Тернопільський національний технічний університет імені Івана Пулюя

Кафедра автоматизації технологічних
процесів та виробництв

Методичні вказівки до виконання лабораторних робіт

з курсу "**Об'єктно-орієнтоване програмування**"

Тернопіль, 2017

Методичні вказівки до виконання лабораторних робіт з курсу "Об'єктно-орієнтоване програмування"/ Уклад. Коноваленко І.В.– Тернопіль: ТНТУ, 2017.

Укладач: к.т.н., доц. каф. автоматизації технологічних процесів і виробництв Коноваленко І.В.

Розглянуто та затверджено на засіданні кафедри автоматизації технологічних процесів і виробництв.

Протокол №3 від 19 вересня 2017 р.

Лабораторна робота №1

Тема. Розробка простого консольного застосунку.

Мета. Ознайомлення з середовищем Microsoft Visual Studio; ознайомлення зі структурою консольного застосунку; вивчення базової структури програми на C#.

Завдання

Розробити консольний застосунок, який розраховуватиме заданий математичний вираз з однією змінною $f(x) = x^2$ в певному діапазоні. Програма повинна забезпечувати ввід мінімального x_{min} та максимального x_{max} значення x, а також кроку його зміни dx. Всі розраховані значення виразу для всіх $x_{min} \le x \le x_{max}$ слід послідовно вивести на екран.

Виконання завдання

1. В середовищі Microsoft Visual Studio створити новий проект консольного застосунку. Для цього вибрати команду меню File/New/Project. У вікні New Project (рис. 1.1) вибрати шаблон ConsoleApplication у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab01, але можна задати іншу) та його розташування на диску.

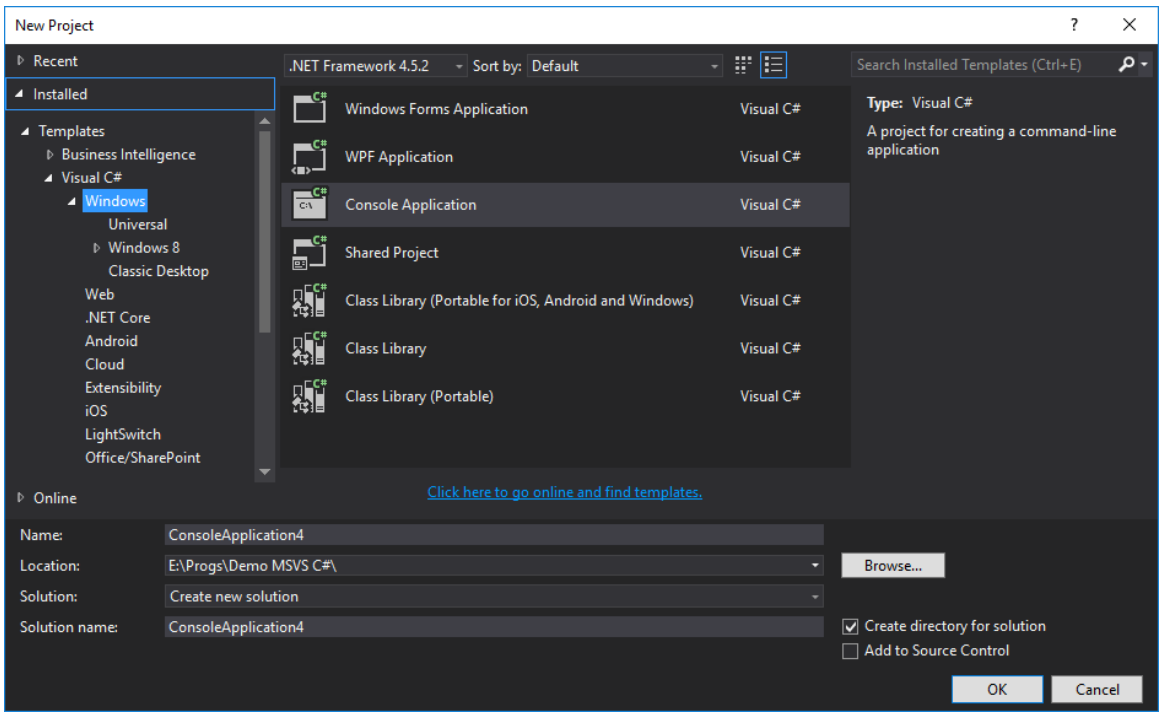


Рис. 1.1. Вікно для задання параметрів нового проекту

2. У вікні Solution Explorer ознайомитися зі структурою проекту (рис. 1.2). Зайти у папку, де було збережено проект, і ознайомитися з його файловою структурою. В загальному випадку проект типу Console Application містить такі частини:
   - Папка Lab01 – папка для рішення (solution)
     a. Файл Lab01.sln – xml-файл для опису параметрів рішення (файл рішення)
     b. Папка Lab01 – папка проекту застосунку
        i. Файл Lab01.csproj – xml-файл для опису параметрів проекту (файл проекту)
        ii. Папка Properties – для зберігання властивостей проекту. Розглядати файл Assemblylnfo.cs зараз не будемо. У ньому міститься додаткова інформація про проект
        iii. Файл Program.cs – код головного модуля застосунку мовою C#

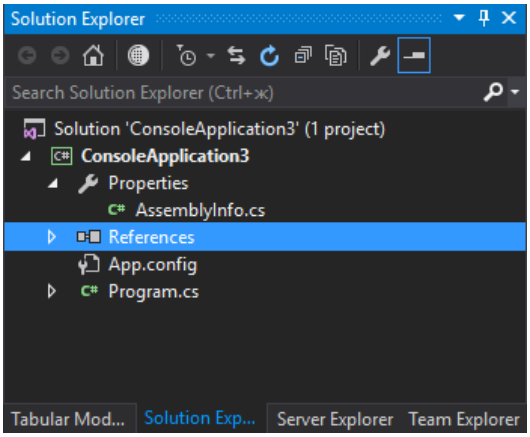


Рис. 1.2. Вікно Solution Explorer

3. Продумати алгоритм програми. Алгоритм для виконання завдання такий:
   1) Ввести *xMin*, *xMax*, *dx*.
   2) Тимчасовій змінній *x* присвоїти значення *xMin*.
   3) У циклі, доки *x* <= *xMax*:
      - Розрахувати значення функції *y = f(x)*;
      - Вивести на екран пару: *x* та *y*;
      - Збільшити значення *x* на *dx*;
   4) Якщо *x* <> *xMax*, то:
      - Розрахувати значення функції *y = f(xMax)*;
      - Вивести на екран пару: *xMax* та *y*;
4. У редакторі коду Microsoft Visual Studio у функцію Main() записати код з лістингу 1.1.
5. Запустити застосунок (F5 в середовищі Microsoft Visual Studio) і перевірити його роботу. Приклад виконання розробленої програми приведено на рис. 1.3.

Лістинг 1.1

```
Console.Write("Введіть початкове значення Xmin: ");
string sxMin = Console.ReadLine();
double xMin = Double.Parse(sxMin);

Console.Write("Введіть кінцеве значення Xmax: ");
string sxMax = Console.ReadLine();
double xMax = double.Parse(sxMax);

Console.Write("Введіть приріст dX: ");
string sdx = Console.ReadLine();
double dx = double.Parse(sdx);

double x = xMin;
double y;

while (x <= xMax)
{
    y = Math.Pow(x, 2);
    Console.WriteLine("x = {0}\t\t y = {1}", x, y);

    x += dx;
```

```
}

if (Math.Abs(x - xMax - dx) > 0.0001)
{
    y = Math.Pow(xMax, 2);
    Console.WriteLine("x = {0}\t\t y = {1}", xMax, y);
}

Console.ReadKey();
```

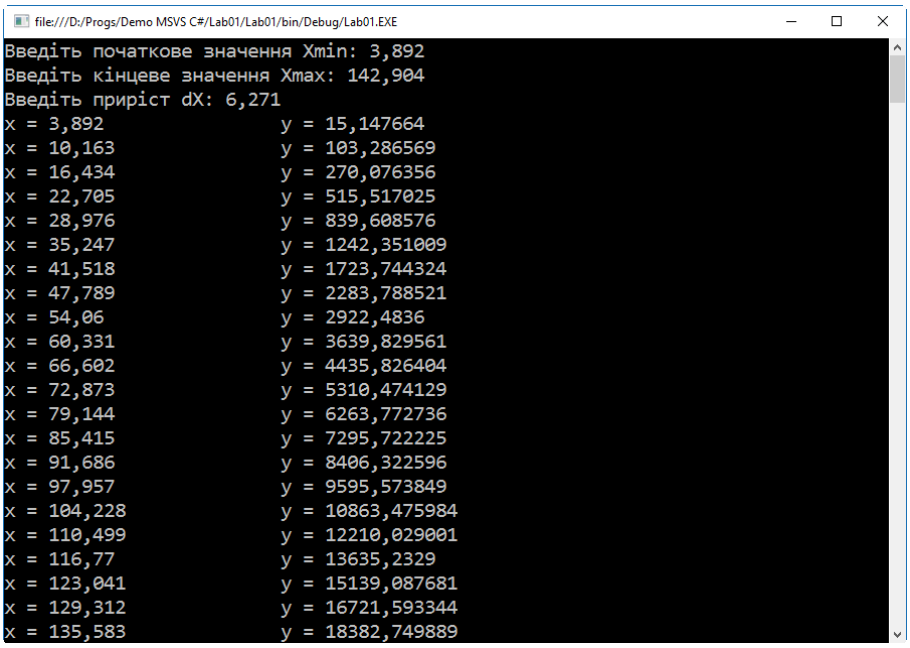


Рис. 1.3. Вивід розробленого застосунку у консоль

Завдання для самостійного опрацювання

1. Вирішити розглянуте раніше завдання для заданої функції *f(x)* згідно варіанту (таблиця 1.1). У виразах прийняти, що x_1=x, x_2=3x.
2. Додатково до попереднього завдання, обчислити та вивести на екран:
   - для варіантів, номери яких закінчуються на 0 чи 5 – суму всіх розрахованих проміжних значень *f(x)*;
   - для варіантів, номери яких закінчуються на 1 чи 6 – добуток всіх розрахованих проміжних значень *f(x)*;
   - для варіантів, номери яких закінчуються на 2 чи 7 – суму синусів всіх розрахованих проміжних значень *f(x)*;
   - для варіантів, номери яких закінчуються на 3 чи 8 – добуток косинусів всіх розрахованих проміжних значень *f(x)*;
   - для варіантів, номери яких закінчуються на 4 чи 9 – суму кубів всіх розрахованих проміжних значень *f(x)*;

Таблиця 1.1.

