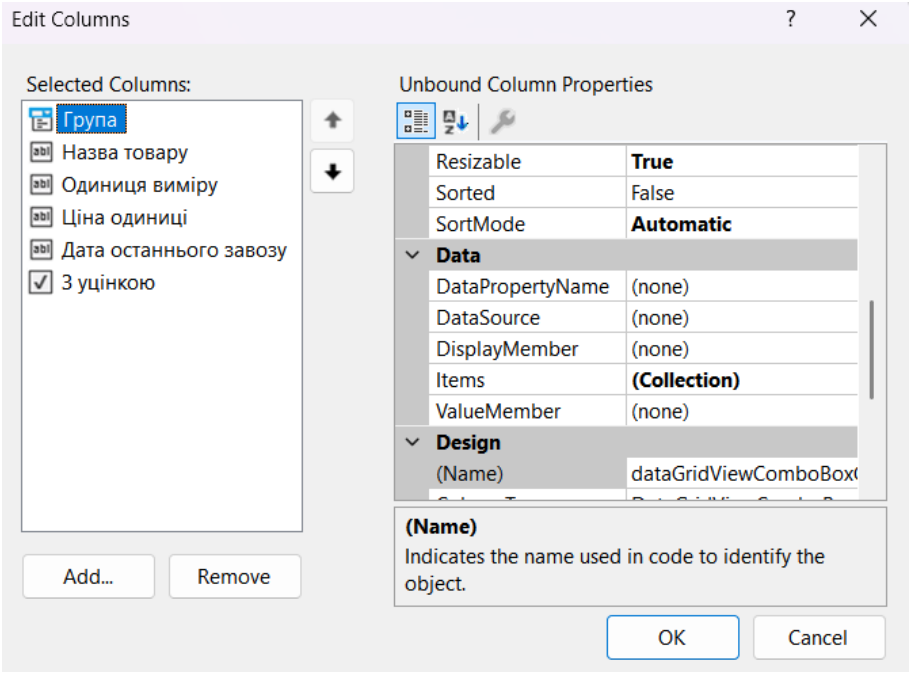

Рисунок 3.1 – Створення об’єкту таблиці `dataGridView1`

Формат для кожного стовпця і назву задаємо окремо:

– для 1-го стовпця – у властивості `HeaderText` вписати *Група*, а для властивості `ColumnType` замість значення за замовчуванням вибрати зі списку значення `DataGridViewComboBoxColumn`.

– для 2-го стовпця – у властивості `HeaderText` вписати *Назва товару*;

– для 3-го стовпця – у властивості `HeaderText` вписати *Одиниця виміру*.

– для 4-го стовпця – у властивості `HeaderText` вписати *Ціна одиниці*;

– для 5-го стовпця – у властивості `HeaderText` вписати *Дата останнього завозу*;

– для 6-го стовпця – у властивості `HeaderText` вписати *з Уцінкою*, у властивості `ColumnType` вибрати зі списку значення `DataGridViewCheckBoxColumn`.

В результаті форма набуде вигляду:

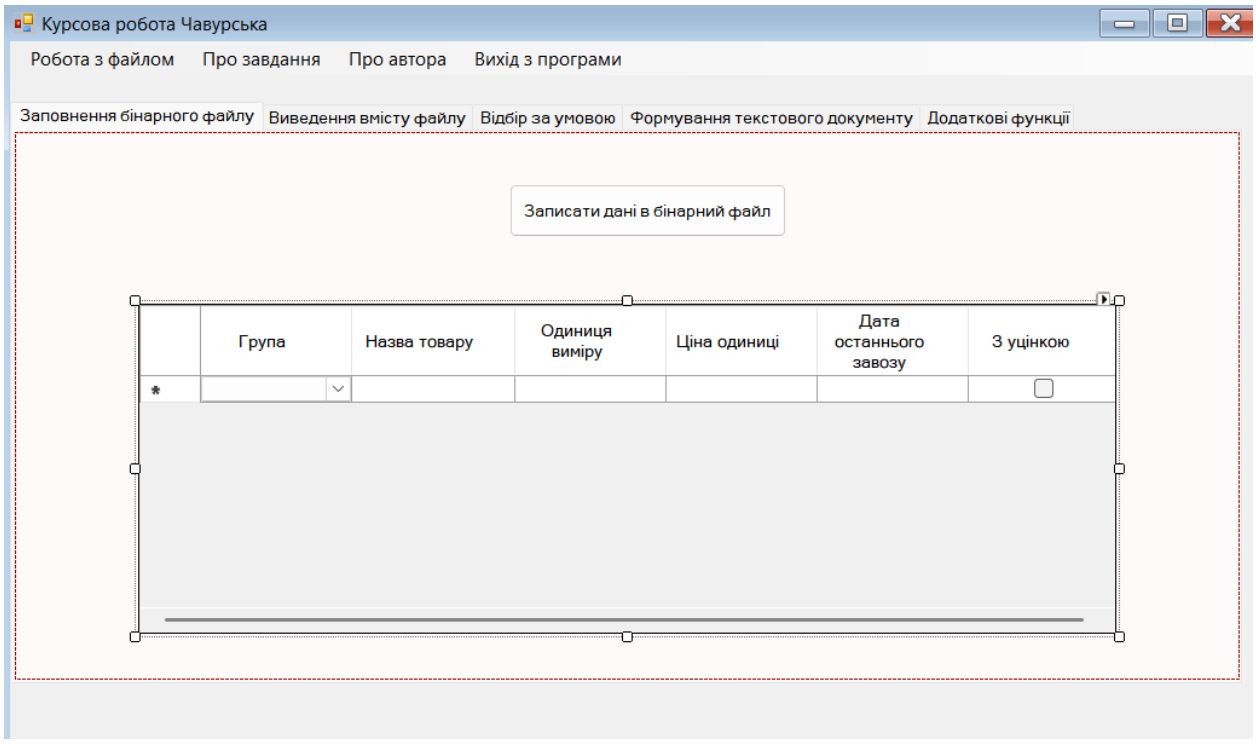

Рисунок 3.2 – Вигляд вкладки *Заповнення бінарного файлу*

Далі для таблиці задамо стиль заголовка стовпців у колекції властивостей `ColumnHeadersDefaultCellStyle` для `Alignment` вибрати значення `MiddleCenter`. Потім потрібно скопіювати `dataGridView1` і вставити його копії на другу, третю та четверту вкладки, створивши таким чином вже налаштовані `dataGridView2`, `dataGridView3` та `dataGridView4`, для яких змінити значення властивості `AllowUserToAddRows` на `false`.

На другій вкладці потрібно розташувати три кнопки `Button2`, `Button3` `Button4` та `comboBox2`. Для `Button2` у поле `text` пишемо *Переглянути бінарний файл,* для `Button3` пишемо *Сортувати товари в алфавітному порядку*, а для `Button4` – *Зберегти відредагований файл.*

Після таких налаштувань елементів на формі, вона набуде наступного вигляду:

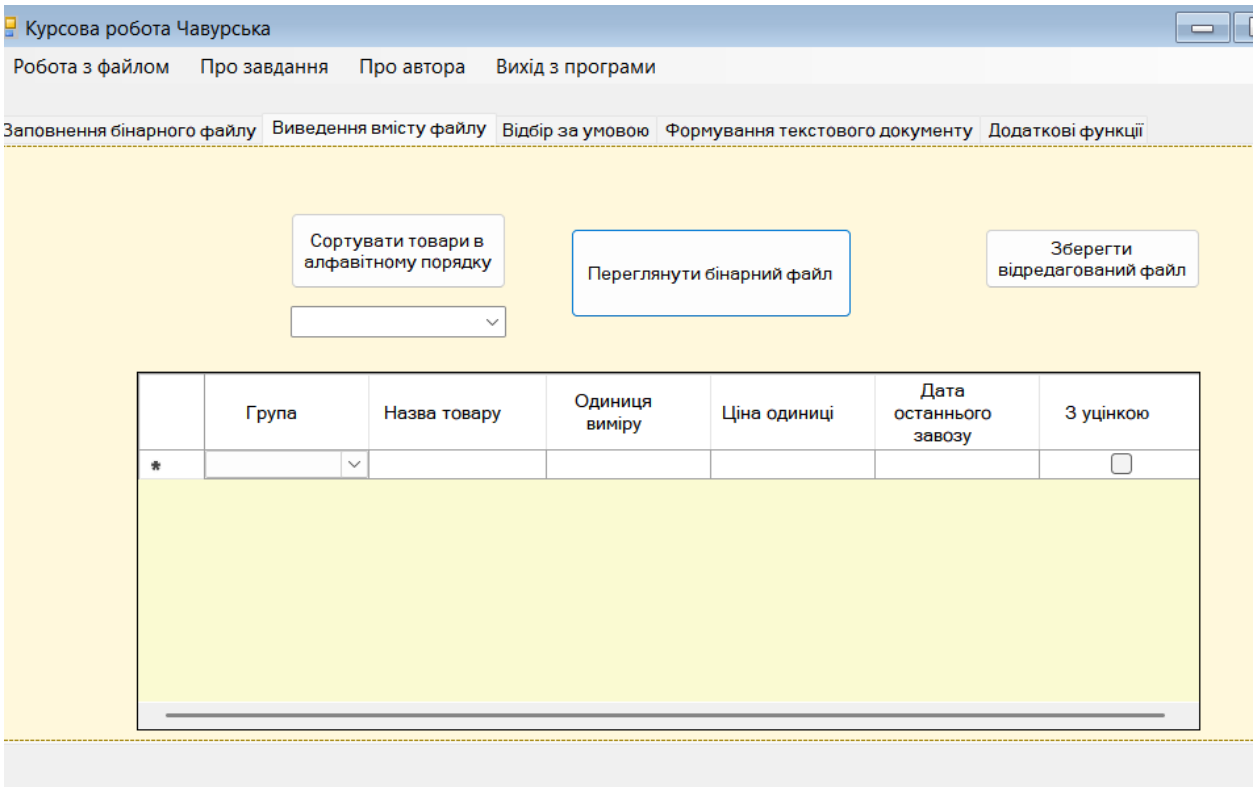

Рисунок 3.3 – Вигляд вкладки *Виведення вмісту файлу*

Відкривши третю вкладку, необхідно розташовати на ній кнопки `Button5`, `Button6` і вписати для них відповідний текст: *Відібрати товари молочного відділу*, *Видалити товари які уцінили* (див. Рис. 3.4).

