СУЧАСНІ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

Microsoft® .NET™

ЧАСТИНА І
ПРАКТИЧНІ РОБОТИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Хіміко-технологічний факультет

СУЧАСНІ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ
ЧАСТИНА І
ПРАКТИЧНІ РОБОТИ

*Рекомендовано Методичною радою НТУУ «КПІ ім. Ігоря Сікорського»
як навчальний посібник для студентів,
які навчаються за спеціальністю 151 «Автоматизація та комп'ютерно-інтегровані технології»*

Київ
КПІ ім. Ігоря Сікорського
2019

СУЧАСНІ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ: ЧАСТИНА І. ПРАКТИЧНІ РОБОТИ [Електронний ресурс]: навч. посіб. для студ. спеціальності 151 – «Автоматизація та комп'ютерно-інтегровані технології» / КПІ ім. Ігоря Сікорського; уклад.: В. І. Бендюг, Б. М. Комариста. – Електронні текстові данні (1 файл: 2,14 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2019. – 269 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 9 від 30.05.2019 р.)
за поданням Вченої ради інституту/факультету (протокол № 4 від 22.04.2019 р.)

Електронне мережне навчальне видання

СУЧАСНІ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ
ЧАСТИНА І
ПРАКТИЧНІ РОБОТИ

Укладачі: Бендюг Владислав Іванович, канд. техн. наук, доцент
Комариста Богдана Миколаївна, канд. техн. наук

Відповідальний редактор: Д. М. Складанний, канд. техн. наук, доцент

Рецензенти: О.І. Букет, канд. техн. наук, доцент

Зміст

ВСТУП 7

ПРАКТИЧНА РОБОТА №1 8

1 Оператори введення/виведення в консольних додатках C++/CLI та C# 8

1.1 Специфіка C++/CLI та C#: форматування виведення 8

1.2 Клавіатурне введення в C++/CLI та C# 12

ПРАКТИЧНА РОБОТА №2 16

2 Сортування та пошук даних в одномірному масиві консольних додатків C++/CLI та C# 16

2.1 Масиви 16

2.2 Сортування одномірних масивів 20

2.3 Пошук в одномірному масиві 21

ПРАКТИЧНА РОБОТА №3 27

3 Робота з багатомірними масивами консольних додатків C++/CLI та C# 27

3.1 Багаторангові масиви 27

3.2 Зубчасті масиви 29

Програмний код 40

Лістинг 3.1 40

Лістинг 3.2 41

Лістинг 3.3 42

Лістинг 3.4 44

ПРАКТИЧНА РОБОТА №4 46

4 Створення додатку Windows Forms C++/CLI та C# 46

4.1 Початок роботи з додатком Windows Forms у CLR 46

4.2 Створення нового шаблону додатку 50

4.3 Початок роботи з додатком Windows Forms у C# 52

4.4 Створення нового додатку Windows Forms 54

Програмний код 59

Лістинг 4.1 59

Лістинг 4.2 60

Лістинг 4.3 66

Лістинг 4.4 67

Лістинг 4.5 71

ПРАКТИЧНА РОБОТА №5 72

5 Створення меню та панелей інструментів у C++/CLI та C# 72

5.1 Додавання меню в додатках Windows Forms 72

5.2 Додавання панелей інструментів в додатках Windows Forms 80

Програмний код 86

Лістинг 5.1 86
Лістинг 5.2 87
ПРАКТИЧНА РОБОТА №6 96
6 Табличне введення даних у C++/CLI та C# 96
6.1 Введення табличних даних 96
6.2 Побудова графіку з використанням елементу Chart 101
Програмний код 110
Лістинг 6.1 110
Лістинг 6.2 113
Лістинг 6.3 115
Лістинг 6.4 117
Лістинг 6.5 121
Лістинг 6.6 123
ПРАКТИЧНА РОБОТА №7 126
7 Редагування графічних даних у C++/CLI та C# 126
7.1 Найпростіше виведення графічного зображення у форму 126
7.2 Використання елементу PictureBox для відображення растрового файлу з можливістю прокрутки 129
7.3 Побудова графіку методами класу Graphics 134
7.4 Створення багатосторінкового інтерфейсу 141
Програмний код 147
Лістинг 7.1 147
Лістинг 7.2 149
Лістинг 7.3 150
Лістинг 7.4 154
Лістинг 7.5 156
Лістинг 7.6 160
Лістинг 7.7 163
Лістинг 7.8 166
Лістинг 7.9 167
ПРАКТИЧНА РОБОТА №8 170
8 Робота з базами даних у C++/CLI та C# 170
8.1 Створення бази даних MS Access 170
8.2 Запис структури таблиці в порожню базу даних MS Access. Програмне підключення до БД 174
8.3 Зчитування даних з БД в сітку даних DataGridView з використанням об'єктів класів Command, Adapter та DataSet 177
8.4 Оновлення записів в таблиці бази даних MS Access 181

Програмний код 192
Лістинг 8.1 192
Лістинг 8.2 196
Лістинг 8.3 198
ПРАКТИЧНА РОБОТА №9 201
9 Використання у програмах C++/CLI та C# функцій MS Word та MS Excel 201
9.1 Перевірка правопису засобами MS Word 201
9.2 Створення таблиці в MS Word 207
9.3 Використання функцій MS Excel 212
9.4 Побудова діаграм засобами MS Excel 218
Програмний код 235
Лістинг 9.1 235
Лістинг 9.2 238
Лістинг 9.3 240
Лістинг 9.4 242
Лістинг 9.5 245
Лістинг 9.6 247
Лістинг 9.7 249
Лістинг 9.8 252
Лістинг 9.9 254
Лістинг 9.10 256
Лістинг 9.11 263
Лістинг 9.12 267

ВСТУП

Практичні роботи виконуються з метою закріплення знань, одержаних при читанні лекцій та самостійній роботі та вироблення вмінь і досвіду використання сучасних засобів програмування. Практичні роботи допомогають підготуватися до виконання індивідуальних завдань на комп'ютерних практикумах.

В рамках дисципліни передбачене виконання 9 приктичних робіт наступної тематики:

№ з/п	Назва комп'ютерного практикуму
1	*Оператори введення/виведення в консольних додатках C++/CLI та C#*
2	*Сортування та пошук даних в одномірному масиві консольних додатків C++/CLI та C#*
3	*Робота з багатомірними масивами консольних додатків C++/CLI та C#*
4	*Створення додатку Windows Forms C++/CLI та C#*
5	*Створення меню та панелей інструментів у C++/CLI та C#*
6	*Табличне введення даних у C++/CLI та C#*
7	*Редагування графічних даних у C++/CLI та C#*
8	*Робота з базами даних у C++/CLI та C#*
9	*Використання у програмах C++/CLI та C# функцій MS Word та MS Excel*

Ваговий бал за практичну роботу – 2. Максимальна кількість балів за всі практичні роботи дорівнює 2 бали * 9 = 18 балів. Рейтингові бали кожної практичної роботи складаються з балів за виконання роботи (від 1 до 2). Таким чином за результатами кожної практичної роботи студент може отримати від 1 до 2 балів.

ПРАКТИЧНА РОБОТА №1

1 Оператори введення/виведення в консольних додатках C++/CLI та C#

1.1 Специфіка C++/CLI та C#: форматування виведення

Функції `Write()` і `WriteLine()` класу `Console` - мають засоби управління форматом виведення, і цей механізм працює абсолютно однаково в обох функціях. Найпростіше зрозуміти його на прикладі. Для початку подивимося, як можна за допомогою одного оператора отримати виведення, яке мало б складатися з трьох операторів.

Приклад коду C++/CLI

```
int packageCount = 25;
Console::WriteLine(L"Наявні {0} пакетів.", packageCount);
```

Приклад коду C#

```
int packageCount = 25;
Console.WriteLine("Наявні {0} пакетів.", packageCount);
```

Тут другий оператор забезпечить наступне виведення

```
Наявні 25 пакетів.
```

Перший аргумент функції `WriteLine()` - це рядок `L"Наявні {0} пакетів."`, в якому фрагмент `{0}` позначає місце, куди буде поміщений другий аргумент. Цей фрагмент містить в собі форматний рядок, який застосовується для виведення другого аргументу, хоча в даному випадку він гранично простий і складається з одного нуля. Аргументи, які йдуть у функції `WriteLine()` за першим, пронумеровані по порядку, починаючи з нуля.

Приклад коду C++/CLI

```
// Порядок аргументів: 0 1 2 і т.д.
Console::WriteLine("Рядок формату", arg2, arg3, arg4, ...);
```

Таким чином, нуль, укладений у фігурні дужки в попередньому фрагменті коду, вказує на те, що аргумент `packageCount` повинен замінити місце фрагмента `{0}` в рядковому аргументі при виведенні його на консоль.

Якщо потрібно вивести ще й вагу пакетів разом з кількістю, напишіть так:

Приклад коду C++/CLI

```
int packageCount = 25; double packageWeight = 7.5;
Console::WriteLine(L"Наявні {0} пакетів вагою {1} кілограмів.",
    packageCount, packageWeight);
```

Приклад коду C#

```
int packageCount = 25; double packageWeight = 7.5;
Console.WriteLine("Наявні {0} пакетів вагою {1} кілограмів.",
    packageCount, packageWeight);
```

Тепер оператор виведення має три аргументи, і до другого і до третього в форматному рядку виконується звернення за номерами 0 і 1 відповідно. Тому це дасть наступне виведення:

```
Наявні 25 пакетів вагою 7,5 кілограмів.
```

Можете також написати оператор, який виводить два останніх аргументу в зворотному порядку:

Приклад коду C++/CLI

```
Console::WriteLine(L"Наявні {1} пакетів вагою {0} кілограм.",
    packageWeight, packageCount);
```

Тепер 0 в рядку формату відноситься до змінної packageWeight, а 1 - до packageCount; виведення буде точно таким ж, як раніше.

Ви також можете вказати, як будуть представлені дані в командному рядку. Припустимо, потрібно вивести значення з плаваючою комою packageWeight з двома десятковими розрядами після точки. Це можна зробити наступним чином:

Приклад коду C++/CLI

```
Console::WriteLine(L"Наявні {0} пакетів вагою {1:F2} кілограм.",
    packageCount, packageWeight);
```

Приклад коду C#

```
Console.WriteLine("Наявні {0} пакетів вагою {1:F2} кілограмів.",
    packageCount, packageWeight);
```

У підрядку {1:F2} двокрапка відокремлює значення індексу 1, що ідентифікує аргумент, від наступного за ним вказівника формату F2. Літера F в визначенні формату вказує, що виведення повинне бути в формі «±ddd.dd ...» (де d представляє десяткову цифру, а 2 - це кількість розрядів після точки). Виведення цього оператора буде таким:

```
Наявні 25 пакетів вагою 7,50 кілограмів.
```

У загальному випадку, можете використовувати специфікацію формату у вигляді {n,w:Axx}, де n - значення індексу, що вказує номер аргументу, наступного за форматованим рядком; w - необов'язкова специфікація ширини

поля; A - односимвольна специфікація формату значення; xx - необов'язкове одно- або двозначне число, що задає точність виведення значення. Специфікація ширини поля - ціле зі знаком. Значення буде вирівняно праворуч у полі w, якщо ширина позитивна, і ліворуч - якщо негативна. Якщо значення займає менше знаків, ніж зазначено в аргументі w, то виведення доповнюється пробілами; якщо значення не вміщується в ширину w, то специфікація ширини ігнорується. Ось ще один приклад.

Приклад коду C++/CLI

```
Console::WriteLine(L"Пакетів:{0,3} Вага:{1,10:F3} кілограм.",
      packageCount, packageWeight);
```

Приклад коду C#

```
Console.WriteLine("Пакетів:{0,3} Вага:{1,10:F3} кілограм.",
      packageCount, packageWeight);
```

Кількість пакетів виводиться в поле шириною три знаки, а вага - в поле шириною десять знаків, тому в результаті отримаємо:

```
Пакетів: 25 Вага:     7,500 кілограм.
```

Існують і інші специфікатори формату, які дозволяють представляти різні типи даних різноманітними способами. Нижче наведені деякі з найбільш часто використовуваних специфікаторів (табл. 1.1).

Таблиця 1.1 – Основні специфікатори форматування консольного виведення

Специфікатор формату	Опис
C або c	Виводить значення в грошовому форматі
D або d	Виводить ціле число в десятковому вигляді. Якщо вказати більш високу точність десяткових знаків в числі, то воно буде доповнено нулями зліва
E або e	Виводить значення з плаваючою комою в науковій нотації, тобто з експонентою. Значення точності вказує кількість десяткових розрядів після точки
F або f	Виводить значення з плаваючою комою як число з фіксованою комою в вигляді ±ddd.dd...
G або g	Виводить значення в найбільш компактному вигляді, в залежності від його типу і зазначеної точності. Якщо точність не вказана, приймається значення точності, задане за замовчуванням
N або n	Виводить десяткове значення з плаваючою комою, використовуючи при необхідності роздільник - кому між групами по три розряди
X або x	Виводить ціле число в шістнадцятковому вигляді. Шістнадцятиричні цифри вводяться в верхньому або нижньому регістрі в залежності від того, зазначений знак X або x

Це дає достатні знання щодо виведення, щоб продовжити вивчати приклади C++/CLI та C#. Тепер подивимося на застосування описаного в дії.

Завдання 1.

Створити консольний додаток для реалізації демонстраційного прикладу програми, що обчислює ціну килимового покриття.

Приклад коду C++/CLI

```
#include "stdafx.h"

using namespace System;

int main(array<System::String ^> ^args)
{
	double carpetPriceSqMt = 27.95;
	double roomWidth = 13.5; // В метрах
	double roomLength = 24.75; // В метрах
	const int feetPerMeter = 3;
	double roomWidthMts = roomWidth / feetPerMeter;
	double roomLengthMts = roomLength / feetPerMeter;
	double carpetPrice = roomWidthMts * roomLengthMts *
		carpetPriceSqMt;
	Console::WriteLine(L"Розмір кімнати {0:F2} метрів на {1:F2}
		метрів",
		roomLengthMts, roomWidthMts);
	Console::WriteLine(L"Площа кімнати {0:F2} квадратних метрів",
		roomLengthMts * roomWidthMts);
	Console::WriteLine(L"Ціна килимового покриття ${0:F2}",
		carpetPrice);
	return 0;
}
```

Приклад коду C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleAppCsh_1
{
	class Program
	{
		static void Main(string[] args)
		{
			double carpetPriceSqMt = 27.95;
			double roomWidth = 13.5; // В метрах
			double roomLength = 24.75; // В метрах
			const int feetPerMeter = 3;
			double roomWidthMts = roomWidth / feetPerMeter;
```

```
                double roomLengthMts = roomLength / feetPerMeter;
                double carpetPrice = roomWidthMts * roomLengthMts *
                    carpetPriceSqMt;
                Console.WriteLine("Розмір кімнати {0:F2} метрів на
                    {1:F2} " + "метрів", roomLengthMts, roomWidthMts);
                Console.WriteLine("Площа кімнати {0:F2} квадратних
                    метрів", roomLengthMts * roomWidthMts);
                Console.WriteLine("Ціна килимового покриття ${0:F2}",
                    carpetPrice);
            }
        }
}
```

Виведення в консолі цієї програми буде наступним

```
Розмір кімнати 8,25 метрів на 4,50 метрів
Площа кімнати 37,13 квадратних метрів
Ціна килимового покриття $1037,64
```

Оператори виведення використовують специфікатор формату F2, щоб обмежити вихідні значення двома десятковими знаками після крапки. Без цього у виведенні може вийти більше знаків, що небажано, особливо коли мова йде про ціну. Щоб побачити різницю, видаліть специфікатор формату.

Оператор виведення площі використовує арифметичний вираз у другому аргументі функції `WriteLine()`. Компілятор спочатку вирахує цей вираз, а потім передасть результат у вигляді аргументу функції. У загальному випадку ви завжди можете використовувати вираз як аргумент функції - до тих пір, поки тип отриманого результату узгоджується з типом параметра функції.

1.2 Клавіатурне введення в C++/CLI та C#

Можливості введення з клавіатури, які є у консольних програм .NET Framework, дещо обмежені. Ви можете прочитати повний рядок введення як текстовий рядок, використовуючи для цього *функцію `ReadLine()`* класу *`Console`*, або ж прочитати окремий символ, застосовуючи для цього *функцію `Read()`* того ж класу. Можна також прочитати натискання клавіші за допомогою функції `ReadKey()`.

Функція `ReadLine()` використовується наступним чином:

Приклад коду C++/CLI

```
String^ line = Console::ReadLine();
```

Приклад коду C#

```
string line = Console.ReadLine();
```

Цей оператор читає весь вхідний рядок тексту, завершений натисканням клавіші <Enter>. Змінна `line` у коді C++/CLI має тип `String^` і зберігає

посилання на рядок, який виходить в результаті виконання функції Console::ReadLine(). Символ ^ після імені типу String вказує, що це - *дескриптор* (handle), який посилається на об'єкт типу String.

У коді на C# змінна line має тип string і символ ^ непотрібний для створення об'єкту певного класу.

Оператор, який читає один символ з клавіатури, виглядає наступним чином:

Приклад коду C++/CLI

```
char ch;
int x;
x = Console::Read();
ch = Convert::ToChar(x);
```

Приклад коду C#

```
char ch;
int x;
x = Console.Read();
ch = Convert.ToChar(x);
```

За допомогою функції Read() можете читати вхідні дані символ за символом. Функція Read() має тип int і зберігає код зчитаного символу в таблиці кодування. Потім прочитані символи необхідно проаналізувати і перетворити їх у відповідні значення. Для перетворення значення у тип char використовується функція ToChar() класу Convert.

Функція ReadKey() класу Console повертає натиснуту клавішу у вигляді об'єкта класу ConsoleKeyInfo, який представляє собою клас типу значення, визначений у просторі імен System. Нижче наведено оператор читання натиснутої клавіші.

Приклад коду C++/CLI

```
ConsoleKeyInfo keyPress = Console::ReadKey(true);
```

Приклад коду C#

```
ConsoleKeyInfo keyPress = Console.ReadKey(true);
```

Аргумент true функції ReadKey() пригнічує відображення натиснутої клавіші в командному рядку. Аргумент false (або відсутність аргументу) змушує функцію відображати символ натиснутої клавіші. Результат виконання функції зберігається в змінній keyPress. Щоб ідентифікувати символ, відповідний натиснутій клавіші (або клавішам), необхідно застосовувати вираз keyPress.KeyChar. Таким чином, щоб вивести повідомлення, яке показує символ натиснутої клавіші, скористайтеся наступним оператором.

Приклад коду C++/CLI

```
Console::WriteLine(L"Натиснута клавіша відповідає символу: {0}",
      keyPress.KeyChar);
```

Приклад коду C#

```
Console.WriteLine("Натиснута клавіша відповідає символу: {0}",
      keyPress.KeyChar);
```

Натиснута кнопка ідентифікується виразом `keyPress.Key`. Цей вираз посилається на значення з переліку символів, що ідентифікує натиснуту клавішу.

Хоча те, що в консольних програмах немає форматування введення, може здатися незручним під час вивчення, на практиці це обмеження несуттєве. Майже всі реальні програми, які вам доведеться писати, приймають введення через компоненти вікна, тому зазвичай немає необхідності читати дані з командного рядка.

Читання числових значень з командного рядка передбачає використання деяких засобів.

Якщо ви читаєте рядок, що містить цілочисельне значення з використанням функції `ReadLine()`, *функція* `Parse()` в класі `Int32` перетворює його автоматично в 32-бітове ціле. Ось як можна читати цілочисельне значення.

Приклад коду C++/CLI

```
Console::Write(L"Введіть ціле число:");
int value = Int32::Parse(Console::ReadLine());
Console::WriteLine(L"Ви ввели {0}", value);
```

Приклад коду C#

```
Console.Write("Введіть ціле число:");
int value = Int32.Parse(Console.ReadLine());
Console.WriteLine("Ви ввели {0}", value);
```

Перший оператор просто запрошує ввести необхідне значення, а другий читає введення. Рядок, який повертається функцією `ReadLine()`, передається у вигляді аргументу функції `Parse()`, що відноситься до класу `Int32`. Це перетворює рядок у 32-бітове ціле і зберігає результат у змінній `value`. Останній оператор виводить значення, щоб показати, що все пройшло правильно. Звичайно, якщо ви введете щось відмінне від цілого числа, виникне помилка.

Інші класи значень, що відповідають базовим «рідним» типам C++, також визначають функцію `Parse()`, тому, наприклад, коли захочете прочитати значення з плаваючою комою з клавіатури, передайте функції `Parse()` рядок, який повертає функція `ReadLine()`. Результатом буде значення типу `double`.

Приклад коду C#

```
Console.Write("Введіть дробове число:");
double value1 = Double.Parse(Console.ReadLine());
Console.WriteLine("Ви ввели {0}", value1);
```

Завдання 2.

Створити консольний додаток для реалізації наведених вище прикладів клавіатурного введення та форматованого виведення інформації.

ПРАКТИЧНА РОБОТА №2

2 Сортування та пошук даних в одномірному масиві консольних додатків C++/CLI та C#

2.1 Масиви

Розглянемо приклад використання масиву.

Приклад коду C++/CLI

```
#include "stdafx.h"
using namespace System;

int main(array<System::String ^> ^args)
{
	// Створити масив, який міститиме 50 значень типу double
	array<double>^ samples = gcnew array<double>(50);
	// Створити випадкові значення елементів
	Random^ generator = gcnew Random;
	for (int i = 0; i < samples->Length; i++)
		samples[i] = 100.0*generator->NextDouble();
	// Вивести вміст samples
	Console::WriteLine(L"Масив містить наступні значення:");
	for (int i = 0; i < samples->Length; i++)
	{
		Console::Write(L"{0,10:F2}", samples[i]);
		if ((i + 1) % 5 == 0)
			Console::WriteLine();
	}
	// Знайти максимальне значення
	double max(0);
	//Використовуємо цикл for each для перебору всіх елементів масиву
	for each(double sample in samples)
		if (max < sample)
			max = sample;
	Console::WriteLine(L"Максимальне значення в масиві дорівнює
		{0:F2}", max);
	return 0;
}
```

Приклад коду C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleAppCsh_1
{
	class Program
```

```
        {
            static void Main(string[] args)
            {
                Створити масив, який міститиме 50 значень типу double
                double[] samples = new double[50];
                // Створити випадкові значення елементів
                Random generator = new Random();
                for (int i = 0; i < samples.Length; i++)
                    samples[i] = 100.0 * generator.NextDouble();
                // Вивести вміст samples
                Console.WriteLine("Масив містить наступні значення:");
                for (int i = 0; i < samples.Length; i++)
                {
                    Console.Write("{0,10:F2}", samples[i]);
                    if ((i + 1) % 5 == 0)
                    Console.WriteLine();
                }
                // Знайти максимальне значення
                double max = 0;
                //Використовуємо цикл foreach для перебору всіх
                //елементів масиву
                foreach (double sample in samples)
                    if (max < sample)
                        max = sample;
                Console.WriteLine("Максимальне значення в масиві дорівнює
                    {0:F2}", max);
            }
        }
}
```

Типове виведення даної програми виглядатиме наступним чином:

```
Масив містить наступні значення:
     94,87     37,68     12,86     44,02     95,27
     27,29     24,43     70,93     87,90     57,73
     11,26     90,93     71,48     19,60     78,37
     85,04     34,81     17,36     48,42     70,81
     68,88     45,75     15,24     24,67     95,56
     96,99     40,96     71,64     76,24     36,98
     77,48     25,91     11,67     43,61     56,56
      8,04     20,27     44,23     74,99     15,14
      9,33     92,48     56,99      3,22     43,81
     72,51     92,15     27,52     12,40     26,37
Максимальне значення в масиві дорівнює 96,99
```

Спочатку створюється масив, здатний зберігати 50 елементів типу double.

Приклад коду C++/CLI

```
array<double>^ samples = gcnew array<double>(50);
```

Приклад коду C#

```
double[] samples = new double[50];
```

З рядків коду, які використовуються для створення масивів, можна помітити суттєві відмінності у синтаксисі між C++/CLI та C#.

Змінна samples типу array в коді C++/CLI повинна бути відстежуваним дескриптором, бо масиви створюються в динамічній пам'яті, контрольованій збирачем «сміття». З огляду на це, всі об'єкти, час життя яких відстежує збирач сміття, у C++/CLI створюються за допомогою оператора gcnew. Це зроблено для того, щоб відрізняти об'єкти мови C++, які створюються за допомогою оператора new. Мова C++/CLI, на відміну від C#, підтримує програмні конструкції мови C++, які можна використовувати безпосередньо в коді. Крім того, об'єкти які контролює збирач сміття, в мові C++/CLI, є дескрипторами і для їх оголошення використовується оператор ^.

Мова C# не підтримує напряму синтаксис C++ і тому немає таких синтаксичних обмежень, як мова C++/CLI. У зв'язку з цим оголошення масивів у C# виглядає значно простіше.

Наступні оператори заповнюють масив псевдовипадковими значеннями типу double.

Приклад коду C++/CLI

```
Random^ generator = gcnew Random;
for (int i = 0; i < samples->Length; i++)
     samples[i] = 100.0 * generator->NextDouble();
```

Приклад коду C#

```
// Створити випадкові значення елементів
Random generator = new Random();
for (int i = 0; i < samples.Length; i++)
     samples[i] = 100.0 * generator.NextDouble();
```

Перший оператор створює в динамічної пам'яті середовища CLR об'єкт типу Random. Об'єкт Random має функції створення псевдовипадкових значень. Тут в циклі використовується функція NextDouble(), яка повертає випадкове значення типу double, що лежить в межах від 0,0 до 1,0. Помноживши це значення на 100,0, отримаємо значення 0,0 - 100,0. Цикл for зберігає випадкове значення в кожному елементі масиву samples.

Об'єкт Random має також функцію Next(), яка повертає випадкове позитивне значення типу int. Якщо передати цілочисельний аргумент при виконанні функції Next(), то вона поверне випадкове позитивне значення, менше значення цього аргументу. Можна також передати два цілочисельних аргументи, що представляють мінімальне і максимальне значення, в межах яких має знаходитися випадкове число.

При детальному розгляді попередніх шматків коду можна також помітити, що на відміну від C++ та C++/CLI у мові C# не використовується оператор -> для

звернення до члену класу через його об'єкт, цей оператор замінює звичайний точковий оператор звернення до члену як об’єкту, так і класу

Наступний цикл виводить вміст масиву по п'ять елементів в рядку.

Приклад коду C++/CLI

```
Console::WriteLine(L"Масив містить наступні значення:");
for (int i = 0; i < samples->Length; i++)
{
      Console::Write(L"{0,10:F2}", samples[i]);
      if ((i + 1) % 5 == 0)
            Console::WriteLine();
}
```

У циклі значення кожного елемента виводиться в поле шириною 10 символів з двома десятковими розрядами. Покажчик ширини поля гарантує вирівнювання значень в стовпцях. Символ нового рядка виводиться щоразу, коли вираз (i + 1)%5 повертає значення 0, що відбувається після виведення значення кожного п'ятого елемента, тому виходить по п'ять значень в рядку.

І нарешті, максимальне значення елементів масиву визначається наступним чином.

Приклад коду C++/CLI

```
double max(0);
for each(double sample in samples)
      if (max < sample)
            max = sample;
```

Тут цикл for each застосовується тільки для демонстрації його можливостей. Цикл порівнює значення змінної max з кожним елементом по черзі, і коли виявляється, що елемент більше поточного значення змінної max, то його значення присвоюється змінній max, і в кінцевому підсумку змінна max містить максимальне з усіх значень елементів масиву.

Приклад коду для C# відрізняється лише формою запису ключового слова foreach, яке в даному випадку записується разом.

Можете використовувати тут цикл for, якщо хочете записати позицію індексу максимального елемента разом з його значенням.

Приклад коду C++/CLI

```
double max = 0;
int index = 0;
for (int i = 0; i < samples->Length; i++)
      if (max < samples[i])
      {
            max = samples[i];
            index = i;
```

```
}
```

Завдання 1.

Створити консольний додаток для реалізації наведеного вище прикладу використання масиву.

2.2 Сортування одномірних масивів

Завдання 2.

Запрограмувати консольний додаток, який створює масив імен, зберігає вагу кожної людини у відповідному елементі другого масиву, а потім сортує обидва масиви в одній операції.

Код програмного модуля для реалізації поставленої задачі буде наступний.

Приклад коду C++/CLI

```
// Сортування масивів за ключем (імена) та пов'язаного з ним масиву вага
#include "stdafx.h"

using namespace System;

int main(array<System::String^> ^args)
{
  array<String^>^ імена ={"Захар", "Орися", "Остап", "Росава", "Назар",
     "Одарка", "Нестор"};
  array<int>^ вага = { 87, 58, 83, 66, 74, 61, 78 };
  Array::Sort(імена, вага);                    // Сортувати масиви
  for each(String^ name in імена)                 // Вивести імена
     Console::Write(L"{0, 10}", name);
  Console::WriteLine();
  for each(int weight in вага)                 // Вивести вагу
     Console::Write(L"{0, 10}", weight);
  Console::WriteLine();
  return 0;
}
```

Приклад коду C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleAppCsh_1
{
     class Program
     {
          static void Main(string[] args)
          {
               string[] імена = {"Захар", "Орися", "Остап", "Росава",
```

```
                    "Назар", "Одарка", "Нестор"};
                int[] вага = { 87, 58, 83, 66, 74, 61, 78 };
                Array.Sort(імена, вага); // Сортувати масиви
                foreach(string name in імена) // Вивести імена
                    Console.Write("{0, 10}", name);
                Console.WriteLine();
                foreach(int weight in вага) // Вивести вагу
                    Console.Write("{0, 10}", weight);
                Console.WriteLine();
            }
        }
}
```

Виведення даної програми буде наступним.

```
    Захар     Назар     Нестор     Одарка      Орися      Остап     Росава
       87        74         78         61         58         83         66
```

Значення масиву вага відповідають вазі персон в тих же позиціях індексу, що і їх імена в масиві імена. Викликана тут функція Sort() сортує обидва масиви, використовуючи перший з них (імена) як ключ сортування, що визначає порядок обох масивів. З результату роботи програми видно, що після сортування імені кожної персони відповідає її вага з другого масиву.

2.3 Пошук в одномірному масиві

Завдання 3.

Розробити консольний додаток, який створює два пов'язаних масиви (масив імен та масив відповідних ваг в кілограмах), а також масив, що містить імена людей, вагу яких потрібно дізнатися.

Розглянемо програмний код додатку для вирішення поставленої задачі.

Приклад коду C++/CLI

```
// Пошук в масивах

#include "stdafx.h"

using namespace System;

int main(array<System::String ^> ^args)
{
    array<String^>^ імена = { "Захар", "Орися", "Остап", "Росава",
        "Назар", "Одарка", "Нестор", "Богуслава", "Данило", "Тарас" };
    array<int>^ вага = { 87, 58, 83, 66, 74, 61, 78, 57, 85, 72 };
    array<String^>^ требаЗнайти = { "Орися", "Назар", "Богуслава",
        "Кирило" };
    Array::Sort(імена, вага);       // Відсортувати масиви
    int порядковийНомер(0);         // Зберігає результуюче значення
    for each(String^ name in требаЗнайти)   // Пошук ваги за іменем
    {
```

```
            // Пошук імені в масиві імен
            порядковийНомер = Array::BinarySearch(імена, name);
            if (порядковийНомер <0)  // Перевірка результату пошуку
                  Console::WriteLine(L"{0} не знайдений.", name);
            else
                  Console::WriteLine(L"{0} має вагу {1} кг.", name,
                        вага[порядковийНомер]);
      }
      return 0;
}
```

Приклад коду C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleAppCsh_2
{
      class Program
      {
            static void Main(string[] args)
            {
                  string[] імена = { "Захар", "Орися", "Остап", "Росава",
            "Назар", "Одарка", "Нестор", "Богуслава", "Данило", "Тарас" };
                  int[] вага = { 87, 58, 83, 66, 74, 61, 78, 57, 85, 72 };
                  string[] требаЗнайти = { "Орися", "Назар", "Богуслава",
            "Кирило" };
                  Array.Sort(імена, вага);// Відсортувати масиви
                  int порядковийНомер = 0;// Зберігає результуюче значення
                  // Пошук ваги за іменем
                  foreach(string name in требаЗнайти)
                  {
                        // Пошук імені в масиві імен
                        порядковийНомер = Array.BinarySearch(імена, name);
                        // Перевірка результату пошуку
                        if (порядковийНомер < 0)
                              Console.WriteLine("{0} не знайдений.", name);
                        else
                              Console.WriteLine("{0} має вагу {1} кг.",
                        name, вага[порядковийНомер]);
                  }
            }
      }
}
```

Результат виконання цієї програми наведений нижче

```
Орися має вагу 58 кг.
Назар має вагу 74 кг.
Богуслава має вагу 57 кг.
```

```
Кирило не знайдений.
```

Створюються два пов'язаних масиви (масив імен та масив відповідних ваг в кілограмах). Крім того, створюється масив требаЗнайти, що містить імена людей, вагу яких потрібно дізнатися.

Масиви імена і вага упорядковано з використанням масиву імена як ключа сортування. Потім в циклі for each (в C# foreach) виконується пошук в масиві імена кожного з імен, що містяться в масиві требаЗнайти. Змінна циклу name по черзі отримує значення кожного з імен масиву требаЗнайти. У циклі здійснюється пошук поточного імені за допомогою оператора

Приклад коду C++/CLI

```
порядковийНомер = Array::BinarySearch(імена, name);
```

Приклад коду C#

```
порядковийНомер = Array.BinarySearch(імена, name);
```

В результаті повертається індекс елемента з масиву імена, який дорівнює name, або негативне значення, якщо такого не знайдено. Результат пошуку перевіряється і виводиться відповідне повідомлення.

Приклад коду C++/CLI

```
if (порядковийНомер <0)  // Перевірка результату пошуку
     Console::WriteLine(L"{0} не знайдений.", name);
else
     Console::WriteLine(L"{0} має вагу {1} кг.", name,
          вага[порядковийНомер]);
```

Приклад коду C#

```
if (порядковийНомер < 0) // Перевірка результату пошуку
     Console.WriteLine("{0} не знайдений.", name);
else
     Console.WriteLine("{0} має вагу {1} кг.", name,
          вага[порядковийНомер]);
```

Оскільки порядок елементів масиву вага визначається порядком елементів масиву імена, для отримання результату з масиву вага можна звернутися до елементу цього масиву за індексом, знайденим при пошуку змінної name в масиві імена. З виведення програми видно, що значення "Кирило" не було знайдено в масиві імена.

Коли бінарний пошук завершується невдало, повертається не просто якесь випадкове негативне значення. Насправді це значення являє собою бітове доповнення позиції індексу першого елемента, який більше шуканого значення. Знаючи це, скористайтеся функцією BinarySearch(), щоб знайти місце в

масиві, куди слід було б вставити новий об'єкт, не порушивши при цьому порядок сортування елементів. Припустимо, ви хочете вставити значення "Кирило" в масив імена. Позицію індексу, куди він повинен бути вставлений, можна знайти за допомогою наступних операторів.

Приклад коду C++/CLI

```
array<String^>^ імена = { "Захар", "Орися", "Остап", "Росава", "Назар",
      "Одарка", "Нестор", "Богуслава", "Данило", "Тарас" };
Array::Sort(імена); // Сортувати масив
String^ name = L"Кирило";
int порядковийНомер = Array::BinarySearch(імена, name);
if (порядковийНомер <0) // Якщо результат негативний,
      // Перекинути біти, щоб отримати індекс вставки
      порядковийНомер = ~ порядковийНомер;
```

Якщо результат пошуку негативний, заміна всіх бітів на протилежне значення (0 на 1 і навпаки) дає позицію індексу, куди слід вставити нове ім'я. Якщо результат позитивний, значить, нове ім'я ідентично імені в позиції з індексом, рівним результату, тому це значення можна використовувати безпосередньо.

Тепер можна скопіювати масив імена в новий масив, розмір якого на один елемент більше, і застосувати обчислену позицію для вставки імені у відповідне місце.

Приклад коду C++/CLI

```
array<String^>^ новіІмена = gcnew array <String^>(імена->Length + 1);
// Копіювати елементи з масиву імена в масив новіІмена
for (int i = 0; i < порядковийНомер; i++)
      новіІмена [i] = імена[i];
новіІмена[порядковийНомер] = name; // Копіювати новий елемент
if (порядковийНомер < імена->Length) // Якщо ще залишилися елементи
      for (int i = порядковийНомер; i < імена->Length; i++)
            новіІмена [i + 1] = імена[i]; // Копіювати їх в newNames
```

Це створює новий масив довжиною на один елемент більше, ніж старий. Потім копіюються елементи зі старого масиву в новий масив - від початку до позиції індексу порядковийНомер - 1. Після цього копіюється нове ім'я, а за ним - решта елементів зі старого масиву. Щоб відкинути старий масив, можна написати просто.

Приклад коду C++/CLI

```
names = nullptr;
```

Для виведення значень створеного масиву новіІмена використаємо цикл for each.

Приклад коду C++/CLI

```
for each(String^ наДрук in новіІмена)
      Console::Write(L"{0,8}", наДрук);
```

Результатом буде наступне виведення.

```
Богуслава  Данило   Захар  Кирило   Назар  Нестор  Одарка   Орися   Остап
Росава   Тарас
```

Повний текст прикладу наведений нижче.

Приклад коду C++/CLI

```
// Вставка значення у відсортований масив

#include "stdafx.h"

using namespace System;

int main(array<System::String ^> ^args)
{
      array<String^>^ імена = { "Захар", "Орися", "Остап", "Росава",
"Назар", "Одарка", "Нестор", "Богуслава", "Данило", "Тарас" };
      Array::Sort(імена); // Сортувати масив
      String^ name = L"Кирило";
      int порядковийНомер = Array::BinarySearch(імена, name);
      if (порядковийНомер <0) // Якщо результат негативний,
            // Перекинути біти, щоб отримати індекс вставки
            порядковийНомер = ~ порядковийНомер;
      array<String^>^ новіІмена = gcnew array <String^>(імена->Length
            + 1);
      // Копіювати елементи з імена в новіІмена
      for (int i = 0; i < порядковийНомер; i++)
            новіІмена[i] = імена[i];
      новіІмена[порядковийНомер] = name; // Копіювати новий елемент
      if (порядковийНомер < імена->Length) // Якщо ще залишилися елементи
            for (int i = порядковийНомер; i < імена->Length; i++)
                  новіІмена[i + 1] = імена[i]; // Копіювати їх в новіІмена
      імена = nullptr;
      for each(String^ наДрук in новіІмена)
            Console::Write(L"{0,10}", name);
      return 0;
}
```

Приклад коду C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ConsoleAppCsh_2
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] імена = { "Захар", "Орися", "Остап", "Росава",
        "Назар", "Одарка", "Нестор", "Богуслава", "Данило", "Тарас" };
            Array.Sort(імена); // Сортувати масив
            string name = "Кирило";
            int порядковийНомер = Array.BinarySearch(імена, name);
            if (порядковийНомер < 0) // Якщо результат негативний,
                // Перекинути біти, щоб отримати індекс вставки
                порядковийНомер = ~порядковийНомер;
            string[] новіІмена = new string[імена.Length + 1];
            // Копіювати елементи з масиву імена в масив новіІмена
            for (int i = 0; i < порядковийНомер; i++)
                новіІмена[i] = імена[i];
            // Копіювати новий елемент
            новіІмена[порядковийНомер] = name;
            // Якщо ще залишилися елементи масиву імена
            if (порядковийНомер < імена.Length)
                for (int i = порядковийНомер; i < імена.Length;
                    i++)
                    // Копіювати їх в newNames
                    новіІмена[i + 1] = імена[i];
            імена = null;
            // Друк масиву імен
            foreach (string наДрук in новіІмена)
            {
                Console.Write("{0,8}", наДрук);
            }
        }
    }
}
```

ПРАКТИЧНА РОБОТА №3

3 Робота з багатомірними масивами консольних додатків C++/CLI та C#

3.1 Багаторангові масиви

Завдання 1.

Необхідно створити консольний додаток, який створює таблицю множення 12x12 в дворанговому масиві.

Спробуємо скористатися багатовимірним масивом в наступному прикладі.

За допомогою наведених нижче операторів створюється двовимірний масив.

Приклад коду C++/CLI

```
const int розмір (12);
array<int, 2>^ таблицяМноження (gcnew array<int, 2>(розмір, розмір));
```

Приклад коду C#

```
int розмір = 12;
int[,] таблицяМноження = new int[розмір, розмір];
```

Перший рядок коду C++/CLI визначає константне цілочисельне значення, яке задає кількість елементів в кожному вимірі масиву. Другий рядок визначає масив рангу 2, що складається з 12 рядків і 12 стовпців. У ньому розміщуються значення добутків таблиці множення розміром 12x12.

Значення елементів масиву формуються у вкладеному циклі.

Приклад коду C++/CLI

```
for (int i = 0; i < розмір; i++)
     for (int j = 0; j < розмір; j++)
          таблицяМноження [i, j] = (i + 1)*(j + 1);
```

Приклад коду C#

```
for (int i = 0; i < розмір; i++)
     for (int j = 0; j < розмір; j++)
          таблицяМноження[i, j] = (i + 1) * (j + 1);
```

Зовнішній цикл перебирає рядки, внутрішній - стовпці. Значення кожного елемента дорівнює добутку індексу рядка на індекс стовпця після того, як кожен з них збільшений на 1. Решта коду функції `main()` займається виключно виведенням на екран. Після виведення заголовка таблиці формується лінія, що позначає шапку таблиці наступним чином.

Приклад коду C++/CLI

```
for (int i = 0; i <= розмір; i++)// Вивести горизонтальну роздільну лінію
      Console::Write(L"_____");
Console::WriteLine();           // Вивести новий рядок
```

Приклад коду C#

```
for (int i = 0; i <= розмір; i++)// Вивести горизонтальну роздільну лінію
      Console.Write("_____");
Console.WriteLine();           // Вивести новий рядок
```

Кожна ітерація циклу виводить п'ять символів підкреслення. Зверніть увагу на те, що верхню межу лічильника циклу визначено як значення розмір включно, тобто виводиться 13 наборів знаків підкреслення, щоб на додаток до 12 стовпців значень сформувати початковий стовпець заголовків рядків.

У наступному циклі виводиться рядок заголовків стовпців таблиці.

Приклад коду C++/CLI

```
Console::Write(L"    |");             // Вивести верхній рядок таблиці
for (int i = 1; i <= розмір; i++)
      Console::Write(L"{0,3} |", i);
Console::WriteLine();           // Вивести новий рядок
```

Приклад коду C#

```
Console.Write("    |");       // Вивести верхній рядок таблиці
for (int i = 1; i <= розмір; i++)
      Console.Write("{0,3} |", i);
Console.WriteLine();           // Вивести новий рядок
```

Спочатку перед виведенням заголовків стовпців окремо виводиться група пробілів, тому що це - спеціальний випадок без вихідного значення; дані пробіли знаходяться над початковим стовпцем заголовків рядків. Кожна мітка стовпця виводиться в циклі, потім іде символ нового рядка, готуючи наступний рядок виводу.

Рядки значень виводяться за допомогою вкладеного циклу.

Приклад коду C++/CLI

```
for (int i = 0; i < розмір; i++)      // Вивести інші рядки
{
      Console::Write(L"{0,3} |", i + 1);
      for (int j = 0; j< розмір; j++)
            Console::Write(L"{0,3} |", таблицяМноження[i, j]);
      Console::WriteLine();      // Вивести новий рядок
}
```

Приклад коду C#

```
for (int i = 0; i < розмір; i++)      // Вивести інші рядки
{
      Console.Write("{0,3} |", i + 1);
      for (int j = 0; j < розмір; j++)
            Console.Write("{0,3} |", таблицяМноження[i, j]);
      Console.WriteLine();       // Вивести новий рядок
}
```

Зовнішній цикл перебирає рядки, а код в внутрішньому циклі формує виведення всього рядка, включаючи його мітку в лівій частині. Внутрішній цикл виводить значення елементів масиву таблицяМноження, відповідних i-му рядку, розділяючи їх вертикальними лініями.

Подальший код виводить нижню лінію таблиці.

Даний приклад формує наступне виведення.

```
Таблиця множення розміром 12:
_____________________________________________________________________
    |  1 |  2 |  3 |  4 |  5 |  6 |  7 |  8 |  9 | 10 | 11 | 12 |
____|____|____|____|____|____|____|____|____|____|____|____|____|
  1 |  1 |  2 |  3 |  4 |  5 |  6 |  7 |  8 |  9 | 10 | 11 | 12 |
  2 |  2 |  4 |  6 |  8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 |
  3 |  3 |  6 |  9 | 12 | 15 | 18 | 21 | 24 | 27 | 30 | 33 | 36 |
  4 |  4 |  8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 | 44 | 48 |
  5 |  5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 55 | 60 |
  6 |  6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 | 60 | 66 | 72 |
  7 |  7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 | 70 | 77 | 84 |
  8 |  8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 |
  9 |  9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 | 90 | 99 |108 |
 10 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 |100 |110 |120 |
 11 | 11 | 22 | 33 | 44 | 55 | 66 | 77 | 88 | 99 |110 |121 |132 |
 12 | 12 | 24 | 36 | 48 | 60 | 72 | 84 | 96 |108 |120 |132 |144 |
_____________________________________________________________________
```

Текст програмного коду даного модуля наведений в лістингу 3.1 (C++/CLI) та лістингу 3.2 (C#).

3.2 Зубчасті масиви

Завдання 2.

Необхідно створити консольний додаток, який створює зубчастий масив (масив масивів) з речень, які користувач увів з клавіатури.

Розмірність зовнішнього масиву, який міститиме масиви типу String у C++/CLI або string у C#, визначатимемо через запит до користувача. Кількість внутрішніх масивів має відповідати кількості введених користувачем речень. Кожне окреме речення, введення якого з клавіатури завершене натисканням клавіши Enter, має сформувати окремий внутрішній масив. Розмірності внутрішніх масивів мають визначатися з кількості слів кожного введеного

речення. Кожне окреме слово введеного речення має зберігатися як елемент певного внутрішнього масиву. Значення елементів масиву виводитимемо в консольному вікні у форматованому та чітко структурованому вигляді.

Щоб створити масив масивів потрібно для початку знати розмірність зовнішнього масиву. Для цього робимо запит до користувача:

Приклад коду C++/CLI

```
Console::WriteLine(L"Введіть розмірність зовнішнього масиву");
size_t n = Int32::Parse(Console::ReadLine());
```

Приклад коду C#

```
Console.WriteLine("Введіть розмірність зовнішнього масиву");
int n = Int32.Parse(Console.ReadLine());
```

Розмірність масиву зберігаємо у змінну n типу `size_t` для C++/CLI або типу `int` для C#, але дані, що передаються функцією `ReadLine()` мають рядковий тип. Права і ліва частина виразу повинні мати однаковий тип, або права частина виразу має бути такого типу, який допускає автоматичне перетворення в тип лівої частини виразу. В нашому випадку неможливо перетворити автоматично рядковий тип у цілочисельний. Тому використовуємо функцію перетворення типів `Parse` з класу цілочисельних значень `Int32`.

Після цього можемо оголосити і визначити зубчастий масив:

Приклад коду C++/CLI

```
array<array<String^>^>^ масивРеченнь = gcnew array<array<String^>^>(n);
```

Приклад коду C#

```
string[][] масивРеченнь = new string[n][];
```

Ми оголосили в C++/CLI масив, елементами якого будуть масиви, що міститимуть рядкові елементи типу `String^`. Це і є масив масивів або зубчастий масив, тому що елементами масиву є інші масиви різної довжини.

Як можна побачити з прикладу коду, у C# оголошення зубчастого масиву виглядає значно простіше.

Після цього організовуємо роботу програми у циклі `for`, кількість повторень якого відповідає кількості елементів зовнішнього масиву

Приклад коду C++/CLI

```
for (size_t i = 0; i < n; i++)
```

Приклад коду C#

```
for (int i = 0; i < n; i++)
```

У C# відсутній тип `size_t`, тому ми використовуємо для змінної циклу тип `int`. В середині циклу оголошуємо змінну, куди будемо запам'ятовувати речення, яке ввів користувач на кожній ітерації циклу.

Приклад коду C++/CLI

```
//Зчитуємо введене речення
String^ речення = Console::ReadLine();
```

Приклад коду C#

```
string речення = Console.ReadLine();
```

Слова у масив будемо запам'ятовувати без знаків пунктуації. Кількість слів у реченні будемо визначати за кількістю пробілів між словами. Тому потрібно провести ряд операцій, щоб прибрати зайві пробіли та символи.

Слова у масив будемо запам'ятовувати лише малими літерами. Для цього перетворимо всі символи речення на малі функцією ToLower():

Приклад коду C++/CLI

```
//Змінюємо всі літери речення на маленькі
речення = речення->ToLower();
```

Приклад коду C#

```
речення = речення.ToLower();
```

Тепер нам потрібно видалити знаки пунктуації. Якщо користувач пропустив пробіл після знаку пунктуації, тоді, якщо видалити знак пунктуації, два слова об'єднаються в одне. Щоб уникнути цієї помилки ми будемо не видаляти знаки пунктуації, а замінювати їх на пробіли. Перебір символів рядкового типу `String^` (для C# типу `string`) можна організувати у циклі, бо змінна типу `String^` являє собою масив символів типу `char` або `wchar_t` (у C# тип `wchar_t` відсутній, тому використовуємо тип `char`) і до них можна звертатися за допомогою індексу – порядкового номеру символу у ряду.

Але є простіший спосіб з використанням *регулярних виразів (regular expressions)* для текстових даних. Щоб скористатися функціями для регулярних виразів, на початку програми потрібно підключити область імен регулярних виразів:

Приклад коду C++/CLI

```
using namespace System::Text::RegularExpressions;
```

Приклад коду C#

```
using System.Text.RegularExpressions;
```

Тепер можемо за допомогою однієї функції, без використання циклічного перебору усіх символів речення в програмі, замінити всі символи з заданого набору символів на вказаний підрядок тексту. Для цього використовується функція `Replace()` класу `Regex` з регулярних виразів. В нашому випадку будемо замінювати кожен символ з вказаного набору на пробіл:

Приклад коду C++/CLI

```
//Замінюємо в реченні знаки пунктуації на пробіл за допомогою бібліотеки регулярних виразів
речення = Regex::Replace(речення, "[-.?!)(,:;]", " ");
```

Приклад коду C#

```
речення = Regex.Replace(речення, "[-.?!)(,:;]", " ");
```

Після заміни знаків пунктуації на пробіли, якщо після знаку стояв вже пробіл, то тепер їх буде по два між словами. А якщо між словами користувач вже до того помилково увів два пробіли, то після заміни знаку пунктуації пробілів може стати три. Так як кількість слів ми можемо визначити саме за пробілами між ними, то нам потрібно, щоб між кожними двома словами був один і лише один пробіл. У зв'язку з цим виникає задача видалення зайвих пробілів між словами. Виконати це завдання ми можемо за допомогою функції Replace() класу `String`, яка замінює один підрядок в тексті на інший.

Приклад коду C++/CLI

```
//Замінюємо в реченні потрійні пробіли на одинарні, якщо такі є
речення = речення->Replace("   ", " ");
//Замінюємо в реченні подвійні пробіли на одинарні, якщо такі є
речення = речення->Replace("  ", " ");
```

Приклад коду C#

```
//Замінюємо в реченні потрійні пробіли на одинарні, якщо такі є
речення = речення.Replace("   ", " ");
//Замінюємо в реченні подвійні пробіли на одинарні, якщо такі є
речення = речення.Replace("  ", " ");
```

Видалимо зайві пробіли на початку та в кінці речення, якщо користувач помилково їх увів, та пробіл, яким ми замінили останній знак пунктуації у реченні. Для цього скористаємося функцією `Trim()` класу `String`:

Приклад коду C++/CLI

```
//Відкидаємо пробіли на початку та в кінці речення
речення = речення->Trim();
```

Приклад коду C#

```
речення = речення.Trim();
```

Тепер безпосередньо потрібно підрахувати кількість пробілів у реченні, за якими і визначимо кількість слів. Для цього знову використаємо бібліотеку регулярних виразів.

Приклад коду C++/CLI

```
//Рахуємо кількість пробілів, що залишились
size_t count(0);
//Створюємо змінну типу Regex і вказуємо рядок,
//який будемо шукати (пробіл)
Regex^ пробіл = gcnew Regex(" ");
//Створюємо змінну типу MatchCollection і присвоюємо їй результати
//пошуку співпадінь
MatchCollection^ співпадіння = пробіл->Matches(речення);
//Запам'ятовуємо в змінну count кількість знайдених пробілів
в реченні + 1
//count = співпадіння->Count + 1;
```

Приклад коду C#

```
//Рахуємо кількість пробілів, що залишились
int count = 0;
//Створюємо змінну типу Regex і вказуємо рядок,
//який будемо шукати (пробіл)
Regex пробіл = new Regex(" ");
//Створюємо змінну типу MatchCollection і присвоюємо їй результати
//пошуку співпадінь
MatchCollection співпадіння = пробіл.Matches(речення);
//Запам'ятовуємо в змінну count кількість знайдених пробілів
в реченні + 1
count = співпадіння.Count + 1;
```

Тут ми створили:

- змінну `count` типу `size_t` (в C# типу `int`), в яку будемо запам’ятовувати знайдену кількість пробілів;

- змінну `пробіл` типу `Regex`, якій присвоїли підрядок, що потрібно знайти (в нашому випадку це пробіл);
- змінну `співпадіння` типу `MatchCollection`, якій присвоюємо результат пошуку співпадінь вказаного підрядку у заданому тексті за допомогою функції `Matches()` класу `Regex`.

Кількість співпадінь дізнаємося з властивості `Count` змінної `співпадіння` класу `MatchCollection`.

Тепер можемо створити внутрішній масив для збереження слів поточного речення. Розмірність цього масиву дорівнює кількості знайдених пробілів плюс 1, тому що після останнього слова в реченні пробілу немає.

Приклад коду C++/CLI

```
//Створюємо масив розмірність якого дорівнює кількості слів у реченні
масивРечень[i] = gcnew array<String^>(count);
```

Приклад коду C#

```
масивРечень[i] = new string[count];
```

В подальшому будемо перебирати у циклі всі символи речення окрім пробілів і формувати їх у слова для запису у внутрішній масив.

Приклад коду C++/CLI

```
String^ слово = "", ^ останнєСлово = "";
for each (wchar_t літера in речення)
{
	if (літера != ' ')
	{
		слово += літера;
		останнєСлово = слово;
	}
	else
	{
		масивРечень[i][index] = слово;
		index++;
		слово = "";
	}
}
//Оскільки після останнього слова у реченні немає пробілу,
//записуємо його окремо після роботи циклу
масивРечень[i][index] = останнєСлово;
```

Приклад коду C#

```
string слово = "", останнєСлово = "";
foreach(char літера in речення)
```

```
{
      if (літера != ' ')
      {
            слово += літера;
            останнєСлово = слово;
      }
      else
      {
            масивРечень[i][index] = слово;
            index++;
            слово = "";
      }
}
//Оскільки після останнього слова у реченні немає пробілу,
//записуємо його окремо після роботи циклу
масивРечень[i][index] = останнєСлово;
```

До кожного символу речення звертаємося у циклі `for each` (у C# цикл `foreach`) через ідентифікатор `літера` типу `wchar_t` (у C# типу `char`). Якщо цей символ не пробіл, додаємо його до вже існуючих символів у змінній `слово` типу `String^` (у C# типу `string`). Якщо символ виявляється пробілом, записуємо змінну `слово` як новий елемент внутрішнього масиву `масивРечень[i]`:

```
масивРечень[i][index] = слово;
```

Після цього збільшуємо змінну `index` (яка відповідає порядковому номеру елементу внутрішнього масиву) на одиницю та витираємо всі символи в змінній `слово`:

```
index++;
слово = "";
```

Оскільки після останнього слова в реченні немає пробілу, ми не потрапимо у блок `else` умовного оператора `if`, де додається новий елемент в масив, і, відповідно, останнє слово речення не буде записане до масиву. Щоб врахувати останні слова речень, ми створили змінну `останнєСлово`:

Приклад коду C++/CLI

```
String^ останнєСлово = "";
```

Приклад коду C#

```
string останнєСлово = "";
```

В дану змінну кожного разу записується поточне значення змінної `слово`, якщо символ не є пробілом:

```
if (літера != ' ')
{
	слово += літера;
	останнєСлово = слово;
}
```

По завершенні роботи циклу for each (foreach у C#) в змінній останнєСлово зберігатиметься останнє слово речення, яке не було додане до масиву. Тепер нам лише потрібно додати це останнє слово у масив:

```
//Оскільки після останнього слова у реченні немає пробілу,
//записуємо його окремо після роботи циклу
масивРечень[i][index] = останнєСлово;
```

Надалі виводимо на екран речення після виконання всіх змін та кількість слів у даному реченні:

Приклад коду C++/CLI

```
Console::WriteLine(L"\nВиправлене речення:");
Console::WriteLine(речення);
Console::WriteLine(L"Кількість слів у реченні = {0}\n",count);
```

Приклад коду C#

```
Console.WriteLine("\nВиправлене речення:");
Console.WriteLine(речення);
Console.WriteLine("Кількість слів у реченні = {0}\n", count);
```

Тепер залишилось лише вивести сформований масив масивів на екран. Це потрібно робити за допомогою двох циклів for, один з яких вкладений в інший.

Приклад коду C++/CLI

```
//Виводимо результуючий масив масивів слів за допомогою індексації
Console::WriteLine(L"\nСформований масив з речень");
for (size_t i = 0; i < масивРечень->Length; i++)
{
	for (size_t j = 0; j < масивРечень[i]->Length; j++)
	{
		Console::Write(L" S({0},{1})={2}\t", i, j,
			МасивРечень[i][j]);
	}
	Console::WriteLine();
}
```

Приклад коду C#

```
//Виводимо результуючий масив масивів слів за допомогою індексації
Console.WriteLine("\nСформований масив з речень");
```

```
for (int i = 0; i<масивРечень.Length; i++)
{
      for (int j = 0; j < масивРечень[i].Length; j++)
      {
            Console.Write(" S({0},{1})={2}\t", i, j, масивРечень[i][j]);
      }
      Console.WriteLine();
}
```

Для порівняння коду ми також додали виведення елементів зубчастого масиву за допомогою циклів `for each`. В даному циклі звернення до елементів набору перераховуваних значень (в нашому випадку елементів масиву) здійснюється не через індекс, а через *змінну-ідентифікатор*.

Приклад коду C++/CLI

```
Console::WriteLine();
//Виводимо результуючий масив масивів слів за допомогою циклу for each
for each (array<String^>^ речення in масивРечень)
{
      for each (String^ слово in речення)
      {
            Console::Write(L" {0}\t", слово);
      }
      Console::WriteLine();
}
```

Приклад коду C#

```
Console.WriteLine();
//Виводимо результуючий масив масивів слів за допомогою циклу foreach
foreach(string[] речення in масивРечень)
{
      foreach(string слово in речення)
      {
            Console.Write(" {0}\t", слово);
      }
      Console.WriteLine();
}
```

При застосуванні циклу `for each` потрібно пам'ятати, що ідентифікатор не дозволяє змінювати елементи масиву, а лише дає можливість зчитувати їх значення. Для зміни елементів масиву у C++/CLI в циклі `for each` ідентифікатор має бути *відстежуваним посиланням*, яке позначається символом %. Для можливості зміни елементів масиву в вище наведеному циклі `for each` потрібно внести наступну змінну у заголовок внутрішнього циклу:

Приклад коду C++/CLI

```
for each (String^ %слово in речення)
```

В мові C# можливість змінювати елементи за допомогою циклу foreach відсутня. Даний цикл може використовуватись лише для пошуку, перегляду чи виведення елементів. Це пов'язане з відсутністю у C# поняття відстежуваних дескрипторів та відстежуваних посилань.

Результатом роботи даної програми буде наступний текст в консольному вікні.

```
Введіть розмірність зовнішнього масиву
4
Добрим словом мур проб'єш, а лихим і в двері не ввійдеш.

Виправлене речення:
добрим словом мур проб'єш а лихим і в двері не ввійдеш
Кількість слів у реченні = 11

Хто мови своєї цурається, хай сам себе стидається.

Виправлене речення:
хто мови своєї цурається хай сам себе стидається
Кількість слів у реченні = 8

Птицю пізнати по пір'ю, а людину по мові.

Виправлене речення:
птицю пізнати по пір'ю а людину по мові
Кількість слів у реченні = 8

Рідна мова - не полова: її за вітром не розвієш.

Виправлене речення:
рідна мова не полова її за вітром не розвієш
Кількість слів у реченні = 9


Сформований масив з речень
 S(0,0)=добрим   S(0,1)=словом   S(0,2)=мур      S(0,3)=проб'єш  S(0,4)=а
S(0,5)=лихим    S(0,6)=і        S(0,7)=в        S(0,8)=двері    S(0,9)=не
S(0,10)=ввійдеш
 S(1,0)=хто          S(1,1)=мови         S(1,2)=своєї      S(1,3)=цурається
S(1,4)=хай      S(1,5)=сам      S(1,6)=себе       S(1,7)=стидається
 S(2,0)=птицю    S(2,1)=пізнати  S(2,2)=по       S(2,3)=пір'ю    S(2,4)=а
S(2,5)=людину   S(2,6)=по       S(2,7)=мові
```

```
 S(3,0)=рідна    S(3,1)=мова     S(3,2)=не       S(3,3)=полова   S(3,4)=її
S(3,5)=за       S(3,6)=вітром   S(3,7)=не       S(3,8)=розвієш

 добрим  словом  мур     проб'єш          а      лихим   і       в
двері   не      ввійдеш
 хто     мови    своєї   цурається       хай     сам     себе    стидається
 птицю   пізнати          по      пір'ю   а       людину  по      мові
 рідна   мова    не      полова  її      за      вітром  не      розвієш
```

В таблиці кодування Windows для консольних додатків в українській мові відсутні літери «ґ» та «і». Літера ґ зустрічається рідше, тому проблему з нею виявити складніше, а от літера «і» зустрічається в багатьох словах і в консолі вона буде замінена на символ «?». Це призведе до невірної роботи програми, тому що ми символ «?» замінюємо на пробіл, як знак пунктуації. Через це слова з літерою «і» розіб'ються на кілька слів. Щоб таких помилок не виникло, при введенні речень українською мовою літеру «і» потрібно замінювати на латинську літеру «`i`».

Повний текст програмного коду наведений в лістингу 3.3 (C++/CLI) та лістингу 3.4 (C#).

Програмний код

Лістинг 3.1

Приклад коду C++/CLI

```
// Використання двомірного масиву

#include "stdafx.h"

using namespace System;

int main(array<System::String ^> ^args)
{
	const int розмір (12);
	array<int, 2>^ таблицяМноження(gcnew array<int, 2>(розмір,
		розмір));
	for (int i = 0; i < розмір; i++)
		for (int j = 0; j < розмір; j++)
			таблицяМноження[i, j] = (i + 1)*(j + 1);
	Console::WriteLine(L"Таблиця множення розміром {0}:", розмір);
	// Вивести горизонтальну роздільну лінію
	for (int i = 0; i <= розмір; i++)
		Console::Write(L"_____");
	Console::WriteLine();                    // Вивести новий рядок
	Console::Write(L"    |");          // Вивести верхній рядок таблиці
	for (int i = 1; i <= розмір; i++)
		Console::Write(L"{0,3} |", i);
	Console::WriteLine();                    // Вивести новий рядок
	// Вивести горизонтальну роздільну лінію з вертикальними рисками
	for (int i = 0; i <= розмір; i++)
		Console::Write(L"____|");
	Console::WriteLine();                              // Вивести новий рядок
	for (int i = 0; i< розмір; i++)          // Вивести інші рядки
	{
		Console::Write(L"{0,3} |", i + 1);
		for (int j = 0; j< розмір; j++)
			Console::Write(L"{0,3} |", таблицяМноження[i, j]);
		Console::WriteLine();                // Вивести новий рядок
	}
	// Вивести горизонтальну роздільну лінію
	for (int i = 0; i <= розмір; i++)
		Console::Write(L"_____");
	Console::WriteLine();                    // Вивести новий рядок
	return 0;
}
```

Лістинг 3.2

Приклад коду C#

```
// Використання двомірного масиву

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleAppCsh_3
{
      class Program
      {
            static void Main(string[] args)
            {
                  int розмір = 12;
                  int[,] таблицяМноження = new int[розмір, розмір];
                  for (int i = 0; i < розмір; i++)
                        for (int j = 0; j < розмір; j++)
                              таблицяМноження[i, j] = (i + 1) * (j + 1);
                  Console.WriteLine("Таблиця множення розміром {0}:",
                        розмір);
                  // Вивести горизонтальну роздільну лінію
                  for (int i = 0; i <= розмір; i++)
                        Console.Write("_____");
                  Console.WriteLine();              // Вивести новий рядок
                  Console.Write("    |");// Вивести верхній рядок таблиці
                  for (int i = 1; i <= розмір; i++)
                        Console.Write("{0,3} |", i);
                  Console.WriteLine();              // Вивести новий рядок
                  // Вивести горизонтальну роздільну лінію з
                  // вертикальними рисками
                  for (int i = 0; i <= розмір; i++)
                        Console.Write("____|");
                  Console.WriteLine();              // Вивести новий рядок
                  for (int i = 0; i < розмір; i++)// Вивести інші рядки
                  {
                        Console.Write("{0,3} |", i + 1);
                        for (int j = 0; j < розмір; j++)
                           Console.Write("{0,3} |", таблицяМноження[i, j]);
                        Console.WriteLine();        // Вивести новий рядок
                  }
            }
      }
}
```

Лістинг 3.3

Приклад коду C++/CLI

```
// Використання зубчастого масиву та робота з рядковими даними

#include "stdafx.h"

using namespace System;
//Підключення бібліотеки для використання регулярних виразів
//класів Regex та MatchCollection
using namespace System::Text::RegularExpressions;

int main(array<System::String ^> ^args)
{
	Console::WriteLine(L"Введіть розмірність зовнішнього масиву");
	size_t n = Int32::Parse(Console::ReadLine());
	//Створюємо масив рядкових (текстових) масивів
	array<array<String^>^>^ масивРеченнь
		= gcnew array<array<String^>^>(n);
	for (size_t i = 0; i < n; i++)
	{
		//Зчитуємо введене речення
		String^ речення = Console::ReadLine();
		//Змінюємо всі літери речення на маленькі
		речення = речення->ToLower();
		//Замінюємо в реченні знаки пунктуації на пробіл
		//за допомогою бібліотеки регулярних виразів
		речення = Regex::Replace(речення, "[-.?!)(,:;]", " ");
		//Замінюємо в реченні потрійні пробіли на одинарні, якщо є
		речення = речення->Replace("   ", " ");
		//Замінюємо в реченні подвійні пробіли на одинарні, якщо є
		речення = речення->Replace("  ", " ");
		//Відкидаємо пробіли на початку та в кінці речення
		речення = речення->Trim();
		//Рахуємо кількість пробілів, що залишились
		size_t count(0);
		//Створюємо змінну типу Regex і вказуємо рядок,
		//який будемо шукати (пробіл)
		Regex^ пробіл = gcnew Regex(" ");
		//Створюємо змінну типу MatchCollection і присвоюємо їй
		//результати пошуку співпадінь
		MatchCollection^ співпадіння = пробіл->Matches(речення);
		//Запам'ятовуємо в змінну count кількість знайдених пробілів
		//в реченні + 1
		count = співпадіння->Count+1;
		//Створюємо масив розмірність якого дорівнює кількості
		//слів у реченні
		масивРечень[i] = gcnew array<String^>(count);
		size_t index(0);
		String^ слово = "", ^ останнєСлово = "";
		for each (wchar_t літера in речення)
		{
			if (літера != ' ')
			{
				слово += літера;
				останнєСлово = слово;
			}
			else
			{
				масивРечень[i][index] = слово;
				index++;
				слово = "";
			}
		}
```

```
                //Оскільки після останнього слова у реченні немає пробілу,
                //записуємо його окремо після роботи циклу
                масивРечень[i][index] = останнєСлово;
                Console::WriteLine(L"\nВиправлене речення:");
                Console::WriteLine(речення);
                Console::WriteLine(L"Кількість слів у реченні = {0}\n",count);
        }
        //Виводимо результуючий масив масивів слів за допомогою індексації
        Console::WriteLine(L"\nСформований масив з речень");
        for (size_t i = 0; i < масивРечень->Length; i++)
        {
                for (size_t j = 0; j < масивРечень[i]->Length; j++)
                {
                        Console::Write(L" S({0},{1})={2}\t", i, j,
                                масивРечень[i][j]);
                }
                Console::WriteLine();
        }
        Console::WriteLine();
        //Виводимо отриманий масив масивів слів за допомогою циклу for each
        for each (array<String^>^ речення in масивРечень)
        {
                for each (String^ слово in речення)
                {
                        Console::Write(L" {0}\t", слово);
                }
                Console::WriteLine();
        }
        return 0;
}
```

Лістинг 3.4

Приклад коду C#

```
// Використання зубчастого масиву та робота з рядковими даними

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
//Підключення простору імен для класу Regex та MatchCollection
using System.Text.RegularExpressions;

namespace ConsoleAppCsh_3_2
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Введіть розмірність зовнішнього масиву");
            int n = Int32.Parse(Console.ReadLine());
            //Створюємо масив рядкових (текстових) масивів
            string[][] масивРечень = new string[n][];
            for (int i = 0; i < n; i++)
            {
                //Зчитуємо введене речення
                string речення = Console.ReadLine();
                //Змінюємо всі літери речення на маленькі
                речення = речення.ToLower();
                //Замінюємо в реченні знаки пунктуації на пробіл за
                    //допомогою бібліотеки регулярних виразів
                речення = Regex.Replace(речення, "[-.?!)(,:;]", " ");
                //Замінюємо в реченні потрійні пробіли на одинарні,
                    //якщо такі є
                речення = речення.Replace("   ", " ");
                //Замінюємо в реченні подвійні пробіли на одинарні,
                    //якщо такі є
                речення = речення.Replace("  ", " ");
                //Відкидаємо пробіли на початку та в кінці речення
                речення = речення.Trim();
                //Рахуємо кількість пробілів, що залишились
                int count = 0;
                //Створюємо змінну типу Regex і вказуємо рядок,
                    //який будемо шукати (пробіл)
                Regex пробіл = new Regex(" ");
                //Створюємо змінну типу MatchCollection і присвоюємо їй
                    //результати пошуку співпадінь
                MatchCollection співпадіння = пробіл.Matches(речення);
                //Запам'ятовуємо в змінну count кількість знайдених
                    //пробілів в реченні + 1
                count = співпадіння.Count + 1;
                //Створюємо масив розмірність якого дорівнює кількості
                    //слів у реченні
                масивРечень[i] = new string[count];
                int index = 0;
                string слово = "", останнєСлово = "";
                foreach(char літера in речення)
                {
                    if (літера != ' ')
                    {
                        слово += літера;
                        останнєСлово = слово;
                    }
                    else
```

```
                    {
                        масивРечень[i][index] = слово;
                        index++;
                        слово = "";
                    }
                }
                //Оскільки після останнього слова у реченні немає
                    //пробілу, записуємо його окремо після роботи циклу
                масивРечень[i][index] = останнєСлово;
                Console.WriteLine("\nВиправлене речення:");
                Console.WriteLine(речення);
                Console.WriteLine("Кількість слів у реченні = {0}\n",
                        count);
            }
            //Виводимо результуючий масив масивів слів
              //за допомогою індексації
            Console.WriteLine("\nСформований масив з речень");
              for (int i = 0; i<масивРечень.Length; i++)
              {
                    for (int j = 0; j<масивРечень[i].Length; j++)
                    {
                        Console.Write(" S({0},{1})={2}\t", i, j,
                            масивРечень[i][j]);
                    }
                Console.WriteLine();
              }
            Console.WriteLine();
              //Виводимо резульуючий масив масивів слів за допомогою
              //циклу foreach
              foreach(string[] речення in масивРечень)
            {
                foreach(string слово in речення)
                {
                    Console.Write(" {0}\t", слово);
                }
                Console.WriteLine();
            }
        }
    }
}
```

ПРАКТИЧНА РОБОТА №4

4 Створення додатку Windows Forms C++/CLI та C#

4.1 Початок роботи з додатком Windows Forms у CLR

Для того, щоб створити проект додатка C++/CLI на основі технології Windows Forms неодхідно виконати наступні операції.

1) Вибрати в головному меню пункт Файл ⇒ Створити ⇒ Проект… (File ⇒ New ⇒ Project…).
2) В діалоговому вікні Створення проекту (New Project) в лівій частині вікна обрати Встановлені ⇒ Шаблони ⇒ Visual C++ ⇒ CLR (Installed ⇒ Templates ⇒ Visual C++ ⇒ CLR).
3) В діалоговому вікні Створення проекту (New Project) в центральній частині вікна обрати пункт Пустий проект CLR (CLR Empty Project).
4) Задати ім'я нового проекту у полі Ім'я: (Name:).
5) Задати місце збереження нового рішення у полі Розташування: (Location:).
6) Задати ім'я нового рішення у полі Ім'я рішення: (Solution name:) та натиснути кнопку OK.

В результаті вище зазначених дій буде створений пустий проект CLR у якого відсутні будь-які елементи (рис. 4.1).

Для можливості подальшого створення віконного додатку Windows за технологією Windows Forms у створений пустий проект CLR необхідно додати пусту форму (Form), яка буде вікном майбутнього додатку.

7) Обрати в меню пункт Проект ⇒ Додати новий елемент… (Project ⇒ Add new item…).
8) З'явиться діалогове вікно Додати новий елемент (Add New Item…), в лівій частині якого слід обрати пункт Встановлені ⇒ Visual C++ ⇒ UI (Installed ⇒ Visual C++ ⇒UI).
9) Надалі в центральній частині діалогового вікна Додати новий елемент (Add New Item) обрати пункт Форма Windows Forms (Form Windows Forms).
10) У полі Ім'я: (Name:) задати ім'я нової форми (файл заголовку з розширенням .h).

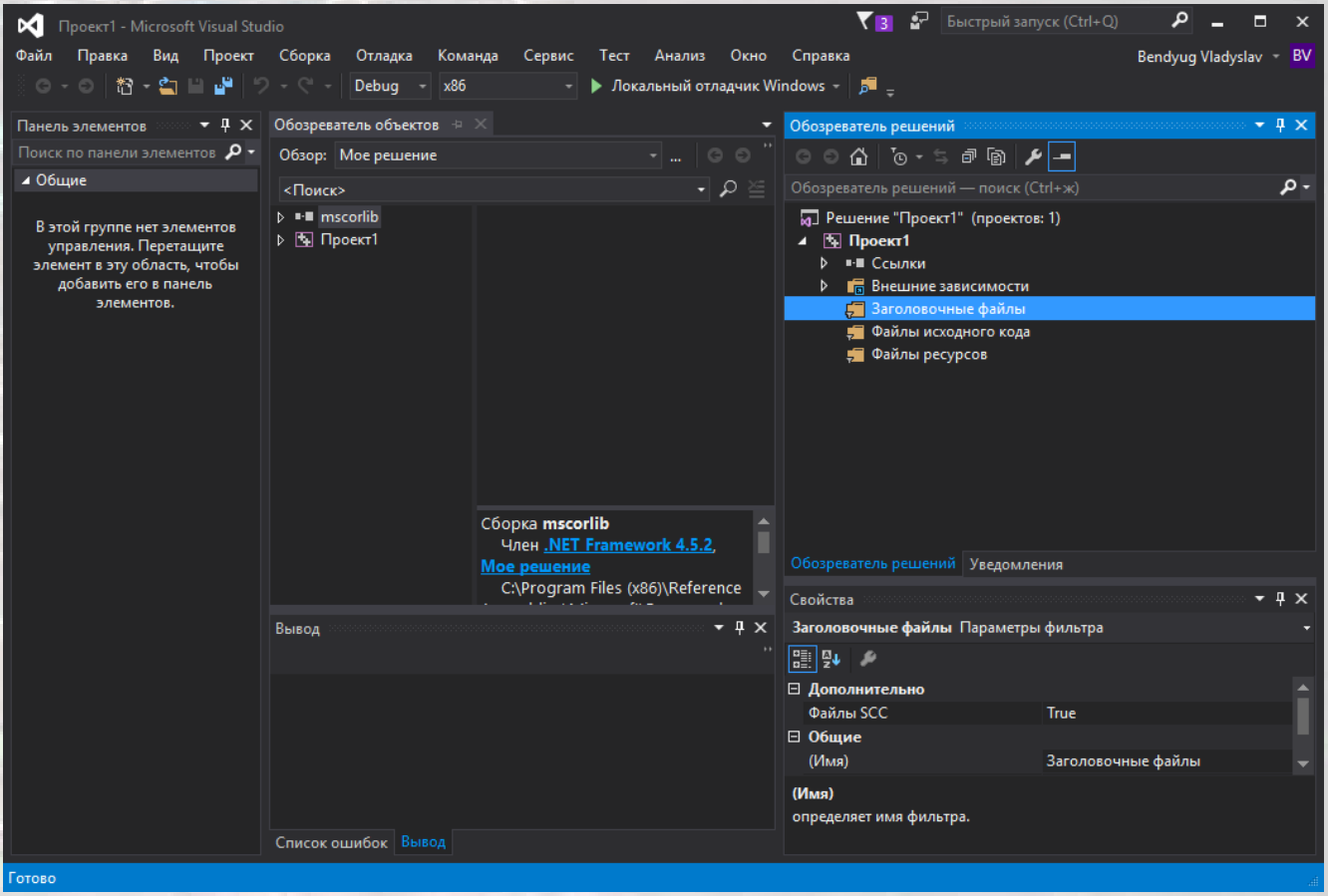


Рисунок 4.1 – Вікно середовища Visual Studio зі створеним пустим проектом CLR

11) У полі Розташування: (Location:) задається місце збереження файлу нової форми.
12) По завершенню вибору натиснути кнопку Додати (Add) для додавання нової форми у проект (рис. 4.2).

При виникненні помилки внаслідок додавання форми до проекту CLR (Visual Studio 2015) або при запуску створеного проекту з доданою формою (Visual Studio 2013) для можливості розробки проекту Windows Forms необхідно додатково виконати певний набір операцій.

1) Додати наступний програмний код у файл форми з розширенням .cpp (за замовчуванням файл MyForm.cpp), який обирається в Оглядачі рішень (Solution Explorer).