Варіант	Завдання	Варіант	Завдання
1.	$y=\frac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000}+65$	**2.**	$y=\sqrt{56x_1+\frac{x_1+x_2+\sin(x_1x_2)}{5-\cos(x_2^{\,2})}}$
3.	$y=\cos(x_1-x_2^{\,2})+31.55x_2x_1^{\,2}$	**4.**	$y=\sqrt[5]{0.1x_1\sin x_2\cos x_1^{\,2}+55x_1x_2}$
5.	$y=\frac{6-\cos(3+x_1)}{34-9x_2^{\,3}+x_2}$	**6.**	$y=\sqrt{\frac{\cos(2x_2)+x_1/x_2}{16x_2x_1}}$
7.	$y=\cos^4\left(x_1-\sqrt{\frac{x_2}{x_1+53x_2^{\,2}}}\right)$	**8.**	$y=\cos\left(\sqrt{x_2}+34x_1\right)-4\sin(x_2)$
9.	$y=23\sin^2(x_1^{\,3}x_2^{\,5})+2x_1+\cos(x_1x_2)$	**10.**	$y=\sin^2\left(x_1\frac{x_2}{x_1+53x_2^{\,2}}\right)$
11.	$y=\sqrt{56+\frac{x_1+x_2+\sin(x_1x_2)}{5\cos(x_2^{\,2})}}$	**12.**	$y=\frac{3x_2-x_1^2}{\cos^3\left(\exp\left(\frac{x_1+2x_2+9}{0.37}\right)\right)}$
13.	$y=45x_1\sin x_2+\sqrt{9x_2x_1^{\,3}}$	**14.**	$y=\cos\left(\sqrt{x_2}+34\cdot\sin(x_1)\right)-4\sin(x_2)$
15.	$y=\frac{1}{4+x_2}\cdot\sqrt{\cos^2\left(\exp\frac{x_2}{x_1}\right)}$	**16.**	$y=\sqrt{\frac{x_2^{\,2}+x_1/x_2}{\cos(x_1^{\,3}x_2^{\,5})+2x_1}}$
17.	$y=\frac{\cos^3(x_1)+45+x_2}{x_1^{\,13}+\cos(x_2)}$	**18.**	$y=0.1x_1\sin x_2\cos x_1^{\,4}+55$
19.	$y=\frac{5\sqrt{x_1^{\,3}+x_2^{\,5}}-\cos(x_2)}{\sin(x_1)}$	**20.**	$y=\frac{\sin^3(x_1)+45+x_2}{2x_1^{\,4}+4x_2}$
21.	$y=\frac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000\cdot\sqrt{x_1+x_2^{\,5}}}+65$	**22.**	$y=\frac{\cos^2\left(\lg_{10}\frac{x_2}{x_1}\right)}{45+x_2}$
23.	$y=23\cos^2(x_1^{\,3}x_2^{\,5})+2x_1$	**24.**	$y=\lg_{10}(x_1x_2^{\,2})+45\sin(x_1+x_2)$
25.	$y=\frac{\sqrt{\cos^3(x_1)+x_2}}{x_1^{\,13}+\sqrt[3]{\cos(x_2)}}$	**26.**	$y=\sqrt{\frac{x_2^{\,2}+x_1/x_2}{16x_2x_1}}$
27.	$y=\sin\left(x_1-x_2^{\,3}+\sqrt{x_1}\right)-1.3x_1^{\,3}$	**28.**	$y=\frac{\ln(x_2)}{\sqrt[5]{0.6x_1\sin x_2\cos x_1^{\,4}}}$
29.	$y=\frac{4\sin(3+x_1x_2)}{34-9x_2^{\,3}}$	**30.**	$y=\cos^3\left(\frac{x_1+2x_2+9}{0.666}\right)+x_2^7$

Лабораторна робота №2

Тема. Форматування виводу.

Мета. Ознайомлення принципами форматування рядків у C#; ознайомлення з описувачами стандартних форматів та рядками форматування стандартних числових форматів.

Завдання

Розробити консольний застосунок, який розраховуватиме заданий математичний вираз з двома змінними $f(x_1, x_2) = x_1^2 + e^{x_2}$ в певному діапазоні для обох змінних. Програма повинна забезпечувати ввід мінімальних $x1_{min}$, $x2_{min}$ максимальних $x1_{max}$, $x2_{max}$ значень, а також кроку зміни *dx1, dx2*. Всі розраховані значення виразу слід послідовно вивести на екран.

Виконання завдання

1. В середовищі Microsoft Visual Studio створити новий проект консольного застосунку. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон ConsoleApplication у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab02) та його розташування на диску.
2. Продумати алгоритм програми. Алгоритм для виконання завдання такий:
   1) Ввести *x1Min*, *x1Max*, *dx1*, *x2Min*, *x2Max*, *dx2*.
   2) Тимчасовій змінній *x1* присвоїти значення *x1Min*.
   3) У циклі, доки *x1* <= *x1Max*:
      - Тимчасовій змінній *x2* присвоїти значення *x2Min*
      - У циклі, доки *x2* <= *x2Max:*
        - Розрахувати значення функції *y = f(x1, x2)*;
        - Вивести на екран: *x1*, *x2* та *y*;
        - Збільшити значення *x2* на *dx2*;
      - Збільшити значення *x1* на *dx1*;
3. У редакторі коду Microsoft Visual Studio у функцію Main() записати код з лістингу 2.1.
4. Запустити застосунок (F5 в середовищі Microsoft Visual Studio) і перевірити його роботу. Приклад виконання розробленої програми приведено на рис. 2.1.

Лістинг 2.1

```
Console.Write("Введіть початкове значення X1min: ");
string sx1Min = Console.ReadLine();
double x1Min = Double.Parse(sx1Min);

Console.Write("Введіть кінцеве значення X1max: ");
string sx1Max = Console.ReadLine();
double x1Max = double.Parse(sx1Max);

Console.Write("Введіть приріст dX1: ");
string sdx1 = Console.ReadLine();
double dx1 = double.Parse(sdx1);
```

```
Console.Write("Введіть початкове значення X2min: ");
string sx2Min = Console.ReadLine();
double x2Min = Double.Parse(sx2Min);

Console.Write("Введіть кінцеве значення X2max: ");
string sx2Max = Console.ReadLine();
double x2Max = double.Parse(sx2Max);

Console.Write("Введіть приріст dX2: ");
string sdx2 = Console.ReadLine();
double dx2 = double.Parse(sdx2);

double y;

double x1 = x1Min;
double x2;

while (x1 <= x1Max)
{
    x2 = x2Min;
    while (x2 <= x2Max)
    {
        y = Math.Pow(x1, 2) + Math.Exp(x2);
        Console.WriteLine(
          "x1 = {0:#.##}\tx2 = {1:#.##}\t\ty = {2:#.##}", x1, x2, y);

        x2 += dx2;
    }

    x1 += dx1;
}

Console.ReadKey();
```