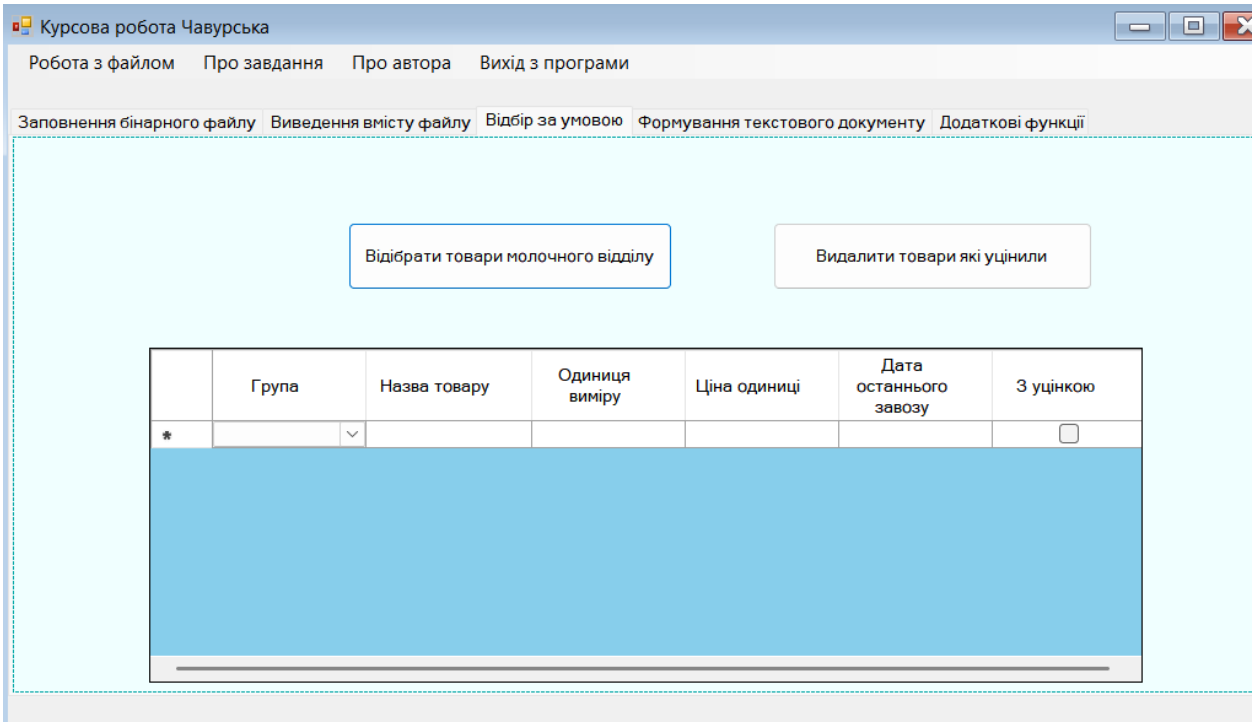

Рисунок 3.4 – Вигляд вкладки *Відбір даних за умовою*

Наступним кроком буде вписування команди для зазначення використання простору імен з програмними засобами файлового введення-виведення, після рядка `using namespace System` в коді головної форми потрібно `записати using namespace System::IO;`

Враховуючи, що ім'я файлу буде використовуватись у програмному коді функцій майже всіх кнопок, оголошуємо глобальну змінну та при створенні форми надаємо їй значення імені опрацьовуваного бінарного файлу. Шаблон функції `Form1_Load` можна створити подвійним клацанням миші на формі, подібно до того, як подвійним клацанням по відповідній кнопці створювали шаблони функцій `button_Click`.

3.2 Відбір даних за умовою з виведенням на екран

Реалізуємо можливість відбору даних на вкладці *Відбір за умовою*, для цього додаємо наступні елементи: `dataGridView3`, `button5` з текстом *Відібрати товари молочного відділу*. (див. Рис. 3.5).

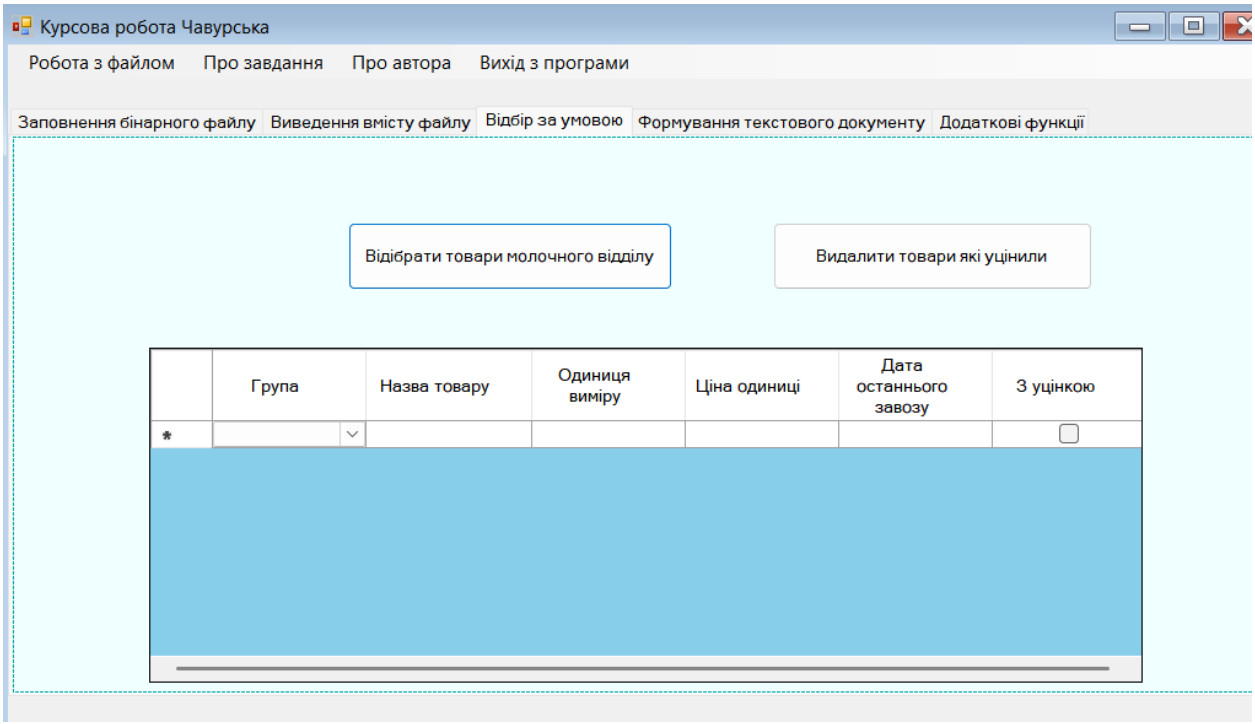

Рисунок 3.5 – Вигляд вкладки *Відбір даних за умовою*

Для реалізації фільтрації асортименту за певною категорією використовується подія натискання кнопки. У лістингу 3.1 здійснюється звернення до бінарного файлу, зчитування структурованих даних про товари та вибіркове відображення лише тих позицій, що належать до молочного відділу.

Лістинг 3.1 – Код кнопки «*Відібрати товари молочного відділу*»

```
	private: System::Void button5_Click(System::Object^ sender,
System::EventArgs^ e) {
	dataGridView3->Rows->Clear();
	if (!File::Exists("file.dat")) return;

	FileStream^ fs = gcnew FileStream("file.dat", FileMode::Open,
FileAccess::Read);
	BinaryReader^ br = gcnew BinaryReader(fs);

	while (br->BaseStream->Position < br->BaseStream->Length)
	{
		String^ group = br->ReadString();
		String^ name = br->ReadString();
		String^ unit = br->ReadString();
		double price = br->ReadDouble();
```

Продовження Лістингу 3.1

```
            DateTime date = DateTime::FromBinary(br->ReadInt64());
            bool discount = br->ReadBoolean();

            if (group == "молочний відділ")
                  dataGridView3->Rows->Add(group, name, unit, price,
date, discount);
      }

      br->Close();
      fs->Close();
}
```

3.3 Відбір даних бінарного файлу за умовою з формуванням текстового документу

Для формування текстового документу використовуємо четверту вкладку з назвою `Формування текстового файлу` (див. Рис. 3.6). Розташовуємо на ній кнопку `Button6` з текстом *Товари завезені за вказаною датою*.

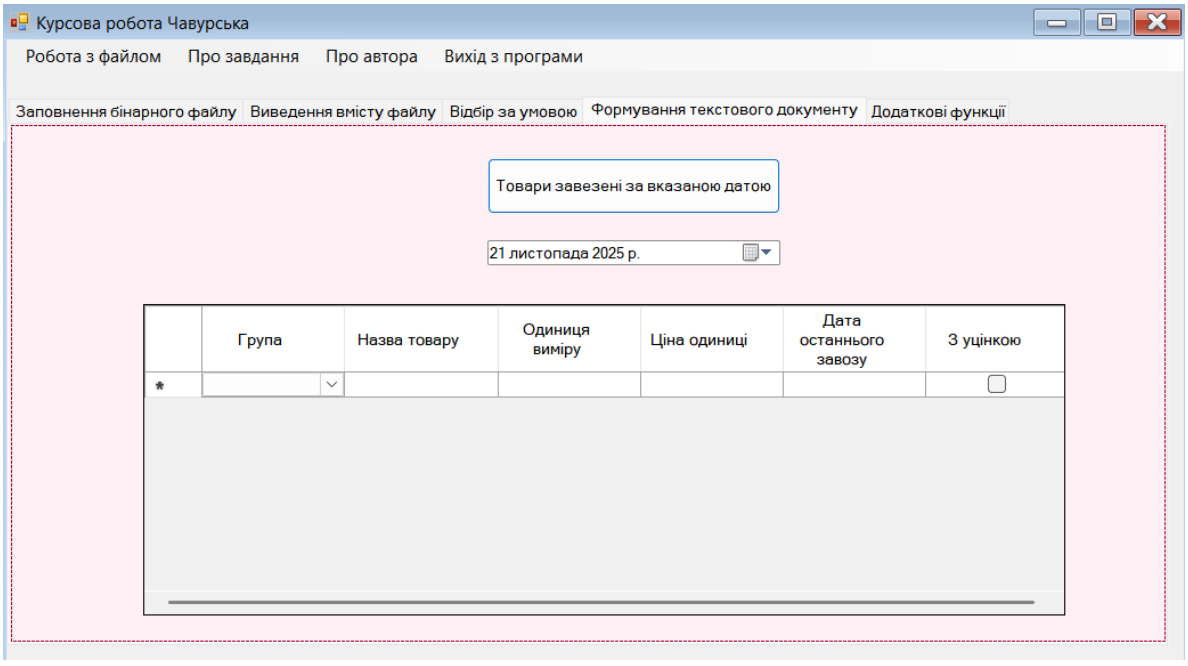

Рисунок 3.6 – Вигляд вкладки `Формування текстового файлу`

Код для опрацювання натиску на кнопку показано у Додатку А.

3.4 Видалення даних з бінарного файлу за умовою

Реалізуємо можливість видалення даних з бінарного файлу на вкладці *Відбір даних за умовою*. За умовою завдання курсової роботи потрібно видалити товари з уцінкою. Для виведення та видалення даних про товари з уцінкою використаємо однойменну кнопку (див. Рис. 3.7).