```
#include "MyForm.h"//ім'я заголовку фашої форми

using namespace Проект1; //ім'я вашого проекту

[STAThreadAttribute]
int main(array<System::String ^> ^args)
{
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    Application::Run(gcnew MyForm());
    return 0;
}
```

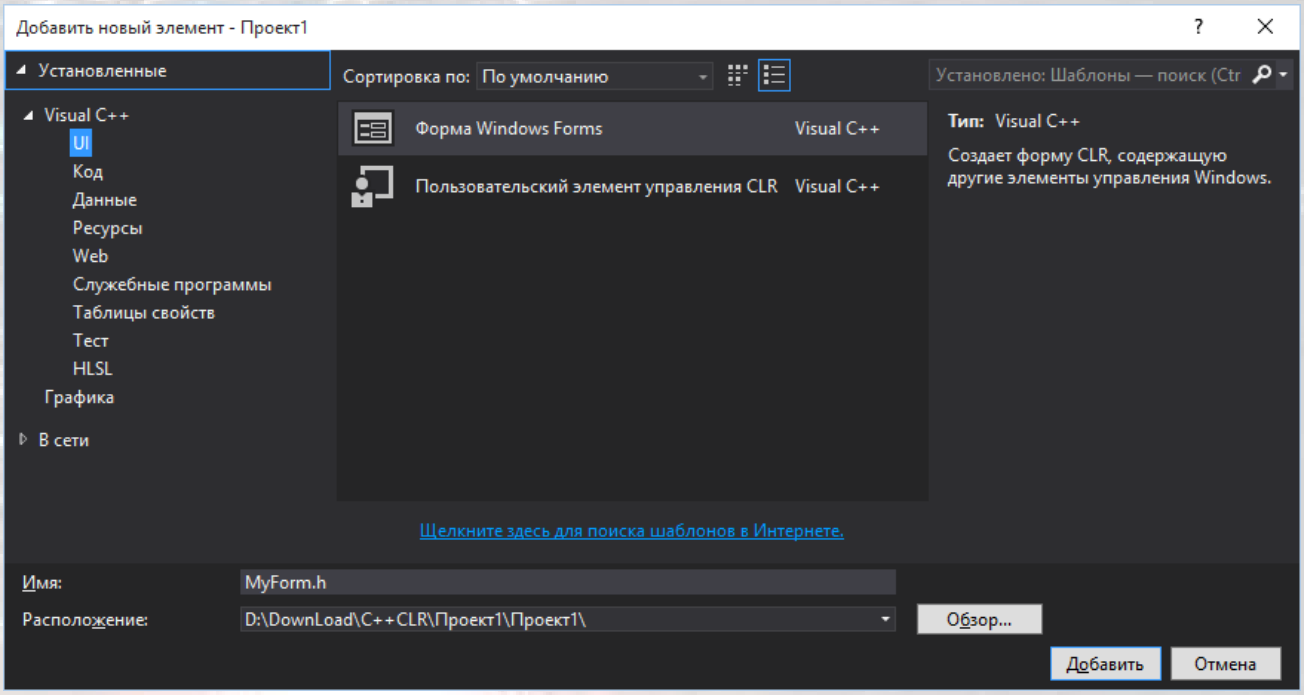


Рисунок 4.2 – Діалогове вікно додавання нового елементу до проекту CLR

2) Обрати пункт меню Проект ⇒ Властивості: Проект1 (Project ⇒ Project1 Propeties…), після чого з'явиться діалогове вікно Сторінки властивостей Проект1 (Project1 Property Pages).
3) В лівій частині діалогового вікна обираємо пункт Властивості конфігурації ⇒ Компонувальник ⇒ Система (Configuration Properties ⇒ Linker ⇒ System).
4) В правій частині діалогового вікна у полі Підсистема (SubSystem) задаємо Windows (/SUBSYSTEM:WINDOWS), як вказано на рис. 4.3.
5) В лівій частині діалогового вікна обираємо пункт Властивості конфігурації ⇒ Компонувальник ⇒ Додатково (Configuration properties ⇒ Linker ⇒ Advanced).
6) В правій частині діалогового вікна у полі Точка входу (Entry Point) задаємо main, як вказано на рис. 4.4 і натискаємо кнопку OK для застосування змін.

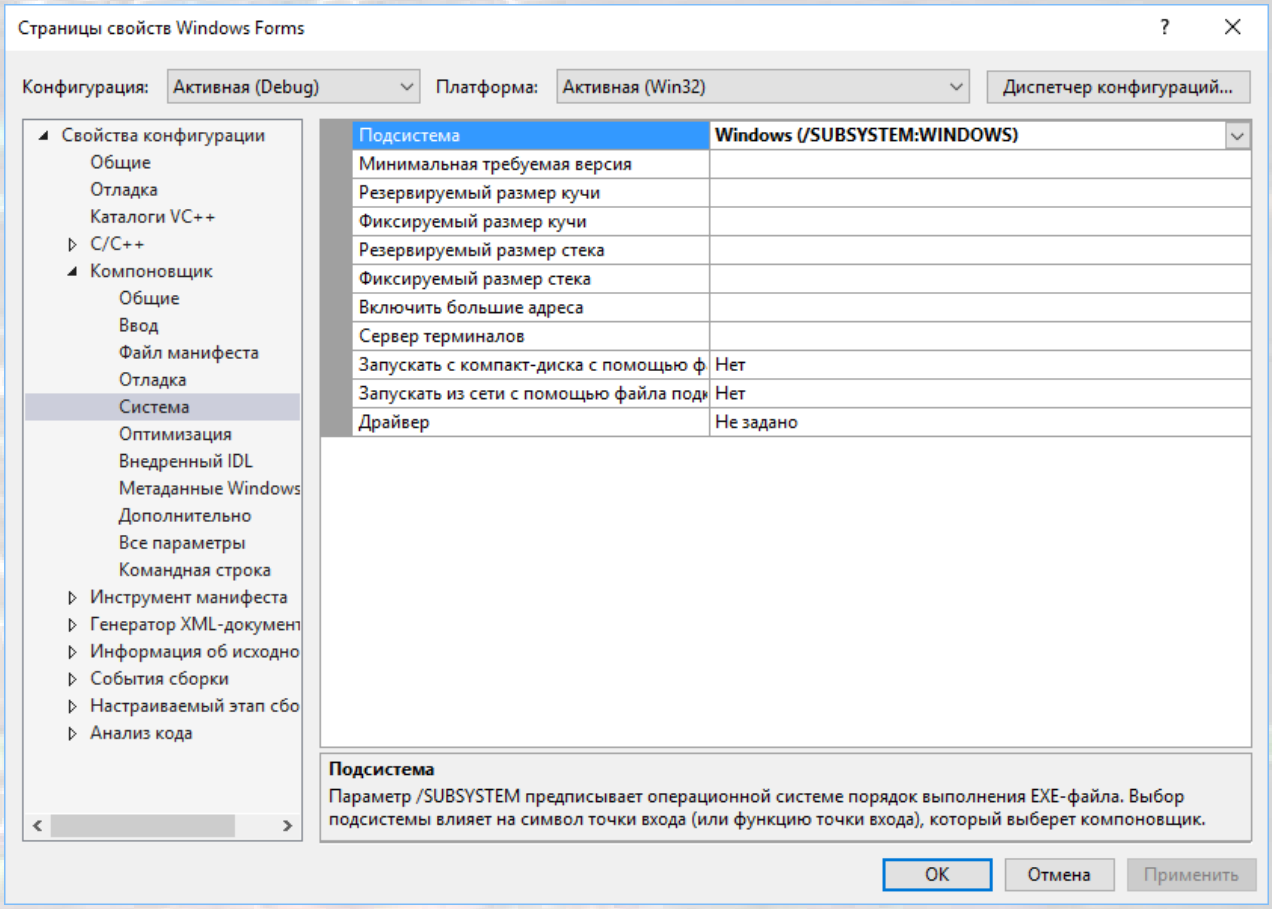


Рисунок 4.3 – Задавання значення поля Підсистема у властивостях проекту

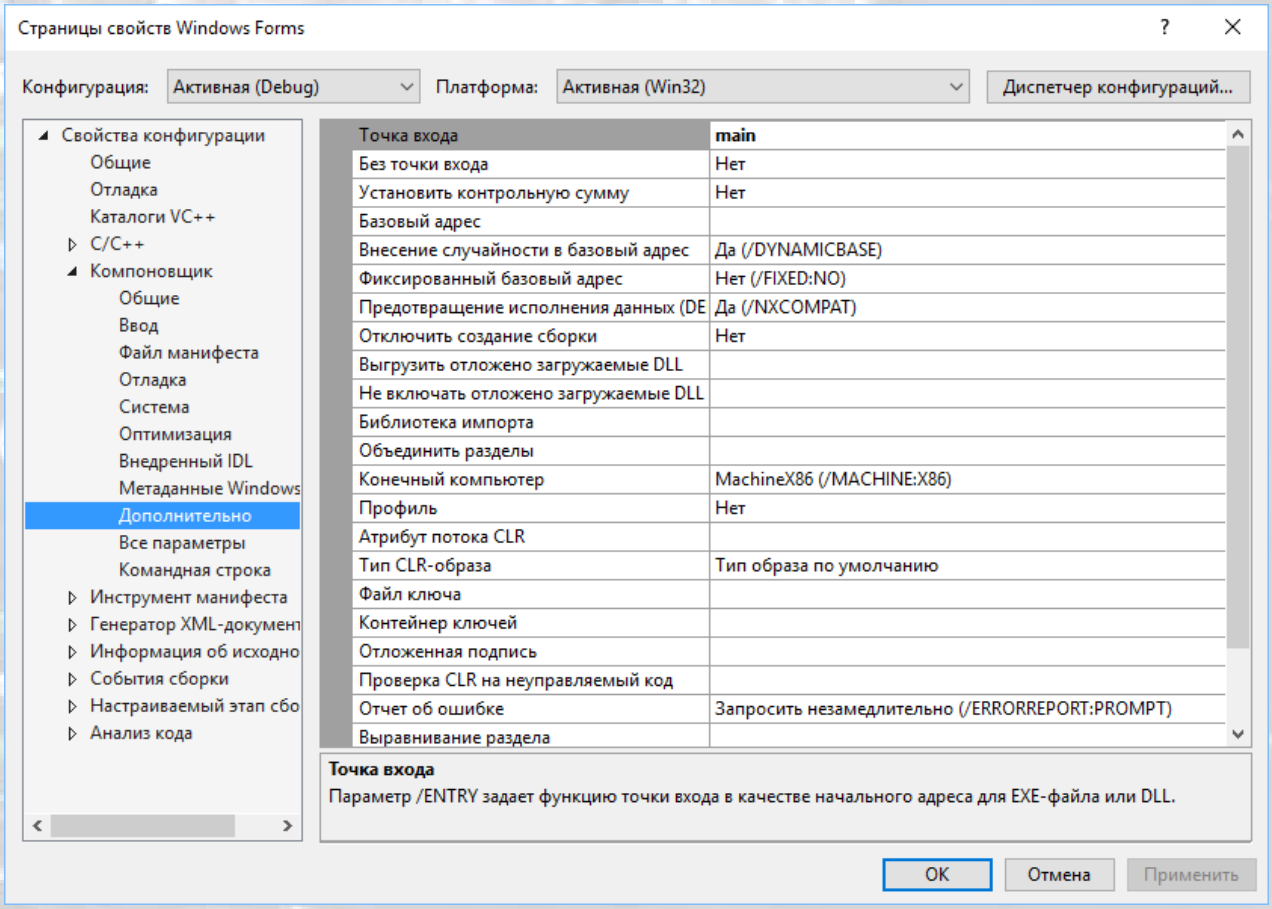


Рисунок 4.4 – Задавання значення поля Точка входу у властивостях проекту

4.2 Створення нового шаблону додатку

Для того, щоб не повторювати кожного разу даний перелік операцій при створенні нового віконного додатку Windows Forms у C++/CLI потрібно створений проект додати до шаблону проектів Visual C++. Для цього виконуємо наступні дії.

1) Обираємо пункт меню Файл ⇒ Експорт шаблону… (File ⇒ Export Template…).
2) З'явиться Майстер експорту шаблонів (Export Template Wizard). У вікні майстра Вибір типу шаблону (Choose Template Type) залишаємо все за замовчуванням та натискаємо кнопку Далі (Next) як показано на рис. 4.5.

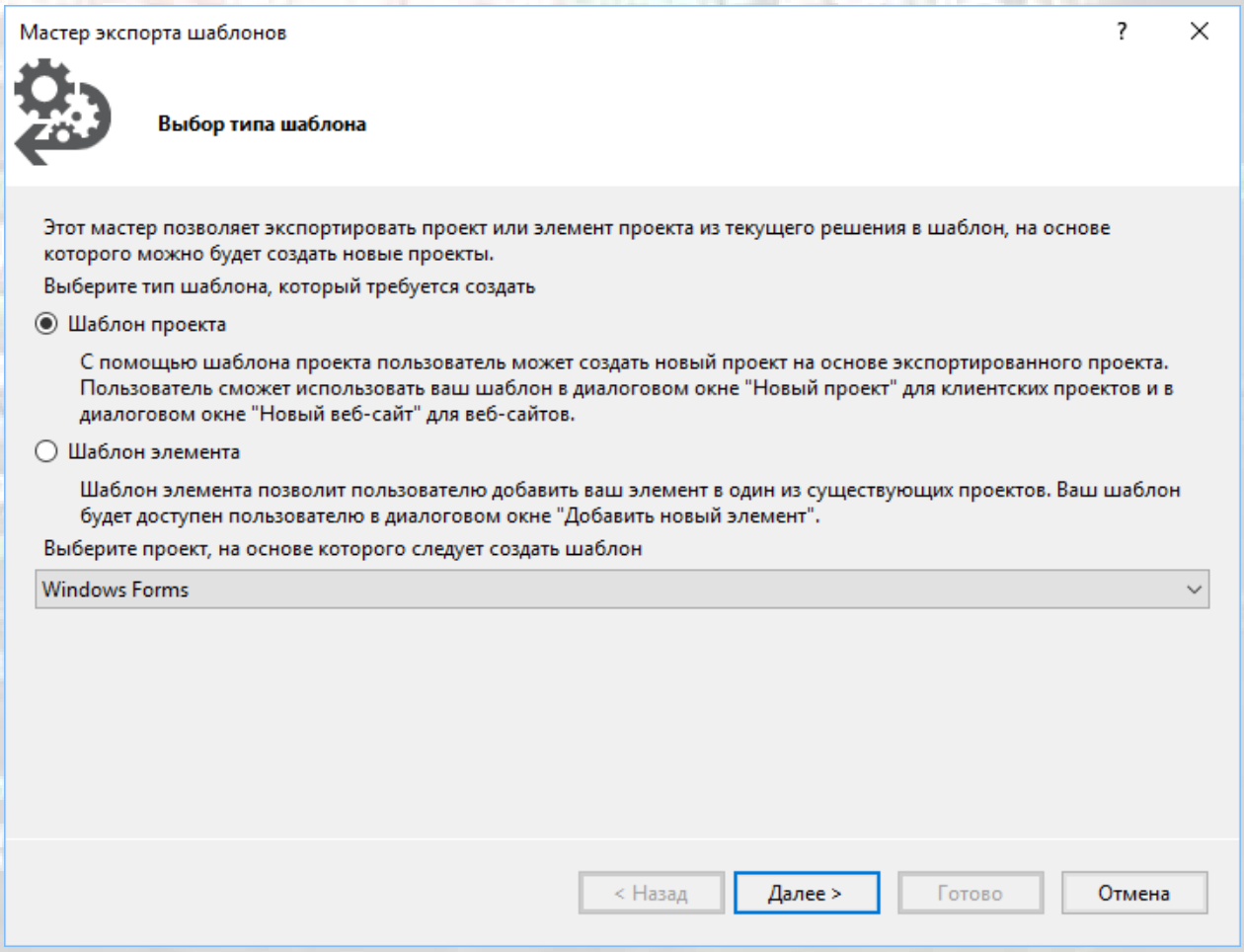


Рисунок 4.5 – Вікно Вибір типу шаблону майстра додавання шаблонів

3) В другому вікні майстра Вибір параметрів шаблону (Export Template Wizard) додаємо наступне:
   a) у полі Ім'я шаблону: (Template name:) записуємо Windows Forms;
   b) у полі Опис шаблону: (Template description:) записуємо текст пояснення «Створення віконного додатку Windows Forms (CLR)»;

c) у полі Зображення значка: (Icon Image:) за бажанням для можливості швидкого знаходження шаблону проекту серед переліку шаблонів Visual C++ додаємо графічне зображення;
d) перевіряємо вібір опції Автоматично імпортувати шаблон у Visual Studio (Automatically import the template into Visual Studio) та натискаємо кнопку Готово (Finish) як зображено на рис. 4.6.

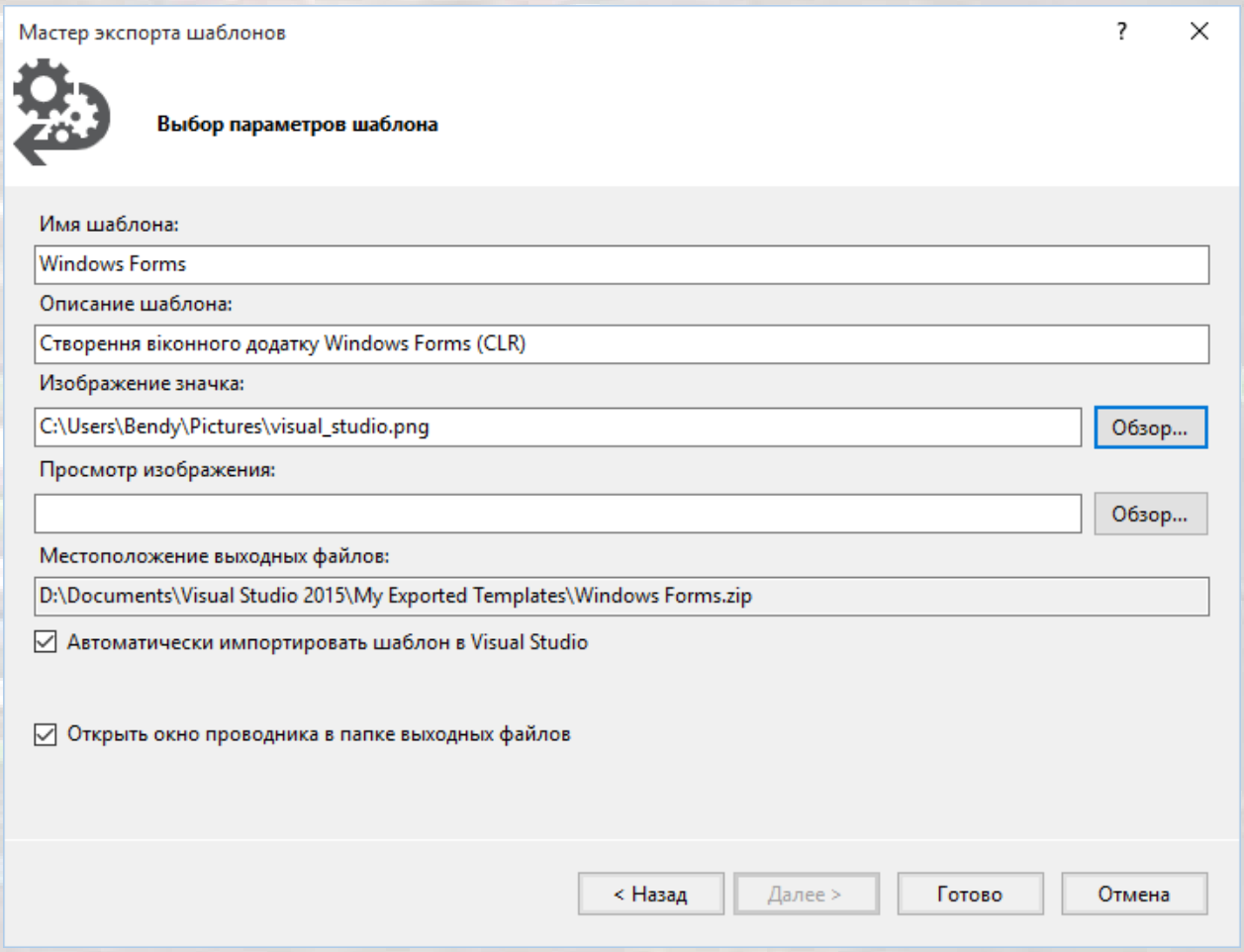


Рисунок 4.6 – Вікно Вибір параметрів шаблону майстра додавання шаблонів

Після додавання шаблону, для створення нового проекту Windows Forms необхідно буде виконати наступний перелік дій.

1) Обрати в головному меню пункт Файл ⇒ Створити ⇒ Проект… (File ⇒ New ⇒ Project…).
2) В діалоговому вікні Створення проекту (New Project) в лівій частині вікна обрати Встановлені ⇒ Шаблони ⇒ Visual C++ (Installed ⇒ Templates ⇒ Visual C++).
3) В діалоговому вікні Створення проекту (New Project) в центральній частині вікна з’явиться пункт створеного нами шаблону Windows Forms, який і потрібно обрати (рис. 4.7).

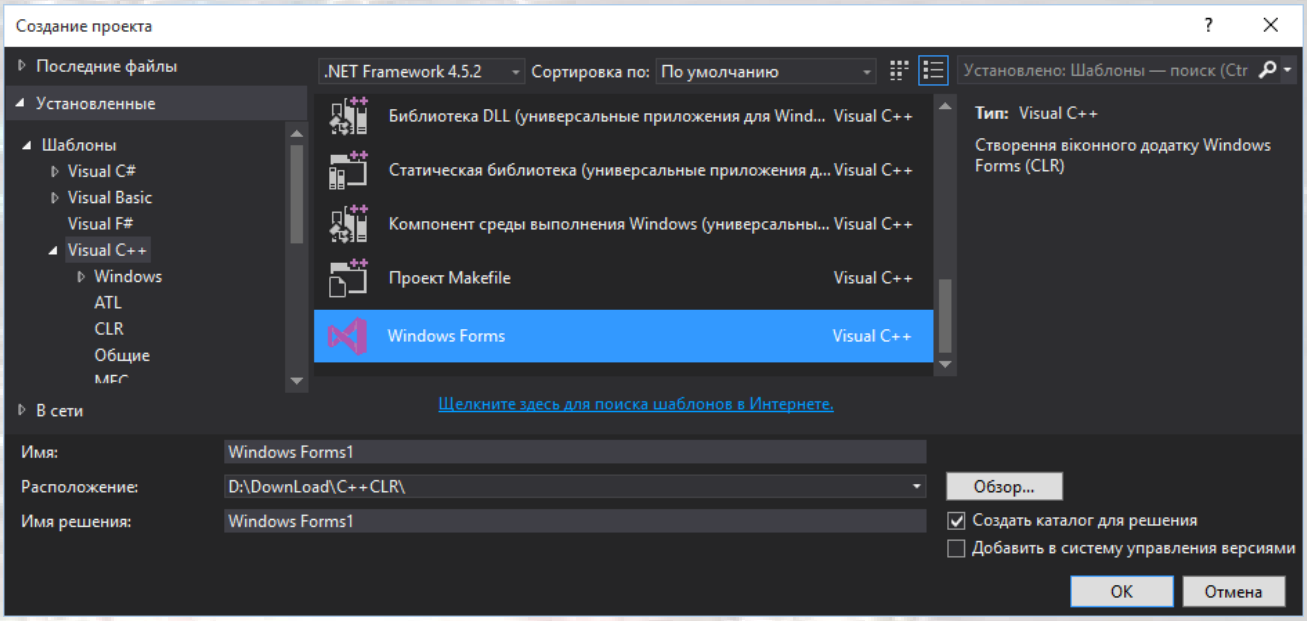


Рисунок 4.7 – Діалогове вікно створення нового проекту з доданим шаблоном Windows Forms

Після цього в IDE відкриється новий проект, який вже міститиме пусту форму (рис. 4.8).

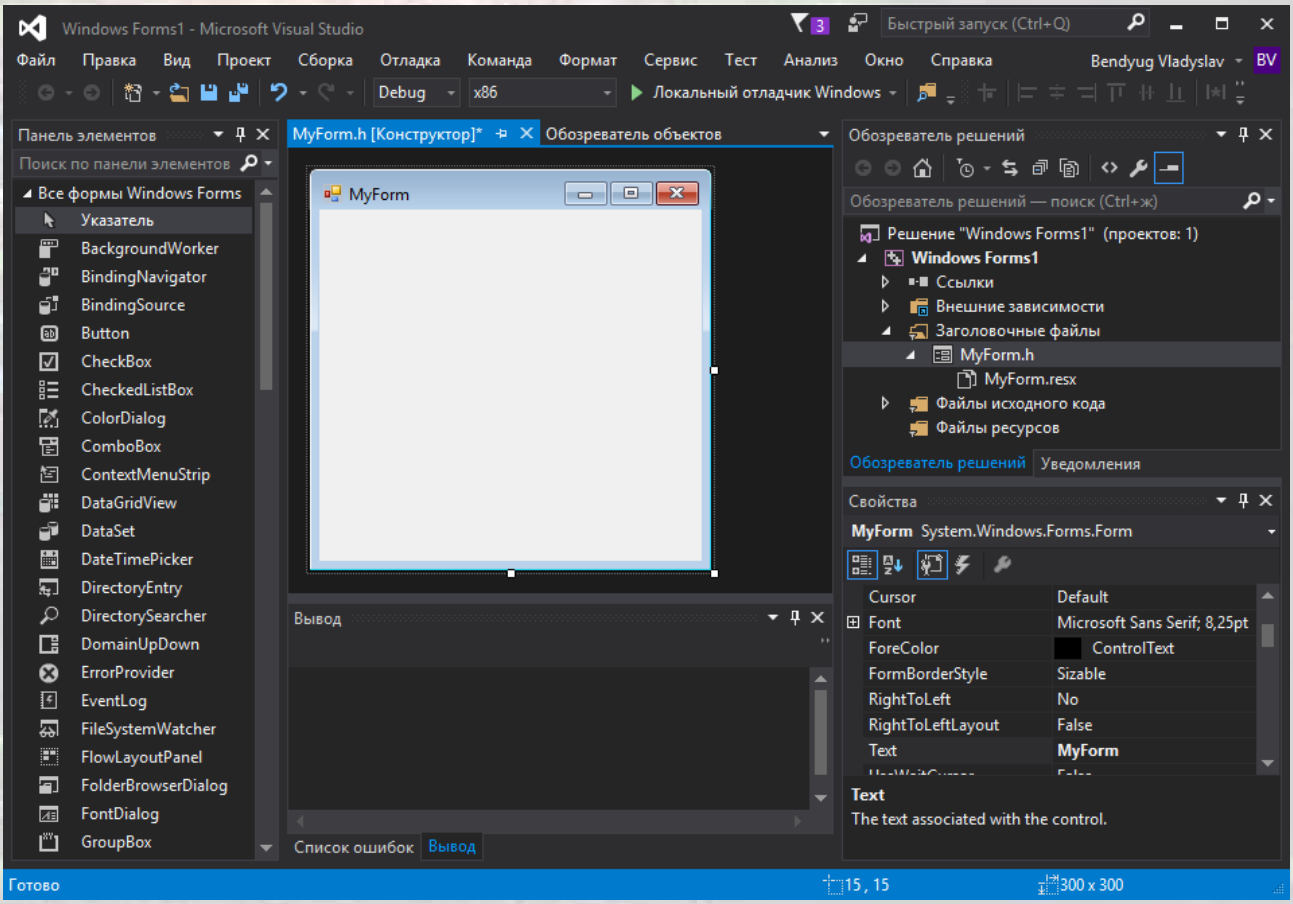


Рисунок 4.8 – Вікно IDE з відкритим проектом Windows Forms у Visual C++

4.3 Початок роботи з додатком Windows Forms у C#

Алгоритм дій для створення віконного додатку за технологією Windows Forms у мові C# виглядає значно простіше. У даному випадку потрібно обрати:

1) Пункт Файл ⇒ Створити ⇒ Проект… (File ⇒ New ⇒ Project…).
2) В діалоговому вікні Створення проекту (New Project) в лівій частині вікна обрати Встановлені ⇒ Visual C# ⇒ Windows Desktop (Installed ⇒ Visual C# ⇒ Windows Desktop).
3) В діалоговому вікні Створення проекту (New Project) в центральній частині вікна обрати пункт Windows Forms App (рис. 4.9).

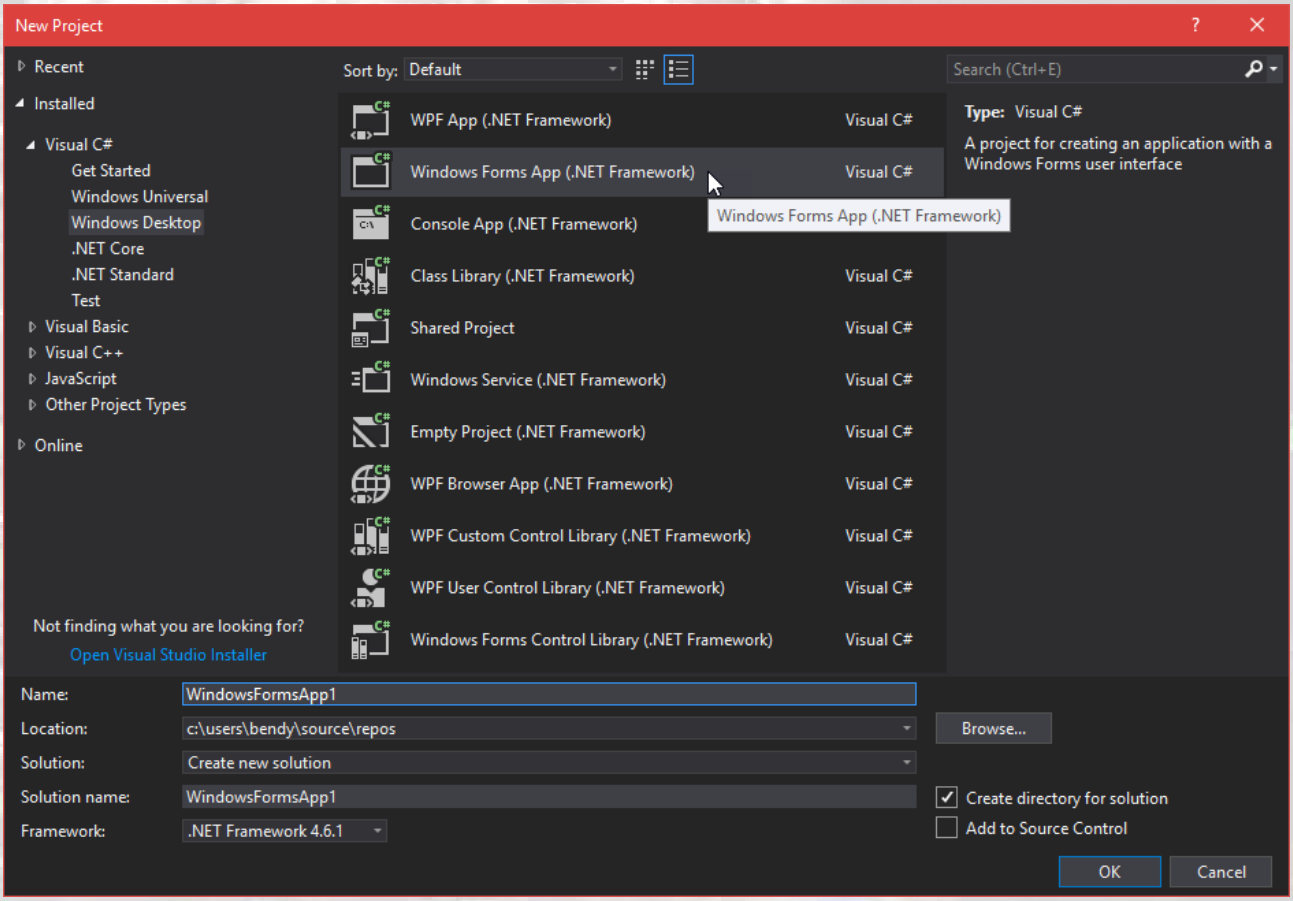


Рисунок 4.9 – Вікно створення нового проекту Windows Forms

Після цього в IDE одразу з'явиться новий проект Windows Forms. В даному проекті вже буде присутня пуста форма з ім'ям Form1, яка є заготовкою вікна майбутнього настільного додатку (рис. 4.10).

Якщо уважно подивитись на рис. 4.8 та 4.10, то можна помітити, що у проекті Windows Forms CLR файл форми зберігається у заголовочному файлі типу *.h, у той час як у проекті Windows Forms C# файл форми зберігається у файлі з розширенням *.cs. І, як ми побачимо далі, це не єдина відмінність. У проекті Windows Forms CLR програмний код з описом класу форми та програмний код з реакцією елементів керування на події – основний текст програмного модуля, містяться в одному файлі. На відміну від цього у проекті Windows Forms C# файл класу форми та файл, де ви безпосередньо будете писати текст програми, є окремими файлами. За замовчуванням вони мають ім'я Form1.Designer.cs та Form1.cs відповідно. Це надає перевагу в тому аспекті, що зменшує об'єм коду програмного файлу, з яким ви безпосередньо працюєте, і полегшує читання та сприйняття коду.

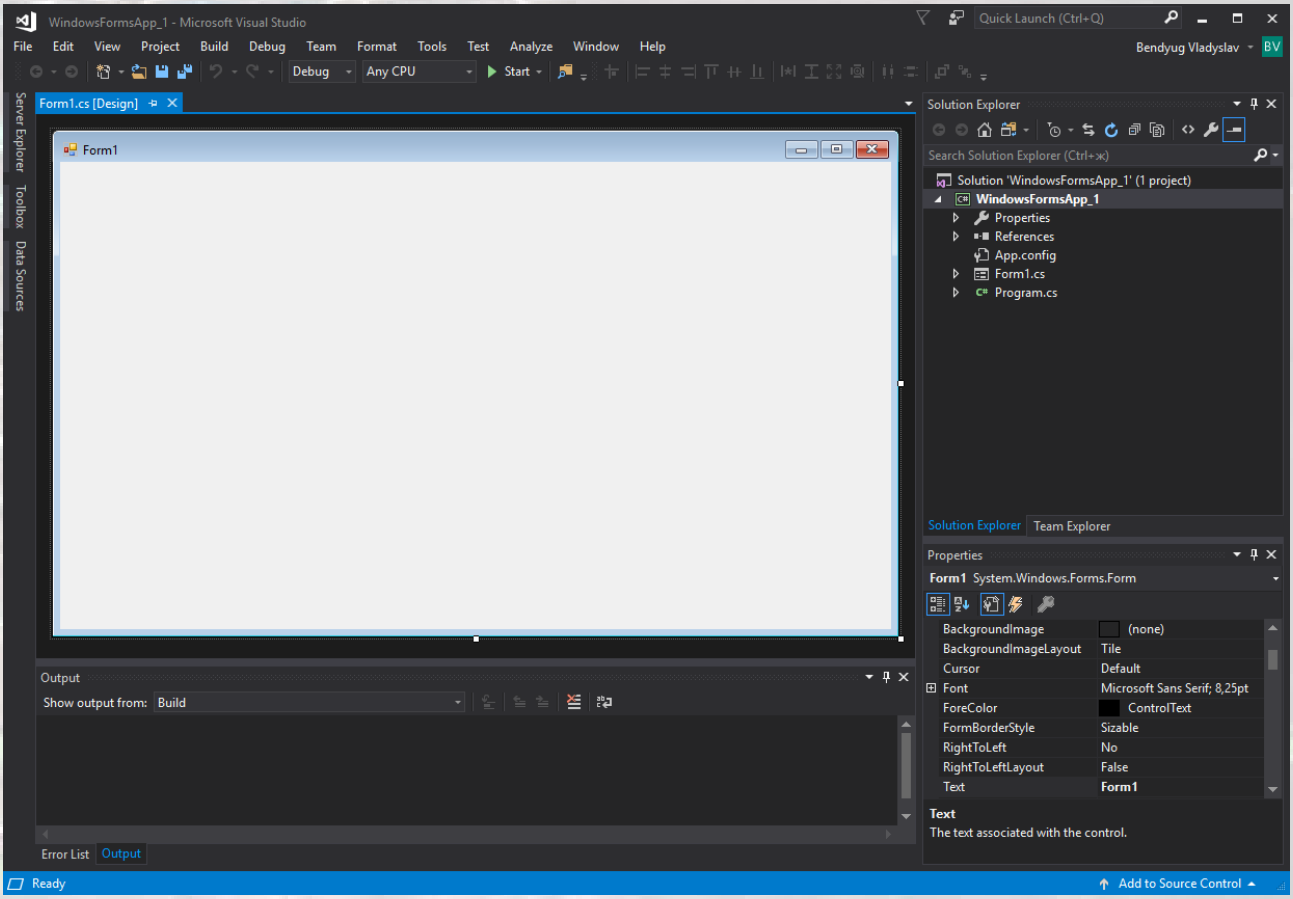


Рисунок 4.10 – Вікно IDE з відкритим проектом `Windows Forms` у `Visual C#`

4.4 Створення нового додатку Windows Forms

Завдання 1.

Необхідно створити додаток, який розраховуватиме площу колового сегменту. У якості вихідних даних повинні задаватись радіус кола, початковий кут сегменту, кінцевий кут сегменту, крок зміни кута. Розрахунок площі сегменту потрібно організувати в циклі змінюючи кут колового сегменту на заданий крок з кожною ітерацією.

Для створення додатку додайте в нього зазначені компоненти інтерфейсу та присвойте їм потрібні властивості суворо у порядку вказаному в табл. 4.1 для вірного розташування об'єктів в межах форми.

Таблиця 4.1 – Об'єкти додатку Windows Forms та їх властивості

№	Об'єкт	Ім'я	Властивість Text	Інші властивості
1	Forms	MyForm (Form1)	Площа сегменту	StartPosition: CenterScreen
2	Panel	panel_inp		Dock:Left
3	Panel	panel_out		Dock: Fill
4	GroupBox	groupBox_input	Розрахунок площі колового сегменту	Dock: Fill
5	Button	buttonCalc	Розрахувати площу	Dock: Bottom
6	Label	label_a_beg	Початковий кут a, градусів	Dock: Top
7	TextBox	textBox_a_beg		Dock: Top
8	Label	label_a_end	Кінцевий кут a, градусів	Dock: Top
9	TextBox	textBox_a_end		Dock: Top

10	Label	label_a_step	Крок зміни кута a, градусів	Dock: Top
11	TextBox	textBox_a_step		Dock: Top
12	Label	label_r	Радіус r, мм	Dock: Top
13	TextBox	textBox_r		Dock: Top
14	Button	button_close	Закрити програму	Dock: Bottom
15	RichTextBox	richTextBox_result		Dock: Fill

Після додавання всіх компонентів вікно Конструктора матиме вигляд як зображено на рис. 4.9.

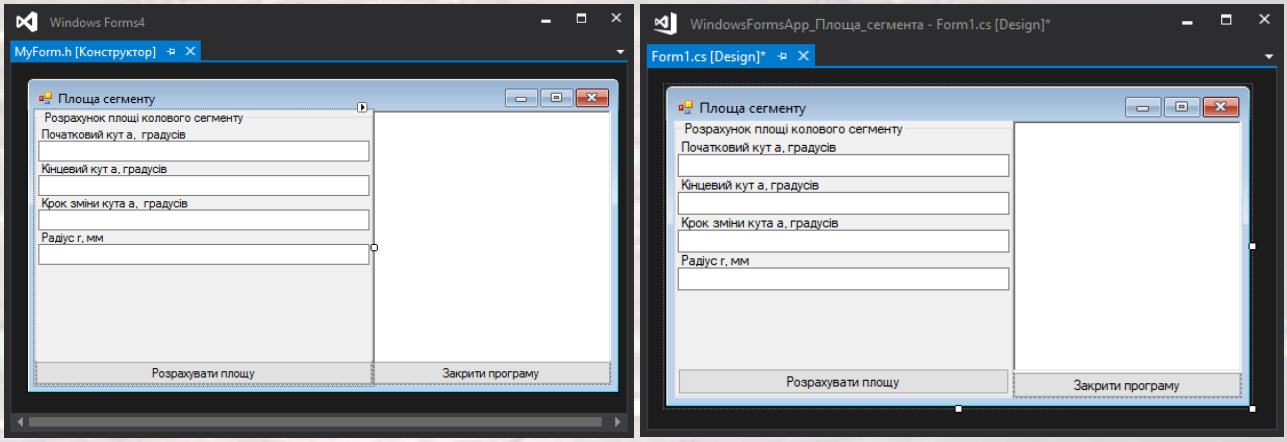


Рисунок 4.9 – Вікно Конструктора форми у Visual C++ та Visual C# з доданими елементами інтерфейсу

Двічі клацніть по об'єкту buttonCalc у вікні Конструктора форми. Після цього відкриється вікно Редактора з пустою подією (рис. 4.10).

Надалі впишіть програмний код для розрахунку площі колового сегменту у подію buttonCalc_Click.

Приклад коду C++/CLI

```
private: System::Void buttonCalc_Click(System::Object^  sender,
System::EventArgs^  e) {
      double r, s;
      int a_beg, a_end, a_step, a;
      //Перетворює рядкове представлення числа в ціле
      a_beg = int::Parse(textBox_a_beg->Text);
      a_end = int::Parse(textBox_a_end->Text);
      a_step = int::Parse(textBox_a_step->Text);
      //Перетворює рядкове представлення числа в дійсне
      r = double::Parse(textBox_r->Text);
      a = a_beg;
      //Додає текст, перетворює дійсне значення в рядкове
      //та додає символ кінця рядка
      richTextBox_result->AppendText("Для довжини радіусу r = "
            + r.ToString("F") + "\r\n");
      while (a <= a_end)
      {
            //Розраховує площу колового сегменту
            s = 0.5*pow(r, 2)*(a - sin(a));
```

```
            richTextBox_result->AppendText("при значенні кута a = "+
                  a.ToString("F") + " розрахована площа колового
                  сегменту S = " + s.ToString("F") + " мм" + "\r\n");
            a += a_step;
      }
      //Виводить діалогове вікно з кінцевим результатом розрахунку
      MessageBox::Show("Для довжини радіусу r = " + r.ToString("F")
            + " при значенні кута a = " + (a - a_step).ToString("F")
            + "\r\n" + "розрахована площа колового сегменту S = "
            + s.ToString("F") + " мм");
}
```

Для C# дещо видозмінимо код реакції на натискання кнопки. Для цього введемо дві проміжні змінні `str1` та `str2` типу `string`, які будемо використовувати для формування рядків з результатами розрахунку.

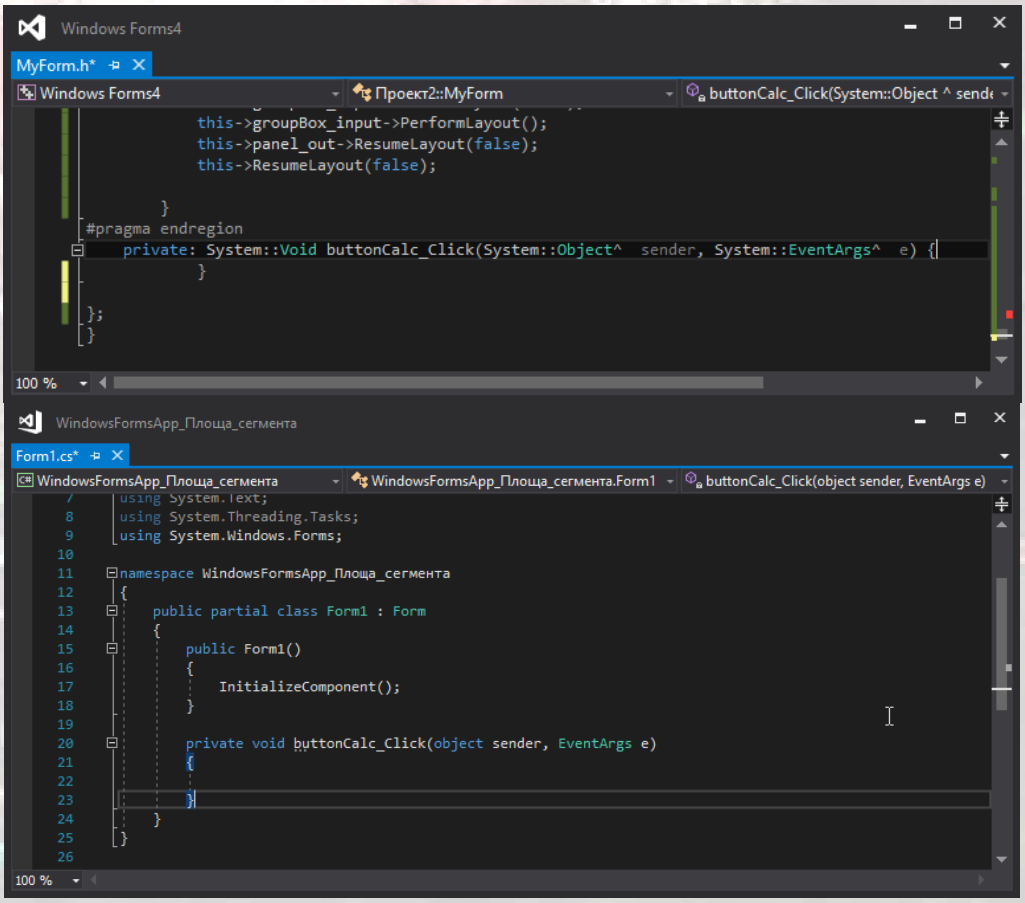


Рисунок 4.10 – Вікно Редактора коду з пустою подією `buttonCalc_Click` у `Visual C++` та `Visual C#`

Приклад коду C#

```
private void buttonCalc_Click(object sender, EventArgs e)
{
      double r, s;
      int a_beg, a_end, a_step, a;
      string str1="", str2="";
      //Перетворює рядкове представлення числа в ціле
      a_beg = int.Parse(textBox_a_beg.Text);
```

```
        a_end = int.Parse(textBox_a_end.Text);
        a_step = int.Parse(textBox_a_step.Text);
        //Перетворює рядкове представлення числа в дійсне
        r = double.Parse(textBox_r.Text);
        a = a_beg;
        //Додає текст, перетворює дійсне значення в рядкове
        //та додає символ кінця рядка
        str1 = "Для довжини радіусу r = " + r.ToString("F") + " ";
        richTextBox_result.AppendText(str1);
        while (a <= a_end)
        {
            //Розраховує площу колового сегменту
            s = 0.5 * Math.Pow(r, 2) * (a - Math.Sin(a));
            str2 =  "при значенні кута a = " + a.ToString("F")
                + " розрахована площа колового сегменту S = "
                + s.ToString("F") + " мм";
            richTextBox_result.AppendText("\r\n" + str2);
            a += a_step;
        }
        //Виводить діалогове вікно з кінцевим результатом розрахунку
        MessageBox.Show(str1+str2);
}
```

Двічі клацніть по об’єкту button_close у вікні Конструктора форми. Після цього відкриється вікно Редактора з пустою подією button_close_Click. Впишіть як реакцію на цю подію функцію закриття програми.

```
private: System::Void button_close_Click(System::Object^  sender, System::EventArgs^  e) {
        Close(); //Закриває вікно програми
}
```

Запустіть проект на виконання та проведіть розрахунки, увівши необхідні вихідні дані. Вікно додатку повинне мати вигляд, як зображено на рис. 4.11.

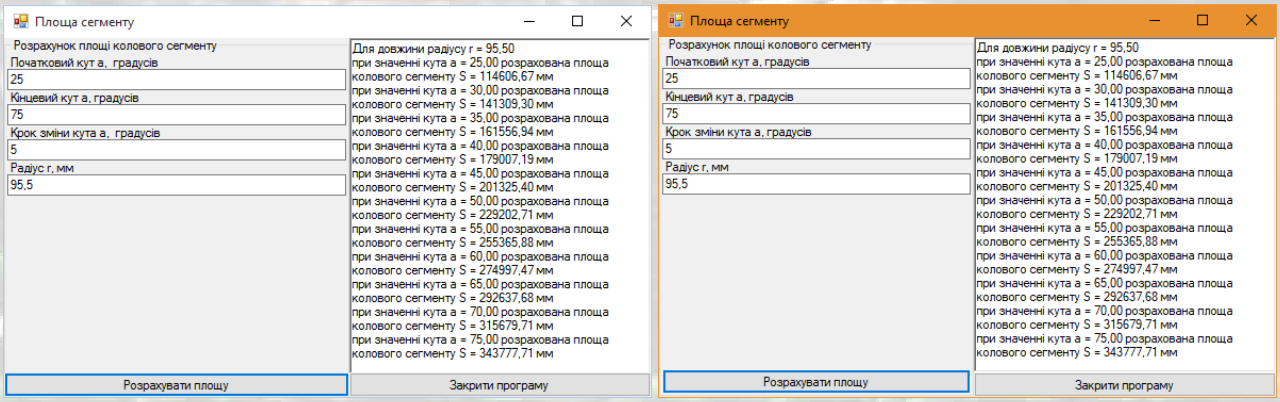


Рисунок 4.11 – Вікно додатку з результатами розрахунку у Visual C++ та Visual C#

Окрім цього по завершенню розрахунків на екран виведеться діалогове вікно з результатом розрахунку останньої ітерації (рис. 4.12).

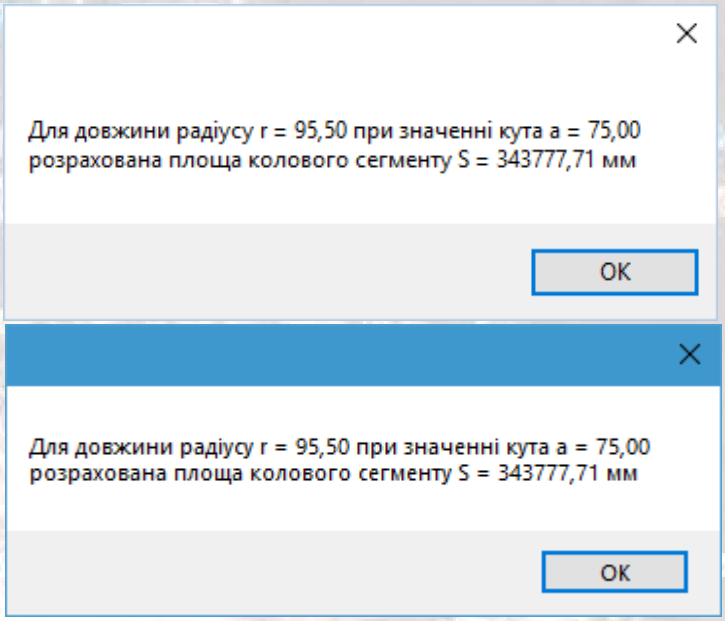


Рисунок 4.12 –Діалогове вікно з результатом розрахунку у Visual C++ та Visual C# останньої ітерації

Повний код проекту наведений нижче в лістингах 4.1 - 4.2 для Visual C++ та у лістингах 4.3 - 4.5 для Visual C#. На прикладі цих лістингів можна порівняти структуру проекту у Visual C++ та у Visual C#. У Visual C# автоматично створюваний код знаходиться у файлах Program.cs та Form1.Designer.cs, а код реакцій на події розміщений у окремому файлі Form1.cs.

Програмний код

Лістинг 4.1

Приклад коду C++/CLI

```
//Програмний код модуля MyForm.cpp

#include "MyForm.h"

using namespace WindowsForms; //ім'я вашого проекту
//#include <math.h>

[STAThreadAttribute]
int main(array<System::String ^> ^args)
{
	//Активізація візуальних ефектів Windows
	//для створення елементів керування
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	//Створення головного вікна та його виконання
	Application::Run(gcnew MyForm());
	return 0;
}
```

Лістинг 4.2

Приклад коду C++/CLI

```
//Програмний код заголовку MyForm.h

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	#include <math.h> //Підключення бібліотеки математичних функцій

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Panel^  panel_inp;
	protected:

	private: System::Windows::Forms::Button^  buttonCalc;
	private: System::Windows::Forms::GroupBox^  groupBox_input;
	protected:

	private: System::Windows::Forms::TextBox^  textBox_r;
	private: System::Windows::Forms::Label^  label_r;
	private: System::Windows::Forms::TextBox^  textBox_a_step;
	private: System::Windows::Forms::Label^  label_a_step;
	private: System::Windows::Forms::TextBox^  textBox_a_end;
	private: System::Windows::Forms::Label^  label_a_end;
	private: System::Windows::Forms::TextBox^  textBox_a_beg;
	private: System::Windows::Forms::Label^  label_a_beg;
	private: System::Windows::Forms::Panel^  panel_out;
	private: System::Windows::Forms::RichTextBox^  richTextBox_result;
	private: System::Windows::Forms::Button^  button_close;

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
```

```
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Требуемый метод для поддержки конструктора — не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        void InitializeComponent(void)
        {
            this->panel_inp
                = (gcnew System::Windows::Forms::Panel());
            this->buttonCalc
                = (gcnew System::Windows::Forms::Button());
            this->groupBox_input
                = (gcnew System::Windows::Forms::GroupBox());
            this->textBox_r
                = (gcnew System::Windows::Forms::TextBox());
            this->label_r
                = (gcnew System::Windows::Forms::Label());
            this->textBox_a_step
                = (gcnew System::Windows::Forms::TextBox());
            this->label_a_step
                = (gcnew System::Windows::Forms::Label());
            this->textBox_a_end
                = (gcnew System::Windows::Forms::TextBox());
            this->label_a_end
                = (gcnew System::Windows::Forms::Label());
            this->textBox_a_beg
                = (gcnew System::Windows::Forms::TextBox());
            this->label_a_beg
                = (gcnew System::Windows::Forms::Label());
            this->panel_out
                = (gcnew System::Windows::Forms::Panel());
            this->button_close
                = (gcnew System::Windows::Forms::Button());
            this->richTextBox_result
                = (gcnew System::Windows::Forms::RichTextBox());
            this->panel_inp->SuspendLayout();
            this->groupBox_input->SuspendLayout();
            this->panel_out->SuspendLayout();
            this->SuspendLayout();
            //
            // panel_inp
            //
            this->panel_inp->Controls->Add(this->buttonCalc);
            this->panel_inp->Controls->Add(this->groupBox_input);
            this->panel_inp->Dock
                = System::Windows::Forms::DockStyle::Left;
            this->panel_inp->Location
                = System::Drawing::Point(0, 0);
            this->panel_inp->Name = L"panel_inp";
            this->panel_inp->Size = System::Drawing::Size(324, 263);
            this->panel_inp->TabIndex = 1;
            //
            // buttonCalc
            //
            this->buttonCalc->Dock
                = System::Windows::Forms::DockStyle::Bottom;
            this->buttonCalc->Location
                = System::Drawing::Point(0, 240);
            this->buttonCalc->Name = L"buttonCalc";
            this->buttonCalc->Size = System::Drawing::Size(324, 23);
            this->buttonCalc->TabIndex = 10;
            this->buttonCalc->Text = L"Розрахувати площу";
            this->buttonCalc->UseVisualStyleBackColor = true;
            this->buttonCalc->Click += gcnew System
```

```
            ::EventHandler(this, &MyForm::buttonCalc_Click);
//
// groupBox_input
//
this->groupBox_input->AutoSize = true;
this->groupBox_input->Controls->Add(this->textBox_r);
this->groupBox_input->Controls->Add(this->label_r);
this->groupBox_input->Controls->
       Add(this->textBox_a_step);
this->groupBox_input->Controls->Add(this->label_a_step);
this->groupBox_input->Controls->
       Add(this->textBox_a_end);
this->groupBox_input->Controls->Add(this->label_a_end);
this->groupBox_input->Controls->
       Add(this->textBox_a_beg);
this->groupBox_input->Controls->Add(this->label_a_beg);
this->groupBox_input->Dock
       = System::Windows::Forms::DockStyle::Fill;
this->groupBox_input->Location
       = System::Drawing::Point(0, 0);
this->groupBox_input->Name = L"groupBox_input";
this->groupBox_input->Size
       = System::Drawing::Size(324, 263);
this->groupBox_input->TabIndex = 1;
this->groupBox_input->TabStop = false;
this->groupBox_input->Text
       = L"Розрахунок площі колового сегменту";
//
// textBox_r
//
this->textBox_r->Dock
       = System::Windows::Forms::DockStyle::Top;
this->textBox_r->Location
       = System::Drawing::Point(3, 128);
this->textBox_r->Name = L"textBox_r";
this->textBox_r->Size = System::Drawing::Size(318, 20);
this->textBox_r->TabIndex = 8;
//
// label_r
//
this->label_r->AutoSize = true;
this->label_r->Dock
       = System::Windows::Forms::DockStyle::Top;
this->label_r->Location
       = System::Drawing::Point(3, 115);
this->label_r->Name = L"label_r";
this->label_r->Size = System::Drawing::Size(67, 13);
this->label_r->TabIndex = 7;
this->label_r->Text = L"Радіус r, мм";
//
// textBox_a_step
//
this->textBox_a_step->Dock
       = System::Windows::Forms::DockStyle::Top;
this->textBox_a_step->Location
       = System::Drawing::Point(3, 95);
this->textBox_a_step->Name = L"textBox_a_step";
this->textBox_a_step->Size
       = System::Drawing::Size(318, 20);
this->textBox_a_step->TabIndex = 6;
//
// label_a_step
//
this->label_a_step->AutoSize = true;
this->label_a_step->Dock
       = System::Windows::Forms::DockStyle::Top;
this->label_a_step->Location
```

```
            = System::Drawing::Point(3, 82);
this->label_a_step->Name = L"label_a_step";
this->label_a_step->Size
            = System::Drawing::Size(148, 13);
this->label_a_step->TabIndex = 5;
this->label_a_step->Text
            = L"Крок зміни кута a,  градусів";
//
// textBox_a_end
//
this->textBox_a_end->Dock
            = System::Windows::Forms::DockStyle::Top;
this->textBox_a_end->Location
            = System::Drawing::Point(3, 62);
this->textBox_a_end->Name = L"textBox_a_end";
this->textBox_a_end->Size
            = System::Drawing::Size(318, 20);
this->textBox_a_end->TabIndex = 4;
//
// label_a_end
//
this->label_a_end->AutoSize = true;
this->label_a_end->Dock
            = System::Windows::Forms::DockStyle::Top;
this->label_a_end->Location
            = System::Drawing::Point(3, 49);
this->label_a_end->Name = L"label_a_end";
this->label_a_end->Size
            = System::Drawing::Size(128, 13);
this->label_a_end->TabIndex = 3;
this->label_a_end->Text = L"Кінцевий кут a, градусів";
this->label_a_end->TextAlign
            = System::Drawing::ContentAlignment::BottomCenter;
//
// textBox_a_beg
//
this->textBox_a_beg->Dock
            = System::Windows::Forms::DockStyle::Top;
this->textBox_a_beg->Location
            = System::Drawing::Point(3, 29);
this->textBox_a_beg->Name = L"textBox_a_beg";
this->textBox_a_beg->Size
            = System::Drawing::Size(318, 20);
this->textBox_a_beg->TabIndex = 2;
//
// label_a_beg
//
this->label_a_beg->AutoSize = true;
this->label_a_beg->Dock
            = System::Windows::Forms::DockStyle::Top;
this->label_a_beg->Location
            = System::Drawing::Point(3, 16);
this->label_a_beg->Name = L"label_a_beg";
this->label_a_beg->Size
            = System::Drawing::Size(146, 13);
this->label_a_beg->TabIndex = 1;
this->label_a_beg->Text
            = L"Початковий кут a,  градусів";
//
// panel_out
//
this->panel_out->Controls->Add(this->button_close);
this->panel_out->Controls->
            Add(this->richTextBox_result);
this->panel_out->Dock
            = System::Windows::Forms::DockStyle::Fill;
this->panel_out->Location
```

```
                = System::Drawing::Point(324, 0);
            this->panel_out->Name = L"panel_out";
            this->panel_out->Size = System::Drawing::Size(229, 263);
            this->panel_out->TabIndex = 2;
            //
            // button_close
            //
            this->button_close->Dock
                = System::Windows::Forms::DockStyle::Bottom;
            this->button_close->Location
                = System::Drawing::Point(0, 240);
            this->button_close->Name = L"button_close";
            this->button_close->Size
                = System::Drawing::Size(229, 23);
            this->button_close->TabIndex = 1;
            this->button_close->Text = L"Закрити програму";
            this->button_close->UseVisualStyleBackColor = true;
            this->button_close->Click += gcnew System
                ::EventHandler(this, &MyForm::button1_Click);
            //
            // richTextBox_result
            //
            this->richTextBox_result->Dock
                = System::Windows::Forms::DockStyle::Fill;
            this->richTextBox_result->Location
                = System::Drawing::Point(0, 0);
            this->richTextBox_result->Name = L"richTextBox_result";
            this->richTextBox_result->Size
                = System::Drawing::Size(229, 263);
            this->richTextBox_result->TabIndex = 0;
            this->richTextBox_result->Text = L"";
            //
            // MyForm
            //
            this->AutoScaleDimensions
                = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode
                = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(553, 263);
            this->Controls->Add(this->panel_out);
            this->Controls->Add(this->panel_inp);
            this->Name = L"MyForm";
            this->StartPosition = System::Windows
                ::Forms::FormStartPosition::CenterScreen;
            this->Text = L"Площа сегменту";
            this->panel_inp->ResumeLayout(false);
            this->panel_inp->PerformLayout();
            this->groupBox_input->ResumeLayout(false);
            this->groupBox_input->PerformLayout();
            this->panel_out->ResumeLayout(false);
            this->ResumeLayout(false);

        }
#pragma endregion
    private: System::Void buttonCalc_Click(System::Object^  sender, System::EventArgs^
e) {
        double r, s;
        int a_beg, a_end, a_step, a;
        //Перетворює рядкове представлення числа в ціле
        a_beg = int::Parse(textBox_a_beg->Text);
        a_end = int::Parse(textBox_a_end->Text);
        a_step = int::Parse(textBox_a_step->Text);
        //Перетворює рядкове представлення числа в дійсне
        r = double::Parse(textBox_r->Text);
        a = a_beg;
        //Додає текст, перетворює дійсне значення в рядкове
        //та додає символ кінця рядка
```

```
                richTextBox_result->AppendText("Для довжини радіусу r = "
                        + r.ToString("F") + "\r\n");
                while (a <= a_end)
                {
                        //Розраховує площу колового сегменту
                        s = 0.5*pow(r, 2)*(a - sin(a));
                        richTextBox_result->AppendText("при значенні кута a = "
                                + a.ToString("F")
                                + " розрахована площа колового сегменту S = "
                                + s.ToString("F") + " мм" + "\r\n");
                        a += a_step;
                }
                //Виводить діалогове вікно з повідомленням
                MessageBox::Show("Для довжини радіусу r = " + r.ToString("F")
                        + " при значенні кута a = " + (a - a_step).ToString("F")
                        + "\r\n" + "розрахована площа колового сегменту S = "
                        + s.ToString("F") + " мм");
        }

private: System::Void button_close_Click(System::Object^  sender, System::EventArgs^  e) {
        Close(); //Закриває вікно програми
}
};
}
```

Лістинг 4.3

Приклад коду C#

```
//Програмний код файлу Program.cs, який був згенерований
//дизайнером коду

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp_Площа_сегмента
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Лістинг 4.4

Приклад коду C#

```
//Програмний код файлу Form1.Designer.cs, який був згенерований //дизайнером коду

namespace WindowsFormsApp_Площа_сегмента
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources
        /// should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.panel_inp = new System.Windows.Forms.Panel();
            this.panel_out = new System.Windows.Forms.Panel();
            this.groupBox_input = new System.Windows.Forms.GroupBox();
            this.buttonCalc = new System.Windows.Forms.Button();
            this.label_a_beg = new System.Windows.Forms.Label();
            this.textBox_a_beg = new System.Windows.Forms.TextBox();
            this.label_a_end = new System.Windows.Forms.Label();
            this.textBox_a_end = new System.Windows.Forms.TextBox();
            this.label_a_step = new System.Windows.Forms.Label();
            this.textBox_a_step = new System.Windows.Forms.TextBox();
            this.label_r = new System.Windows.Forms.Label();
            this.textBox_r = new System.Windows.Forms.TextBox();
            this.button_close = new System.Windows.Forms.Button();
            this.richTextBox_result
                    = new System.Windows.Forms.RichTextBox();
            this.panel_inp.SuspendLayout();
            this.panel_out.SuspendLayout();
            this.groupBox_input.SuspendLayout();
            this.SuspendLayout();
            //
            // panel_inp
            //
            this.panel_inp.Controls.Add(this.groupBox_input);
            this.panel_inp.Dock = System.Windows.Forms.DockStyle.Left;
            this.panel_inp.Location = new System.Drawing.Point(0, 0);
            this.panel_inp.Name = "panel_inp";
            this.panel_inp.Size = new System.Drawing.Size(296, 242);
            this.panel_inp.TabIndex = 0;
```

```
//
// panel_out
//
this.panel_out.Controls.Add(this.richTextBox_result);
this.panel_out.Controls.Add(this.button_close);
this.panel_out.Dock = System.Windows.Forms.DockStyle.Fill;
this.panel_out.Location = new System.Drawing.Point(296, 0);
this.panel_out.Name = "panel_out";
this.panel_out.Size = new System.Drawing.Size(200, 242);
this.panel_out.TabIndex = 1;
//
// groupBox_input
//
this.groupBox_input.Controls.Add(this.textBox_r);
this.groupBox_input.Controls.Add(this.label_r);
this.groupBox_input.Controls.Add(this.textBox_a_step);
this.groupBox_input.Controls.Add(this.label_a_step);
this.groupBox_input.Controls.Add(this.textBox_a_end);
this.groupBox_input.Controls.Add(this.label_a_end);
this.groupBox_input.Controls.Add(this.textBox_a_beg);
this.groupBox_input.Controls.Add(this.label_a_beg);
this.groupBox_input.Controls.Add(this.buttonCalc);
this.groupBox_input.Dock
        = System.Windows.Forms.DockStyle.Fill;
this.groupBox_input.Location
        = new System.Drawing.Point(0, 0);
this.groupBox_input.Name = "groupBox_input";
this.groupBox_input.Size = new System.Drawing.Size(296, 242);
this.groupBox_input.TabIndex = 0;
this.groupBox_input.TabStop = false;
this.groupBox_input.Text
        = "Розрахунок площі колового сегменту";
//
// buttonCalc
//
this.buttonCalc.Dock = System.Windows.Forms.DockStyle.Bottom;
this.buttonCalc.Location = new System.Drawing.Point(3, 216);
this.buttonCalc.Name = "buttonCalc";
this.buttonCalc.Size = new System.Drawing.Size(290, 23);
this.buttonCalc.TabIndex = 0;
this.buttonCalc.Text = "Розрахувати площу";
this.buttonCalc.UseVisualStyleBackColor = true;
this.buttonCalc.Click
        += new System.EventHandler(this.buttonCalc_Click);
//
// label_a_beg
//
this.label_a_beg.AutoSize = true;
this.label_a_beg.Dock = System.Windows.Forms.DockStyle.Top;
this.label_a_beg.Location = new System.Drawing.Point(3, 16);
this.label_a_beg.Name = "label_a_beg";
this.label_a_beg.Size = new System.Drawing.Size(143, 13);
this.label_a_beg.TabIndex = 1;
this.label_a_beg.Text = "Початковий кут a, градусів";
//
// textBox_a_beg
//
this.textBox_a_beg.Dock = System.Windows.Forms.DockStyle.Top;
this.textBox_a_beg.Location
        = new System.Drawing.Point(3, 29);
this.textBox_a_beg.Name = "textBox_a_beg";
this.textBox_a_beg.Size = new System.Drawing.Size(290, 20);
this.textBox_a_beg.TabIndex = 2;
//
// label_a_end
//
this.label_a_end.AutoSize = true;
```

```
this.label_a_end.Dock = System.Windows.Forms.DockStyle.Top;
this.label_a_end.Location = new System.Drawing.Point(3, 49);
this.label_a_end.Name = "label_a_end";
this.label_a_end.Size = new System.Drawing.Size(128, 13);
this.label_a_end.TabIndex = 3;
this.label_a_end.Text = "Кінцевий кут a, градусів";
//
// textBox_a_end
//
this.textBox_a_end.Dock = System.Windows.Forms.DockStyle.Top;
this.textBox_a_end.Location
        = new System.Drawing.Point(3, 62);
this.textBox_a_end.Name = "textBox_a_end";
this.textBox_a_end.Size = new System.Drawing.Size(290, 20);
this.textBox_a_end.TabIndex = 4;
//
// label_a_step
//
this.label_a_step.AutoSize = true;
this.label_a_step.Dock = System.Windows.Forms.DockStyle.Top;
this.label_a_step.Location = new System.Drawing.Point(3, 82);
this.label_a_step.Name = "label_a_step";
this.label_a_step.Size = new System.Drawing.Size(145, 13);
this.label_a_step.TabIndex = 5;
this.label_a_step.Text = "Крок зміни кута a, градусів";
//
// textBox_a_step
//

this.textBox_a_step.Dock
        = System.Windows.Forms.DockStyle.Top;
this.textBox_a_step.Location
        = new System.Drawing.Point(3, 95);
this.textBox_a_step.Name = "textBox_a_step";
this.textBox_a_step.Size = new System.Drawing.Size(290, 20);
this.textBox_a_step.TabIndex = 6;
//
// label_r
//
this.label_r.AutoSize = true;
this.label_r.Dock = System.Windows.Forms.DockStyle.Top;
this.label_r.Location = new System.Drawing.Point(3, 115);
this.label_r.Name = "label_r";
this.label_r.Size = new System.Drawing.Size(67, 13);
this.label_r.TabIndex = 7;
this.label_r.Text = "Радіус r, мм";
//
// textBox_r
//
this.textBox_r.Dock = System.Windows.Forms.DockStyle.Top;
this.textBox_r.Location = new System.Drawing.Point(3, 128);
this.textBox_r.Name = "textBox_r";
this.textBox_r.Size = new System.Drawing.Size(290, 20);
this.textBox_r.TabIndex = 8;
//
// button_close
//
this.button_close.Dock
        = System.Windows.Forms.DockStyle.Bottom;
this.button_close.Location
        = new System.Drawing.Point(0, 219);
this.button_close.Name = "button_close";
this.button_close.Size = new System.Drawing.Size(200, 23);
this.button_close.TabIndex = 0;
this.button_close.Text = "Закрити програму";
this.button_close.UseVisualStyleBackColor = true;
this.button_close.Click
```

```
                += new System.EventHandler(this.button_close_Click);
            //
            // richTextBox_result
            //
            this.richTextBox_result.Dock
                    = System.Windows.Forms.DockStyle.Fill;
            this.richTextBox_result.Location
                    = new System.Drawing.Point(0, 0);
            this.richTextBox_result.Name
                    = "richTextBox_result";
            this.richTextBox_result.Size
                    = new System.Drawing.Size(200, 219);
            this.richTextBox_result.TabIndex = 1;
            this.richTextBox_result.Text = "";
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(496, 242);
            this.Controls.Add(this.panel_out);
            this.Controls.Add(this.panel_inp);
            this.Name = "Form1";
            this.Text = "Площа сегменту";
            this.panel_inp.ResumeLayout(false);
            this.panel_out.ResumeLayout(false);
            this.groupBox_input.ResumeLayout(false);
            this.groupBox_input.PerformLayout();
            this.ResumeLayout(false);

        }

        #endregion


        private System.Windows.Forms.Panel panel_inp;
        private System.Windows.Forms.GroupBox groupBox_input;
        private System.Windows.Forms.Label label_a_beg;
        private System.Windows.Forms.Button buttonCalc;
        private System.Windows.Forms.Panel panel_out;
        private System.Windows.Forms.TextBox textBox_r;
        private System.Windows.Forms.Label label_r;
        private System.Windows.Forms.TextBox textBox_a_step;
        private System.Windows.Forms.Label label_a_step;
        private System.Windows.Forms.TextBox textBox_a_end;
        private System.Windows.Forms.Label label_a_end;
        private System.Windows.Forms.TextBox textBox_a_beg;
        private System.Windows.Forms.RichTextBox richTextBox_result;
        private System.Windows.Forms.Button button_close;
    }
}
```