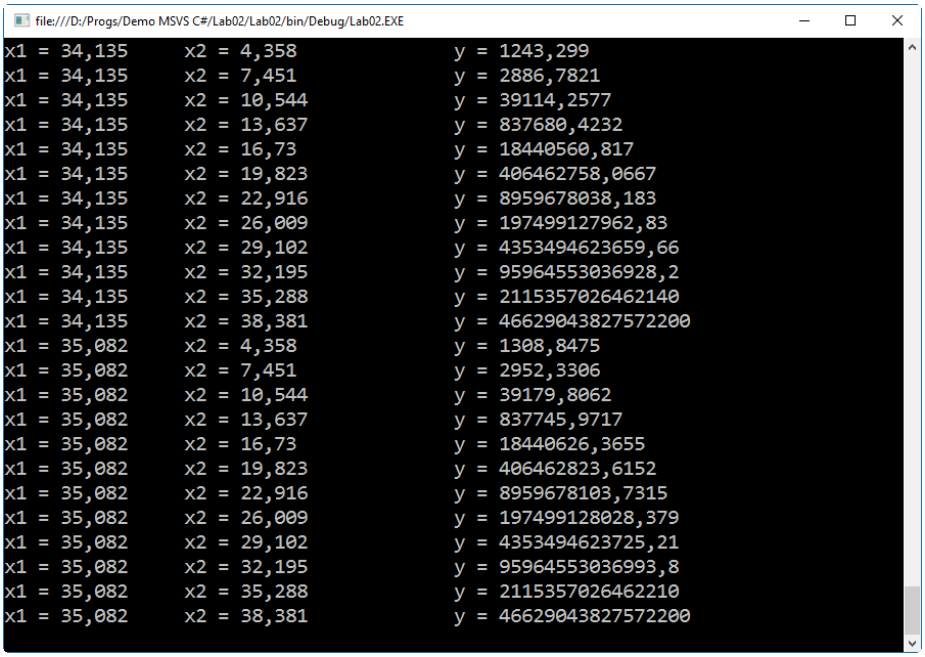


Рис. 2.1. Вивід розробленого застосунку у консоль

Завдання для самостійного опрацювання

1. Вирішити розглянуте раніше завдання для заданої функції $f(x_1, x_2)$ згідно варіанту.
2. Додатково до попереднього завдання, обчислити та вивести на екран:
   - для варіантів, номери яких закінчуються на 0 чи 5 – суму всіх додатних розрахованих проміжних значень $f(x_1, x_2)$;
   - для варіантів, номери яких закінчуються на 1 чи 6 – добуток всіх від'ємних розрахованих проміжних значень $f(x_1, x_2)$;
   - для варіантів, номери яких закінчуються на 2 чи 7 – суму додатних синусів всіх розрахованих проміжних значень $f(x_1, x_2)$;
   - для варіантів, номери яких закінчуються на 3 чи 8 – добуток від'ємних косинусів всіх розрахованих проміжних значень $f(x_1, x_2)$;
   - для варіантів, номери яких закінчуються на 4 чи 9 – суму додатних кубів всіх розрахованих проміжних значень $f(x_1, x_2)$;
3. При виводі числових значень на екран:
   - для варіантів, номери яких закінчуються на 0...3 – виводити 4 знаки після коми;
   - для варіантів, номери яких закінчуються на 4...6 – виводити 3 знаки після коми;
   - для інших варіантів – виводити значення в експоненційному форматі.

Таблиця 2.1.

Варіант	Завдання	Варіант	Завдання
1.	$y = \frac{\sin^3(x_1) + 45 + x_2}{2x_1^4 + 4x_2}$	**2.**	$y = 0.1x_1 \sin x_2 \cos x_1^4 + 55$
3.	$y = \sqrt{\frac{x_2^2 + x_1 / x_2}{\cos(x_1^3 x_2^5) + 2x_1}}$	**4.**	$y = \cos\left(\sqrt{x_2} + 34 \cdot \sin(x_1)\right) - 4\sin(x_2)$
5.	$y = \frac{3x_2 - x_1^2}{\cos^3\left(\frac{x_1 + 2x_2 + 9}{0.37}\right)}$	**6.**	$y = \cos^3\left(\exp\left(\frac{x_1 + 2x_2 + 9}{0.666}\right)\right)$
7.	$y = \frac{\ln(x_2)}{\sqrt[5]{0.6x_1 \sin x_2 \cos x_1^4}}$	**8.**	$y = \sqrt{\frac{x_2^2 + x_1 / x_2}{16x_2x_1}}$
9.	$y = 45\sin\left(x_1 + x_2 + \lg_{10}(x_1 x_2^2)\right)$	**10.**	$y = \frac{\cos^2\left(\lg_{10}\frac{x_2}{x_1}\right)}{45 + x_2}$
11.	$y = \frac{5\sqrt{x_1^3 + x_2^5 - \cos(x_2)}}{\exp(x_1)}$	**12.**	$y = \frac{\cos^3(x_1) + 45 + x_2}{x_1^{13} + \cos(x_2)}$
13.	$y = \frac{1}{4 + x_2} \cdot \sqrt{\cos^2\left(\frac{x_2}{x_1}\right)}$	**14.**	$y = 45x_1 \sin x_2 + \sqrt{9x_2x_1^3}$
15.	$y = \sqrt{56 + \frac{x_1 + x_2 + \sin(x_1x_2)}{5\cos(x_2^2)}}$	**16.**	$y = 23\sin^2(x_1^3 x_2^5) + 2x_1 + \cos(x_1x_2)$

Варіант	Завдання	Варіант	Завдання
17.	$y=\cos^4\left(x_1-\sqrt{\frac{x_2}{x_1+53x_2^{\,2}}}\right)$	**18.**	$y=\frac{6-\cos(3+x_1)}{34-9x_2^{\,3}+x_2}$
19.	$y=\exp\left(x_1-x_2^{\,2}\right)+31.55x_2x_1^{\,2}$	**20.**	$y=\frac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000}+65$
21.	$y=\sin^2\left(x_1\frac{x_2}{x_1+53x_2^{\,2}}\right)$	**22.**	$y=\cos\left(\sqrt{x_2}+34x_1\right)-4\sin(x_2)$
23.	$y=\sqrt{\frac{\cos(2x_2)+x_1/x_2}{16x_2x_1}}$	**24.**	$y=\sqrt[5]{0.1x_1\sin x_2\cos x_1^{\,2}+55x_1x_2}$
25.	$y=\sqrt{56x_1+\frac{x_1+x_2+\sin(x_1x_2)}{5-\cos\left(x_2^{\,2}\right)}}$	**26.**	$y=\frac{4\sin(3+x_1x_2)}{34-9x_2^{\,3}}$
27.	$y=\sin\left(x_1-x_2^{\,3}+\sqrt{x_1}\right)-1.3x_1^{\,3}$	**28.**	$y=\frac{\sqrt{\cos^3(x_1)+x_2}}{x_1^{13}+\sqrt[3]{\cos(x_2)}}$
29.	$y=23\cos^2(x_1^{\,3}x_2^{\,5})+2x_1$	**30.**	$y=\frac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000\cdot\sqrt{x_1+x_2^{\,5}}}+65$

Лабораторна робота №3

Тема. Керування потоком інструкцій програми: умови та цикли.

Мета. Вивчення принципів використання інструкцій для керування потоком виконання програми.

Завдання

Розробити консольний застосунок, який розраховуватиме інтеграл неперервної функції з однією змінною $f(x)=x^2$ на заданому відрізку інтегрування [*a, b*]. Програма повинна забезпечувати ввід початку і кінця відрізку інтегрування *a* та *b*, а також кількості інтервалів інтегрування *n*. Результат обчислення слід вивести на екран.

Пояснення до завдання

Інтеграл функції можна уявити як площу фігури, обмеженої зверху графіком функції, знизу – віссю абсцис, а зліва і справа – вертикальними лініями, які проходять через краї відрізку інтегрування *a* та *b* (рис. 3.1).

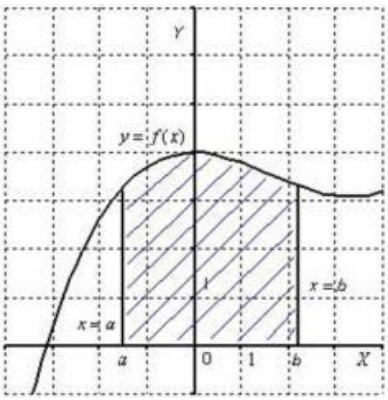


Рис. 3.1. Площа під кривою

Якщо цю фігуру розділити на n вертикальних смуг, то при $n \to \infty$ інтеграл дорівнюватиме сумі площ цих смуг:

$$S = \int_a^b f(x)\cdot dx = \lim_{n\to\infty}\sum_{i=1}^{n} f(x_i)\cdot dx\,.$$

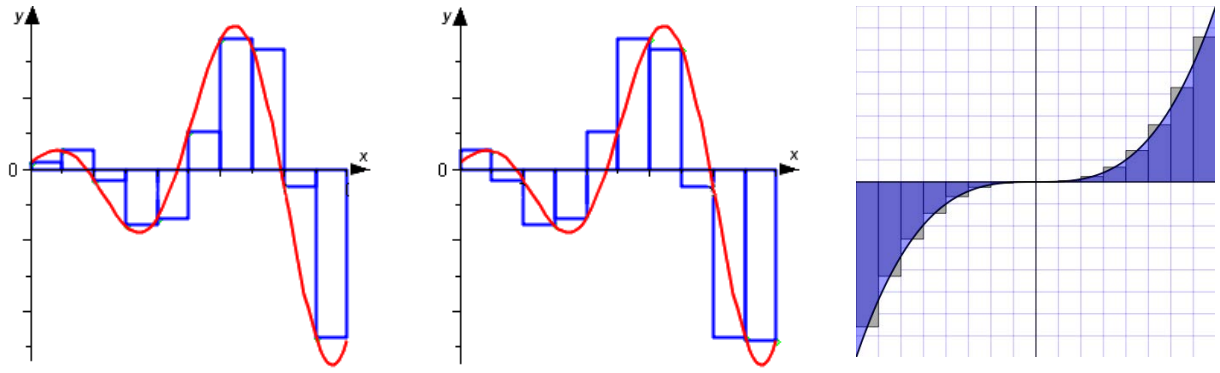
При цьому є кілька підходів до формування цих елементарних смуг, які відрізняються способом вибору двох верхніх точок (дві нижні точки завжди розташовані на осі абсцис).

Перший підхід передбачає, що елементарні смуги є прямокутними. Його називають *методом прямокутників*. Для чисельного обчислювання інтегралів методом прямокутників проміжок інтегрування [*a*, *b*] поділяють на *n* рівних частин. Задану функцію *f(x)* на кожному відрізку замінюють на прямі лінії, паралельні до осі абсцис. При цьому криволінійна функція замінюється на *n* прямокутників. Інтеграл розраховують як суму площ *n* прямокутників, отриманих шляхом розбиття відрізка інтегрування [*a*, *b*] на *n* рівних частин. Чим більша кількість ділянок *n*, на які ділимо інтервал [*a*, *b*], тим точніше сукупність прямокутників відтворює функцію, і тим точнішим буде інтеграл.