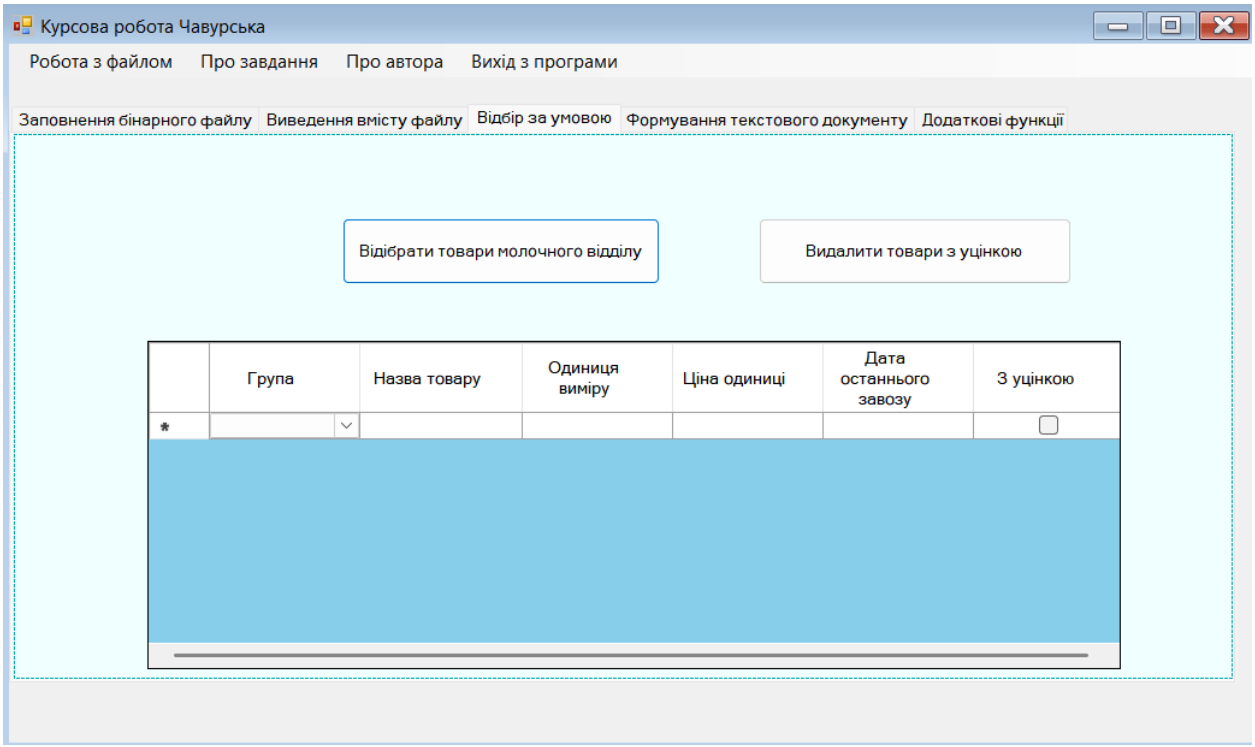

Рисунок 3.7 – Вигляд вкладки *Відбір даних за умовою*

Код для кнопки «*Видалити товари з уцінкою*» наведено у Додатку Б.

3.5 Створення додаткового функціоналу

Додатково було реалізовано функцію яка буде з шукати найдешевші товари. Створюємо `tabPage5`, розмістивши на ньому такі елементи, як, `dataGridView5, button8.` Для `button8` впишемо у властивості `Text` – *Знайти найдешевші товари*.

У результаті цих дій вікно набуде наступного вигляду:

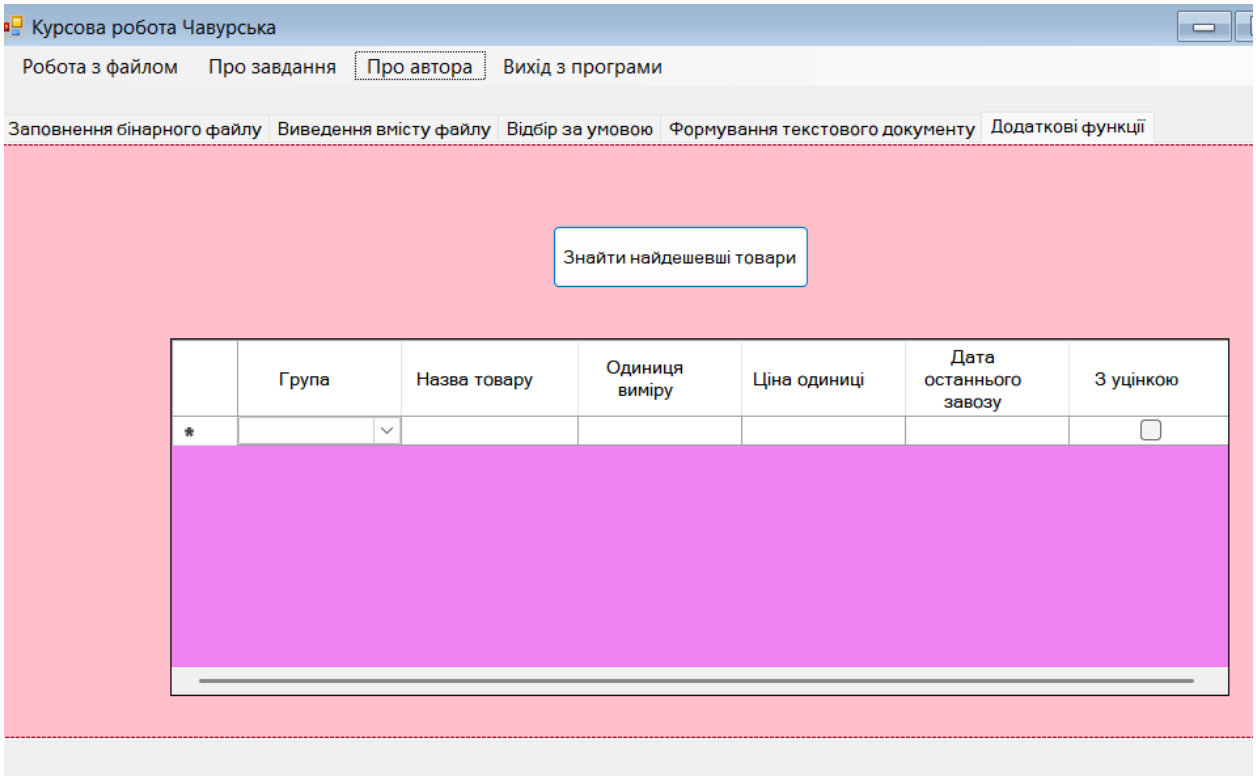

Рисунок 3.8 – Вигляд вкладки з додатковими функціями

Щоб реалізувати пошук найдешевших товарів, на вкладці tabPage5 було додано спеціальну кнопку, яка викликає відповідну функцію обробки. Під час її натискання програма зчитує інформацію з файлу, визначає мінімальну ціну та заповнює таблицю товарами, що їй відповідають. Логіка роботи кнопки подана в лістингу 3.2.

Лістинг 3.2 – Код кнопки «*Знайти найдешевші товари*»

```
private: System::Void button8_Click(System::Object^ sender,
System::EventArgs^ e)
{

	dataGridView5->Rows->Clear();

	String^ fileName = "file.dat";

	if (!File::Exists(fileName))
{
		MessageBox::Show("Файл не існує.");
```

Продовження Лістингу 3.2

```
            return;
}
      BinaryReader^ reader = gcnew
BinaryReader(File::OpenRead(fileName));

      double minPrice = Double::MaxValue;
      try {

            while (reader->BaseStream->Position < reader->BaseStream-
>Length) {
                  String^ group = reader->ReadString();
                  String^ name = reader->ReadString();
                  String^ unit = reader->ReadString();
                  double price = reader->ReadDouble();
                  Int64 ticks = reader->ReadInt64();
                  bool discount = reader->ReadBoolean();

                  if (price < minPrice) minPrice = price;
            }
            reader->BaseStream->Seek(0, SeekOrigin::Begin);

            while (reader->BaseStream->Position < reader->BaseStream-
>Length) {

                  String^ group = reader->ReadString();
                  String^ name = reader->ReadString();
                  String^ unit = reader->ReadString();
                  double price = reader->ReadDouble();
                  Int64 ticks = reader->ReadInt64();
                  DateTime date(ticks);
                  bool discount = reader->ReadBoolean();

                  if (price == minPrice) {
                        dataGridView5->Rows->Add(group, name, unit,
price, date, discount);
                  }
            }
      }

      finally {
            reader->Close();
      }
      if (dataGridView5->Rows->Count == 0)
{
            MessageBox::Show("Товари не знайдено.");
      }
}
```

4 ДОДАТКОВІ ФОРМИ ТА МЕНЮ

4.1 Функції основного модуля для команд меню

У створюваному проекті додаємо три нові форми:

1) форма з довідковою інформацією про автора роботи;

2) форма, яка дасть можливість ознайомитись із завданням на курсову роботу (у вигляді виведення вмісту текстового документа). Програмний код для цієї форми надає додаткові можливості редагування, форматування параметрів шрифту та зберігання змін у документі;

3) форма-заставка з тематичним рисунком, яка автоматично з’являтиметься після запуску програми та через тридцять секунд зникне. Програмний перехід до створених форм та додаткових функцій для роботи з файлом даних доцільно організувати за допомогою меню такої структури:

- робота з файлом;
- довідка про файл;
- резервне копіювання бінарного файлу;
- очистити бінарний файл;
- про завдання;
- про автора;
- вихід з програми.

На формі програмного проекту розміщуємо елемент меню `menuStrip1` та вписуємо відповідні надписи. Оскільки `menuStrip1` не є візуальною компонентою, вона відображається в додатковій стрічці знизу (див. Рис. 4.1).

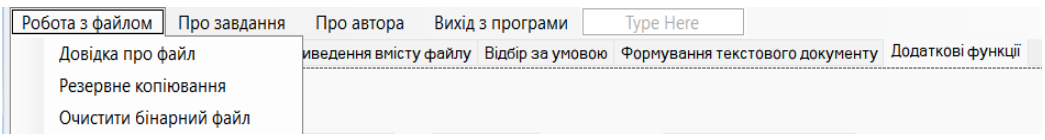

Рисунок 4.1 – Вигляд меню програми

Код для кожної із вкладок меню наведено у Додатку В.

4.2 Створення модуля Про автора

Для створення і додавання нової форми з інформацією про автора до проекту додаємо нове вікно з іменем `Avtor`. За допомогою елементів `label` вписати інформацію про автора курсової роботи, кнопки `button`, щоб завершити перегляд форми `Avtor` та `pictureBox1`, щоб вставити зображення заповнюємо вікно.

Зображення для елемента `pictureBox1` можна вибрати за допомогою властивості `Image`, а параметри шрифту для елементів `label` можна задати за допомогою колекції властивості `Font`. Після цього слід двічі клацнути по кнопці `button1`, використавши функцію закриття форми. Текст наведено у лістингу 4.1.