Лістинг 4.5

Приклад коду C#

```
//Програмний код файлу Form1.cs, в якому міститься написаний код

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp_Площа_сегмента
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonCalc_Click(object sender, EventArgs e)
        {
            double r, s;
            int a_beg, a_end, a_step, a;
            string str1="", str2="";
            //Перетворює рядкове представлення числа в ціле
            a_beg = int.Parse(textBox_a_beg.Text);
            a_end = int.Parse(textBox_a_end.Text);
            a_step = int.Parse(textBox_a_step.Text);
            //Перетворює рядкове представлення числа в дійсне
            r = double.Parse(textBox_r.Text);
            a = a_beg;
            //Додає текст, перетворює дійсне значення в рядкове
            //та додає символ кінця рядка
            str1 = "Для довжини радіусу r = " + r.ToString("F") + " ";
            richTextBox_result.AppendText(str1);
            while (a <= a_end)
            {
                //Розраховує площу колового сегменту
                s = 0.5 * Math.Pow(r, 2) * (a - Math.Sin(a));
                str2 =  "при значенні кута a = " + a.ToString("F")
                            + " розрахована площа колового сегменту S = "
                            + s.ToString("F") + " мм" ;
                richTextBox_result.AppendText("\r\n" + str2);
                a += a_step;
            }
            //Виводить діалогове вікно з повідомленням
            MessageBox.Show(str1+str2);
        }

        private void button_close_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

ПРАКТИЧНА РОБОТА №5

5 Створення меню та панелей інструментів у C++/CLI та C#

5.1 Додавання меню в додатках Windows Forms

Завдання 1.

Необхідно створити додаток, який розраховуватиме витрату потужності на перемішування для пропелерної мішалки. У якості вихідних даних повинні задаватись змінні за допомогою текстових полів.

Для введення вхідних даних додаємо до форми 4 об'єкти класу TextBox та 4 об'єкти класу Label для підпису полів введення. Даємо імена доданим об'єктам в залежності від змінних, які будуть вводитись: label_Kn; textBox_Kn; label_n; textBox_n; label_Rc; textBox_Rc; label_dm; textBox_dm.

У властивості Text об'єктів label записуємо пояснення до кожного поля введення. У файлі MyForm.h внаслідок цього додадуться наступні рядки коду у відповідних розділах:

Приклад коду C++/CLI

```
this->label_Kn->Text = L"Коефіцієнт Kn";
this->label_n->Text = L"Швидкість мішалки, об/с";
this->label_Rc->Text = L"Густина рідини, кг/м3";
this->label_dm->Text = L"Діаметр мішалки, м";
```

Надалі додаємо елемент головного меню MenuStrip до форми. За допомогою конструктора меню додаємо чотрири пункти головного меню та один роздільник (у властивості Text для цього об'єкту ставиться символ «-») - об'єкти класу ToolStripMenuItem. Назви пунктів меню зберігаються у властивості Text об'єктів ToolStripMenuItem. Щоб додати до пунктів меню сполучення швидкіх клавіш для можливості виклику події обробки вибору певного пункту меню, необхідно скористатись властивістю ShortcutKeys об'єктів ToolStripMenuItem. Для додавання піктограм підказок до пунктів меню слід у властивості Image обрати потрібний файл графічного зображення.

Щоб додати обробник подій для пункту меню необхідно двічі клацнути мишею по ньому у вікні редактора форми або обрати потрібний пункт меню та на вкладці Events (Події) вікна Властивості (Properties) та в розділі Action обрати подію Click і двічі клацнути у пустому полі праворуч (рис. 5.1). Як наслідок, з'явиться пуста функція обробника події Click для відповідного пункту меню, де в подальшому потрібно буде додати програмний код.

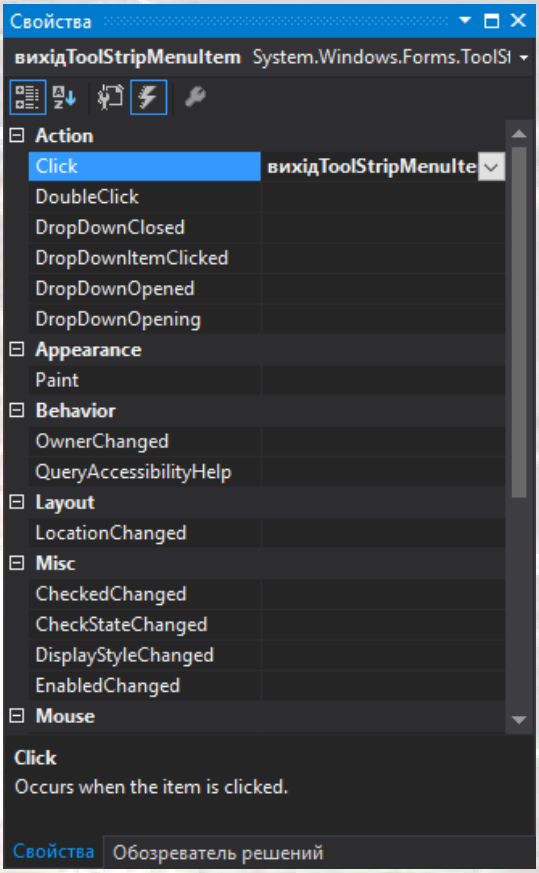


Рисунок 5.1 – Вкладка Events вікна Властивості з обробником події Click

Після виконання всіх цих дій, у файлі MyForm.h додасться автоматично наступний код:

Приклад коду C++/CLI

```
//
// menuStrip1
//
this->menuStrip1->Items->AddRange(gcnew cli::array<
     System::Windows::Forms::ToolStripItem^  >(1) { this-
>розрахунокToolStripMenuItem });
this->menuStrip1->Location = System::Drawing::Point(0, 0);
this->menuStrip1->Name = L"menuStrip1";
this->menuStrip1->Size = System::Drawing::Size(491, 24);
this->menuStrip1->TabIndex = 0;
this->menuStrip1->Text = L"menuStrip1";
//
// розрахунокToolStripMenuItem
//
this->розрахунокToolStripMenuItem->DropDownItems->
     AddRange(gcnew cli::array< System::Windows::Forms
          ::ToolStripItem^  >(5) {
     this->тестToolStripMenuItem,
     this->очиститиToolStripMenuItem,
     this->розрахуватиToolStripMenuItem,
     this->toolStripMenuItem1,
     this->вихідToolStripMenuItem
```

```
        });
    this->розрахунокToolStripMenuItem->Name = L"розрахунокToolStripMenuItem";
    this->розрахунокToolStripMenuItem->Size = System::Drawing::Size(82, 20);
    this->розрахунокToolStripMenuItem->Text = L"Розрахунок";
    //
    // тестToolStripMenuItem
    //
    this->тестToolStripMenuItem->Image =
    (cli::safe_cast<System::Drawing::Image^>
        (resources->GetObject(L"тестToolStripMenuItem.Image")));
    this->тестToolStripMenuItem->Name = L"тестToolStripMenuItem";
    this->тестToolStripMenuItem->ShortcutKeys =
        static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::
        Keys::Control | System::Windows::Forms::Keys::T));
    this->тестToolStripMenuItem->Size = System::Drawing::Size(206, 22);
    this->тестToolStripMenuItem->Text = L"&Тест";
    this->тестToolStripMenuItem->Click += gcnew System::EventHandler
        (this, &MyForm::тестToolStripMenuItem_Click);
    //
    // очиститиToolStripMenuItem
    //
    this->очиститиToolStripMenuItem->Image
        = (cli::safe_cast<System::Drawing::Image^>
        (resources->GetObject(L"очиститиToolStripMenuItem.Image")));
    this->очиститиToolStripMenuItem->Name = L"очиститиToolStripMenuItem";
    this->очиститиToolStripMenuItem->ShortcutKeys
        = static_cast<System::Windows::Forms
        ::Keys>((System::Windows::Forms::Keys::Alt |
        System::Windows::Forms::Keys::O));
    this->очиститиToolStripMenuItem->Size = System::Drawing::Size(206, 22);
    this->очиститиToolStripMenuItem->Text = L"&Очистити";
    this->очиститиToolStripMenuItem->Click += gcnew System::EventHandler
        (this, &MyForm::очиститиToolStripMenuItem_Click);
    //
    // розрахуватиToolStripMenuItem
    //
    this->розрахуватиToolStripMenuItem->Image =
        (cli::safe_cast<System::Drawing::Image^>
        (resources->GetObject(L"розрахуватиToolStripMenuItem.Image")));
    this->розрахуватиToolStripMenuItem->Name
        = L"розрахуватиToolStripMenuItem";
    this->розрахуватиToolStripMenuItem->ShortcutKeys
        = static_cast<System::Windows::Forms::Keys>
        (((System::Windows::Forms::Keys::Control |
        System::Windows::Forms::Keys::Alt) |
        System::Windows::Forms::Keys::C));
    this->розрахуватиToolStripMenuItem->Size
        = System::Drawing::Size(206, 22);
    this->розрахуватиToolStripMenuItem->Text = L"&Розрахувати";
    this->розрахуватиToolStripMenuItem->Click
        += gcnew System::EventHandler(this, &MyForm
        ::розрахуватиToolStripMenuItem_Click);
    //
```

```
// toolStripMenuItem1
//
this->toolStripMenuItem1->Name = L"toolStripMenuItem1";
this->toolStripMenuItem1->Size = System::Drawing::Size(203, 6);
//
// вихідToolStripMenuItem
//
this->вихідToolStripMenuItem->Image
     = (cli::safe_cast<System::Drawing::Image^>
     (resources->GetObject(L"вихідToolStripMenuItem.Image")));
this->вихідToolStripMenuItem->Name = L"вихідToolStripMenuItem";
this->вихідToolStripMenuItem->ShortcutKeys
     = static_cast<System::Windows::Forms::Keys>
     ((System::Windows::Forms::Keys::Alt |
     System::Windows::Forms::Keys::F4));
this->вихідToolStripMenuItem->Size = System::Drawing::Size(206, 22);
this->вихідToolStripMenuItem->Text = L"Ви&хід";
this->вихідToolStripMenuItem->Click += gcnew System
     ::EventHandler(this, &MyForm::вихідToolStripMenuItem_Click);
```

Аналогічний код автоматично генерується у C# у файлі Form1.Designer.cs. Нижче наведений фрагмент коду C# з файлу Form1.Designer.cs, який стосується доданого об'єкту головного меню menuStrip1 та команд (пунктів меню), які до нього входять.

Приклад коду C#

```
//
// menuStrip1
//
this.menuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
this.розрахунокToolStripMenuItem});
this.menuStrip1.Location = new System.Drawing.Point(0, 0);
this.menuStrip1.Name = "menuStrip1";
this.menuStrip1.Size = new System.Drawing.Size(200, 24);
this.menuStrip1.TabIndex = 2;
this.menuStrip1.Text = "menuStrip1";
//
// розрахунокToolStripMenuItem
//
this.розрахунокToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
this.тестToolStripMenuItem,
this.очиститиToolStripMenuItem,
this.розрахуватиToolStripMenuItem,
this.toolStripMenuItem1,
this.вихідToolStripMenuItem});
this.розрахунокToolStripMenuItem.Name = "розрахунокToolStripMenuItem";
this.розрахунокToolStripMenuItem.Size = new System.Drawing.Size(82, 20);
this.розрахунокToolStripMenuItem.Text = "Розрахунок";
//
// toolStripMenuItem1
```

```
//
this.toolStripMenuItem1.Name = "toolStripMenuItem1";
this.toolStripMenuItem1.Size = new System.Drawing.Size(203, 6);
//
// вихідToolStripMenuItem
//
this.вихідToolStripMenuItem.Image =
global::WindowsFormsApp_Площа_сегмента.Properties.Resources.screen;
this.вихідToolStripMenuItem.Name = "вихідToolStripMenuItem";
this.вихідToolStripMenuItem.ShortcutKeys =
((System.Windows.Forms.Keys)((System.Windows.Forms.Keys.Alt |
System.Windows.Forms.Keys.F4)));
this.вихідToolStripMenuItem.Size = new System.Drawing.Size(206, 22);
this.вихідToolStripMenuItem.Text = "Вихід";
//
// тестToolStripMenuItem
//
this.тестToolStripMenuItem.Image =
global::WindowsFormsApp_Площа_сегмента.Properties.Resources.cable;
this.тестToolStripMenuItem.Name = "тестToolStripMenuItem";
this.тестToolStripMenuItem.ShortcutKeys =
((System.Windows.Forms.Keys)((System.Windows.Forms.Keys.Control |
System.Windows.Forms.Keys.T)));
this.тестToolStripMenuItem.Size = new System.Drawing.Size(206, 22);
this.тестToolStripMenuItem.Text = "Тест";
//
// очиститиToolStripMenuItem
//
this.очиститиToolStripMenuItem.Image =
global::WindowsFormsApp_Площа_сегмента.Properties.Resources.computing_clo
ud;
this.очиститиToolStripMenuItem.Name = "очиститиToolStripMenuItem";
this.очиститиToolStripMenuItem.ShortcutKeys =
((System.Windows.Forms.Keys)((System.Windows.Forms.Keys.Alt |
System.Windows.Forms.Keys.O)));
this.очиститиToolStripMenuItem.Size = new System.Drawing.Size(206, 22);
this.очиститиToolStripMenuItem.Text = "Очистити";
//
// розрахуватиToolStripMenuItem
//
this.розрахуватиToolStripMenuItem.Image =
global::WindowsFormsApp_Площа_сегмента.Properties.Resources.cpu;
this.розрахуватиToolStripMenuItem.Name = "розрахуватиToolStripMenuItem";
this.розрахуватиToolStripMenuItem.ShortcutKeys =
((System.Windows.Forms.Keys)(((System.Windows.Forms.Keys.Control |
System.Windows.Forms.Keys.Alt)
| System.Windows.Forms.Keys.C)));
this.розрахуватиToolStripMenuItem.Size = new System.Drawing.Size(206,
22);
this.розрахуватиToolStripMenuItem.Text = "Розрахувати";
```

Як можна побачити, структура та зміст цих фрагментів коду у C++ та C# є ідентичними. Тому надалі немає сенсу дублювати процес створення програмного додатку у C#.

Внаслідок виконання вище зазначених дій, редактор форми матиме наступний вигляд (рис. 5.2).

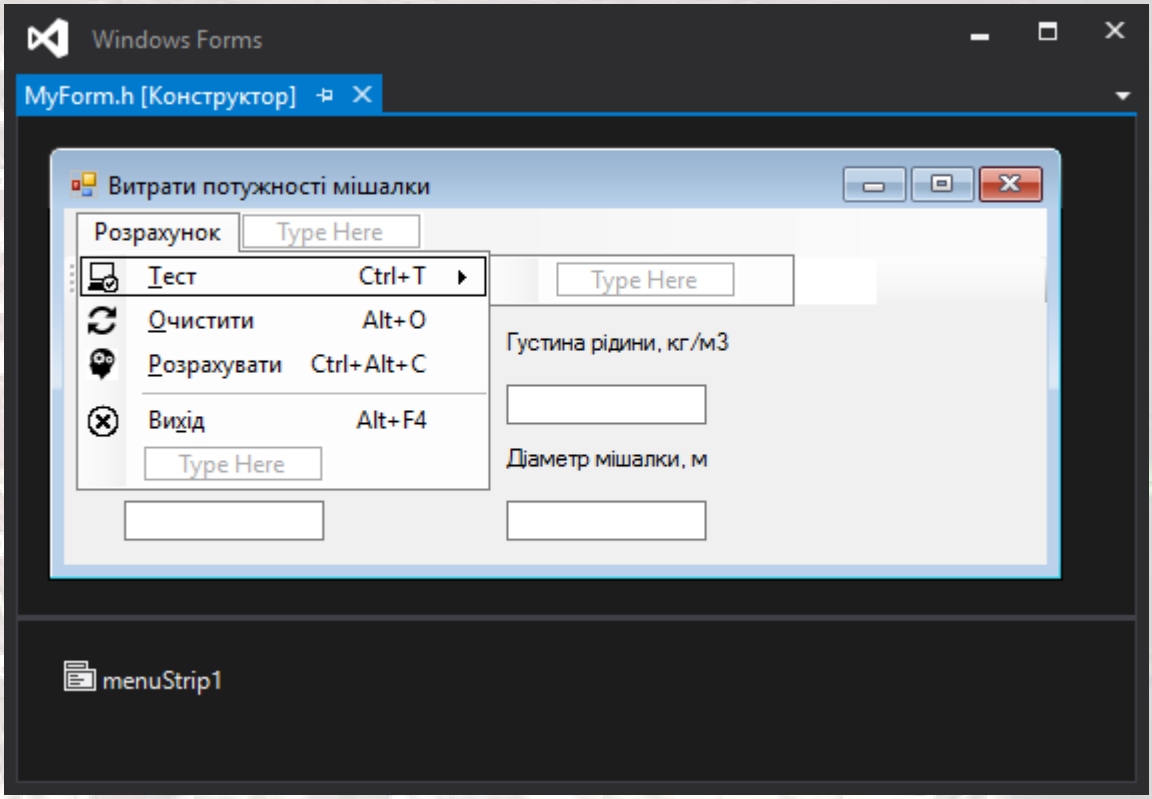


Рисунок 5.2 – Створення головного меню у вікні редактора форми

Для додавання контекстного меню необхідно розмістити у вікні редактора форми об'єкт класу ContextMenuStrip. Щоб контекстне меню відображалося після натискання правої кнопки миші у межах вікна програми, необхідно у властивості форми ContextMenuStrip обрати відповідний об'єкт контекстного меню.

Створення пунктів контекстного меню повністю аналогічне створенню пунктів головного меню. Пункти контекстного меню зазвичай можуть повторювати дії, що виконуються в головному меню. В цьому випадку для створення обробника події не потрібно заново писати той самий код. Для пункту контекстного меню треба перейти до вкладки Events (Події) вікна Властивості (Properties) та в розділі Action обрати подію Click. У пустому полі праворуч вже буде випадаючий перелік існуючих функцій обробки подій головного меню. Із цього переліку потрібно обрати відповідну функцію, що була створена для обробки події аналогічного пункту головного меню (рис. 5.3).

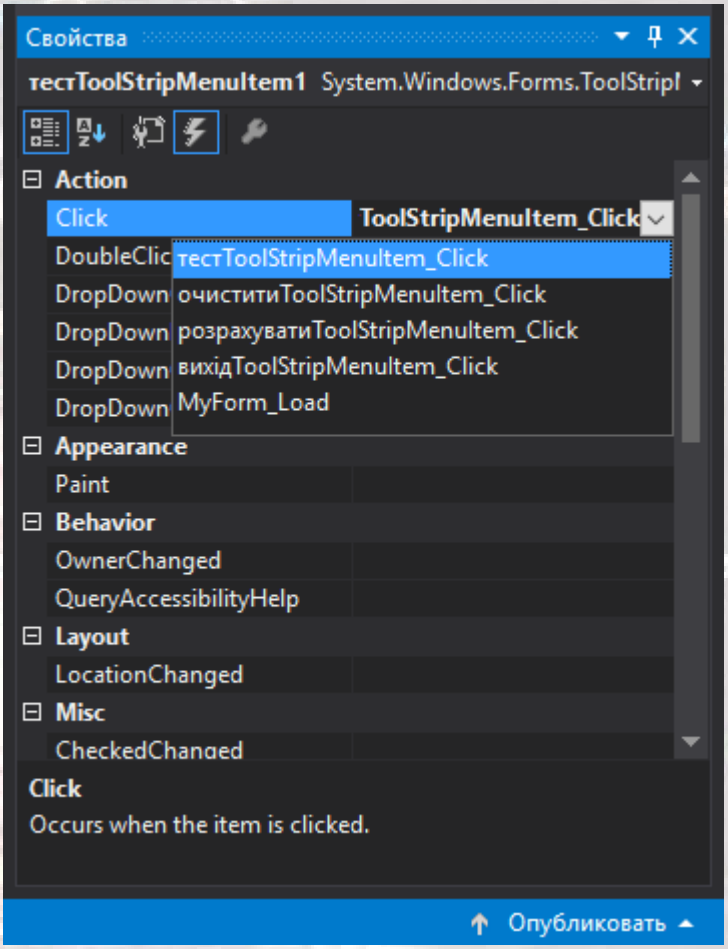


Рисунок 5.3 – Вибір обробника події для пункту меню

По завершенню створення пунктів контекстного меню, вікно редактора форми повинне мати вигляд, як зображено на рис. 5.4.

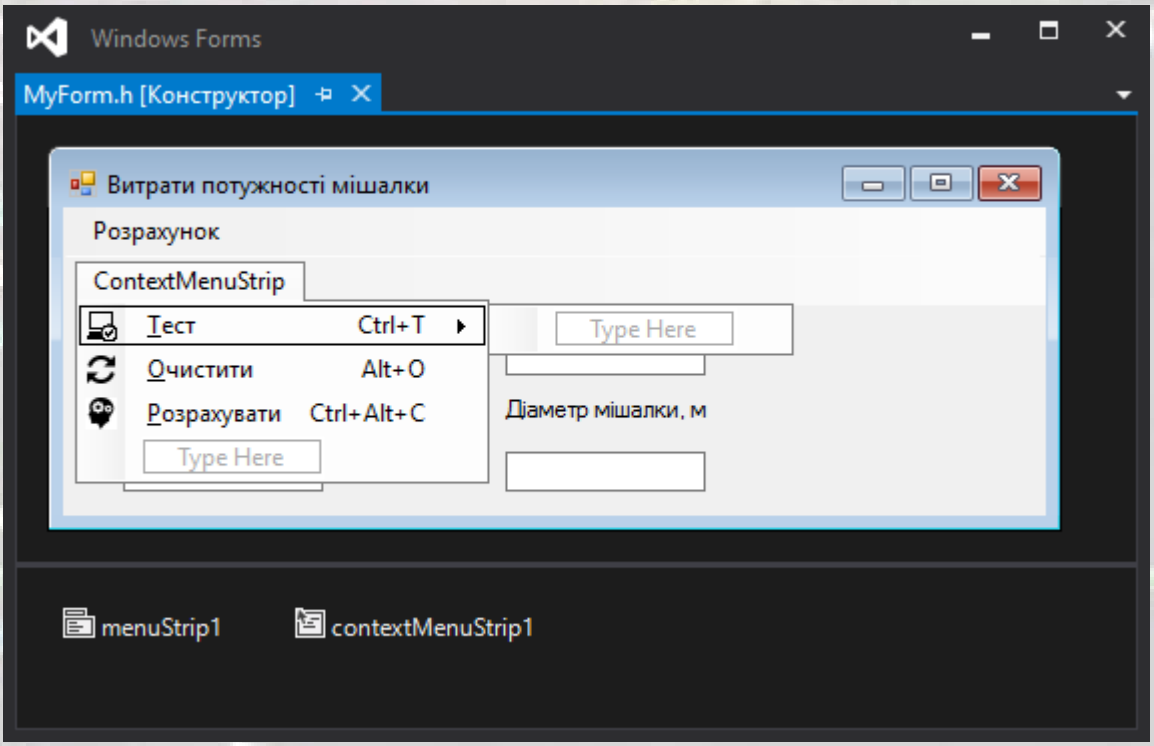


Рисунок 5.4 – Створення контекстного меню у вікні редактора форми

Після виконання всіх необхідних дій по створенню контекстного меню, до файлу MyForm.h додастся автоматично створений програмний код.

Приклад коду C++/CLI

```
//
// contextMenuStrip1
//
this->contextMenuStrip1->Items->AddRange(gcnew cli::array<
     System::Windows::Forms::ToolStripItem^  >(3) {
          this->тестToolStripMenuItem1,
          this->очиститиToolStripMenuItem1,
          this->розрахуватиToolStripMenuItem1
     });
this->contextMenuStrip1->Name = L"contextMenuStrip1";
this->contextMenuStrip1->Size = System::Drawing::Size(207, 70);
//
// тестToolStripMenuItem1
//
this->тестToolStripMenuItem1->Image
     = (cli::safe_cast<System::Drawing::Image^>
     (resources->GetObject(L"тестToolStripMenuItem1.Image")));
this->тестToolStripMenuItem1->Name = L"тестToolStripMenuItem1";
this->тестToolStripMenuItem1->ShortcutKeys
     = static_cast<System::Windows::Forms
     ::Keys>((System::Windows::Forms::Keys::Control |
     System::Windows::Forms::Keys::T));
this->тестToolStripMenuItem1->Size = System::Drawing::Size(206, 22);
this->тестToolStripMenuItem1->Text = L"&Тест";
this->тестToolStripMenuItem1->Click += gcnew System::EventHandler(this,
     &MyForm::тестToolStripMenuItem_Click);
//
// очиститиToolStripMenuItem1
//
this->очиститиToolStripMenuItem1->Image =
     (cli::safe_cast<System::Drawing::Image^>(resources-
     >GetObject(L"очиститиToolStripMenuItem1.Image")));
this->очиститиToolStripMenuItem1->Name = L"очиститиToolStripMenuItem1";
this->очиститиToolStripMenuItem1->ShortcutKeys =
     static_cast<System::Windows::Forms::Keys>((System::Windows::Forms
     ::Keys::Alt | System::Windows::Forms::Keys::O));
this->очиститиToolStripMenuItem1->Size = System::Drawing::Size(206, 22);
this->очиститиToolStripMenuItem1->Text = L"&Очистити";
this->очиститиToolStripMenuItem1->Click
     += gcnew System::EventHandler(this, &MyForm
     ::очиститиToolStripMenuItem_Click);
//
// розрахуватиToolStripMenuItem1
//
this->розрахуватиToolStripMenuItem1->Image =
     (cli::safe_cast<System::Drawing::Image^>(resources->
     GetObject(L"розрахуватиToolStripMenuItem1.Image")));
```

```
this->розрахуватиToolStripMenuItem1->Name
     = L"розрахуватиToolStripMenuItem1";
this->розрахуватиToolStripMenuItem1->ShortcutKeys =
     static_cast<System::Windows::Forms::Keys>(((System::Windows
     ::Forms::Keys::Control | System::Windows::Forms::Keys::Alt) |
     System::Windows::Forms::Keys::C));
this->розрахуватиToolStripMenuItem1->Size
     = System::Drawing::Size(206, 22);
this->розрахуватиToolStripMenuItem1->Text = L"&Розрахувати";
this->розрахуватиToolStripMenuItem1->Click +=
     gcnew System::EventHandler(this, &MyForm
     ::розрахуватиToolStripMenuItem_Click);
```

5.2 Додавання панелей інструментів в додатках Windows Forms

Надалі нам необхідно додати панель інструментів до вікна нашої програми. Для цього розміщуємо у вікні редактора форми об'єкт класу ToolStrip. Уредакторі панелі інструментів додаємо чотири кнопки – об'єкти класу ToolStripButton, один роздільник - об'єкт класу ToolStripSeparator, одне текстове поле – об'єкт класу ToolStripLabel та одне поле введення - об'єкт класу ToolStripTextBox. Чотири кнопки панелі інструментів будуть відповідати чотирьом пунктам головного меню, які буде відділяти роздільник від поля Label та поля TextBox. Об'єктам панелі інструментів надаємо імена в полі Name в залежності від їх призначення.

Для того, щоб на кнопках ToolStripButton панелі інструментів відображався окрім малюнка ще й текст, значення властивості DisplayStyle для чотирьох кнопок панелі інструментів у вікні Властивості (Properties) міняємо на ImageAndText. Після цього у властивості Text вказуємо текст, а у властивості Image обираємо малюнок, які будуть відображатися на кнопці.

Обробник натискання на кнопку панелі інструментів - подія Click розташована у вкладці Events (Події) вікна Властивості (Properties) в розділі Action. Оскільки обробники подій у нас вже створені для пунктів головного меню, у пустому полі праворуч події Click з випадаючого переліку існуючих функцій обробки подій обираємо потрібну. Дану операцію повторюємо для всіх чотирьох кнопок панелі інструментів.

По завершенню створення панелі інструментів до файлу MyForm.h автоматично буде додано наступний програмний код.

Приклад коду C++/CLI

```
//
// toolStrip1
//
this->toolStrip1->Items->AddRange(gcnew cli::array<
     System::Windows::Forms::ToolStripItem^  >(7) {
     this->toolStripButton_Тест,
     this->toolStripButton_Розрахунок,
     this->toolStripButton_Очищення,
```

```
        this->toolStripButton_Вихід,
        this->toolStripSeparator1,
        this->toolStripLabel_N,
        this->toolStripTextBox_N
        });
    this->toolStrip1->Location = System::Drawing::Point(0, 24);
    this->toolStrip1->Name = L"toolStrip1";
    this->toolStrip1->Size = System::Drawing::Size(491, 25);
    this->toolStrip1->TabIndex = 2;
    this->toolStrip1->Text = L"toolStrip1";
    //
    // toolStripButton_Тест
    //
    this->toolStripButton_Тест->Image
        = (cli::safe_cast<System::Drawing::Image^>
        (resources->GetObject(L"toolStripButton_Тест.Image")));
    this->toolStripButton_Тест->ImageTransparentColor
        = System::Drawing::Color::Magenta;
    this->toolStripButton_Тест->Name = L"toolStripButton_Тест";
    this->toolStripButton_Тест->Size = System::Drawing::Size(51, 22);
    this->toolStripButton_Тест->Text = L"Тест";
    this->toolStripButton_Тест->Click += gcnew System
        ::EventHandler(this, &MyForm::тестToolStripMenuItem_Click);
    //
    // toolStripButton_Розрахунок
    //
    this->toolStripButton_Розрахунок->Image
        = (cli::safe_cast<System::Drawing::Image^>
        (resources->GetObject(L"toolStripButton_Розрахунок.Image")));
    this->toolStripButton_Розрахунок->ImageTransparentColor
        = System::Drawing::Color::Magenta;
    this->toolStripButton_Розрахунок->Name = L"toolStripButton_Розрахунок";
    this->toolStripButton_Розрахунок->Size = System::Drawing::Size(94, 22);
    this->toolStripButton_Розрахунок->Text = L"Розрахувати";
    this->toolStripButton_Розрахунок->Click
        += gcnew System::EventHandler(this, &MyForm
        ::розрахуватиToolStripMenuItem_Click);
    //
    // toolStripButton_Очищення
    //
    this->toolStripButton_Очищення->Image = (cli::safe_cast<System::Drawing::
        Image^>(resources->GetObject(L"toolStripButton_Очищення.Image")));
    this->toolStripButton_Очищення->ImageTransparentColor
        = System::Drawing::Color::Magenta;
    this->toolStripButton_Очищення->Name = L"toolStripButton_Очищення";
    this->toolStripButton_Очищення->Size = System::Drawing::Size(80, 22);
    this->toolStripButton_Очищення->Text = L"Очистити";
    this->toolStripButton_Очищення->Click +=
        gcnew System::EventHandler(this,
    &MyForm::очиститиToolStripMenuItem_Click);
    //
    // toolStripButton_Вихід
    //
```

```
this->toolStripButton_Вихід->Image = (cli::safe_cast<System::Drawing::
    Image^>(resources->GetObject(L"toolStripButton_Вихід.Image")));
this->toolStripButton_Вихід->ImageTransparentColor
    = System::Drawing::Color::Magenta;
this->toolStripButton_Вихід->Name = L"toolStripButton_Вихід";
this->toolStripButton_Вихід->Size = System::Drawing::Size(55, 22);
this->toolStripButton_Вихід->Text = L"Вихід";
this->toolStripButton_Вихід->Click += gcnew System::EventHandler(this,
    &MyForm::вихідToolStripMenuItem_Click);
//
// toolStripSeparator1
//
this->toolStripSeparator1->Name = L"toolStripSeparator1";
this->toolStripSeparator1->Size = System::Drawing::Size(6, 25);
//
// toolStripLabel_N
//
this->toolStripLabel_N->Name = L"toolStripLabel_N";
this->toolStripLabel_N->Size = System::Drawing::Size(10, 22);
this->toolStripLabel_N->Text = L" ";
//
// toolStripTextBox_N
//
this->toolStripTextBox_N->Name = L"toolStripTextBox_N";
this->toolStripTextBox_N->Size = System::Drawing::Size(100, 25);
```

Після запуску вікно нашої програми з головним та контекстним меню і панеллю інструментів матиме вигляд, як зображено на рис. 5.5.

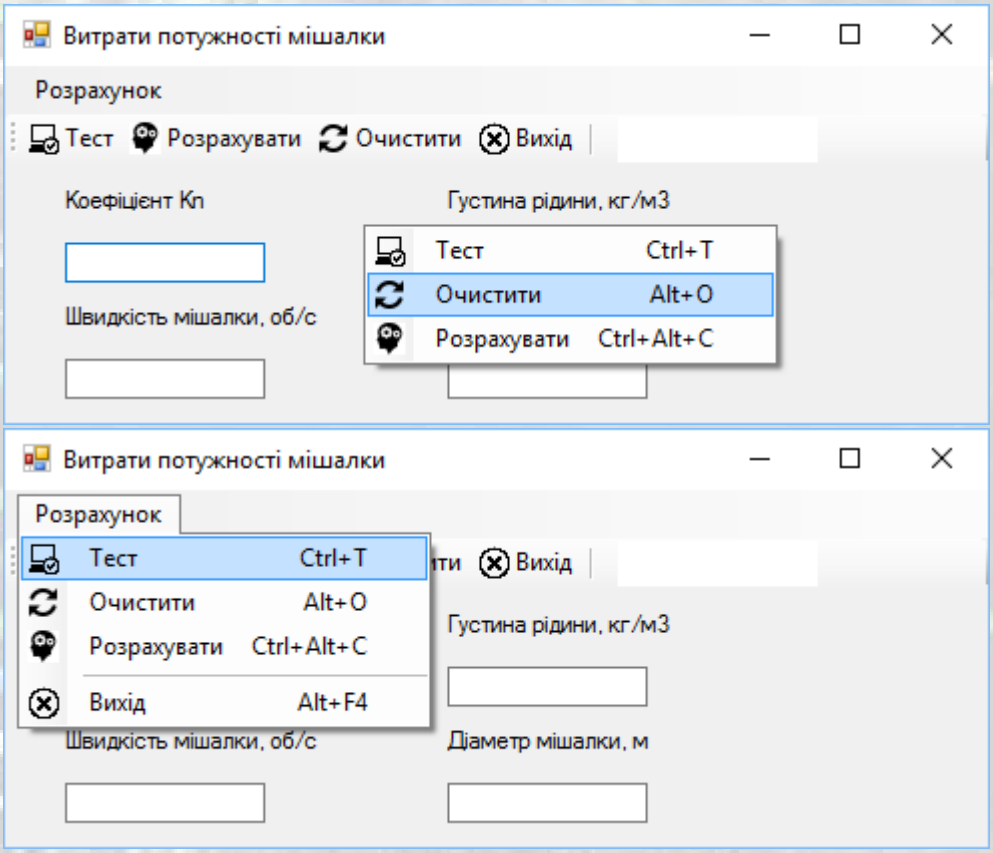


Рисунок 5.5 – Вікно програми з панеллю інструментів та меню

Для того, щоб обробники подій головного меню запрацювали, необхідно додати в них відповідний програмний код, як показано нижче.

Приклад коду C++/CLI

```
#pragma endregion

private: System::Void тестToolStripMenuItem_Click(System::Object^
sender, System::EventArgs^  e) {
     //Задавання значень змінних для тестового розрахунку
     textBox_Kn->Text = "0,32";
     textBox_n->Text = "3,96";
     textBox_Rc->Text = "1200";
     textBox_dm->Text = "2,8";
}
private: System::Void очиститиToolStripMenuItem_Click(System::Object^
sender, System::EventArgs^  e) {
     //Очищення поля введення
     textBox_Kn->Text = "";
     textBox_n->Text = "";
     textBox_Rc->Text = "";
     textBox_dm->Text = "";
     toolStripLabel_N->Text = "";
     toolStripTextBox_N->Text = "";
}
private: System::Void розрахуватиToolStripMenuItem_Click(System::Object^
sender, System::EventArgs^  e) {
     float Kn, n, Rc, dm, N;
     //Перетворює рядкове представлення числа в дійсне
     Kn = float::Parse(textBox_Kn->Text);
     n = float::Parse(textBox_n->Text);
     Rc = float::Parse(textBox_Rc->Text);
     dm = float::Parse(textBox_dm->Text);
     N = Kn*pow(n, 3)*Rc*pow(dm, 5);
     toolStripLabel_N->Text = "Витрати потужності на перемішування ";
     MessageBox::Show("Витрати потужності на перемішування = "
          + N.ToString("F"));
     toolStripTextBox_N->Text = N.ToString("F");
}
private: System::Void вихідToolStripMenuItem_Click(System::Object^
sender, System::EventArgs^  e) {
     Close(); //Закрити програму
}
```

Код, який додається у обробники подій C# (у файл форми Form1.cs за замовчуванням), матиме аналогічний вигляд.

Приклад коду C#

```
private void тестToolStripMenuItem_Click(object sender, EventArgs e)
{
     //Задавання значень змінних для тестового розрахунку
```

```
        textBox_Kn.Text = "0,32";
        textBox_n.Text = "3,96";
        textBox_Rc.Text = "1200";
        textBox_dm.Text = "2,8";
    }

    private void очиститиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        //Очищення поля введення
        textBox_Kn.Text = "";
        textBox_n.Text = "";
        textBox_Rc.Text = "";
        textBox_dm.Text = "";
        toolStripLabel_N.Text = "";
        toolStripTextBox_N.Text = "";
    }

    private void розрахуватиToolStripMenuItem_Click(object sender, EventArgs
    e)
    {
        double Kn, n, Rc, dm, N;
        //Перетворює рядкове представлення числа в дійсне
        Kn = double.Parse(textBox_Kn.Text);
        n = double.Parse(textBox_n.Text);
        Rc = double.Parse(textBox_Rc.Text);
        dm = double.Parse(textBox_dm.Text);
        N = Kn * Math.Pow(n, 3) * Rc * Math.Pow(dm, 5);
        toolStripLabel_N.Text = "Витрати потужності на перемішування ";
        MessageBox.Show("Витрати потужності на перемішування = "
            + N.ToString("F"));
        toolStripTextBox_N.Text = N.ToString("F");
    }

    private void вихідToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Close(); //Закрити програму
    }
```

Як видно з програмного коду обробки події розрахуватиToolStripMenuItem_Click, розраховане значення втрати потужності мішалки ми виводими в діалоговому вікні MessageBox, а також у полі toolStripTextBox_N з поясненням в toolStripLabel_N на панелі інструментів. Результат роботи програми наведений на рис. 5.6.

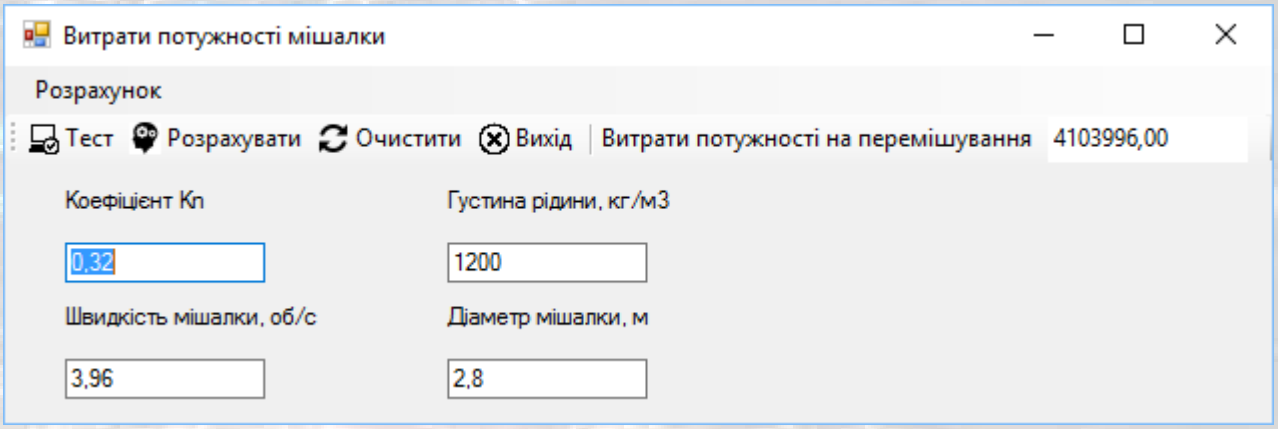


Рисунок 5.6 – Вікно програми з результатами розрахунку

Повний код створеного програмного модуля наведений нижче в лістингу 5.1 – файл `MyForm.cpp`, та лістингу 5.2 – заголовочний файл форми `MyForm.h`.

Програмний код

Лістинг 5.1

Приклад коду C++/CLI

```
//Програмний код модуля MyForm.cpp
#include "MyForm.h"

using namespace WindowsForms; //ім'я вашого проекту

[STAThreadAttribute]
int main(array<System::String ^> ^args)
{
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	Application::Run(gcnew MyForm());
	return 0;
}
```

Лістинг 5.2

Приклад коду C++/CLI

```
//Програмний код модуля MyForm.h
#pragma once

namespace WindowsForms {

   using namespace System;
   using namespace System::ComponentModel;
   using namespace System::Collections;
   using namespace System::Windows::Forms;
   using namespace System::Data;
   using namespace System::Drawing;
   #include <math.h> //Підключення бібліотеки математичних функцій

   /// <summary>
   /// Сводка для MyForm
   /// </summary>
   public ref class MyForm : public System::Windows::Forms::Form
   {
   public:
      MyForm(void)
      {
         InitializeComponent();
         //
         //TODO: добавьте код конструктора
         //
      }

   protected:
      /// <summary>
      /// Освободить все используемые ресурсы.
      /// </summary>
      ~MyForm()
      {
         if (components)
         {
            delete components;
         }
      }
   private: System::Windows::Forms::MenuStrip^  menuStrip1;
   private: System::Windows::Forms::ContextMenuStrip^  contextMenuStrip1;
   private: System::Windows::Forms::ToolStrip^  toolStrip1;
   private: System::Windows::Forms
      ::ToolStripButton^  toolStripButton_Тест;
   private: System::Windows::Forms
      ::ToolStripSeparator^  toolStripSeparator1;
   private: System::Windows::Forms
      ::ToolStripMenuItem^  розрахунокToolStripMenuItem;
   private: System::Windows::Forms
      ::ToolStripMenuItem^  тестToolStripMenuItem;
   private: System::Windows::Forms
      ::ToolStripMenuItem^  очиститиToolStripMenuItem;
   private: System::Windows::Forms
      ::ToolStripMenuItem^  розрахуватиToolStripMenuItem;
   private: System::Windows::Forms
      ::ToolStripSeparator^  toolStripMenuItem1;
   private: System::Windows::Forms
      ::ToolStripMenuItem^  вихідToolStripMenuItem;
   private: System::Windows::Forms
      ::ToolStripButton^  toolStripButton_Розрахунок;
   private: System::Windows::Forms
      ::ToolStripButton^  toolStripButton_Очищення;
```

```
        private: System::Windows::Forms
            ::ToolStripButton^  toolStripButton_Вихід;
        private: System::Windows::Forms
            ::ToolStripTextBox^  toolStripTextBox_N;
        private: System::Windows::Forms::Label^  label_Kn;
        private: System::Windows::Forms::TextBox^  textBox_Kn;
        private: System::Windows::Forms::TextBox^  textBox_n;
        private: System::Windows::Forms::Label^  label_n;
        private: System::Windows::Forms::TextBox^  textBox_Rc;
        private: System::Windows::Forms::Label^  label_Rc;
        private: System::Windows::Forms::TextBox^  textBox_dm;
        private: System::Windows::Forms::Label^  label_dm;
        private: System::Windows::Forms::ToolStripLabel^  toolStripLabel_N;
        private: System::Windows::Forms
            ::ToolStripMenuItem^  тестToolStripMenuItem1;
        private: System::Windows::Forms
            ::ToolStripMenuItem^  очиститиToolStripMenuItem1;
        private: System::Windows::Forms
            ::ToolStripMenuItem^  розрахуватиToolStripMenuItem1;
        private: System::ComponentModel::IContainer^  components;

        protected:

        private:
            /// <summary>
            /// Обязательная переменная конструктора.
            /// </summary>


    #pragma region Windows Form Designer generated code
            /// <summary>
            /// Требуемый метод для поддержки конструктора — не изменяйте
            /// содержимое этого метода с помощью редактора кода.
            /// </summary>
            void InitializeComponent(void)
            {
                this->components = (gcnew System::ComponentModel::Container());
                System::ComponentModel::ComponentResourceManager^  resources
                    = (gcnew System::ComponentModel
                    ::ComponentResourceManager(MyForm::typeid));
                this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());
                this->розрахунокToolStripMenuItem = (gcnew System::Windows
                    ::Forms::ToolStripMenuItem());
                this->тестToolStripMenuItem = (gcnew System::Windows
                    ::Forms::ToolStripMenuItem());
                this->очиститиToolStripMenuItem = (gcnew System::Windows
                    ::Forms::ToolStripMenuItem());
                this->розрахуватиToolStripMenuItem = (gcnew System::Windows
                    ::Forms::ToolStripMenuItem());
                this->toolStripMenuItem1 = (gcnew System::Windows
                    ::Forms::ToolStripSeparator());
                this->вихідToolStripMenuItem = (gcnew System::Windows
                    ::Forms::ToolStripMenuItem());
                this->contextMenuStrip1 = (gcnew System::Windows
                    ::Forms::ContextMenuStrip(this->components));
                this->тестToolStripMenuItem1 = (gcnew System::Windows
                    ::Forms::ToolStripMenuItem());
                this->очиститиToolStripMenuItem1 = (gcnew System::Windows
                    ::Forms::ToolStripMenuItem());
                this->розрахуватиToolStripMenuItem1 = (gcnew System::Windows
                    ::Forms::ToolStripMenuItem());
                this->toolStrip1 = (gcnew System::Windows::Forms::ToolStrip());
                this->toolStripButton_Тест = (gcnew System::Windows
                    ::Forms::ToolStripButton());
                this->toolStripButton_Розрахунок = (gcnew System::Windows
                    ::Forms::ToolStripButton());
                this->toolStripButton_Очищення = (gcnew System::Windows
```

```
    ::Forms::ToolStripButton());
this->toolStripButton_Вихід = (gcnew System::Windows
    ::Forms::ToolStripButton());
this->toolStripSeparator1 = (gcnew System::Windows
    ::Forms::ToolStripSeparator());
this->toolStripLabel_N = (gcnew System::Windows
    ::Forms::ToolStripLabel());
this->toolStripTextBox_N = (gcnew System::Windows
    ::Forms::ToolStripTextBox());
this->label_Kn = (gcnew System::Windows::Forms::Label());
this->textBox_Kn = (gcnew System::Windows::Forms::TextBox());
this->textBox_n = (gcnew System::Windows::Forms::TextBox());
this->label_n = (gcnew System::Windows::Forms::Label());
this->textBox_Rc = (gcnew System::Windows::Forms::TextBox());
this->label_Rc = (gcnew System::Windows::Forms::Label());
this->textBox_dm = (gcnew System::Windows::Forms::TextBox());
this->label_dm = (gcnew System::Windows::Forms::Label());
this->menuStrip1->SuspendLayout();
this->contextMenuStrip1->SuspendLayout();
this->toolStrip1->SuspendLayout();
this->SuspendLayout();
//
// menuStrip1
//
this->menuStrip1->Items->AddRange(gcnew cli::array< System
    ::Windows::Forms::ToolStripItem^  >(1) { this->
    розрахунокToolStripMenuItem });
this->menuStrip1->Location = System::Drawing::Point(0, 0);
this->menuStrip1->Name = L"menuStrip1";
this->menuStrip1->Size = System::Drawing::Size(491, 24);
this->menuStrip1->TabIndex = 0;
this->menuStrip1->Text = L"menuStrip1";
//
// розрахунокToolStripMenuItem
//
this->розрахунокToolStripMenuItem->DropDownItems->
  AddRange(gcnew cli::array< System::Windows::Forms
  ::ToolStripItem^  >(5) {
      this->тестToolStripMenuItem,
      this->очиститиToolStripMenuItem,
      this->розрахуватиToolStripMenuItem,
      this->toolStripMenuItem1,
      this->вихідToolStripMenuItem
  });
this->розрахунокToolStripMenuItem->Name
    = L"розрахунокToolStripMenuItem";
this->розрахунокToolStripMenuItem->Size
    = System::Drawing::Size(82, 20);
this->розрахунокToolStripMenuItem->Text = L"Розрахунок";
//
// тестToolStripMenuItem
//
this->тестToolStripMenuItem->Image = (cli::safe_cast<System
    ::Drawing::Image^>(resources->
    GetObject(L"тестToolStripMenuItem.Image")));
this->тестToolStripMenuItem->Name = L"тестToolStripMenuItem";
this->тестToolStripMenuItem->ShortcutKeys = static_cast<System
    ::Windows::Forms::Keys>((System::Windows::Forms
    ::Keys::Control | System::Windows::Forms::Keys::T));
this->тестToolStripMenuItem->Size
    = System::Drawing::Size(206, 22);
this->тестToolStripMenuItem->Text = L"&Тест";
this->тестToolStripMenuItem->Click += gcnew System
    ::EventHandler(this, &MyForm::тестToolStripMenuItem_Click);
//
// очиститиToolStripMenuItem
//
```

```
this->очиститиToolStripMenuItem->Image = (cli::safe_cast<System
   ::Drawing::Image^>(resources->
   GetObject(L"очиститиToolStripMenuItem.Image")));
this->очиститиToolStripMenuItem->Name
   = L"очиститиToolStripMenuItem";
this->очиститиToolStripMenuItem->ShortcutKeys = static_cast
   <System::Windows::Forms::Keys>((System::Windows
   ::Forms::Keys::Alt | System::Windows::Forms::Keys::O));
this->очиститиToolStripMenuItem->Size
   = System::Drawing::Size(206, 22);
this->очиститиToolStripMenuItem->Text = L"&Очистити";
this->очиститиToolStripMenuItem->Click += gcnew System
   ::EventHandler(this, &MyForm
   ::очиститиToolStripMenuItem_Click);
//
// розрахуватиToolStripMenuItem
//
this->розрахуватиToolStripMenuItem->Image = (cli
   ::safe_cast<System::Drawing::Image^>(resources->
   GetObject(L"розрахуватиToolStripMenuItem.Image")));
this->розрахуватиToolStripMenuItem->Name
   = L"розрахуватиToolStripMenuItem";
this->розрахуватиToolStripMenuItem->ShortcutKeys
   = static_cast<System::Windows::Forms
  ::Keys>(((System::Windows::Forms::Keys::Control |
  System::Windows::Forms::Keys::Alt)
  | System::Windows::Forms::Keys::C));
this->розрахуватиToolStripMenuItem->Size
   = System::Drawing::Size(206, 22);
this->розрахуватиToolStripMenuItem->Text = L"&Розрахувати";
this->розрахуватиToolStripMenuItem->Click += gcnew System
   ::EventHandler(this, &MyForm
   ::розрахуватиToolStripMenuItem_Click);
//
// toolStripMenuItem1
//
this->toolStripMenuItem1->Name = L"toolStripMenuItem1";
this->toolStripMenuItem1->Size = System::Drawing::Size(203, 6);
//
// вихідToolStripMenuItem
//
this->вихідToolStripMenuItem->Image
   = (cli::safe_cast<System::Drawing::Image^>(resources->
   GetObject(L"вихідToolStripMenuItem.Image")));
this->вихідToolStripMenuItem->Name = L"вихідToolStripMenuItem";
this->вихідToolStripMenuItem->ShortcutKeys = static_cast<System
   ::Windows::Forms::Keys>((System::Windows::Forms::Keys::Alt |
   System::Windows::Forms::Keys::F4));
this->вихідToolStripMenuItem->Size
   = System::Drawing::Size(206, 22);
this->вихідToolStripMenuItem->Text = L"Ви&хід";
this->вихідToolStripMenuItem->Click
   += gcnew System::EventHandler(this, &MyForm
   ::вихідToolStripMenuItem_Click);
/
// contextMenuStrip1
//
this->contextMenuStrip1->Items->AddRange(gcnew cli
   ::array< System::Windows::Forms::ToolStripItem^  >(3) {
      this->тестToolStripMenuItem1,
      this->очиститиToolStripMenuItem1,
      this->розрахуватиToolStripMenuItem1
});
this->contextMenuStrip1->Name = L"contextMenuStrip1";
this->contextMenuStrip1->Size = System::Drawing::Size(207, 70);
//
// тестToolStripMenuItem1
```

```
//
this->тестToolStripMenuItem1->Image
   = (cli::safe_cast<System::Drawing::Image^>(resources->
   GetObject(L"тестToolStripMenuItem1.Image")));
this->тестToolStripMenuItem1->Name = L"тестToolStripMenuItem1";
this->тестToolStripMenuItem1->ShortcutKeys
   = static_cast<System::Windows::Forms
  ::Keys>((System::Windows::Forms::Keys::Control |
  System::Windows::Forms::Keys::T));
this->тестToolStripMenuItem1->Size
   = System::Drawing::Size(206, 22);
this->тестToolStripMenuItem1->Text = L"&Тест";
this->тестToolStripMenuItem1->Click += gcnew System
   ::EventHandler(this, &MyForm::тестToolStripMenuItem_Click);
//
// очиститиToolStripMenuItem1
//
this->очиститиToolStripMenuItem1->Image = (cli::safe_cast<System
   ::Drawing::Image^>(resources->
   GetObject(L"очиститиToolStripMenuItem1.Image")));
this->очиститиToolStripMenuItem1->Name
   = L"очиститиToolStripMenuItem1";
this->очиститиToolStripMenuItem1->ShortcutKeys
   = static_cast<System::Windows::Forms
   ::Keys>((System::Windows::Forms::Keys::Alt |
   System::Windows::Forms::Keys::O));
this->очиститиToolStripMenuItem1->Size
   = System::Drawing::Size(206, 22);
this->очиститиToolStripMenuItem1->Text = L"&Очистити";
this->очиститиToolStripMenuItem1->Click += gcnew System::
   EventHandler(this, &MyForm::очиститиToolStripMenuItem_Click);
//
// розрахуватиToolStripMenuItem1
//
this->розрахуватиToolStripMenuItem1->Image = (cli
   ::safe_cast<System::Drawing::Image^>(resources->
   GetObject(L"розрахуватиToolStripMenuItem1.Image")));
this->розрахуватиToolStripMenuItem1->Name
   = L"розрахуватиToolStripMenuItem1";
this->розрахуватиToolStripMenuItem1->ShortcutKeys
   = static_cast<System::Windows::Forms
   ::Keys>(((System::Windows::Forms::Keys::Control |
   System::Windows::Forms::Keys::Alt) |
   System::Windows::Forms::Keys::C));
this->розрахуватиToolStripMenuItem1->Size
   = System::Drawing::Size(206, 22);
this->розрахуватиToolStripMenuItem1->Text = L"&Розрахувати";
this->розрахуватиToolStripMenuItem1->Click += gcnew System
   ::EventHandler(this, &MyForm
   ::розрахуватиToolStripMenuItem_Click);
//
// toolStrip1
//
this->toolStrip1->Items->AddRange(gcnew cli::array< System
   ::Windows::Forms::ToolStripItem^  >(7) {
      this->toolStripButton_Тест,
      this->toolStripButton_Розрахунок,
      this->toolStripButton_Очищення,
      this->toolStripButton_Вихід,
      this->toolStripSeparator1,
      this->toolStripLabel_N,
      this->toolStripTextBox_N
});
this->toolStrip1->Location = System::Drawing::Point(0, 24);
this->toolStrip1->Name = L"toolStrip1";
this->toolStrip1->Size = System::Drawing::Size(491, 25);
this->toolStrip1->TabIndex = 2;
```

```
this->toolStrip1->Text = L"toolStrip1";
//
// toolStripButton_Тест
//
this->toolStripButton_Тест->Image = (cli::safe_cast<System
   ::Drawing::Image^>(resources->
   GetObject(L"toolStripButton_Тест.Image")));
this->toolStripButton_Тест->ImageTransparentColor
   = System::Drawing::Color::Magenta;
this->toolStripButton_Тест->Name = L"toolStripButton_Тест";
this->toolStripButton_Тест->Size
   = System::Drawing::Size(51, 22);
this->toolStripButton_Тест->Text = L"Тест";
this->toolStripButton_Тест->Click += gcnew System
   ::EventHandler(this, &MyForm::тестToolStripMenuItem_Click);
//
// toolStripButton_Розрахунок
//
this->toolStripButton_Розрахунок->Image = (cli::safe_cast<System
   ::Drawing::Image^>(resources->
   GetObject(L"toolStripButton_Розрахунок.Image")));
this->toolStripButton_Розрахунок->ImageTransparentColor
   = System::Drawing::Color::Magenta;
this->toolStripButton_Розрахунок->Name
   = L"toolStripButton_Розрахунок";
this->toolStripButton_Розрахунок->Size
   = System::Drawing::Size(94, 22);
this->toolStripButton_Розрахунок->Text = L"Розрахувати";
this->toolStripButton_Розрахунок->Click += gcnew System
   ::EventHandler(this, &MyForm
   ::розрахуватиToolStripMenuItem_Click);
//
// toolStripButton_Очищення
//
this->toolStripButton_Очищення->Image = (cli::safe_cast<System
   ::Drawing::Image^>(resources->
   GetObject(L"toolStripButton_Очищення.Image")));
this->toolStripButton_Очищення->ImageTransparentColor
   = System::Drawing::Color::Magenta;
this->toolStripButton_Очищення->Name
   = L"toolStripButton_Очищення";
this->toolStripButton_Очищення->Size
   = System::Drawing::Size(80, 22);
this->toolStripButton_Очищення->Text = L"Очистити";
this->toolStripButton_Очищення->Click += gcnew System
   ::EventHandler(this, &MyForm
   ::очиститиToolStripMenuItem_Click);
//
// toolStripButton_Вихід
//
this->toolStripButton_Вихід->Image = (cli::safe_cast<System
   ::Drawing::Image^>(resources->
   GetObject(L"toolStripButton_Вихід.Image")));
this->toolStripButton_Вихід->ImageTransparentColor
   = System::Drawing::Color::Magenta;
this->toolStripButton_Вихід->Name = L"toolStripButton_Вихід";
this->toolStripButton_Вихід->Size
   = System::Drawing::Size(55, 22);
this->toolStripButton_Вихід->Text = L"Вихід";
this->toolStripButton_Вихід->Click += gcnew System
   ::EventHandler(this, &MyForm::вихідToolStripMenuItem_Click);
//
// toolStripSeparator1
//
this->toolStripSeparator1->Name = L"toolStripSeparator1";
this->toolStripSeparator1->Size = System::Drawing::Size(6, 25);
//
```

```
// toolStripLabel_N
//
this->toolStripLabel_N->Name = L"toolStripLabel_N";
this->toolStripLabel_N->Size = System::Drawing::Size(10, 22);
this->toolStripLabel_N->Text = L" ";
//
// toolStripTextBox_N
//
this->toolStripTextBox_N->Name = L"toolStripTextBox_N";
this->toolStripTextBox_N->Size = System::Drawing::Size(100, 25);
//
// label_Kn
//
this->label_Kn->AutoSize = true;
this->label_Kn->Location = System::Drawing::Point(27, 60);
this->label_Kn->Name = L"label_Kn";
this->label_Kn->Size = System::Drawing::Size(77, 13);
this->label_Kn->TabIndex = 3;
this->label_Kn->Text = L"Коефіцієнт Kn";
//
// textBox_Kn
//
this->textBox_Kn->Location = System::Drawing::Point(30, 88);
this->textBox_Kn->Name = L"textBox_Kn";
this->textBox_Kn->Size = System::Drawing::Size(100, 20);
this->textBox_Kn->TabIndex = 4;
//
// textBox_n
//
this->textBox_n->Location = System::Drawing::Point(30, 146);
this->textBox_n->Name = L"textBox_n";
this->textBox_n->Size = System::Drawing::Size(100, 20);
this->textBox_n->TabIndex = 6;
//
// label_n
//
this->label_n->AutoSize = true;
this->label_n->Location = System::Drawing::Point(27, 118);
this->label_n->Name = L"label_n";
this->label_n->Size = System::Drawing::Size(133, 13);
this->label_n->TabIndex = 5;
this->label_n->Text = L"Швидкість мішалки, об/с";
//
// textBox_Rc
//
this->textBox_Rc->Location = System::Drawing::Point(221, 88);
this->textBox_Rc->Name = L"textBox_Rc";
this->textBox_Rc->Size = System::Drawing::Size(100, 20);
this->textBox_Rc->TabIndex = 8;
//
// label_Rc
//
this->label_Rc->AutoSize = true;
this->label_Rc->Location = System::Drawing::Point(218, 60);
this->label_Rc->Name = L"label_Rc";
this->label_Rc->Size = System::Drawing::Size(118, 13);
this->label_Rc->TabIndex = 7;
this->label_Rc->Text = L"Густина рідини, кг/м3";
//
// textBox_dm
//
this->textBox_dm->Location = System::Drawing::Point(221, 146);
this->textBox_dm->Name = L"textBox_dm";
this->textBox_dm->Size = System::Drawing::Size(100, 20);
this->textBox_dm->TabIndex = 10;
//
// label_dm
```

```
            //
            this->label_dm->AutoSize = true;
            this->label_dm->Location = System::Drawing::Point(218, 118);
            this->label_dm->Name = L"label_dm";
            this->label_dm->Size = System::Drawing::Size(108, 13);
            this->label_dm->TabIndex = 9;
            this->label_dm->Text = L"Діаметр мішалки, м";
            //
            // MyForm
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode
               = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(491, 178);
            this->ContextMenuStrip = this->contextMenuStrip1;
            this->Controls->Add(this->textBox_dm);
            this->Controls->Add(this->label_dm);
            this->Controls->Add(this->textBox_Rc);
            this->Controls->Add(this->label_Rc);
            this->Controls->Add(this->textBox_n);
            this->Controls->Add(this->label_n);
            this->Controls->Add(this->textBox_Kn);
            this->Controls->Add(this->label_Kn);
            this->Controls->Add(this->toolStrip1);
            this->Controls->Add(this->menuStrip1);
            this->MainMenuStrip = this->menuStrip1;
            this->Name = L"MyForm";
            this->StartPosition = System::Windows
               ::Forms::FormStartPosition::CenterScreen;
            this->Text = L"Витрати потужності мішалки";
            this->menuStrip1->ResumeLayout(false);
            this->menuStrip1->PerformLayout();
            this->contextMenuStrip1->ResumeLayout(false);
            this->toolStrip1->ResumeLayout(false);
            this->toolStrip1->PerformLayout();
            this->ResumeLayout(false);
            this->PerformLayout();
        }

#pragma endregion

private: System::Void тестToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
       //Задавання значень змінних для тестового розрахунку
       textBox_Kn->Text = "0,32";
       textBox_n->Text = "3,96";
       textBox_Rc->Text = "1200";
       textBox_dm->Text = "2,8";
}

private: System::Void очиститиToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
       //Очищення поля введення
       textBox_Kn->Text = "";
       textBox_n->Text = "";
       textBox_Rc->Text = "";
       textBox_dm->Text = "";
       toolStripLabel_N->Text = "";
       toolStripTextBox_N->Text = "";
}

private: System::Void розрахуватиToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
       float Kn, n, Rc, dm, N;
       //Перетворює рядкове представлення числа в дійсне
       Kn = float::Parse(textBox_Kn->Text);
       n = float::Parse(textBox_n->Text);
```

```
            Rc = float::Parse(textBox_Rc->Text);
            dm = float::Parse(textBox_dm->Text);
            N = Kn*pow(n, 3)*Rc*pow(dm, 5);
            toolStripLabel_N->Text = "Витрати потужності на перемішування ";
            MessageBox::Show("Витрати потужності на перемішування = "
                  + N.ToString("F"));
            toolStripTextBox_N->Text = N.ToString("F");
      }

      private: System::Void вихідToolStripMenuItem_Click(System::Object^  sender,
      System::EventArgs^  e) {
            Close(); //Закрити програму
      }
      };
      }
```

ПРАКТИЧНА РОБОТА №6

6 Табличне введення даних у C++/CLI та C#

6.1 Введення табличних даних

Завдання 1.

Необхідно створити проект Windows Forms для відображення даних у табличному вигляді та збереження даних у файлі XML.

Програма, що розглядається в даному прикладі, пропонує вам заповнити таблицю телефонів знайомих, співробітників, родичів, коханих і т. д. Якщо клацнути на кнопці Запис дана таблиця записується на диск у файл у форматі XML. Для спрощення тексту програми передбачений запис в один і той же файл D:\tabl.xml. При наступних запусках даної програми таблиця буде зчитуватися з файлу, і ви зможете продовжити редагування таблиці. Тому цю програму можна голосно назвати «табличним редактором». Клацаючи на заголовках колонок, можна розташувати записи в колонках в алфавітному порядку для зручного пошуку необхідного телефону.

Для написання програми у C++/CLI запустимо Visual Studio і в вікні New Project виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms середовища Visual C++. Далі потрібно з панелі управління Toolbox перенести мишею наступні елементи управління: сітку даних DataGridView і кнопку Button. Текст програми приведений в наступному лістингу.

Приклад коду C++/CLI

```
#pragma endregion
    // Програма пропонує користувачеві заповнити таблицю телефонів його
    // знайомих, співробітників, родичів, коханих і т. д. Якщо клацнути
    // на кнопці Запис дана таблиця записується на диск у файл формату
    // XML. Для спрощення тексту програми передбачений запис в один і
    // той же файл C: \ tabl.xml. При наступних запусках даної програми
    // таблиця буде зчитуватися з цього файлу, і користувач може
    // продовжувати редагування таблиці
    // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
    DataTable^ Таблиця; // Об'явлення объекта "Таблиця"
    DataSet^ НабірДаних; // Об'явлення объекта "НабірДаних"
    private: System::Void Form1_Load(System::Object^ sender,
        System::EventArgs^ e)
        {
            this->Text = "Простий табличний редактор";
            button1->Text = "Запис";
            Таблиця = gcnew DataTable();
            НабірДаних = gcnew DataSet();
            if (IO::File::Exists("D:\\tabl.xml") == false)
            {
                // Якщо XML-файлу НЕМАЄ:
```

```
                    dataGridView1->DataSource = Таблиця;
                    // Заповнення "шапки" таблиці
                    Таблиця->Columns->Add("Імена");
                    Таблиця->Columns->Add("Номери телефонів");
                    // Додати об'єкт Таблиця в DataSet
                    НаборДаних->Tables->Add(Таблиця);
                }
                else // Якщо XML-файл Є:
                {
                    НаборДаних->ReadXml("D:\\tabl.xml");
                    // Вміст DataSet у вигляді рядка XML для відладки:
                    String ^ РядокXML = НаборДаних->GetXml();
                    dataGridView1->DataMember = "Назва таблиці";
                    dataGridView1->DataSource = НаборДаних;
                }
            }
        private: System::Void button1_Click(System::Object^ sender,
                System::EventArgs^ e)
                {
                    // Зберегти файл tabl.xml:
                    Таблиця->TableName = "Назва таблиці";
                    НаборДаних->WriteXml("D:\\tabl.xml");
                }
```

Для написання програми у C# запустимо Visual Studio і в вікні New Project виберемо у групі Visual C# розділ Windows Desktop додаток шаблону Windows Forms App (.NET Framework). Далі потрібно з панелі управління Toolbox перенести мишею наступні елементи управління: сітку даних DataGridView і кнопку Button. Текст програми приведений в наступному лістингу.

Приклад коду C#

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsAppMenu
{
   public partial class Form1 : Form
   {
      public Form1()
      {
            InitializeComponent();
      }
```

```
        // Програма пропонує користувачеві заповнити таблицю телефонів його
        // знайомих, співробітників, родичів, коханих і т. д. Якщо клацнути
        // на кнопці Запис дана таблиця записується на диск у файл формату
        // XML. Для спрощення тексту програми передбачений запис в один і
        // той же файл C: \ tabl.xml. При наступних запусках даної програми
        // таблиця буде зчитуватися з цього файлу, і користувач може
        // продовжувати редагування таблиці
        // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
        DataTable Таблиця; // Об'явлення объекта "Таблиця"
        DataSet НабірДаних; // Об'явлення объекта "НабірДаних"

        private void Form1_Load(object sender, EventArgs e)
        {
            this.Text = "Простий табличний редактор";
            button1.Text = "Запис";
            Таблиця = new DataTable();
            НабірДаних = new DataSet();
            if (System.IO.File.Exists("D:\\tabl.xml") == false)
            {
                // Якщо XML-файлу НЕМАЄ:
                dataGridView1.DataSource = Таблиця;
                // Заповнення "шапки" таблиці
                Таблиця.Columns.Add("Імена");
                Таблиця.Columns.Add("Номери телефонів");
                // Додати об'єкт Таблиця в DataSet
                НабірДаних.Tables.Add(Таблиця);
            }
            else // Якщо XML-файл Є:
            {
                НабірДаних.ReadXml("D:\\tabl.xml");
                // Вміст DataSet у вигляді рядка XML для відладки:
                String РядокXML = НабірДаних.GetXml();
                dataGridView1.DataMember = "Назва таблиці";
                dataGridView1.DataSource = НабірДаних;
            }

        }
        private void button1_Click(object sender, EventArgs e)
        {
            // Зберегти файл tabl.xml:
            Таблиця.TableName = "Назва таблиці";
            НабірДаних.WriteXml("D:\\tabl.xml");
        }
    }
}
```

Як видно з тексту програми, треба було всього лише кілька рядків програмного коду для створення такої багатофункціональної програми. Це стало можливим завдяки використанню потужної сучасної технології ADO.NET. На початку класу оголошені два об'єкти цієї технології: набір даних DataSet і таблиця даних DataTable. Об'єкт класу DataSet є основним компонентом

архітектури ADO.NET. DataSet представляє кеш даних, розташований в оперативній пам'яті. DataSet складається з колекції об'єктів класу DataTable. Тобто в один об'єкт класу DataSet може входити кілька таблиць, а інформацію про них ми можемо записувати в файл на диск одним оператором WriteXml, відповідно читати - ReadXML. Таким чином, в цій програмі ми маємо справу переважно з трьома об'єктами: DataSet - кеш даних, DataTable - представляє одну таблицю з даними та DataGridView - елемент управління для відображення даних.

Відразу після ініціалізації компонентів форми ми обробили дві ситуації. Якщо файлу, в який ми зберігаємо інформацію про таблиці, не існує Exists («D: \\ tabl.xml») == false, то призначаємо в якості джерела даних DataSource для DataGrid об'єкт класу DataTable і заповнюємо «шапку» таблиці, тобто вказуємо назви колонок: «Імена» і «Номери телефонів», а потім додаємо об'єкт DataTable в об'єкт DataSet. Тепер користувач бачить порожню таблицю з двома колонками і може почати її заповнення. Якщо файл існує (гілка else), то дані в об'єкт DataSet відправляємо з XML-файла (ReadXML). Тут уже як джерело даних для сітки даних DataGrid вказуємо об'єкт DataSet.

При натисканні мишею на кнопці Запис (рис. 6.1) - подія button1.Click - відбувається запис XML-файла на диск (WriteXml).

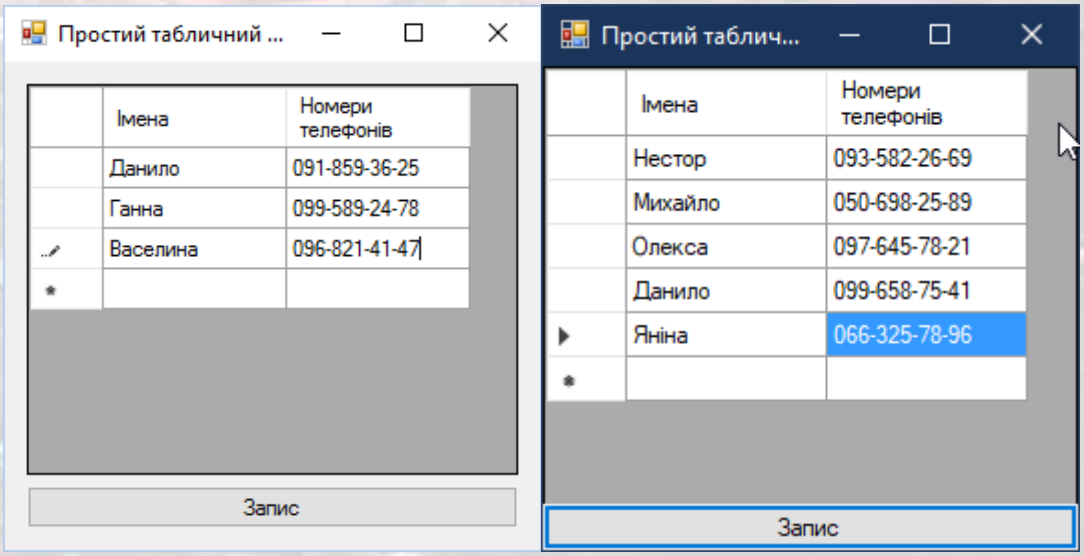


Рисунок 6.1 – Вікно програми з введеними даними в C++/CLI та C# (праворуч)

Тут використовуються, так звані, XML-файли. Формат цих файлів дозволяє легко і надійно передавати дані за допомогою Інтернету навіть на комп'ютери іншої платформи (наприклад, Macintosh). Ми не ставили перед собою мету працювати в Інтернеті, а всього лише скористалися цією технологією. Файл формату XML можна переглянути Блокнотом або за допомогою MS Word, оскільки це текстовий файл. Однак слід врахувати, що цей файл записаний в кодуванні UTF-8, тому іншими текстовими редакторами його прочитати важко. XML-документ відкривається веб-браузером, XML-editor (входить до складу Visual Studio), MS Office SharePoint Designer, MS Front Page і іншими програмами. При цьому застосування відступів і різних колірних рішень дозволяє

більш наочно продемонструвати структуру даного файлу. XML-файл можна відкрити табличним редактором MS Excel, і при цьому він може відобразитися у вигляді таблиці (рис. 6.2). На рис. 6.3 наведено зразок подання XML-файла в браузері.