Є три підвиди методу прямокутників, які визначають, як саме формуватиметься верхня сторона кожної ділянки:

1) *Метод лівих прямокутників*. Верхня сторона ділянки проходить через точку перетину лівої сторони ділянки з функцією (рис. 3.2, а).

2) *Метод правих прямокутників*. Верхня сторона ділянки проходить через точку перетину правої сторони ділянки з функцією (рис. 3.2, б).
3) *Метод центральних прямокутників*. Верхня сторона ділянки проходить через точку перетину вертикальної осі ділянки з функцією (рис. 3.2, в).



а б в

Рис.3.2. Ілюстрація методу лівих прямокутників (а), правих прямокутників (б), та центральних прямокутників (в)

Ще один підхід до формування елементарних смуг передбачає, що вони мають форму трапеції. Його називають *методом трапецій*. Фактично цей підхід полягає у заміні кривої підінтегральної функції на ламану. Проміжок [*a*, *b*] розбивають на *n* рівних частин, та сполучають прямими лініями значення функцій на кінцях відрізків, тобто, площу криволінійної функції наближено замінюємо на суму площ *n* трапецій (рис. 3.3). Чим більша кількість ділянок *n*, на які розділено інтервал [*a*, *b*], тим точніше сукупність трапецій відтворює функцію, і тим точнішим буде інтеграл.

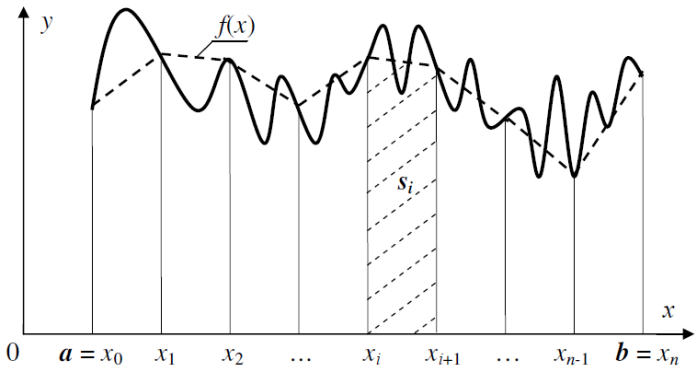


Рис. 3.3. Ілюстрація методу трапецій

Виконання завдання

1. В середовищі Microsoft Visual Studio створити новий проект консольного застосунку. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон ConsoleApplication у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab03) та його розташування на диску.
2. Продумати алгоритм програми. Алгоритм для виконання завдання такий:

1) Ввести *a*, *b*, *n*.
2) Розрахувати величину кроку інтегрування *dx*.
3) Тимчасовій змінній *Intgrl* присвоїти значення *0*.
4) У циклі, для *i = 0...n-1*:
    - Розрахувати початкову та кінцеву межу *i*-ої поточної ділянки;
    - Розрахувати значення функції *f(x)* на межах поточної ділянки;
    - Розрахувати площу поточної ділянки і додати її до змінної *Intgrl*;
5) Вивести на екран результат розрахунку інтегралу:
6) Перепитати користувача, чи здійснити ще один розрахунок
    - Якщо так, перейти до п.1;

3. У редакторі коду Microsoft Visual Studio в класі Program (безпосередньо перед функцією Main()) записати код з лістингу 3.1.
4. У функцію Main() записати код з лістингу 3.2.
5. Запустити застосунок (F5 в середовищі Microsoft Visual Studio) і перевірити його роботу. Приклад виконання розробленої програми приведено на рис. 3.4.

Лістинг 3.1

```
static double Function(double x)
{
    return x * x;
}
```

Лістинг 3.2

```
StartOfCalculations:

Console.Write("Введіть початок відрізку інтегрування a: ");
string sa = Console.ReadLine();
double a = double.Parse(sa);

Console.Write("Введіть кінець відрізку інтегрування b: ");
string sb = Console.ReadLine();
double b = double.Parse(sb);

Console.Write("Введіть кількість ділянок n: ");
string sn = Console.ReadLine();
double n = double.Parse(sn);

double dx = (b - a) / n;
double y1, y2;
double x1, x2;
double Intgrl = 0;


for (int i = 0; i < n; i++)
{
    x1 = a + i * dx;
    x2 = x1 + dx;
    y1 = Function(x1);
    y2 = Function(x2);

    // Обчислення інтегралу методом центральних прямокутників
    Intgrl += (y1 + y2) / 2 * dx;
}

Console.WriteLine(
  "Інтеграл функції на відрізку [{0}, {1}] становить {2:0.0000}", a, b,
Intgrl);
```

```
Console.Write("Повторити розрахунок (y - так) ? ");
ConsoleKeyInfo pressedKey = Console.ReadKey();
Console.WriteLine();
if (pressedKey.Key == ConsoleKey.Y)
{
    Console.WriteLine();
    goto StartOfCalculations;
}
```

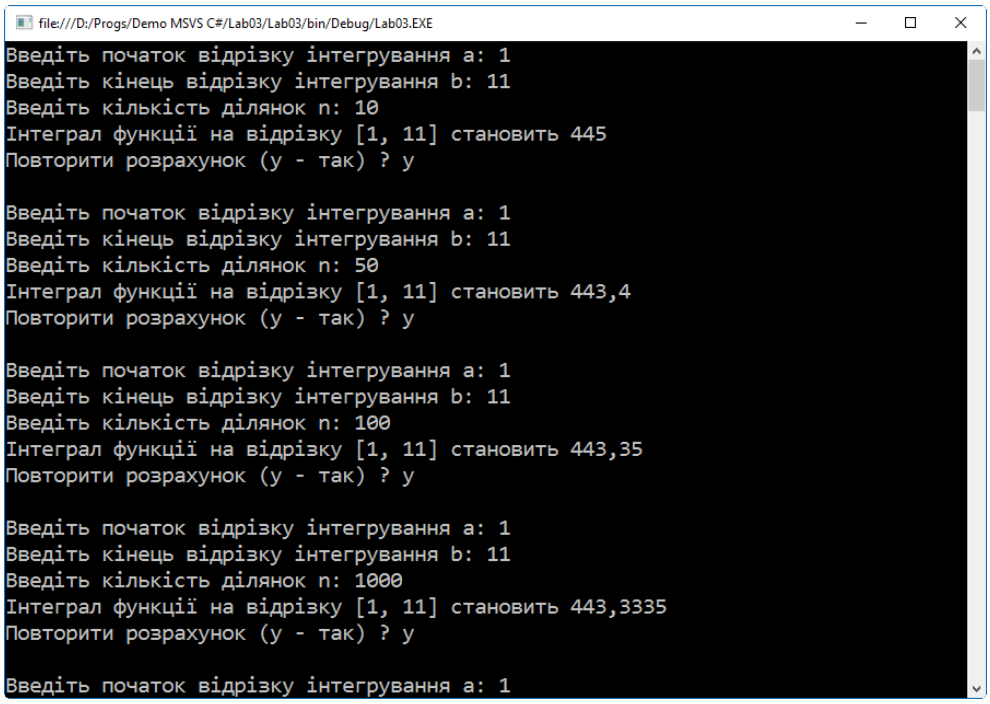


Рис. 3.4. Вивід розробленого застосунку у консоль

Завдання для самостійного опрацювання

1. Вирішити розглянуте раніше завдання для заданої функції *f(x)* згідно варіанту. У виразах прийняти, що $x_1=2x$, $x_2=5x$. При цьому:
    - для варіантів, номери яких закінчуються на 0...3 – використати метод лівих прямокутників; при виводі результату виводити 4 знаки після коми;
    - для варіантів, номери яких закінчуються на 4...6 – використати метод правих прямокутників; при виводі результату виводити 5 знаків після коми;
    - для варіантів, номери яких закінчуються на 7...9 – використати метод трапецій; при виводі результату виводити 6 знаків після коми;

Таблиця 3.1.