Лістинг 4.1 – Текст обробника подій закриття форми

```
private: System::Void button1_Click(System::Object^ sender,System::
EventArgs^ e) {
Close();
      }
```

Для того, щоб прив’язати нову форму до основної, у коді файлу `Avtor.h`, після команди `#pragma once` вписуємо команду `#include "Avtor.h"`. В результаті попередніх дій, отримаємо форму, яка матиме такий вигляд (див. Рис.4.2):

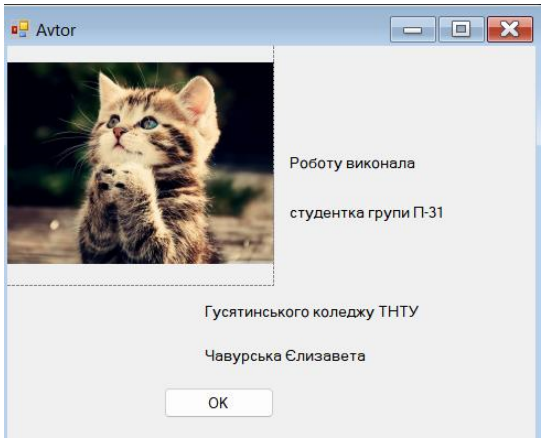

Рисунок 4.2 – Вікно *Про автора*

4.3 Створення модуля Завдання на роботу

До проєкту додаємо нову форму, для того щоб мати можливість переглянути постановку завдання у самій програмі, тому ми створюємо форму та задаємо їй ім’я – `Zavdan.h`

Налаштування форми `Zavdan` можна розпочати зі збільшення її ширини та формування на ній надпису *Завдання на роботу* (властивість `Text`), після чого розташувати на ній елементи `richTextBox1, menuStrip1, fontDialog1, openFileDialog1` та `saveFileDialog1`. В меню (елемент menuStrip1) цієї форми ввести текст чотирьох команд: *Показати завдання, Змінити шрифт, Зберегти зміни* та *Вихід.* Після цього форма набуде такого вигляду:

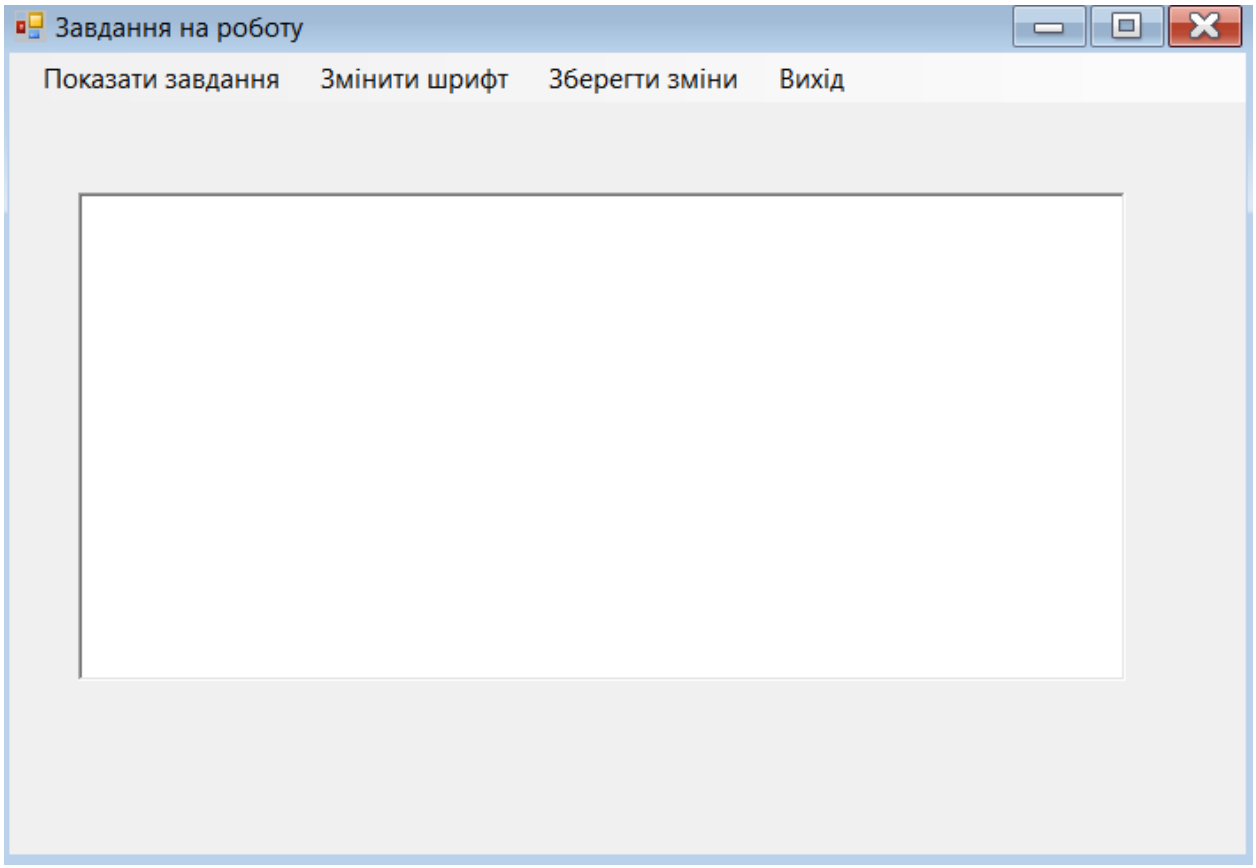

Рисунок 4.3 – Налаштування форми *Завдання на роботу*

Подвійним клацанням на пунктах меню автоматично створюються шаблони відповідних обробників подій, у які потрібно внести програмний код, наведений у Додатку Г. Після цього слід забезпечити зв’язок новоствореної

форми з головною формою застосунку. Для цього у верхній частині файлу головної форми необхідно додати `#include "Zavdan.h"`. Це дає змогу викликати додаткове вікно та працювати з ним як з частиною загального інтерфейсу програми.

Щоб користувач мав змогу переглядати текст завдання, створюється окремий документ у Word, у який вноситься повний опис роботи. Після заповнення документ зберігається у форматі RTF, оскільки такий тип файлів легко відкривати без сторонніх бібліотек. У програмі передбачено пункт меню *Завдання на роботу*, при натисканні якого запускається відповідний файл RTF для перегляду, результат матиме такий вигляд:

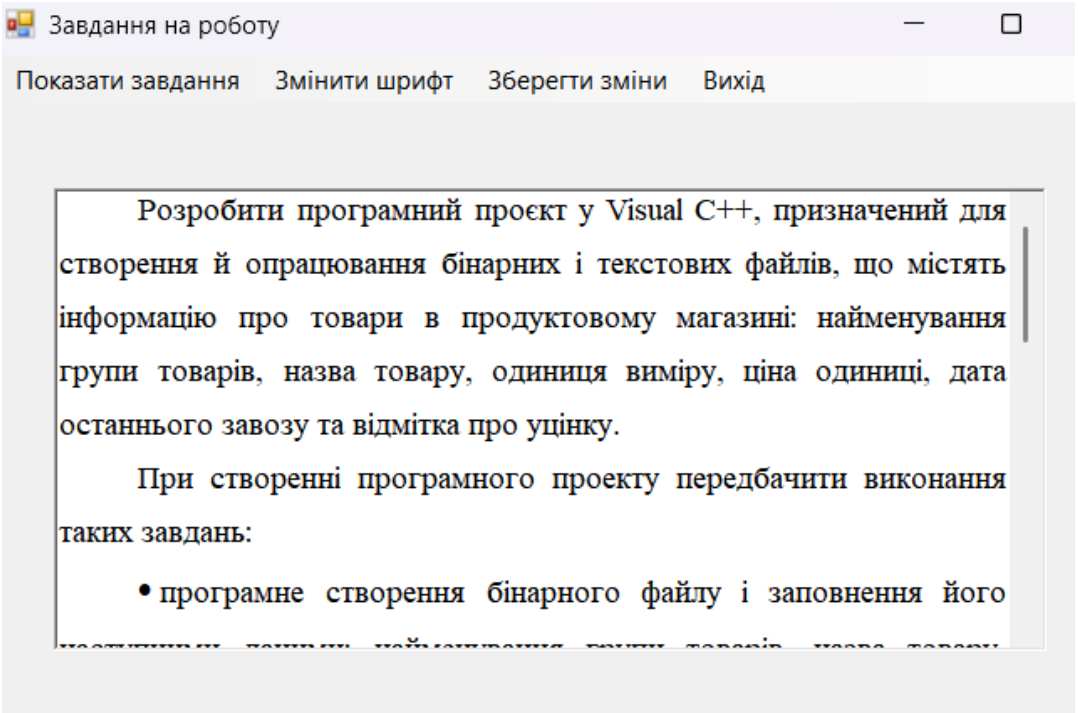

Рисунок 4.4 – Вікно інформації про завдання

4.4 Створення модуля Заставка

Створюємо нову форму-заставку. Розміщуємо на формі елементи `timer1` та `pictureBox1`, зображення до якого завантажити за допомого властивості `Image`. В результаті отримаємо вікно, яке наведено на рисунку 4.6.

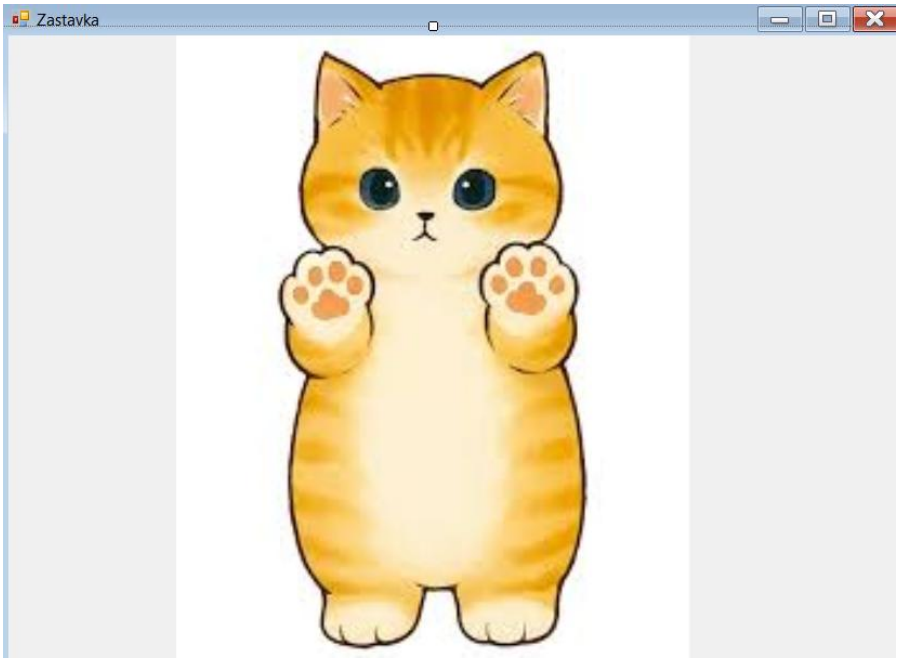

Рисунок 4.5 – Вікно заставки

У файлі `MyForm.h`, на початку проекту, долучимо модуль щойно створеної форми `#include "Zastavka.h"` Крім того, у функцію `Form1_Load` допишемо до того оператора, що там є, ще оператор для запуску форми–заставки (Див. Лістинг 4.3).