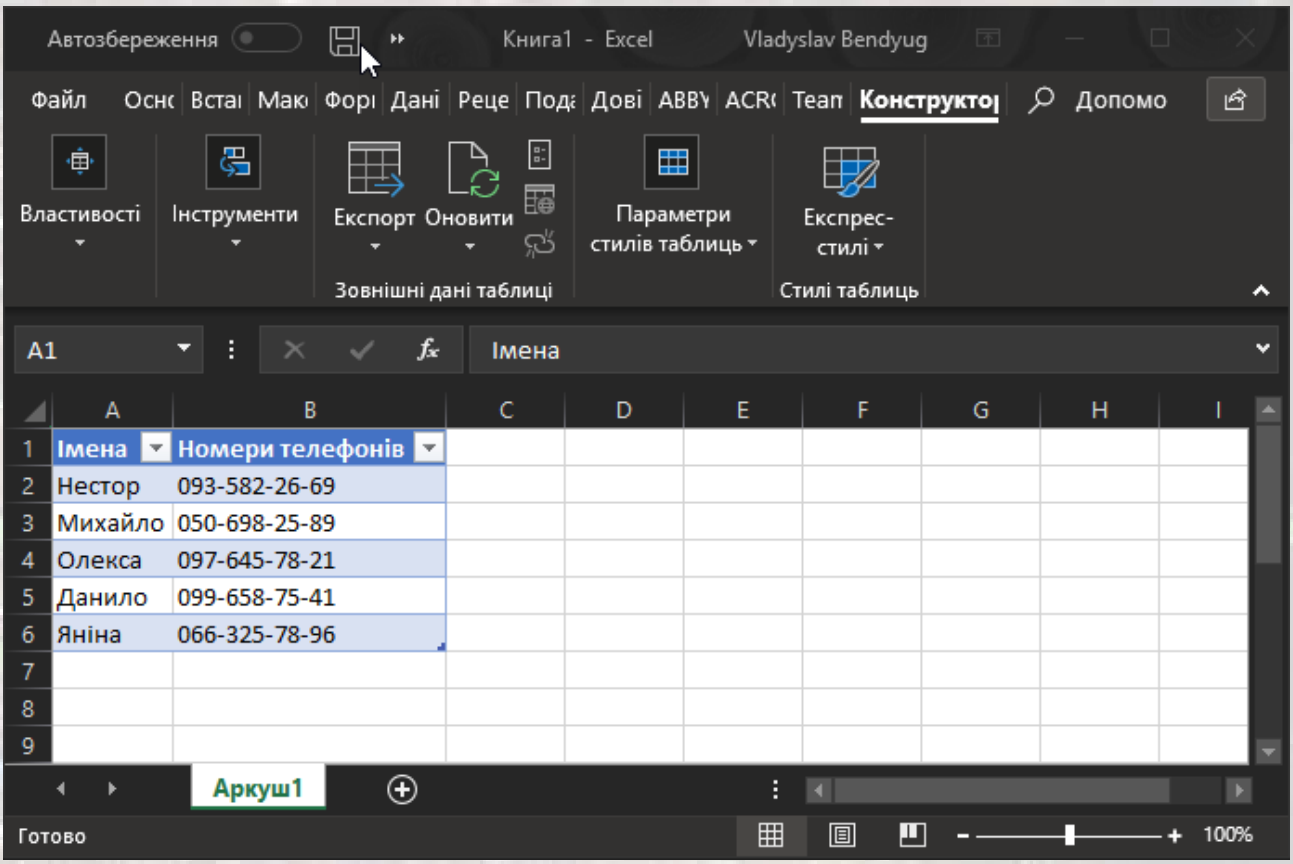


Рисунок 6.2 – Файл XML, який відкритий у MS Excel

При використанні нами XML-файлів для програмування найпростішого табличного редактора зовсім не обов'язково вникати в його структуру, тим більше при виборі імені файлу для збереження зовсім не обов'язково встановлювати розширення файлу xml, файл з будь-яким розширенням буде читатися методом ReadXml як XML-файл.

Програміст може отримати доступ до полів таблиці. Наприклад, доступ до лівої верхньої комірки (поля) таблиці можна отримати, використовуючи властивість об'єкта класу DataTable: Таблiця.Rows.Item(0).Item(0). Однак запис цього поля, наприклад, в послідовний файл буде некоректним навіть при використанні додаткової змінної через те, що технологія ADO.NET передбачає кешування даних. Таким чином, читання і запис даних для подібних таблиць слід організовувати тільки через методи об'єкта DataSet.

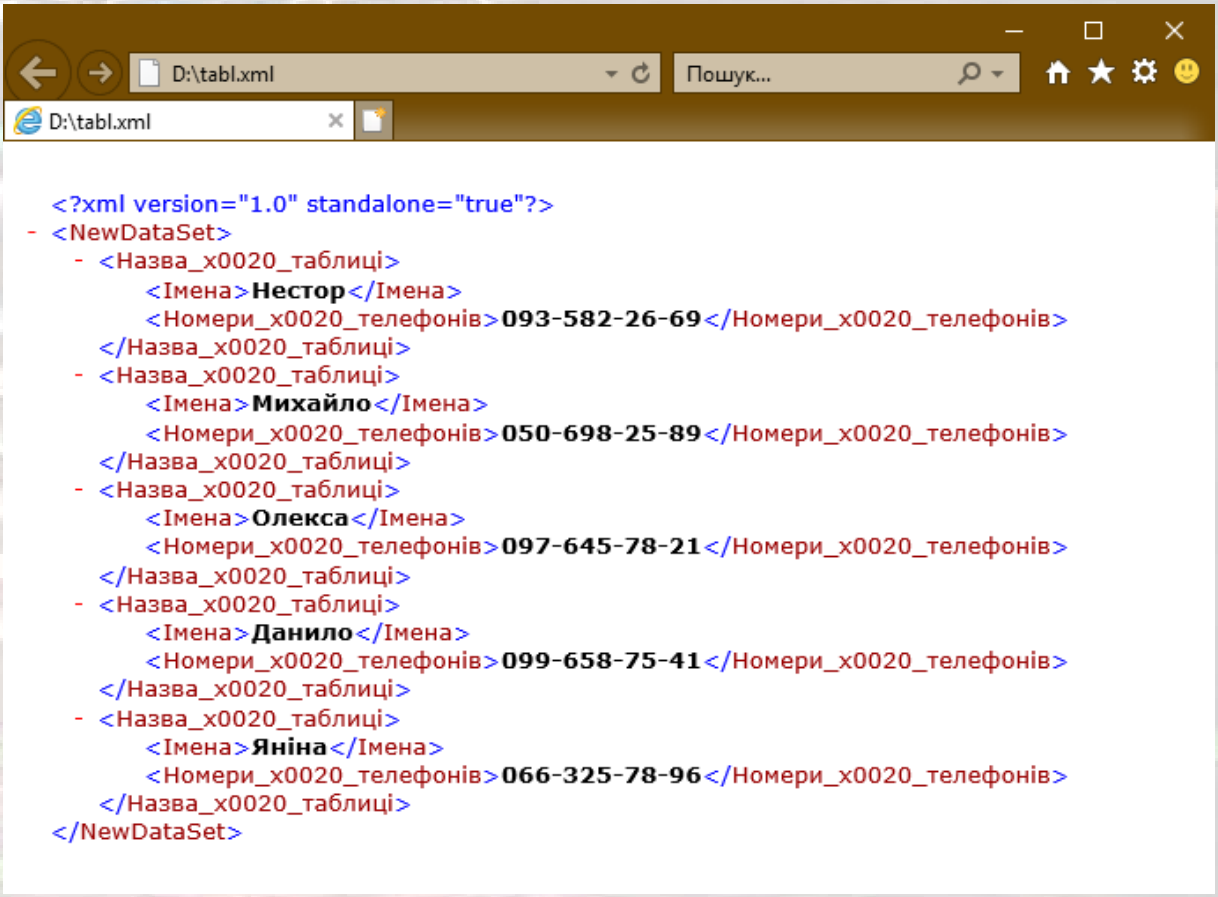


Рисунок 6.3 – Файл XML у вікні браузера

Відзначимо, що дана програма може також бути інструментом для створення XML-файлів. Текст програмного модуля у C++/CLI наведений в лістингу 6.1. Ще раз нагадаємо, що у C# файл коду дизайнера форми (лістинг 6.2) та файл коду програмного модуля форми (лістинг 6.3) містяться у різних файлах.

6.2 Побудова графіку з використанням елементу Chart

Завдання 2.

Необхідно створити проект Windows Forms який використовує елементи керування Chart і DataGridView та виводить графік (діаграму) залежнсті обсягів продажів від часу за місяцями.

Для вирішення цього завдання запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms. З Панелі елементів (ToolBox) перенесемо в проектовану екранну форму наступні елементи: діаграму Chart, сітку даних DataGridView та спадаючий перелік рядків ComboBox. У лістингу приведений програмний код вирішення задачі.

Приклад коду C++/CLI

```
private: System::Void MyForm_Load(System::Object^  sender,
System::EventArgs^  e) {
    this->Text = "Побудова діаграми";
```

```
DataTable ^ Таблиця = gcnew DataTable();
// У цій таблиці замовляємо дві колонки "Місяць"
// і "Обсяг продажів":
Таблиця->Columns->Add("Місяць", String::typeid);
// У C #: Табліца.Columns.Add("Місяць", typeof(String));
// Значення в другій колонці призначаємо типу long:
Таблиця->Columns->Add("Обсяг продажів", long::typeid);
// У C #: Табліца.Columns.Add("Обсяг продажів", typeof(long));
// Заповнити перший рядок (ряд) в таблиці:
DataRow ^ Ряд = Таблиця->NewRow();
Ряд["Місяць"] = "Травень"; Ряд["Обсяг продажів"] = 15;
Таблиця->Rows->Add(Ряд);
// Додаємо другий рядок в таблиці:
Ряд = Таблиця->NewRow();
Ряд["Місяць"] = "Червень"; Ряд["Обсяг продажів"] = 35;
Таблиця->Rows->Add(Ряд);
// Додаємо третій рядок:
Ряд = Таблиця->NewRow();
Ряд["Місяць"] = "Липень"; Ряд["Обсяг продажів"] = 65;
Таблиця->Rows->Add(Ряд);
// Додаємо четвертий рядок:
Ряд = Таблиця->NewRow();
Ряд["Місяць"] = "Серпень"; Ряд["Обсяг продажів"] = 85;
Таблиця->Rows->Add(Ряд);
// Додаємо п'ятий рядок:
Ряд = Таблиця->NewRow();
Ряд["Місяць"] = "Вересень"; Ряд["Обсяг продажів"] = 71;
Таблиця->Rows->Add(Ряд);

// Наведені нижче властивості можна задати у вікні Властивості
// для об'єкту chart1
// Складену таблицю вказуємо як джерело даних:
chart1->DataSource = Таблиця;
// На одному графіку можна зобразити кілька залежностей.
// Наприклад, перша залежність - обсяги продажів за вказаними
// Місяців в 2014 році, і друга залежність - продажу по
// тим же місяцям в 2015 році.
// В даному графіку ми покажемо тільки одну залежність, дані
// для відображення цієї залежності назвемо "Series1"
// На горизонтальній осі відкладаємо назви місяців:
chart1->Series["Series1"]->XValueMember = "Місяць";
// А по вертикальній осі відкладаємо обсяги продажів:
chart1->Series["Series1"]->YValueMembers = "Обсяг продажів";
// Назва графіка (діаграми):
chart1->Titles->Add("Обсяг продажів за місяцями");
// Задаємо тип діаграми - стовпчикова гістограма:
chart1->Series["Series1"]->ChartType = System::Windows::Forms::
    DataVisualization::Charting::SeriesChartType::Column;
// Тип діаграми може бути іншим, наприклад: Pie, Line і ін.
chart1->Series["Series1"]->Color = Color::Aqua;
// Легенду на графіку не відображаємо:
chart1->Series["Series1"]->IsVisibleInLegend = false;
// Прив'язка графіка до джерела даних:
```

```
        chart1->DataBind();
        // Для сітки даних вказати джерело даних
        dataGridView1->DataSource = Таблиця;

        // Задаємо перелік типів діаграм у об'єкті comboBox1
        // Це можна задати у вікні властивостей comboBox1 в полі Items
        comboBox1->Items->Add("Точкова");              //Point
        comboBox1->Items->Add("Графік");          //Line
        comboBox1->Items->Add("Згладжений графік"); //Spline
        comboBox1->Items->Add("Секторна");             //Pie
        comboBox1->Items->Add("Пелюсткова");      //Radar
        comboBox1->Items->Add("Гістограма");      //Bar
        comboBox1->Items->Add("Стовпчаста");      //Column
        comboBox1->Text = "Оберіть тип діаграми";

        // Встановлюємо вирівнювання об'єктів в межах форми
        chart1->Dock = System::Windows::Forms::DockStyle::Top;
        dataGridView1->Dock = System::Windows::Forms::DockStyle::Bottom;
        chart1->Height = this->Height - dataGridView1->Height - 40;
        comboBox1->Left = 0;
        comboBox1->Top = 0;
    }

    private: System::Void comboBox1_SelectedIndexChanged(System::Object^
    sender, System::EventArgs^  e) {
        // Змінюємо тип діаграми в залежності від обраного рядка
        // в об'єкті    comboBox1
        switch (comboBox1->SelectedIndex)
        {
            case 0:
                chart1->Series["Series1"]->ChartType = System::Windows
                    ::Forms::DataVisualization::Charting
                    ::SeriesChartType::Point;
                break;
            case 1:
                chart1->Series["Series1"]->ChartType = System::Windows
                    ::Forms::DataVisualization::Charting
                    ::SeriesChartType::Line;
                break;
            case 2:
                chart1->Series["Series1"]->ChartType = System::Windows
                    ::Forms::DataVisualization::Charting
                    ::SeriesChartType::Spline;
                break;
            case 3:
                chart1->Series["Series1"]->ChartType = System::Windows
                    ::Forms::DataVisualization::Charting
                    ::SeriesChartType::Pie;
                break;
            case 4:
                chart1->Series["Series1"]->ChartType = System::Windows
                    ::Forms::DataVisualization::Charting
                    ::SeriesChartType::Radar;
```

```
                break;
            case 5:
                chart1->Series["Series1"]->ChartType = System::Windows
                    ::Forms::DataVisualization::Charting
                    ::SeriesChartType::Bar;
                break;
            default:
                chart1->Series["Series1"]->ChartType = System::Windows
                    ::Forms::DataVisualization::Charting
                    ::SeriesChartType::Column;
                break;
        }
}

private: System::Void MyForm_Resize(System::Object^  sender,
System::EventArgs^  e) {
    // Підбір висоти об'єкту chart1 при зміні розміру вікна
    chart1->Height = this->Height - dataGridView1->Height - 40;
}
```

Для вирішення цього завдання засобами C#, запустимо Visual Studio і у вікні New Project виберемо у групі Visual C# розділ Windows Desktop додаток шаблону Windows Forms App (.NET Framework). З Панелі елементів (ToolBox) перенесемо в проектовану екранну форму наступні елементи: діаграму Chart, сітку даних DataGridView та спадаючий перелік рядків ComboBox. У лістингу приведений програмний код обробників трьох подій - Form1_Load(), comboBox1_SelectedIndexChanged() та Form1_Resize() у C#.

Приклад коду C#

```
private void Form1_Load(object sender, EventArgs e)
{
    this.Text = "Побудова діаграми";
    DataTable Таблиця = new DataTable();
    // У цій таблиці замовляємо дві колонки "Місяць"
    //і "Обсяг продажів":
    Таблиця.Columns.Add("Місяць", typeof(String));
    // У C #: Табліца.Columns.Add("Місяць", typeof(String));
    // Значення в другій колонці призначаємо типу long:
    Таблиця.Columns.Add("Обсяг продажів", typeof(long));
    // У C #: Табліца.Columns.Add("Обсяг продажів", typeof(long));
    // Заповнити перший рядок (ряд) в таблиці:
    DataRow Ряд = Таблиця.NewRow();
    Ряд["Місяць"] = "Травень"; Ряд["Обсяг продажів"] = 15;
    Таблиця.Rows.Add(Ряд);
    // Додаємо другий рядок в таблиці:
    Ряд = Таблиця.NewRow();
    Ряд["Місяць"] = "Червень"; Ряд["Обсяг продажів"] = 35;
    Таблиця.Rows.Add(Ряд);
    // Додаємо третій рядок:
```

```
Ряд = Таблиця.NewRow();
Ряд["Місяць"] = "Липень"; Ряд["Обсяг продажів"] = 65;
Таблиця.Rows.Add(Ряд);
// Додаємо четвертий рядок:
Ряд = Таблиця.NewRow();
Ряд["Місяць"] = "Серпень"; Ряд["Обсяг продажів"] = 85;
Таблиця.Rows.Add(Ряд);
// Додаємо п'ятий рядок:
Ряд = Таблиця.NewRow();
Ряд["Місяць"] = "Вересень"; Ряд["Обсяг продажів"] = 71;
Таблиця.Rows.Add(Ряд);

// Наведені нижче властивості можна задати у вікні Властивості
// для об'єкту chart1
// Складену таблицю вказуємо як джерело даних:
chart1.DataSource = Таблиця;
// На одному графіку можна зобразити кілька залежностей.
// Наприклад, перша залежність - обсяги продажів за вказаними
// Місяців в 2014 році, і друга залежність - продажу по
// тим же місяцям в 2015 році.
// В даному графіку ми покажемо тільки одну залежність, дані
// для відображення цієї залежності назвемо "Series1"
// На горизонтальній осі відкладаємо назви місяців:
chart1.Series["Series1"].XValueMember = "Місяць";
// А по вертикальній осі відкладаємо обсяги продажів:
chart1.Series["Series1"].YValueMembers = "Обсяг продажів";
// Назва графіка (діаграми):
chart1.Titles.Add("Обсяг продажів за місяцями");
// Задаємо тип діаграми - стовпчикова гістограма:
chart1.Series["Series1"].ChartType = System.Windows.Forms.
     DataVisualization.Charting.SeriesChartType.Column;
// Тип діаграми може бути іншим, наприклад: Pie, Line і ін.
chart1.Series["Series1"].Color = Color.Aqua;
// Легенду на графіку не відображаємо:
chart1.Series["Series1"].IsVisibleInLegend = false;
// Прив'язка графіка до джерела даних:
chart1.DataBind();
// Для сітки даних вказати джерело даних
dataGridView1.DataSource = Таблиця;

// Задаємо перелік типів діаграм у об'єкті comboBox1
// Це можна задати у вікні властивостей comboBox1 в полі Items
comboBox1.Items.Add("Точкова");           //Point
comboBox1.Items.Add("Графік");            //Line
comboBox1.Items.Add("Згладжений графік"); //Spline
comboBox1.Items.Add("Секторна");          //Pie
comboBox1.Items.Add("Пелюсткова");        //Radar
comboBox1.Items.Add("Гістограма");        //Bar
comboBox1.Items.Add("Стовпчаста");        //Column
comboBox1.Text = "Оберіть тип діаграми";

// Встановлюємо вирівнювання об'єктів в межах форми
chart1.Dock = System.Windows.Forms.DockStyle.Top;
```

```
        dataGridView1.Dock = System.Windows.Forms.DockStyle.Bottom;
        chart1.Height = this.Height - dataGridView1.Height - 40;
        comboBox1.Left = 0;
        comboBox1.Top = 0;

    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        // Змінюємо тип діаграми в залежності від обраного рядка в об'єкті
        comboBox1
        switch (comboBox1.SelectedIndex)
        {
            case 0:
                chart1.Series["Series1"].ChartType =
                System.Windows.Forms.
                    DataVisualization.Charting.SeriesChartType.Point;
                break;

            case 1:
                chart1.Series["Series1"].ChartType = System.
                    Windows.Forms.DataVisualization.Charting.
                    SeriesChartType.Line;
                break;

            case 2:
                chart1.Series["Series1"].ChartType = System.Windows.
                    Forms.DataVisualization.Charting.
                    SeriesChartType.Spline;
                break;

            case 3:
                chart1.Series["Series1"].ChartType = System.Windows.
                    Forms.DataVisualization.Charting.SeriesChartType.
                    Pie;
                break;

            case 4:
                chart1.Series["Series1"].ChartType = System.Windows.
                    Forms.DataVisualization.Charting.SeriesChartType.
                    Radar;
                break;

            case 5:
                chart1.Series["Series1"].ChartType = System.Windows.
                    Forms.DataVisualization.Charting.SeriesChartType.
                    Bar;
                break;

            default:
                chart1.Series["Series1"].ChartType = System.Windows.
                    Forms.DataVisualization.Charting.SeriesChartType.
                    Column;
```

```
                break;

            }
        }

        private void Form1_Resize(object sender, EventArgs e)
        {
            // Підбір висоти об'єкту chart1 при зміні розміру вікна
            chart1.Height = this.Height - dataGridView1.Height - 40;
        }
```

Значну частину з наведеного коду можна скоротити шляхом задавання значень властивостей у вікні Властивості (Propeties) під час проектування інтерфейсу. Це спрощує процес створення інтерфейсу і не вимагає ручного написання коду. Під час розробки інтерфейсу проекту потрібно максимально використовувати можливості вікна Властивості (Propeties). В наведеному коді всі властивості задаються та змінюються програмно для можливості швидкого відтворення прикладу шляхом простого вставлення вище наведеного коду у новий проект.

При створенні даної програми використано три події, реакцією на які є наведений програмний код: події Load та Resize форми і подія SelectedIndexChanged об’єкту ComboBox.

У програмному коді при обробці події завантаження форми оголошуємо об'єкт Таблиця класу DataTable. Цей об'єкт представляє одну таблицю даних в оперативній пам'яті. Щоб візуалізувати цю таблицю на екрані, використовують елемент - сітка даних DataGridView. Об'єкт класу DataTable використовують в якості вихідних даних і для сітки даних DataGridView, і для діаграми Chart.

У таблиці DataTable визначаємо її схему, замовляючи дві колонки «Місяць» та «Обсяг продажів». А далі заповнюємо таблицю по її рядках, використовуючи метод Add. Заповнену п'ятьма рядками таблицю вказуємо як джерело даних для елементів Chart і DataGridView. Далі оформляємо зовнішній вигляд діаграми, що детально представлено в коментарях до програмного коду.

Фрагмент роботи програми показаний на рис. 6.4.

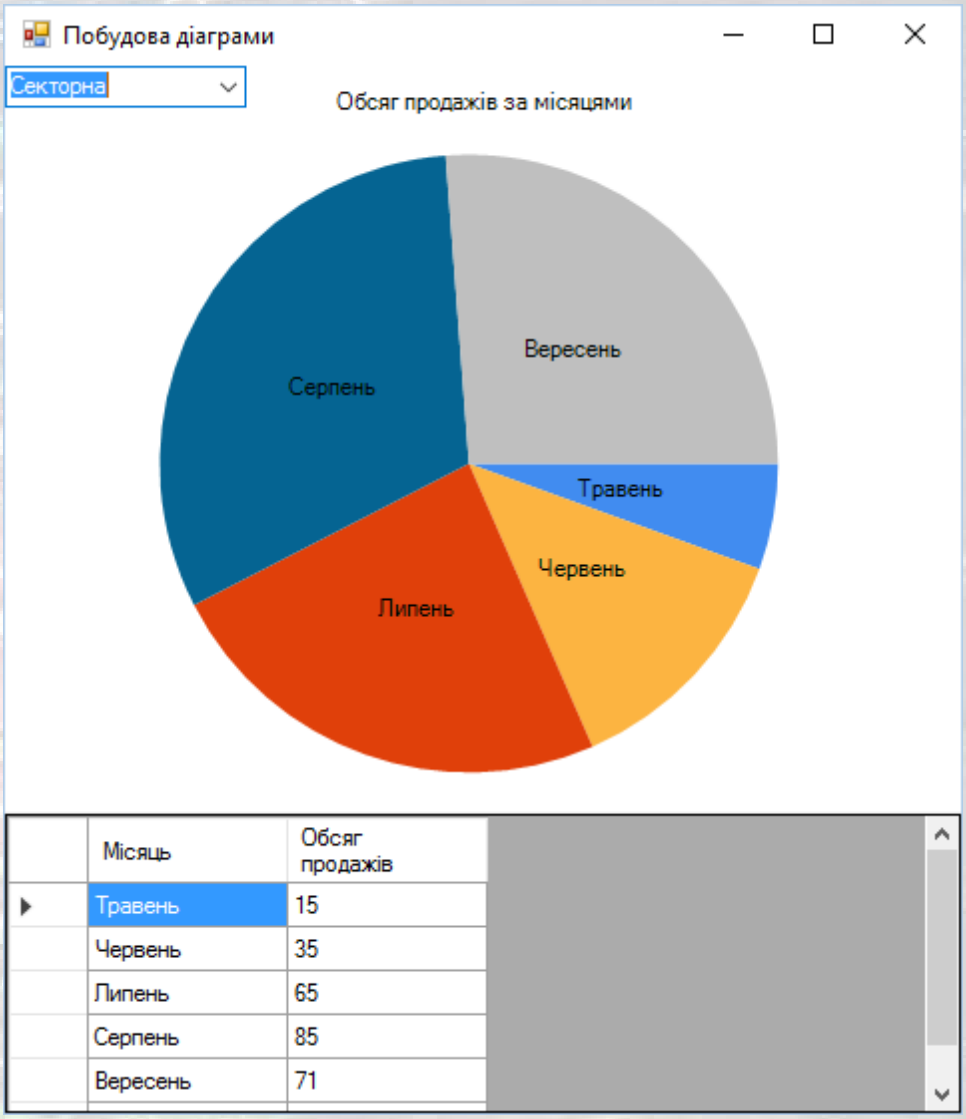


Рисунок 6.4 – Вікно прорами, яке створене у C++/CLI

В об’єкті ComboBox у вікні запущеного доданку можна зі спадаючого списку обрати один з семи передбачених в програмі типів діаграм. Після обрання типу діаграми виникає подія SelectedIndexChanged, в якій здійснюється зміна типу діаграми (рис. 6.5).

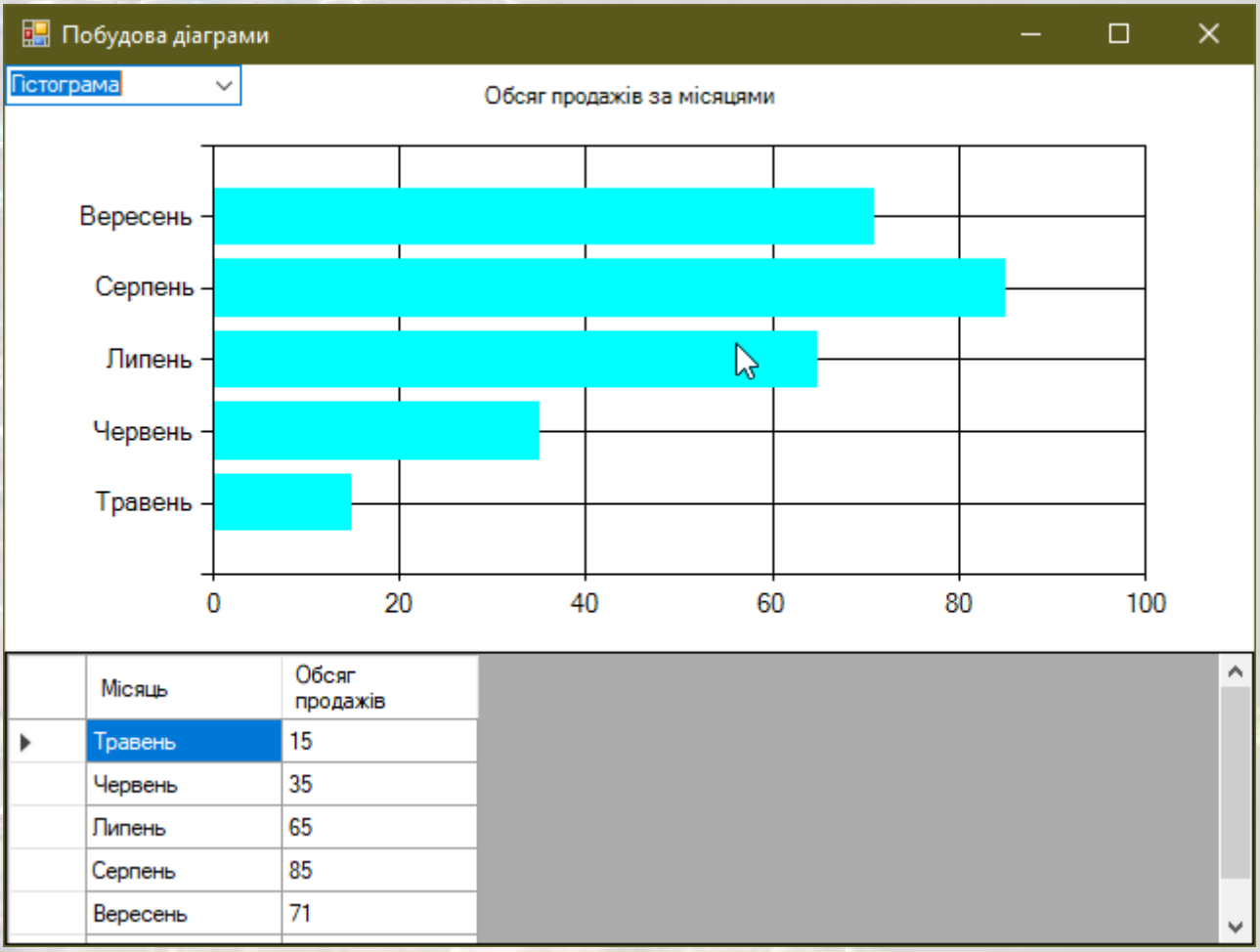


Рисунок 6.5 – Вікно програми, яке створене у C#

Повний текст програмного модуля наведений в лістингу 6.4 (C++/CLI) та лістингах 6.5-6.6 для C#.

Програмний код

Лістинг 6.1

Приклад коду C++/CLI

```
//Програма ілюстрації роботи з табличними даними
//Код модуля Form1.h

#pragma once

namespace ТаблВвод {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для Form1
	/// </summary>
	public ref class Form1 : public System::Windows::Forms::Form
	{
	public:
		Form1(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~Form1()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::DataGridView^  dataGridView1;
	protected:
	private: System::Windows::Forms::Button^  button1;

	private:
		/// <summary>
		/// Требуется переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Обязательный метод для поддержки конструктора - не изменяйте
		/// содержимое данного метода при помощи редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());
			this->button1 = (gcnew System::Windows::Forms::Button());
```

```
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>dataGridView1))->BeginInit();
                    this->SuspendLayout();
                    //
                    // dataGridView1
                    //
                    this->dataGridView1->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
                    this->dataGridView1->Location = System::Drawing::Point(12, 12);
                    this->dataGridView1->Name = L"dataGridView1";
                    this->dataGridView1->Size = System::Drawing::Size(268, 213);
                    this->dataGridView1->TabIndex = 0;
                    //
                    // button1
                    //
                    this->button1->Location = System::Drawing::Point(12, 231);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(268, 23);
                    this->button1->TabIndex = 1;
                    this->button1->Text = L"button1";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&Form1::button1_Click);
                    //
                    // Form1
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(292, 266);
                    this->Controls->Add(this->button1);
                    this->Controls->Add(this->dataGridView1);
                    this->Name = L"Form1";
                    this->Text = L"Form1";
                    this->Load += gcnew System::EventHandler(this, &Form1::Form1_Load);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>dataGridView1))->EndInit();
                    // ..............
                    // Программный код, расположенный выше, создан средой Visual Studio
                    // автоматически, поэтому автором не приводится
                    this->ResumeLayout(false);
        }
#pragma endregion
        // Програма пропонує користувачеві заповнити таблицю телефонів його
        // знайомих, співробітників, родичів, коханих і т. д. Якщо клацнути
        // на кнопці Запис дана таблиця записується на диск у файл формату
        // XML. Для спрощення тексту програми передбачений запис в один і
        // той же файл C: \ tabl.xml. При наступних запусках даної програми
        // таблиця буде зчитуватися з цього файлу, і користувач може
        // продовжувати редагування таблиці
        // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
        DataTable^ Таблиця; // Об'явлення объекта "Таблиця"
        DataSet^ НабірДаних; // Об'явлення объекта "НабірДаних"
        private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
        {
                this->Text = "Простий табличний редактор";
                button1->Text = "Запис";
                Таблиця = gcnew DataTable();
                НабірДаних = gcnew DataSet();
                if (IO::File::Exists("D:\\tabl.xml") == false)
                {
                    // Якщо XML-файлу НЕМАЄ:
                    dataGridView1->DataSource = Таблиця;
                    // Заповнення "шапки" таблиці
                    Таблиця->Columns->Add("Імена");
                    Таблиця->Columns->Add("Номери телефонів");
                    // Додати об'єкт Таблиця в DataSet
                    НабірДаних->Tables->Add(Таблиця);
```

```
            }
            else // Якщо XML-файл Є:
            {
                  НаборДаних->ReadXml("D:\\tabl.xml");
                  // Вміст DataSet у вигляді рядка XML для відладки:
                  String ^ РядокXML = НаборДаних->GetXml();
                  dataGridView1->DataMember = "Назва таблиці";
                  dataGridView1->DataSource = НаборДаних;
            }
      }
      private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
      {
            // Зберегти файл tabl.xml:
            Таблиця->TableName = "Назва таблиці";
            НаборДаних->WriteXml("D:\\tabl.xml");
      }
   };
}
```

Лістинг 6.2

Приклад коду C#

```
//Програма ілюстрації роботи з табличними даними
//Код модуля Form1.Designer.cs

namespace WindowsFormsAppMenu
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.dataGridView1 = new System.Windows.Forms.DataGridView();
            this.button1 = new System.Windows.Forms.Button();
            ((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).BeginInit();
            this.SuspendLayout();
            //
            // dataGridView1
            //
            this.dataGridView1.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            this.dataGridView1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.dataGridView1.Location = new System.Drawing.Point(0, 0);
            this.dataGridView1.Name = "dataGridView1";
            this.dataGridView1.Size = new System.Drawing.Size(800, 450);
            this.dataGridView1.TabIndex = 0;
            //
            // button1
            //
            this.button1.Dock = System.Windows.Forms.DockStyle.Bottom;
            this.button1.Location = new System.Drawing.Point(0, 427);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(800, 23);
            this.button1.TabIndex = 1;
            this.button1.Text = "button1";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // Form1
```

```
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(800, 450);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.dataGridView1);
            this.Name = "Form1";
            this.Text = "Form1";
            this.Load += new System.EventHandler(this.Form1_Load);
            ((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.DataGridView dataGridView1;
        private System.Windows.Forms.Button button1;
    }
}
```

Лістинг 6.3

Приклад коду C#

```
//Програма ілюстрації роботи з табличними даними
//Код модуля Form1.cs

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsAppMenu
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        // Програма пропонує користувачеві заповнити таблицю телефонів його
        // знайомих, співробітників, родичів, коханих і т. д. Якщо клацнути
        // на кнопці Запис дана таблиця записується на диск у файл формату
        // XML. Для спрощення тексту програми передбачений запис в один і
        // той же файл C: \ tabl.xml. При наступних запусках даної програми
        // таблиця буде зчитуватися з цього файлу, і користувач може
        // продовжувати редагування таблиці
        // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
        DataTable Таблиця; // Об'явлення объекта "Таблиця"
        DataSet НабірДаних; // Об'явлення объекта "НабірДаних"

        private void Form1_Load(object sender, EventArgs e)
        {
            this.Text = "Простий табличний редактор";
            button1.Text = "Запис";
            Таблиця = new DataTable();
            НабірДаних = new DataSet();
            if (System.IO.File.Exists("D:\\tabl.xml") == false)
            {
                // Якщо XML-файлу НЕМАЄ:
                dataGridView1.DataSource = Таблиця;
                // Заповнення "шапки" таблиці
                Таблиця.Columns.Add("Імена");
                Таблиця.Columns.Add("Номери телефонів");
                // Додати об'єкт Таблиця в DataSet
                НабірДаних.Tables.Add(Таблиця);
            }
            else // Якщо XML-файл Є:
            {
                НабірДаних.ReadXml("D:\\tabl.xml");
                // Вміст DataSet у вигляді рядка XML для відладки:
                String РядокXML = НабірДаних.GetXml();
                dataGridView1.DataMember = "Назва таблиці";
                dataGridView1.DataSource = НабірДаних;
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
```

```
                // Зберегти файл tabl.xml:
                Таблиця.TableName = "Назва таблиці";
                НабірДаних.WriteXml("D:\\tabl.xml");
            }
        }
    }
```

Лістинг 6.4

Приклад коду C++/CLI

```
// Програма ілюстрації створення діаграм за табличними даними

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::DataVisualization::Charting::Chart^  chart1;
	protected:
	private: System::Windows::Forms::DataGridView^  dataGridView1;
	private: System::Windows::Forms::ComboBox^  comboBox1;

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			System::Windows::Forms::DataVisualization::Charting::ChartArea^
chartArea1 = (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());
			System::Windows::Forms::DataVisualization::Charting::Legend^  legend1 =
(gcnew System::Windows::Forms::DataVisualization::Charting::Legend());
```

```
                    System::Windows::Forms::DataVisualization::Charting::Series^  series1 =
(gcnew System::Windows::Forms::DataVisualization::Charting::Series());
                    this->chart1 = (gcnew
System::Windows::Forms::DataVisualization::Charting::Chart());
                    this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());
                    this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>chart1))->BeginInit();
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->BeginInit();
                    this->SuspendLayout();
                    //
                    // chart1
                    //
                    chartArea1->Name = L"ChartArea1";
                    this->chart1->ChartAreas->Add(chartArea1);
                    legend1->Name = L"Legend1";
                    this->chart1->Legends->Add(legend1);
                    this->chart1->Location = System::Drawing::Point(0, 0);
                    this->chart1->Name = L"chart1";
                    series1->ChartArea = L"ChartArea1";
                    series1->Legend = L"Legend1";
                    series1->Name = L"Series1";
                    this->chart1->Series->Add(series1);
                    this->chart1->Size = System::Drawing::Size(477, 300);
                    this->chart1->TabIndex = 0;
                    this->chart1->Text = L"chart1";
                    //
                    // dataGridView1
                    //
                    this->dataGridView1->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
                    this->dataGridView1->Location = System::Drawing::Point(0, 297);
                    this->dataGridView1->Name = L"dataGridView1";
                    this->dataGridView1->Size = System::Drawing::Size(396, 150);
                    this->dataGridView1->TabIndex = 1;
                    //
                    // comboBox1
                    //
                    this->comboBox1->FormattingEnabled = true;
                    this->comboBox1->Location = System::Drawing::Point(391, 41);
                    this->comboBox1->Name = L"comboBox1";
                    this->comboBox1->Size = System::Drawing::Size(121, 21);
                    this->comboBox1->TabIndex = 2;
                    this->comboBox1->SelectedIndexChanged += gcnew
System::EventHandler(this, &MyForm::comboBox1_SelectedIndexChanged);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(512, 447);
                    this->Controls->Add(this->comboBox1);
                    this->Controls->Add(this->dataGridView1);
                    this->Controls->Add(this->chart1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->Resize += gcnew System::EventHandler(this,
&MyForm::MyForm_Resize);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>chart1))->EndInit();
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->EndInit();
                    this->ResumeLayout(false);

                }
```

```
#pragma endregion
            // Програма використовує елементи керування Chart і DataGridView, виводить
            // графік (діаграму) залежнсті обсягів продажів від часу за місяцями.
            // При цьому в якості джерела даних вказуємо об'єкт класу DataTable
            // Стиль діаграми можна міняти за допомогою переліку з comboBox1
        private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
            this->Text = "Побудова діаграми";
            DataTable ^ Таблиця = gcnew DataTable();
            // У цій таблиці замовляємо дві колонки "Місяць" і "Обсяг продажів":
            Таблиця->Columns->Add("Місяць", String::typeid);
            // У C #: Табліца.Columns.Add("Місяць", typeof(String));
            // Значення в другій колонці призначаємо типу long:
            Таблиця->Columns->Add("Обсяг продажів", long::typeid);
            // У C #: Табліца.Columns.Add("Обсяг продажів", typeof(long));
            // Заповнити перший рядок (ряд) в таблиці:
            DataRow ^ Ряд = Таблиця->NewRow();
            Ряд["Місяць"] = "Травень"; Ряд["Обсяг продажів"] = 15;
            Таблиця->Rows->Add(Ряд);
            // Додаємо другий рядок в таблиці:
            Ряд = Таблиця->NewRow();
            Ряд["Місяць"] = "Червень"; Ряд["Обсяг продажів"] = 35;
            Таблиця->Rows->Add(Ряд);
            // Додаємо третій рядок:
            Ряд = Таблиця->NewRow();
            Ряд["Місяць"] = "Липень"; Ряд["Обсяг продажів"] = 65;
            Таблиця->Rows->Add(Ряд);
            // Додаємо четвертий рядок:
            Ряд = Таблиця->NewRow();
            Ряд["Місяць"] = "Серпень"; Ряд["Обсяг продажів"] = 85;
            Таблиця->Rows->Add(Ряд);
            // Додаємо п'ятий рядок:
            Ряд = Таблиця->NewRow();
            Ряд["Місяць"] = "Вересень"; Ряд["Обсяг продажів"] = 71;
            Таблиця->Rows->Add(Ряд);

            // Наведені нижче властивості можна задати у вікні Властивості для об'єкту
    chart1
            // Складену таблицю вказуємо як джерело даних:
            chart1->DataSource = Таблиця;
            // На одному графіку можна зобразити кілька залежностей.
            // Наприклад, перша залежність - обсяги продажів за вказаними
            // Місяців в 2014 році, і друга залежність - продажу по
            // тим же місяцям в 2015 році.
            // В даному графіку ми покажемо тільки одну залежність, дані
            // для відображення цієї залежності назвемо "Series1"
            // На горизонтальній осі відкладаємо назви місяців:
            chart1->Series["Series1"]->XValueMember = "Місяць";
            // А по вертикальній осі відкладаємо обсяги продажів:
            chart1->Series["Series1"]->YValueMembers = "Обсяг продажів";
            // Назва графіка (діаграми):
            chart1->Titles->Add("Обсяг продажів за місяцями");
            // Задаємо тип діаграми - стовпчикова гістограма:
            chart1->Series["Series1"]->ChartType = System::Windows::Forms::
                  DataVisualization::Charting::SeriesChartType::Column;
            // Тип діаграми може бути іншим, наприклад: Pie, Line і ін.
            chart1->Series["Series1"]->Color = Color::Aqua;
            // Легенду на графіку не відображаємо:
            chart1->Series["Series1"]->IsVisibleInLegend = false;
            // Прив'язка графіка до джерела даних:
            chart1->DataBind();
            // Для сітки даних вказати джерело даних
            dataGridView1->DataSource = Таблиця;

            // Задаємо перелік типів діаграм у об'єкті comboBox1
            // Це можна задати у вікні властивостей comboBox1 в полі Items
            comboBox1->Items->Add("Точкова");              //Point
            comboBox1->Items->Add("Графік");               //Line
```

```
            comboBox1->Items->Add("Згладжений графік"); //Spline
            comboBox1->Items->Add("Секторна");                    //Pie
            comboBox1->Items->Add("Пелюсткова");            //Radar
            comboBox1->Items->Add("Гістограма");            //Bar
            comboBox1->Items->Add("Стовпчаста");            //Column
            comboBox1->Text = "Оберіть тип діаграми";

            // Встановлюємо вирівнювання об'єктів в межах форми
            chart1->Dock = System::Windows::Forms::DockStyle::Top;
            dataGridView1->Dock = System::Windows::Forms::DockStyle::Bottom;
            comboBox1->Left = 0;
            comboBox1->Top = 0;
        }

    private: System::Void comboBox1_SelectedIndexChanged(System::Object^  sender,
    System::EventArgs^  e) {
        // Змінюємо тип діаграми в залежності від обраного рядка в об'єкті  comboBox1
        switch (comboBox1->SelectedIndex)
            {
            case 0:
                chart1->Series["Series1"]->ChartType =
    System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Point;
                break;
            case 1:
                chart1->Series["Series1"]->ChartType =
    System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Line;
                break;
            case 2:
                chart1->Series["Series1"]->ChartType =
    System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Spline;
                break;
            case 3:
                chart1->Series["Series1"]->ChartType =
    System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Pie;
                break;
            case 4:
                chart1->Series["Series1"]->ChartType =
    System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Radar;
                break;
            case 5:
                chart1->Series["Series1"]->ChartType =
    System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Bar;
                break;
            default:
                chart1->Series["Series1"]->ChartType =
    System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Column;
                break;
            }
        }
    private: System::Void MyForm_Resize(System::Object^  sender, System::EventArgs^  e) {
        // Підбір висоти об'єкту chart1 при зміні розміру вікна
        chart1->Height = this->Height - dataGridView1->Height - 40;
    }
    };
    }
```

Лістинг 6.5

Приклад коду C#

```
// Програма ілюстрації створення діаграм за табличними даними
// Код модуля Form1.Designer.cs

namespace WindowsFormsAppChart
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea5 = new
System.Windows.Forms.DataVisualization.Charting.ChartArea();
            System.Windows.Forms.DataVisualization.Charting.Legend legend5 = new
System.Windows.Forms.DataVisualization.Charting.Legend();
            System.Windows.Forms.DataVisualization.Charting.Series series5 = new
System.Windows.Forms.DataVisualization.Charting.Series();
            this.chart1 = new System.Windows.Forms.DataVisualization.Charting.Chart();
            this.dataGridView1 = new System.Windows.Forms.DataGridView();
            this.comboBox1 = new System.Windows.Forms.ComboBox();
            ((System.ComponentModel.ISupportInitialize)(this.chart1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).BeginInit();
            this.SuspendLayout();
            //
            // chart1
            //
            chartArea5.Name = "ChartArea1";
            this.chart1.ChartAreas.Add(chartArea5);
            legend5.Name = "Legend1";
            this.chart1.Legends.Add(legend5);
            this.chart1.Location = new System.Drawing.Point(515, 225);
            this.chart1.Name = "chart1";
            series5.ChartArea = "ChartArea1";
            series5.Legend = "Legend1";
            series5.Name = "Series1";
            this.chart1.Series.Add(series5);
            this.chart1.Size = new System.Drawing.Size(300, 300);
            this.chart1.TabIndex = 0;
```

```
            this.chart1.Text = "chart1";
            //
            // dataGridView1
            //
            this.dataGridView1.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            this.dataGridView1.Location = new System.Drawing.Point(338, 110);
            this.dataGridView1.Name = "dataGridView1";
            this.dataGridView1.Size = new System.Drawing.Size(240, 150);
            this.dataGridView1.TabIndex = 1;
            //
            // comboBox1
            //
            this.comboBox1.FormattingEnabled = true;
            this.comboBox1.Location = new System.Drawing.Point(466, 47);
            this.comboBox1.Name = "comboBox1";
            this.comboBox1.Size = new System.Drawing.Size(121, 21);
            this.comboBox1.TabIndex = 2;
            this.comboBox1.SelectedIndexChanged += new
System.EventHandler(this.comboBox1_SelectedIndexChanged);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(800, 450);
            this.Controls.Add(this.comboBox1);
            this.Controls.Add(this.dataGridView1);
            this.Controls.Add(this.chart1);
            this.Name = "Form1";
            this.Text = "Form1";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.Resize += new System.EventHandler(this.Form1_Resize);
            ((System.ComponentModel.ISupportInitialize)(this.chart1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.DataVisualization.Charting.Chart chart1;
        private System.Windows.Forms.DataGridView dataGridView1;
        private System.Windows.Forms.ComboBox comboBox1;
    }
}
```

Лістинг 6.6

Приклад коду C#

```
// Програма ілюстрації створення діаграм за табличними даними
// Код модуля Form1.cs

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsAppChart
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.Text = "Побудова діаграми";
            DataTable Таблиця = new DataTable();
            // У цій таблиці замовляємо дві колонки "Місяць" і "Обсяг продажів":
            Таблиця.Columns.Add("Місяць", typeof(String));
            // У C #: Табліца.Columns.Add("Місяць", typeof(String));
            // Значення в другій колонці призначаємо типу long:
            Таблиця.Columns.Add("Обсяг продажів", typeof(long));
            // У C #: Табліца.Columns.Add("Обсяг продажів", typeof(long));
            // Заповнити перший рядок (ряд) в таблиці:
            DataRow Ряд = Таблиця.NewRow();
            Ряд["Місяць"] = "Травень"; Ряд["Обсяг продажів"] = 15;
            Таблиця.Rows.Add(Ряд);
            // Додаємо другий рядок в таблиці:
            Ряд = Таблиця.NewRow();
            Ряд["Місяць"] = "Червень"; Ряд["Обсяг продажів"] = 35;
            Таблиця.Rows.Add(Ряд);
            // Додаємо третій рядок:
            Ряд = Таблиця.NewRow();
            Ряд["Місяць"] = "Липень"; Ряд["Обсяг продажів"] = 65;
            Таблиця.Rows.Add(Ряд);
            // Додаємо четвертий рядок:
            Ряд = Таблиця.NewRow();
            Ряд["Місяць"] = "Серпень"; Ряд["Обсяг продажів"] = 85;
            Таблиця.Rows.Add(Ряд);
            // Додаємо п'ятий рядок:
            Ряд = Таблиця.NewRow();
            Ряд["Місяць"] = "Вересень"; Ряд["Обсяг продажів"] = 71;
            Таблиця.Rows.Add(Ряд);

            // Наведені нижче властивості можна задати у вікні Властивості
            // для об'єкту chart1
            // Складену таблицю вказуємо як джерело даних:
            chart1.DataSource = Таблиця;
            // На одному графіку можна зобразити кілька залежностей.
            // Наприклад, перша залежність - обсяги продажів за вказаними
            // Місяців в 2014 році, і друга залежність - продажу по
            // тим же місяцям в 2015 році.
```

```
            // В даному графіку ми покажемо тільки одну залежність, дані
            // для відображення цієї залежності назвемо "Series1"
            // На горизонтальній осі відкладаємо назви місяців:
            chart1.Series["Series1"].XValueMember = "Місяць";
            // А по вертикальній осі відкладаємо обсяги продажів:
            chart1.Series["Series1"].YValueMembers = "Обсяг продажів";
            // Назва графіка (діаграми):
            chart1.Titles.Add("Обсяг продажів за місяцями");
            // Задаємо тип діаграми - стовпчикова гістограма:
            chart1.Series["Series1"].ChartType = System.Windows.Forms.
                    DataVisualization.Charting.SeriesChartType.Column;
            // Тип діаграми може бути іншим, наприклад: Pie, Line і ін.
            chart1.Series["Series1"].Color = Color.Aqua;
            // Легенду на графіку не відображаємо:
            chart1.Series["Series1"].IsVisibleInLegend = false;
            // Прив'язка графіка до джерела даних:
            chart1.DataBind();
            // Для сітки даних вказати джерело даних
            dataGridView1.DataSource = Таблиця;

            // Задаємо перелік типів діаграм у об'єкті comboBox1
            // Це можна задати у вікні властивостей comboBox1 в полі Items
            comboBox1.Items.Add("Точкова");           //Point
            comboBox1.Items.Add("Графік");            //Line
            comboBox1.Items.Add("Згладжений графік"); //Spline
            comboBox1.Items.Add("Секторна");          //Pie
            comboBox1.Items.Add("Пелюсткова");        //Radar
            comboBox1.Items.Add("Гістограма");        //Bar
            comboBox1.Items.Add("Стовпчаста");        //Column
            comboBox1.Text = "Оберіть тип діаграми";

            // Встановлюємо вирівнювання об'єктів в межах форми
            chart1.Dock = System.Windows.Forms.DockStyle.Top;
            dataGridView1.Dock = System.Windows.Forms.DockStyle.Bottom;
            chart1.Height = this.Height - dataGridView1.Height - 40;
            comboBox1.Left = 0;
            comboBox1.Top = 0;

        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            // Змінюємо тип діаграми в залежності від обраного рядка в об'єкті   comboBox1
            switch (comboBox1.SelectedIndex)
            {
                case 0:
                    chart1.Series["Series1"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;
                    break;
                case 1:
                    chart1.Series["Series1"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
                    break;
                case 2:
                    chart1.Series["Series1"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
                    break;
                case 3:
                    chart1.Series["Series1"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Pie;
                    break;
                case 4:
                    chart1.Series["Series1"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Radar;
                    break;
                case 5:
```

```
                    chart1.Series["Series1"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Bar;
                    break;
                default:
                    chart1.Series["Series1"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;
                    break;
            }

        }

        private void Form1_Resize(object sender, EventArgs e)
        {
            // Підбір висоти об'єкту chart1 при зміні розміру вікна
            chart1.Height = this.Height - dataGridView1.Height - 40;
        }
    }
}
```

ПРАКТИЧНА РОБОТА №7

7 Редагування графічних даних у C++/CLI та C#

7.1 Найпростіше виведення графічного зображення у форму

Завдання 1.

Необхідно створити проект Windows Forms для виведення графічного зображення з вказаного файлу за допомогою об'єкту класу Image та методу малювання зображення у формі DrawImage графічного об'єкту Graphics з аргументу е процедури OnPaint.

Створюємо процедуру обробки події OnPaint звичайним способом, тобто на вкладці конструктора форми в панелі властивостей вікна Властивості (Properties) клацаємо на значку блискавки і в списку, що з'явився, всіх подій для об'єкта MyForm виберемо подію Paint. Об'єкт Graphics отримуємо з аргументу е події Paint.

Приклад коду C++/CLI

```
// Найпростіше виведення зображення в форму
private: System::Void MyForm_Paint(System::Object^  sender,
System::Windows::Forms::PaintEventArgs^  e)
{
      // У властивостях форми клацаємо значок блискавки і в списку, що
      // з'явився всіх подій для об'єкта MyForm виберемо подію Paint.
      // Подія Paint - це подія малювання форми:
      this->Text = "Малюнок"; // Заголовок вікна
      // Створюємо об'єкт для роботи із зображенням:
      Image ^ Малюнок = gcnew Bitmap("D:\\PICTUREs\\ЕмблемаКХТП.png");
      // або Image ^ Малюнок =
      // Image::FromFile("D:\\PICTUREs\\ЕмблемаКХТП.png");
      // Виведення зображення в форму:
      e->Graphics->DrawImage(Малюнок, 5, 5);
}
```

Приклад коду C#

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
      // У властивостях форми клацаємо значок блискавки і в списку, що
      // з'явився всіх подій для об'єкта MyForm виберемо подію Paint.
      // Подія Paint - це подія малювання форми:
      this.Text = "Малюнок"; // Заголовок вікна
      // Створюємо об'єкт для роботи із зображенням:
      Image Малюнок = new Bitmap("D:\\PICTUREs\\ЕмблемаКХТП.png");
      // або Image Малюнок =
      // Image.FromFile("D:\\PICTUREs\\ЕмблемаКХТП.png");
      // Виведення зображення в форму:
      e.Graphics.DrawImage(Малюнок, 5, 5);
```

```
}
```

Результат роботи програми зображений на рис. 7.1.

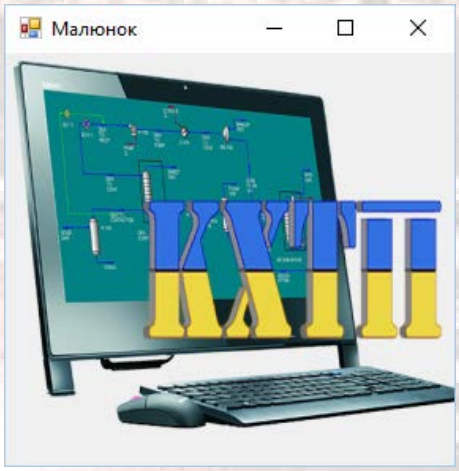


Рисунок 7.1 – Вікно програми з графічним зображенням без масштабування

Завдання 2.

Необхідно створити проект Windows Forms для виведення графічного зображення з файлу по натисканню кнопки, шляхом створення об'єктів класу Image та Graphics.

Розглянемо ще один спосіб виведення графіки в форму. В цьому випадку при натисканні на командній кнопці відбувається безпосереднє створення об'єкта класу `Graphics`. Програмний код представлений в лістингу нижче.

Приклад коду C++/CLI

```
// Найпростіше виведення зображення в форму
private: System::Void Form1_Load(System::Object ^ sender,
System::EventArgs ^ e)
{
      // Подія завантаження форми:
      MyForm::Text = "Малюнок";
      button1->Text = "Показати малюнок";
}

private: System::Void button1_Click(System::Object ^ sender,
System::EventArgs ^ e)
{
      // Подія "клацання на кнопці"
      Image ^ Малюнок = gcnew
            Bitmap("D:\\DOCs\\PICTUREs\\ЕмблемаХТФ.png");
      // Створення графічного об'єкта:
      Graphics ^ Графіка = this->CreateGraphics();
      // Або Graphics ^ Графіка = CreateGraphics ();
      Графіка->DrawImage(Малюнок, 5, 5);
}
```

Приклад коду C#

```
// Найпростіше виведення зображення в форму
private void Form1_Load(object sender, EventArgs e)
{
// Подія завантаження форми:
      Text = "Малюнок";
      button1.Text = "Показати малюнок";
}

private void button1_Click(object sender, EventArgs e)
{
      // Подія "клацання на кнопці"
      Image Малюнок = new Bitmap("D:\\VIO77\\Зображення\\C#_1.jpg");
      // Створення графічного об'єкта:
      Graphics Графіка = this.CreateGraphics();
      // Або Graphics Графіка = CreateGraphics();
      Графіка.DrawImage(Малюнок, 5, 5);
}
```

Результат роботи програми після натискання кнопки у вікні зображений на рис. 7.2. Повний текст програми наведений в лістингу 7.1 (C++/CLI) та в лістингу 7.2 (C#).

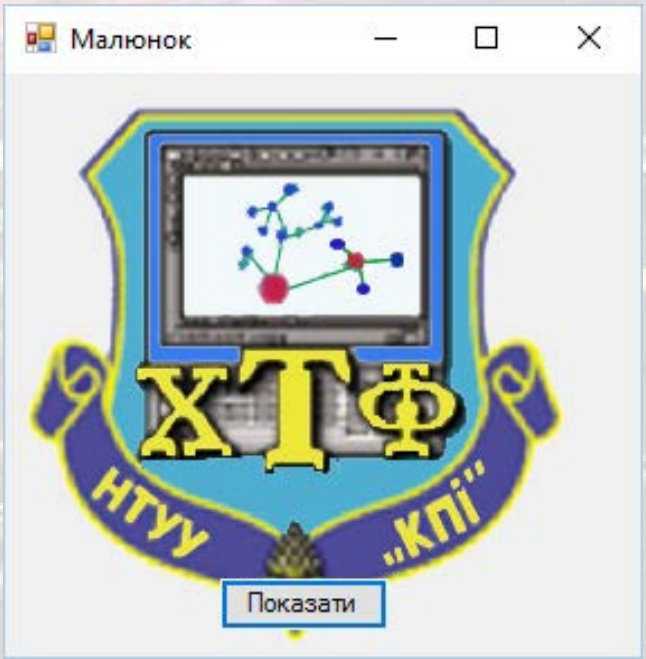


Рисунок 7.2 – Вікно програми з графічним зображенням без масштабування

З розглянутими в даному розділі методами можна працювати не тільки для виведення зображень графічних файлів в форму, але і вирішувати багато інших завдань, пов'язаних з графікою.

7.2 Використання елементу PictureBox для відображення растрового файлу з можливістю прокрутки

Завдання 3.

Необхідно створити проект Windows Forms, який дозволяє відкривати довільний графічний файл з можливістю прокрутки зображення та зберігати його під іншим ім'ям за допомогою об'єктів класу OpenFileDialog та SaveFileDialog.

Запустимо Visual Studio і у вікні Створення проекту (New Project) виберемо потрібний шаблон проекту Windows Form. З Панелі елементів (Toolbox) перетягнемо на форму елемент управління Panel, а на нього помістимо елемент PictureBox. Розмістимо ще один елемент Panel у формі та додамо всередину об'єкту два елементи Button (рис. 7.3).

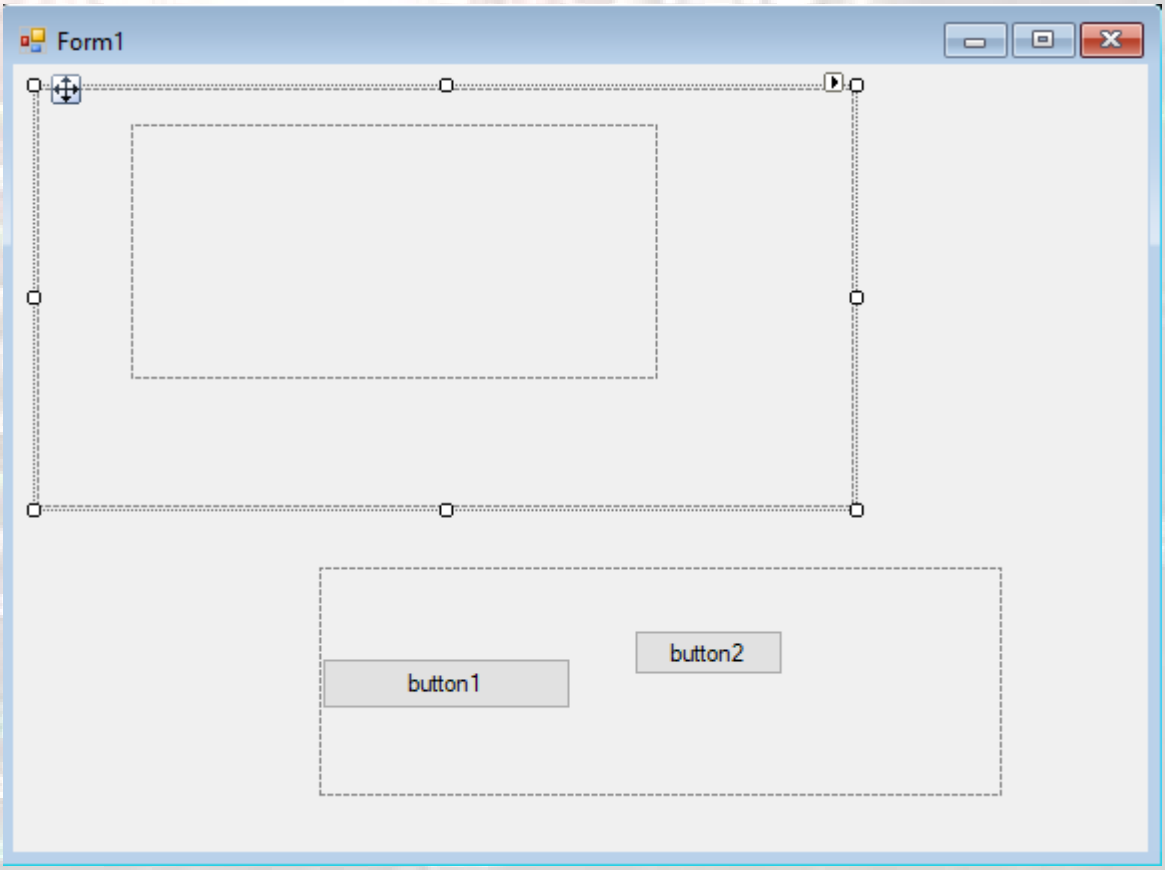


Рисунок 7.3 – Вікно редактора форми з доданими елементами

Далі перейдемо на вкладку програмного коду за допомогою клавіши F7 і введемо текст, представлений у лістингу.

Приклад коду C++/CLI

```
private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {
     OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
     if (openFileDialog1->ShowDialog() ==
          System::Windows::Forms::DialogResult::OK)
```

```
        {
            pictureBox1->Image = Image
                ::FromFile(openFileDialog1->FileName);
            // Примусово викликаємо обробник події Resize форми
            //для встановлення розмірів об'єктів panel1 та button1
            MyForm_Resize(nullptr, nullptr);
        }
    }

    private: System::Void MyForm_Load(System::Object^  sender,
    System::EventArgs^  e) {
        this->Text = "Графічний файл";
        button1->Text = "Відкрити";
        button2->Text = "Зберегти";
        panel2->Height = 40;
        panel2->Dock = System::Windows::Forms::DockStyle::Bottom;
        button1->Dock = System::Windows::Forms::DockStyle::Right;
        panel1->Dock = System::Windows::Forms::DockStyle::Top;
        pictureBox1->SizeMode = PictureBoxSizeMode::AutoSize;
        panel1->AutoScroll = true; // Дозволяємо прокрутку зображення
        button2->Dock = System::Windows::Forms::DockStyle::Fill;
    }

    private: System::Void MyForm_Resize(System::Object^  sender,
    System::EventArgs^  e) {
        // Встановлення розмірів об'єктів panel1 та button1
        panel1->Height = this->Height - panel2->Height - 40;
        button1->Width = this->panel2->Width / 2;
    }

    private: System::Void button2_Click(System::Object^  sender,
    System::EventArgs^  e) {
        // Відобразити вікно SaveFileDialog щоб користувач
        // міг зберегти зображення
        SaveFileDialog ^ saveFileDialog1 = gcnew SaveFileDialog();
        saveFileDialog1->Filter = "Зображення JPeg|*.jpg
            |Зображення Bitmap|*.bmp|Зображення Gif|*.gif";
        saveFileDialog1->Title = "Зберегти файл зображення";
        saveFileDialog1->ShowDialog();
        // Якщо ім'я файлу не пустий рядок, зберегти зображення
        if (saveFileDialog1->FileName != "")
        {
            // Зберегти зображення через FileStream,
            //створений методом OpenFile
            System::IO::FileStream ^ fs = safe_cast<System::IO
                ::FileStream^>(saveFileDialog1->OpenFile());
            // Можна зберегти зображення у відповідному форматі
            // ImageFormat на основі типу файлу, обраного в діалоговому
            // вікні. Тип файлу береться з властивості FilterIndex
            switch (saveFileDialog1->FilterIndex)
            {
                case 1:
                    this->pictureBox1->Image->Save(fs,
```

```
                        System::Drawing::Imaging::ImageFormat::Jpeg);
                    break;
                case 2:
                    this->pictureBox1->Image->Save(fs,
                        System::Drawing::Imaging::ImageFormat::Bmp);
                    break;
                case 3:
                    this->pictureBox1->Image->Save(fs,
                        System::Drawing::Imaging::ImageFormat::Gif);
                    break;
            }
            fs->Close();
        }
}
```

В наступному лістингу приведений варіант програмного коду обробників подій у C#.

Приклад коду C#

```
//Програма ілюстрації відкриття графічного файлу у формі

private void button1_Click(object sender, EventArgs e)
{
    Text = "Test";
    OpenFileDialog openFileDialog1 = new OpenFileDialog();
    if (openFileDialog1.ShowDialog() ==
        System.Windows.Forms.DialogResult.OK)
    {
        pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);
        // Примусово викликаємо обробник події Resize форми
        //для встановлення розмірів об'єктів panel1 та button1
        Form1_Resize(null, null);
    }
}

private void Form1_Load(object sender, EventArgs e)
{
    this.Text = "Графічний файл";
    button1.Text = "Відкрити";
    button2.Text = "Зберегти";
    panel2.Height = 40;
    panel2.Dock = System.Windows.Forms.DockStyle.Bottom;
    button1.Dock = System.Windows.Forms.DockStyle.Right;
    panel1.Dock = System.Windows.Forms.DockStyle.Top;
    pictureBox1.SizeMode = PictureBoxSizeMode.AutoSize;
    panel1.AutoScroll = true; // Дозволяємо прокрутку зображення
    button2.Dock = System.Windows.Forms.DockStyle.Fill;
    Form1_Resize(null, null);
}

private void Form1_Resize(object sender, EventArgs e)
```

```
{
      // Встановлення розмірів об'єктів panel1 та button1
      panel1.Height = this.Height - panel2.Height - 40;
      button1.Width = this.panel2.Width / 2;
}

private void button2_Click(object sender, EventArgs e)
{
      // Відобразити вікно SaveFileDialog щоб користувач
      // міг зберегти зображення
      SaveFileDialog saveFileDialog1 = new SaveFileDialog();
      saveFileDialog1.Filter =     "Зображення JPeg|*.jpg|
            Зображення Bitmap|*.bmp|Зображення Gif|*.gif";
      saveFileDialog1.Title = "Зберегти файл зображення";
      saveFileDialog1.ShowDialog();
      // Якщо ім'я файлу не пустий рядок, зберегти зображення
      if (saveFileDialog1.FileName != "")
      {
      // Зберегти зображення через FileStream, створений методом OpenFile
            System.IO.FileStream fs = new System.IO.
                  FileStream(saveFileDialog1.FileName,
                  System.IO.FileMode.Create);
            // Можна зберегти зображення у відповідному форматі
            // ImageFormat на основі типу файлу, обраного в діалоговому
            // вікні. Тип файлу береться з властивості FilterIndex
            switch (saveFileDialog1.FilterIndex)
            {
                  case 1:
                        this.pictureBox1.Image.Save(fs,
                        System.Drawing.Imaging.ImageFormat.Jpeg);
                        break;
                  case 2:
                        this.pictureBox1.Image.Save(fs,
                        System.Drawing.Imaging.ImageFormat.Bmp);
                        break;
                  case 3:
                        this.pictureBox1.Image.Save(fs,
                        System.Drawing.Imaging.ImageFormat.Gif);
                        break;
            }
            fs.Close();
      }
}
```

Окрім відображення графічного файлу з можливістю прокрутки, дана програма дозволяє вибирати довільний файл для відкриття за допомогою елементу діалогового вікна OpenFileDialog. Відкритий файл можна зберегти під іншим ім’ям і в іншому графічному форматі за допомогою елементу діалогового вікна SaveFileDialog.

Об’єкти класів OpenFileDialog та SaveFileDialog створюються програмно, хоча обидва цих елементи можна було перетягти з Панелі

елементів у форму вручну і тоді б наступні рядки коду не потрібно було б писати:

Приклад коду C++/CLI

```
OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
SaveFileDialog ^ saveFileDialog1 = gcnew SaveFileDialog();
```

Приклад коду C#

```
OpenFileDialog openFileDialog1 = new OpenFileDialog();
SaveFileDialog saveFileDialog1 = new SaveFileDialog();
```

Також програма виконує масштабування елементів свого інтерфейсу при зміні розміру вікна (рис. 7.4).

Всі властивості обробника події `Form1_Load` можна було задати в режимі проектування у вікні `Властивості` для кожного з наведених там об'єктів і тоді даний обробник події програмувати було б непотрібно.

Обробник події `Form1_Resize`, розом з встановленими властивостями розмірів та розташування елементів інтерфейсу у вікні, відповідає за правильне масштабування інтерфейсу програми при зміні розміру вікна.

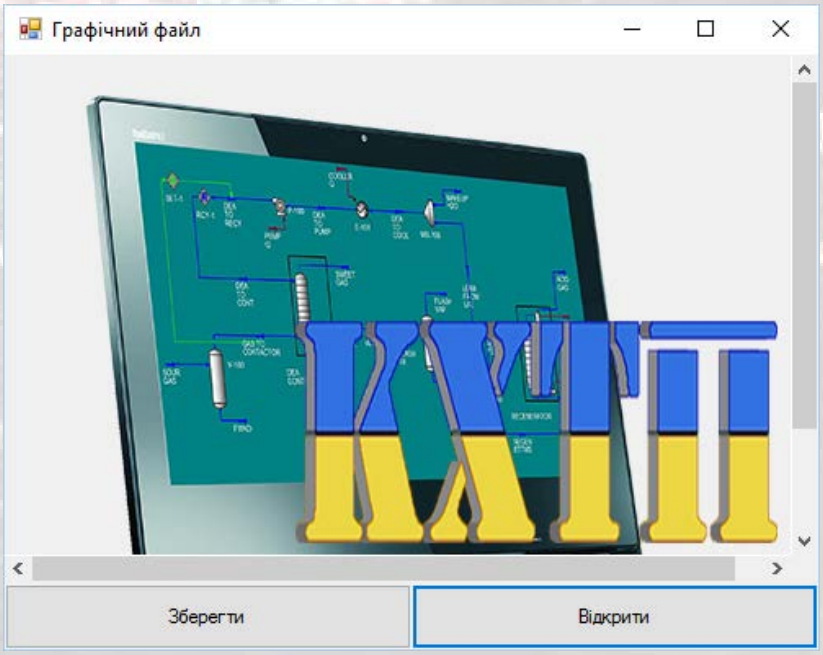


Рисунок 7.4 – Вікно програми з графічним зображенням

Обробник події `button1_Click` виникає при натисканні на кнопку `Відкрити` у вікні програми та відповідає за відображення діалогового вікна відкриття файлу і розміщення обраного зображення в об'єкті `pictureBox1`.

Обробник події `button2_Click` виникає при натисканні на кнопку `Зберегти` у вікні програми та дозволяє за допомогою діалогового вікна збереження файлу зберегти відкритий файл під іншим ім'ям в одному з трьох графічних форматів, які задаються у властивості `Filter` об'єкту

saveFileDialog1. Повний текст програми наведений в лістингу 7.3 (C++/CLI) та лістингу 7.4 (C#).

7.3 Побудова графіку методами класу Graphics

Завдання 4.

Необхідно створити проект Windows Forms, в якому реалізувати побудову графіку в ході виконання програми за допомогою об'єктів класів Bitmap, Graphics і Pen та методів DrawString, DrawLine та DrawEllipse класу Graphics.

Для вирішення цього завдання запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо додаток шаблону Windows Forms. Перенесемо з Панелі елементів (Toolbox) в проектовану форму два об'єкти контейнери Panel. В один контейнер Panel додайте об'єкт кнопки Button та встановіть її властивість Dock у значення Fill для того, щоб кнопка займала всю вільну область об'єкта Panel. Властивість Dock об'єкта Panel встановіть у значення Bottom у вікні Властивості (Properties) для того щоб даний об'єкт розтягувався на всю ширину вікна вздовж нижньої його частини.

В другий контейнер Panel додайте елемент управління PictureBox (графічними поле). Властивість Dock об'єктів PictureBox та Panel встановіть у значення Fill у вікні Властивості (Properties) для того щоб дані об'єкти займали всю вільну область вікна окрім тієї, що займає перший об'єкт Panel.

Далі перейдемо на вкладку програмного коду і введемо текст, представлений нижче.

Приклад коду C++/CLI

```
// Програма малює графік продажів по місяцях. Зрозуміло, що таким же
// чином можна побудувати будь-який графік по точках для інших прикладних
// цілей
// ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
// Вихідні дані для побудови графіка (тобто вихідні точки):
array <String ^> ^ Months;
array <int> ^ Sales;
Graphics ^ Графіка;
// Далі, створюємо об'єкт Bitmap, який має
// Той же розмір і роздільну здатність, що і PictureBox
Bitmap ^ Растр;
int ВідступЛіворуч, ВідступПраворуч, ВідступЗнизу, ВідступЗверху;
int ДовжинаВертОсі, ДовжинаГоризОсі, YГоризОсі, XВертОсі, Xmax,
XПочЕпюри;
// Крок градуювання по горизонтальній і вертикальній осях:
double ГоризКрок;
int ВертКрок;
// ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private: System::Void button1_Click(System::Object ^ sender,
System::EventArgs ^ e)
```

```
{// Побудова графіку:
      Months = gcnew array <String ^> { "Січ", "Лют", "Бер", "Квіт",
            "Трав", "Черв", "Лип", "Серп", "Вер", "Жовт", "Лист", "Груд"};
      Sales = gcnew array <int> {335, 414, 572, 629, 750, 931, 753, 599,
            422, 301, 245, 155};
      ВідступЛіворуч = 35; ВідступПраворуч = 15;
      ВідступЗнизу = 20; ВідступЗверху = 10;
      Растр = gcnew Bitmap(pictureBox1->Width, pictureBox1->Height,
                  pictureBox1->CreateGraphics());
      //Рамка навколо графіка
      pictureBox1->BorderStyle = BorderStyle::FixedSingle;
      YГоризОсі = pictureBox1->Height - ВідступЗнизу;
      XВертОсі = pictureBox1->Width - ВідступЛіворуч;
      Xmax = pictureBox1->Width - ВідступПраворуч;
      ДовжинаГоризОсі = pictureBox1->Width - (ВідступЛіворуч
            + ВідступПраворуч);
      ДовжинаВертОсі = YГоризОсі - ВідступЗверху;
      ГоризКрок = (double)(ДовжинаГоризОсі / Sales->Length + 3);
      ВертКрок = (int)(ДовжинаВертОсі / 10);
      XПочЕпюри = ВідступЛіворуч;//Початкове значення графіку по вісі X
      // Послідовно викликаємо наступні процедури:
      Графіка = Graphics::FromImage(Растр);
      МалюємоВісі();
      МалюємоГоризЛінії();
      МалюємоВертЛінії();
      МалюємоЕпюру();
      pictureBox1->Image = Растр;
      // Звільняємо ресурси, використовувані об'єктом класу Graphics:
      delete Графіка; // - Еквівалент C#: Графіка.Dispose
} // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private: void МалюємоВісі()
{
      Pen ^ Перо = gcnew Pen(Color::Black, 2);
      // Малювання вертикальної осі координат:
      Графіка->DrawLine(Перо, ВідступЛіворуч, YГоризОсі, ВідступЛіворуч,
            ВідступЗверху);
      // Малювання горизонтальної осі координат:
      Графіка->DrawLine(Перо, ВідступЛіворуч, YГоризОсі, Xmax,
            YГоризОсі);
      Drawing::Font ^ Шрифт = gcnew Drawing::Font("Arial", 8);
      // Підписуємо значення по вісі Y
      for (int i = 1; i <= 10; i++)
      {// Малюємо "вусики" на вертикальній координатної осі:
            int Y = YГоризОсі - i * ВертКрок;
            Графіка->DrawLine(Перо, ВідступЛіворуч - 5, Y,
                  ВідступЛіворуч, Y);
            // Підписуємо значення продажів через кожні 100 одиниць:
            Графіка->DrawString((i * 100).ToString(), Шрифт,
                  Brushes::Black, 2, Y - 5.F);
      }
      // Підписуємо значення по вісі X
      for (int i = 0; i <= Months->Length - 1; i++)
```