Варіант	Завдання	Варіант	Завдання
1.	$y=\frac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000}+65$	**2.**	$y=\cos\left(x_1-x_2^{\,2}\right)+31.55x_2x_1^{\,2}$
3.	$y=\frac{6-\cos(3+x_1)}{34-9x_2^{\,3}+x_2}$	**4.**	$y=\cos^4\left(x_1-\sqrt{\frac{x_2}{x_1+53x_2^{\,2}}}\right)$
5.	$y=23\sin^2(x_1^{\,3}x_2^{\,5})+2x_1+\cos(x_1x_2)$	**6.**	$y=\sqrt{56+\frac{x_1+x_2+\sin(x_1x_2)}{5\cos(x_2^{\,2})}}$
7.	$y=45x_1\sin x_2+\sqrt{9x_2x_1^{\,3}}$	**8.**	$y=\frac{1}{4+x_2}\cdot\sqrt{\cos^2\left(\exp\frac{x_2}{x_1}\right)}$
9.	$y=\frac{\cos^3(x_1)+45+x_2}{x_1^{\,13}+\cos(x_2)}$	**10.**	$y=\frac{5\sqrt{x_1^{\,3}+x_2^{\,5}}-\cos(x_2)}{\sin(x_1)}$
11.	$y=\frac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000\cdot\sqrt{x_1+x_2^{\,5}}}+65$	**12.**	$y=23\cos^2(x_1^{\,3}x_2^{\,5})+2x_1$
13.	$y=\frac{\sqrt{\cos^3(x_1)+x_2}}{x_1^{\,13}+3/\cos(x_2)}$	**14.**	$y=\sin\left(x_1-x_2^{\,3}+\sqrt{x_1}\right)-1.3x_1^{\,3}$
15.	$y=\frac{4\sin(3+x_1x_2)}{34-9x_2^{\,3}}$	**16.**	$y=\sqrt{56x_1+\frac{x_1+x_2+\sin(x_1x_2)}{5-\cos(x_2^{\,2})}}$
17.	$y=\sqrt[5]{0.1x_1\sin x_2\cos x_1^{\,2}+55x_1x_2}$	**18.**	$y=\sqrt{\frac{\cos(2x_2)+x_1/x_2}{16x_2x_1}}$
19.	$y=\cos\left(\sqrt{x_2}+34x_1\right)-4\sin(x_2)$	**20.**	$y=\sin^2\left(x_1\frac{x_2}{x_1+53x_2^{\,2}}\right)$
21.	$y=\frac{3x_2-x_1^2}{\cos^3\left(\exp\left(\frac{x_1+2x_2+9}{0.37}\right)\right)}$	**22.**	$y=\cos\left(\sqrt{x_2}+34\cdot\sin(x_1)\right)-4\sin(x_2)$
23.	$y=\sqrt{\frac{x_2^{\,2}+x_1/x_2}{\cos(x_1^{\,3}x_2^{\,5})+2x_1}}$	**24.**	$y=0.1x_1\sin x_2\cos x_1^{\,4}+55$
25.	$y=\frac{\sin^3(x_1)+45+x_2}{2x_1^{\,4}+4x_2}$	**26.**	$y=\frac{\cos^2\left(\lg_{10}\frac{x_2}{x_1}\right)}{45+x_2}$
27.	$y=\lg_{10}(x_1x_2^{\,2})+45\sin(x_1+x_2)$	**28.**	$y=\sqrt{\frac{x_2^{\,2}+x_1/x_2}{16x_2x_1}}$
29.	$y=\frac{\ln(x_2)}{\sqrt[5]{0.6x_1\sin x_2\cos x_1^{\,4}}}$	**30.**	$y=\cos^3\left(\frac{x_1+2x_2+9}{0.666}\right)+x_2^7$

Лабораторна робота №4

Тема. Робота з масивом.

Мета. Вивчення засобів C# для роботи з масивами.

Завдання

Розробити консольний застосунок, який створить масив та заповнить його значеннями функції $f(x)=x^2$ в певному діапазоні. Програма повинна сортувати масив за спаданням значень та розрахувати мінімальне *aMin*, максимальне *aMax* та середнє *aAvg* значення масиву. Відсортований масив та знайдені *aMin*, *aMax, aAvg* слід вивести на екран.

Виконання завдання

1. В середовищі Microsoft Visual Studio створити новий проект консольного застосунку. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон ConsoleApplication у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab04) та його розташування на диску.
2. Продумати алгоритм програми. Алгоритм для виконання завдання такий:
    1) Оголосити і створити масив *arr*, заповнити його значеннями функції *y = f(x)*.
    2) Відсортувати масив *arr* за спаданням значень.
    3) Вивести масив *arr* на екран.
    4) Знайти мінімальне *aMin*, максимальне *aMax* та середнє *aAvg* значення масиву.
    5) Вивести знайдені значення *aMin*, *aMax, aAvg* на екран.
3. У класі Program оголосити константи та метод для розрахунку функції (лістинг 4.1).
4. У функцію Main() записати код з лістингу 4.2.
5. Запустити застосунок (F5 в середовищі Microsoft Visual Studio) і перевірити його роботу. Приклад виконання розробленої програми приведено на рис. 4.1.

Лістинг 4.1

```
const double StartX = 10.3;
const double dX = 0.7;

static double Function(double x)
{
    return x * x;
}
```

Лістинг 4.2

```
double[] arr = new double[10];

double x = StartX;
for (int i = arr.GetLowerBound(0); i <= arr.GetUpperBound(0); i++)
{
```

```
    arr[i] = Function(x);
    x += dX;
}

Array.Sort(arr);
Array.Reverse(arr);

Console.WriteLine("Відсортовані за спаданням значення масиву: ");

for (int i = arr.GetLowerBound(0); i <= arr.GetUpperBound(0); i++)
{
    Console.WriteLine("arr[{0:00}] = {1:0.0000}", i, arr[i]);
}

double aMin = arr[arr.GetUpperBound(0)];
double aMax = arr[arr.GetLowerBound(0)];

double aAvg = 0;
for (int i = arr.GetLowerBound(0); i <= arr.GetUpperBound(0); i++)
{
    aAvg += arr[i];
}
aAvg = aAvg / arr.GetLength(0);

Console.WriteLine("Мінімальне значення масиву: {0:0.0000}", aMin);
Console.WriteLine("Максимальне значення масиву: {0:0.0000}", aMax);
Console.WriteLine("Середнє значення масиву: {0:0.0000}", aAvg);

Console.ReadKey(true);
```

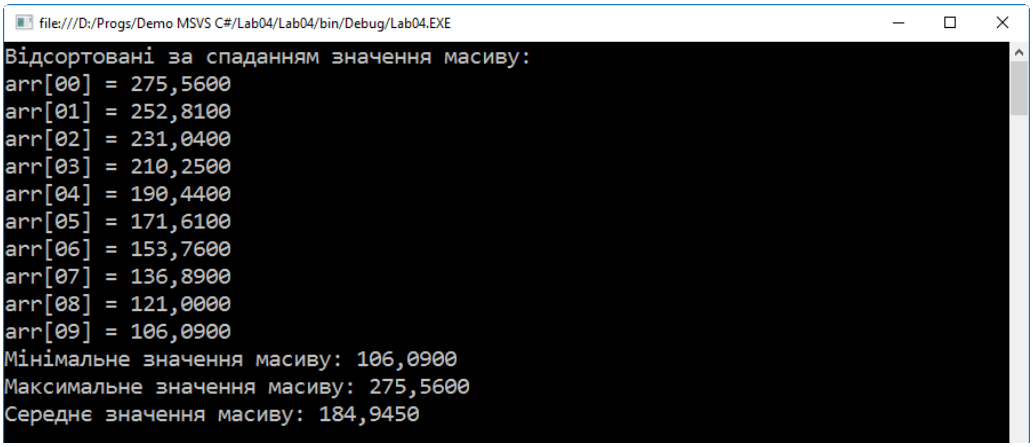


Рис. 4.1. Вивід розробленого застосунку у консоль

Завдання для самостійного опрацювання

1. Вирішити розглянуте раніше завдання для заданої функції *f(x)* згідно варіанту. У виразах прийняти, що x_1=2.76x, x_2=0.5x.
2. Додатково до попереднього завдання, обчислити та вивести на екран значення R згідно варіанту.

Таблиця 4.1.

Варіант	Завдання	Варіант	Завдання
1.	$y=\dfrac{3x_2-x_1^2}{\cos^3\left(\exp\left(\dfrac{x_1+2x_2+9}{0.37}\right)\right)}$	**2.**	$y=\cos\left(\sqrt{x_2}+34\cdot\sin(x_1)\right)-4\sin(x_2)$
3.	$y=\sqrt{\dfrac{x_2^{\,2}+x_1/x_2}{\cos(x_1^{\,3}x_2^{\,5})+2x_1}}$	**4.**	$y=0.1x_1\sin x_2\cos x_1^{\,4}+55$
5.	$y=\dfrac{\sin^3(x_1)+45+x_2}{2x_1^{\,4}+4x_2}$	**6.**	$y=\dfrac{\cos^2\left(\lg_{10}\dfrac{x_2}{x_1}\right)}{45+x_2}$
7.	$y=\lg_{10}(x_1x_2^{\,2})+45\sin(x_1+x_2)$	**8.**	$y=\sqrt{\dfrac{x_2^{\,2}+x_1/x_2}{16x_2x_1}}$
9.	$y=\dfrac{\ln(x_2)}{\sqrt[5]{0.6x_1\sin x_2\cos x_1^{\,4}}}$	**10.**	$y=\cos^3\left(\dfrac{x_1+2x_2+9}{0.666}\right)+x_2^7$
11.	$y=\sqrt{56x_1+\dfrac{x_1+x_2+\sin(x_1x_2)}{5-\cos(x_2^{\,2})}}$	**12.**	$y=\sqrt[5]{0.1x_1\sin x_2\cos x_1^{\,2}+55x_1x_2}$
13.	$y=\sqrt{\dfrac{\cos(2x_2)+x_1/x_2}{16x_2x_1}}$	**14.**	$y=\cos\left(\sqrt{x_2}+34x_1\right)-4\sin(x_2)$
15.	$y=\sin^2\left(x_1\dfrac{x_2}{x_1+53x_2^{\,2}}\right)$	**16.**	$y=\dfrac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000\cdot\sqrt{x_1+x_2^{\,5}}}+65$
17.	$y=23\cos^2(x_1^{\,3}x_2^{\,5})+2x_1$	**18.**	$y=\dfrac{\sqrt{\cos^3(x_1)+x_2}}{x_1^{\,13}+\sqrt[3]{\cos(x_2)}}$
19.	$y=\sin\left(x_1-x_2^{\,3}+\sqrt{x_1}\right)-1.3x_1^{\,3}$	**20.**	$y=\dfrac{4\sin(3+x_1x_2)}{34-9x_2^{\,3}}$
21.	$y=\dfrac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000}+65$	**22.**	$y=\cos\left(x_1-x_2^{\,2}\right)+31.55x_2x_1^{\,2}$
23.	$y=\dfrac{6-\cos(3+x_1)}{34-9x_2^{\,3}+x_2}$	**24.**	$y=\cos^4\left(x_1-\sqrt{\dfrac{x_2}{x_1+53x_2^{\,2}}}\right)$
25.	$y=23\sin^2(x_1^{\,3}x_2^{\,5})+2x_1+\cos(x_1x_2)$	**26.**	$y=\sqrt{56+\dfrac{x_1+x_2+\sin(x_1x_2)}{5\cos(x_2^{\,2})}}$
27.	$y=45x_1\sin x_2+\sqrt{9x_2x_1^{\,3}}$	**28.**	$y=\dfrac{1}{4+x_2}\cdot\sqrt{\cos^2\left(\exp\dfrac{x_2}{x_1}\right)}$
29.	$y=\dfrac{\cos^3(x_1)+45+x_2}{x_1^{\,13}+\cos(x_2)}$	**30.**	$y=\dfrac{5\sqrt{x_1^{\,3}+x_2^{\,5}}-\cos(x_2)}{\sin(x_1)}$