Лістинг 4.3 – Обробник події запуску заставки.

```
private: System::Void timer1_Tick(System::Object^ sender,
System::EventArgs^ e) {
     this->Opacity -= 0.02;
          if (this->Opacity == 0) Close();
}
private: System::Void Zastavka_Load(System::Object^ sender,
System::EventArgs^ e) {
     this->timer1->Start();

}
```

5 ЗДОБУТІ РЕЗУЛЬТАТИ У ВИГЛЯДІ ВІКОН

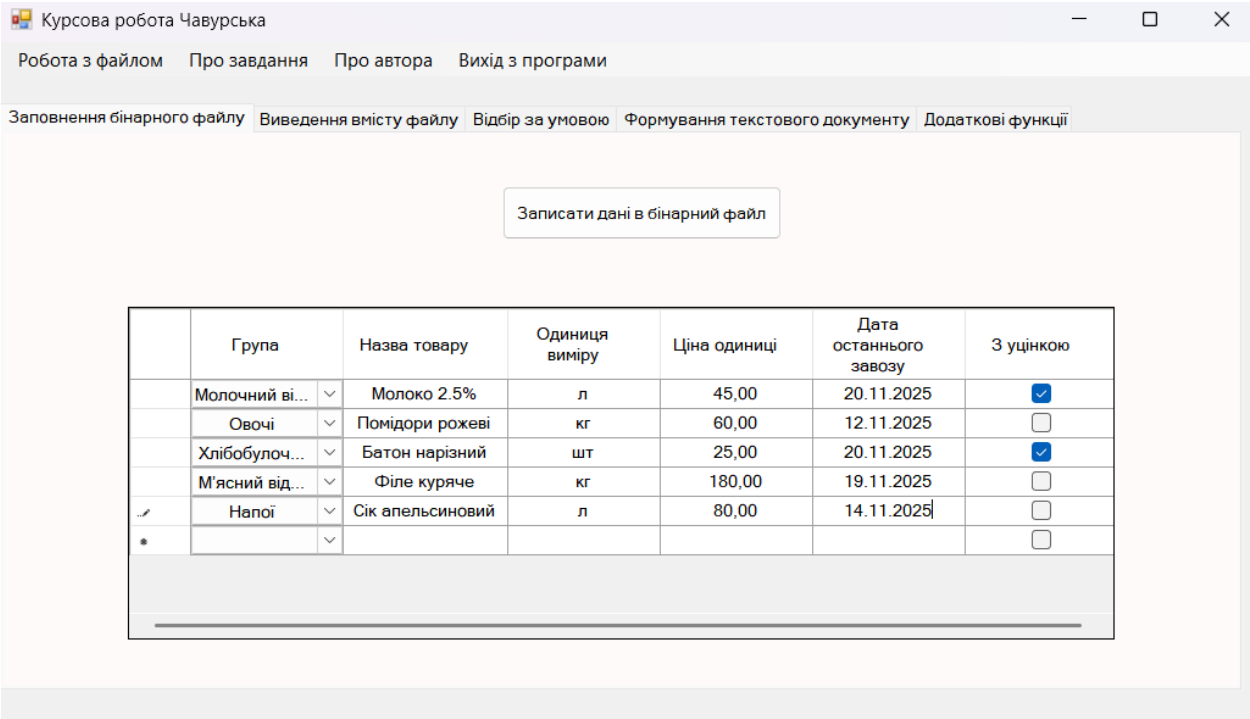

Рисунок 5.1 – Вкладка *Заповнення бінарного файлу*

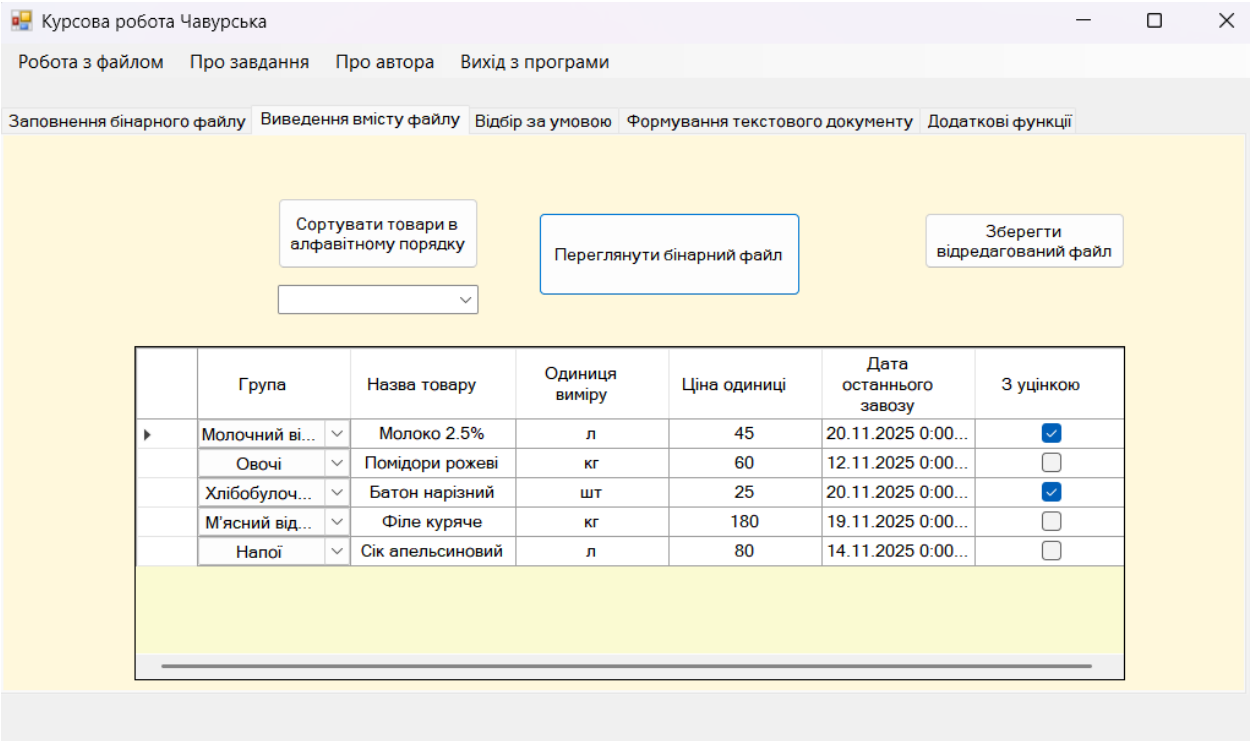

Рисунок 5.2 – Вкладка *Виведення вмісту файлу*

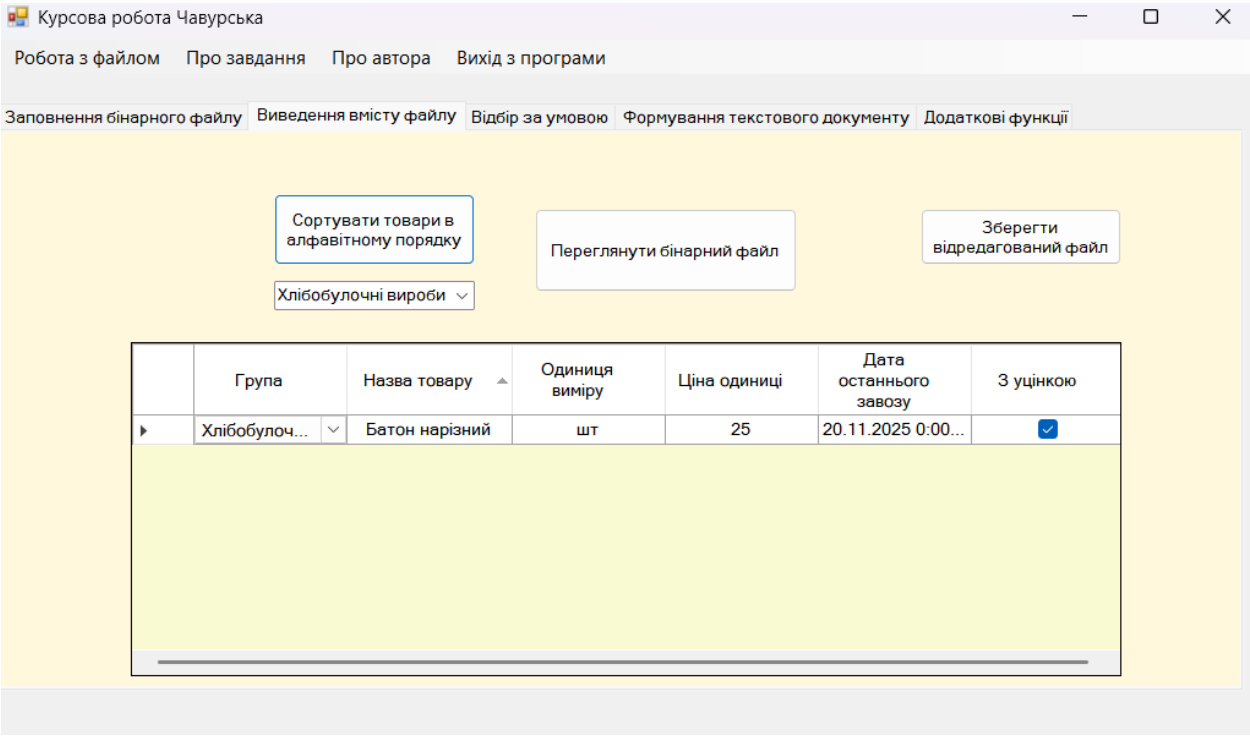

Рисунок 5.3 – Вигляд бінарного файлу після сортування товарів певної групи за алфавітом