```
        {// Малюємо "вусики" на горизонтальній координатної осі:
            int X = ХПочЕпюри + (int)(ГоризКрок * (i + 1));
            Графіка->DrawLine(Перо, X, YГоризОсі + 5, X, YГоризОсі );
            // Підписуємо місяці на горизонтальній осі:
            Графіка->DrawString(Months[i], Шрифт, Brushes::Black,
                ВідступЛіворуч - 10.F + (int)(i * ГоризКрок),
                YГоризОсі + 4.F);
        }
    } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

    private: void МалюємоГоризЛінії()
    {
        Pen ^ ТонкеПеро = gcnew Pen(Color::LightGray, 1);
        for (int i = 1; i <= 10; i++)
        {// Малюємо горизонтальні майже "прозорі" лінії:
            int Y = YГоризОсі - ВертКрок * i;
            Графіка->DrawLine(ТонкеПеро, ВідступЛіворуч, Y, Xmax, Y);
        }
        delete ТонкеПеро; // - Еквівалент C#: ТонкоеПеро.Dispose();
    } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

    private: void МалюємоВертЛінії()
    {// Малюємо вертикальні майже "прозорі" лінії
        Pen ^ ТонкеПеро = gcnew Pen(Color::Bisque, 1);
        for (int i = 0; i <= Months->Length - 1; i++)
        {
            int X = ХПочЕпюри + (int)(ГоризКрок * i);
            Графіка->DrawLine(ТонкеПеро, X, ВідступЗверху, X,
                YГоризОсі - 4);
        }
        delete ТонкеПеро;
    } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

    private: void МалюємоЕпюру()
    {
        double ВертМасштаб = (double)ДовжинаВертОсі / 1000;
        // Значення ординат на екрані:
        array <int> ^ Y = gcnew array <int>(Sales->Length);
        // Значення абсцис на екрані:
        array <int> ^ X = gcnew array <int>(Sales->Length);
        for (int i = 0; i <= Sales->Length - 1; i++)
        {// Обчислюємо графічні координати точок:
            Y[i] = YГоризОсі - (int)(Sales[i] * ВертМасштаб);
            // Віднімаємо значення продажів, оскільки вісь Y екрану
            // спрямована вниз
            X[i] = ХПочЕпюри + (int)(ГоризКрок * i);
        }
        // Малюємо перше коло:
        Pen ^ Перо = gcnew Pen(Color::Blue, 3);
        Графіка->DrawEllipse(Перо, X[0] - 2, Y[0] - 2, 4, 4);
        for (int i = 0; i <= Sales->Length - 2; i++)
        {// Цикл по лініях між точками:
            Графіка->DrawLine(Перо, X[i], Y[i], X[i + 1], Y[i + 1]);
```

```
            // Віднімаємо 2, оскільки діаметр (ширина) точки = 4:
            Графіка->DrawEllipse(Перо, X[i + 1] - 2, Y[i + 1] - 2, 4, 4);
        }
    } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

    private: System::Void MyForm_Load(System::Object^  sender,
    System::EventArgs^  e)
    { // Встановлення заголовку вікна та тексту кнопки
    // при запуску вікна програми
        this->Text = "Побудова графіка";
        button1->Text = "Намалювати графік";
    }
```

Приклад реалізація завдання на мові C#.

Приклад коду C#

```
// Програма малює графік продажів по місяцях. Зрозуміло, що таким же
// чином можна побудувати будь-який графік по точках для інших прикладних
// цілей
// ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
// Вихідні дані для побудови графіка (тобто вихідні точки):
String[] Months;
int[] Sales;
Graphics Графіка;
// Далі, створюємо об'єкт Bitmap, який має
// Той же розмір і роздільну здатність, що і PictureBox
Bitmap Растр;
int ВідступЛіворуч, ВідступПраворуч, ВідступЗнизу, ВідступЗверху;
int ДовжинаВертОсі, ДовжинаГоризОсі, YГоризОсі, XВертОсі, Xmax,
XПочЕпюри;
// Крок градуювання по горизонтальній і вертикальній осях:
double ГоризКрок;
int ВертКрок;
// ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private void Form1_Load(object sender, EventArgs e)
{// Встановлення заголовку вікна та тексту кнопки
// при запуску вікна програми
    this.Text = "Побудова графіка";
    button1.Text = "Намалювати графік";
}

public Form1()
{
    InitializeComponent();
}

private void button1_Click(object sender, EventArgs e)
{// Побудова графіку:
    Months = new String[] {"Січ", "Лют", "Бер", "Квіт",
    "Трав", "Черв", "Лип", "Серп", "Вер", "Жовт", "Лист", "Груд"};
```

```
        Sales = new int[] {335, 414, 572, 629, 750, 931, 753, 599,
              422, 301, 245, 155};
        ВідступЛіворуч = 35; ВідступПраворуч = 15;
        ВідступЗнизу = 20; ВідступЗверху = 10;
        Растр = new Bitmap(pictureBox1.Width, pictureBox1.Height,
              pictureBox1.CreateGraphics());
        //Рамка навколо графіка
        pictureBox1.BorderStyle = BorderStyle.FixedSingle;
        YГоризОсі = pictureBox1.Height - ВідступЗнизу;
        XВертОсі = pictureBox1.Width - ВідступЛіворуч;
        Xmax = pictureBox1.Width - ВідступПраворуч;
        ДовжинаГоризОсі = pictureBox1.Width - (ВідступЛіворуч
              + ВідступПраворуч);
        ДовжинаВертОсі = YГоризОсі - ВідступЗверху;
        ГоризКрок = (double)(ДовжинаГоризОсі / Sales.Length + 3);
        ВертКрок = (int)(ДовжинаВертОсі / 10);
        XПочЕпюри = ВідступЛіворуч;//Початкове значення графіку по вісі X
        // Послідовно викликаємо наступні процедури:
        Графіка = Graphics.FromImage(Растр);
        МалюємоВісі();
        МалюємоГоризЛінії();
        МалюємоВертЛінії();
        МалюємоЕпюру();
        pictureBox1.Image = Растр;
        // Звільняємо ресурси, використовувані об'єктом класу Graphics:
        Графіка.Dispose(); // - Еквівалент C++: delete Графіка;
  } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

  private void МалюємоВісі()
  {
        Pen Перо = new Pen(Color.Black, 2);
        // Малювання вертикальної осі координат:
        Графіка.DrawLine(Перо, ВідступЛіворуч, YГоризОсі, ВідступЛіворуч,
              ВідступЗверху);
        // Малювання горизонтальної осі координат:
        Графіка.DrawLine(Перо, ВідступЛіворуч, YГоризОсі, Xmax,
              YГоризОсі);
        System.Drawing.Font Шрифт = new System.Drawing.Font("Arial", 8);
        // Підписуємо значення по вісі Y
        for (int i = 1; i <= 10; i++)
        {// Малюємо "вусики" на вертикальній координатної осі:
              int Y = YГоризОсі - i * ВертКрок;
              Графіка.DrawLine(Перо, ВідступЛіворуч - 5, Y,
                    ВідступЛіворуч, Y);
              // Підписуємо значення продажів через кожні 100 одиниць:
              Графіка.DrawString((i * 100).ToString(), Шрифт,
                    Brushes.Black, 2, Y - 5);
        }
        // Підписуємо значення по вісі X
        for (int i = 0; i <= Months.Length - 1; i++)
        {// Малюємо "вусики" на горизонтальній координатної осі:
              int X = XПочЕпюри + (int)(ГоризКрок * (i + 1));
              Графіка.DrawLine(Перо, X, YГоризОсі + 5, X, YГоризОсі);
```

```
            // Підписуємо місяці на горизонтальній осі:
            Графіка.DrawString(Months[i], Шрифт, Brushes.Black,
                  ВідступЛіворуч - 10 + (int)(i * ГоризКрок),
                  YГоризОсі + 4);
      }
} // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private void МалюємоГоризЛінії()
{
      Pen ТонкеПеро = new Pen(Color.LightGray, 1);
      for (int i = 1; i <= 10; i++)
      {// Малюємо горизонтальні майже "прозорі" лінії:
            int Y = YГоризОсі - ВертКрок * i;
            Графіка.DrawLine(ТонкеПеро, ВідступЛіворуч, Y, Xmax, Y);
      }
      ТонкеПеро.Dispose();
} // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private void МалюємоВертЛінії()
{// Малюємо вертикальні майже "прозорі" лінії
      Pen ТонкеПеро = new Pen(Color.Bisque, 1);
      for (int i = 0; i <= Months.Length - 1; i++)
      {
            int X = XПочЕпюри + (int)(ГоризКрок * i);
            Графіка.DrawLine(ТонкеПеро, X, ВідступЗверху, X,
                  YГоризОсі - 4);
      }
      ТонкеПеро.Dispose();
} // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private void МалюємоЕпюру()
{
      double ВертМасштаб = (double)ДовжинаВертОсі / 1000;
      // Значення ординат на екрані:
      int[] Y = new int[Sales.Length];
      // Значення абсцис на екрані:
      int[] X = new int[Sales.Length];
      for (int i = 0; i <= Sales.Length - 1; i++)
      {// Обчислюємо графічні координати точок:
            Y[i] = YГоризОсі - (int)(Sales[i] * ВертМасштаб);
            // Віднімаємо значення продажів, оскільки вісь Y екрану
            // спрямована вниз
            X[i] = XПочЕпюри + (int)(ГоризКрок * i);
      }
      // Малюємо перше коло:
      Pen Перо = new Pen(Color.Blue, 3);
      Графіка.DrawEllipse(Перо, X[0] - 2, Y[0] - 2, 4, 4);
      for (int i = 0; i <= Sales.Length - 2; i++)
      {// Цикл по лініях між точками:
            Графіка.DrawLine(Перо, X[i], Y[i], X[i + 1], Y[i + 1]);
            // Віднімаємо 2, оскільки діаметр (ширина) точки = 4:
            Графіка.DrawEllipse(Перо, X[i + 1] - 2, Y[i + 1] - 2, 4, 4);
      }
```

```
} // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
```

Після того, як програмний код набраний, потрібно перевірити чи присвоєна подія Click для об'єкту Button1 у вікні Властивості (Properties) на вкладці Events (події). Якщо ні, то потрібно обрати button1_Click зі спадаючого списку у полі події Click. Аналогіно потрібно перевірити подію Load для форми і за необхідності вказати потрібну подію зі списку.

Якщо все зроблено вірно, то після запуску додатку з'явиться пусте вікно з кнопкою. По натисканню кнопки «Намалювати графік» у вікні з'явиться графічна залежність (рис. 7.5).

Якщо розмір графіку замалий, можна розтягти вікно і знову натиснути кнопку «Намалювати графік». Графік буде перебудований у новому масштабі таким чином, щоб зайняти весь вільний простір вікна.

Як видно з тексту програми, спочатку оголошуємо деякі змінні так, щоб вони були видні з усіх процедур класу. Строковий масив Months містить назви місяців, які користувач нашого програмного коду може змінювати в залежності від контексту графіка, який будується. У будь-якому випадку записані рядки в цьому масиві будуть відображатися по горизонтальній осі графіка. Масив цілих чисел Sales містить обсяги продажів по кожному місяцю, вони відповідають ординатам графіка. Обидва масиву повинні мати однакову розмірність, але не обов'язково рівну дванадцяти.

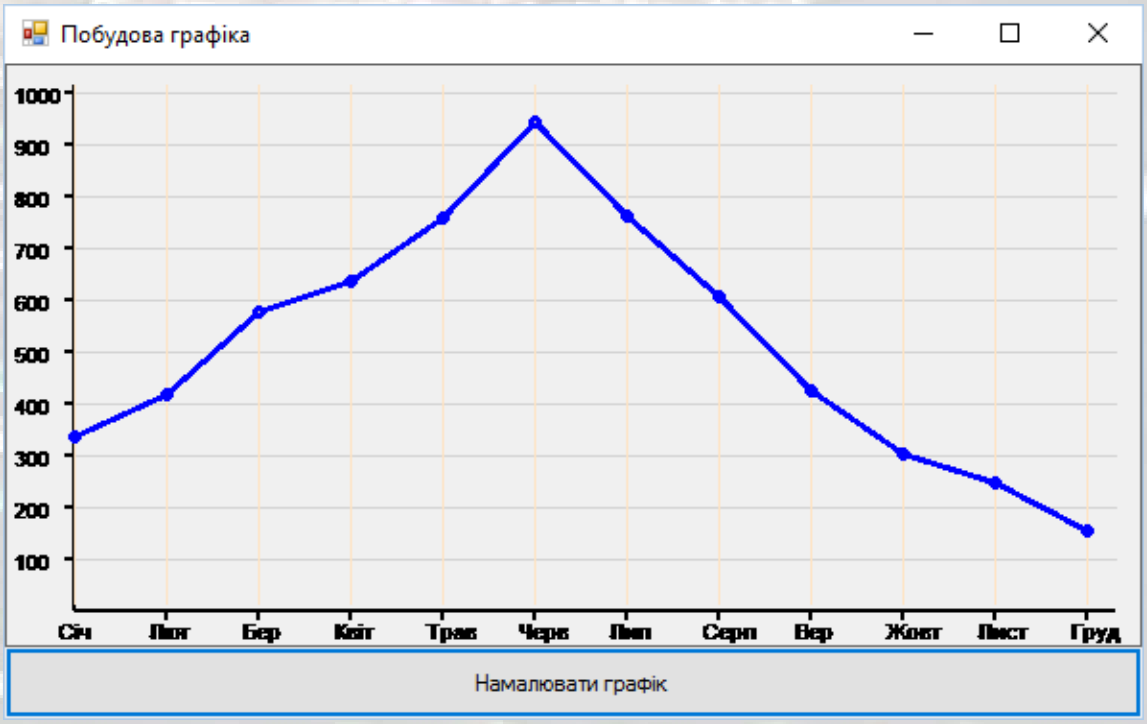


Рисунок 7.5 – Вікно програми побудови графіка

При обробці події натискання кнопки Button створюємо об'єкт класу Graphics, використовуючи елемент управління PictureBox (графічне поле), а потім, викликаючи відповідні процедури, поетапно малюємо координатні осі, сітку з горизонтальних і вертикальних ліній і безпосередньо епюру. Щоб успішно, мінімальними зусиллями і з можливістю подальшого вдосконалення програми

побудувати графік, слід якомога зрозуміліше назвати деякі ключові інтервали і координати на малюнку. Назви цих інтервалів повинні бути змістовними. Скажімо, змінна ВідступЛіворуч зберігає число пікселів, на яке слід відступити, щоб побудувати на графіку, наприклад, вертикальну вісь продажів. Крім очевидних назв згадаємо змінну YГоріз0сі, це графічна ордината (вісь x спрямована зліва направо, а вісь y - зверху вниз) горизонтальної осі графіка, на якій підписуються місяці. Змінна Xmax містить в собі значення максимальної абсциси (див. рис. 7.5), правіше якої вже ніяких побудов немає. Змінна ХПочЕпюри - це значення абсциси першої побудованої точки графіка. Використовуючі такі зрозумілі з контексту назви змінних, та ще й українською мовою, ми значно полегшуємо весь процес програмування, спрощуємо супровід і модифікацію програми. Текст програми наведений в лістингу 7.5 (C++/CLI) та в лістингу 7.6 (C#).

7.4 Створення багатосторінкового інтерфейсу

Завдання 5.

Необхідно створити проект Windows Forms, в якому реалізувати відкриття різних типів файлів з використанням діалогового вікна OpenFileDialog на різних сторінках TabPage елементу TabControl.

Розглянемо приклад створення програми, яка по натисканню кнопки відкриватиме графічні файли чи файли формату RTF в залежності від того, яка сторінка об'єкту класу TabControl активна.

Для вирішення цього завдання у C++/CLI запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms.

Для створення додатку C# необхідно у діалоговому вікні Створення проекту (New Project) обрати в лівій частині Встановлені ⇒ Visual C# ⇒ Windows Desktop (Installed ⇒ Visual C# ⇒ Windows Desktop). Надалі з переліку доступних шаблонів проектів обрати пункт Windows Forms App (.NET Framework).

Перенесемо з Панелі елементів (Toolbox) в проектовану форму елемент управління кнопка Button. Для властивості Dock кнопки у вікні Properties задамо значення Bottom, щоб кнопка розтягувалась вздовж всієї нижньої частини форми. У властивості Text кнопки вкажемо значення «Відкрити файл».

Перетягнемо з розділу Контейнери вікна Панелі елементів об'єкт класу TabControl. Для властивості Dock елементу TabControl у вікні Properties задамо значення Fill, щоб вкладки сторінок розтягувалась по всій вільній області форми.

На першій сторінці в об'єкті tabPage1 класу TabPage розмістимо об'єкт класу PictureBox з розділу Стандартні елементи управління вікна

Панелі елементів. Для властивості Dock об'єкту pictureBox1 задамо значення Fill. Для властивості Text об'єкту tabPage1 вкажемо значення «Графічний файл» (рис. 7.6).

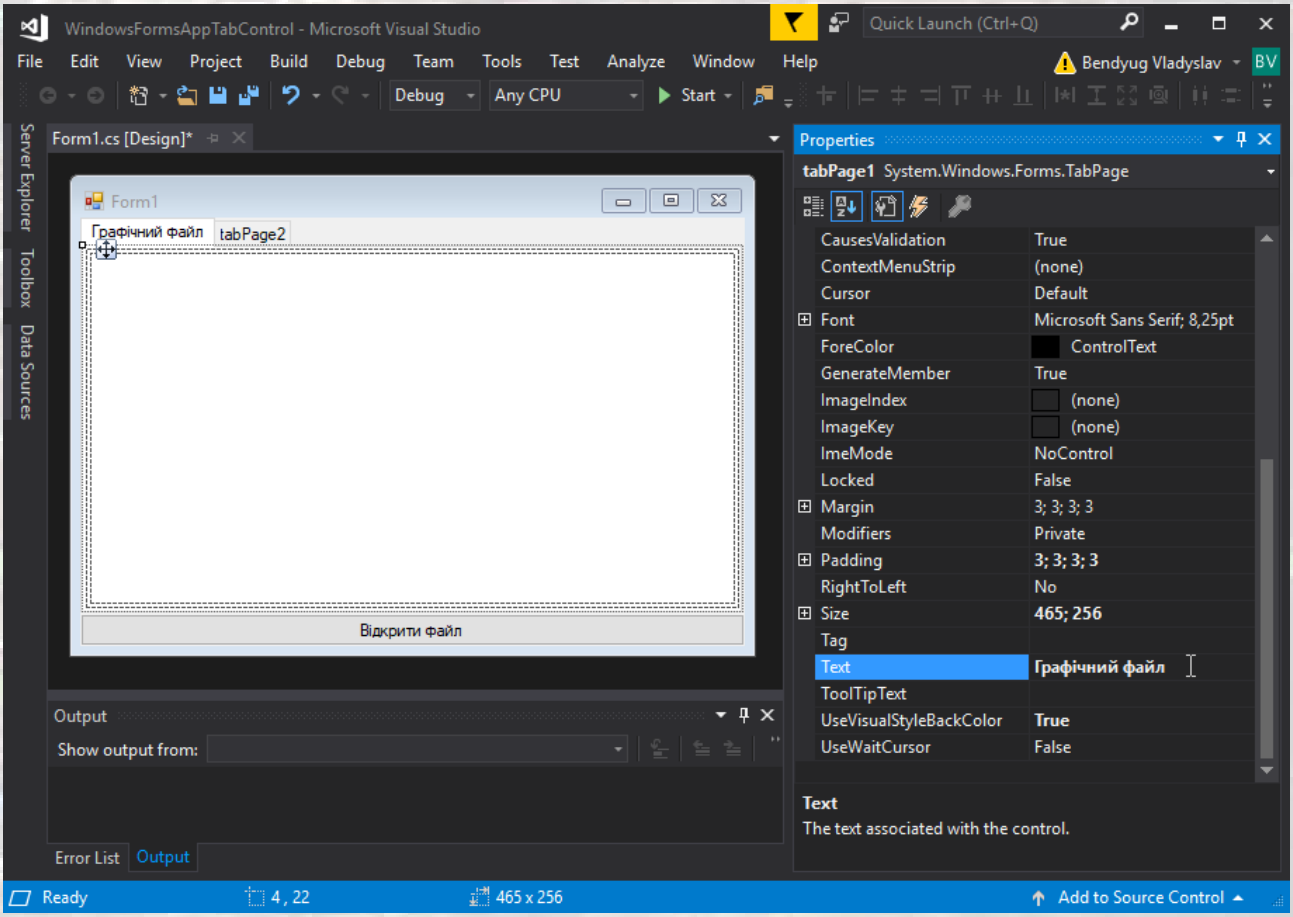


Рисунок 7.6 – Додавання об'єктів у форму та зміна їх властивостей

На другій сторінці в об'єкті tabPage2 класу TabPage розмістимо об'єкт класу RichTextBox з розділу Стандартні елементи управління вікна Панелі елементів. Для властивості Dock об'єкту richTextBox1 задамо значення Fill. Для властивості Text об'єкту tabPage2 вкажемо значення «Файл RTF».

Надалі в обробник події Click кнопки додамо наступний програмний код.

Приклад коду C++/CLI

```
private: System::Void buttonВідкрити_Click(System::Object^  sender,
System::EventArgs^  e) {
     if (tabControl1->SelectedIndex == 0)
     //Перевірка чи активна перша вкладка
     {    //Тоді відкриваємо графічний файл
          //Створюємо об'єкт класу OpenFileDialog
          OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
          //Встановлюємо в діалоговому вікні фільтр для відкриття файлів
          openFileDialog1->Filter = L"Файли зображень |
               *.bmp;*.jpg;*.gif; *.png";
          if (openFileDialog1->ShowDialog() == System::Windows::Forms
```

```
                    ::DialogResult::OK  && openFileDialog1->FileName
                    ->Length > 0)
                //Якщо користувач обрав в діалоговому вікні файл
                //і його ім'я не пусте
                {
                    //Відкриваємо графічний файл в об'єкті pictureBox1
                    pictureBox1->Image = Image::FromFile(openFileDialog1
                        ->FileName);
                    //Розміщуємо ім'я відкритого файлу в заголовку вкладки
                    tabPage1->Text = Path::GetFileName(openFileDialog1
                        ->FileName);
                }
            }
            else //Активна друга вкладка
            {   //Тоді відкриваємо файл *.rtf
                //Створюємо об'єкт класу OpenFileDialog
                OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
                //Встановлюємо в діалоговому вікні фільтр для відкриття файлів
                openFileDialog1->Filter = L"Текстові файли | *.rtf";
                if (openFileDialog1->ShowDialog() == System::Windows::Forms
                    ::DialogResult::OK  && openFileDialog1->FileName
                    ->Length > 0)
                //Якщо користувач обрав в діалоговому вікні файл
                //і його ім'я не пусте
                {
                    //Відкриваємо файл формату RTF в об'єкті richTextBox1
                    richTextBox1->LoadFile(openFileDialog1->FileName);
                    //Розміщуємо ім'я відкритого файлу в заголовку вкладки
                    tabPage2->Text = Path::GetFileName(openFileDialog1
                        ->FileName);
                }
            }
        }
```

Приклад коду C#

```
private void buttonВідкрити_Click (object sender, EventArgs e)
{
    if (tabControl1.SelectedIndex == 0)
    //Перевірка чи активна перша вкладка
    {   //Тоді відкриваємо графічний файл
        //Створюємо об'єкт класу OpenFileDialog
        OpenFileDialog openFileDialog1 = new OpenFileDialog();
        //Встановлюємо в діалоговому вікні фільтр
        //для відкриття файлів
        openFileDialog1.Filter = "Файли зображень | *.bmp;*.jpg;
            *.gif; *.png";
        if (openFileDialog1.ShowDialog() == System.Windows.Forms
            .DialogResult.OK && openFileDialog1.FileName.Length > 0)
        //Якщо користувач обрав в діалоговому вікні файл
        //і його ім'я не пусте
        {
            //Відкриваємо графічний файл в об'єкті pictureBox1
```

```
                    pictureBox1.Image = Image
                        .FromFile(openFileDialog1.FileName);
                    //Розміщуємо ім'я відкритого файлу в заголовку вкладки
                    tabPage1.Text = System.IO.Path
                        .GetFileName(openFileDialog1.FileName);
                }
            }
            else //Активна друга вкладка
            {//Відкриваємо файл *.rtf і створюємо об'єкт класу OpenFileDialog
                OpenFileDialog openFileDialog1 = new OpenFileDialog();
                //Встановлюємо в діалоговому вікні фільтр для відкриття файлів
                openFileDialog1.Filter = "Текстові файли | *.rtf";
                if (openFileDialog1.ShowDialog() == System.Windows.Forms
                    .DialogResult.OK && openFileDialog1.FileName.Length > 0)
                //Якщо користувач обрав в діалоговому вікні файл
                //і його ім'я не пусте
                {
                    // Відкриваємо файл формату RTF в об'єкті richTextBox1
                    richTextBox1.LoadFile(openFileDialog1.FileName);
                    //Розміщуємо ім'я відкритого файлу в заголовку вкладки
                    tabPage2.Text = System.IO.Path
                        .GetFileName(openFileDialog1.FileName);
                }
            }
        }
```

При натисканні кнопки buttonВідкрити перевіряється яка з вкладок активна

Приклад коду C#

```
//Перевірка чи активна перша вкладка
if (tabControl1.SelectedIndex == 0)
{    //Тоді відкриваємо графічний файл

}
else //Активна друга вкладка
{    //Тоді відкриваємо файл *.rtf

}
```

Якщо активна перша вкладка, створюємо об’єкт класу діалогового вікна відкриття файлу OpenFileDialog і задаємо фільтр файлів в діалоговому вікні, щоб користувач не міг відкрити файли неприпустимого типу.

Приклад коду C#

```
//Створюємо об'єкт класу OpenFileDialog
OpenFileDialog openFileDialog1 = new OpenFileDialog();
//Встановлюємо в діалоговому вікні фільтр для відкриття файлів
openFileDialog1.Filter = "Файли зображень | *.bmp;*.jpg;*.gif; *.png";
```

Після цього відкриваємо діалогове вікно і перевіряємо чи обрав користувач файл для відкриття.

Приклад коду C#

```
if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK
 && openFileDialog1.FileName.Length > 0)
//Якщо користувач обрав в діалоговому вікні файл і його ім'я не пусте
```

Якщо дана умова виконується, відкриваємо графічний файл в об'єкті pictureBox1 та записуємо ім'я відкритого файлу у заголовок вкладки.

Приклад коду C#

```
//Відкриваємо графічний файл в об'єкті pictureBox1
pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);
//Розміщуємо ім'я відкритого файлу в заголовку вкладки
tabPage1.Text = System.IO.Path.GetFileName(openFileDialog1.FileName);
```

Функція GetFileName() класу Path з простору імен System.IO виокремлює ім'я файлу з шляху та імені файлу, які зберігаються у властивості FileName об'єкту openFileDialog1. Для можливості використання даної функції потрібно підключити відповідний простір імен або вказувати повний шлях до цієї функції.

Приклад коду C++/CLI

```
//Підключення простору імен для методу Path::GetFileName()
using namespace System::IO;
```

Приклад коду C#

```
using System.IO;  //Підключення простору імен для методу Path.GetFileName()
```

Аналогічним чином виконується відкриття файлів RTF в об'єкті richTextBox1, якщо активною є друга вкладка tabPage2.

Приклад коду C#

```
//Відкриваємо файл формату RTF в об'єкті richTextBox1
richTextBox1.LoadFile(openFileDialog1.FileName);
//Розміщуємо ім'я відкритого файлу в заголовку вкладки
tabPage2.Text = System.IO.Path.GetFileName(openFileDialog1.FileName);
```

Результат роботи програми зображений на рис. 7.7.

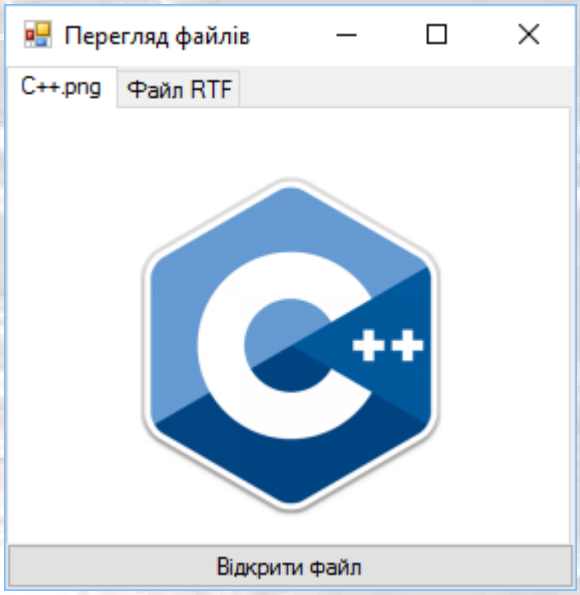


Рисунок 7.7 – Вікно програми відкриття різних типів файлів на двох окремих сторінках

Для того, щоб відкриті зображення масштабувались в межах розмірів вікна, для об'єкту `pictureBox1` було встановлене значення `Zoom` властивості `SizeMode`.

Повний текст програми наведений нижче в лістингу 7.7 (C++/CLI) та лістингах 7.8-7.9 (C#).

Програмний код

Лістинг 7.1

Приклад коду C++/CLI

```
//Програма ілюстрації відкриття графічного файлу у формі

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  button1;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->SuspendLayout();
			//
			// button1
			//
```

```
                    this->button1->Location = System::Drawing::Point(96, 226);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(75, 23);
                    this->button1->TabIndex = 0;
                    this->button1->Text = L"button1";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->button1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::Form1_Load);
                    this->ResumeLayout(false);

                }
#pragma endregion
        // Найпростіше виведення зображення в форму
        private: System ::
                Void Form1_Load(System::Object ^ sender, System::EventArgs ^ e)
                {
                    // Подія завантаження форми:
                    MyForm::Text = "Малюнок";
                    button1->Text = "Показати малюнок";
                }
        private: System::Void button1_Click(System::Object ^ sender,
                System::EventArgs ^ e)
                {
                    // Подія "клацання на кнопці"
                    Image ^ Малюнок = gcnew Bitmap("D:\\DOCs\\PICTUREs\\ЕмблемаХТФ.png");
                    // Створення графічного об'єкта:
                    Graphics ^ Графіка = this->CreateGraphics();
                    // Або Graphics ^ Графіка = CreateGraphics ();
                    Графіка->DrawImage(Малюнок, 5, 5);
                }
        };
}
```

Лістинг 7.2

Приклад коду C#

```
//Програма ілюстрації відкриття графічного файлу у формі

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsAppImage
{// Найпростіше виведення зображення в форму
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // Подія завантаження форми:
            Text = "Малюнок";
            button1.Text = "Показати малюнок";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // Подія "клацання на кнопці"
            Image Малюнок = new
Bitmap("D:\\VIO77\\Зображення\\C#_1.jpg");//("D:\\DOCs\\PICTUREs\\ЕмблемаХТФ.png");
            // Створення графічного об'єкта:
            Graphics Графіка = this.CreateGraphics();
            // Або Graphics Графіка = CreateGraphics();
            Графіка.DrawImage(Малюнок, 5, 5);

        }
    }
}
```

Лістинг 7.3

Приклад коду C++/CLI

```
//Приклад використання діалогових вікон OpenFileDialog та SaveFileDialog
//а також відкриття графічних зображень в об'єкті PictureBox

#pragma once

namespace WindowsForms {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Panel^  panel1;
	private: System::Windows::Forms::PictureBox^  pictureBox1;

	private: System::Windows::Forms::Panel^  panel2;
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::Button^  button2;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->panel1 = (gcnew System::Windows::Forms::Panel());
			this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
```

```
            this->panel2 = (gcnew System::Windows::Forms::Panel());
            this->button2 = (gcnew System::Windows::Forms::Button());
            this->button1 = (gcnew System::Windows::Forms::Button());
            this->panel1->SuspendLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->BeginInit();
            this->panel2->SuspendLayout();
            this->SuspendLayout();
            //
            // panel1
            //
            this->panel1->Controls->Add(this->pictureBox1);
            this->panel1->Location = System::Drawing::Point(0, 0);
            this->panel1->Name = L"panel1";
            this->panel1->Size = System::Drawing::Size(239, 127);
            this->panel1->TabIndex = 0;
            //
            // pictureBox1
            //
            this->pictureBox1->Location = System::Drawing::Point(50, 25);
            this->pictureBox1->Name = L"pictureBox1";
            this->pictureBox1->Size = System::Drawing::Size(100, 50);
            this->pictureBox1->TabIndex = 2;
            this->pictureBox1->TabStop = false;
            //
            // panel2
            //
            this->panel2->Controls->Add(this->button2);
            this->panel2->Controls->Add(this->button1);
            this->panel2->Location = System::Drawing::Point(0, 146);
            this->panel2->Name = L"panel2";
            this->panel2->Size = System::Drawing::Size(239, 115);
            this->panel2->TabIndex = 2;
            //
            // button2
            //
            this->button2->Location = System::Drawing::Point(99, 24);
            this->button2->Name = L"button2";
            this->button2->Size = System::Drawing::Size(75, 23);
            this->button2->TabIndex = 3;
            this->button2->Text = L"button2";
            this->button2->UseVisualStyleBackColor = true;
            this->button2->Click += gcnew System::EventHandler(this,
&MyForm::button2_Click);
            //
            // button1
            //
            this->button1->Location = System::Drawing::Point(50, 72);
            this->button1->Name = L"button1";
            this->button1->Size = System::Drawing::Size(138, 31);
            this->button1->TabIndex = 2;
            this->button1->Text = L"button1";
            this->button1->UseVisualStyleBackColor = true;
            this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
            //
            // MyForm
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(284, 261);
            this->Controls->Add(this->panel2);
            this->Controls->Add(this->panel1);
            this->Name = L"MyForm";
            this->Text = L"MyForm";
            this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
```

```
                    this->Resize += gcnew System::EventHandler(this,
&MyForm::MyForm_Resize);
                    this->panel1->ResumeLayout(false);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->EndInit();
                    this->panel2->ResumeLayout(false);
                    this->ResumeLayout(false);

             }

#pragma endregion
      private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
             OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
             if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
             {
                    pictureBox1->Image = Image::FromFile(openFileDialog1->FileName);
                    // Примусово викликаємо обробник події Resize форми для встановлення
розмірів об'єктів panel1 та button1
                    MyForm_Resize(nullptr, nullptr);
             }

      }

      private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
             this->Text = "Графічний файл";
             button1->Text = "Відкрити";
             button2->Text = "Зберегти";
             panel2->Height = 40;
             panel2->Dock = System::Windows::Forms::DockStyle::Bottom;
             button1->Dock = System::Windows::Forms::DockStyle::Right;
             panel1->Dock = System::Windows::Forms::DockStyle::Top;
             pictureBox1->SizeMode = PictureBoxSizeMode::AutoSize;
             panel1->AutoScroll = true; // Дозволяємо прокрутку зображення
             button2->Dock = System::Windows::Forms::DockStyle::Fill;
      }

      private: System::Void MyForm_Resize(System::Object^  sender, System::EventArgs^  e) {
             // Встановлення розмірів об'єктів panel1 та button1
             panel1->Height = this->Height - panel2->Height - 40;
             button1->Width = this->panel2->Width / 2;
      }

      private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
             // Відобразити вікно SaveFileDialog щоб користувач міг зберегти зображення
             SaveFileDialog ^ saveFileDialog1 = gcnew SaveFileDialog();
             saveFileDialog1->Filter = "Зображення JPeg|*.jpg|Зображення
Bitmap|*.bmp|Зображення Gif|*.gif";
             saveFileDialog1->Title = "Зберегти файл зображення";
             saveFileDialog1->ShowDialog();
             // Якщо ім'я файлу не пустий рядок, зберегти зображення
             if (saveFileDialog1->FileName != "")
             {
                    // Зберегти зображення через FileStream, створений методом OpenFile
                    System::IO::FileStream ^ fs =
safe_cast<System::IO::FileStream^>(saveFileDialog1->OpenFile());
                    // Можна зберегти зображення у відповідному форматі ImageFormat на
основі
                    // типу файлу, обраного в діалоговому вікні.
                    // Тип файлу береться з властивості FilterIndex
                    switch (saveFileDialog1->FilterIndex)
                    {
                    case 1:
                           this->pictureBox1->Image->Save(fs,
                                  System::Drawing::Imaging::ImageFormat::Jpeg);
                           break;
                    case 2:
                           this->pictureBox1->Image->Save(fs,
```

```
                                System::Drawing::Imaging::ImageFormat::Bmp);
                        break;
                    case 3:
                        this->pictureBox1->Image->Save(fs,
                                System::Drawing::Imaging::ImageFormat::Gif);
                        break;
                    }
                    fs->Close();
                }
            }
        };
        }
```

Лістинг 7.4

Приклад коду C#

```
//Приклад використання діалогових вікон OpenFileDialog та SaveFileDialog
//а також відкриття графічних зображень в об'єкті PictureBox

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp_OpenFileDialog
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Text = "Test";
            OpenFileDialog openFileDialog1 = new OpenFileDialog();
            if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);
                // Примусово викликаємо обробник події Resize форми
                //для встановлення розмірів об'єктів panel1 та button1
                Form1_Resize(null, null);
            }

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.Text = "Графічний файл";
            button1.Text = "Відкрити";
            button2.Text = "Зберегти";
            panel2.Height = 40;
            panel2.Dock = System.Windows.Forms.DockStyle.Bottom;
            button1.Dock = System.Windows.Forms.DockStyle.Right;
            panel1.Dock = System.Windows.Forms.DockStyle.Top;
            pictureBox1.SizeMode = PictureBoxSizeMode.AutoSize;
            panel1.AutoScroll = true; // Дозволяємо прокрутку зображення
            button2.Dock = System.Windows.Forms.DockStyle.Fill;
            Form1_Resize(null, null);

        }

        private void Form1_Resize(object sender, EventArgs e)
        {
            // Встановлення розмірів об'єктів panel1 та button1
            panel1.Height = this.Height - panel2.Height - 40;
            button1.Width = this.panel2.Width / 2;
        }

        private void button2_Click(object sender, EventArgs e)
        {
```

```
            // Відобразити вікно SaveFileDialog щоб користувач міг зберегти зображення
            SaveFileDialog saveFileDialog1 = new SaveFileDialog();
            saveFileDialog1.Filter =
    "Зображення JPeg|*.jpg|Зображення Bitmap|*.bmp|Зображення Gif|*.gif";
            saveFileDialog1.Title = "Зберегти файл зображення";
            saveFileDialog1.ShowDialog();
            // Якщо ім'я файлу не пустий рядок, зберегти зображення
            if (saveFileDialog1.FileName != "")
            {
                // Зберегти зображення через FileStream, створений методом OpenFile
                System.IO.FileStream fs = new System.IO.FileStream(saveFileDialog1.FileName,
System.IO.FileMode.Create);
                // Можна зберегти зображення у відповідному форматі ImageFormat на
                // основі типу файлу, обраного в діалоговому вікні.
                // Тип файлу береться з властивості FilterIndex
                switch (saveFileDialog1.FilterIndex)
                {
                    case 1:
                        this.pictureBox1.Image.Save(fs,
                            System.Drawing.Imaging.ImageFormat.Jpeg);
                        break;
                    case 2:
                        this.pictureBox1.Image.Save(fs,
                            System.Drawing.Imaging.ImageFormat.Bmp);
                        break;
                    case 3:
                        this.pictureBox1.Image.Save(fs,
                            System.Drawing.Imaging.ImageFormat.Gif);
                        break;
                }
                fs.Close();
            }
        }
    }
}
```

Лістинг 7.5

Приклад коду C++/CLI

```
//Приклад побудови графіку за допомогою класу Graphics

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Panel^  panel1;
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::PictureBox^  pictureBox1;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->panel1 = (gcnew System::Windows::Forms::Panel());
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
			this->panel1->SuspendLayout();
```

```
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->BeginInit();
                    this->SuspendLayout();
                    //
                    // panel1
                    //
                    this->panel1->Controls->Add(this->button1);
                    this->panel1->Dock = System::Windows::Forms::DockStyle::Bottom;
                    this->panel1->Location = System::Drawing::Point(0, 207);
                    this->panel1->Name = L"panel1";
                    this->panel1->Size = System::Drawing::Size(284, 54);
                    this->panel1->TabIndex = 0;
                    //
                    // button1
                    //
                    this->button1->Dock = System::Windows::Forms::DockStyle::Fill;
                    this->button1->Location = System::Drawing::Point(0, 0);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(284, 54);
                    this->button1->TabIndex = 0;
                    this->button1->Text = L"button1";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // pictureBox1
                    //
                    this->pictureBox1->Dock = System::Windows::Forms::DockStyle::Fill;
                    this->pictureBox1->Location = System::Drawing::Point(0, 0);
                    this->pictureBox1->Name = L"pictureBox1";
                    this->pictureBox1->Size = System::Drawing::Size(284, 207);
                    this->pictureBox1->TabIndex = 1;
                    this->pictureBox1->TabStop = false;
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->pictureBox1);
                    this->Controls->Add(this->panel1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->panel1->ResumeLayout(false);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->EndInit();
                    this->ResumeLayout(false);
                }
#pragma endregion
                // Програма малює графік продажів по місяцях. Зрозуміло, що таким же чином
                // Можна побудувати будь-який графік по точках для інших прикладних цілей
                // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
                // Вихідні дані для побудови графіка (тобто вихідні точки):
                array <String ^> ^ Months;
                array <int> ^ Sales;
                Graphics ^ Графіка;
                // Далі, створюємо об'єкт Bitmap, який має
                // Той же розмір і роздільну здатність, що і PictureBox
                Bitmap ^ Растр;
                int ВідступЛіворуч, ВідступПраворуч, ВідступЗнизу, ВідступЗверху;
                int ДовжинаВертОсі, ДовжинаГоризОсі, YГоризОсі, XВертОсі, Xmax, XПочЕпюри;
                // Крок градуювання по горизонтальній і вертикальній осях:
                double ГоризКрок;
                int ВертКрок;
                // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
```

```
	private: System::Void button1_Click(System::Object ^ sender,
		System::EventArgs ^ e)
	{// Побудова графіку:
		Months = gcnew array <String ^> { "Січ", "Лют", "Бер", "Квіт", "Трав",
			"Черв", "Лип", "Серп", "Вер", "Жовт", "Лист", "Груд"};
		Sales = gcnew array <int> {335, 414, 572, 629, 750, 931, 753, 599, 422, 301,
245, 155};
		ВідступЛіворуч = 35; ВідступПраворуч = 15;
		ВідступЗнизу = 20; ВідступЗверху = 10;
		Растр = gcnew Bitmap(pictureBox1->Width, pictureBox1->Height, pictureBox1-
>CreateGraphics());
		pictureBox1->BorderStyle = BorderStyle::FixedSingle;//Рамка навколо графіка
		YГоризОсі = pictureBox1->Height - ВідступЗнизу;
		XВертОсі = pictureBox1->Width - ВідступЛіворуч;
		Xmax = pictureBox1->Width - ВідступПраворуч;
		ДовжинаГоризОсі = pictureBox1->Width - (ВідступЛіворуч + ВідступПраворуч);
		ДовжинаВертОсі = YГоризОсі - ВідступЗверху;
		ГоризКрок = (double)(ДовжинаГоризОсі / Sales->Length + 3);
		ВертКрок = (int)(ДовжинаВертОсі / 10);
		XПочЕпюри = ВідступЛіворуч;//Початкове значення графіку по вісі X
					// Послідовно викликаємо наступні процедури:
		Графіка = Graphics::FromImage(Растр);
		МалюємоВісі();
		МалюємоГоризЛінії();
		МалюємоВертЛінії();
		МалюємоЕпюру();
		pictureBox1->Image = Растр;
		// Звільняємо ресурси, використовувані об'єктом класу Graphics:
		delete Графіка; // - Еквівалент C#: Графіка.Dispose
	} // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

	private: void МалюємоВісі()
	{
		Pen ^ Перо = gcnew Pen(Color::Black, 2);
		// Малювання вертикальної осі координат:
		Графіка->DrawLine(Перо, ВідступЛіворуч, YГоризОсі, ВідступЛіворуч,
ВідступЗверху);
		// Малювання горизонтальної осі координат:
		Графіка->DrawLine(Перо, ВідступЛіворуч, YГоризОсі, Xmax, YГоризОсі);
		Drawing::Font ^ Шрифт = gcnew Drawing::Font("Arial", 8);
		// Підписуємо значення по вісі Y
		for (int i = 1; i <= 10; i++)
		{// Малюємо "вусики" на вертикальній координатної осі:
			int Y = YГоризОсі - i * ВертКрок;
			Графіка->DrawLine(Перо, ВідступЛіворуч - 5, Y, ВідступЛіворуч, Y);
			// Підписуємо значення продажів через кожні 100 одиниць:
			Графіка->DrawString((i * 100).ToString(), Шрифт, Brushes::Black, 2, Y -
5.F);
		}
		// Підписуємо значення по вісі X
		for (int i = 0; i <= Months->Length - 1; i++)
		{// Малюємо "вусики" на горизонтальній координатної осі:
			int X = XПочЕпюри + (int)(ГоризКрок * (i + 1));
			Графіка->DrawLine(Перо, X, YГоризОсі + 5, X, YГоризОсі);
			// Підписуємо місяці на горизонтальній осі:
			Графіка->DrawString(Months[i], Шрифт, Brushes::Black, ВідступЛіворуч -
10.F + (int)(i * ГоризКрок), YГоризОсі + 4.F);
		}
	} // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

	private: void МалюємоГоризЛінії()
	{
		Pen ^ ТонкеПеро = gcnew Pen(Color::LightGray, 1);
		for (int i = 1; i <= 10; i++)
		{// Малюємо горизонтальні майже "прозорі" лінії:
			int Y = YГоризОсі - ВертКрок * i;
			Графіка->DrawLine(ТонкеПеро, ВідступЛіворуч, Y, Xmax, Y);
```

```
            }
            delete ТонкеПеро; // - Еквівалент C#: ТонкоеПеро.Dispose();
        } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

        private: void МалюємоВертЛінії()
        {// Малюємо вертикальні майже "прозорі" лінії
            Pen ^ ТонкеПеро = gcnew Pen(Color::Bisque, 1);
            for (int i = 0; i <= Months->Length - 1; i++)
            {
                int X = ХПочЕпюри + (int)(ГоризКрок * i);
                Графіка->DrawLine(ТонкеПеро, X, ВідступЗверху, X, УГоризОсі - 4);
            }
            delete ТонкеПеро;
        } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

        private: void МалюємоЕпюру()
        {
            double ВертМасштаб = (double)ДовжинаВертОсі / 1000;
            // Значення ординат на екрані:
            array <int> ^ Y = gcnew array <int>(Sales->Length);
            // Значення абсцис на екрані:
            array <int> ^ X = gcnew array <int>(Sales->Length);
            for (int i = 0; i <= Sales->Length - 1; i++)
            {// Обчислюємо графічні координати точок:
                Y[i] = УГоризОсі - (int)(Sales[i] * ВертМасштаб);
                // Віднімаємо значення продажів, оскільки вісь Y екрану спрямована вниз
                X[i] = ХПочЕпюри + (int)(ГоризКрок * i);
            }
            // Малюємо перше коло:
            Pen ^ Перо = gcnew Pen(Color::Blue, 3);
            Графіка->DrawEllipse(Перо, X[0] - 2, Y[0] - 2, 4, 4);
            for (int i = 0; i <= Sales->Length - 2; i++)
            {// Цикл по лініях між точками:
                Графіка->DrawLine(Перо, X[i], Y[i], X[i + 1], Y[i + 1]);
                // Віднімаємо 2, оскільки діаметр (ширина) точки = 4:
                Графіка->DrawEllipse(Перо, X[i + 1] - 2, Y[i + 1] - 2, 4, 4);
            }
        } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

    private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e)
        {// Встановлення заголовку вікна та тексту кнопки при запуску вікна програми
            this->Text = "Побудова графіка";
            button1->Text = "Намалювати графік";
        }
    };
    }
```

Лістинг 7.6

Приклад коду C#

```
//Приклад побудови графіку за допомогою класу Graphics

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp_Graphics
{
    public partial class Form1 : Form
    {

        // Програма малює графік продажів по місяцях. Зрозуміло, що таким же
        // чином можна побудувати будь-який графік по точках для інших прикладних
        // цілей
        // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
        // Вихідні дані для побудови графіка (тобто вихідні точки):
        String[] Months;
        int[] Sales;
        Graphics Графіка;
        // Далі, створюємо об'єкт Bitmap, який має
        // Той же розмір і роздільну здатність, що і PictureBox
        Bitmap Растр;
        int ВідступЛіворуч, ВідступПраворуч, ВідступЗнизу, ВідступЗверху;
        int ДовжинаВертОсі, ДовжинаГоризОсі, УГоризОсі, ХВертОсі, Хmax, ХПочЕпюри;
        // Крок градуювання по горизонтальній і вертикальній осях:
        double ГоризКрок;
        int ВертКрок;
        // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

        private void Form1_Load(object sender, EventArgs e)
        {// Встановлення заголовку вікна та тексту кнопки
         // при запуску вікна програми
            this.Text = "Побудова графіка";
            button1.Text = "Намалювати графік";
        }

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {// Побудова графіку:
            Months = new String[] {"Січ", "Лют", "Бер", "Квіт",
                "Трав", "Черв", "Лип", "Серп", "Вер", "Жовт", "Лист", "Груд"};
            Sales = new int[] {335, 414, 572, 629, 750, 931, 753, 599,
                422, 301, 245, 155};
            ВідступЛіворуч = 35; ВідступПраворуч = 15;
            ВідступЗнизу = 20; ВідступЗверху = 10;
            Растр = new Bitmap(pictureBox1.Width, pictureBox1.Height,
        pictureBox1.CreateGraphics());
            //Рамка навколо графіка
            pictureBox1.BorderStyle = BorderStyle.FixedSingle;
            УГоризОсі = pictureBox1.Height - ВідступЗнизу;
            ХВертОсі = pictureBox1.Width - ВідступЛіворуч;
```

```
        Xmax = pictureBox1.Width - ВідступПраворуч;
        ДовжинаГоризОсі = pictureBox1.Width - (ВідступЛіворуч
    + ВідступПраворуч);
        ДовжинаВертОсі = YГоризОсі - ВідступЗверху;
        ГоризКрок = (double)(ДовжинаГоризОсі / Sales.Length + 3);
        ВертКрок = (int)(ДовжинаВертОсі / 10);
        XПочЕпюри = ВідступЛіворуч;//Початкове значення графіку по вісі X
                                   // Послідовно викликаємо наступні процедури:
        Графіка = Graphics.FromImage(Растр);
        МалюємоВісі();
        МалюємоГоризЛінії();
        МалюємоВертЛінії();
        МалюємоЕпюру();
        pictureBox1.Image = Растр;
        // Звільняємо ресурси, використовувані об'єктом класу Graphics:
        Графіка.Dispose(); // - Еквівалент C++: delete Графіка;
    } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

    private void МалюємоВісі()
    {
        Pen Перо = new Pen(Color.Black, 2);
        // Малювання вертикальної осі координат:
        Графіка.DrawLine(Перо, ВідступЛіворуч, YГоризОсі, ВідступЛіворуч,
    ВідступЗверху);
        // Малювання горизонтальної осі координат:
        Графіка.DrawLine(Перо, ВідступЛіворуч, YГоризОсі, Xmax,
    YГоризОсі);
        System.Drawing.Font Шрифт = new System.Drawing.Font("Arial", 8);
        // Підписуємо значення по вісі Y
        for (int i = 1; i <= 10; i++)
        {// Малюємо "вусики" на вертикальній координатної осі:
            int Y = YГоризОсі - i * ВертКрок;
            Графіка.DrawLine(Перо, ВідступЛіворуч - 5, Y,
    ВідступЛіворуч, Y);
            // Підписуємо значення продажів через кожні 100 одиниць:
            Графіка.DrawString((i * 100).ToString(), Шрифт,
    Brushes.Black, 2, Y - 5);
        }
        // Підписуємо значення по вісі X
        for (int i = 0; i <= Months.Length - 1; i++)
        {// Малюємо "вусики" на горизонтальній координатної осі:
            int X = XПочЕпюри + (int)(ГоризКрок * (i + 1));
            Графіка.DrawLine(Перо, X, YГоризОсі + 5, X, YГоризОсі);
            // Підписуємо місяці на горизонтальній осі:
            Графіка.DrawString(Months[i], Шрифт, Brushes.Black,
    ВідступЛіворуч - 10 + (int)(i * ГоризКрок),
    YГоризОсі + 4);
        }
    } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

    private void МалюємоГоризЛінії()
    {
        Pen ТонкеПеро = new Pen(Color.LightGray, 1);
        for (int i = 1; i <= 10; i++)
        {// Малюємо горизонтальні майже "прозорі" лінії:
            int Y = YГоризОсі - ВертКрок * i;
            Графіка.DrawLine(ТонкеПеро, ВідступЛіворуч, Y, Xmax, Y);
        }
        ТонкеПеро.Dispose();
    } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

    private void МалюємоВертЛінії()
    {// Малюємо вертикальні майже "прозорі" лінії
        Pen ТонкеПеро = new Pen(Color.Bisque, 1);
        for (int i = 0; i <= Months.Length - 1; i++)
        {
            int X = XПочЕпюри + (int)(ГоризКрок * i);
```

```
                Графіка.DrawLine(ТонкеПеро, X, ВідступЗверху, X,
            YГоризОсі - 4);
                }
                ТонкеПеро.Dispose();
            } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

            private void МалюємоЕпюру()
            {
                double ВертМасштаб = (double)ДовжинаВертОсі / 1000;
                // Значення ординат на екрані:
                int[] Y = new int[Sales.Length];
                // Значення абсцис на екрані:
                int[] X = new int[Sales.Length];
                for (int i = 0; i <= Sales.Length - 1; i++)
                {// Обчислюємо графічні координати точок:
                    Y[i] = YГоризОсі - (int)(Sales[i] * ВертМасштаб);
                    // Віднімаємо значення продажів, оскільки вісь Y екрану
                    // спрямована вниз
                    X[i] = XПочЕпюри + (int)(ГоризКрок * i);
                }
                // Малюємо перше коло:
                Pen Перо = new Pen(Color.Blue, 3);
                Графіка.DrawEllipse(Перо, X[0] - 2, Y[0] - 2, 4, 4);
                for (int i = 0; i <= Sales.Length - 2; i++)
                {// Цикл по лініях між точками:
                    Графіка.DrawLine(Перо, X[i], Y[i], X[i + 1], Y[i + 1]);
                    // Віднімаємо 2, оскільки діаметр (ширина) точки = 4:
                    Графіка.DrawEllipse(Перо, X[i + 1] - 2, Y[i + 1] - 2, 4, 4);
                }
            } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

        }
    }
```

Лістинг 7.7

```
//Приклад використання об'єктів TabControl, TabPage та відкриття файлів
//за допомогою діалогового вікна OpenFileDialog в об'єктах PictureBox та
//RichTextBox

#pragma once

namespace Windows_Forms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::IO; //Підключення простору імен для методу
Path::GetFileName()

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::TabControl^  tabControl1;
	protected:
	private: System::Windows::Forms::TabPage^  tabPage1;
	private: System::Windows::Forms::TabPage^  tabPage2;
	private: System::Windows::Forms::PictureBox^  pictureBox1;

	private: System::Windows::Forms::Button^  buttonВідкрити;
	private: System::Windows::Forms::RichTextBox^  richTextBox1;


	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
```

```
		void InitializeComponent(void)
		{
			this->tabControl1 = (gcnew System::Windows::Forms::TabControl());
			this->tabPage1 = (gcnew System::Windows::Forms::TabPage());
			this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
			this->tabPage2 = (gcnew System::Windows::Forms::TabPage());
			this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());
			this->buttonВідкрити = (gcnew System::Windows::Forms::Button());
			this->tabControl1->SuspendLayout();
			this->tabPage1->SuspendLayout();
			(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->BeginInit();
			this->tabPage2->SuspendLayout();
			this->SuspendLayout();
			//
			// tabControl1
			//
			this->tabControl1->Controls->Add(this->tabPage1);
			this->tabControl1->Controls->Add(this->tabPage2);
			this->tabControl1->Dock = System::Windows::Forms::DockStyle::Fill;
			this->tabControl1->Location = System::Drawing::Point(0, 0);
			this->tabControl1->Name = L"tabControl1";
			this->tabControl1->SelectedIndex = 0;
			this->tabControl1->Size = System::Drawing::Size(284, 261);
			this->tabControl1->TabIndex = 0;
			//
			// tabPage1
			//
			this->tabPage1->Controls->Add(this->pictureBox1);
			this->tabPage1->Location = System::Drawing::Point(4, 22);
			this->tabPage1->Name = L"tabPage1";
			this->tabPage1->Padding = System::Windows::Forms::Padding(3);
			this->tabPage1->Size = System::Drawing::Size(276, 235);
			this->tabPage1->TabIndex = 0;
			this->tabPage1->Text = L"Графічний файл";
			this->tabPage1->UseVisualStyleBackColor = true;
			//
			// pictureBox1
			//
			this->pictureBox1->Dock = System::Windows::Forms::DockStyle::Fill;
			this->pictureBox1->Location = System::Drawing::Point(3, 3);
			this->pictureBox1->Name = L"pictureBox1";
			this->pictureBox1->Size = System::Drawing::Size(270, 229);
			this->pictureBox1->TabIndex = 0;
			this->pictureBox1->TabStop = false;
			//
			// tabPage2
			//
			this->tabPage2->Controls->Add(this->richTextBox1);
			this->tabPage2->Location = System::Drawing::Point(4, 22);
			this->tabPage2->Name = L"tabPage2";
			this->tabPage2->Padding = System::Windows::Forms::Padding(3);
			this->tabPage2->Size = System::Drawing::Size(276, 235);
			this->tabPage2->TabIndex = 1;
			this->tabPage2->Text = L"Файл *.rtf";
			this->tabPage2->UseVisualStyleBackColor = true;
			//
			// richTextBox1
			//
			this->richTextBox1->Dock = System::Windows::Forms::DockStyle::Fill;
			this->richTextBox1->Location = System::Drawing::Point(3, 3);
			this->richTextBox1->Name = L"richTextBox1";
			this->richTextBox1->Size = System::Drawing::Size(270, 229);
			this->richTextBox1->TabIndex = 0;
			this->richTextBox1->Text = L"";
			//
			// buttonВідкрити
```

```
                    //
                    this->buttonВідкрити->Dock = System::Windows::Forms::DockStyle::Bottom;
                    this->buttonВідкрити->Location = System::Drawing::Point(0, 238);
                    this->buttonВідкрити->Name = L"buttonВідкрити";
                    this->buttonВідкрити->Size = System::Drawing::Size(284, 23);
                    this->buttonВідкрити->TabIndex = 1;
                    this->buttonВідкрити->Text = L"Відкрити файл";
                    this->buttonВідкрити->UseVisualStyleBackColor = true;
                    this->buttonВідкрити->Click += gcnew System::EventHandler(this,
&MyForm::buttonВідкрити_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->buttonВідкрити);
                    this->Controls->Add(this->tabControl1);
                    this->Name = L"MyForm";
                    this->Text = L"Перегляд файлів";
                    this->tabControl1->ResumeLayout(false);
                    this->tabPage1->ResumeLayout(false);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->EndInit();
                    this->tabPage2->ResumeLayout(false);
                    this->ResumeLayout(false);
                }

#pragma endregion
private: System::Void buttonВідкрити_Click(System::Object^  sender, System::EventArgs^  e) {
        if (tabControl1->SelectedIndex == 0) //Перевірка чи активна перша вкладка
        {       //Тоді відкриваємо графічний файл
                //Створюємо об'єкт класу OpenFileDialog
                OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
                //Встановлюємо в діалоговому вікні фільтр для відкриття файлів
                openFileDialog1->Filter = L"Файли зображень | *.bmp;*.jpg;*.gif; *.png";
                if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK
                 && openFileDialog1->FileName->Length > 0)
                //Якщо користувач обрав в діалоговому вікні файл і його ім'я не пусте
                {
                        //Відкриваємо графічний файл в об'єкті pictureBox1
                        pictureBox1->Image = Image::FromFile(openFileDialog1->FileName);
                        //Розміщуємо ім'я відкритого файлу в заголовку вкладки
                        tabPage1->Text = Path::GetFileName(openFileDialog1->FileName);
                }
        }
        else //Активна друга вкладка
        {       //Тоді відкриваємо файл *.rtf
                //Створюємо об'єкт класу OpenFileDialog
                OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
                //Встановлюємо в діалоговому вікні фільтр для відкриття файлів
                openFileDialog1->Filter = L"Текстові файли | *.rtf";
                if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK
                 && openFileDialog1->FileName->Length > 0)
                //Якщо користувач обрав в діалоговому вікні файл і його ім'я не пусте
                {
                        //Відкриваємо файл формату RTF в об'єкті richTextBox1
                        richTextBox1->LoadFile(openFileDialog1->FileName);
                        //Розміщуємо ім'я відкритого файлу в заголовку вкладки
                        tabPage2->Text = Path::GetFileName(openFileDialog1->FileName);
                }
        }
}
};
}
```

Лістинг 7.8

```
//Приклад використання об'єктів TabControl, TabPage та відкриття файлів
//за допомогою діалогового вікна OpenFileDialog в об'єктах PictureBox та
//RichTextBox
// Код модуля Form1.cs

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;  //Підключення простору імен для методу Path.GetFileName()

namespace WindowsFormsAppTabControl
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonВідкрити_Click(object sender, EventArgs e)
        {
            if (tabControl1.SelectedIndex == 0) //Перевірка чи активна перша вкладка
            {   //Тоді відкриваємо графічний файл
                //Створюємо об'єкт класу OpenFileDialog
                OpenFileDialog openFileDialog1 = new OpenFileDialog();
                //Встановлюємо в діалоговому вікні фільтр для відкриття файлів
                openFileDialog1.Filter = "Файли зображень | *.bmp;*.jpg;*.gif; *.png";
                if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK
                    && openFileDialog1.FileName.Length > 0)
                //Якщо користувач обрав в діалоговому вікні файл і його ім'я не пусте
                {
                    //Відкриваємо графічний файл в об'єкті pictureBox1
                    pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);
                    //Розміщуємо ім'я відкритого файлу в заголовку вкладки
                    tabPage1.Text = Path.GetFileName(openFileDialog1.FileName);
                }
            }
            else //Активна друга вкладка
            {   //Тоді відкриваємо файл *.rtf
                //Створюємо об'єкт класу OpenFileDialog
                OpenFileDialog openFileDialog1 = new OpenFileDialog();
                //Встановлюємо в діалоговому вікні фільтр для відкриття файлів
                openFileDialog1.Filter = "Текстові файли | *.rtf";
                if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK
                    && openFileDialog1.FileName.Length > 0)
                //Якщо користувач обрав в діалоговому вікні файл і його ім'я не пусте
                {
                    //Відкриваємо файл формату RTF в об'єкті richTextBox1
                    richTextBox1.LoadFile(openFileDialog1.FileName);
                    //Розміщуємо ім'я відкритого файлу в заголовку вкладки
                    tabPage2.Text = Path.GetFileName(openFileDialog1.FileName);
                }
            }

        }
    }
}
```

Лістинг 7.9

```
// Код модуля Form1.Designer.cs


namespace WindowsFormsAppTabControl
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.buttonВідкрити = new System.Windows.Forms.Button();
            this.tabControl1 = new System.Windows.Forms.TabControl();
            this.tabPage1 = new System.Windows.Forms.TabPage();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.tabPage2 = new System.Windows.Forms.TabPage();
            this.richTextBox1 = new System.Windows.Forms.RichTextBox();
            this.tabControl1.SuspendLayout();
            this.tabPage1.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
            this.tabPage2.SuspendLayout();
            this.SuspendLayout();
            //
            // buttonВідкрити
            //
            this.buttonВідкрити.Dock = System.Windows.Forms.DockStyle.Bottom;
            this.buttonВідкрити.Location = new System.Drawing.Point(0, 282);
            this.buttonВідкрити.Name = "buttonВідкрити";
            this.buttonВідкрити.Size = new System.Drawing.Size(473, 23);
            this.buttonВідкрити.TabIndex = 0;
            this.buttonВідкрити.Text = "Відкрити файл";
            this.buttonВідкрити.UseVisualStyleBackColor = true;
            this.buttonВідкрити.Click += new System.EventHandler(this.buttonВідкрити_Click);
            //
            // tabControl1
            //
            this.tabControl1.Controls.Add(this.tabPage1);
            this.tabControl1.Controls.Add(this.tabPage2);
            this.tabControl1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.tabControl1.Location = new System.Drawing.Point(0, 0);
            this.tabControl1.Name = "tabControl1";
```

```
            this.tabControl1.SelectedIndex = 0;
            this.tabControl1.Size = new System.Drawing.Size(473, 282);
            this.tabControl1.TabIndex = 1;
            //
            // tabPage1
            //
            this.tabPage1.Controls.Add(this.pictureBox1);
            this.tabPage1.Location = new System.Drawing.Point(4, 22);
            this.tabPage1.Name = "tabPage1";
            this.tabPage1.Padding = new System.Windows.Forms.Padding(3);
            this.tabPage1.Size = new System.Drawing.Size(465, 256);
            this.tabPage1.TabIndex = 0;
            this.tabPage1.Text = "Графічний файл";
            this.tabPage1.UseVisualStyleBackColor = true;
            //
            // pictureBox1
            //
            this.pictureBox1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.pictureBox1.Location = new System.Drawing.Point(3, 3);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(459, 250);
            this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;
            this.pictureBox1.TabIndex = 0;
            this.pictureBox1.TabStop = false;
            //
            // tabPage2
            //
            this.tabPage2.Controls.Add(this.richTextBox1);
            this.tabPage2.Location = new System.Drawing.Point(4, 22);
            this.tabPage2.Name = "tabPage2";
            this.tabPage2.Padding = new System.Windows.Forms.Padding(3);
            this.tabPage2.Size = new System.Drawing.Size(465, 256);
            this.tabPage2.TabIndex = 1;
            this.tabPage2.Text = "Файл RTF";
            this.tabPage2.UseVisualStyleBackColor = true;
            //
            // richTextBox1
            //
            this.richTextBox1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.richTextBox1.Location = new System.Drawing.Point(3, 3);
            this.richTextBox1.Name = "richTextBox1";
            this.richTextBox1.Size = new System.Drawing.Size(459, 250);
            this.richTextBox1.TabIndex = 0;
            this.richTextBox1.Text = "";
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(473, 305);
            this.Controls.Add(this.tabControl1);
            this.Controls.Add(this.buttonВідкрити);
            this.Name = "Form1";
            this.Text = "Перегляд файлів";
            this.tabControl1.ResumeLayout(false);
            this.tabPage1.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
            this.tabPage2.ResumeLayout(false);
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.Button buttonВідкрити;
        private System.Windows.Forms.TabControl tabControl1;
        private System.Windows.Forms.TabPage tabPage1;
```

```
            private System.Windows.Forms.PictureBox pictureBox1;
            private System.Windows.Forms.TabPage tabPage2;
            private System.Windows.Forms.RichTextBox richTextBox1;
        }
    }
```

ПРАКТИЧНА РОБОТА №8

8 Робота з базами даних у C++/CLI та C#

8.1 Створення бази даних MS Access

Завдання 1.

*Необхідно створити програму, яка під час своєї роботи створює базу даних Access, тобто файл типу *.mdb. Ця база даних буде порожньою, тобто вона не буде містити жодної таблиці. Наповнювати базу даних таблицями можна згодом як з програмного коду, так і використовуючи MS Access.*

Запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо додаток шаблону Windows Forms. Дамо ім'я новому проекту БазаДаних. Щоб додати до нашого проекту DLL-бібліотеки ADO потрібно виконати наступні дії.

1) У вікні Оглядача рішень (Solution Explorer) клацаємо мишею по імені проекту БазаДаних для його виділення.
2) В пункті меню Проект (Project) виберемо команду Додати посилання… (Add Reference…) (рис. 8.1).

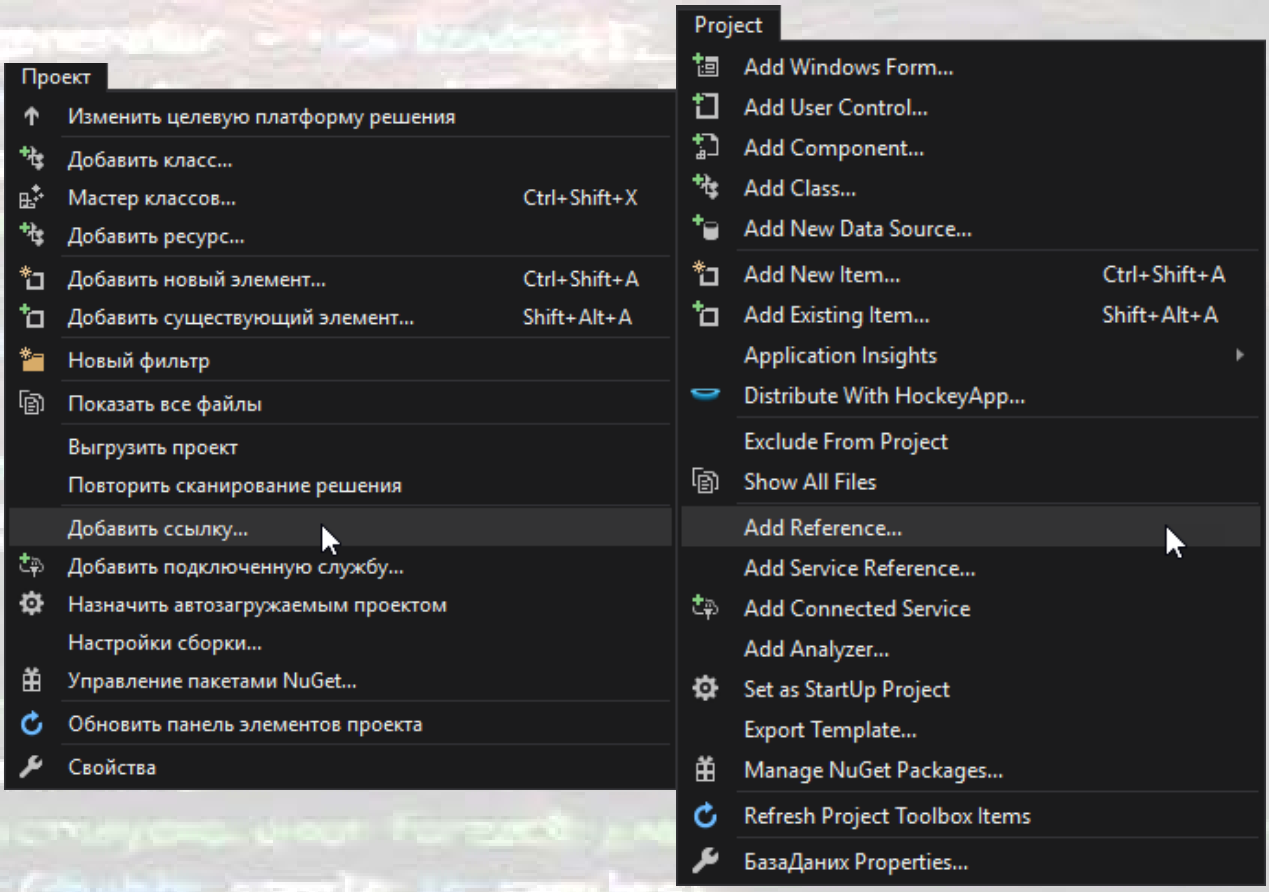


Рисунок 8.1 – Команда Додати посилання… меню Проект

3) На вкладці COM двічі клацнемо по посиланню Microsoft ADO Ext. 6.0 for DDL and Security (або більш ранню версію Microsoft ADO Ext.

2.8 for DDL and Security), додавши тим самим цю бібліотеку в поточний проект (рис. 8.2).

4) Переконатися в тому, що тепер існує посилання на цю бібліотеку, можна у вікні Властивості (Properties). Тут, клацнувши на вузлі Посилання (References), побачимо гілку ADOX (рис. 8.3). Тепер ми можемо посилатися на це ім'я в програмному коді.

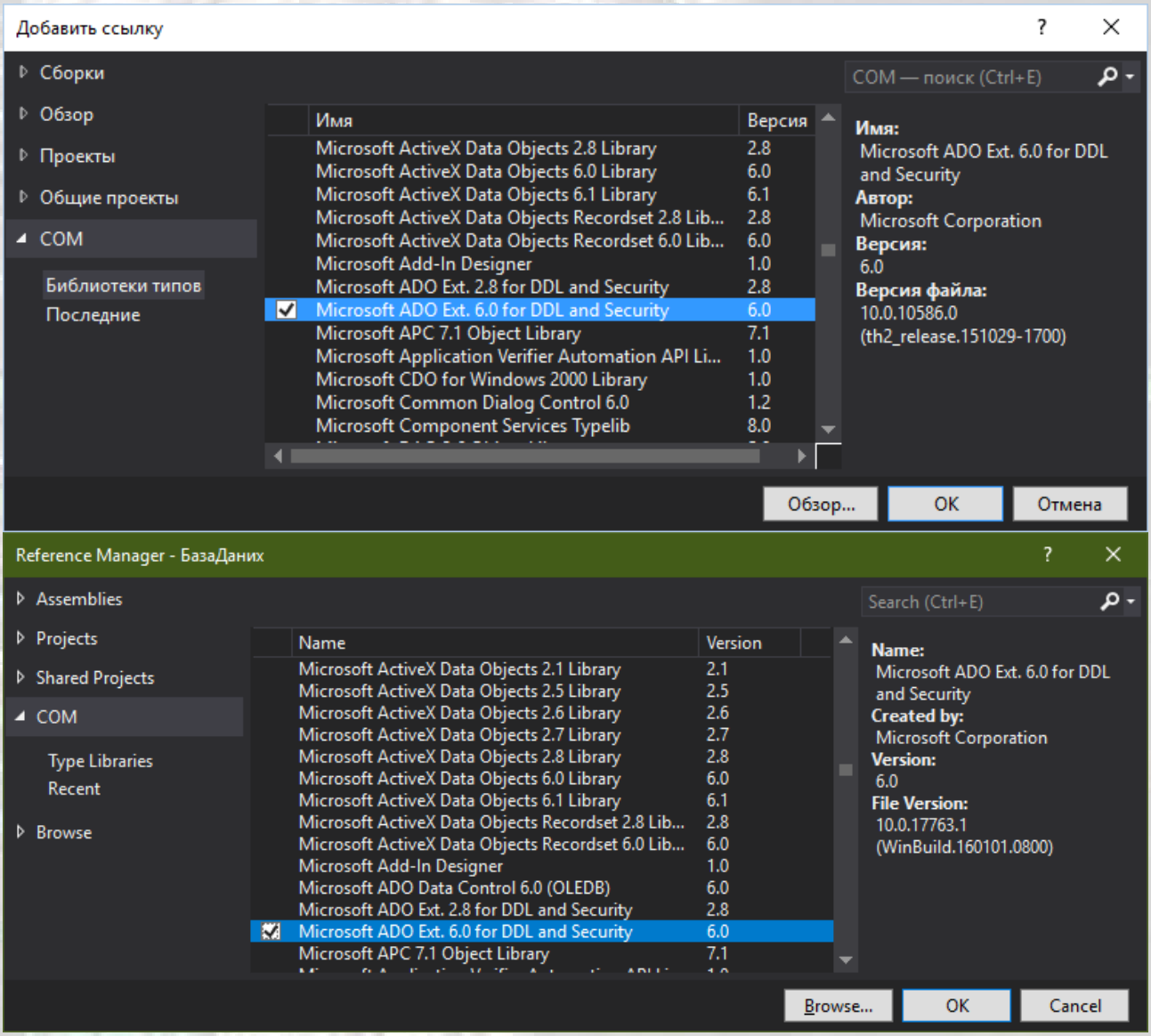


Рисунок 8.2 –Додавання бібліотеки Microsoft ADO Ext. 6.0 for DDL and Security

Далі переходимо у вікно конструктора форми і двічі клацнувши по формі, потрапляємо до обробника події Form1_Load, де вводимо програмний код, наведений у лістингу нижче.

Приклад коду C++/CLI