Таблиця 4.2.

Остання цифра номера варіанту	R
0	Кількість елементів масиву, менших за середнє значення
1	Сума елементів масиву, менших за середнє значення
2	Кількість елементів масиву, більших за середнє значення
3	Сума елементів масиву, більших за середнє значення
4	Кількість елементів масиву, які знаходяться в діапазоні *aMin…aMin+10%aMin*
5	Сума елементів масиву, які знаходяться в діапазоні *aMin…aMin+10%aMin*
6	Кількість елементів масиву, які знаходяться в діапазоні *aMax-10%aMax…aMax*
7	Сума елементів масиву, які знаходяться в діапазоні *aMax-10%aMax…aMax*
8	Кількість елементів масиву, які знаходяться в діапазоні *aAvg-10%aAvg…aAvg+10%aAvg*
9	Суму елементів масиву, які знаходяться в діапазоні *aAvg-10%aAvg…aAvg+10%aAvg*

Лабораторна робота №5

Тема. Проектування простого класу.

Мета. Вивчення принципів розробки та використання класів; вивчення принципів роботи з об'єктами.

Завдання

Розробити консольний застосунок, у якому описати клас, що представляє об'єкт "місто". Клас має містити не менше 7 полів різних типів і один метод. Програма має забезпечити ввід даних для одного екземпляру, створити об'єкт розробленого класу, провести дії, представлені розробленим методом класу, і вивести результати на екран.

Виконання завдання

1. В середовищі Microsoft Visual Studio створити новий проект консольного застосунку. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон ConsoleApplication у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab05) та його розташування на диску.
2. Продумати алгоритм програми. Для цього слід:
    1) Проаналізувати предметну область. Потрібно виділити важливі властивості об'єкта – вони будуть представлені полями. Виявити, яку дію слід виконати над даними об'єкта – вона буде реалізована методом класу. Скласти перелік полів (з типами) та методів.
    2) Описати у програмі клас (поля, методи та реалізація методів).
    3) Далі у програмі слід:
        - Скласти програмні інструкції для введення даних, які будуть записані у поля об'єкта – екземпляра класу.
        - Створити об'єкт – екземпляр розробленого класу.
        - Записати введені дані у відповідні поля об'єкта.
        - Виконати метод, який обробляє дані об'єкта
        - Вивести на екран всі поля об'єкта та результат роботи метода.
3. Для заданої у завданні предметної області ("місто") вважаємо важливими такі "властивості":
    - Назва міста
    - Назва країни
    - Назва регіону (області, провінції)
    - Населення (кількість)
    - Річний дохід міста
    - Площа
    - Чи має місто порт

- Чи має місто аеропорт

Над даними об'єкта слід виконати таку дію:

- Обчислити середній річний дохід, який припадає на одного мешканця

4. У просторі імен проекту записати оголошення класу з лістингу 5.1.
5. У функцію Main() записати код з лістингу 5.2.
6. Запустити застосунок (F5 в середовищі Microsoft Visual Studio) і перевірити його роботу. Приклад виконання розробленої програми приведено на рис. 5.1.

Лістинг 5.1

```
class Town
{
    public string Name;
    public string Country;
    public string Region;
    public int Population;
    public double YearIncome;
    public double Square;
    public bool HasPort;
    public bool HasAirport;

    public double GetYearIncomePerInhabitant()
    {
        return YearIncome / Population;
    }
}
```

Лістинг 5.2

```
Console.Write("Введіть назву міста: ");
string sName = Console.ReadLine();

Console.Write("Введіть назву країни: ");
string sCountry = Console.ReadLine();

Console.Write("Введіть назву регіону: ");
string sRegion = Console.ReadLine();

Console.Write("Введіть кількість населення: ");
string sPopulation = Console.ReadLine();

Console.Write("Введіть річний дохід: ");
string sYearIncome = Console.ReadLine();

Console.Write("Введіть площу, кв. км: ");
string sSquare = Console.ReadLine();

Console.Write("Чи є у місті порт? (y-так, n-ні): ");
ConsoleKeyInfo keyHasPort = Console.ReadKey();
Console.WriteLine();

Console.Write("Чи є у місті аеропорт? (y-так, n-ні): ");
ConsoleKeyInfo keyHasAirport = Console.ReadKey();
Console.WriteLine();

Town OurTown = new Town();

OurTown.Name = sName;
OurTown.Country = sCountry;
OurTown.Region = sRegion;
OurTown.Population = int.Parse(sPopulation);
OurTown.YearIncome = double.Parse(sYearIncome);
OurTown.Square = double.Parse(sSquare);
OurTown.HasPort = keyHasPort.Key == ConsoleKey.Y ? true : false;
OurTown.HasAirport = keyHasAirport.Key == ConsoleKey.Y ? true : false;
```

```
double YearIncomePerInhabitant = OurTown.GetYearIncomePerInhabitant();

Console.WriteLine();
Console.WriteLine("-------------------------------------------------");
Console.WriteLine("Дані про об`єкт: ");
Console.WriteLine("-------------------------------------------------");
Console.WriteLine("Назва: " + OurTown.Name);
Console.WriteLine("Країна: " + OurTown.Country);
Console.WriteLine("Регіон: " + OurTown.Region);
Console.WriteLine("Кількість населення: " +
    OurTown.Population.ToString());
Console.WriteLine("Річний дохід: " +
    OurTown.YearIncome.ToString("0.00"));
Console.WriteLine("Площа: " + OurTown.Square.ToString("0.000"));
Console.WriteLine(OurTown.HasPort ? "У місті є порт" :
    "У місті нема порту");
Console.WriteLine(OurTown.HasAirport ? "У місті є аеропорт" :
    "У місті нема аеропорту");
Console.WriteLine();
Console.WriteLine("Середній річний дохід на одного громадянина: " +
YearIncomePerInhabitant.ToString("0.00"));

Console.ReadKey();
```

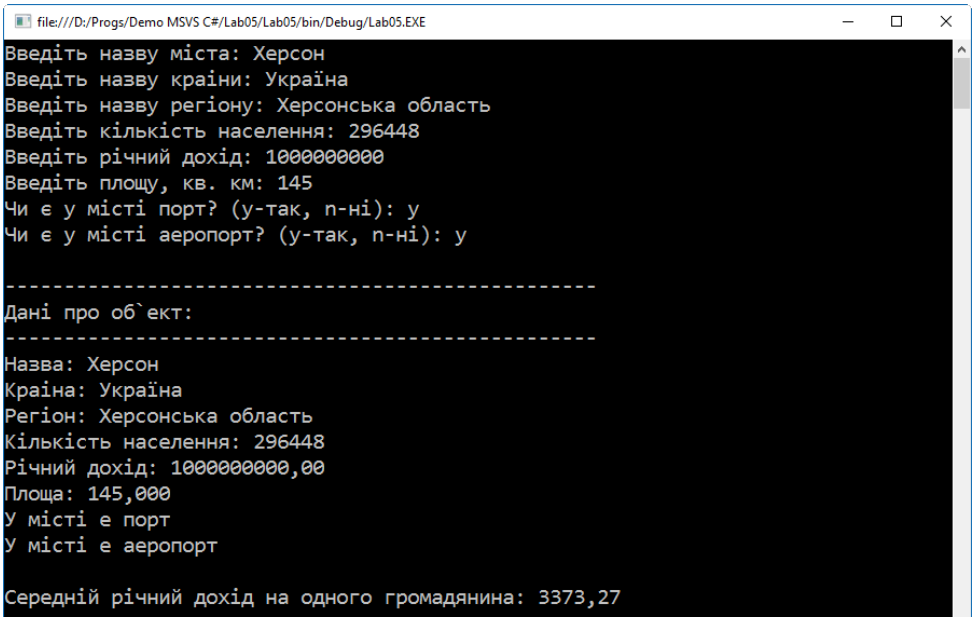


Рис. 5.1. Вивід розробленого застосунку у консоль

Завдання для самостійного опрацювання

Вирішити розглянуте раніше завдання щодо заданої згідно варіанту предметної області. Клас має містити не менше 7 полів та не менше одного методу. Характеристики об’єкта вибрати самостійно, довільно, відповідно до предметної області.