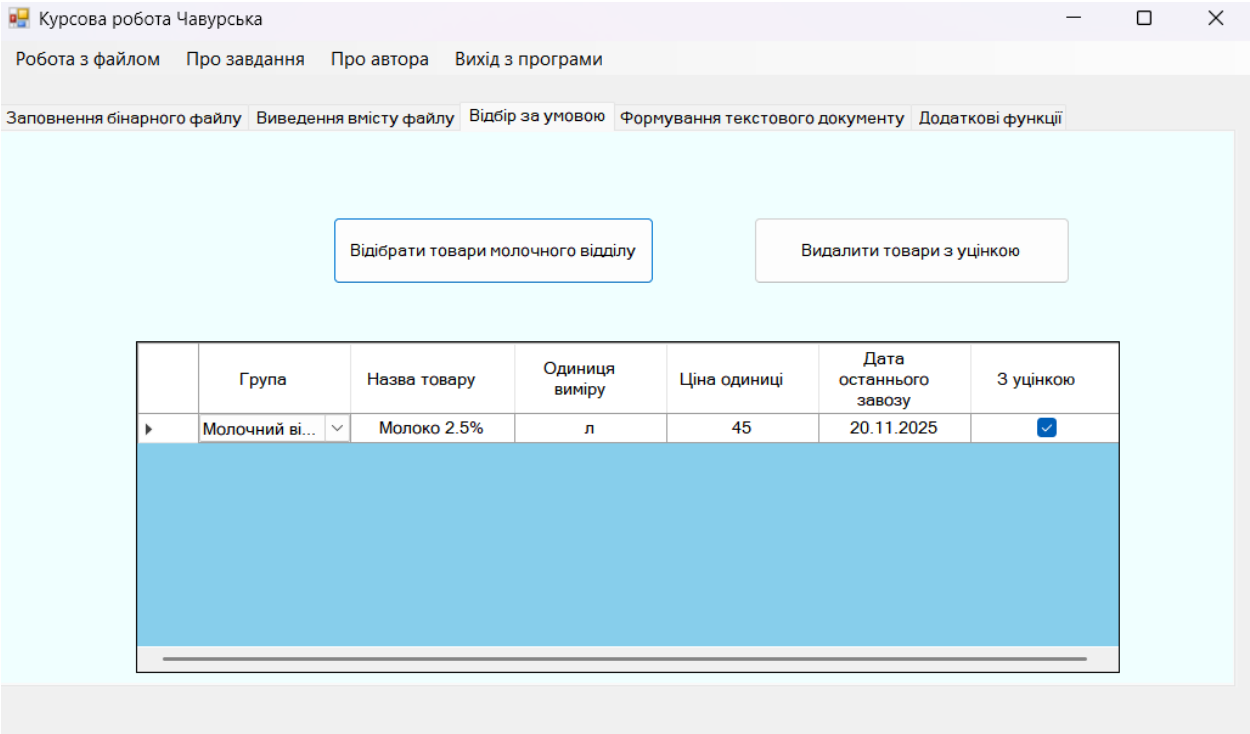

Рисунок 5.4 – Вкладка *Відбір даних за умовою*

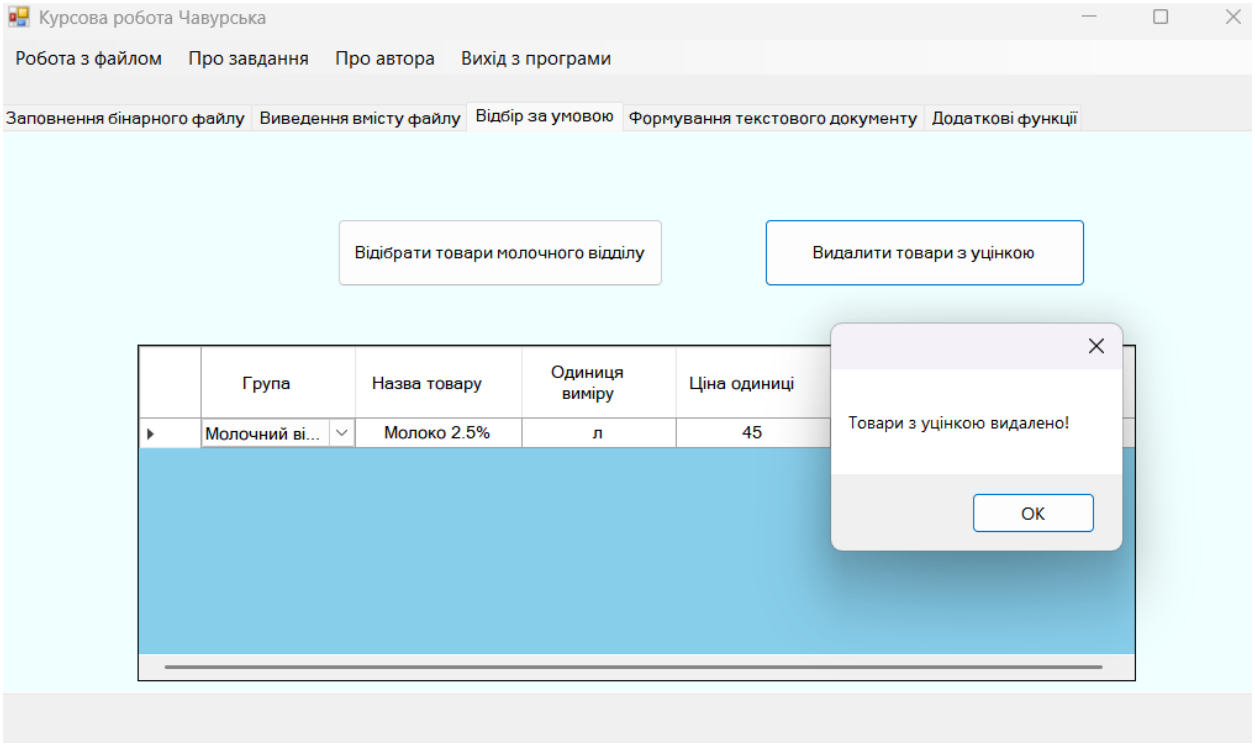

Рисунок 5.5 – Вкладка *Відбір за умовою*

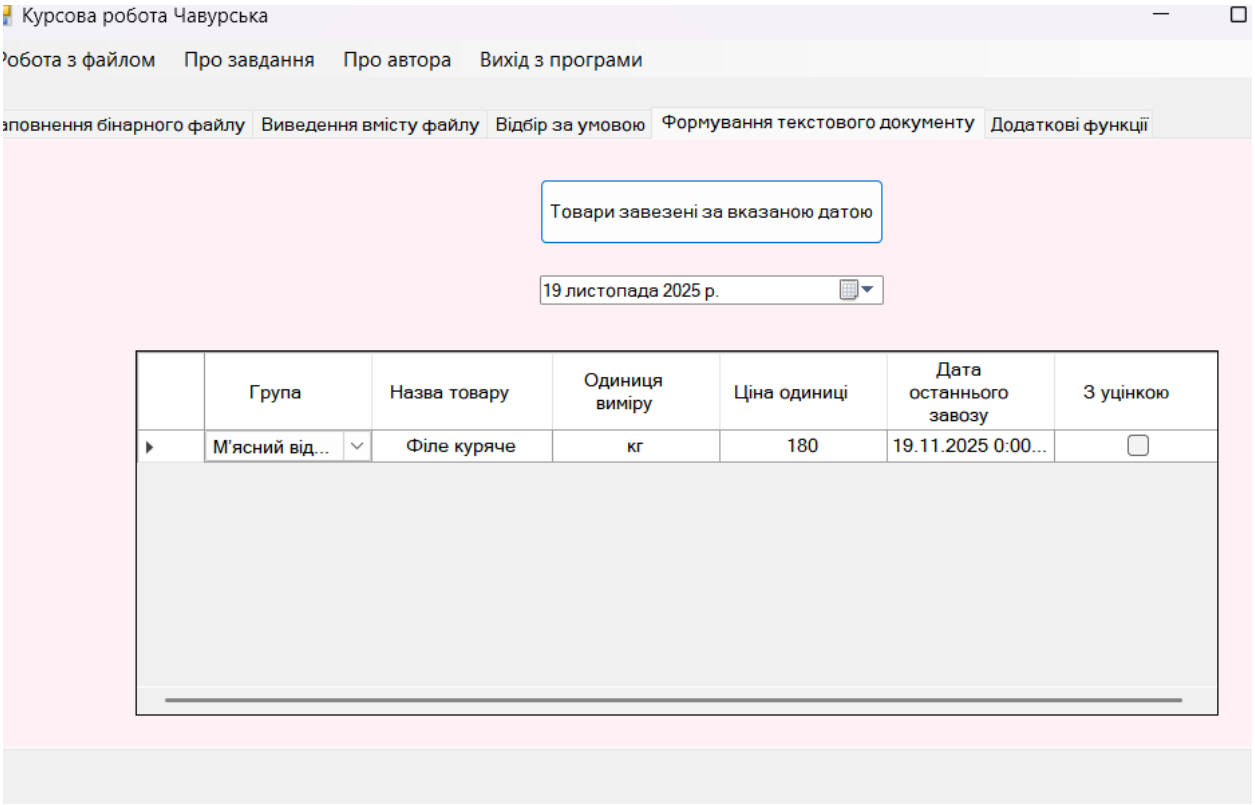

Рисунок 5.6 – Вкладка *Форматування текстового файлу*

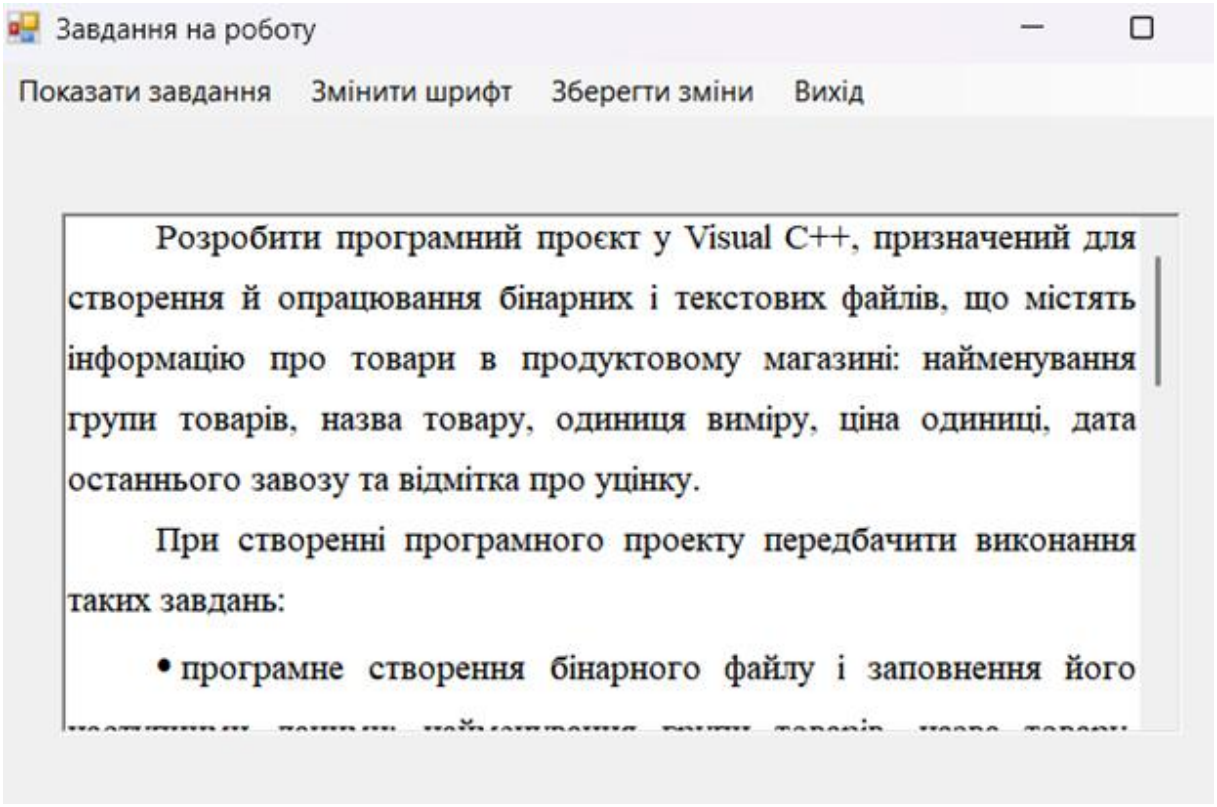

Рисунок 5.7 – Вигляд вікна *Завдання на роботу*

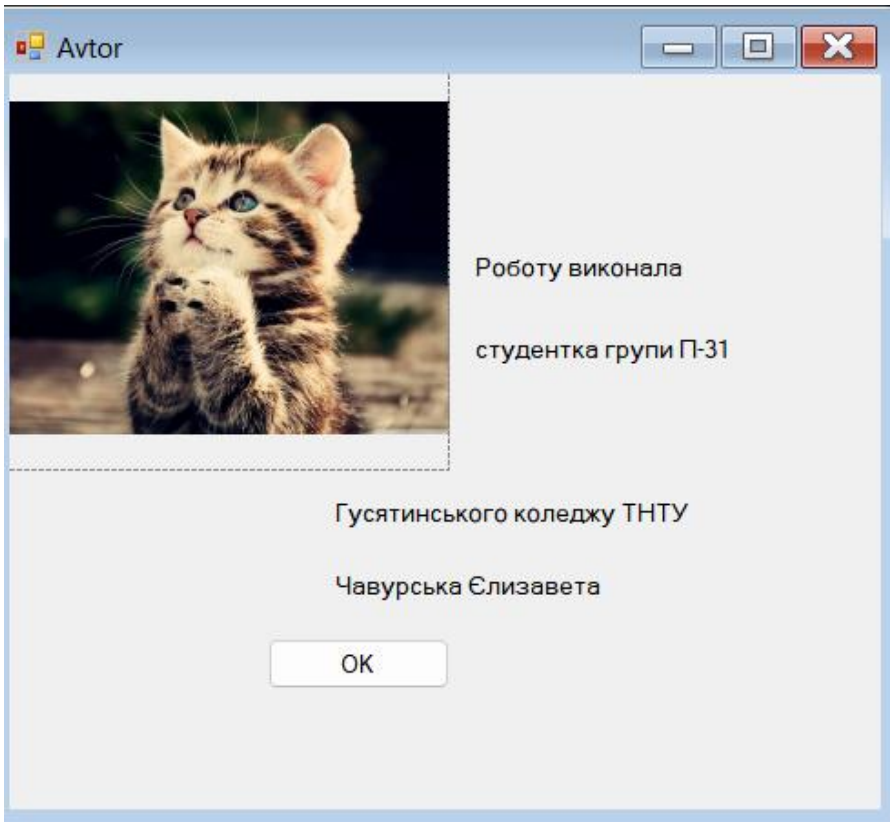

Рисунок 5.8 – Вигляд вікна *Про автора*

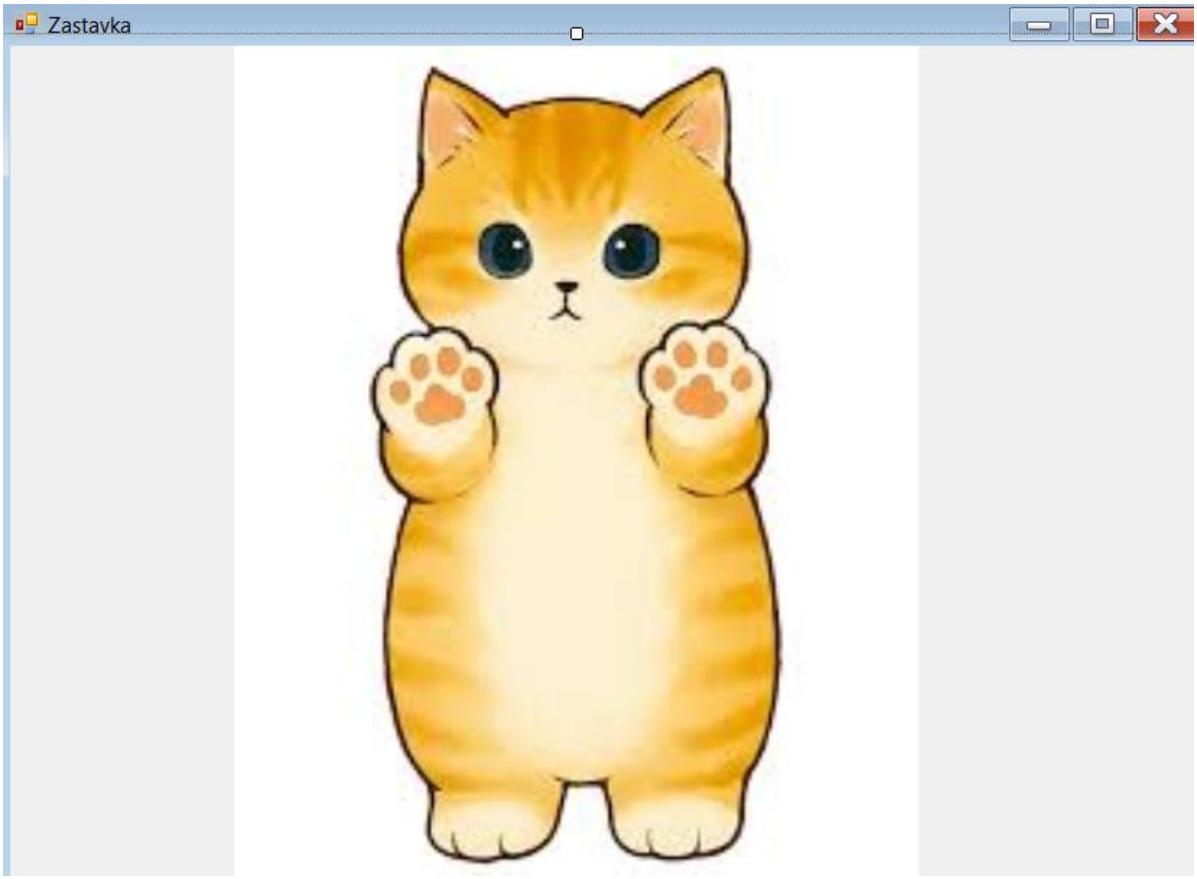

Рисунок 5.9 – Вікно заставки

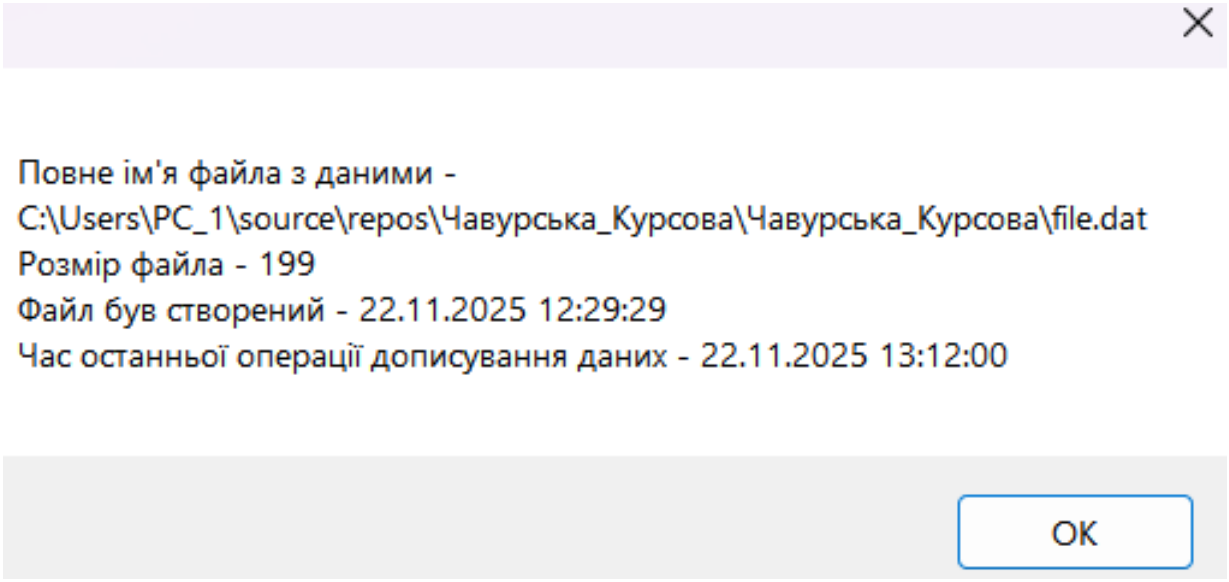

Рисунок 5.10 – Вигляд вікна довідки про файл

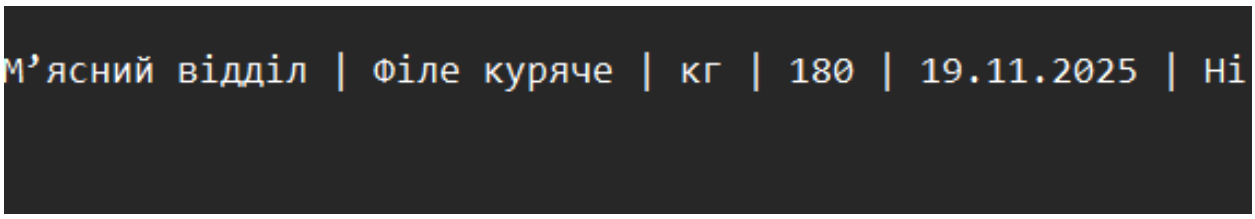

Рисунок 5.11 – Вміст текстового файлу `result.txt`

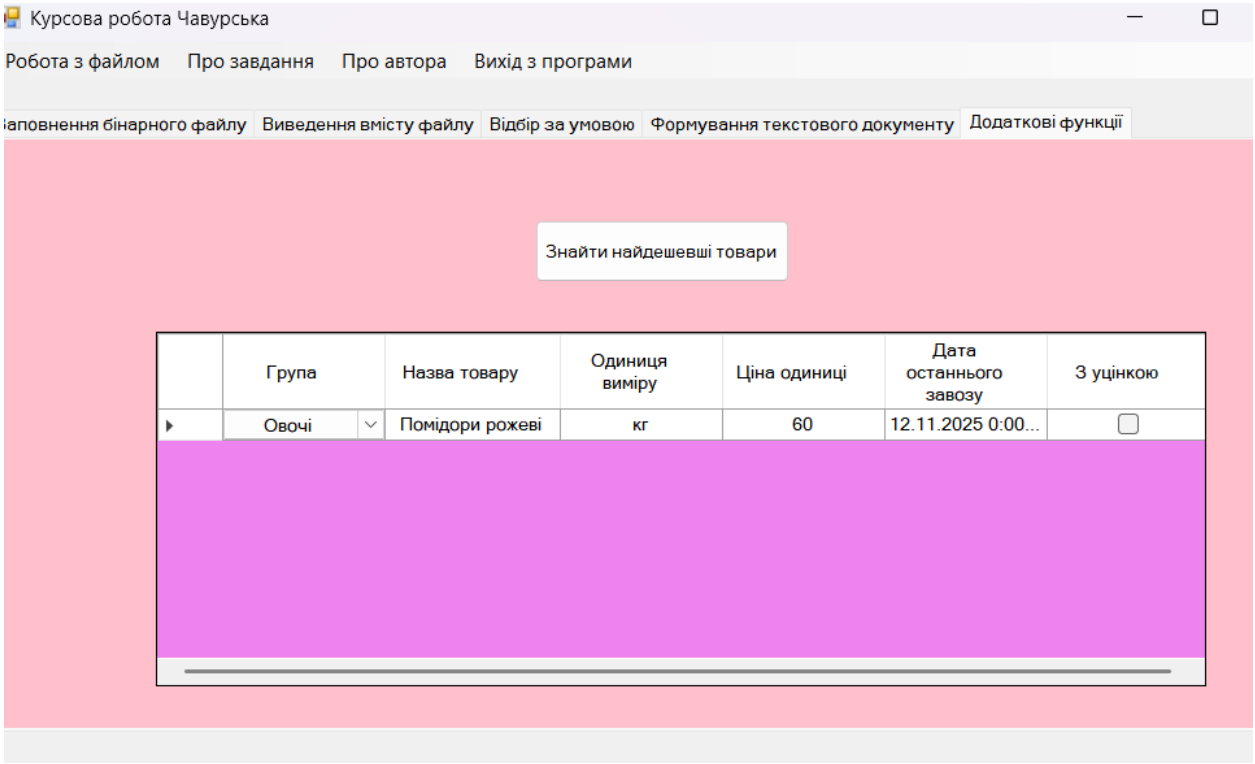

Рисунок 5.12 – Вкладка *Додаткові функції*

ВИСНОВКИ

Сучасний розвиток технологій сприяє появі нових програмних рішень, що значно полегшують роботу користувачів, зокрема тих, які пов’язані з обробкою даних у текстових та бінарних файлах. Виконуючи даний проєкт, були опановані практичні навички роботи з Windows Forms та управління різними типами файлів: створення і запис інформації, її відображення, вибірка даних за заданими критеріями, редагування і видалення записів, а також формування додаткових файлів, таких як звіти, на основі головного масиву даних.

Створена програма надає можливість ефективно вести облік і обробку інформації про товари продуктової торгової точки, включно з назвою товару, категорією, одиницею виміру, ціною, датою надходження та станом уцінки. Користувач може швидко фільтрувати товари за групами, сортувати за назвою, знаходити найвигідніші пропозиції, формувати текстові звіти і видаляти товари, які підлягають уцінці.

Інтерфейс програми розроблено з урахуванням простоти й зручності використання, що дозволяє легко застосовувати її у щоденних процесах магазину для роботи з великим обсягом інформації про асортимент товарів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С++. Теорія та практика : навч. посіб. / за ред. О. Г. Трофименко, Ю. Одеса: ОНАЗ ім. О.С. Попова , 2011. 587 с.

2. Трофименко О. Г., Прокоп Ю. В., Задерейко О. В. Алгоритмізація та програмування : навч.-метод. посіб. Одеса : Фенікс, 2020. 310 с.

3. Ковалюк Т. В. Основи програмування. Ковалюк. – К. : BHV, 2005. 384 с.

4. Васильєв О. В 191 Програмування на С++ в прикладах і задачах : Навч. посіб. / О. Васильєв. – Київ : Видавництво Ліра-К, 2017. – 382 с.

5. Бєлов Ю. А., Карнаух Т. О., Коваль Ю. В., Ставровський А. Б. Вступ до програмування мовою С++. Організація обчислень : навч. посіб./ Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставровський. – К. : Видавничополіграфічний центр “Київський університет”, 2012. – 175 с.

6. Шпак З.Я. Програмування мовою С / З.Я. Шпак. – Львів: ОріянаНова, 2006. – 432 с.

7. Основи програмування на С++. Зеленський О.С., Лисенко В.С. – Кривий Ріг: Державний університет економіки і технологій, 2023.-269 с.

8. Кривцова О.П. Програмування мовою С++. Технологія візуального програмування : навч. посіб. – Полтава : ПНПУ імені В.Г. Короленка, 2020. – 144 с.