```
#pragma endregion
private: System::Void MyForm_Load(System::Object^  sender,
System::EventArgs^  e) {
    ADOX::Catalog ^ Каталог = gcnew ADOX::Catalog();
    try
```

```
        {
            Каталог->Create("Provider=Microsoft.Jet." +
                "OLEDB.4.0;Data Source=d:\\Нова_БД.mdb");
            MessageBox::Show("База даних d:\\Нова_БД.mdb успішно
                створена", "Створення нової БД MS Access",
                MessageBoxButtons::OK,
                MessageBoxIcon::Information);
        }
        catch (System::Runtime::InteropServices::COMException ^ Ситуація)
        {
            MessageBox::Show(Ситуація->Message, "Створення нової БД MS
                Access", MessageBoxButtons::OK,
                MessageBoxIcon::Warning);
        }
        finally
        { Каталог = nullptr; }
}
```

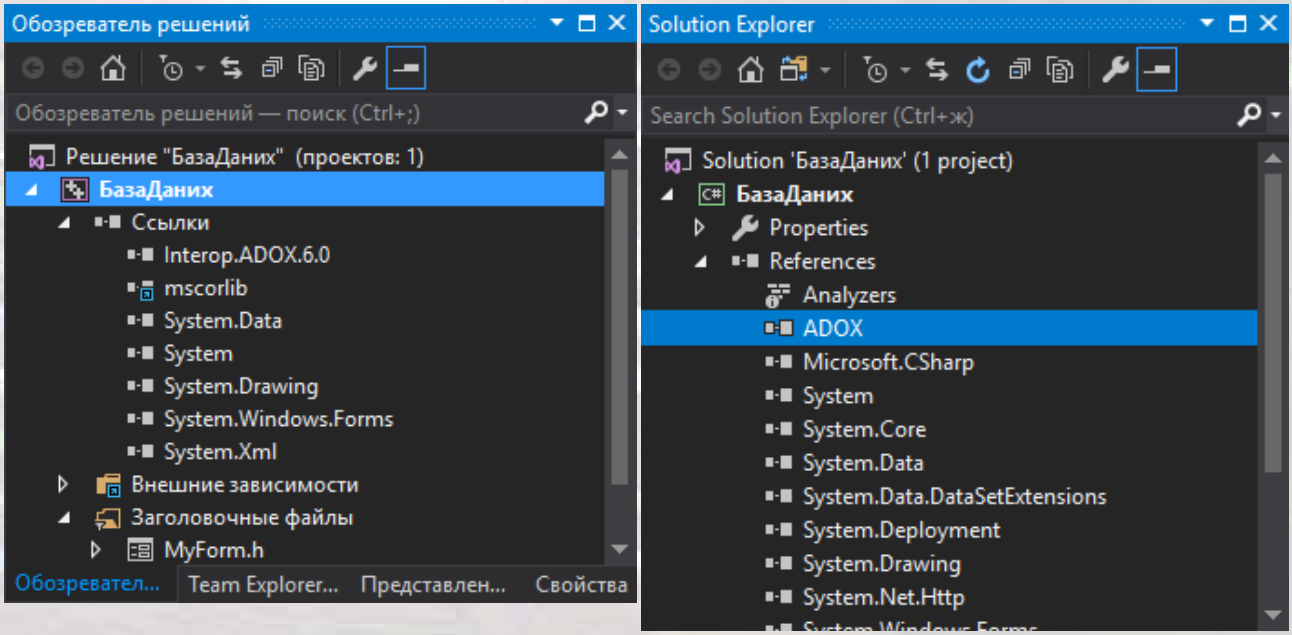


Рисунок 8.3 – Гілка ADOX у вузлі Посилання вікна Оглядача рішень

Приклад коду C#

```
private void Form1_Load(object sender, EventArgs e)
{
        ADOX.Catalog Каталог = new ADOX.Catalog();
        try
        {
            Каталог.Create("Provider=Microsoft.Jet." +
                "OLEDB.4.0;Data Source=d:\\Нова_БД.mdb");
            MessageBox.Show("База даних d:\\Нова_БД.mdb успішно створена",
                "Створення нової БД MS Access", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
        }
        catch (System.Runtime.InteropServices.COMException Ситуація)
        {
            MessageBox.Show(Ситуація.Message, "Створення нової БД MS
```

```
                Access", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
        finally
        { Каталог = null; }
}
```

Програма працює наступним чином: створюємо екземпляр класу ADOX: `catalog`, одна з його функцій `Create` здатна створювати базу даних, якщо на її вхід подати рядок підключення. Зауважимо, що в рядок підключення входить також і повний шлях до створюваної БД. Функція `Create` укладена в блоки `try ... catch`, які обробляють виняткові ситуації. Після запуску цієї програми отримаємо повідомлення про створення бази даних (рис. 8.4).

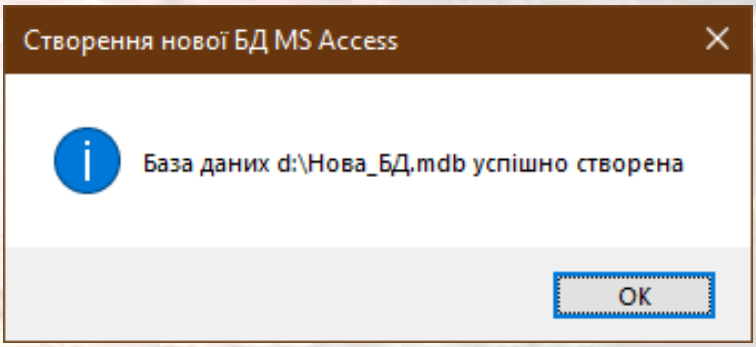


Рисунок 8.4 – Повідомлення про успішне створення нової БД

Якщо запустити наш додаток ще раз, то ми отримаємо повідомлення про те, що така база даних вже існує (рис. 8.5), оскільки БД `Нова_БД.mdb` щойно створена.

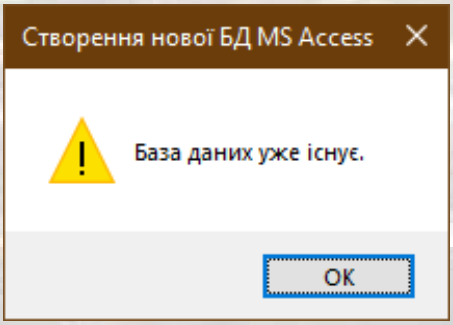


Рисунок 8.5 – Повідомлення про існування БД

Текст даного повідомлення генерувався оброблювачем виняткової ситуації `Ситуація.Message`. Ми додали власний заголовок вікна `"Створення нової БД MS Access"`, вказали, що в ньому має бути лише кнопка OK, та обрали тип вікна повідомлення `Warning`.

Приклад коду C++/CLI

```
MessageBox::Show(Ситуація->Message, "Створення нової БД MS Access",
    MessageBoxButtons::OK, MessageBoxIcon::Warning);
```

Приклад коду C#

```
MessageBox.Show(Ситуація.Message, "Створення нової БД MS Access",
      MessageBoxButtons.OK, MessageBoxIcon.Warning);
```

8.2 Запис структури таблиці в порожню базу даних MS Access. Програмне підключення до БД

Завдання 2.

Необхідно створити програму, яка записує структуру таблиці, тобто «шапку» таблиці, в існуючу БД. У цій БД може не бути жодної таблиці, тобто БД може бути порожньою. Або в БД можуть вже бути таблиці, але назва нової таблиці повинна бути унікальною.

Створимо базу даних Нова_БД.mdb в кореневому каталозі логічного диска D:, використовуючи MS Access або програмним шляхом, як це було показано в попередньому підрозділі. Ніякі таблиці в базі даних створювати не будемо, тобто наша БД буде порожньою. Тепер запустимо Visual Studio і в вікні Створення проекта (New Project) виберемо додаток шаблону Windows Forms. Потім у обробник події Form1_Load запишемо програмний код, представлений в лістингу нижче.

Приклад коду C++/CLI

```
#pragma endregion
private: System::Void MyForm_Load(System::Object^  sender,
System::EventArgs^  e) {
      // ЗАПИС СТРУКТУРИ ТАБЛИЦІ у порожню БД:
      // Створення примірника об'єкта OleDbConnection
      // із зазначенням рядка підключення:
      OleDbConnection ^ Підключення = gcnew OleDbConnection(
            "Provider = Microsoft.Jet.OLEDB.4.0; Data Source =
            D:\\Нова_БД.mdb");
      // Відкриття підключення:
      Підключення->Open();
      // Створення примірника об'єкта класу Command
      // Із завданням SQL-запиту:
      OleDbCommand ^ Команда = gcnew OleDbCommand(
            "CREATE TABLE [БД телефонів]" +
            " ([Номер п/п] counter, [ПІБ] char(20), " +
            " [Номер телефону] char(20))", Підключення);
      try // Виконання команди SQL:
      {
            Команда->ExecuteNonQuery();
            MessageBox::Show("Структура таблиці 'БД телефонів' записана
                  в порожню БД", "Створення структури таблиці MS Access",
                  MessageBoxButtons::OK, MessageBoxIcon::Information);
      }
      catch (Exception ^ Ситуація)
      {
```

```
            MessageBox::Show(Ситуація->Message, "Створення структури
                  таблиці MS Access", MessageBoxButtons::OK,
                  MessageBoxIcon::Warning);
      }
      Підключення->Close();
}
```

Приклад коду C#

```
public partial class Form1: Form
{
      public Form1()
      {
            InitializeComponent();
      }

      private void Form1_Load(object sender, EventArgs e)
      {
            // ЗАПИС СТРУКТУРИ ТАБЛИЦІ у порожню БД:
            // Створення примірника об'єкта OleDbConnection із зазначенням
            // рядка підключення:
            OleDbConnection Підключення = new OleDbConnection(
                  "Provider = Microsoft.Jet.OLEDB.4.0; Data Source =
                  D:\\Нова_БД.mdb");
            // Відкриття підключення:
            Підключення.Open();
            // Створення примірника об'єкта класу Command
            // Із завданням SQL-запиту:
            OleDbCommand Команда = new OleDbCommand("CREATE TABLE
                  [БД телефонів]" +
                  " ([Номер п/п] counter, [ПІБ] char(20), " +
                  " [Номер телефону] char(20))", Підключення);
            try // Виконання команди SQL:
            {
                  Команда.ExecuteNonQuery();
                  MessageBox.Show("Структура таблиці 'БД телефонів'
                        записана в порожню БД", "Створення структури
                        таблиці MS Access",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception Ситуація)
            {
                  MessageBox.Show(Ситуація.Message, "Створення структури
                        таблиці MS Access", MessageBoxButtons.OK,
                        MessageBoxIcon.Warning);
            }
            Підключення.Close();
      }
}
```

В програмний код ми також додали наступну директиву для більш короткого звернення до класів обробки даних:

Приклад коду C++/CLI

```
using namespace System::Data::OleDb;
```

Приклад коду C#

```
using System.Data.OleDb;
```

Потрібно звернути увагу, що оператор

Приклад коду C++/CLI

```
OleDbCommand ^ Команда = gcnew OleDbCommand(
                    "CREATE TABLE [БД телефонів]" +
                    " ([Номер п/п] counter, [ПІБ] char(20), " +
                    " [Номер телефону] char(20))", Підключення);
```

Приклад коду C#

```
OleDbCommand Команда = new OleDbCommand("CREATE TABLE [БД телефонів]" +
                        " ([Номер п/п] counter, [ПІБ] char(20), " +
                        " [Номер телефону] char(20))", Підключення);
```

формує SQL-запит у середині подвійних лапок у вигляді рядка тексту і в наслідок цього, на відміну від звичайних операторів C++, даний рядок не можна довільно розбивати на частини. Розбити оператор на кілька рядків можна лише за допомогою оператора *конкатенації +*, узявши кожну частину рядка SQL-запиту у подвійні лапки, як і було зроблено у операторі вище. У вікні редактору коду цей рядок можна не розбивати і записати разом. Тоді SQL-запит у подвійних лапках матиме вигляд

```
"CREATE TABLE [БД телефонів] ([Номер п/п] counter, [ПІБ] char(20), [Номер телефону] char(20))"
```

що набагато зрозуміліше, ніж у лістингу програмного коду.

Як видно з тексту програми, спочатку ми створюємо екземпляр класу `OleDbConnection` із зазначенням рядка підключення, це дозволить нам керувати цим рядком програмно. Далі створюємо екземпляр класу `OleDbCommand` із завданням SQL-запиту. У цьому запиті створюємо (`CREATE`) нову таблицю з ім'ям `БД телефонів` з трьома полями: `Номер п/п` типу лічильник (`counter`), `ПІБ` і `Номер телефону`. Ім'я таблиці і імена полів укладені в квадратні дужки, оскільки вони містять пробіли.

Щоб виконати цю SQL-команду, викликаємо метод `ExecuteNonQuery()`, який укладемо в блоки `try ... catch` для обробки виняткових ситуацій. Якщо SQL- запит успішно виконався, то отримуємо повідомлення: «`Структура таблиці 'БД телефонів' записана в порожню БД`» (рис. 8.6).

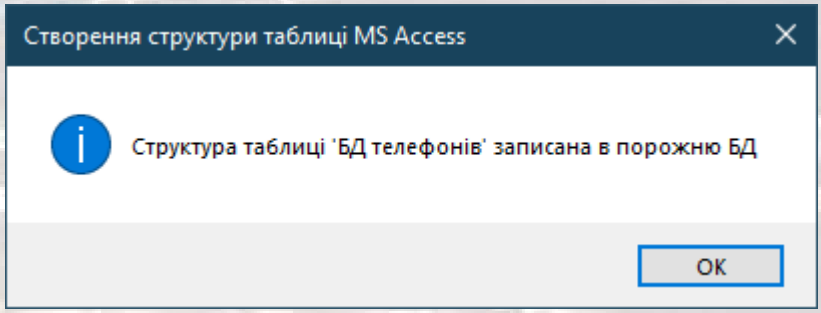


Рисунок 8.6 – Повідомлення про успішне створення структури таблиці БД

А якщо, наприклад, таблиця з таким ім'ям вже є в базі даних, то управління передається блоку `catch` (перехоплення виняткової ситуації), і ми отримуємо повідомлення про те, що така таблиця бази даних вже існує (рис. 8.7).

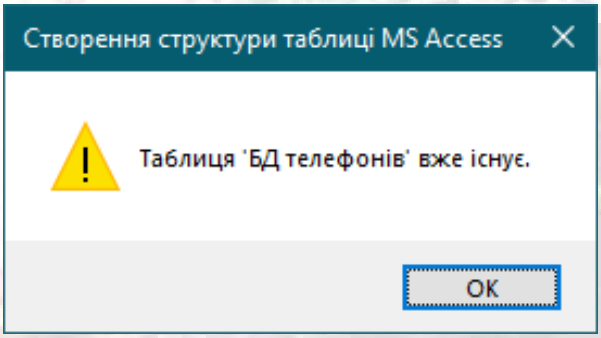


Рисунок 8.7 – Повідомлення про існування таблиці БД

Таким чином, в даній програмі спочатку організовано за допомогою об'єкту `Підключення` класу `OleDbConnection` підключення до БД через рядок підключення і відкриття підключення `Open()`. Потім здійснюється завдання SQL-запиту в об'єкті `Команда` класу `OleDbCommand` і виконання запиту функцією `ExecuteNonQuery()`. Якщо зв'язування даних організувати програмно, то ми отримаємо велику гнучкість для тих випадків, коли, наприклад, на стадії розробки невідомо заздалегідь, де (на якому диску, в якій папці) буде перебувати БД.

8.3 Зчитування даних з БД в сітку даних DataGridView з використанням об'єктів класів Command, Adapter та DataSet

Завдання 3.

Необхідно створити програму, яка зчитує таблиці за допомогою об'єкта Adapter з бази даних шляхом вибору потрібних даних і передачі їх об'єкту DataSet. Дуже зручно прочитати таблицю, записану в DataSet, використовуючи елемент управління DataGridView (сітка даних, тобто таблиця даних), вказавши в якості джерела даних для сітки DataGridView об'єкт класу DataSet.

Створимо новий додаток з шаблону `Windows Forms`. З Панелі елементів (`Toolbox`) перетягнемо до форми елемент управління `DataGridView` та заповнимо ним всю форму за допомогою властивості

Dock.Fill. Після цього запишемо наступний програмний код до обробника події Form1_Load форми.

Приклад коду C++/CLI

```
#pragma endregion
private: System::Void MyForm_Load(System::Object^  sender,
System::EventArgs^  e) {
      this->Text = "Читання таблиці з БД:";
      auto Підключення = gcnew OleDb::OleDbConnection(
            "Data Source = D:\\Нова_БД.mdb; User ID =" +
            " Admin; Provider = Microsoft.Jet.OLEDB.4.0;");
      Підключення->Open();
      auto Команда = gcnew OleDb::OleDbCommand(
            "Select * From [БД телефонів]", Підключення);
      // Вибираємо з таблиці тільки ті записи, поле ПІБ яких
      // починається на букву "М":
      // Auto Команда =
      // gcnew OleDb::OleDbCommand("SELECT * FROM " +
      // "[БД телефонів] WHERE (ПІБ LIKE 'м%')", Підключення);
      // Створюємо об'єкт класу Adapter і виконуємо SQL-запит
      auto Адаптер = gcnew OleDb::OleDbDataAdapter(Команда);
      // Створюємо об'єкт класу DataSet
      auto НаборДаних = gcnew DataSet();
      // Заповнюємо DataSet результатом SQL-запиту
      Адаптер->Fill(НаборДаних, "БД телефонів");
      // Вміст DataSet у вигляді рядка XML для налагодження:
      auto РядокXML = НаборДаних->GetXml();
      // Вказуємо джерело даних для сітки даних:
      dataGridView1->DataSource = НаборДаних;
      // Вказуємо ім'я таблиці в наборі даних:
      dataGridView1->DataMember = "БД телефонів";
      Підключення->Close();
}
```

Приклад коду C#

```
private void Form1_Load(object sender, EventArgs e)
{
      this.Text = "Читання таблиці з БД:";
      var Підключення = new System.Data.OleDb.OleDbConnection(
            "Data Source = D:\\Нова_БД.mdb; User ID =" +
            " Admin; Provider = Microsoft.Jet.OLEDB.4.0;");
      Підключення.Open();
      var Команда = new System.Data.OleDb.OleDbCommand(
            "Select * From [БД телефонів]", Підключення);
      // Вибираємо з таблиці тільки ті записи, поле ПІБ яких
      // починається на букву "М":
      // var Команда =
      // new System.Data.OleDb.OleDbCommand("SELECT * FROM " +
      // "[БД телефонів] WHERE (ПІБ LIKE 'м%')", Підключення);
      // Створюємо об'єкт класу Adapter і виконуємо SQL-запит
```

```
        var Адаптер = new System.Data.OleDb.OleDbDataAdapter(Команда);
        // Створюємо об'єкт класу DataSet
        var НабірДаних = new DataSet();
        // Заповнюємо DataSet результатом SQL-запиту
        Адаптер.Fill(НабірДаних, "БД телефонів");
        // Вміст DataSet у вигляді рядка XML для налагодження:
        var РядокXML = НабірДаних.GetXml();
        // Вказуємо джерело даних для сітки даних:
        dataGridView1.DataSource = НабірДаних;
        // Вказуємо ім'я таблиці в наборі даних:
        dataGridView1.DataMember = "БД телефонів";
        Підключення.Close();
}
```

Саму базу даних відкриваємо та наповнюємо за допомогою MS Access для того, щоб надалі можна було скористатися запитом на вибірку даних (рис. 8.8).

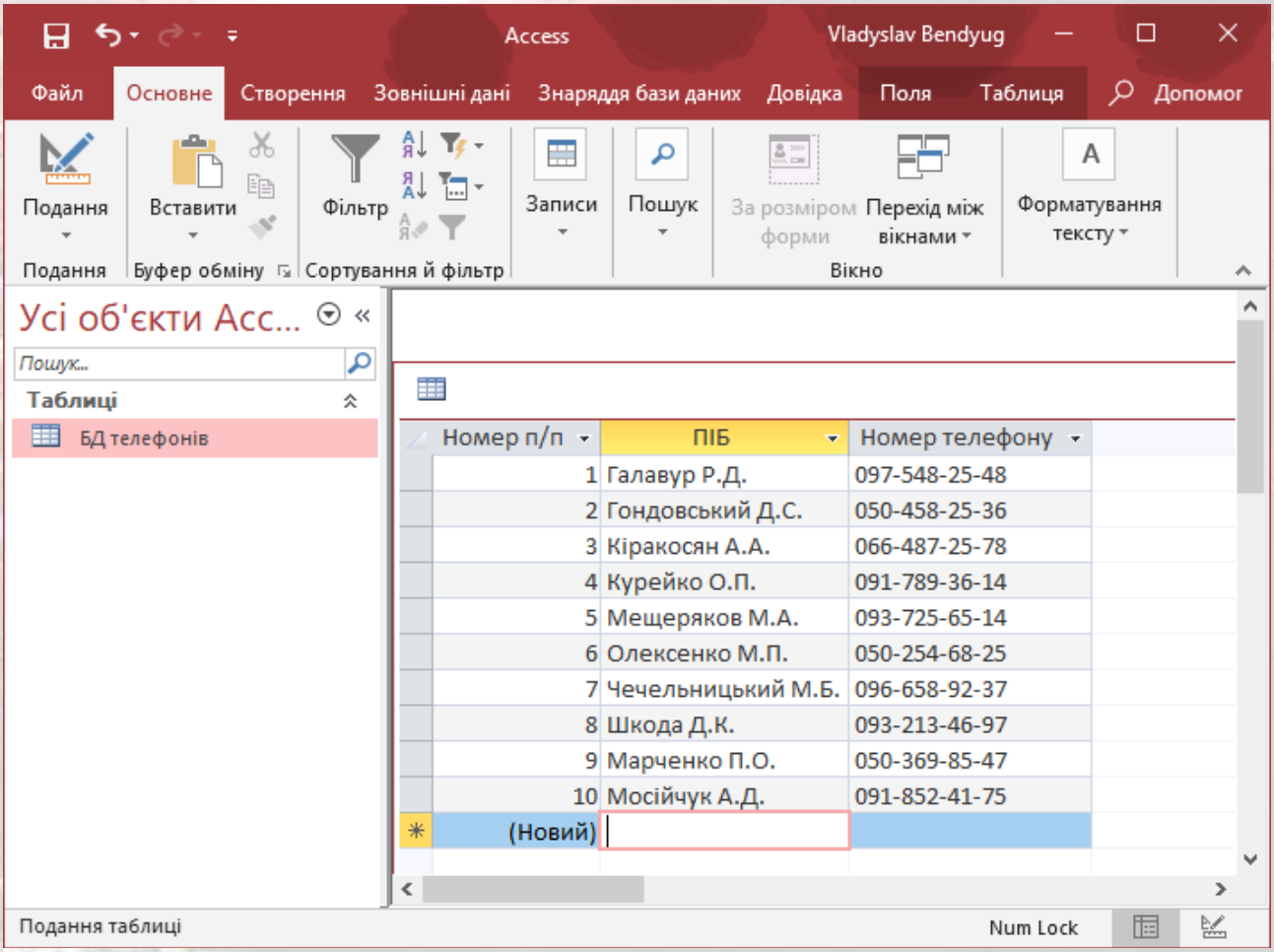
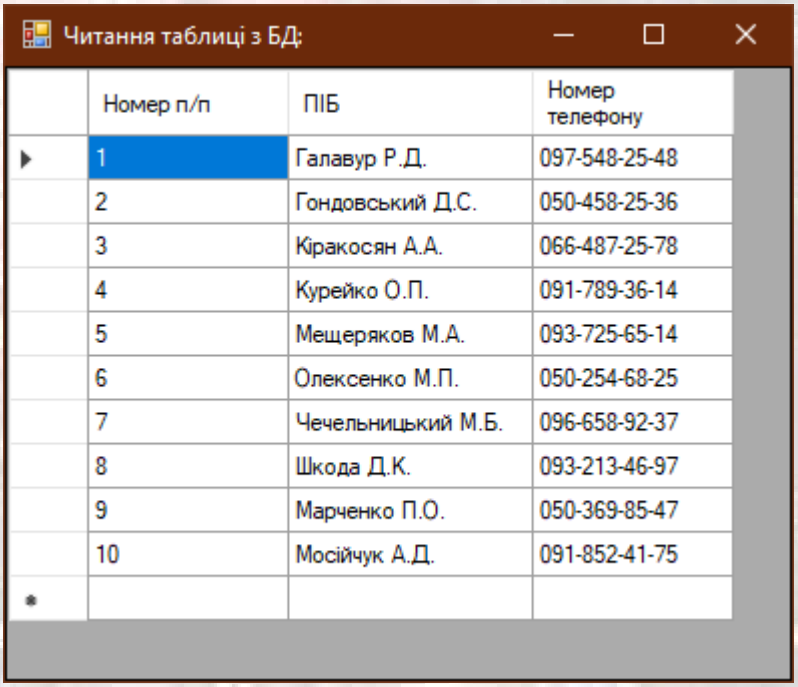


Рисунок 8.8 – Заповнення таблиці БД телефонів в MS Access

Як видно з тексту програми, спочатку ми створили об'єкт класу OleDbConnection, передаючи рядок підключення. Потім, створюючи об'єкт класу OleDbCommand, задаємо SQL-команду вибору всіх записів з таблиці БД телефонів. Тут ми можемо поставити будь-яку SQL-команду. У коментарі наведено приклад такої команди, яка містить SELECT і LIKE, в якій пропонується

вибрати з таблиці БД телефонів тільки записи, в яких поле ПІБ починається на «м». Оператор LIKE використовується для *пошуку за шаблоном* (pattern matching) разом з *символами універсальної підстановки* (метасимвол) «зірочка» (*) і «знак питання» (?). Рядок шаблону укладений в апострофи. Зауважимо також, що більшість баз даних використовує символ % замість значка * в LIKE-виразах.

Далі при створенні об'єкта класу OleDbDataAdapter виконуємо SQL-команду і при виконанні методу Fill заповнюємо об'єкт класу DataSet таблицею, отриманою в результаті SQL-запиту. Потім вказуємо як джерело даних для сітки даних dataGridView1 об'єкт класу DataSet. Цього виявляється достатнім для виведення на екран результатів SQL-запиту рис. 8.9.



	Номер п/п	ПІБ	Номер телефону
▶	1	Галавур Р.Д.	097-548-25-48
	2	Гондовський Д.С.	050-458-25-36
	3	Кіракосян А.А.	066-487-25-78
	4	Курейко О.П.	091-789-36-14
	5	Мещеряков М.А.	093-725-65-14
	6	Олексенко М.П.	050-254-68-25
	7	Чечельницький М.Б.	096-658-92-37
	8	Шкода Д.К.	093-213-46-97
	9	Марченко П.О.	050-369-85-47
	10	Мосійчук А.Д.	091-852-41-75
*			

Рисунок 8.9 – Приклад роботи програми з відображення таблиці даних

Так само як і при використанні об'єкта класу DataReader в попередньому прикладі, в отриманій таблиці ми можемо сортувати записи за кожною з колонок. Можемо редагувати (змінювати) вміст комірок, але в базу даних ці зміни не потраплять (збереження не відбудеться).

Якщо застосувати запит, який був наведений як коментарі у попередньому прикладі програмного коду і закоментувати запит на вибірку всіх записів з таблиці, то у вікні програми виведуться лише записи з прізвищами людей, що починаються з літери «м». Результат роботи цього запиту відображений на рис. 8.10, а фрагмент коду з запитом наведений нижче.

Приклад коду C#

```
/* Запит на вибірку всіх записів з таблиці
var Команда = new System.Data.OleDb.OleDbCommand(
"Select * From [БД телефонів]", Підключення); */
// Вибираємо з таблиці тільки ті записи, поле ПІБ яких
```

```
// починається на літеру "М":
var Команда = new System.Data.OleDb.OleDbCommand("SELECT * FROM " +
      "[БД телефонів] WHERE (ПІБ LIKE 'м%')", Підключення);
```

Звертаємо також увагу, що у мові C# замість оператора auto, який використовується у C++ для автоматичного визначення компілятором типу змінної за ініціалізуючим виразом, використовується оператор var. *Оператор var* використовується в C# для непрямого визначення типу змінної за її ініціалізуючим виразом.

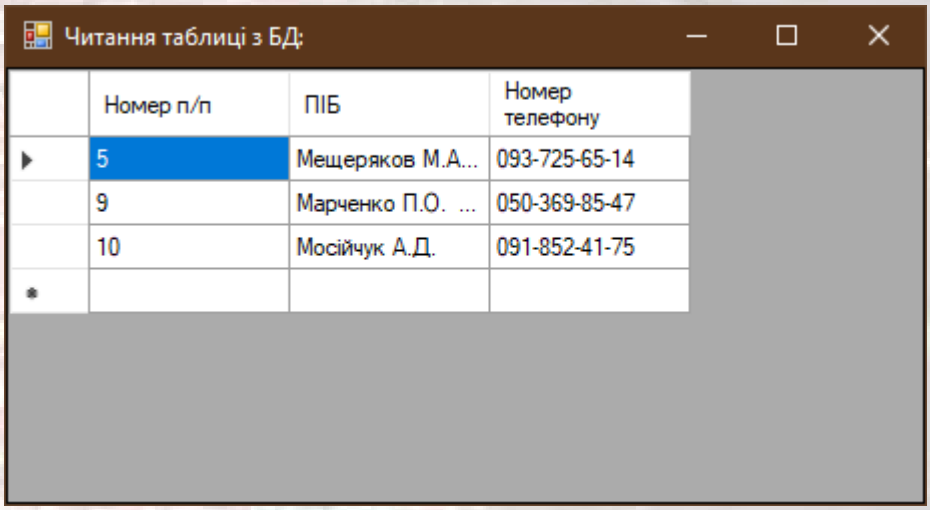


Рисунок 8.10 – Приклад роботи запиту з відображення частини таблиці даних

Зауважимо, що тут за допомогою візуального проектування виконано тільки перетягування в форму сітки даних DataGridView, інше зроблено програмно, що забезпечує більшу гнучкість програми.

8.4 Оновлення записів в таблиці бази даних MS Access

Завдання 4.

Необхідно створити програму, яка дозволяє редагувати, оновлювати та видаляти записи в таблиці бази даних.

Наведена в даному прикладі програма має форму, сітку даних DataGridView, в яку з бази даних зчитується таблиця при натисканні кнопки Читати з БД. Користувач має можливість *редагувати* дані в цій таблиці, після чого, при натисканні кнопки Зберегти в БД, дані в базі даних будуть *модифіковані*, тобто замінені новими.

Для цієї задачи створимо новий додаток з шаблону Windows Forms. З Панелі елементів (Toolbox) додамо до форми елемент Panel, в який помістимо два елементи кнопок Button. Також перетягнемо до форми ще один елемент Panel та помістимо в нього елемент управління DataGridView. Після цього запишемо наступний програмний код.

Приклад коду C++/CLI

```
#pragma endregion
      // Програма оновлює записи (Update) в таблиці бази даних MS Access
      // ~ ~ ~ ~ ~ ~ ~ ~
      // Оголошуємо ці змінні поза всіма процедур, щоб
      // Вони були видні з будь-якої з процедур:
      DataSet ^ НабірДаних;
      OleDb::OleDbDataAdapter ^ Адаптер;
      OleDb::OleDbConnection ^ Підключення;
      OleDb::OleDbCommand ^ Команда;

private: System::Void MyForm_Load(System::Object^  sender,
System::EventArgs^  e) {
      this->panel1->Height = this->button1->Height*2;
      this->button2->Dock = System::Windows::Forms::DockStyle::Bottom;
      this->button1->Dock = System::Windows::Forms::DockStyle::Bottom;
      this->panel1->Dock = System::Windows::Forms::DockStyle::Bottom;
      this->panel2->Dock = System::Windows::Forms::DockStyle::Fill;
      this->dataGridView1->Dock =
            System::Windows::Forms::DockStyle::Fill;
      this->Text = "Редагування БД";
      НабірДаних = gcnew DataSet();
      Підключення = gcnew OleDb::
            OleDbConnection("Data Source=D:\\Нова_БД.mdb; User ID=" +
                  "Admin; Provider=Microsoft.Jet.OLEDB.4.0;");
      Команда = gcnew OleDb::OleDbCommand();
      button1->Text = "Читати з БД"; button1->TabIndex = 0;
      button2->Text = "Зберегти в БД";
      }

private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {
      // Читати з БД:
      if (Підключення->State ==
            ConnectionState::Closed) Підключення->Open();
      Адаптер = gcnew OleDb::OleDbDataAdapter(
            "Select * From [БД телефонів]", Підключення);
      // Заповнюємо DataSet результатом SQL-запиту
      Адаптер->Fill(НабірДаних, "БД телефонів");
      // Вміст DataSet у вигляді рядка XML для відладки:
      String ^ РядокXML = НабірДаних->GetXml();
      // Вказати джерело даних для сітки даних:
      dataGridView1->DataSource = НабірДаних;
      // Вказуємо ім'я таблиці в наборі даних:
      dataGridView1->DataMember = "БД телефонів";
      Підключення->Close();
      }

private: System::Void button2_Click(System::Object^  sender,
System::EventArgs^  e) {
      // Зберегти в базі даних
      Команда->CommandText =
```

```
            "UPDATE [БД телефонів] SET [Номер телефону] =?," +
            " ПІБ =? WHERE ([Номер п/п] =?)";
        // Ім'я, тип і довжина параметра
        Команда->Parameters->Add("Номер телефону",
            OleDb::OleDbType::VarWChar, 50, "Номер телефону");
        Команда->Parameters->Add(
            "ПІБ", OleDb::OleDbType::VarWChar, 50, "ПІБ");
        Команда->Parameters->Add
            (gcnew OleDb::OleDbParameter("Original_Номер_п_п",
                OleDb::OleDbType::Integer, 0,
                System::Data::ParameterDirection::
                Input, false, (Byte)0, (Byte)0, "Номер п/п",
                System::Data::DataRowVersion::Original, nullptr));
        Адаптер->UpdateCommand = Команда;
        Команда->Connection = Підключення;
        try
        {// Update повертає кількість змінених рядків
            int kol = Адаптер->Update(НабірДаних, "БД телефонів");
            MessageBox::Show("Оновлено " + kol + " записів",
            "Оновлення таблиці", MessageBoxButtons::OK,
            MessageBoxIcon::Information);
        }
        catch (Exception ^ Ситуація)
        {
            MessageBox::Show(Ситуація->Message, "Оновлення таблиці",
                MessageBoxButtons::OK, MessageBoxIcon::Error);
        }
}
```

Як видно з коду, ми маємо три процедури обробки подій: завантаження форми MyForm_Load, клацання на кнопці Читати з БД button1_Click і клацання на кнопці Зберегти в БД button2_Click. Щоб об'єкти класів DataSet, DataAdapter, Connection і Command було видно в цих трьох процедурах, оголошуємо ці об'єкти зовнішніми всередині класу MyForm.

При спробі запустити дану програму у операційній системі MS Windows 64-bit може виникнути помилка з повідомленням *Microsoft.Jet.OLEDB.4.0' provider is not registered on the local machine*. Це пов'язане з несумісністю драйверів Microsoft для 32-bit та для 64-bit. У цьому разі потрібно завантажити універсальний драйвер Microsoft Access Database Engine 2010 Redistributable. Після переходу за посиланням з'явится вікно для завантаження потрібного драйверу (рис. 8.11)

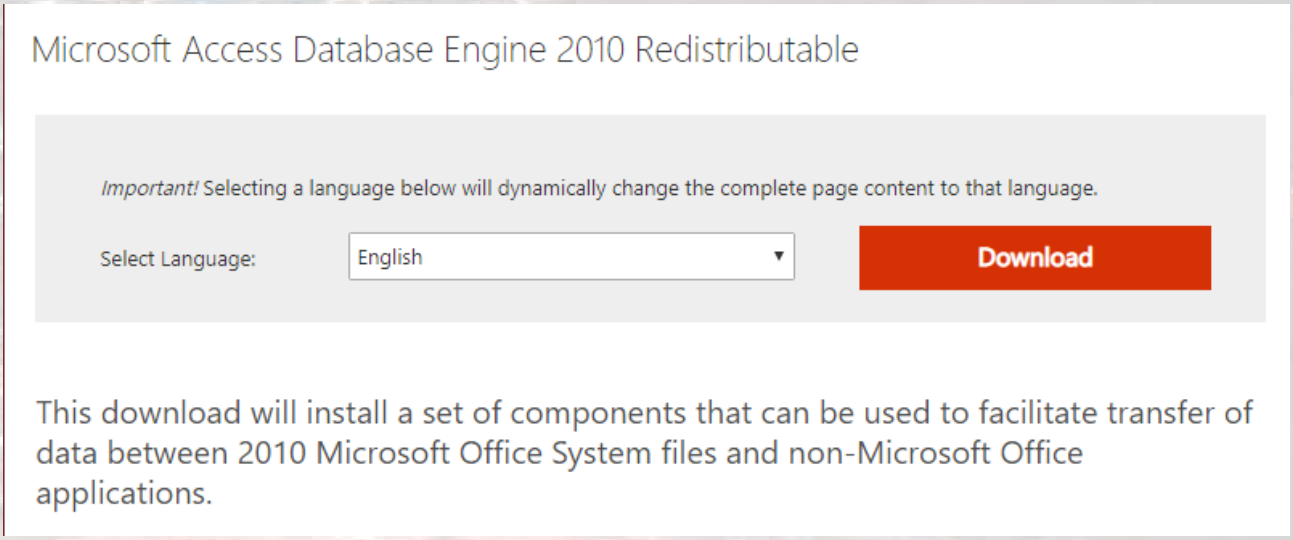


Рисунок 8.11 – Вікно для завантаження драйвера

Після натискання кнопки Download переходимо до вікна вибору потрібних компонентів драйверу (рис. 8.12). Відмічаємо у даному вікні всі компоненти та наатискаємо кнопку Next. Після цього вас запитають про місце зберігання установочного файлу обраного вами драйверу. По завершенні завантаження запускаємо отриманий файл установки драйверу на виконання.

У разі вдалої установки драйверу на вашу систему MS Windows, по завершенні процесу установки з'явиться діалогове вікно з повідомленням, як зображено на рис. 8.13.

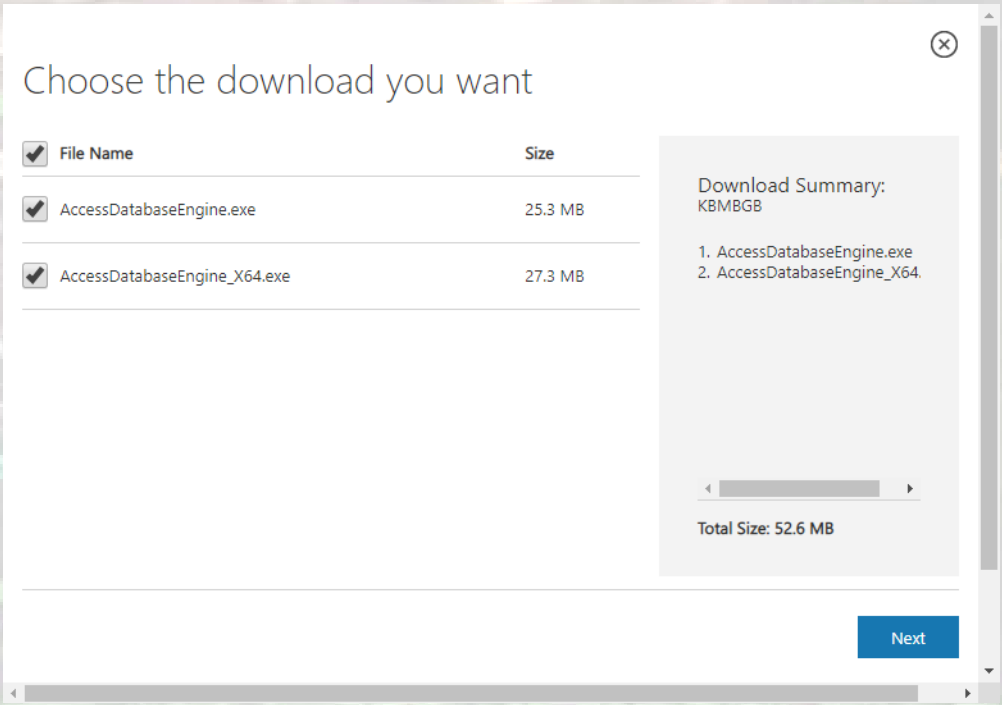


Рисунок 8.12 – Вікно вибору компонентів драйвера для завантаження

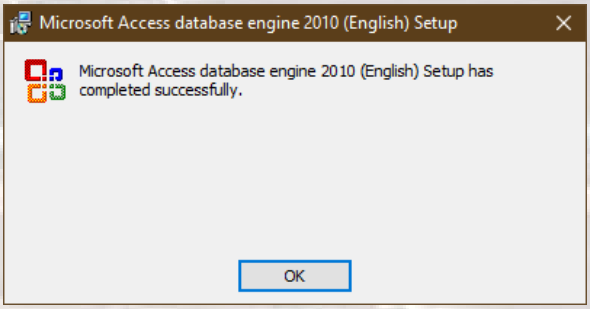


Рисунок 8.13 – Діалогове вікно з повідомленням про вдале вустановлення драйвера

По завершенню встановлення драйверу на вашу систему потрібно буде змінити рядок коду, в якому зазначений провайдер даних та замість рядку `Provider=Microsoft.Jet.System.Data.OleDb.4.0` вписати наступного провайдера `Provider=Microsoft.ACE.OLEDB.12.0`.

Приклад коду C#

```
Підключення = new System.Data.OleDb.
      OleDbConnection("Data Source=D:\\Нова_БД.mdb; User ID=" +
      "Admin; Provider=Microsoft.ACE.OLEDB.12.0;");
```

Тоді код програмного модуля матиме наступний вигляд.

Приклад коду C#

```
namespace РедагуванняБД
{
    public partial class Form1 : Form
    {
      // Програма оновлює записи (Update) в таблиці бази даних MS Access
      // ~ ~ ~ ~ ~ ~ ~ ~
      // Оголошуємо ці змінні поза всіма процедур, щоб
      // Вони були видні з будь-якої з процедур:
      DataSet  НаборДаних;
        System.Data.OleDb.OleDbDataAdapter  Адаптер;
        System.Data.OleDb.OleDbConnection  Підключення;
        System.Data.OleDb.OleDbCommand  Команда;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
          this.panel1.Height = this.button1.Height * 2;
          this.button2.Dock = System.Windows.Forms.DockStyle.Bottom;
          this.button1.Dock = System.Windows.Forms.DockStyle.Bottom;
          this.panel1.Dock = System.Windows.Forms.DockStyle.Bottom;
          this.panel2.Dock = System.Windows.Forms.DockStyle.Fill;
          this.dataGridView1.Dock = System.Windows.Forms.DockStyle.Fill;
```

```
    this.Text = "Редагування БД";
    НаборДаних = new DataSet();
    Підключення = new System.Data.OleDb.
      OleDbConnection("Data Source=D:\\Нова_БД.mdb; User ID=" +
      "Admin; Provider=Microsoft.ACE.OLEDB.12.0;");
    Команда = new System.Data.OleDb.OleDbCommand();
    button1.Text = "Читати з БД"; button1.TabIndex = 0;
    button2.Text = "Зберегти в БД";
  }

  private void button1_Click(object sender, EventArgs e)
  {
    // Читати з БД:
    if (Підключення.State ==
      ConnectionState.Closed) Підключення.Open();
    Адаптер = new System.Data.OleDb.OleDbDataAdapter(
      "Select * From [БД телефонів]", Підключення);
    // Заповнюємо DataSet результатом SQL-запиту
    Адаптер.Fill(НаборДаних, "БД телефонів");
    // Вміст DataSet у вигляді рядка XML для відладки:
    String  РядокXML = НаборДаних.GetXml();
    // Вказати джерело даних для сітки даних:
    dataGridView1.DataSource = НаборДаних;
    // Вказуємо ім'я таблиці в наборі даних:
    dataGridView1.DataMember = "БД телефонів";
    Підключення.Close();
  }

  private void button2_Click(object sender, EventArgs e)
  {
    // Зберегти в базі даних
    Команда.CommandText =
     "UPDATE [БД телефонів] SET [Номер телефону] =?," +
     " ПІБ =? WHERE ([Номер п/п] =?)";
    // Ім'я, тип і довжина параметра
    Команда.Parameters.Add("Номер телефону",
      System.Data.OleDb.OleDbType.VarWChar, 50, "Номер телефону");
    Команда.Parameters.Add(
      "ПІБ", System.Data.OleDb.OleDbType.VarWChar, 50, "ПІБ");
    Команда.Parameters.Add
      (new System.Data.OleDb.OleDbParameter("Original_Номер_п_п",
      System.Data.OleDb.OleDbType.Integer, 0,
      System.Data.ParameterDirection.
      Input, false, (Byte)0, (Byte)0, "Номер п/п",
      System.Data.DataRowVersion.Original, null));
    Адаптер.UpdateCommand = Команда;
    Команда.Connection = Підключення;
    try
    {// Update повертає кількість змінених рядків
      int kol = Адаптер.Update(НаборДаних, "БД телефонів");
      MessageBox.Show("Оновлено " + kol + " записів",
        "Оновлення таблиці", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
```

```
            }
            catch (Exception  Ситуація)
            {
              MessageBox.Show(Ситуація.Message, "Оновлення таблиці",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
          }
      }
}
```

При програмуванні читання з бази даних спочатку за допомогою SQL-запиту ми вибрали всі записи з таблиці (`Select * From [БД телефонів]`) і за допомогою об'єкта класу `Adapter` помістили в набір даних `DataSet`. А потім вказали об'єкт класу `DataSet` як джерело (`DataSource`) для сітки даних `dataGridView1`. Фрагмент роботи програми після читання з бази даних представлений на рис. 8.14.

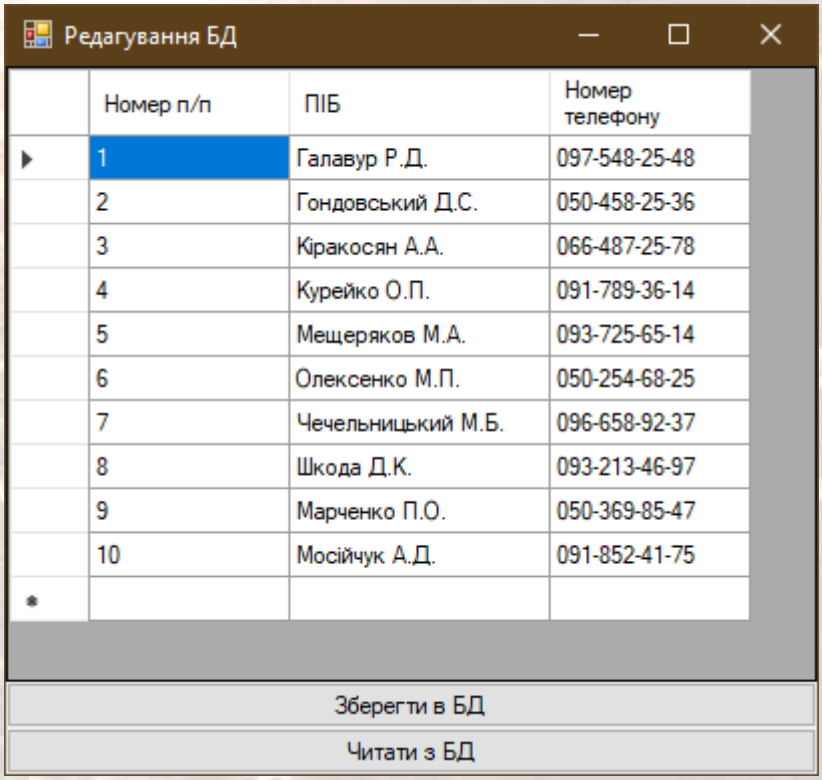


Рисунок 8.14 – Приклад роботи програми редагування таблиці даних

Для нас буде представляти інтерес програмування модифікації записів бази даних. Ця можливість реалізується при обробці події «клацання мишею на кнопці `Зберегти в БД`». Тут властивості `CommandText` присвоєно значення тексту SQL-запиту. В якості замінників параметрів використовуються знаки питання. В даному SQL-запиті мають місце три знаки питання. Їм відповідають три параметра, які повинні вказуватися суворо в порядку проходження знаків питання. Ці параметри задаємо з використанням методу `Parameters.Add`. Тут вказуємо ім'я поля (наприклад, `Номер телефону`), тип, довжину параметра і значення за замовчуванням. Зауважимо, що третій параметр (`Номер п/п`) задається як новий, оскільки він не повинен підлягати редагуванню з боку

користувача, а буде встановлюватися автоматично, незалежно від дій користувача.

Далі в блоці `try ... catch` викликаємо безпосередньо метод Update, який повертає кількість (kol) оновлених записів. У разі невдалого оновлення обробляється виняткова ситуація Exception: об'єкт Exception забезпечує відповідне повідомлення про помилку. Результат роботи програми зі збереженням змін у записах зображений на рис. 8.15.

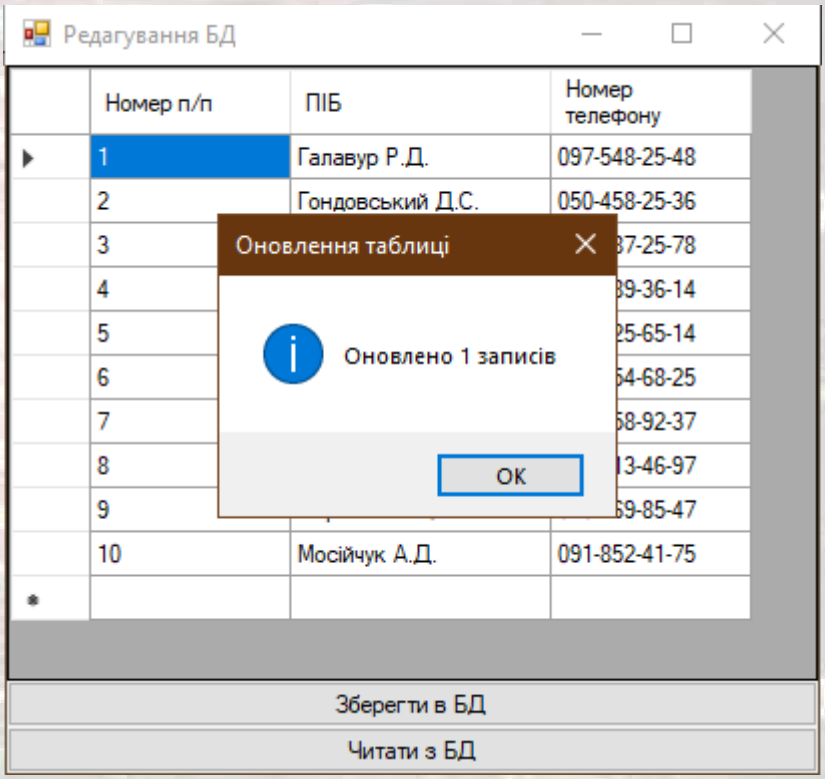


Рисунок 8.15 – Приклад роботи програми зі збереженням змін у таблиці даних

Можна також видаляти записи (рядки з таблиці БД), формуючи в програмному коді відповідний SQL-запит, який передається в об'єкт класу Command. Саме об'єкт Command забезпечує прив'язку SQL-виразу до з'єднання з базою даних. Напишемо найпростіший приклад такої програми.

Для вирішення поставленої задачі достатньо наступного програмного коду.

Приклад коду C++/CLI

```
private: System::Void button3_Click(System::Object^  sender,
System::EventArgs^  e) {
     // Створюємо об'єкт Connection і передаємо йому рядок підключення
     // Рядок підключення:
     auto Підключення = gcnew Data::OleDb::OleDbConnection(
               "Data Source = D:\\Нова_БД.mdb; User ID = Admin;" +
               " Provider = Microsoft.Jet.OLEDB.4.0;");
     Підключення->Open();
     // Створюємо об'єкт класу Command, передаючи йому SQL-команду
     auto Команда = gcnew Data::OleDb::OleDbCommand(
```

```
            "Delete * From [БД телефонів] Where ПІБ Like 'Ма%'",
            Підключення);
        // Виконання команди SQL
        int i = Команда->ExecuteNonQuery();
        // i - кількість видалених записів
        if (i> 0) MessageBox::Show(
            "Вилучено " + i.ToString() + " записів, які містять в полі
            ПІБ фрагмент 'Ма*'",
            "Видалення записів", MessageBoxButtons::OK,
            MessageBoxIcon::Information);
        if (i == 0) MessageBox::Show(
            "Запис, що містить в полі ПІБ фрагмент 'Ма*', не знайдено",
            "Видалення записів", MessageBoxButtons::OK,
            MessageBoxIcon::Information);
        Підключення->Close();
}
```

Приклад коду C#

```
private void button3_Click(object sender, EventArgs e)
{
        // Створюємо об'єкт Connection і передаємо йому рядок підключення
        // Рядок підключення:
        var Підключення = new System.Data.OleDb.OleDbConnection(
            "Data Source = D:\\Нова_БД.mdb; User ID = Admin;" +
            " Provider = Microsoft.Jet.OLEDB.4.0;");
        Підключення.Open();
        // Створюємо об'єкт класу Command, передаючи йому SQL-команду
        var Команда = new System.Data.OleDb.OleDbCommand(
            "Delete * From [БД телефонів] Where ПІБ Like 'Ма%'",
        Підключення);
        // Виконання команди SQL
        int i = Команда.ExecuteNonQuery();
        // i - кількість видалених записів
        if (i > 0) MessageBox.Show(
            "Вилучено " + i.ToString() +
            " записів, які містять в полі ПІБ фрагмент 'Ма*'",
            "Видалення записів", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
        if (i == 0) MessageBox.Show(
            "Запис, що містить в полі ПІБ фрагмент 'Ма*', не знайдено",
            "Видалення записів", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
        Підключення.Close();
}
```

В даному випадку ми скористались програмою з попереднього прикладу, в яку додали об’єкт button3 і програмний код розмістили у обробнику події – клацання на кнопці button3_Click. Кнопка button3 була розміщена всередині об’єкту panel1 та вирівняна за нижнім краєм, а також її властивості Text ми

присвоїли значення "Видалити записи". В результаті вікно програми набуло вигляду (рис. 8.16).

Тут при створенні об'єкта класу OleDbCommand заданий SQL-запит на видалення (Delete) всіх записів, що містить в полі ПІБ фрагмент тексту Ма*, причому малі та великі літери є рівнозначними, тобто будуть видалені записи, що містять Ма*, ма*, МА* та інші комбінації. Таким чином, пошук записів ведеться без урахування регістру (case-insensitive search).

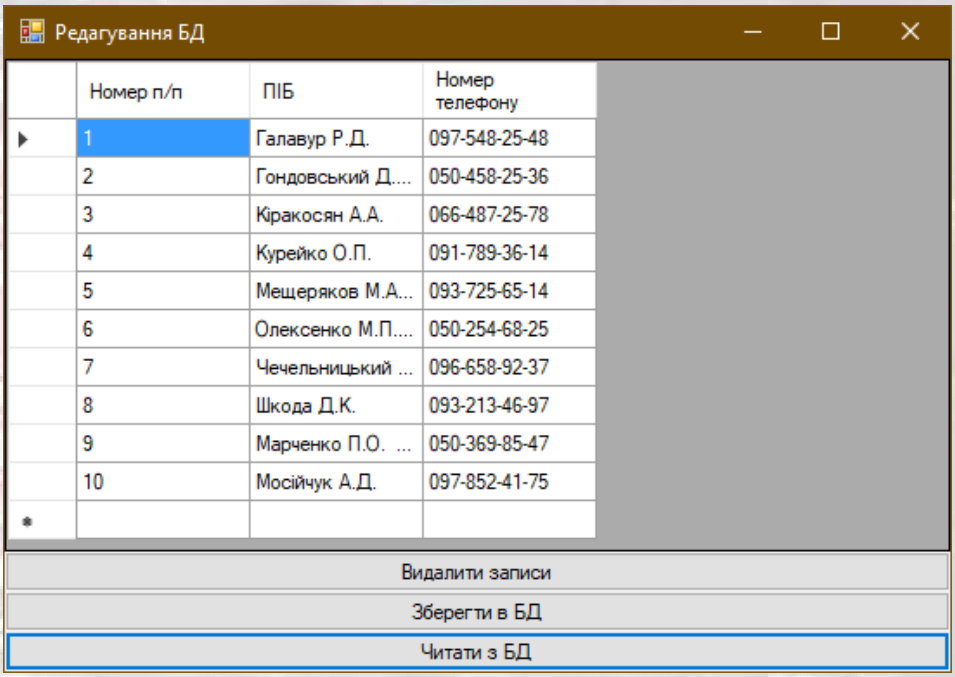


Рисунок 8.16 – Приклад роботи програми редагування таблиці даних з видаленням записів

Зауважимо, що тут для виконання команди SQL використаний метод ExecuteNonQuery(). Він повертає в змінну i кількість видалених записів (рис. 8.17).

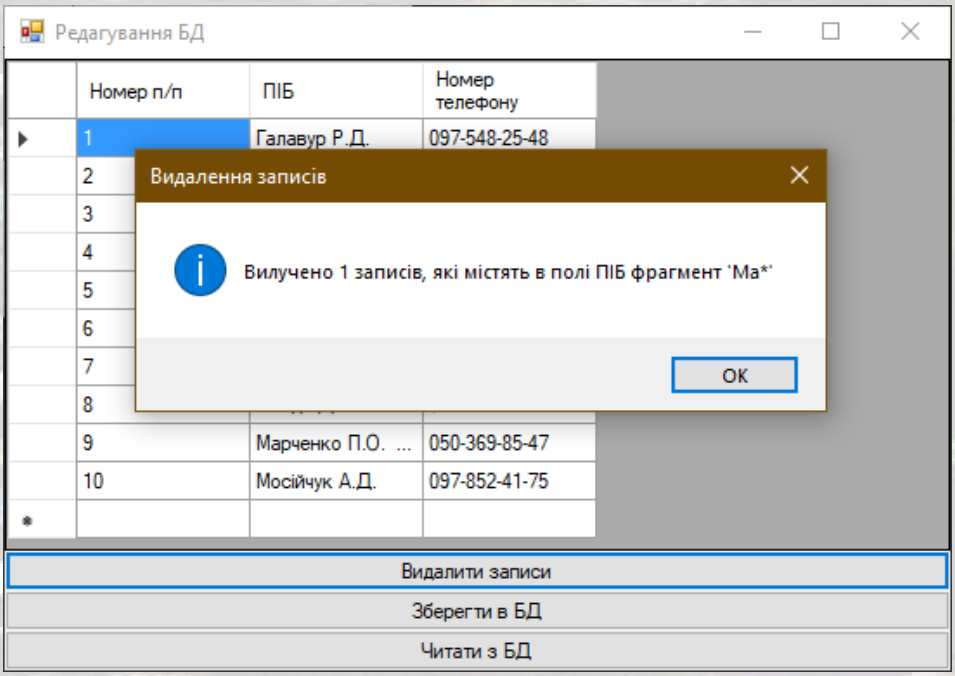


Рисунок 8.17 – Повідомлення про видалення записів з таблиці

Якщо `i = 0`, значить, записів з таким контекстом, не знайдено, і жоден запис не очищено. Після запуску команди Видалити записи в таблиці даних був знайдений запис з прізвищем, що починається на «Ма». В даному прикладі це був запис «Марченко П.О.» і видалений з таблиці даних. Після повторного зчитування таблиці бази даних у об'єк dataGridView1, у вікні програми вже відсутній запис з прізвищем «Марченко» (рис. 8.18).

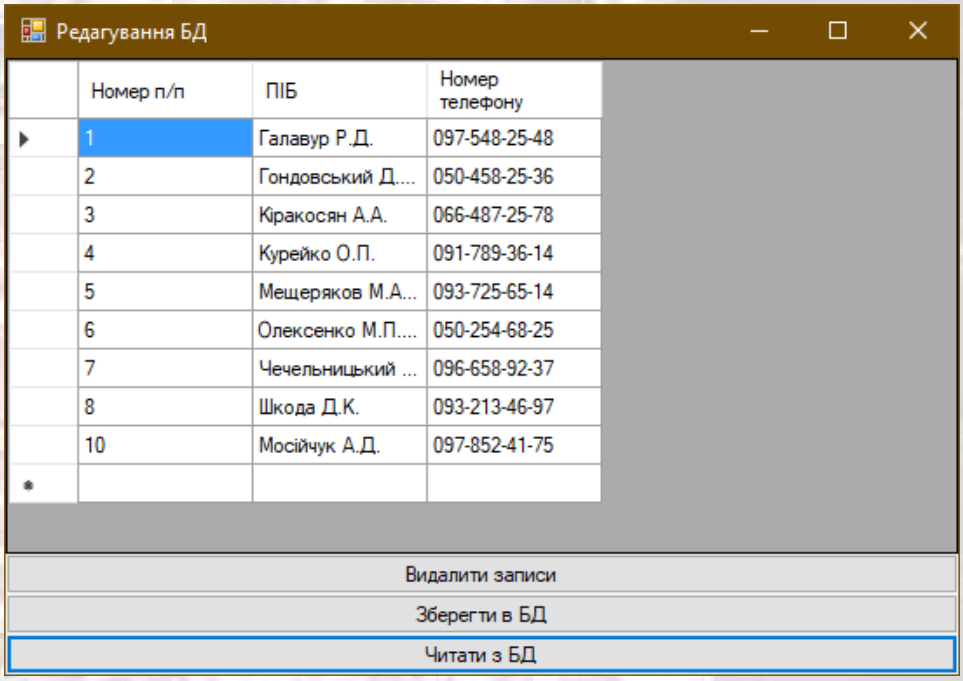


Рисунок 8.18 – Результат роботи команди Видалити записи

В лістингу 8.1 (C++/CLI) та лістингах 8.2-8.3 (C#) наведений код програми, яка дозволяє переглядати, редагувати та видаляти записи з таблиці бази даних.

Програмний код

Лістинг 8.1

Приклад коду C++/CLI

```
// Програма перегляду, редагування та видалення записів з таблиці БД
#pragma once
namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::Data::OleDb;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}

	private: System::Windows::Forms::Panel^  panel1;
	private: System::Windows::Forms::Button^  button2;
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::Panel^  panel2;
	private: System::Windows::Forms::DataGridView^  dataGridView1;
	private: System::Windows::Forms::Button^  button3;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
```

```
        {
                this->panel1 = (gcnew System::Windows::Forms::Panel());
                this->button2 = (gcnew System::Windows::Forms::Button());
                this->button1 = (gcnew System::Windows::Forms::Button());
                this->panel2 = (gcnew System::Windows::Forms::Panel());
                this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());
                this->button3 = (gcnew System::Windows::Forms::Button());
                this->panel1->SuspendLayout();
                this->panel2->SuspendLayout();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->BeginInit();
                this->SuspendLayout();
                //
                // panel1
                //
                this->panel1->Controls->Add(this->button3);
                this->panel1->Controls->Add(this->button2);
                this->panel1->Controls->Add(this->button1);
                this->panel1->Location = System::Drawing::Point(0, 161);
                this->panel1->Name = L"panel1";
                this->panel1->Size = System::Drawing::Size(284, 100);
                this->panel1->TabIndex = 3;
                //
                // button2
                //
                this->button2->Location = System::Drawing::Point(157, 74);
                this->button2->Name = L"button2";
                this->button2->Size = System::Drawing::Size(75, 23);
                this->button2->TabIndex = 4;
                this->button2->Text = L"button2";
                this->button2->UseVisualStyleBackColor = true;
                this->button2->Click += gcnew System::EventHandler(this,
&MyForm::button2_Click);
                //
                // button1
                //
                this->button1->Location = System::Drawing::Point(47, 74);
                this->button1->Name = L"button1";
                this->button1->Size = System::Drawing::Size(75, 23);
                this->button1->TabIndex = 3;
                this->button1->Text = L"button1";
                this->button1->UseVisualStyleBackColor = true;
                this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                //
                // panel2
                //
                this->panel2->Controls->Add(this->dataGridView1);
                this->panel2->Location = System::Drawing::Point(32, 26);
                this->panel2->Name = L"panel2";
                this->panel2->Size = System::Drawing::Size(200, 100);
                this->panel2->TabIndex = 4;
                //
                // dataGridView1
                //
                this->dataGridView1->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
                this->dataGridView1->Location = System::Drawing::Point(-42, -28);
                this->dataGridView1->Name = L"dataGridView1";
                this->dataGridView1->Size = System::Drawing::Size(284, 157);
                this->dataGridView1->TabIndex = 1;
                //
                // button3
                //
                this->button3->Location = System::Drawing::Point(107, 45);
                this->button3->Name = L"button3";
                this->button3->Size = System::Drawing::Size(75, 23);
```

```
                    this->button3->TabIndex = 5;
                    this->button3->Text = L"button3";
                    this->button3->UseVisualStyleBackColor = true;
                    this->button3->Click += gcnew System::EventHandler(this,
&MyForm::button3_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->panel2);
                    this->Controls->Add(this->panel1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->panel1->ResumeLayout(false);
                    this->panel2->ResumeLayout(false);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->EndInit();
                    this->ResumeLayout(false);

                }
#pragma endregion

                // Програма оновлює записи (Update) в таблиці бази даних MS Access
                // ~ ~ ~ ~ ~ ~ ~ ~ ~
                // Оголошуємо ці змінні поза всіма процедур, щоб
                // Вони були видні з будь-якої з процедур:
                DataSet ^ НабірДаних;
                OleDb::OleDbDataAdapter ^ Адаптер;
                OleDb::OleDbConnection ^ Підключення;
                OleDb::OleDbCommand ^ Команда;
private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
        panel1->Height = button1->Height*3;
        button3->Dock = System::Windows::Forms::DockStyle::Bottom;
        button2->Dock = System::Windows::Forms::DockStyle::Bottom;
        button1->Dock = System::Windows::Forms::DockStyle::Bottom;
        panel1->Dock = System::Windows::Forms::DockStyle::Bottom;
        panel2->Dock = System::Windows::Forms::DockStyle::Fill;
        dataGridView1->Dock = System::Windows::Forms::DockStyle::Fill;
        Text = "Редагування БД";
        НабірДаних = gcnew DataSet();
        Підключення = gcnew OleDb::
              OleDbConnection("Data Source=D:\\Нова_БД.mdb; User ID=" +
                    "Admin; Provider=Microsoft.Jet.OLEDB.4.0;");
        Команда = gcnew OleDb::OleDbCommand();
        button1->Text = "Читати з БД";
        button1->TabIndex = 0;
        button2->Text = "Зберегти в БД";
        button3->Text = "Видалити записи";
        }

private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
        // Читати з БД:
        if (Підключення->State ==
              ConnectionState::Closed) Підключення->Open();
        Адаптер = gcnew OleDb::OleDbDataAdapter(
              "Select * From [БД телефонів]", Підключення);
        // Заповнюємо DataSet результатом SQL-запиту
        Адаптер->Fill(НабірДаних, "БД телефонів");
        // Вміст DataSet у вигляді рядка XML для відладки:
        String ^ РядокXML = НабірДаних->GetXml();
        // Вказати джерело даних для сітки даних:
        dataGridView1->DataSource = НабірДаних;
        // Вказуємо ім'я таблиці в наборі даних:
        dataGridView1->DataMember = "БД телефонів";
```

```
        Підключення->Close();
        }

private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
        // Зберегти в базі даних
        Команда->CommandText = "UPDATE [БД телефонів] SET [Номер телефону] =?," +
                " ПІБ =? WHERE ([Номер п/п] =?)";
        // Ім'я, тип і довжина параметра
        Команда->Parameters->Add("Номер телефону",
                OleDb::OleDbType::VarWChar, 50, "Номер телефону");
        Команда->Parameters->Add(
                "ПІБ", OleDb::OleDbType::VarWChar, 50, "ПІБ");
        Команда->Parameters->Add
                (gcnew OleDb::OleDbParameter("Original_Номер_п_п",
                        OleDb::OleDbType::Integer, 0, System::Data::ParameterDirection::
                        Input, false, (Byte)0, (Byte)0, "Номер п/п",
                        System::Data::DataRowVersion::Original, nullptr));
        Адаптер->UpdateCommand = Команда;
        Команда->Connection = Підключення;
        try
        {// Update повертає кількість змінених рядків
                int kol = Адаптер->Update(НабірДаних, "БД телефонів");
                MessageBox::Show("Оновлено " + kol + " записів", "Оновлення таблиці",
MessageBoxButtons::OK, MessageBoxIcon::Information);
        }
        catch (Exception ^ Ситуація)
        {
                MessageBox::Show(Ситуація->Message, "Оновлення таблиці",
MessageBoxButtons::OK, MessageBoxIcon::Error);
        }
        }

private: System::Void button3_Click(System::Object^  sender, System::EventArgs^  e) {
        // Створюємо об'єкт Connection і передаємо йому рядок підключення
        // Рядок підключення:
        auto Підключення = gcnew Data::OleDb::OleDbConnection(
                        "Data Source = D:\\Нова_БД.mdb; User ID = Admin;" +
                        " Provider = Microsoft.Jet.OLEDB.4.0;");
        Підключення->Open();
        // Створюємо об'єкт класу Command, передаючи йому SQL-команду
        auto Команда = gcnew Data::OleDb::OleDbCommand(
                "Delete * From [БД телефонів] Where ПІБ Like 'Ма%'", Підключення);
        // Виконання команди SQL
        int i = Команда->ExecuteNonQuery();
        // i - кількість видалених записів
        if (i> 0) MessageBox::Show(
                "Вилучено " + i.ToString() +
                " записів, які містять в полі ПІБ фрагмент 'Ма*'",
                "Видалення записів", MessageBoxButtons::OK, MessageBoxIcon::Information);
        if (i == 0) MessageBox::Show(
                "Запис, що містить в полі ПІБ фрагмент 'Ма*', не знайдено",
                "Видалення записів", MessageBoxButtons::OK, MessageBoxIcon::Information);
        Підключення->Close();
}
};
}
```

Лістинг 8.2

Приклад коду C#

```
// Програма перегляду, редагування та видалення записів з таблиці БД
//Файл Form1.Designer.cs

namespace РедагуванняБД
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.panel1 = new System.Windows.Forms.Panel();
            this.button2 = new System.Windows.Forms.Button();
            this.button1 = new System.Windows.Forms.Button();
            this.panel2 = new System.Windows.Forms.Panel();
            this.dataGridView1 = new System.Windows.Forms.DataGridView();
            this.button3 = new System.Windows.Forms.Button();
            this.panel1.SuspendLayout();
            this.panel2.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).BeginInit();
            this.SuspendLayout();
            //
            // panel1
            //
            this.panel1.Controls.Add(this.button3);
            this.panel1.Controls.Add(this.button2);
            this.panel1.Location = new System.Drawing.Point(158, 48);
            this.panel1.Name = "panel1";
            this.panel1.Size = new System.Drawing.Size(200, 100);
            this.panel1.TabIndex = 0;
            //
            // button2
            //
            this.button2.Location = new System.Drawing.Point(73, 55);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(75, 23);
            this.button2.TabIndex = 1;
            this.button2.Text = "button2";
```

```
            this.button2.UseVisualStyleBackColor = true;
            this.button2.Click += new System.EventHandler(this.button2_Click);
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(439, 64);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(75, 23);
            this.button1.TabIndex = 0;
            this.button1.Text = "button1";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // panel2
            //
            this.panel2.Controls.Add(this.dataGridView1);
            this.panel2.Location = new System.Drawing.Point(158, 170);
            this.panel2.Name = "panel2";
            this.panel2.Size = new System.Drawing.Size(200, 100);
            this.panel2.TabIndex = 1;
            //
            // dataGridView1
            //
            this.dataGridView1.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            this.dataGridView1.Location = new System.Drawing.Point(73, 34);
            this.dataGridView1.Name = "dataGridView1";
            this.dataGridView1.Size = new System.Drawing.Size(240, 150);
            this.dataGridView1.TabIndex = 0;
            //
            // button3
            //
            this.button3.Location = new System.Drawing.Point(18, 26);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(75, 23);
            this.button3.TabIndex = 3;
            this.button3.Text = "button3";
            this.button3.UseVisualStyleBackColor = true;
            this.button3.Click += new System.EventHandler(this.button3_Click);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(546, 353);
            this.Controls.Add(this.panel2);
            this.Controls.Add(this.panel1);
            this.Controls.Add(this.button1);
            this.Name = "Form1";
            this.Text = "Form1";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.panel1.ResumeLayout(false);
            this.panel2.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();
            this.ResumeLayout(false);
        }

        #endregion

        private System.Windows.Forms.Panel panel1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Panel panel2;
        private System.Windows.Forms.DataGridView dataGridView1;
        private System.Windows.Forms.Button button3;
    }
}
```

Лістинг 8.3

Приклад коду C#

```
// Програма перегляду, редагування та видалення записів з таблиці БД
//Файл Form1.cs


using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace РедагуванняБД
{
    public partial class Form1 : Form
    {
        // Програма оновлює записи (Update) в таблиці бази даних MS Access
        // ~ ~ ~ ~ ~ ~ ~ ~ ~
        // Оголошуємо ці змінні поза всіма процедур, щоб
        // Вони були видні з будь-якої з процедур:
        DataSet  НабірДаних;
           System.Data.OleDb.OleDbDataAdapter  Адаптер;
           System.Data.OleDb.OleDbConnection  Підключення;
           System.Data.OleDb.OleDbCommand  Команда;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.panel1.Height = this.button1.Height * 2;
            this.button3.Dock = DockStyle.Bottom;
            this.button2.Dock = DockStyle.Bottom;
            this.button1.Dock = DockStyle.Bottom;
            this.panel1.Dock = DockStyle.Bottom;
            this.panel2.Dock = DockStyle.Fill;
            this.dataGridView1.Dock = DockStyle.Fill;
            this.Text = "Редагування БД";
            НабірДаних = new DataSet();
            Підключення = new System.Data.OleDb.
                OleDbConnection("Data Source=D:\\Нова_БД.mdb; User ID=" +
                    "Admin; Provider=Microsoft.ACE.OLEDB.12.0;");
            //Замінений рядок Provider=Microsoft.Jet.System.Data.OleDb.4.0
            Команда = new System.Data.OleDb.OleDbCommand();
            button1.Text = "Читати з БД"; button1.TabIndex = 0;
            button2.Text = "Зберегти в БД";
            button3.Text = "Видалити записи";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // Читати з БД:
            if (Підключення.State ==
                ConnectionState.Closed) Підключення.Open();
            Адаптер = new System.Data.OleDb.OleDbDataAdapter(
                "Select * From [БД телефонів]", Підключення);
            // Заповнюємо DataSet результатом SQL-запиту
```