Таблиця 5.1.

Варіант	Завдання	Варіант	Завдання
1.	Комп'ютер	**2.**	Автомобіль
3.	Принтер	**4.**	Літак
5.	Фотоапарат	**6.**	Процесор
7.	Телефон	**8.**	Телевізор
9.	Велосипед	**10.**	Планшет
11.	Абонент телефонної мережі	**12.**	Книга
13.	Студент	**14.**	Персона
15.	Документ (паспорт)	**16.**	Будівля
17.	Планета	**18.**	Футбольна команда
19.	Рослина	**20.**	Тварина
21.	Жорсткі диски до ЕОМ	**22.**	Країна
23.	Область	**24.**	Ріка
25.	Озеро	**26.**	Завод
27.	Університет	**28.**	Фільм
29.	Музичний виконавець	**30.**	Ферма

Лабораторна робота №6

Тема. Розробка бібліотеки класів.

Мета. Вивчення принципів розробки та використання бібліотеки класів; вивчення принципів роботи з масивами та об'єктами; ознайомлення з принципами формування рішень з кількома проектами.

Завдання

Розробити бібліотеку з класом, створеним на основі класу з лабораторної роботи №5. Клас доповнити однією властивістю.

Розробити консольний застосунок, який використовує клас з бібліотеки. Програма має забезпечити ввід даних для масиву об'єктів спроектованого класу, і вивести результати для всіх об'єктів масиву на екран.

Виконання завдання

Розробка бібліотеки

1. В середовищі Microsoft Visual Studio створити новий проект бібліотеки класів. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон ClassLibrary у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab06Lib) та його розташування на диску.
2. Дати назву файлу, в якому описуватимемо клас. Для цього у вікні Solution Explorer викликати контекстне меню на елементі Class1.cs і вибрати команду Rename. Задати назву файлу відповідно до його призначення (наприклад, ClassesLib).
3. У файлі ClassesLib.cs замість автоматично створеного класу Class1 ввести код свого класу з лабораторної роботи №5. Доповнити клас однією властивістю, спроектувавши її відповідно до предметної області. Повний код класу для об'єкта типу "місто" приведено у лістингу 6.1. Клас доповнено властивістю yearIncomePerInhabitant (лише для читання), яка повертає дані, обчислені методом GetYearIncomePerInhabitant() на основі даних об'єкта.
4. Скомпілювати бібліотеку. Для цього у вікні Solution Explorer викликати контекстне меню на елементі проекту бібліотеки (Lab06Lib) та вибрати команду Build. Якщо код не містить синтаксичних помилок, у відповідній папці, де збережено проект бібліотеки, буде створено файл бібліотеки з розширенням dll (в нашому випадку – Lab06Lib.dll). На цьому розробка бібліотеки закінчена.

Розробка застосунку, який використовуватиме бібліотеку

1. У поточному рішенні створити новий проект консольного застосунку. Для цього у вікні Solution Explorer викликати контекстне меню на елементі рішення (Solution 'Lab06Lib') і вибрати команду Add/ New Project. У вікні Add New Project вибрати шаблон

ConsoleApplication у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab06App) та його розташування на диску. Після створення нового проекту у вікні Solution Explorer будуть присутні два проекти: розроблений перед цим Lab06Lib та щойно створений Lab06App (рис. 6.1).

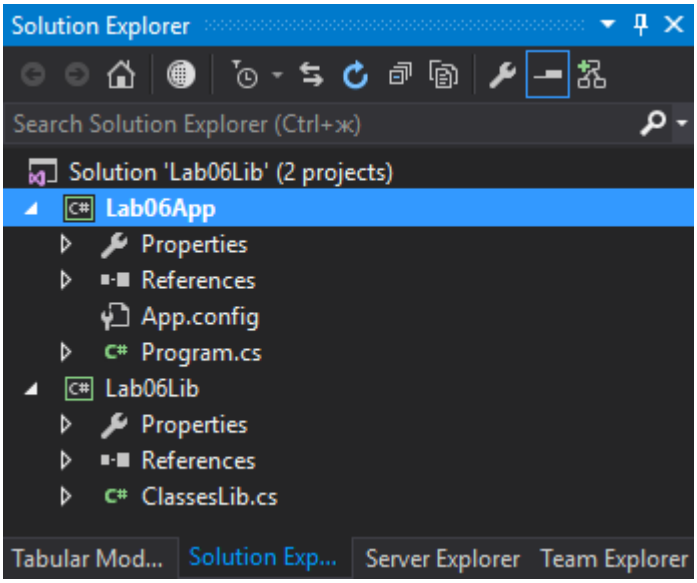


Рис. 6.1.

2. Щоб в новому проекті можна було використовувати типи з бібліотеки, її слід додати до посилань (references) проекту. Для цього у вікні Solution Explorer викликати контекстне меню на елементі References нового проекту (Lab06App) і вибрати команду Add Reference. У вікні Reference Manager (рис. 6.2) перейти у розділ Projects/ Solution. У центральній частині вікна буде перелік доступних посилань на збірки поточного рішення. В нашому випадку буде лише одне посилання – на попередньо розроблену бібліотеку Lab06Lib. Слід відмітити його маркером і натиснути кнопку OK.

> Зауважимо, що додати посилання на бібліотеку можна також безпосередньо вказавши її файл. Наприклад, замість дій, описаних в цьому пункті, у вікні Reference Manager можна натиснути кнопку Browse і вибрати файл Lab06Lib.dll, скомпільований при розробці бібліотеки. Так роблять, якщо проект бібліотеки не входить до складу поточного рішення.

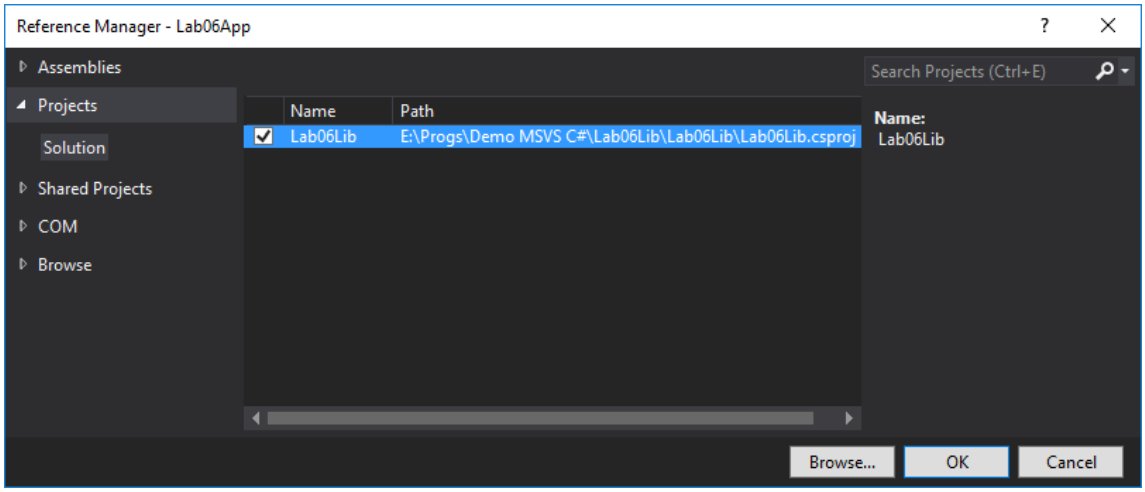


Рис. 6.2. Додання нового посилання

3. Щоб у коді застосунку можна було використовувати типи з під'єднаної бібліотеки за їх скороченими іменами, в верхню частину файла Program.cs слід додати рядок

```
using Lab06Lib;
```

Тут Lab06Lib – простір імен у проекті бібліотеки (див. лістинг 6.1).

4. Продумати алгоритм програми. Алгоритм для виконання завдання такий:

1) Ввести кількість об'єктів для масиву *cntTowns*.
2) Оголосити і створити масив *arrTowns*.
3) У циклі ввести дані для всіх міст масиву, задавши значення всіх полів.
4) У циклі вивести дані всіх об'єктів масиву на екран.

5. У метод Main() записати код з лістингу 6.2.
6. Запустити застосунок (F5 в середовищі Microsoft Visual Studio) і перевірити його роботу. Приклад виконання розробленої програми приведено на рис. 6.3.

Завдання для самостійного опрацювання

Вирішити розглянуте раніше завдання щодо заданої згідно варіанту предметної області (див. завдання до лабораторної роботи №5 – табл. 5.1).