9. Проектування програмних доданків: Частина І. Комп’ютерні практикуми: навч. посіб. для студ. спеціальності 151 – «Автоматизація та комп’ютерно-інтегровані технології» / КПІ ім. Ігоря Сікорського; уклад.: В. І. Бендюг, Б. М. Комариста. – Київ: КПІ ім. Ігоря Сікорського, 2018. – 285 с.

10. ДСТУ 3008 – 95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. URL :

11. Сайт з ресурсами по вивченню С++. URL: https://cplusplus.com/

12. Visual Studio 2015 C++. URL:https://learn.microsoft.com/enus/previous-versions/60k1461a(v=vs.140)?redirectedfrom=MSDN

13. Методичні вказівки до виконання курсової роботи з дисципліни Основи програмування та алгоритмічні мови для студентів спеціальності 121 «Інженерія програмного забезпечення» / Упоряд. К. І. Барціховська. Гусятин: ГК ТНТУ, 2019. 60с.

Додаток А

Код кнопки «*Товари завезені за вказаною датою.*»

```
	private: System::Void button6_Click(System::Object^ sender,
System::EventArgs^ e) {
	dataGridView4->Rows->Clear();
	if (!File::Exists("file.dat")) return;

	DateTime selectedDate = dateTimePicker1->Value.Date;

	StreamWriter^ sw = gcnew StreamWriter("result.txt", false);

	FileStream^ fs = gcnew FileStream("file.dat", FileMode::Open,
FileAccess::Read);
	BinaryReader^ br = gcnew BinaryReader(fs);

	while (br->BaseStream->Position < br->BaseStream->Length)
	{
		String^ group = br->ReadString();
		String^ name = br->ReadString();
		String^ unit = br->ReadString();
		double price = br->ReadDouble();
		DateTime date = DateTime::FromBinary(br->ReadInt64());
		bool discount = br->ReadBoolean();

		if (date.Date == selectedDate)
		{
			dataGridView4->Rows->Add(group, name, unit, price,
date, discount);

			sw->WriteLine(group + " | " + name + " | " + unit +
				" | " + price + " | " + date.ToShortDateString()
+
				" | " + (discount ? "Так" : "Ні"));
		}
	}

	sw->Close();
	br->Close();
	fs->Close();
}
```

Додаток Б

Код кнопки «*Видалити товари з уцінкою*»

```
private: System::Void button7_Click(System::Object^ sender,
System::EventArgs^ e) {
	dataGridView1->Rows->Clear();
	String^ fileName = "file.dat";
	String^ tmpFileName = "tmp.dat";
	if (!File::Exists(fileName)) {
		MessageBox::Show("Файл не існує.");
		return;
	}
	BinaryReader^ reader = gcnew
BinaryReader(File::OpenRead(fileName));
	BinaryWriter^ writer = gcnew BinaryWriter(File::Open(tmpFileName,
FileMode::Create));

	try {
		while (reader->BaseStream->Position < reader->BaseStream-
>Length) {
			String^ group = reader->ReadString();
			String^ name = reader->ReadString();
			String^ unit = reader->ReadString();
			double price = reader->ReadDouble();
			Int64 ticks = reader->ReadInt64();
			bool discount = reader->ReadBoolean();

			DateTime date(ticks);

			if (!discount) {
				writer->Write(group);
				writer->Write(name);
				writer->Write(unit);
				writer->Write(price);
				writer->Write(date.Ticks);
				writer->Write(discount);
			}
		}
	}
	finally {
		reader->Close();
		writer->Close();
	}

	File::Delete(fileName);