```
            Адаптер.Fill(НабірДаних, "БД телефонів");
            // Вміст DataSet у вигляді рядка XML для відладки:
            String  РядокXML = НабірДаних.GetXml();
            // Вказати джерело даних для сітки даних:
            dataGridView1.DataSource = НабірДаних;
            // Вказуємо ім'я таблиці в наборі даних:
            dataGridView1.DataMember = "БД телефонів";
            Підключення.Close();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            // Зберегти в базі даних
            Команда.CommandText = "UPDATE [БД телефонів] SET [Номер телефону] =?," +
                " ПІБ =? WHERE ([Номер п/п] =?)";
            // Ім'я, тип і довжина параметра
            Команда.Parameters.Add("Номер телефону", System.Data.OleDb.OleDbType.VarWChar,
50, "Номер телефону");
            Команда.Parameters.Add("ПІБ", System.Data.OleDb.OleDbType.VarWChar, 50, "ПІБ");
            Команда.Parameters.Add
                (new System.Data.OleDb.OleDbParameter("Original_Номер_п_п",
                    System.Data.OleDb.OleDbType.Integer, 0, ParameterDirection.
                     Input, false, (Byte)0, (Byte)0, "Номер п/п",
                    DataRowVersion.Original, null));
            Адаптер.UpdateCommand = Команда;
            Команда.Connection = Підключення;
            try
            {// Update повертає кількість змінених рядків
                int kol = Адаптер.Update(НабірДаних, "БД телефонів");
                MessageBox.Show("Оновлено " + kol + " записів", "Оновлення таблиці",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception  Ситуація)
               {
                MessageBox.Show(Ситуація.Message, "Оновлення таблиці",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            // Створюємо об'єкт Connection і передаємо йому рядок підключення
            // Рядок підключення:
            var Підключення = new System.Data.OleDb.OleDbConnection(
                    "Data Source = D:\\Нова_БД.mdb; User ID = Admin;" +
                    " Provider = Microsoft.Jet.OLEDB.4.0;");
            Підключення.Open();
            // Створюємо об'єкт класу Command, передаючи йому SQL-команду
            var Команда = new System.Data.OleDb.OleDbCommand(
                "Delete * From [БД телефонів] Where ПІБ Like 'Ма%'", Підключення);
            // Виконання команди SQL
            int i = Команда.ExecuteNonQuery();
            // i - кількість видалених записів
            if (i > 0) MessageBox.Show(
                 "Вилучено " + i.ToString() + " записів, які містять в полі ПІБ фрагмент
'Ма*'",
                 "Видалення записів", MessageBoxButtons.OK, MessageBoxIcon.Information);
            if (i == 0) MessageBox.Show(
                "Запис, що містить в полі ПІБ фрагмент 'Ма*', не знайдено",
                "Видалення записів", MessageBoxButtons.OK, MessageBoxIcon.Information);
            Підключення.Close();

        }
    }
}
```

ПРАКТИЧНА РОБОТА №9

9 Використання у програмах C++/CLI та C# функцій MS Word та MS Excel

9.1 Перевірка правопису засобами MS Word

Завдання 1.

Необхідно створити програму, яка дозволяє користувачеві ввести будь-які слова, речення в текстове поле і після натиснення відповідної кнопки перевірити орфографію введеного тексту. Для безпосередньої перевірки орфографії скористаємося функцією Checkspelling об'єктної бібліотеки MS Word.

Для вирішення цієї задачі запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо додаток шаблону Windows Forms. Перенесемо з Панелі елементів (Toolbox) в проектовану форму елемент управління Panel і в ньому розмістимо елемент RichTextBox. Також на формі розмістимо елемент Button.

Для можливості використання функцій з бібліотеки MS Word потрібно через меню Project ⇒ Add Reference… підключити відповідну бібліотеку. В залежності від встановленої версії програмного пакету MS Office назва бібліотеки може відрізнятися номером. Для пакету MS Office 2016 бібліотека матиме назву Microsoft Word 16.0 Object Libary (рис. 9.1).

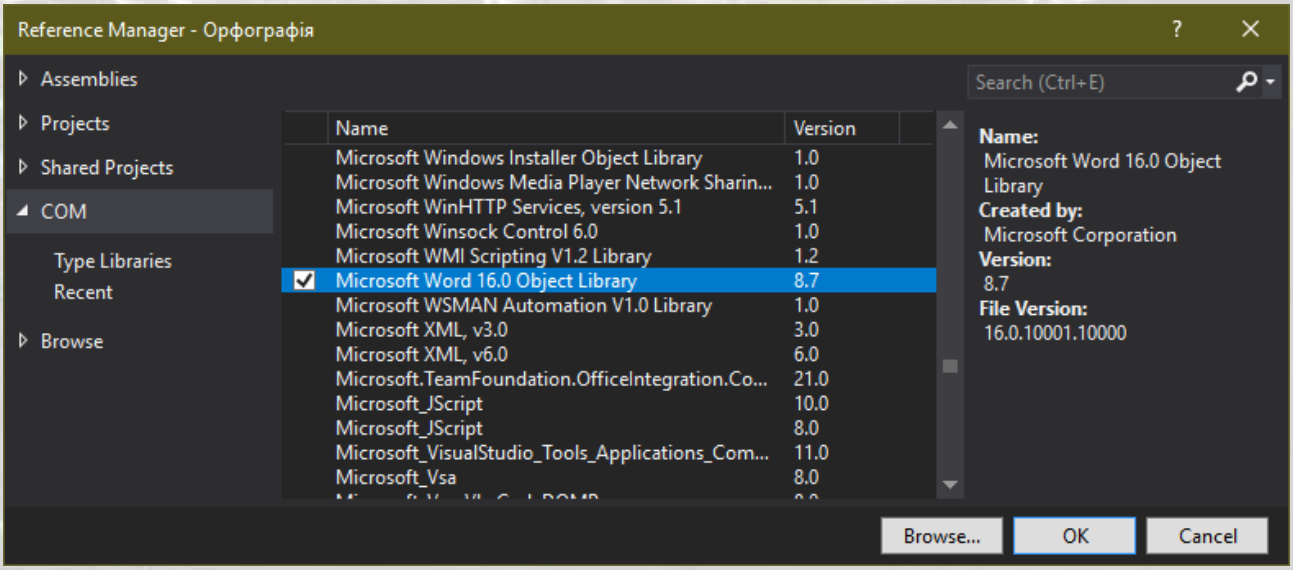


Рисунок 9.1 – Діалогове вікно додавання бібліотеки MS Word

У разі вдалого підключення бібліотеки вона з'явится у вікні оглядача рішень під назвою Microsoft.Office.Interop.Word (рис. 9.2).

Далі введемо програмний код, представлений в лістингу нижче.

Приклад коду C++/CLI

```
#pragma endregion

private: System::Void MyForm_Load(System::Object^  sender,
System::EventArgs^  e) {
      // У пункті меню Project виберемо команду Add Reference.
      // Потім, якщо на вашому комп'ютері встановлений MS Office 2016,
      // то на вкладці COM двічі клацнемо по посиланню
      // на бібліотеку Microsoft Word 16.0 Object Libary.
      richTextBox1->Clear();
      button1->Text = "Перевірка орфографії";
      richTextBox1->TabIndex = 0;
      button1->TabIndex = 1;
      button1->Dock = System::Windows::Forms::DockStyle::Bottom;
      panel1->Dock = System::Windows::Forms::DockStyle::Fill;
      richTextBox1->Dock = System::Windows::Forms::DockStyle::Fill;
      Text = "Орфографія";
}

private: System::Void button1_Click_1(System::Object^  sender,
System::EventArgs^  e) {
      auto Ворд1 = gcnew Microsoft::Office::Interop::Word::Application();
      // Ворд1-> Visible = false;
      System::Object ^ t = Type::Missing;
      // Відкриваємо новий порожній документ:
      auto Документ = Ворд1->Documents->Add(t, t, t, t);
      // Вводимо в порожній документ текст з текстового поля:
      Документ->Words->First->InsertBefore(richTextBox1->Text);
      // Перевірка орфографії:
      Документ->CheckSpelling(t, t, t, t, t, t, t, t, t, t, t, t);
      // Отримуємо виправлений текст:
      String ^ ВиправленнийТекст = Документ->Content-> default;
      // Повертаємо в текстове поле виправлений текст:
      richTextBox1->Text = ВиправленнийТекст->Replace("\r", "");
      System::Object ^ tt = false;
      Ворд1->Documents->Close(tt, t, t);
      // Закрити документ Word без збереження:
      Ворд1->Quit(tt, t, t);
      Ворд1 = nullptr;
}
```

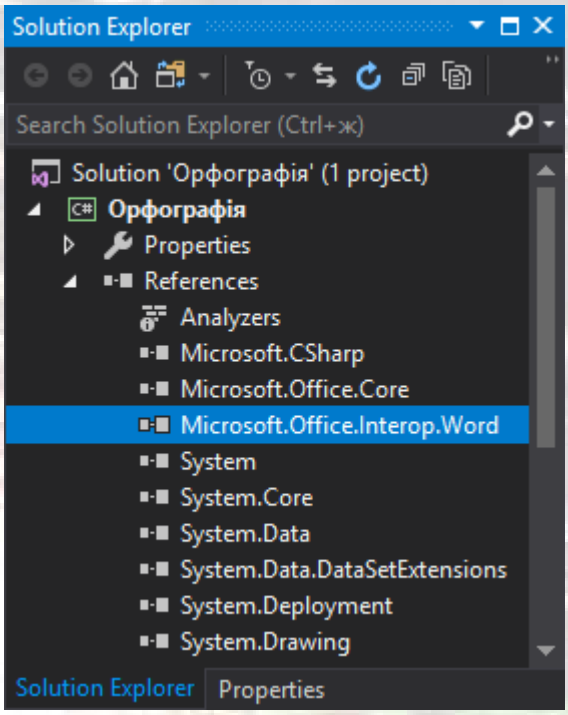


Рисунок 9.2 – Додана бібліотека `Microsoft.Office.Interop.Word`

Приклад коду C#

```
private void Form1_Load(object sender, EventArgs e)
{
      // У пункті меню Project виберемо команду Add Reference.
      // Потім, якщо на вашому комп'ютері встановлений MS Office 2016,
      // То на вкладці COM двічі клацнемо по посиланню
      // На бібліотеку Microsoft Word 16.0 Object Libary.
      richTextBox1.Clear();
      button1.Text = "Перевірка орфографії";
      richTextBox1.TabIndex = 0;
      button1.TabIndex = 1;
      button1.Dock = System.Windows.Forms.DockStyle.Bottom;
      panel1.Dock = System.Windows.Forms.DockStyle.Fill;
      richTextBox1.Dock = System.Windows.Forms.DockStyle.Fill;
      Text = "Орфографія";
}

private void button1_Click(object sender, EventArgs e)
{
      var Ворд1 = new Microsoft.Office.Interop.Word.Application();
      // Ворд1. Visible = false;
      System.Object  t = Type.Missing;
      // Відкриваємо новий порожній документ:
      var Документ = Ворд1.Documents.Add(t, t, t, t);
      // Вводимо в порожній документ текст з текстового поля:
      Документ.Words.First.InsertBefore(richTextBox1.Text);
      // Перевірка орфографії:
      Документ.CheckSpelling(t, t, t, t, t, t, t, t, t, t, t, t);
```

```
        // Отримуємо виправлений текст:
        String  ВиправленнийТекст = Документ.Content.Text;
        // Повертаємо в текстове поле виправлений текст:
        richTextBox1.Text = ВиправленнийТекст.Replace("\r", "");
        System.Object  tt =
Microsoft.Office.Interop.Word.WdSaveOptions.wdDoNotSaveChanges;
        Ворд1.Documents.Close(tt, t, t);
        // Закрити документ Word без збереження:
        Ворд1.Quit(tt, t, t);
        Ворд1 = null;
}
```

Як видно з тексту програми, при обробці події завантаження форми Form1_Load очищається текстове поле, ініціалізується назва кнопки Перевірка орфографії та заголовок вікна Орфографія, виконується вирівнювання елементів інтерфейсу в межах форми, а також за допомогою властивості TabIndex задається порядок переключення фокусу між об'єктами форми за допомогою клавіші Tab.

При обробці події клацання по кнопці Button1_Click створюється новий об'єкт Ворд1 класу Word.Application, і командою Documents.Add відкривається новий документ. Тут ми використовували поле Type.Missing для отримання значення параметра методу за замовчуванням. Далі весь введений користувачем текст копіюється в цей документ (команда InsertBefore). Потім відбувається безпосередня перевірка орфографії методом Checkspelling. Якщо текст перевірений, то правильний варіант написання слова знаходиться тепер в одноній з властивостей об'єкта класу Word.Document, а саме у властивості Content.Text. Транслятор Visual C++ вимагає, щоб до цієї властивості ми звернулися, використовуючи ключове слово default. Тепер виправлений текст повертаємо назад в текстове поле. Далі документ MS Word закриваємо без збереження змін wdDoNotSaveChanges, тому при роботі програми ми цей документ навіть не помічаємо.

Тепер перевіримо, як працює написана нами програма. Запустимо її, натиснувши на функціональну клавішу F5, і в текстове поле введемо яке-небудь слово з помилкою. Після клацання на кнопці Перевірка орфографії отримаємо діалогове вікно, подібне представленому на рис. 9.3.

Виберемо правильний варіант написання слова і клацнемо на кнопці Замінити, при цьому діалогове вікно закриється, а в нашому текстовому полі на формі з'явиться виправлене слово.

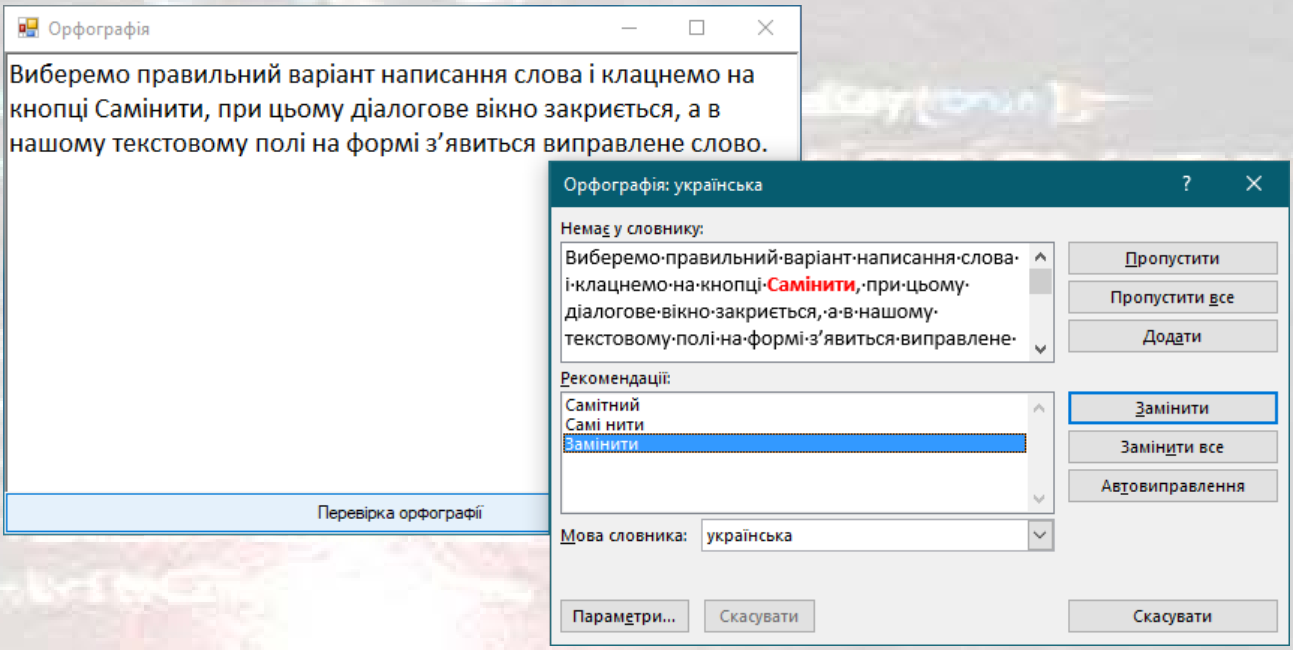


Рисунок 9.3 – Вікно програми з діалоговим вікном Орфографія

У наступному лістингу наведемо інший, більш короткий варіант вирішення цього завдання з використанням об'єкта класу `Selection`.

Приклад коду C++/CLI

```
#pragma endregion

private: System::Void MyForm_Load(System::Object^  sender,
System::EventArgs^  e) {
     // У пункті меню Project виберемо команду Add Reference.
     // Потім, якщо на вашому комп'ютері встановлений MS Office 2016,
     // То на вкладці COM двічі клацнемо по посиланню
     // На бібліотеку Microsoft Word 16.0 Object Libary.
     richTextBox1->Clear();
     button1->Text = "Перевірка орфографії";
     richTextBox1->TabIndex = 0;
     button1->TabIndex = 1;
     button1->Dock = System::Windows::Forms::DockStyle::Bottom;
     panel1->Dock = System::Windows::Forms::DockStyle::Fill;
     richTextBox1->Dock = System::Windows::Forms::DockStyle::Fill;
     Text = "Орфографія";
}

private: System::Void button1_Click_1(System::Object^  sender,
System::EventArgs^  e) {
     auto Ворд1 = gcnew Microsoft::Office::Interop::Word::Application();
     // Ворд1-> Visible = false;
     // Змінна з "порожнім" значенням:
     System::Object  ^ t = Type::Missing;
     // Відкриваємо новий порожній документ MS Word:
     Ворд1->Documents->Add(t, t, t, t);
```

```
        // Копіюємо вміст текстового вікна в документ
        Ворд1->Selection->default = richTextBox1->Text;
        // Для VB і C # було б: Ворд1->Selection->Text
        // Безпосередня перевірка орфографії:
        Ворд1->ActiveDocument->CheckSpelling(t, t, t, t, t, t, t, t, t, t,
t, t);
        // Копіюємо результат назад в текстове поле
        richTextBox1->Text = Ворд1->Selection->default;
        Object ^ tt = false;
        Ворд1->Documents->Close(tt, t, t);
        // Закрити документ Word без збереження:
        Ворд1->Quit(tt, t, t);
        Ворд1 = nullptr;
}
```

Приклад коду C#

```
private void Form1_Load(object sender, EventArgs e)
{
        // У пункті меню Project виберемо команду Add Reference.
        // Потім, якщо на вашому комп'ютері встановлений MS Office 2016,
        // То на вкладці COM двічі клацнемо по посиланню
        // На бібліотеку Microsoft Word 16.0 Object Libary.
        richTextBox1.Clear();
        button1.Text = "Перевірка орфографії";
        richTextBox1.TabIndex = 0;
        button1.TabIndex = 1;
        button1.Dock = System.Windows.Forms.DockStyle.Bottom;
        panel1.Dock = System.Windows.Forms.DockStyle.Fill;
        richTextBox1.Dock = System.Windows.Forms.DockStyle.Fill;
        Text = "Орфографія";
}

private void button1_Click(object sender, EventArgs e)
{
        var Ворд1 = new Microsoft.Office.Interop.Word.Application();
        // Ворд1.Visible = false;
        // Змінна з "порожнім" значенням:
        System.Object t = Type.Missing;
        // Відкриваємо новий порожній документ MS Word:
        Ворд1.Documents.Add(t, t, t, t);
        // Копіюємо вміст текстового вікна в документ
        Ворд1.Selection.Text = richTextBox1.Text;
        // Безпосередня перевірка орфографії:
        Ворд1.ActiveDocument.CheckSpelling(t, t, t, t, t, t, t, t, t, t, t,
t);
        // Копіюємо результат назад в текстове поле
        richTextBox1.Text = Ворд1.Selection.Text;
        Object tt = false;
        Ворд1.Documents.Close(tt, t, t);
        // Закрити документ Word без збереження:
```

```
        Ворд1.Quit(tt, t, t);
        Ворд1 = null;
}
```

У цьому програмному коді аналогічно створюємо новий об'єкт Ворд1 класу Word.Application і, використовуючи метод Add, відкриваємо документ MS Word. Далі скористаємося об'єктом Ворд1.Selection, він оперує з поточною позицією «невидимого» курсору, наприклад, може подати команду введення тексту в документ MS Word (метод TypeText). Ми використовуємо властивість Text цього об'єкта для копіювання вмісту текстового вікна в документ MS Word. Відповідно доступ до цієї властивості має бути реалізований за допомогою операторoa: Ворд1.Selection.Text. Однак транслятор C++ вимагає, щоб до цієї властивості ми звернулися, використовуючи ключове слово default, тому у варіанті коду для C++/CLI використовуються наступні рядки:

```
Ворд1->Selection->default = richTextBox1->Text;
richTextBox1->Text = Ворд1->Selection->default;
```

Далі відбувається безпосередня перевірка орфографії аналогічно попередньому варіанту вирішення цього завдання. Повний текст даного програмного модуля наведений в лістингу 9.1 (C++/CLI) та лістингах 9.2-9.3 (C#).

9.2 Створення таблиці в MS Word

Завдання 2.

Необхідно створити програму, яка дозволяє користувачеві створювати таблиці з використанням функції MS Word.

Отже, запускаємо Visual Studio і в вікні Створення проекту (New Project) виберемо додаток шаблону Windows Forms. Далі до поточного проекту додамо об'єктну бібліотеку MS Word. Для цього в меню Проект (Project) вкажемо команду Додати посилання… (Add Reference…) і на вкладці COM двічі клацнемо по посиланню на бібліотеку Microsoft Word 16.0 Object Library (або інша версія MS Word, наприклад 12.0 Object Library). На екранну форму перенесемо командну кнопку Button, щоб робота програми виглядала більш виразно. Тобто саме після клацання на кнопці буде формуватися таблиця і викликатися MS Word для її відображення. Далі введемо програмний код, представлений в лістингу нижче.

Приклад коду C++/CLI

```
#pragma endregion

private: System::Void MyForm_Load(System::Object^  sender,
System::EventArgs^  e) {
    // У меню Project вкажемо команду Add Reference і на
    // Вкладці COM двічі клацнемо по посиланню на
```

```
        // Бібліотеку Microsoft Word 16.0 Object Library
        button1->Text = "Пуск"; this->Text = "Побудова таблиці";
    }

    private: System::Void button1_Click(System::Object^  sender,
    System::EventArgs^  e) {
        // Ініціалізуємо два рядкових масиви:
        array <String ^> ^ Імена = { "Андрій (роб)", "Світла (моб)",
            "Микола (дом)", "Кафедра (роб)", "Олександр Степанович",
            "Сергій (дом)", "Тетяна Петрівна", "Таксі", "Кінотеатр" };
        array <String ^> ^ Телефон = { "274-88-17", "+38 (067) 7030356",
            "22-345-72", "204-82-12", "223-67-67 доп 32-67", "570-38-76",
            "201-72-23", "555", "216-40-22" };
        // Створюємо новий екземпляр класу Word::Application:
        auto Ворд1 = gcnew Microsoft::Office::Interop::Word::Application();
        Ворд1->Visible = true;
        // Змінна з "порожнім" значенням:
        System::Object ^ t = Type::Missing;
        // Відкриваємо новий документ MS Word:
        auto Документ = Ворд1->Documents->Add(t, t, t, t);
        // Вводимо текст в документ MS WORD з поточної позиції:
        Ворд1->Selection->TypeText("ТАБЛИЦЯ ТЕЛЕФОНІВ");
        // Параметр, який вказує чи показувати межі комірок:
        System::Object ^ t1 = Microsoft::Office::Interop::
            Word::WdDefaultTableBehavior::wdWord9TableBehavior;
        // Параметр, який вказує чи буде додаток Word автоматично
        // змінювати розмір комірок у таблиці для підгонки вмісту:
        System::Object ^ t2 = Microsoft::Office::Interop::Word
            ::WdAutoFitBehavior::wdAutoFitContent;
        // Створюємо таблицю з 9 рядків і 2 стовпців:
        Ворд1->ActiveDocument->Tables->Add(Ворд1->Selection->
            Range, 9, 2, t1, t2);
        // Заповнювати комірки таблиці можна так:
        for (int i = 1; i <= 9; i++)
        {
            Ворд1->ActiveDocument->Tables[1]->Cell(i, 1)->
                default->InsertAfter(Імена[i - 1]);
            Ворд1->ActiveDocument->Tables[1]->Cell(i, 2)->
                default->InsertAfter(Телефон[i - 1]);
            //Програмуючи на C# ми написали б: Ворд1.ActiveDocument.
            //Tables[1].Cell(i,2).Range.InsertAfter(Tel[i-1]);
        }
        // Призначаємо одиниці вимірювання в документі додатка MS Word:
        Object ^ t3 = Microsoft::Office::Interop::Word::WdUnits::wdLine;
        // Параметр, який вказує на дев'ятий рядок у документі MS Word:
        Object ^ рядок9 = 9;
        // Перевести поточну позицію (Selection) за межі таблиці,
        // (В дев'ятий рядок), щоб тут вивести будь-якій текст:
        Ворд1->Selection->MoveDown(t3, рядок9, t);
        // І тут друкуємо наступний текст:
        Ворд1->Selection->TypeText("Який-небудь текст після таблиці");
```

```
        // Зберігати документ немає сенсу, але це вирішить користувач:
        // або можна задати безпосередньо шлях та ім'я файлу
        Object ^ ім’яФайла = "D:\\Таблиця.docx";
        // і одразу зберегти
        Ворд1->ActiveDocument->SaveAs(ім’яФайла, t, t, t, t, t, t, t, t, t,
              t, t, t, t, t, t);
}
```

Приклад коду C#

```
private void Form1_Load(object sender, EventArgs e)
{
        // У меню Project вкажемо команду Add Reference і на
        // Вкладці COM двічі клацнемо по посиланню на
        // Бібліотеку Microsoft Word 16.0 Object Library
        button1.Text = "Пуск"; this.Text = "Побудова таблиці";
}

private void Button1_Click(object sender, EventArgs e)
{
        // Ініціалізуємо два рядкових масиви:
        string[] Імена = {"Андрій (роб)", "Світла (моб)", "Микола (дом)",
          "Кафедра (роб)", "Олександр Степанович", "Сергій (дом)",
          "Тетяна Петрівна", "Таксі", "Кінотеатр" };
        string[] Телефон = {"274-88-17", "+38 (067) 7030356", "22-345-72",
          "204-82-12", "223-67-67 доп 32-67", "570-38-76",
          "201-72-23", "555", "216-40-22" };
        // Створюємо новий екземпляр класу Word.Application:
        var Ворд1 = new Microsoft.Office.Interop.Word.Application
        {
              Visible = true
        };
        // Додаємо затримку у роботі програми в 1 сек
        // для уникнення помилки через затримку у роботі MS Word
        System.Threading.Thread.Sleep(1000);
        // Змінна з "порожнім" значенням:
        System.Object  t = Type.Missing;
        // Відкриваємо новий документ MS Word:
        var Документ = Ворд1.Documents.Add(t, t, t, t);
        // Вводимо текст в документ MS WORD з поточної позиції:
        Ворд1.Selection.TypeText("ТАБЛИЦЯ ТЕЛЕФОНІВ");
        // Параметр, який вказує чи показувати межі комірок:
        System.Object  t1 = Microsoft.Office.Interop.
              Word.WdDefaultTableBehavior.wdWord9TableBehavior;
        // Параметр, який вказує чи буде додаток Word автоматично
        // змінювати розмір комірок у таблиці для підгонки вмісту:
        System.Object t2 = Microsoft.Office.Interop.Word.
              WdAutoFitBehavior.wdAutoFitContent;
        // Створюємо таблицю з 9 рядків і 2 стовпців:
        Ворд1.ActiveDocument.Tables.Add(Ворд1.Selection.Range,
              9, 2, t1, t2);
```

```
        // Заповнювати комірки таблиці можна так:
        for (int i = 1; i <= 9; i++)
        {
            Ворд1.ActiveDocument.Tables[1].Cell(i, 1).Range
                .InsertAfter(Імена[i - 1]);
            Ворд1.ActiveDocument.Tables[1].Cell(i, 2).Range
                .InsertAfter(Телефон[i - 1]);
        }
        // Призначаємо одиниці вимірювання в документі додатка MS Word:
        Object  t3 = Microsoft.Office.Interop.Word.WdUnits.wdLine;
        // Параметр, який вказує на дев'ятий рядок у документі MS Word:
        Object  рядок9 = 9;
        // Перевести поточну позицію (Selection) за межі таблиці,
        // (В дев'ятий рядок), щоб тут вивести будь-якій текст:
        Ворд1.Selection.MoveDown(t3, рядок9, t);
        // І тут друкуємо наступний текст:
        Ворд1.Selection.TypeText("Який-небудь текст після таблиці");
        // Зберігати документ немає сенсу, але це вирішить користувач:
        // або можна задати безпосередньо шлях та ім'я файлу
        Object  назваФайлу = "D:\\Таблиця.docx";
        // і одразу зберегти
        Ворд1.ActiveDocument.SaveAs(назваФайлу, t, t, t, t, t, t, t, t, t,
            t, t, t, t, t, t);
}
```

Дані знаходяться в двох масивах: Имена[] і Телефон[]. Ми створюємо екземпляр об'єкта Word.Application і відкриваємо новий документ Document.Add. Тут ми використовували поле Type.Missing для отримання значення параметра методу за замовчуванням. Потім демонструємо, як можна додавати будь-які тексти в новий документ з C++ - програми. Наприклад, ми вводимо в активний документ текст «ТАБЛИЦЯ ТЕЛЕФОНІВ», використовуючи об'єкт Selection, що визначає поточну позицію «невидимого» курсору і містить методи для введення в документ MS Word.

Потім ми створюємо таблицю, що складається з дев'яти рядків і двох стовпців, причому ширина стовпців буде регулюватися в залежності від вмісту комірок (wdAutoFitContent). Потім в циклі заповнюємо елементи таблиці і виводимо курсор (Selection) за межі таблиці, щоб написати будь-який текст.

Після запуску цієї програми прямо на наших очах в редакторі MS Word сформується таблиця (рис. 9.4), яку при бажанні можна редагувати, зберігати і роздруковувати на принтері.

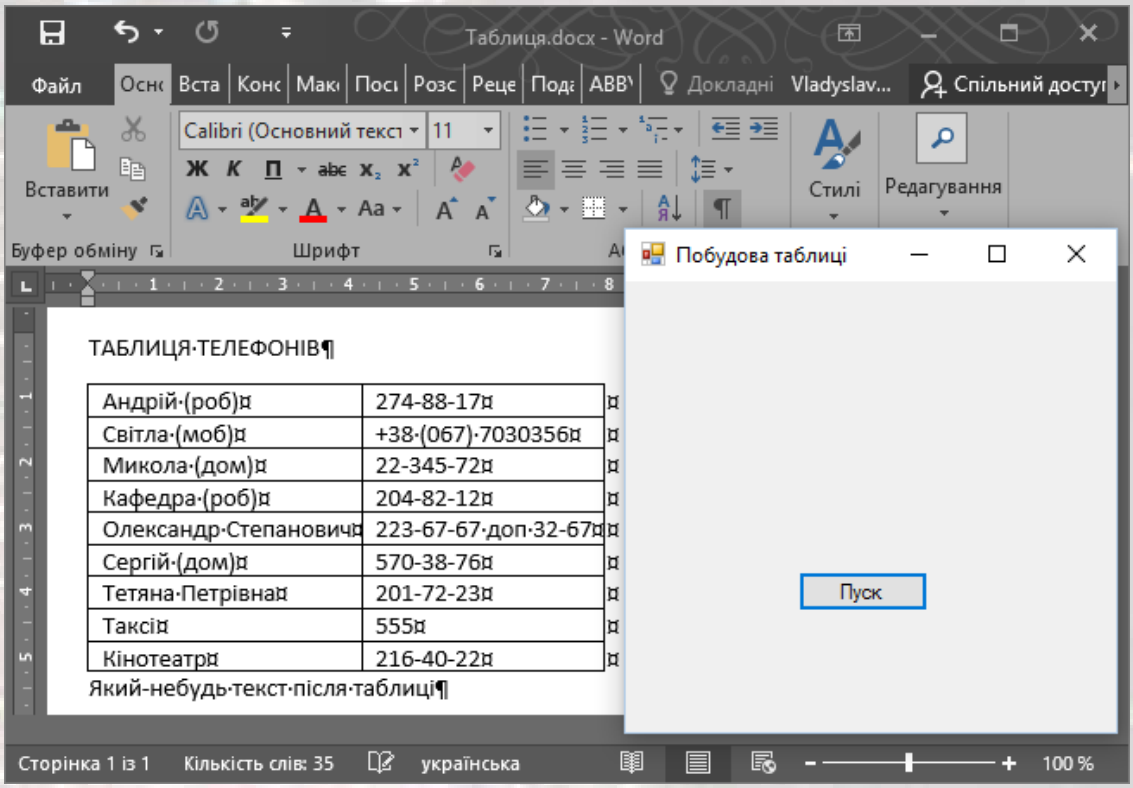


Рисунок 9.4 – Формування таблиці в MS Word з додатку

Окрім того, можна самостійно з додатку зберегти створений документ MS Word, якщо задати шлях збереження та ім'я файлу. Наприклад, наступним чином

Приклад коду C++/CLI

```
// Задаємо безпосередньо шлях та ім'я файлу
Object ^ ім'яФайла = "D:\\Таблиця.docx";
// і одразу зберегти
Ворд1->ActiveDocument->SaveAs(ім'яФайла, t, t, t, t, t, t, t, t, t, t, t,
t, t, t, t);
```

Приклад коду C#

```
// Зберігати документ немає сенсу, але це вирішить користувач:
// або можна задати безпосередньо шлях та ім'я файлу
Object  назваФайлу = "D:\\Таблиця.docx";
// і одразу зберегти
Ворд1.ActiveDocument.SaveAs(назваФайлу, t, t, t, t, t, t, t, t, t,
t, t, t, t, t, t);
```

За допомогою інструментів роботи з MS Word дуже зручно формувати звіти роботи програми у зв'язку з широкими можливостями текстового редактору MS Word по підготовці, форматуванню та оформленню текстових документів.

Повний текст даного програмного модуля наведений в лістингу 9.4 (C++/CLI) та лістингах 9.5-9.6 (C#).

9.3 Використання функцій MS Excel

Завдання 3.

Необхідно створити програму, яка дозволяє користувачеві вирішувати системи лінійних рівнянь з використанням функцій MS Excel.

Використовуючи функції MS Excel в своїй програмі можна вирішувати і більш серйозні завдання. Наприклад, розглянемо, як вирішити систему лінійних алгебраїчних рівнянь.

Вирішувати систему лінійних алгебраїчних рівнянь будемо в матричній формі. Система лінійних алгебраїчних рівнянь виду

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

в матричній формі записується як $A \cdot X = B$, де A – основна матриця системи, або матриця коефіцієнтів

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix},$$

X – матриця-стовпчик невідомих змінних

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix},$$

B – матриця-стовпчик вільних членів

$$B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

Рішення системи лінійних алгебраїчних рівнянь матричним методом визначається за залежністю:

$$X = A^{-1} \times B,$$

де A^{-1} – зворотна матриця коефіцієнтів при невідомих.

Для знаходження зворотної матриці в MS Excel існує функція MINVERSE() (в російськомовній версії МОБР()), аналогічна функції MInverse() в об'єкті WorksheetFunction з бібліотеки об'єктів Microsoft Excel.

Для множення зворотної матриці на вектор вільних членів є відповідно функції MMULT() (в російськомовній версії МПРОИЗ()) та MMult(). Такі функції відсутні в Visual Studio, і в даному випадку ми отримуємо реальний позитивний ефект від підключення до функцій MS Excel.

Розглянемо приклад, в якому знайдемо рішення для наступної системи лінійних алгебраїчних рівнянь:

$$x_1 + 2x_2 + x_3 = -1,$$
$$3x_1 - x_2 - x_3 = -1,$$
$$-2x_1 + 2x_2 + 3x_3 = 5$$

Для вирішення цієї задачі запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо додаток шаблону Windows Forms. У проектовану екранну форму з Панелі елементів (Toolbox) перенесемо три мітки, в які запишемо наші рівняння, і кнопку. У поточний проект підключаємо бібліотеку об'єктів MS Excel. Для цього в меню Проект (Project) виберемо команду Додати посилання... (Add Reference...), потім на вкладці COM двічі клацнемо на посиланні Microsoft Excel 16.0 Object Library (рис. 9.5).

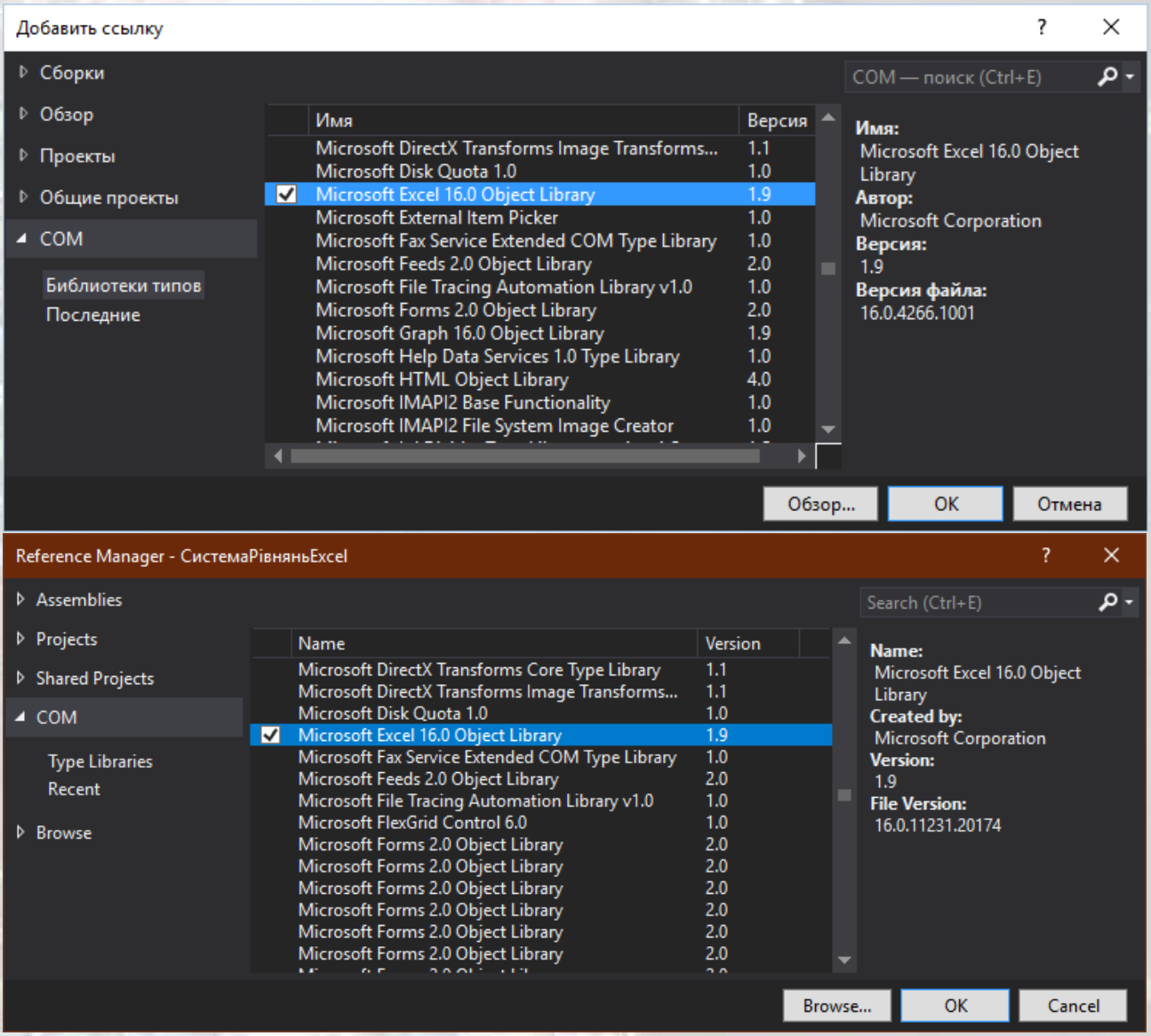


Рисунок 9.5 – Додавання бібліотеки

Програму побудуємо таким чином, що в обробнику події завантаження форми Form1_Load запишемо наші алгебраїчні рівняння в об’єкти Label та

додамо напис на кнопці. Розрахунок безпосередньо будемо проводити у обробнику події натискання кнопки button1_Click, тому в цьому обробнику події і задамо (ініціалізуємо) пряму матрицю у вигляді двовимірного масиву та рядок вільних членів у вигляді одновимірного масиву.

Приклад коду C++/CLI

```
#pragma endregion

private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {
     // Вирішуємо систему
     // x1 + 2*x2 + x3 = -1
     // 3*x1 - x2 - x3 = -1
     // -2*x1 + 2*x2 + 3*x3 = 5
     // Для цієї системи пряма матриця буде мати вигляд:
     // array<double,2> ^ A = gcnew array<double,2>(n, n);
     array<double, 2> ^ A = { { 1, 2, 1 },
                              { 3, -1, -1 },
                              { -2, 2, 3 } };
     // Рядок вільних членів:
     // array<double> ^ B = gcnew array<double>(n);
     // Вільні члени:
     array<double> ^ B = { -1, -1, 5 };
     // Створення екземпляра класу Excel::Application:
     auto XL1 = gcnew Microsoft::Office::Interop::Excel::Application();
     // Обчислення детермінанта матриці A
     double визначник_A = XL1->Application->WorksheetFunction
          ->MDeterm(A);
     // Якщо визначник_A != 0, то вихід з процедури:
     if (визначник_A == 0)
     {
          MessageBox::Show("Система не має рішення, оскільки\n" +
               "визначник дорівнює нулю", "Немає рішення",
          MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
          return;
     }
     // Отримання зворотної матриці зворотна_A:
     Object ^ зворотна_A = XL1->Application->WorksheetFunction
          ->MInverse(A);
     // Функція Transpose перетворює рядок вільних
     // членів у вектор:
     Object ^ вектор_B = XL1->Application->WorksheetFunction
          ->Transpose(B);
     // Множення матриці на вектор вільних членів:
     Object ^ X =XL1->Application->WorksheetFunction
          ->MMult(зворотна_A, вектор_B);
     // Щоб відповідь набула індексованих властивостей,
     // перетворимо її в масив:
     array<Object^, 2> ^ рішення_X = (array<Object^, 2> ^)X;
```

```
        // Отримуємо двовимірний масив, індекси якого
        // починаються з одиниці:
        MessageBox::Show("Невідомі дорівнюють: x1=" + рішення_X[1, 1]
            + " x2=" + рішення_X[2, 1] + " x3=" + рішення_X[3, 1]);
}

private: System::Void MyForm_Load(System::Object^  sender,
System::EventArgs^  e) {
        label1->Text = "x1 + 2*x2 + x3 = -1";
        label2->Text = "3*x1 - x2 - x3 = -1";
        label3->Text = "-2*x1 + 2*x2 + 3*x3 = 5";
        button1->Text = "Рішення";
        Text = "Система лінійних алгебраїчних рівнянь";
}
```

Текст програми на C# з доданими рядками для впорядкування об’єктів Lable та кнопки Button в межах форми, а також розміру форми наведений нижче.

Приклад коду C#

```
private void Form1_Load(object sender, EventArgs e)
{
        label1.Text = "x1 + 2*x2 + x3 = -1";
        label2.Text = "3*x1 - x2 - x3 = -1";
        label3.Text = "-2*x1 + 2*x2 + 3*x3 = 5";
        button1.Text = "Рішення";
        Text = "Система лінійних алгебраїчних рівнянь";
        // Встановлюємо положення об'єктів label відносно
        //лівого краю вікна
        label1.Left = 20;
        label2.Left = 20;
        label3.Left = 20;
        // Встановлюємо положення об'єктів label відносно
        //верхнього краю вікна
        label1.Top = label1.Height * 2;
        label2.Top = label1.Height * 4;
        label3.Top = label1.Height * 6;
        // Вирівнюємо положення кнопки
        //вздовж нижнього краю вікна
        button1.Dock = DockStyle.Bottom;
        // Задаємо висоту вікна з урахування висоти
        //та положення розташованих в ньому об'єктів
        this.Height = label1.Height * 6 + button1.Height + 100;
        // Задаємо ширину вікна
        this.Width = 400;
}

private void Button1_Click(object sender, EventArgs e)
{
```

```
        // Вирішуємо систему
        // x1 + 2*x2 + x3 = -1
        // 3*x1 - x2 - x3 = -1
        // -2*x1 + 2*x2 + 3*x3 = 5
        // Для цієї системи пряма матриця буде мати вигляд:
        // double[,]  A = double[n, n];
        double[,] A = {{ 1, 2, 1 },
                       { 3, -1, -1 },
                       { -2, 2, 3 }
        };
        // Рядок вільних членів:
        // double[] B = double[n];
        // Вільні члени:
        double[] B = { -1, -1, 5 };
        // Створення екземпляра класу Excel.Application:
        var XL1 = new Microsoft.Office.Interop.Excel.Application();
        // Обчислення детермінанта матриці A
        double визначник_A = XL1.Application.WorksheetFunction.MDeterm(A);
        // Якщо визначник_A != 0, то вихід з процедури:
        if (визначник_A == 0)
        {
            MessageBox.Show("Система не має рішення, оскільки\n" +
                "визначник дорівнює нулю", "Немає рішення",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            return;
        }
        // Отримання зворотної матриці зворотна_A:
        Object  зворотна_A = XL1.Application.WorksheetFunction.MInverse(A);
        // Функція Transpose перетворює рядок вільних
        // членів у вектор:
        Object  вектор_B = XL1.Application.WorksheetFunction.Transpose(B);
        // Множення матриці на вектор вільних членів:
        Object  X =
            XL1.Application.WorksheetFunction.MMult(зворотна_A, вектор_B);
        // Щоб відповідь набула індексованих властивостей,
        // перетворимо її в масив:
        Object[,] рішення_X = (Object[,])X;
        // Отримуємо двовимірний масив, індекси якого
        // починаються з одиниці:
        MessageBox.Show("Невідомі дорівнюють: x1=" + рішення_X[1, 1]
            + " x2=" + рішення_X[2, 1] + " x3=" + рішення_X[3, 1]);
}
```

Як видно з тексту програми, задаючи пряму матрицю, значення коефіцієнтів присвоюємо відразу при оголошенні двовимірного масиву. Аналогічно поступаємо з рядком (одновимірним масивом) вільних членів. Згідно з вимогою об'єкта `WorksheetFunction` результуючі зворотна матриця і вектор невідомих повинні бути оголошені як об'єктні змінні (типу `Object`). Спочатку обчислюємо визначник (детермінант) прямої матриці, використовуючи функцію

MS Excel MDeterm(). Якщо пряма матриця погано обумовлена, тобто визначник дорівнює 0, то виходимо з процедури і повідомляємо користувачеві в діалоговому вікні MessageBox, що система не має рішення (рис. 9.6).

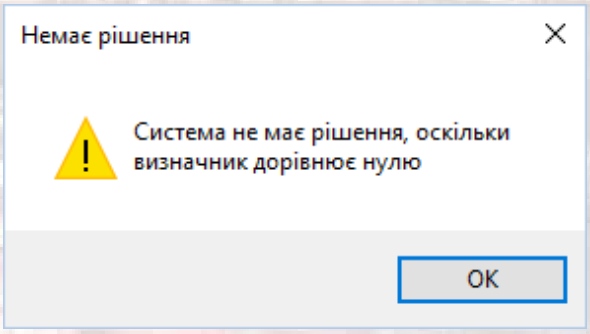


Рисунок 9.6 – Повідомлення про відсутність рішення

Якщо визначник матриці не дорівнює 0, то з допомогою функції MS Excel MInverse() знаходимо обернену матрицю. Тепер рядок (одновимірний масив) вільних членів слід перетворити у вектор, для цієї мети використовуємо функцію MS Excel Transpose() (тобто транспонувати масив B). Далі обернену матрицю за допомогою функції MS Excel MMult() множимо на вектор вільних членів. У результаті функція MMult() повертає двовимірний масив, індекси якого починаються з одиниці (але не з нуля). Наступним оператором форматуємо відповідь у діалоговому вікні MessageBox.

Результат роботи програми наведений на рис. 9.7.

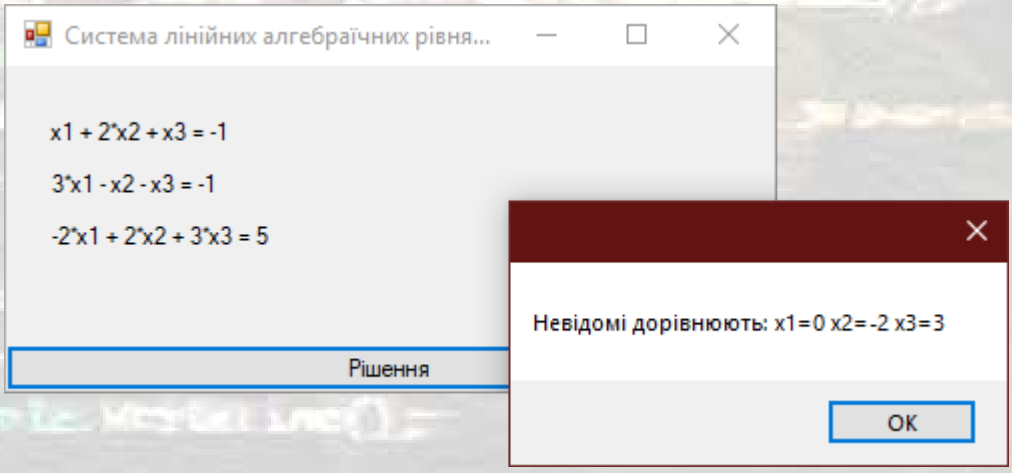


Рисунок 9.7 – Результат роботи програми з використанням математичних функцій MS Excel

Як бачимо, досить складні завдання можна вирішувати за допомогою коротенької програми завдяки зверненню до функцій MS Excel. Причому на комп'ютері, де буде працювати ця програма, зовсім не обов'язково повинен бути інстальований MS Excel. Однак інсталяційний пакет вашої програми повинен містити відповідну dll-бібліотеку функцій Microsoft Excel (файл Interop.Microsoft.Office.Interop.Excel.1.9.dll).

Повний текст даного програмного модуля наведений у лістингу 9.7 (C++/CLI) та лістингах 9.8-9.9 (C#).

9.4 Побудова діаграм засобами MS Excel

Завдання 4.

Необхідно створити проект, який дозволяє користувачеві будувати діаграми у програмі з використанням засобів (об'єктів компонентної бібліотеки) MS Excel.

Для вирішення цієї задачі запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо додаток шаблону Windows Forms. У проектовану екранну форму з Панелі елементів (Toolbox) перенесемо:

1) елемент Panel (об'єкт panel1), а в ньому розмістимо елемент DataGridView для відображення табличних даних;
2) елемент Panel (об'єкт panel2), а в ньому розмістимо два елементи Button (об'єкти button1 та button2);
3) елемент Panel (об'єкт panel3), а в ньому розмістимо три елементи Label, три елементи TextBox та одну кнопку Button;
4) два елементи діалогових вікон SaveFileDialog (об'єкти saveFileDialog1 та saveFileDialog2).

Після цього вікно Конструктора (Design) повинно мати наступний вигляд (рис. 9.8).

У поточний проект підключаємо бібліотеку об'єктів MS Excel. Для цього в меню Проект (Project) виберемо команду Додати посилання... (Add Reference...), потім на вкладці COM двічі клацнемо на посиланні Microsoft Excel 16.0 Object Library (рис. 9.5).

Далі додаємо програмний код наведений нижче.

Приклад коду C++/CLI

```
#pragma endregion
            DataTable^ Таблиця;  // Об'явлення объекта "Таблиця"
            DataSet^ НабірДаних; // Об'явлення объекта "НабірДаних"
            DataColumn^ Стовпчик1; // Об'явлення объекта Стовпчик1
            DataColumn^ Стовпчик2; // Об'явлення объекта Стовпчик2

private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {
      // Створюємо екземпляр класу Excel::Application:
      Microsoft::Office::Interop::Excel::Application ^ XL1 = gcnew
            Microsoft::Office::Interop::Excel::Application();
      XL1->Visible = true;
      // Задаємо параметр за замовчуванням для його подальшого
      // використання у відповідних методах:
      Object ^ t = Type::Missing;
      // Створюємо нову книгу MS Excel:
      Microsoft::Office::Interop::Excel::Workbook ^ Книга =
            XL1->Workbooks->Add(t);
      // Об'являємо аркуші в книзі:
```

```
Microsoft::Office::Interop::Excel::Sheets ^ Аркуші =
      Книга->Worksheets;
// Вибираємо перший аркуш:
Microsoft::Office::Interop::Excel::Worksheet ^ Аркуш =
      (Microsoft::Office::Interop::Excel::Worksheet ^)Аркуші
      ->Item[1];
// Якщо хочемо додати ще один аркуш (четвертий) до вже існуючих:
//_Worksheet ^ Аркуш = safe_cast<_Worksheet^>(Аркуші
//->Item[ (Object^)1 ]);
// Записуємо дані у відповідних комірках:
String ^ colA;
String ^ colB;
// Записуємо заголовки стовпчиків на лист Excel
Аркуш->Range["A1", t]->Value2 = Стовпчик1->Caption;
Аркуш->Range["B1", t]->Value2 = Стовпчик2->Caption;
// Записуємо таблицю на лист Excel
for (int i = 0; i < Таблиця->Rows->Count; i++)
{
      colA = "A" + (i+2).ToString();
      colB = "B" + (i+2).ToString();
      Аркуш->Range[colA, t]->Value2 = dataGridView1->Rows[i]->
            Cells[0]->Value;
      Аркуш->Range[colB, t]->Value2 = dataGridView1->Rows[i]->
            Cells[1]->Value;
}
// Замовляємо побудову діаграми (графіка) з параметрами
// за замовчуванням:
Microsoft::Office::Interop::Excel::Chart ^ Графік =
      (Microsoft::Office::Interop::Excel::Chart ^)XL1->Charts
      ->Add(t, t, t, t);
// Задаємо діапазон значень для побудови графіку:
Графік->SetSourceData(Аркуш->Range["A2", colB],
      Microsoft::Office::Interop::Excel::XlRowCol::xlColumns);
// Задаємо тип графіка "стовпчикова діаграма" (гістограма):
Графік->ChartType = Microsoft::Office::Interop::Excel
      ::XlChartType::xlColumnClustered;
Графік->HasLegend = false; // Відключаємо легенду графіка
Графік->HasTitle = true; // Графік має заголовок
Графік->ChartTitle->Caption = Таблиця->TableName;
// Підпис осі X:
Microsoft::Office::Interop::Excel::Axis ^ ГоризонтальнаОсь =
      (Microsoft::Office::Interop::Excel::Axis^)Графік->Axes(
      Microsoft::Office::Interop::Excel::XlAxisType::xlCategory,
      Microsoft::Office::Interop::Excel::XlAxisGroup::xlPrimary);
ГоризонтальнаОсь->HasTitle = true;
ГоризонтальнаОсь->AxisTitle->Text = Стовпчик1->Caption;
// Підпис осі Y:
Microsoft::Office::Interop::Excel::Axis ^ ВертикальнаОсь =
      (Microsoft::Office::Interop::Excel::Axis^)Графік->Axes(
      Microsoft::Office::Interop::Excel::XlAxisType::xlValue,
      Microsoft::Office::Interop::Excel::XlAxisGroup::xlPrimary);
```

```
        ВертикальнаОсь->HasTitle = true;
        ВертикальнаОсь->AxisTitle->Text = Стовпчик2->Caption;
        // Збереження графіка в растровому файлі:
        if (MessageBox::Show("Зберегти діаграму?", "Діаграма",
              MessageBoxButtons::YesNo, MessageBoxIcon::Question) ==
              System::Windows::Forms::DialogResult::Yes)
        {
              if (saveFileDialog2->ShowDialog() ==
                    System::Windows::Forms::DialogResult::OK)
              {
              // Якщо користувач натиснув Зберегти, зберігаємо діаграму
                XL1->ActiveChart->Export(saveFileDialog2->FileName, t, t);
              }
        }
}

private: System::Void MyForm_Load(System::Object^  sender,
System::EventArgs^  e) {
        // Задаємо написи на кнопках
        button3->Text = "Створити таблицю";
        button1->Text = "Створити діаграму Excel";
        button2->Text = "Зберегти таблицю у файл xml";
        // Задаємо заголовок вікна
        Text = "Створення діаграми в MS Excel";
        // Задаємо висоту кнопок
        button1->Height = 25;
        button2->Height = button1->Height;
        button3->Height = button1->Height;
        // Задаємо написи на мітках Lable
        label1->Text = "Назва таблиці";
        label2->Text = "Ім'я 1-го стовпчика";
        label3->Text = "Ім'я 2-го стовпчика";
        // Вирівнюємо кнопки в межах об'єктів Panel
        button1->Dock = DockStyle::Bottom;
        button2->Dock = DockStyle::Bottom;
        button3->Dock = DockStyle::Bottom;
        // Вирівнюємо об'кти Lable та TextBox в межах об'єкту panel3
        label1->Width = label3->Width;
        label2->Width = label3->Width;
        label1->Top = 10; label1->Left = 10;
        label2->Top = (label1->Height + 10) + 10; label2->Left = 10;
        label3->Top = (label1->Height + 10) * 2 + 10; label3->Left = 10;
        textBox1->Top = label1->Top; textBox1->Left = label3->Width + 30;
        textBox2->Top = label2->Top; textBox2->Left = label3->Width + 30;
        textBox3->Top = label3->Top; textBox3->Left = label3->Width + 30;
        textBox1->Width = 110;
        textBox2->Width = textBox1->Width;
        textBox3->Width = textBox1->Width;
        // Задаємо ширину форми з урахуванням ширини
        //об'єктів Lable та TextBox
        Width = 10 + label1->Width + 20 + textBox1->Width + 60;
```

```
        // Задаємо висоту panel3 з урахуванням висоти
        //об'єктів Lable, TextBox, button3
        panel3->Height = (label1->Height + 10) * 3 + button3->Height + 30;
        // Задаємо висоту panel2
        panel2->Height = button1->Height * 2;
        // Вирівнюємо об'єкти Panel в межах форми
        panel2->Dock = DockStyle::Bottom;
        panel3->Dock = DockStyle::Bottom;
        dataGridView1->Dock = DockStyle::Fill;
        panel1->Dock = DockStyle::Fill;
        // Властивіст DockStyle::Fill працює не завжди коректно
        //щоб виправити ситуацію можна скористатись "чарівним оператором",
        //який встановить для проблемного об'єкта індекс 0 властивості
        //SetChildIndex батьківського об'єкту
        this->Controls->SetChildIndex(panel1, 0);
        // Приховуємо панель panel2 з кнопками побудови
        //діаграми та збереження таблиці
        panel2->Visible = false;
        // Встановлюємо фільтр для збереження таблиці у файл XML
        saveFileDialog1->Filter
              = "Файли XML(*.xml)|*.xml|Всі файли (*.*)|*.*";
        // Встановлюємо фільтр для збереження діаграми у файл JPG
        saveFileDialog2->Filter = "Файли зображень JPG(*.jpg)|*.jpg|
              Всі файли (*.*)|*.*";
}

private: System::Void button2_Click(System::Object^  sender,
System::EventArgs^  e) {
        // Зберегти файл *.xml:
        if (saveFileDialog1->ShowDialog()
              == System::Windows::Forms::DialogResult::OK)
        {
              // Якщо користувач натиснув Зберегти, зберігаємо тиблицю
              НаборДаних->WriteXml(saveFileDialog1->FileName);
        }
}

private: System::Void button3_Click(System::Object^  sender,
System::EventArgs^  e) {
        Таблиця = gcnew DataTable();
        НаборДаних = gcnew DataSet();
        // Зчитуємо назви стовпчиків з полів textBox
        Стовпчик1 = Таблиця->Columns->Add(textBox2->Text);
        Стовпчик2 = Таблиця->Columns->Add(textBox3->Text);
        // Зчитуємо назву таблиці з поля textBox1
        Таблиця->TableName = textBox1->Text;
        // Пов'язуємо об'єкт dataGridView1 з Таблицею
        dataGridView1->DataSource = Таблиця;
        // Додати об'єкт Таблиця в DataSet
        НаборДаних->Tables->Add(Таблиця);
        // Відображаємо панель з кнопками
```

```
        panel2->Visible = true;
}
```

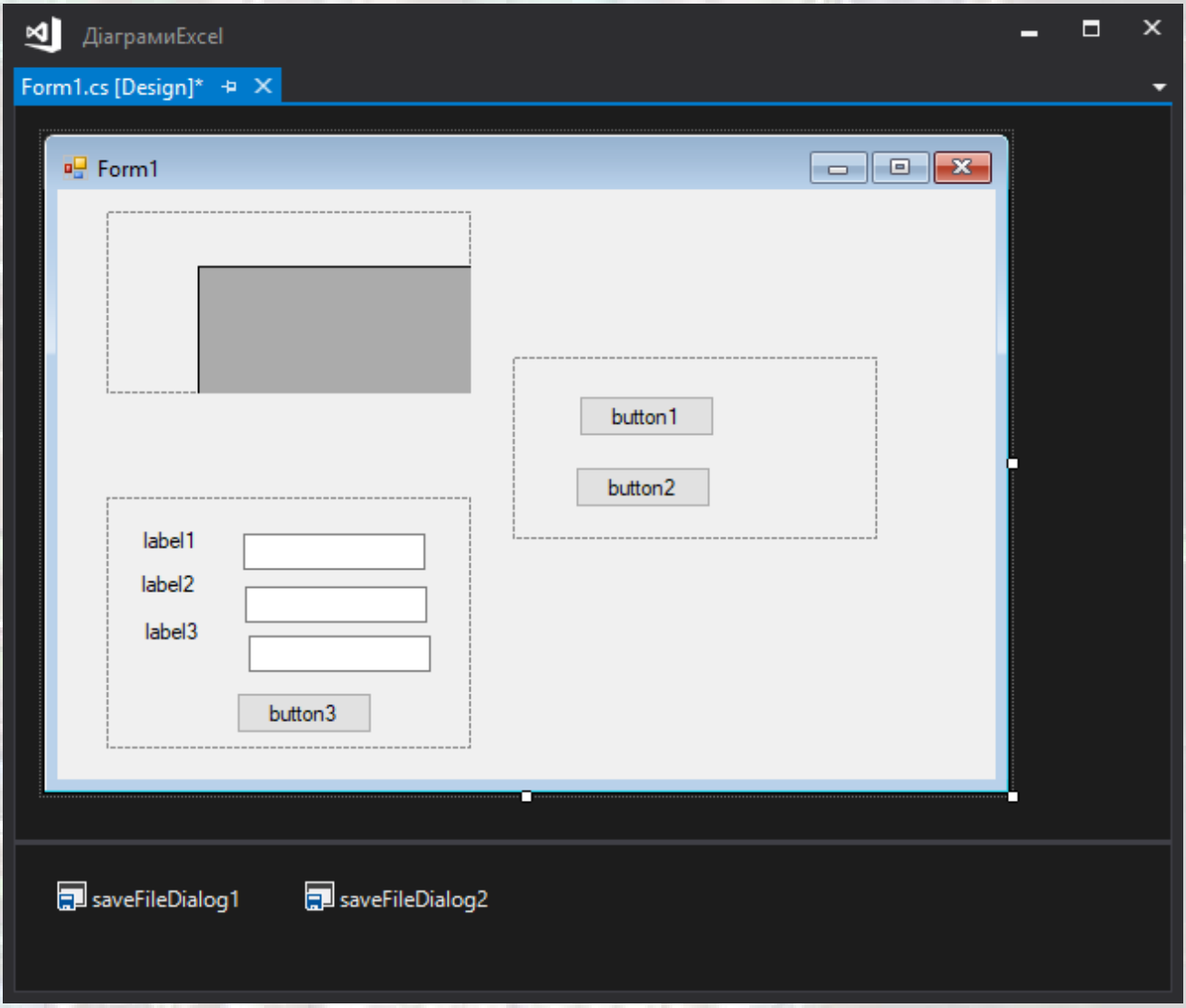


Рисунок 9.8 – Вікно конструктора форми після додавання всіх елементів

Нижче наведенийй приклад коду виконання поставленої задачі на мові C#.

Приклад коду C#

```
public partial class Form1 : Form
{

  DataTable Таблиця;  // Об'явлення объекта "Таблиця"
  DataSet НабірДаних; // Об'явлення объекта "НабірДаних"
  DataColumn Стовпчик1; // Об'явлення объекта Стовпчик1
  DataColumn Стовпчик2; // Об'явлення объекта Стовпчик2

  public Form1()
  {
      InitializeComponent();
  }
```

```
private void Form1_Load(object sender, EventArgs e)
{
    // Задаємо написи на кнопках
    button3.Text = "Створити таблицю";
    button1.Text = "Створити діаграму Excel";
    button2.Text = "Зберегти таблицю у файл xml";
    // Задаємо заголовок вікна
    Text = "Створення діаграми в MS Excel";
    // Задаємо висоту кнопок
    button1.Height = 25;
    button2.Height = button1.Height;
    button3.Height = button1.Height;
    // Задаємо написи на мітках Lable
    label1.Text = "Назва таблиці";
    label2.Text = "Ім'я 1-го стовпчика";
    label3.Text = "Ім'я 2-го стовпчика";
    // Вирівнюємо кнопки в межах об'єктів Panel
    button1.Dock = DockStyle.Bottom;
    button2.Dock = DockStyle.Bottom;
    button3.Dock = DockStyle.Bottom;
    // Вирівнюємо об'кти Lable та TextBox в межах об'єкту panel3
    label1.Width = label3.Width;
    label2.Width = label3.Width;
    label1.Top = 10; label1.Left = 10;
    label2.Top = (label1.Height + 10) + 10; label2.Left = 10;
    label3.Top = (label1.Height + 10) * 2 + 10; label3.Left = 10;
    textBox1.Top = label1.Top; textBox1.Left = label3.Width + 30;
    textBox2.Top = label2.Top; textBox2.Left = label3.Width + 30;
    textBox3.Top = label3.Top; textBox3.Left = label3.Width + 30;
    textBox1.Width = 110;
    textBox2.Width = textBox1.Width;
    textBox3.Width = textBox1.Width;
    // Задаємо ширину форми з урахуванням ширини
    //об'єктів Lable та TextBox
    Width = 10 + label1.Width + 20 + textBox1.Width + 60;
    // Задаємо висоту panel3 з урахуванням висоти
    //об'єктів Lable, TextBox, button3
    panel3.Height = (label1.Height + 10) * 3 + button3.Height + 30;
    // Задаємо висоту panel2
    panel2.Height = button1.Height * 2;
    // Вирівнюємо об'єкти Panel в межах форми
    panel2.Dock = DockStyle.Bottom;
    panel3.Dock = DockStyle.Bottom;
    dataGridView1.Dock = DockStyle.Fill;
    panel1.Dock = DockStyle.Fill;
    // Властивіст DockStyle.Fill працює не завжди коректно
    //щоб виправити ситуацію можна скористатись "чарівним оператором",
    //який встановить для проблемного об'єкта індекс 0 властивості
    //SetChildIndex батьківського об'єкту
    this.Controls.SetChildIndex(panel1, 0);
```

```
        // Приховуємо панель panel2 з кнопками побудови діаграми
        //та збереження таблиці
        panel2.Visible = false;
        // Встановлюємо фільтр для збереження таблиці у файл XML
        saveFileDialog1.Filter
             = "Файли XML(*.xml)|*.xml|Всі файли (*.*)|*.*";
        // Встановлюємо фільтр для збереження діаграми у файл JPG
        saveFileDialog2.Filter
             = "Файли зображень JPG(*.jpg)|*.jpg|Всі файли(*.*)| *.* ";
    }

    private void Button1_Click(object sender, EventArgs e)
    {
        // Створюємо екземпляр класу Excel.Application:
        Microsoft.Office.Interop.Excel.Application XL1 = new
             Microsoft.Office.Interop.Excel.Application
        {
             Visible = true
        };
        // Задаємо параметр за замовчуванням для його подальшого
        // використання у відповідних методах:
        Object  t = Type.Missing;
        // Створюємо нову книгу MS Excel:
        Microsoft.Office.Interop.Excel.Workbook  Книга
             = XL1.Workbooks.Add(t);
        // Об'являємо аркуші в книзі:
        Microsoft.Office.Interop.Excel.Sheets  Аркуші = Книга.Worksheets;
        // Вибираємо перший аркуш:
        Microsoft.Office.Interop.Excel.Worksheet  Аркуш =
             (Microsoft.Office.Interop.Excel.Worksheet )Аркуші.Item[1];
        // Якщо хочемо додати ще один аркуш (четвертий) до вже існуючих:
        // _Worksheet  Аркуш
        //= safe_cast<_Worksheet^>(Аркуші.Item[ (Object^)1 ]);
        // Записуємо дані у відповідних комірках:
        string  colA="";
        string  colB="";
        // Записуємо заголовки стовпчиків на лист Excel
        Аркуш.Range["A1", t].Value2 = Стовпчик1.Caption;
        Аркуш.Range["B1", t].Value2 = Стовпчик2.Caption;
        // Записуємо таблицю на лист Excel
        for (int i = 0; i < Таблиця.Rows.Count; i++)
        {
             colA = "A" + (i + 2).ToString();
             colB = "B" + (i + 2).ToString();
             Аркуш.Range[colA, t].Value2
                  = dataGridView1.Rows[i].Cells[0].Value;
             Аркуш.Range[colB, t].Value2
                  = dataGridView1.Rows[i].Cells[1].Value;
        }
        // Замовляємо побудову діаграми (графіка) з параметрами
        //за замовчуванням:
```

```
        Microsoft.Office.Interop.Excel.Chart  Графік =
         (Microsoft.Office.Interop.Excel.Chart )XL1.Charts.Add(t, t, t, t);
        // Задаємо діапазон значень для побудови графіку:
        Графік.SetSourceData(Аркуш.Range["A2", colB],
              Microsoft.Office.Interop.Excel.XlRowCol.xlColumns);
        // Задаємо тип графіка "стовпчикова діаграма" (гістограма):
        Графік.ChartType =
              Microsoft.Office.Interop.Excel.XlChartType.xlColumnClustered;
        Графік.HasLegend = false; // Відключаємо легенду графіка
        Графік.HasTitle = true; // Графік має заголовок
        Графік.ChartTitle.Caption = Таблиця.TableName;
        // Підпис осі X:
        Microsoft.Office.Interop.Excel.Axis  ГоризонтальнаОсь =
              (Microsoft.Office.Interop.Excel.Axis )Графік.Axes(
              Microsoft.Office.Interop.Excel.XlAxisType.xlCategory,
              Microsoft.Office.Interop.Excel.XlAxisGroup.xlPrimary);
        ГоризонтальнаОсь.HasTitle = true;
        ГоризонтальнаОсь.AxisTitle.Text = Стовпчик1.Caption;
        // Підпис осі Y:
        Microsoft.Office.Interop.Excel.Axis  ВертикальнаОсь =
              (Microsoft.Office.Interop.Excel.Axis )Графік.Axes(
              Microsoft.Office.Interop.Excel.XlAxisType.xlValue,
              Microsoft.Office.Interop.Excel.XlAxisGroup.xlPrimary);
        ВертикальнаОсь.HasTitle = true;
        ВертикальнаОсь.AxisTitle.Text = Стовпчик2.Caption;
        // Збереження графіка в растровому файлі:
        if (MessageBox.Show("Зберегти діаграму?", "Діаграма",
              MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
              System.Windows.Forms.DialogResult.Yes)
        {
              if (saveFileDialog2.ShowDialog() ==
                    System.Windows.Forms.DialogResult.OK)
              {
                    // Якщо користувач натиснув Зберегти,
                    //зберігаємо діаграму
                    XL1.ActiveChart.Export(saveFileDialog2.FileName, t, t);
              }
        }
  }

  private void Button2_Click(object sender, EventArgs e)
  {
  // Зберегти файл *.xml:
        if (saveFileDialog1.ShowDialog()
              == System.Windows.Forms.DialogResult.OK)
        {
              // Якщо користувач натиснув Зберегти, зберігаємо тиблицю
              НаборДаних.WriteXml(saveFileDialog1.FileName);
        }
  }
```

```
private void Button3_Click(object sender, EventArgs e)
{
    Таблиця = new DataTable();
    НабірДаних = new DataSet();
    // Зчитуємо назви стовпчиків з полів textBox
    Стовпчик1 = Таблиця.Columns.Add(textBox2.Text);
    Стовпчик2 = Таблиця.Columns.Add(textBox3.Text);
    // Зчитуємо назву таблиці з поля textBox1
    Таблиця.TableName = textBox1.Text;
    // Пов'язуємо об'єкт dataGridView1 з Таблицею
    dataGridView1.DataSource = Таблиця;
    // Додати об'єкт Таблиця в DataSet
    НабірДаних.Tables.Add(Таблиця);
    // Відображаємо панель з кнопками
    panel2.Visible = true;
    }
}
```

Код даного програмного модуля досить об'ємний, що пов'язане з програмуванням вирівнювання елементів інтерфейсу, та певною універсальністю даної програми. Програма пропоноє користувачу задати ім'я таблиці та імена двох стовпчиків даних. Після цього за допомогою натискання кнопки створюється таблиця даних з двома стовпцями, в які можна вводити довільні дані з будь-якою кількістю рядків. На основі введених двох стовпчиків даних і буде виконуватись побудова діаграмив MS Excel.

У обробнику події Form1_Load присвоюються значення різним властивостям об'єктів форми. Більшість цих властивостей зручніше задавати в режимі проектування у вікні Властивості (Properties) для відповідного об'єкту. Також у обробнику події Form1_Load деякі властивості задаються у вигляді залежностей для досягнення ідеального розташування та розмірів елементів інтерфейсу. Слід звернути увагу, що при встановленні властивості Dock у значення Fill (заповнення) для об'єктів, ця властивість часто працює некоректно і займає площу і інших вже вирівняних об'єктів. Щоб об'єкти інтерфейсу після вирівнювання не перекривали один одного, потрібно додати наступний рядок коду:

```
this.Controls.SetChildIndex(panel1, 0);
```

де this – це батьківський об'єкт-контейнер в межах якого вирівнюються інші об'єкти (в даному випадку це форма); SetChildIndex() – функція, в яку в якості першого параметру передаємо ім'я об'єкту, який має властивість Dock зі значенням Fill (заповнення), в нашому випадку це panel1.

Після запуску програми ми побачимо вікно форми, в якому ще немає таблиці. Відповідно немає даних, за якими потрібно будувати діаграму, або зберігати таблицю у файл XML. За ці дії в нас відповідають дві кнопки, які

розташовані на панелі panel2. Тому при запуску форми ми приховали цю панель з двома кнопками (рис. 9.9).