	File::Move(tmpFileName, fileName);
	MessageBox::Show("Товари з уцінкою видалено!");
}
```

Додаток В

Код вкладок menuStrip1

```
private: System::Void
довідкаПроФайлToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
	if (!File::Exists(bfname)) {
		MessageBox::Show("file not exists");
		return;
	}
	FileInfo^ f = gcnew FileInfo(bfname);
	MessageBox::Show("Повне ім'я файла з даними - " + f->FullName +
		"\nРозмір файла - " + f->Length.ToString() +
		"\nФайл був створений - " + f->CreationTime.ToString() +
		"\nЧас останньої операції дописування даних - " +
		f->LastWriteTime.ToString());
}
private: System::Void
резервнеКопіюванняБінарногоФайлуToolStripMenuItem_Click(System::Object
^ sender, System::EventArgs^ e) {
	if (!Directory::Exists("Copy"))
Directory::CreateDirectory("Copy");
FileInfo^ f = gcnew FileInfo(bfname);
f->CopyTo("Copy\\" + "RezervCopy.dat", true);
}
private: System::Void проАвтораToolStripMenuItem_Click(System::Object^
sender, System::EventArgs^ e) {
	Avtor^ fr2 = gcnew Avtor();
	fr2->ShowDialog();
}
private: System::Void очиститиToolStripMenuItem_Click(System::Object^
sender, System::EventArgs^ e) {
	BinaryWriter^ fb = gcnew BinaryWriter(File::Open(bfname,
FileMode::Create));
	fb->Close();
	dataGridView2->Rows->Clear();
}
private: System::Void
проЗавданняToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
	Zavdan^ fr2 = gcnew Zavdan();
	fr2->ShowDialog();
}
private: System::Void
вихідЗПрограмиToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
	Close();
}
```

Додаток Г

Код форми «Zavdan.h»

```
private: System::Void
показатиЗавданняToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
     openFileDialog1->Filter = "Doc files(*.rtf)|*.rtf";
     if (openFileDialog1->ShowDialog() ==
System::Windows::Forms::DialogResult::OK) {
          richTextBox1->LoadFile(openFileDialog1->FileName);
     }
}
private: System::Void
змінитиШрифтToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
     fontDialog1->ShowColor = true;
     fontDialog1->Font = richTextBox1->Font;
     if (fontDialog1->ShowDialog() !=
System::Windows::Forms::DialogResult::Cancel) {
          richTextBox1->SelectionFont = fontDialog1->Font;
          richTextBox1->SelectionColor = fontDialog1->Color;
     }
}
private: System::Void
зберегтиЗміниToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
     saveFileDialog1->Filter = "Doc files(*.rtf)|*.rtf";
     if (saveFileDialog1->ShowDialog() ==
System::Windows::Forms::DialogResult::OK) {
          richTextBox1->SaveFile(saveFileDialog1->FileName);
          richTextBox1->Modified = false;
     }
}
private: System::Void вихідToolStripMenuItem_Click(System::Object^
sender, System::EventArgs^ e) {

     if (richTextBox1->Modified) {
          if (MessageBox::Show("Текст було змінено.\nЗберегти зміни?",
"Закрити форму", MessageBoxButtons::YesNo, MessageBoxIcon::Question)
== System::Windows::Forms::DialogResult::Yes) {

     зберегтиЗміниToolStripMenuItem_Click(зберегтиЗміниToolStripMenuIt
em, e);
          }
     }
     Close();
}
```