```
// Приховуємо панель panel2 з кнопками побудови діаграми та збереження таблиці
panel2.Visible = false;
```

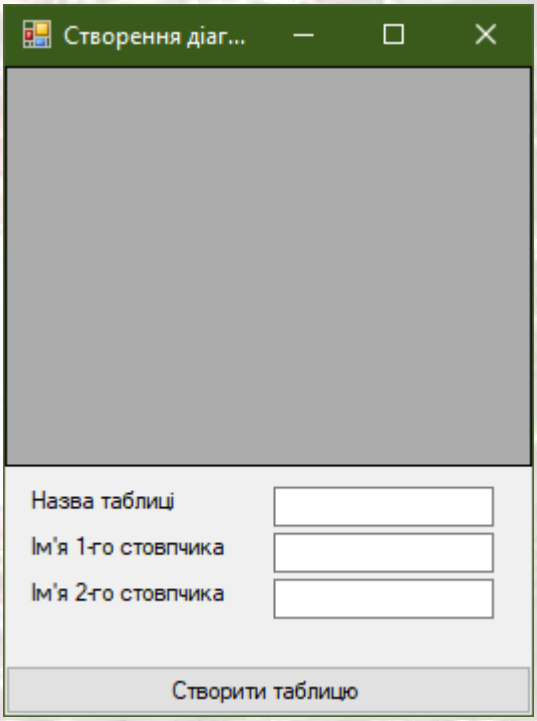


Рисунок 9.9 – Вікно програми після запуску з введеними назвами стовпчиків та таблиці

Коли користувач введе імена стовпчиків та назву таблиці і натисне кнопку Створити таблицю (обробник події button3_Click), у вікні з'явиться пуста таблиця, а також панель з двома кнопками. В таблицю потрібно ввести дані за якими і будуватимемо діаграму (рис. 9.10).

При натисканні кнопки Зберегти таблицю у файл XML вікликається обробник події button2_Click, в якому передбачене виведення стандартного діалогового вікна збереження файлу для можливості обрання місця збереження та імені файлу користувачем (рис. 9.11).

Приклад коду C++/CLI

```
// Зберегти файл *.xml:
if (saveFileDialog1->ShowDialog() ==
System::Windows::Forms::DialogResult::OK)
{
    // Якщо користувач натиснув Зберегти, зберігаємо тиблицю
    НаборДаних->WriteXml(saveFileDialog1->FileName);
}
```

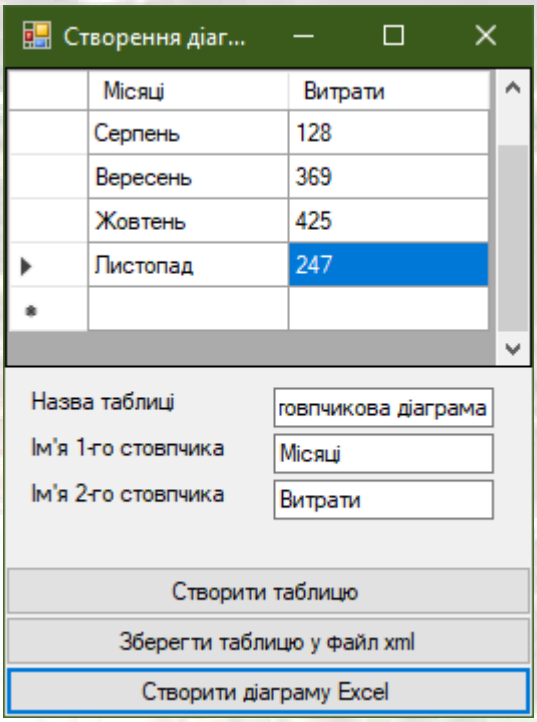


Рисунок 9.10 – Вікно програми після введення таблиці даних

Приклад коду C#

```
// Зберегти файл *.xml:
if (saveFileDialog1.ShowDialog()
      == System.Windows.Forms.DialogResult.OK)
{
      // Якщо користувач натиснув Зберегти, зберігаємо тиблицю
      НаборДаних.WriteXml(saveFileDialog1.FileName);
}
```

За діалогове вікно збереження файлу відповідає об'єкт `saveFileDialog1`, в якому встановлене значення фільтру для відображення за замовчуванням лише файлів XML.

Після збереження таблиці даних, їх можна перегрянути у сторонній програмі, наприклад, у інтернет браузері.

Кнопка `Створити діаграму Excel` безпосередньо відповідає за створення діаграми (обробник події `button1_Click`). Для можливості створення діаграми ми спочатку записуємо нашу таблицю на аркуш `Excel`. За це відповідає наступна частина коду.

Приклад коду C++/CLI

```
// Створюємо екземпляр класу Excel::Application:
Microsoft::Office::Interop::Excel::Application ^ XL1 = gcnew
      Microsoft::Office::Interop::Excel::Application();
// Відображення вікна Excel
XL1->Visible = true;
// Задаємо параметр за замовчуванням для його подальшого
```

```
// використання у відповідних методах:
Object ^ t = Type::Missing;
// Створюємо нову книгу MS Excel:
Microsoft::Office::Interop::Excel::Workbook ^ Книга = XL1->Workbooks-
>Add(t);
// Об'являємо аркуші в книзі:
Microsoft::Office::Interop::Excel::Sheets ^ Аркуші = Книга->Worksheets;
// Вибираємо перший аркуш:
Microsoft::Office::Interop::Excel::Worksheet ^ Аркуш =
     (Microsoft::Office::Interop::Excel::Worksheet ^)Аркуші->Item[1];
// Якщо хочемо додати ще один аркуш до вже існуючих:
// _Worksheet ^ Аркуш = safe_cast<_Worksheet^>(Аркуші->Item[ (Object^)1
]);
// Записуємо дані у відповідних комірках:
String ^ colA;
String ^ colB;
// Записуємо заголовки стовпчиків на лист Excel
Аркуш->Range["A1", t]->Value2 = Стовпчик1->Caption;
Аркуш->Range["B1", t]->Value2 = Стовпчик2->Caption;
// Записуємо таблицю на лист Excel
for (int i = 0; i < Таблиця->Rows->Count; i++)
{
     colA = "A" + (i+2).ToString();
     colB = "B" + (i+2).ToString();
     Аркуш->Range[colA, t]->Value2 = dataGridView1->Rows[i]
          ->Cells[0]->Value;
     Аркуш->Range[colB, t]->Value2 = dataGridView1->Rows[i]
          ->Cells[1]->Value;
}
```

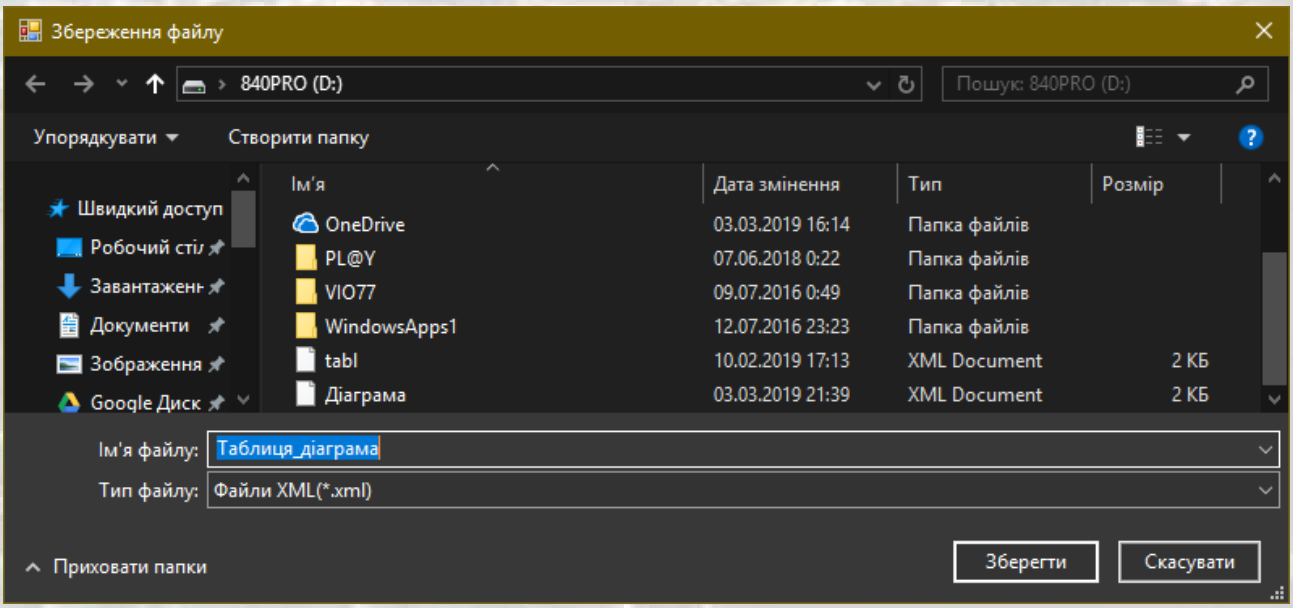


Рисунок 9.11 – Діалогове вікно збереження таблиці у файл XML

Приклад коду C#

```
// Створюємо екземпляр класу Excel.Application:
```

```
Microsoft.Office.Interop.Excel.Application XL1 = new
     Microsoft.Office.Interop.Excel.Application
{
     Visible = true
};
// Задаємо параметр за замовчуванням для його подальшого
// використання у відповідних методах:
Object  t = Type.Missing;
// Створюємо нову книгу MS Excel:
Microsoft.Office.Interop.Excel.Workbook  Книга
     = XL1.Workbooks.Add(t);
// Об'являємо аркуші в книзі:
Microsoft.Office.Interop.Excel.Sheets  Аркуші = Книга.Worksheets;
// Вибираємо перший аркуш:
Microsoft.Office.Interop.Excel.Worksheet  Аркуш =
     (Microsoft.Office.Interop.Excel.Worksheet )Аркуші.Item[1];
// Якщо хочемо додати ще один аркуш (четвертий) до вже існуючих:
// _Worksheet  Аркуш
//= safe_cast<_Worksheet^>(Аркуші.Item[ (Object^)1 ]);
// Записуємо дані у відповідних комірках:
string  colA="";
string  colB="";
// Записуємо заголовки стовпчиків на лист Excel
Аркуш.Range["A1", t].Value2 = Стовпчик1.Caption;
Аркуш.Range["B1", t].Value2 = Стовпчик2.Caption;
// Записуємо таблицю на лист Excel
for (int i = 0; i < Таблиця.Rows.Count; i++)
{
     colA = "A" + (i + 2).ToString();
     colB = "B" + (i + 2).ToString();
     Аркуш.Range[colA, t].Value2
          = dataGridView1.Rows[i].Cells[0].Value;
     Аркуш.Range[colB, t].Value2
          = dataGridView1.Rows[i].Cells[1].Value;
}
```

При цьому ми робимо видимим вікно MS Excel і безпосередньо спостерігаємо заповнення таблиці на аркуші книги Excel. Після створення таблиці даних починається безпосередньо побудова діаграми.

Приклад коду C++/CLI

```
// Замовляємо побудову діаграми (графіка) з параметрами за замовчуванням:
Microsoft::Office::Interop::Excel::Chart ^ Графік =
 (Microsoft::Office::Interop::Excel::Chart^)XL1->Charts->Add(t, t, t, t);
// Задаємо діапазон значень для побудови графіку:
Графік->SetSourceData(Аркуш->Range["A2", colB],
     Microsoft::Office::Interop::Excel::XlRowCol::xlColumns);
// Задаємо тип графіка "стовпчикова діаграма" (гістограма):
Графік->ChartType =
```

```
        Microsoft::Office::Interop::Excel::XlChartType::xlColumnClustered;
// Відключаємо легенду графіка:
Графік->HasLegend = false;
// Графік має заголовок:
Графік->HasTitle = true;
Графік->ChartTitle->Caption = Таблиця->TableName;
// Підпис осі X:
Microsoft::Office::Interop::Excel::Axis ^ ГоризонтальнаОсь =
     (Microsoft::Office::Interop::Excel::Axis^)Графік->Axes(
     Microsoft::Office::Interop::Excel::XlAxisType::xlCategory,
     Microsoft::Office::Interop::Excel::XlAxisGroup::xlPrimary);
ГоризонтальнаОсь->HasTitle = true;
ГоризонтальнаОсь->AxisTitle->Text = Стовпчик1->Caption;
// Підпис осі Y:
Microsoft::Office::Interop::Excel::Axis ^ ВертикальнаОсь =
     (Microsoft::Office::Interop::Excel::Axis^)Графік->Axes(
     Microsoft::Office::Interop::Excel::XlAxisType::xlValue,
     Microsoft::Office::Interop::Excel::XlAxisGroup::xlPrimary);
ВертикальнаОсь->HasTitle = true;
ВертикальнаОсь->AxisTitle->Text = Стовпчик2->Caption;
```

Приклад коду C#

```
// Замовляємо побудову діаграми (графіка) з параметрами
//за замовчуванням:
Microsoft.Office.Interop.Excel.Chart  Графік =
 (Microsoft.Office.Interop.Excel.Chart )XL1.Charts.Add(t, t, t, t);
// Задаємо діапазон значень для побудови графіку:
Графік.SetSourceData(Аркуш.Range["A2", colB],
     Microsoft.Office.Interop.Excel.XlRowCol.xlColumns);
// Задаємо тип графіка "стовпчикова діаграма" (гістограма):
Графік.ChartType =
     Microsoft.Office.Interop.Excel.XlChartType.xlColumnClustered;
Графік.HasLegend = false; // Відключаємо легенду графіка
Графік.HasTitle = true; // Графік має заголовок
Графік.ChartTitle.Caption = Таблиця.TableName;
// Підпис осі X:
Microsoft.Office.Interop.Excel.Axis  ГоризонтальнаОсь =
     (Microsoft.Office.Interop.Excel.Axis )Графік.Axes(
     Microsoft.Office.Interop.Excel.XlAxisType.xlCategory,
     Microsoft.Office.Interop.Excel.XlAxisGroup.xlPrimary);
ГоризонтальнаОсь.HasTitle = true;
ГоризонтальнаОсь.AxisTitle.Text = Стовпчик1.Caption;
// Підпис осі Y:
Microsoft.Office.Interop.Excel.Axis  ВертикальнаОсь =
     (Microsoft.Office.Interop.Excel.Axis )Графік.Axes(
     Microsoft.Office.Interop.Excel.XlAxisType.xlValue,
     Microsoft.Office.Interop.Excel.XlAxisGroup.xlPrimary);
ВертикальнаОсь.HasTitle = true;
ВертикальнаОсь.AxisTitle.Text = Стовпчик2.Caption;
```

Результатом буде поява стовпчикової діаграми у вікні MS Excel з заголовком та підписами вісей (рис. 9.12).

Зрозуміло, що тип діаграми можна задавати і інший у програмному коді. При цьому різновидів діаграм набагато більше, ніж у об'єкті Chart мови C++/CLI.

У програмі також передбачена можливість збереження отриманої у MS Excel діаграми у вигляді графічного файлу з використанням діалогового вікна saveFileDialog2. За це від повідає наступний код.

Приклад коду C++/CLI

```
// Збереження графіка в растровому файлі:
if (MessageBox::Show("Зберегти діаграму?", "Діаграма",
      MessageBoxButtons::YesNo, MessageBoxIcon::Question) ==
      System::Windows::Forms::DialogResult::Yes)
{
      if (saveFileDialog2->ShowDialog() ==
            System::Windows::Forms::DialogResult::OK)
      {
            // Якщо користувач натиснув Зберегти, зберігаємо діаграму
            XL1->ActiveChart->Export(saveFileDialog2->FileName, t, t);
      }
}
```

Приклад коду C#

```
// Збереження графіка в растровому файлі:
if (MessageBox.Show("Зберегти діаграму?", "Діаграма",
      MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
      System.Windows.Forms.DialogResult.Yes)
{
      if (saveFileDialog2.ShowDialog() ==
            System.Windows.Forms.DialogResult.OK)
      {
            // Якщо користувач натиснув Зберегти,
            //зберігаємо діаграму
            XL1.ActiveChart.Export(saveFileDialog2.FileName, t, t);
      }
}
```

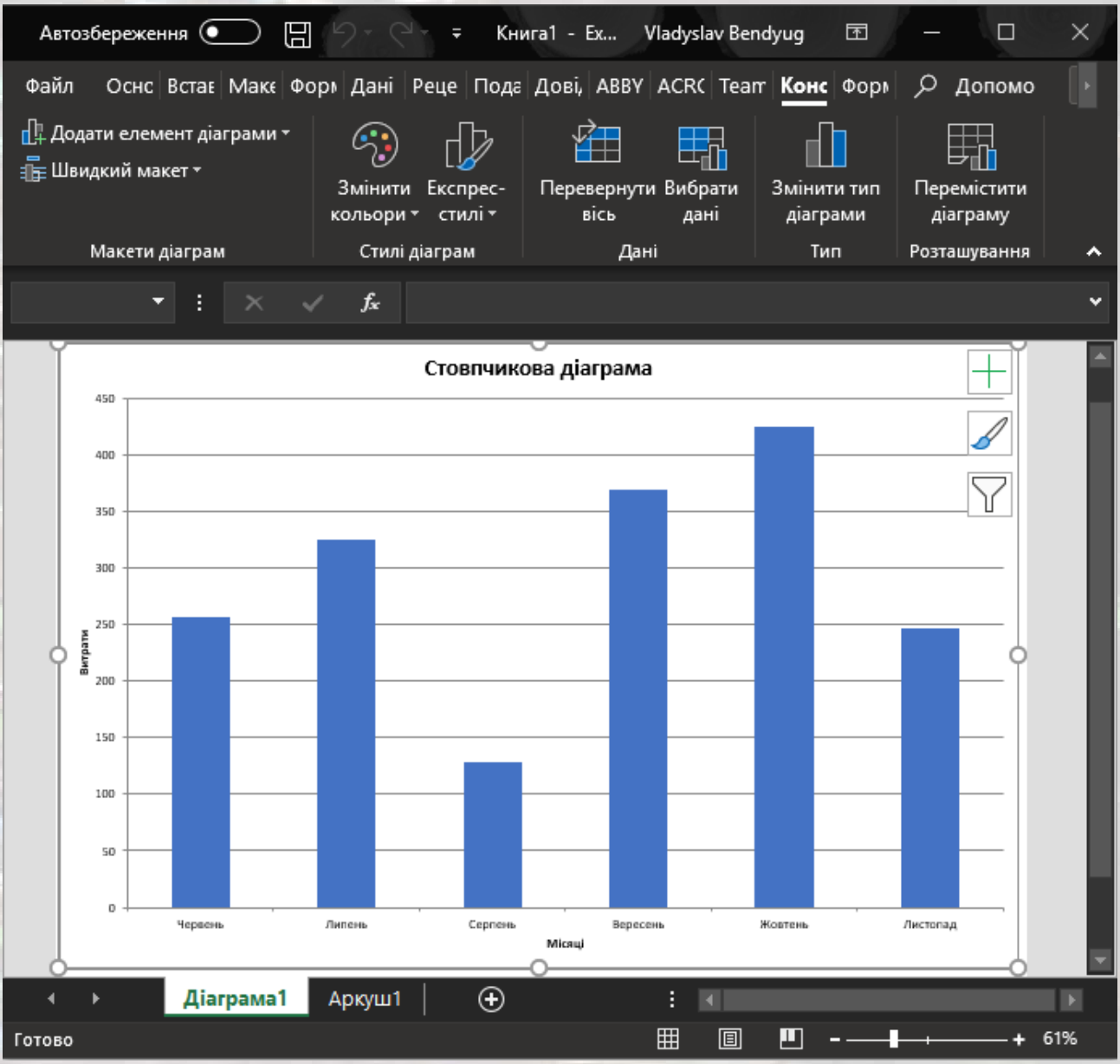


Рисунок 9.12 – Побудована діаграма у вікні MS Excel

Після побудови діаграми на екрані з’явиться діалогове вікно з запитом на збереження діаграми, створене за допомогою функції MessageBox (рис. 9.13).

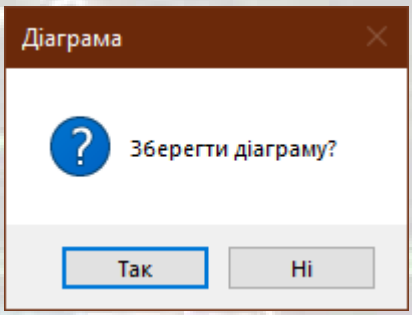


Рисунок 9.13 – Запит на збереження діаграми

Якщо користувач натисне Так, то з'явиться стандартне діалогове вікно збереження файлу (об'єкт saveFileDialog2), в якому потрібно вказати ім'я та розташування файлу JPG, в якому буде збережено створену діаграму.

В даному прикладі ми зробили вікно MS Excel видимим і спостерігали за виконанням операцій. Після цього книгу MS Excel можна зберегти, а також змінювати параметри діаграми вже безпосередньо у вікні MS Excel. Проте, якщо додаткові операції в MS Excel нам не потрібні, ми могли б, як і у попередніх прикладах не показувати вікно MS Excel, а створити в ньому діаграму, експортувати її у растрове зображення, яке б відкрили у об'єкті PictureBox форми. Тоді б все виглядало так, ніби всі дії виконані безпосередньо в нашій програмі. При цьому наша програма отримала потужний інструментарій створення діаграм MS Excel.

Повний текст даного програмного модуля наведений у лістингу 9.10 (C++/CLI) та лістингах 9.11-9.12 (C#).

Програмний код

Лістинг 9.1

Приклад коду C++/CLI

```
// Програма дозволяє користувачеві ввести будь-які слова, речення
// в текстове поле і після натиснення відповідної кнопки перевірити
// орфографію введеного тексту. Для безпосередньої перевірки орфографії
// скористаємося функцією CheckSpelling об'єктної бібліотеки MS Word

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  button1;
	protected:
	private: System::Windows::Forms::Panel^  panel1;
	private: System::Windows::Forms::RichTextBox^  richTextBox1;

	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
```

```
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->panel1 = (gcnew System::Windows::Forms::Panel());
			this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());
			this->panel1->SuspendLayout();
			this->SuspendLayout();
			//
			// button1
			//
			this->button1->Dock = System::Windows::Forms::DockStyle::Bottom;
			this->button1->Location = System::Drawing::Point(0, 238);
			this->button1->Name = L"button1";
			this->button1->Size = System::Drawing::Size(284, 23);
			this->button1->TabIndex = 1;
			this->button1->Text = L"button1";
			this->button1->UseVisualStyleBackColor = true;
			this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click_1);
			//
			// panel1
			//
			this->panel1->Controls->Add(this->richTextBox1);
			this->panel1->Dock = System::Windows::Forms::DockStyle::Fill;
			this->panel1->Location = System::Drawing::Point(0, 0);
			this->panel1->Name = L"panel1";
			this->panel1->Size = System::Drawing::Size(284, 238);
			this->panel1->TabIndex = 2;
			//
			// richTextBox1
			//
			this->richTextBox1->Dock = System::Windows::Forms::DockStyle::Fill;
			this->richTextBox1->Location = System::Drawing::Point(0, 0);
			this->richTextBox1->Name = L"richTextBox1";
			this->richTextBox1->Size = System::Drawing::Size(284, 238);
			this->richTextBox1->TabIndex = 1;
			this->richTextBox1->Text = L"";
			//
			// MyForm
			//
			this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
			this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
			this->ClientSize = System::Drawing::Size(284, 261);
			this->Controls->Add(this->panel1);
			this->Controls->Add(this->button1);
			this->Name = L"MyForm";
			this->Text = L"MyForm";
			this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
			this->panel1->ResumeLayout(false);
			this->ResumeLayout(false);

		}
#pragma endregion

private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
	// У пункті меню Project виберемо команду Add Reference.
	// Потім, якщо на вашому комп'ютері встановлений MS Office 2007,
	// То на вкладці COM двічі клацнемо по посиланню
	// На бібліотеку Microsoft Word 12.0 Object Libary.
	richTextBox1->Clear();
	button1->Text = "Перевірка орфографії";
```

```
        richTextBox1->TabIndex = 0;
        button1->TabIndex = 1;
        button1->Dock = System::Windows::Forms::DockStyle::Bottom;
        panel1->Dock = System::Windows::Forms::DockStyle::Fill;
        richTextBox1->Dock = System::Windows::Forms::DockStyle::Fill;
        Text = "Орфографія";
}

private: System::Void button1_Click_1(System::Object^  sender, System::EventArgs^  e) {
        auto Ворд1 = gcnew Microsoft::Office::Interop::Word::Application();
        // Ворд1-> Visible = false;
        // Змінна з "порожнім" значенням:
        System::Object  ^ t = Type::Missing;
        // Відкриваємо новий порожній документ MS Word:
        Ворд1->Documents->Add(t, t, t, t);
        // Копіюємо вміст текстового вікна в документ
        Ворд1->Selection->default = richTextBox1->Text;
        // Для VB і C # було б: Ворд1->Selection->Text
        // Безпосередня перевірка орфографії:
        Ворд1->ActiveDocument->CheckSpelling(t, t, t, t, t, t, t, t, t, t, t, t, t);
        // Копіюємо результат назад в текстове поле
        richTextBox1->Text = Ворд1->Selection->default;
        Object ^ tt = false;
        Ворд1->Documents->Close(tt, t, t);
        // Закрити документ Word без збереження:
        Ворд1->Quit(tt, t, t);
        Ворд1 = nullptr;
}
};
}
```

Лістинг 9.2

Приклад коду C#

```
// Програма дозволяє користувачеві ввести будь-які слова, речення
//Файл Form1.Designer.cs

namespace Орфографія
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.panel1 = new System.Windows.Forms.Panel();
            this.richTextBox1 = new System.Windows.Forms.RichTextBox();
            this.button1 = new System.Windows.Forms.Button();
            this.panel1.SuspendLayout();
            this.SuspendLayout();
            //
            // panel1
            //
            this.panel1.Controls.Add(this.richTextBox1);
            this.panel1.Location = new System.Drawing.Point(418, 84);
            this.panel1.Name = "panel1";
            this.panel1.Size = new System.Drawing.Size(200, 100);
            this.panel1.TabIndex = 0;
            //
            // richTextBox1
            //
            this.richTextBox1.Location = new System.Drawing.Point(59, 33);
            this.richTextBox1.Name = "richTextBox1";
            this.richTextBox1.Size = new System.Drawing.Size(100, 96);
            this.richTextBox1.TabIndex = 0;
            this.richTextBox1.Text = "";
            //
            // button1
            //
```

```
            this.button1.Location = new System.Drawing.Point(477, 312);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(100, 28);
            this.button1.TabIndex = 1;
            this.button1.Text = "button1";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(800, 450);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.panel1);
            this.Name = "Form1";
            this.Text = "Form1";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.panel1.ResumeLayout(false);
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.Panel panel1;
        private System.Windows.Forms.RichTextBox richTextBox1;
        private System.Windows.Forms.Button button1;
    }
}
```

Лістинг 9.3

Приклад коду C#

```
// Програма дозволяє користувачеві ввести будь-які слова, речення
//Файл Form1.cs

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Орфографія
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // У пункті меню Project виберемо команду Add Reference.
            // Потім, якщо на вашому комп'ютері встановлений MS Office 2016,
            // То на вкладці COM двічі клацнемо по посиланню
            // На бібліотеку Microsoft Word 16.0 Object Libary.
            richTextBox1.Clear();
            button1.Text = "Перевірка орфографії";
            richTextBox1.TabIndex = 0;
            button1.TabIndex = 1;
            button1.Dock = System.Windows.Forms.DockStyle.Bottom;
            panel1.Dock = System.Windows.Forms.DockStyle.Fill;
            richTextBox1.Dock = System.Windows.Forms.DockStyle.Fill;
            Text = "Орфографія";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            var Ворд1 = new Microsoft.Office.Interop.Word.Application();
            // Ворд1.Visible = false;
            // Змінна з "порожнім" значенням:
            System.Object t = Type.Missing;
            // Відкриваємо новий порожній документ MS Word:
            Ворд1.Documents.Add(t, t, t, t);
            // Копіюємо вміст текстового вікна в документ
            Ворд1.Selection.Text = richTextBox1.Text;
            // Безпосередня перевірка орфографії:
            Ворд1.ActiveDocument.CheckSpelling(t, t, t, t, t, t, t, t, t, t, t, t);
            // Копіюємо результат назад в текстове поле
            richTextBox1.Text = Ворд1.Selection.Text;
            Object tt = false;
            Ворд1.Documents.Close(tt, t, t);
            // Закрити документ Word без збереження:
            Ворд1.Quit(tt, t, t);
            Ворд1 = null;
        }
```

```
        }
    }
```

Лістинг 9.4

Приклад коду C++/CLI

```
// Програма виведення таблиці засобами MS Word: запускається
// програма, користувач спостерігає, як запускається редактор
// MS Word і автоматично відбувається побудова таблиці

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  button1;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->SuspendLayout();
```

```
                    //
                    // button1
                    //
                    this->button1->Location = System::Drawing::Point(100, 168);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(75, 23);
                    this->button1->TabIndex = 0;
                    this->button1->Text = L"button1";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->button1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->ResumeLayout(false);

             }
#pragma endregion

      private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
             // У меню Project вкажемо команду Add Reference і на
             // Вкладці COM двічі клацнемо по посиланню на
             // Бібліотеку Microsoft Word 16.0 Object Library
             button1->Text = "Пуск"; this->Text = "Побудова таблиці";
      }

      private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
             // Ініціалізуємо два рядкових масиви:
             array <String ^> ^ Імена = { "Андрій (роб)", "Світла (моб)", "Микола (дом)",
"Кафедра (роб)", "Олександр Степанович", "Сергій (дом)", "Тетяна Петрівна", "Таксі",
"Кінотеатр" };
             array <String ^> ^ Телефон = { "274-88-17", "+38 (067) 7030356", "22-345-72",
"204-82-12", "223-67-67 доп 32-67", "570-38-76", "201-72-23", "555", "216-40-22" };
             // Створюємо новий екземпляр класу Word::Application:
             auto Ворд1 = gcnew Microsoft::Office::Interop::Word::Application();
             Ворд1->Visible = true;
             // Змінна з "порожнім" значенням:
             System::Object ^ t = Type::Missing;
             // Відкриваємо новий документ MS Word:
             auto Документ = Ворд1->Documents->Add(t, t, t, t);
             // Вводимо текст в документ MS WORD з поточної позиції:
             Ворд1->Selection->TypeText("ТАБЛИЦЯ ТЕЛЕФОНІВ");
             // Параметр, який вказує чи показувати межі комірок:
             System::Object ^ t1 = Microsoft::Office::Interop::
                    Word::WdDefaultTableBehavior::wdWord9TableBehavior;
             // Параметр, який вказує чи буде додаток Word автоматично
             // змінювати розмір комірок у таблиці для підгонки вмісту:
             System::Object ^ t2 =
Microsoft::Office::Interop::Word::WdAutoFitBehavior::wdAutoFitContent;
             // Створюємо таблицю з 9 рядків і 2 стовпців:
             Ворд1->ActiveDocument->Tables->Add(Ворд1->Selection->Range, 9, 2, t1, t2);
             // Заповнювати комірки таблиці можна так:
             for (int i = 1; i <= 9; i++)
             {
                    Ворд1->ActiveDocument->Tables[1]->Cell(i, 1)->default-
>InsertAfter(Імена[i - 1]);
                    Ворд1->ActiveDocument->Tables[1]->Cell(i, 2)->default-
>InsertAfter(Телефон[i - 1]);
```

```
                    // Програмуючи на C# ми написали б:
                    // Ворд1.ActiveDocument.Tables[1].Cell(i, 2).Range.InsertAfter(Tel[i -
1]);
                }
                // Призначаємо одиниці вимірювання в документі додатка MS Word:
                Object ^ t3 = Microsoft::Office::Interop::Word::WdUnits::wdLine;
                // Параметр, який вказує на дев'ятий рядок у документі MS Word:
                Object ^ рядок9 = 9;
                // Перевести поточну позицію (Selection) за межі таблиці,
                // (В дев'ятий рядок), щоб тут вивести будь-якій текст:
                Ворд1->Selection->MoveDown(t3, рядок9, t);
                // І тут друкуємо наступний текст:
                Ворд1->Selection->TypeText("Який-небудь текст після таблиці");
                // Зберігати документ немає сенсу, але це вирішить користувач:
                // або можна задати безпосередньо шлях та ім'я файлу
                Object ^ ім’яФайла = "D:\\Таблиця.docx";
                // і одразу зберегти
                Ворд1->ActiveDocument->SaveAs(ім’яФайла, t, t, t, t, t, t, t, t, t, t, t, t,
t, t, t);
            }
        };
}
```

Лістинг 9.5

Приклад коду C#

```
// Програма виведення таблиці засобами MS Word: запускається
// програма, користувач спостерігає, як запускається редактор
// MS Word і автоматично відбувається побудова таблиці
//Файл Form1.Designer.cs

namespace ТаблицяWord
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.button1 = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(382, 286);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(75, 23);
            this.button1.TabIndex = 0;
            this.button1.Text = "button1";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.Button1_Click);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(800, 450);
            this.Controls.Add(this.button1);
            this.Name = "Form1";
            this.Text = "Form1";
            this.Load += new System.EventHandler(this.Form1_Load);
```

```
            this.ResumeLayout(false);

        }
        #endregion

        private System.Windows.Forms.Button button1;
    }
}
```

Лістинг 9.6

Приклад коду C#

```
// Програма виведення таблиці засобами MS Word: запускається
// програма, користувач спостерігає, як запускається редактор
// MS Word і автоматично відбувається побудова таблиці
//Файл Form1.cs

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ТаблицяWord
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // У меню Project вкажемо команду Add Reference і на
            // Вкладці COM двічі клацнемо по посиланню на
            // Бібліотеку Microsoft Word 16.0 Object Library
            button1.Text = "Пуск"; this.Text = "Побудова таблиці";
        }

        private void Button1_Click(object sender, EventArgs e)
        {
            // Ініціалізуємо два рядкових масиви:
            string[] Імена = {"Андрій (роб)", "Світла (моб)", "Микола (дом)",
                "Кафедра (роб)", "Олександр Степанович", "Сергій (дом)",
                "Тетяна Петрівна", "Таксі", "Кінотеатр" };
            string[] Телефон = {"274-88-17", "+38 (067) 7030356", "22-345-72",
                "204-82-12", "223-67-67 доп 32-67", "570-38-76",
                "201-72-23", "555", "216-40-22" };
            // Створюємо новий екземпляр класу Word.Application:
            var Ворд1 = new Microsoft.Office.Interop.Word.Application
            {
                Visible = true
            };
            // Додаємо затримку у роботі програми в 1 сек
            // для уникнення помилки через затримку у роботі MS Word
            System.Threading.Thread.Sleep(1000);
            // Змінна з "порожнім" значенням:
            System.Object  t = Type.Missing;
            // Відкриваємо новий документ MS Word:
            var Документ = Ворд1.Documents.Add(t, t, t, t);
            // Вводимо текст в документ MS WORD з поточної позиції:
            Ворд1.Selection.TypeText("ТАБЛИЦЯ ТЕЛЕФОНІВ");
            // Параметр, який вказує чи показувати межі комірок:
            System.Object  t1 = Microsoft.Office.Interop.
```

```
                    Word.WdDefaultTableBehavior.wdWord9TableBehavior;
                // Параметр, який вказує чи буде додаток Word автоматично
                // змінювати розмір комірок у таблиці для підгонки вмісту:
                System.Object t2 =
Microsoft.Office.Interop.Word.WdAutoFitBehavior.wdAutoFitContent;
                // Створюємо таблицю з 9 рядків і 2 стовпців:
                Ворд1.ActiveDocument.Tables.Add(Ворд1.Selection.Range, 9, 2, t1, t2);
                // Заповнювати комірки таблиці можна так:
                for (int i = 1; i <= 9; i++)
                {
                    Ворд1.ActiveDocument.Tables[1].Cell(i, 1).Range.InsertAfter(Імена[i - 1]);
                    Ворд1.ActiveDocument.Tables[1].Cell(i, 2).Range.InsertAfter(Телефон[i - 1]);
                }
                // Призначаємо одиниці вимірювання в документі додатка MS Word:
                Object  t3 = Microsoft.Office.Interop.Word.WdUnits.wdLine;
                // Параметр, який вказує на дев'ятий рядок у документі MS Word:
                Object  рядок9 = 9;
                // Перевести поточну позицію (Selection) за межі таблиці,
                // (В дев'ятий рядок), щоб тут вивести будь-якій текст:
                Ворд1.Selection.MoveDown(t3, рядок9, t);
                // І тут друкуємо наступний текст:
                Ворд1.Selection.TypeText("Який-небудь текст після таблиці");
                // Зберігати документ немає сенсу, але це вирішить користувач:
                // або можна задати безпосередньо шлях та ім'я файлу
                Object  назваФайлу = "D:\\Таблиця.docx";
                // і одразу зберегти
                Ворд1.ActiveDocument.SaveAs(назваФайлу, t, t, t, t, t, t, t, t, t,
                    t, t, t, t, t, t);
            }
        }
    }
```

Лістинг 9.7

Приклад коду C++/CLI

```
// Програма вирішує систему рівнянь за допомогою функцій об'єктної
бібліотеки MS Excel

#pragma once
// Для підключення бібліотеки об'єктів MS Excel в пункті меню Project
// виберемо команду Add Reference. Потім, якщо на вашому комп'ютері
// встановлений MS Office 2016, то на вкладці COM двічі клацнемо по
// посиланню на бібліотеку Microsoft Excel 16.0 Object Library

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::Label^  label1;
	private: System::Windows::Forms::Label^  label2;
	private: System::Windows::Forms::Label^  label3;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
```

```
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->label1 = (gcnew System::Windows::Forms::Label());
			this->label2 = (gcnew System::Windows::Forms::Label());
			this->label3 = (gcnew System::Windows::Forms::Label());
			this->SuspendLayout();
			//
			// button1
			//
			this->button1->Location = System::Drawing::Point(101, 210);
			this->button1->Name = L"button1";
			this->button1->Size = System::Drawing::Size(75, 23);
			this->button1->TabIndex = 0;
			this->button1->Text = L"button1";
			this->button1->UseVisualStyleBackColor = true;
			this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
			//
			// label1
			//
			this->label1->AutoSize = true;
			this->label1->Location = System::Drawing::Point(12, 9);
			this->label1->Name = L"label1";
			this->label1->Size = System::Drawing::Size(35, 13);
			this->label1->TabIndex = 1;
			this->label1->Text = L"label1";
			//
			// label2
			//
			this->label2->AutoSize = true;
			this->label2->Location = System::Drawing::Point(12, 38);
			this->label2->Name = L"label2";
			this->label2->Size = System::Drawing::Size(35, 13);
			this->label2->TabIndex = 2;
			this->label2->Text = L"label2";
			//
			// label3
			//
			this->label3->AutoSize = true;
			this->label3->Location = System::Drawing::Point(12, 69);
			this->label3->Name = L"label3";
			this->label3->Size = System::Drawing::Size(35, 13);
			this->label3->TabIndex = 3;
			this->label3->Text = L"label3";
			this->label3->Click += gcnew System::EventHandler(this,
&MyForm::label3_Click);
			//
			// MyForm
			//
			this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
			this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
			this->ClientSize = System::Drawing::Size(284, 261);
			this->Controls->Add(this->label3);
			this->Controls->Add(this->label2);
			this->Controls->Add(this->label1);
			this->Controls->Add(this->button1);
			this->Name = L"MyForm";
			this->Text = L"MyForm";
			this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
			this->ResumeLayout(false);
			this->PerformLayout();
		}
```

```
#pragma endregion

        private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
                // Вирішуємо систему
                // x1 + 2*x2 + x3 = -1
                // 3*x1 - x2 - x3 = -1
                // -2*x1 + 2*x2 + 3*x3 = 5
                // Для цієї системи пряма матриця буде мати вигляд:
                // array<double,2> ^ A = gcnew array<double,2>(n, n);
                array<double, 2> ^ A = { { 1, 2, 1 },
                                                { 3, -1, -1 },
                                                { -2, 2, 3 } };
                // Рядок вільних членів:
                // array<double> ^ B = gcnew array<double>(n);
                // Вільні члени:
                array<double> ^ B = { -1, -1, 5 };
                // Створення екземпляра класу Excel::Application:
                auto XL1 = gcnew Microsoft::Office::Interop::Excel::Application();
                // Обчислення детермінанта матриці A
                double визначник_A = XL1->Application->WorksheetFunction->MDeterm(A);
                // Якщо визначник_A != 0, то вихід з процедури:
                if (визначник_A == 0)
                {
                        MessageBox::Show("Система не має рішення, оскільки\n" +
                                "визначник дорівнює нулю", "Немає рішення",
MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
                        return;
                }
                // Отримання зворотної матриці зворотна_A:
                Object ^ зворотна_A = XL1->Application->WorksheetFunction->MInverse(A);
                // Функція Transpose перетворює рядок вільних
                // членів у вектор:
                Object ^ вектор_B = XL1->Application->WorksheetFunction->Transpose(B);
                // Множення матриці на вектор вільних членів:
                Object ^ X =XL1->Application->WorksheetFunction->MMult(зворотна_A, вектор_B);
                // Щоб відповідь набула індексованих властивостей,
                // перетворимо її в масив:
                array<Object^, 2> ^ рішення_X = (array<Object^, 2> ^)X;
                // Отримуємо двовимірний масив, індекси якого
                // починаються з одиниці:
                MessageBox::Show("Невідомі дорівнюють: x1=" + рішення_X[1, 1] +
                        " x2=" + рішення_X[2, 1] + " x3=" + рішення_X[3, 1]);
        }

        private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
                label1->Text = "x1 + 2*x2 + x3 = -1";
                label2->Text = "3*x1 - x2 - x3 = -1";
                label3->Text = "-2*x1 + 2*x2 + 3*x3 = 5";
                button1->Text = "Рішення";
                Text = "Система лінійних алгебраїчних рівнянь";
        }
};
}
```

Лістинг 9.8

Приклад коду C#

```
// Програма вирішує систему рівнянь за допомогою функцій об'єктної
бібліотеки MS Excel
//Файл Form1.Designer.cs

namespace СистемаРівняньExcel
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.button1 = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(325, 87);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(35, 13);
            this.label1.TabIndex = 0;
            this.label1.Text = "label1";
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(326, 114);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(35, 13);
            this.label2.TabIndex = 1;
            this.label2.Text = "label2";
```

```
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(327, 140);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(35, 13);
            this.label3.TabIndex = 2;
            this.label3.Text = "label3";
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(367, 240);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(75, 23);
            this.button1.TabIndex = 3;
            this.button1.Text = "button1";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.Button1_Click);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(800, 450);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Name = "Form1";
            this.Text = "Form1";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Button button1;
    }
}
```

Лістинг 9.9

Приклад коду C#

```
// Програма вирішує систему рівнянь за допомогою функцій об'єктної
бібліотеки MS Excel
//Файл Form1.cs

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace СистемаРівняньExcel
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            label1.Text = "x1 + 2*x2 + x3 = -1";
            label2.Text = "3*x1 - x2 - x3 = -1";
            label3.Text = "-2*x1 + 2*x2 + 3*x3 = 5";
            button1.Text = "Рішення";
            Text = "Система лінійних алгебраїчних рівнянь";
            // Встановлюємо положення об'єктів label відносно
            //лівого краю вікна
            label1.Left = 20;
            label2.Left = 20;
            label3.Left = 20;
            // Встановлюємо положення об'єктів label відносно
            //верхнього краю вікна
            label1.Top = label1.Height * 2;
            label2.Top = label1.Height * 4;
            label3.Top = label1.Height * 6;
            // Вирівнюємо положення кнопки
            //вздовж нижнього краю вікна
            button1.Dock = DockStyle.Bottom;
            // Задаємо висоту вікна з урахування висоти
            //та положення розташованих в ньому об'єктів
            this.Height = label1.Height * 6 + button1.Height + 100;
            // Задаємо ширину вікна
            this.Width = 400;
        }

        private void Button1_Click(object sender, EventArgs e)
        {
            // Вирішуємо систему
            // x1 + 2*x2 + x3 = -1
            // 3*x1 - x2 - x3 = -1
            // -2*x1 + 2*x2 + 3*x3 = 5
            // Для цієї системи пряма матриця буде мати вигляд:
            // double[,]  A = double[n, n];
```

```
            double[,] A = {{ 1, 2, 1 },
                           { 3, -1, -1 },
                           { -2, 2, 3 }
            };
            // Рядок вільних членів:
            // double[] B = double[n];
            // Вільні члени:
            double[] B = { -1, -1, 5 };
            // Створення екземпляра класу Excel.Application:
            var XL1 = new Microsoft.Office.Interop.Excel.Application();
            // Обчислення детермінанта матриці A
            double визначник_A = XL1.Application.WorksheetFunction.MDeterm(A);
            // Якщо визначник_A != 0, то вихід з процедури:
            if (визначник_A == 0)
            {
                MessageBox.Show("Система не має рішення, оскільки\n" +
                    "визначник дорівнює нулю", "Немає рішення",
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return;
            }
            // Отримання зворотної матриці зворотна_A:
            Object  зворотна_A = XL1.Application.WorksheetFunction.MInverse(A);
            // Функція Transpose перетворює рядок вільних
            // членів у вектор:
            Object  вектор_B = XL1.Application.WorksheetFunction.Transpose(B);
            // Множення матриці на вектор вільних членів:
            Object  X =
                XL1.Application.WorksheetFunction.MMult(зворотна_A, вектор_B);
            // Щоб відповідь набула індексованих властивостей,
            // перетворимо її в масив:
            Object[,] рішення_X = (Object[,])X;
            // Отримуємо двовимірний масив, індекси якого
            // починаються з одиниці:
            MessageBox.Show("Невідомі дорівнюють: x1=" + рішення_X[1, 1] + " x2=" +
                рішення_X[2, 1] + " x3=" + рішення_X[3, 1]);
        }
    }
}
```

Лістинг 9.10

Приклад коду C++/CLI

```
// Програма побудови діаграми на основі двох стовпчиків даних,
//які ввів користувач, засобами MS Excel з можливістю збереження
//отриманої діаграми у растровому зображенні

#pragma once
namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	//using namespace Microsoft::Office::Interop::Excel;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Panel^  panel1;
	protected:

	private: System::Windows::Forms::Panel^  panel3;
	private: System::Windows::Forms::Button^  button3;
	private: System::Windows::Forms::TextBox^  textBox3;
	private: System::Windows::Forms::TextBox^  textBox2;
	private: System::Windows::Forms::TextBox^  textBox1;
	private: System::Windows::Forms::Label^  label3;
	private: System::Windows::Forms::Label^  label2;
	private: System::Windows::Forms::Label^  label1;
	private: System::Windows::Forms::DataGridView^  dataGridView1;
	private: System::Windows::Forms::Panel^  panel2;
	private: System::Windows::Forms::Button^  button2;
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::SaveFileDialog^  saveFileDialog1;
	private: System::Windows::Forms::SaveFileDialog^  saveFileDialog2;
	private:
```

```
        /// <summary>
        /// Обязательная переменная конструктора.
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Требуемый метод для поддержки конструктора — не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        void InitializeComponent(void)
        {
            this->panel1 = (gcnew System::Windows::Forms::Panel());
            this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());
            this->panel3 = (gcnew System::Windows::Forms::Panel());
            this->button3 = (gcnew System::Windows::Forms::Button());
            this->textBox3 = (gcnew System::Windows::Forms::TextBox());
            this->textBox2 = (gcnew System::Windows::Forms::TextBox());
            this->textBox1 = (gcnew System::Windows::Forms::TextBox());
            this->label3 = (gcnew System::Windows::Forms::Label());
            this->label2 = (gcnew System::Windows::Forms::Label());
            this->label1 = (gcnew System::Windows::Forms::Label());
            this->panel2 = (gcnew System::Windows::Forms::Panel());
            this->button2 = (gcnew System::Windows::Forms::Button());
            this->button1 = (gcnew System::Windows::Forms::Button());
            this->saveFileDialog1 = (gcnew
System::Windows::Forms::SaveFileDialog());
            this->saveFileDialog2 = (gcnew
System::Windows::Forms::SaveFileDialog());
            this->panel1->SuspendLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->BeginInit();
            this->panel3->SuspendLayout();
            this->panel2->SuspendLayout();
            this->SuspendLayout();
            //
            // panel1
            //
            this->panel1->Controls->Add(this->dataGridView1);
            this->panel1->Location = System::Drawing::Point(0, 0);
            this->panel1->Name = L"panel1";
            this->panel1->Size = System::Drawing::Size(346, 178);
            this->panel1->TabIndex = 2;
            //
            // dataGridView1
            //
            this->dataGridView1->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
            this->dataGridView1->Location = System::Drawing::Point(12, 12);
            this->dataGridView1->Name = L"dataGridView1";
            this->dataGridView1->Size = System::Drawing::Size(263, 120);
            this->dataGridView1->TabIndex = 1;
            //
            // panel3
            //
            this->panel3->Controls->Add(this->button3);
            this->panel3->Controls->Add(this->textBox3);
            this->panel3->Controls->Add(this->textBox2);
            this->panel3->Controls->Add(this->textBox1);
            this->panel3->Controls->Add(this->label3);
            this->panel3->Controls->Add(this->label2);
            this->panel3->Controls->Add(this->label1);
            this->panel3->Location = System::Drawing::Point(12, 273);
            this->panel3->Name = L"panel3";
            this->panel3->Size = System::Drawing::Size(400, 130);
```

```
                this->panel3->TabIndex = 1;
                //
                // button3
                //
                this->button3->Location = System::Drawing::Point(103, 91);
                this->button3->Name = L"button3";
                this->button3->Size = System::Drawing::Size(75, 23);
                this->button3->TabIndex = 4;
                this->button3->Text = L"button3";
                this->button3->UseVisualStyleBackColor = true;
                this->button3->Click += gcnew System::EventHandler(this,
&MyForm::button3_Click);
                //
                // textBox3
                //
                this->textBox3->Location = System::Drawing::Point(163, 65);
                this->textBox3->Name = L"textBox3";
                this->textBox3->Size = System::Drawing::Size(146, 20);
                this->textBox3->TabIndex = 3;
                //
                // textBox2
                //
                this->textBox2->Location = System::Drawing::Point(163, 39);
                this->textBox2->Name = L"textBox2";
                this->textBox2->Size = System::Drawing::Size(146, 20);
                this->textBox2->TabIndex = 2;
                //
                // textBox1
                //
                this->textBox1->Location = System::Drawing::Point(163, 13);
                this->textBox1->Name = L"textBox1";
                this->textBox1->Size = System::Drawing::Size(146, 20);
                this->textBox1->TabIndex = 1;
                //
                // label3
                //
                this->label3->AutoSize = true;
                this->label3->Location = System::Drawing::Point(22, 64);
                this->label3->Name = L"label3";
                this->label3->Size = System::Drawing::Size(35, 13);
                this->label3->TabIndex = 2;
                this->label3->Text = L"label3";
                //
                // label2
                //
                this->label2->AutoSize = true;
                this->label2->Location = System::Drawing::Point(22, 38);
                this->label2->Name = L"label2";
                this->label2->Size = System::Drawing::Size(35, 13);
                this->label2->TabIndex = 1;
                this->label2->Text = L"label2";
                //
                // label1
                //
                this->label1->AutoSize = true;
                this->label1->Location = System::Drawing::Point(22, 13);
                this->label1->Name = L"label1";
                this->label1->Size = System::Drawing::Size(35, 13);
                this->label1->TabIndex = 0;
                this->label1->Text = L"label1";
                //
                // panel2
                //
                this->panel2->Controls->Add(this->button2);
                this->panel2->Controls->Add(this->button1);
```

```
                    this->panel2->Location = System::Drawing::Point(25, 194);
                    this->panel2->Name = L"panel2";
                    this->panel2->Size = System::Drawing::Size(368, 73);
                    this->panel2->TabIndex = 5;
                    //
                    // button2
                    //
                    this->button2->Location = System::Drawing::Point(163, 48);
                    this->button2->Name = L"button2";
                    this->button2->Size = System::Drawing::Size(75, 23);
                    this->button2->TabIndex = 2;
                    this->button2->Text = L"button2";
                    this->button2->UseVisualStyleBackColor = true;
                    this->button2->Click += gcnew System::EventHandler(this,
&MyForm::button2_Click);
                    //
                    // button1
                    //
                    this->button1->Location = System::Drawing::Point(47, 48);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(75, 23);
                    this->button1->TabIndex = 1;
                    this->button1->Text = L"button1";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(412, 414);
                    this->Controls->Add(this->panel2);
                    this->Controls->Add(this->panel3);
                    this->Controls->Add(this->panel1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->panel1->ResumeLayout(false);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->EndInit();
                    this->panel3->ResumeLayout(false);
                    this->panel3->PerformLayout();
                    this->panel2->ResumeLayout(false);
                    this->ResumeLayout(false);
                }
#pragma endregion
                DataTable^ Таблиця;  // Об'явлення объекта "Таблиця"
                DataSet^ НабірДаних; // Об'явлення объекта "НабірДаних"
                DataColumn^ Стовпчик1; // Об'явлення объекта Стовпчик1
                DataColumn^ Стовпчик2; // Об'явлення объекта Стовпчик2

        private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
                // Створюємо екземпляр класу Excel::Application:
                Microsoft::Office::Interop::Excel::Application ^ XL1 = gcnew
Microsoft::Office::Interop::Excel::Application();
                XL1->Visible = true;
                // Задаємо параметр за замовчуванням для його подальшого
                // використання у відповідних методах:
                Object ^ t = Type::Missing;
                // Створюємо нову книгу MS Excel:
                Microsoft::Office::Interop::Excel::Workbook ^ Книга = XL1->Workbooks->Add(t);
                // Об'являємо аркуші в книзі:
                Microsoft::Office::Interop::Excel::Sheets ^ Аркуші = Книга->Worksheets;
```

```
                // Вибираємо перший аркуш:
                Microsoft::Office::Interop::Excel::Worksheet ^ Аркуш =
(Microsoft::Office::Interop::Excel::Worksheet ^)Аркуші->Item[1];
                // Якщо хочемо додати ще один аркуш (четвертий) до вже існуючих:
                // _Worksheet ^ Аркуш = safe_cast<_Worksheet^>(Аркуші->Item[ (Object^)1 ]);
                // Записуємо дані у відповідних комірках:
                String ^ colA;
                String ^ colB;
                // Записуємо заголовки стовпчиків на лист Excel
                Аркуш->Range["A1", t]->Value2 = Стовпчик1->Caption;
                Аркуш->Range["B1", t]->Value2 = Стовпчик2->Caption;
                // Записуємо таблицю на лист Excel
                for (int i = 0; i < Таблиця->Rows->Count; i++)
                {
                        colA = "A" + (i+2).ToString();
                        colB = "B" + (i+2).ToString();
                        Аркуш->Range[colA, t]->Value2 = dataGridView1->Rows[i]->Cells[0]-
>Value;
                        Аркуш->Range[colB, t]->Value2 = dataGridView1->Rows[i]->Cells[1]-
>Value;
                }
                // Замовляємо побудову діаграми (графіка) з параметрами за замовчуванням:
                Microsoft::Office::Interop::Excel::Chart ^ Графік =
(Microsoft::Office::Interop::Excel::Chart ^)XL1->Charts->Add(t, t, t, t);
                // Задаємо діапазон значень для побудови графіку:
                Графік->SetSourceData(Аркуш->Range["A2", colB],
Microsoft::Office::Interop::Excel::XlRowCol::xlColumns);
                // Задаємо тип графіка "стовпчикова діаграма" (гістограма):
                Графік->ChartType =
Microsoft::Office::Interop::Excel::XlChartType::xlColumnClustered;
                // Відключаємо легенду графіка:
                Графік->HasLegend = false;
                // Графік має заголовок:
                Графік->HasTitle = true;
                Графік->ChartTitle->Caption = Таблиця->TableName;
                // Підпис осі X:
                Microsoft::Office::Interop::Excel::Axis ^ ГоризонтальнаОсь =
(Microsoft::Office::Interop::Excel::Axis^)Графік->Axes(
                        Microsoft::Office::Interop::Excel::XlAxisType::xlCategory,
Microsoft::Office::Interop::Excel::XlAxisGroup::xlPrimary);
                ГоризонтальнаОсь->HasTitle = true;
                ГоризонтальнаОсь->AxisTitle->Text = Стовпчик1->Caption;
                // Підпис осі Y:
                Microsoft::Office::Interop::Excel::Axis ^ ВертикальнаОсь =
(Microsoft::Office::Interop::Excel::Axis^)Графік->Axes(
                        Microsoft::Office::Interop::Excel::XlAxisType::xlValue,
Microsoft::Office::Interop::Excel::XlAxisGroup::xlPrimary);
                ВертикальнаОсь->HasTitle = true;
                ВертикальнаОсь->AxisTitle->Text = Стовпчик2->Caption;
                // Збереження графіка в растровому файлі:
                if (MessageBox::Show("Зберегти діаграму?", "Діаграма",
                        MessageBoxButtons::YesNo, MessageBoxIcon::Question) ==
System::Windows::Forms::DialogResult::Yes)
                {
                        if (saveFileDialog2->ShowDialog() ==
System::Windows::Forms::DialogResult::OK)
                        {
                                // Якщо користувач натиснув Зберегти, зберігаємо діаграму
                                XL1->ActiveChart->Export(saveFileDialog2->FileName, t, t);
                        }
                }
        }

        private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
                // Задаємо написи на кнопках
```

```
            button3->Text = "Створити таблицю";
            button1->Text = "Створити діаграму Excel";
            button2->Text = "Зберегти таблицю у файл xml";
            // Задаємо заголовок вікна
            Text = "Створення діаграми в MS Excel";
            // Задаємо висоту кнопок
            button1->Height = 25;
            button2->Height = button1->Height;
            button3->Height = button1->Height;
            // Задаємо написи на мітках Lable
            label1->Text = "Назва таблиці";
            label2->Text = "Ім'я 1-го стовпчика";
            label3->Text = "Ім'я 2-го стовпчика";
            // Вирівнюємо кнопки в межах об'єктів Panel
            button1->Dock = DockStyle::Bottom;
            button2->Dock = DockStyle::Bottom;
            button3->Dock = DockStyle::Bottom;
            // Вирівнюємо об'кти Lable та TextBox в межах об'єкту panel3
            label1->Width = label3->Width;
            label2->Width = label3->Width;
            label1->Top = 10; label1->Left = 10;
            label2->Top = (label1->Height + 10) + 10; label2->Left = 10;
            label3->Top = (label1->Height + 10) * 2 + 10; label3->Left = 10;
            textBox1->Top = label1->Top; textBox1->Left = label3->Width + 30;
            textBox2->Top = label2->Top; textBox2->Left = label3->Width + 30;
            textBox3->Top = label3->Top; textBox3->Left = label3->Width + 30;
            textBox1->Width = 110;
            textBox2->Width = textBox1->Width;
            textBox3->Width = textBox1->Width;
            // Задаємо ширину форми з урахуванням ширини об'єктів Lable та TextBox
            Width = 10 + label1->Width + 20 + textBox1->Width + 60;
            // Задаємо висоту panel3 з урахуванням висоти об'єктів Lable, TextBox, button3
            panel3->Height = (label1->Height + 10) * 3 + button3->Height + 30;
            // Задаємо висоту panel2
            panel2->Height = button1->Height * 2;
            // Вирівнюємо об'єкти Panel в межах форми
            panel2->Dock = DockStyle::Bottom;
            panel3->Dock = DockStyle::Bottom;
            dataGridView1->Dock = DockStyle::Fill;
            panel1->Dock = DockStyle::Fill;
            // Властивіст DockStyle::Fill працює не завжди коректно
            //щоб виправити ситуацію можна скористатись "чарівним оператором",
            //який встановить для проблемного об'єкта індекс 0 властивості
            //SetChildIndex батьківського об'єкту
            this->Controls->SetChildIndex(panel1, 0);//////////////////////////////////
            // Приховуємо панель panel2 з кнопками побудови діаграми та збереження таблиці
            panel2->Visible = false;
            // Встановлюємо фільтр для збереження таблиці у файл XML
            saveFileDialog1->Filter = "Файли XML(*.xml)|*.xml|Всі файли (*.*)|*.*";
            // Встановлюємо фільтр для збереження діаграми у файл JPG
            saveFileDialog2->Filter = "Файли зображень JPG(*.jpg)|*.jpg|Всі файли
(*.*)|*.*";
        }

private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
        // Зберегти файл tabl.xml:
        if (saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
        {
            // Якщо користувач натиснув Зберегти, зберігаємо тиблицю
            НаборДаних->WriteXml(saveFileDialog1->FileName);
        }
}

private: System::Void button3_Click(System::Object^  sender, System::EventArgs^  e) {
        Таблиця = gcnew DataTable();
```

```
            НабірДаних = gcnew DataSet();
            // Зчитуємо назви стовпчиків з полів textBox
            Стовпчик1 = Таблиця->Columns->Add(textBox2->Text);
            Стовпчик2 = Таблиця->Columns->Add(textBox3->Text);
            // Зчитуємо назву таблиці з поля textBox1
            Таблиця->TableName = textBox1->Text;
            // Пов'язуємо об'єкт dataGridView1 з Таблицею
            dataGridView1->DataSource = Таблиця;
            // Додати об'єкт Таблиця в DataSet
            НабірДаних->Tables->Add(Таблиця);
            // Відображаємо панель з кнопками
            panel2->Visible = true;
        }
        };
        }
```

Лістинг 9.11

Приклад коду C#

```
// Програма побудови діаграми на основі двох стовпчиків даних,
//які ввів користувач, засобами MS Excel з можливістю збереження
//отриманої діаграми у растровому зображенні
//Файл Form1.Designer.cs

namespace ДіаграмиExcel
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.panel1 = new System.Windows.Forms.Panel();
            this.dataGridView1 = new System.Windows.Forms.DataGridView();
            this.panel2 = new System.Windows.Forms.Panel();
            this.button1 = new System.Windows.Forms.Button();
            this.button2 = new System.Windows.Forms.Button();
            this.panel3 = new System.Windows.Forms.Panel();
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.textBox3 = new System.Windows.Forms.TextBox();
            this.button3 = new System.Windows.Forms.Button();
            this.saveFileDialog1 = new System.Windows.Forms.SaveFileDialog();
            this.saveFileDialog2 = new System.Windows.Forms.SaveFileDialog();
            this.panel1.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).BeginInit();
            this.panel2.SuspendLayout();
            this.panel3.SuspendLayout();
            this.SuspendLayout();
            //
```

```
            // panel1
            //
            this.panel1.Controls.Add(this.dataGridView1);
            this.panel1.Location = new System.Drawing.Point(27, 12);
            this.panel1.Name = "panel1";
            this.panel1.Size = new System.Drawing.Size(200, 100);
            this.panel1.TabIndex = 0;
            //
            // dataGridView1
            //
            this.dataGridView1.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            this.dataGridView1.Location = new System.Drawing.Point(50, 30);
            this.dataGridView1.Name = "dataGridView1";
            this.dataGridView1.Size = new System.Drawing.Size(240, 150);
            this.dataGridView1.TabIndex = 0;
            //
            // panel2
            //
            this.panel2.Controls.Add(this.button2);
            this.panel2.Controls.Add(this.button1);
            this.panel2.Location = new System.Drawing.Point(250, 92);
            this.panel2.Name = "panel2";
            this.panel2.Size = new System.Drawing.Size(200, 100);
            this.panel2.TabIndex = 1;
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(36, 21);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(75, 23);
            this.button1.TabIndex = 0;
            this.button1.Text = "button1";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.Button1_Click);
            //
            // button2
            //
            this.button2.Location = new System.Drawing.Point(34, 60);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(75, 23);
            this.button2.TabIndex = 1;
            this.button2.Text = "button2";
            this.button2.UseVisualStyleBackColor = true;
            this.button2.Click += new System.EventHandler(this.Button2_Click);
            //
            // panel3
            //
            this.panel3.Controls.Add(this.button3);
            this.panel3.Controls.Add(this.textBox3);
            this.panel3.Controls.Add(this.textBox2);
            this.panel3.Controls.Add(this.textBox1);
            this.panel3.Controls.Add(this.label3);
            this.panel3.Controls.Add(this.label2);
            this.panel3.Controls.Add(this.label1);
            this.panel3.Location = new System.Drawing.Point(27, 169);
            this.panel3.Name = "panel3";
            this.panel3.Size = new System.Drawing.Size(200, 138);
            this.panel3.TabIndex = 2;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(18, 17);
            this.label1.Name = "label1";
```

```
this.label1.Size = new System.Drawing.Size(35, 13);
this.label1.TabIndex = 0;
this.label1.Text = "label1";
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(17, 41);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(35, 13);
this.label2.TabIndex = 1;
this.label2.Text = "label2";
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(19, 67);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(35, 13);
this.label3.TabIndex = 2;
this.label3.Text = "label3";
//
// textBox1
//
this.textBox1.Location = new System.Drawing.Point(75, 20);
this.textBox1.Name = "textBox1";
this.textBox1.Size = new System.Drawing.Size(100, 20);
this.textBox1.TabIndex = 3;
//
// textBox2
//
this.textBox2.Location = new System.Drawing.Point(76, 49);
this.textBox2.Name = "textBox2";
this.textBox2.Size = new System.Drawing.Size(100, 20);
this.textBox2.TabIndex = 4;
//
// textBox3
//
this.textBox3.Location = new System.Drawing.Point(78, 76);
this.textBox3.Name = "textBox3";
this.textBox3.Size = new System.Drawing.Size(100, 20);
this.textBox3.TabIndex = 5;
//
// button3
//
this.button3.Location = new System.Drawing.Point(71, 107);
this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(75, 23);
this.button3.TabIndex = 6;
this.button3.Text = "button3";
this.button3.UseVisualStyleBackColor = true;
this.button3.Click += new System.EventHandler(this.Button3_Click);
//
// Form1
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(515, 324);
this.Controls.Add(this.panel3);
this.Controls.Add(this.panel2);
this.Controls.Add(this.panel1);
this.Name = "Form1";
this.Text = "Form1";
this.Load += new System.EventHandler(this.Form1_Load);
this.panel1.ResumeLayout(false);
```

```
            ((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();
            this.panel2.ResumeLayout(false);
            this.panel3.ResumeLayout(false);
            this.panel3.PerformLayout();
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.Panel panel1;
        private System.Windows.Forms.DataGridView dataGridView1;
        private System.Windows.Forms.Panel panel2;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Panel panel3;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.SaveFileDialog saveFileDialog1;
        private System.Windows.Forms.SaveFileDialog saveFileDialog2;
    }
}
```

Лістинг 9.12

Приклад коду C#

```
// Програма побудови діаграми на основі двох стовпчиків даних,
//які ввів користувач, засобами MS Excel з можливістю збереження
//отриманої діаграми у растровому зображенні
//Файл Form1.cs

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ДіаграмиExcel
{
    public partial class Form1 : Form
    {
       DataTable Таблиця;  // Об'явлення объекта "Таблиця"
       DataSet НабірДаних; // Об'явлення объекта "НабірДаних"
       DataColumn Стовпчик1; // Об'явлення объекта Стовпчик1
       DataColumn Стовпчик2; // Об'явлення объекта Стовпчик2

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // Задаємо написи на кнопках
            button3.Text = "Створити таблицю";
            button1.Text = "Створити діаграму Excel";
            button2.Text = "Зберегти таблицю у файл xml";
            // Задаємо заголовок вікна
            Text = "Створення діаграми в MS Excel";
            // Задаємо висоту кнопок
            button1.Height = 25;
            button2.Height = button1.Height;
            button3.Height = button1.Height;
            // Задаємо написи на мітках Lable
            label1.Text = "Назва таблиці";
            label2.Text = "Ім'я 1-го стовпчика";
            label3.Text = "Ім'я 2-го стовпчика";
            // Вирівнюємо кнопки в межах об'єктів Panel
            button1.Dock = DockStyle.Bottom;
            button2.Dock = DockStyle.Bottom;
            button3.Dock = DockStyle.Bottom;
            // Вирівнюємо об'кти Lable та TextBox в межах об'єкту panel3
            label1.Width = label3.Width;
            label2.Width = label3.Width;
            label1.Top = 10; label1.Left = 10;
            label2.Top = (label1.Height + 10) + 10; label2.Left = 10;
            label3.Top = (label1.Height + 10) * 2 + 10; label3.Left = 10;
            textBox1.Top = label1.Top; textBox1.Left = label3.Width + 30;
            textBox2.Top = label2.Top; textBox2.Left = label3.Width + 30;
```

```
            textBox3.Top = label3.Top; textBox3.Left = label3.Width + 30;
            textBox1.Width = 110;
            textBox2.Width = textBox1.Width;
            textBox3.Width = textBox1.Width;
            // Задаємо ширину форми з урахуванням ширини об'єктів Lable та TextBox
            Width = 10 + label1.Width + 20 + textBox1.Width + 60;
            // Задаємо висоту panel3 з урахуванням висоти об'єктів Lable, TextBox, button3
            panel3.Height = (label1.Height + 10) * 3 + button3.Height + 30;
            // Задаємо висоту panel2
            panel2.Height = button1.Height * 2;
            // Вирівнюємо об'єкти Panel в межах форми
            panel2.Dock = DockStyle.Bottom;
            panel3.Dock = DockStyle.Bottom;
            dataGridView1.Dock = DockStyle.Fill;
            panel1.Dock = DockStyle.Fill;
            // Властивіст DockStyle.Fill працює не завжди коректно
            //щоб виправити ситуацію можна скористатись "чарівним оператором",
            //який встановить для проблемного об'єкта індекс 0 властивості
            //SetChildIndex батьківського об'єкту
            this.Controls.SetChildIndex(panel1, 0);
            // Приховуємо панель panel2 з кнопками побудови діаграми та збереження таблиці
            panel2.Visible = false;
            // Встановлюємо фільтр для збереження таблиці у файл XML
            saveFileDialog1.Filter = "Файли XML(*.xml)|*.xml|Всі файли (*.*)|*.*";
            // Встановлюємо фільтр для збереження діаграми у файл JPG
            saveFileDialog2.Filter = "Файли зображень JPG(*.jpg)|*.jpg|Всі файли(*.*)| *.*";
        }

        private void Button1_Click(object sender, EventArgs e)
        {
            // Створюємо екземпляр класу Excel.Application:
            Microsoft.Office.Interop.Excel.Application XL1 = new
Microsoft.Office.Interop.Excel.Application
            {
                Visible = true
            };
            // Задаємо параметр за замовчуванням для його подальшого
            // використання у відповідних методах:
            Object  t = Type.Missing;
            // Створюємо нову книгу MS Excel:
            Microsoft.Office.Interop.Excel.Workbook  Книга = XL1.Workbooks.Add(t);
            // Об'являємо аркуші в книзі:
            Microsoft.Office.Interop.Excel.Sheets  Аркуші = Книга.Worksheets;
            // Вибираємо перший аркуш:
            Microsoft.Office.Interop.Excel.Worksheet  Аркуш =
                (Microsoft.Office.Interop.Excel.Worksheet )Аркуші.Item[1];
            // Якщо хочемо додати ще один аркуш (четвертий) до вже існуючих:
            // _Worksheet  Аркуш = safe_cast<_Worksheet^>(Аркуші.Item[ (Object^)1 ]);
            // Записуємо дані у відповідних комірках:
            string  colA="";
            string  colB="";
            // Записуємо заголовки стовпчиків на лист Excel
            Аркуш.Range["A1", t].Value2 = Стовпчик1.Caption;
            Аркуш.Range["B1", t].Value2 = Стовпчик2.Caption;
            // Записуємо таблицю на лист Excel
            for (int i = 0; i < Таблиця.Rows.Count; i++)
            {
                colA = "A" + (i + 2).ToString();
                colB = "B" + (i + 2).ToString();
                Аркуш.Range[colA, t].Value2 = dataGridView1.Rows[i].Cells[0].Value;
                Аркуш.Range[colB, t].Value2 = dataGridView1.Rows[i].Cells[1].Value;
            }
            // Замовляємо побудову діаграми (графіка) з параметрами за замовчуванням:
            Microsoft.Office.Interop.Excel.Chart  Графік =
              (Microsoft.Office.Interop.Excel.Chart )XL1.Charts.Add(t, t, t, t);
```

```
            // Задаємо діапазон значень для побудови графіку:
            Графік.SetSourceData(Аркуш.Range["A2", colB],
Microsoft.Office.Interop.Excel.XlRowCol.xlColumns);
            // Задаємо тип графіка "стовпчикова діаграма" (гістограма):
            Графік.ChartType =
                Microsoft.Office.Interop.Excel.XlChartType.xlColumnClustered;
            Графік.HasLegend = false; // Відключаємо легенду графіка
            Графік.HasTitle = true; // Графік має заголовок
            Графік.ChartTitle.Caption = Таблиця.TableName;
            // Підпис осі X:
            Microsoft.Office.Interop.Excel.Axis  ГоризонтальнаОсь =
                (Microsoft.Office.Interop.Excel.Axis )Графік.Axes(
                 Microsoft.Office.Interop.Excel.XlAxisType.xlCategory,
                 Microsoft.Office.Interop.Excel.XlAxisGroup.xlPrimary);
            ГоризонтальнаОсь.HasTitle = true;
            ГоризонтальнаОсь.AxisTitle.Text = Стовпчик1.Caption;
            // Підпис осі Y:
            Microsoft.Office.Interop.Excel.Axis  ВертикальнаОсь =
                (Microsoft.Office.Interop.Excel.Axis )Графік.Axes(
                 Microsoft.Office.Interop.Excel.XlAxisType.xlValue,
                 Microsoft.Office.Interop.Excel.XlAxisGroup.xlPrimary);
            ВертикальнаОсь.HasTitle = true;
            ВертикальнаОсь.AxisTitle.Text = Стовпчик2.Caption;
            // Збереження графіка в растровому файлі:
            if (MessageBox.Show("Зберегти діаграму?", "Діаграма",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
                System.Windows.Forms.DialogResult.Yes)
            {
                if (saveFileDialog2.ShowDialog() ==
                    System.Windows.Forms.DialogResult.OK)
                {
                    // Якщо користувач натиснув Зберегти, зберігаємо діаграму
                    XL1.ActiveChart.Export(saveFileDialog2.FileName, t, t);
                }
            }
        }

        private void Button2_Click(object sender, EventArgs e)
        {
            // Зберегти файл *.xml:
            if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                // Якщо користувач натиснув Зберегти, зберігаємо тиблицю
                НаборДаних.WriteXml(saveFileDialog1.FileName);
            }
        }

        private void Button3_Click(object sender, EventArgs e)
        {
            Таблиця = new DataTable();
            НаборДаних = new DataSet();
            // Зчитуємо назви стовпчиків з полів textBox
            Стовпчик1 = Таблиця.Columns.Add(textBox2.Text);
            Стовпчик2 = Таблиця.Columns.Add(textBox3.Text);
            // Зчитуємо назву таблиці з поля textBox1
            Таблиця.TableName = textBox1.Text;
            // Пов'язуємо об'єкт dataGridView1 з Таблицею
            dataGridView1.DataSource = Таблиця;
            // Додати об'єкт Таблиця в DataSet
            НаборДаних.Tables.Add(Таблиця);
            // Відображаємо панель з кнопками
            panel2.Visible = true;
        }
    }
}
```