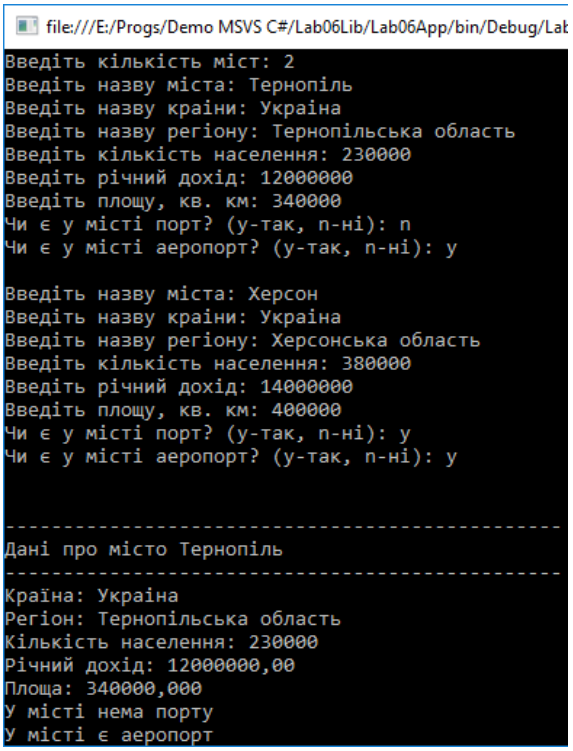


Рис. 6.3. Вивід розробленого застосунку у консоль

Лістинг 6.1

```
public class Town
    {
        public string Name;
        public string Country;
        public string Region;
        public int Population;
        public double YearIncome;
        public double Square;
        public bool HasPort;
        public bool HasAirport;

        public double yearIncomePerInhabitant
        {
            get
            {
                return GetYearIncomePerInhabitant();
            }
        }

        public double GetYearIncomePerInhabitant()
        {
            return YearIncome / Population;
        }
    }
```

Лістинг 6.2

```
Town[] arrTowns;

Console.Write("Введіть кількість міст: ");
int cntTowns = int.Parse(Console.ReadLine());
arrTowns = new Town[cntTowns];

for (int i = 0; i < cntTowns; i++)
{
    Console.Write("Введіть назву міста: ");
```

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

    Console.Write("Введіть назву країни: ");
    string sCountry = Console.ReadLine();

    Console.Write("Введіть назву регіону: ");
    string sRegion = Console.ReadLine();

    Console.Write("Введіть кількість населення: ");
    string sPopulation = Console.ReadLine();

    Console.Write("Введіть річний дохід: ");
    string sYearIncome = Console.ReadLine();

    Console.Write("Введіть площу, кв. км: ");
    string sSquare = Console.ReadLine();

    Console.Write("Чи є у місті порт? (у-так, n-ні): ");
    ConsoleKeyInfo keyHasPort = Console.ReadKey();
    Console.WriteLine();

    Console.Write("Чи є у місті аеропорт? (у-так, n-ні): ");
    ConsoleKeyInfo keyHasAirport = Console.ReadKey();
    Console.WriteLine();

    Console.WriteLine();

    Town theTown = new Town();

    theTown.Name = sName;
    theTown.Country = sCountry;
    theTown.Region = sRegion;
    theTown.Population = int.Parse(sPopulation);
    theTown.YearIncome = double.Parse(sYearIncome);
    theTown.Square = double.Parse(sSquare);
    theTown.HasPort = keyHasPort.Key == ConsoleKey.Y ? true : false;
    theTown.HasAirport =
        keyHasAirport.Key == ConsoleKey.Y ? true : false;

    arrTowns[i] = theTown;
}

foreach (Town t in arrTowns)
{
    Console.WriteLine();
    Console.WriteLine("-----------------------------------------");
    Console.WriteLine("Дані про місто {0}", t.Name);
    Console.WriteLine("-----------------------------------------");
    Console.WriteLine("Країна: " + t.Country);
    Console.WriteLine("Регіон: " + t.Region);
    Console.WriteLine("Кількість населення: " +
        t.Population.ToString());
    Console.WriteLine("Річний дохід: " +
        t.YearIncome.ToString("0.00"));
    Console.WriteLine("Площа: " + t.Square.ToString("0.000"));
    Console.WriteLine(t.HasPort ? "У місті є порт" :
        "У місті нема порту");
    Console.WriteLine(t.HasAirport ? "У місті є аеропорт" :
        "У місті нема аеропорту");
    Console.WriteLine();
    Console.WriteLine("Середній річний дохід на одного громадянина: " +
        t.yearIncomePerInhabitant.ToString("0.00"));
}

Console.ReadKey();
```

Лабораторна робота №7

Тема. Розробка простого застосунку Windows Forms.

Мета. Вивчення принципів розробки застосунків з графічним інтерфейсом користувача на основі технології Windows Forms.

Завдання

Розробити одновіконний застосунок на основі технології Windows Forms для розрахунку арифметичного виразу на основі функції з двома змінними $f(x_1,x_2)$. Програма повинна забезпечувати ввід значень x_1 та x_2, розрахунок значення функції $f(x_1,x_2)=X_1 \cdot X_2$, вивід результату розрахунку у вікно та очищення полів для наступного розрахунку.

Виконання завдання

В середовищі Microsoft Visual Studio створити новий проект застосунку Windows Forms. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон Windows Forms Application у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab07) та його розташування на диску. Буде створено проект з однією порожньою формою (рис. 7.1).

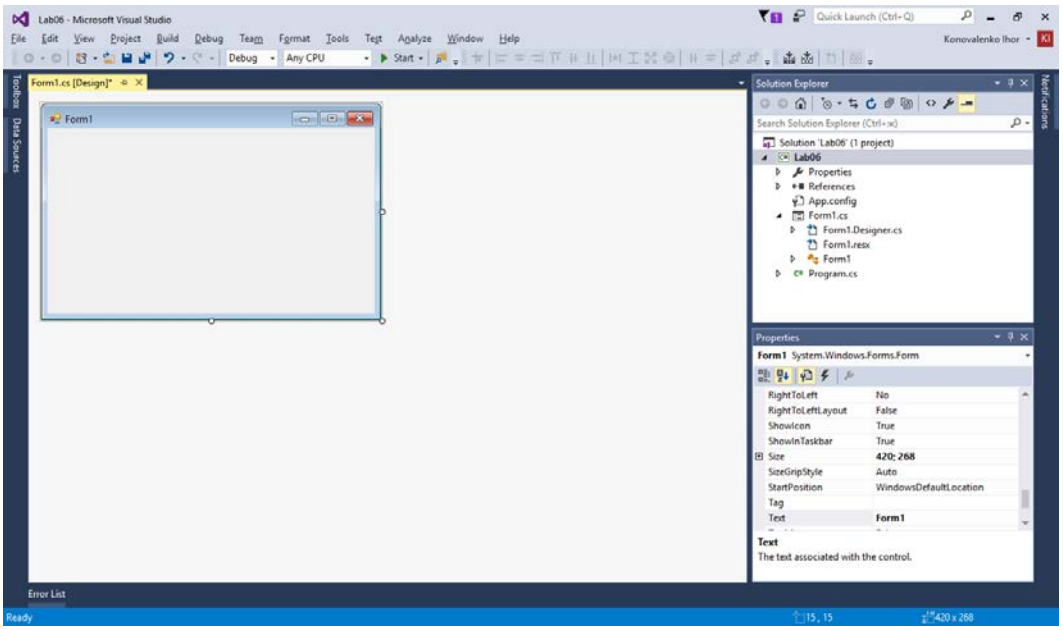


Рис. 7.1. Новий проект з порожньою формою

Розробка вікна застосунку

Вікно застосунку проектують відповідно до його цілей. Для виконання нашого завдання вікно повинно мати поля для вводу значень x_1 та x_2 та поле для виводу результату розрахунку функції $f(x_1,x_2)$. Крім цього, інтерфейс має містити засоби для ініціювання потрібних дій (розрахунок, очищення форми, вихід). Загальний вигляд вікна застосунку, розробка якого описана далі, показано на рис. 7.2.

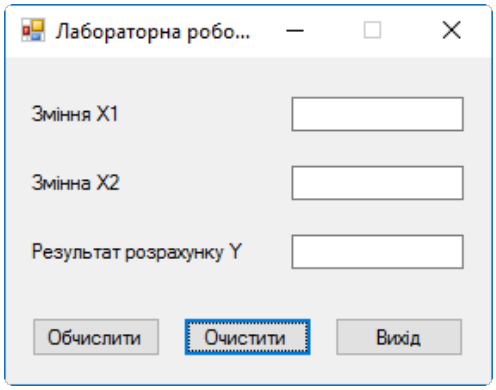


Рис. 7.2. Вікно застосунку

1. Змінити назву файлу, який описує форму (Form1.cs), та назву самої форми, на fMain [1]. Для цього у вікні Solution Explorer викликати контекстне меню на елементі Form1.cs та вибрати команду Rename. Перейменувати файл у fMain.cs. При цьому середовище перепитає, чи змінити у проекті також і змінну Form1, яка представляє форму. Вибрати "Так". Якщо вибрати "Ні", то назву форми слід змінити вручну. Для цього в дизайнері форм клікнути один раз на формі, далі у вікні Properties знайти властивість Name та задати для неї значення fMain.
2. Задати для форми такі властивості:
   - Text = "Лабораторна робота №7" (текстові значення вводять без лапок)
   - Name = fMain
   - StartPosition = CenterScreen
   - Icon->MaximizeBox = false

   Щоб задати властивість будь-якого компонента, слід спочатку вибрати його у дизайнері форм, клікнувши на ньому один раз. Далі у вікні Properties слід знайти потрібну властивість (за її назвою) і ввести для неї значення. Вікно Properties завжди відображує перелік властивостей для вибраного в дизайнері форм компонента. Щоб вибрати форму, слід клікнути на ній у тому місці, де немає інших компонентів.
3. Перейти до вікна Toolbox, що розташоване зліва у середовищі Microsoft Visual Studio (рис. 7.3). Це вікно містить перелік компонентів, які можна використовувати при проектуванні застосунку. Зокрема, тут є багато компонентів, які представляють різні елементи керування для вікон програми (кнопки, списки, меню тощо). Кожен з цих компонентів описаний окремим класом (ці класи поставляються разом з Microsoft Visual Studio). Відповідно, кожен клас має властивий йому набір властивостей, методів та подій (з їх описом можна ознайомитися у довідці MSDN).

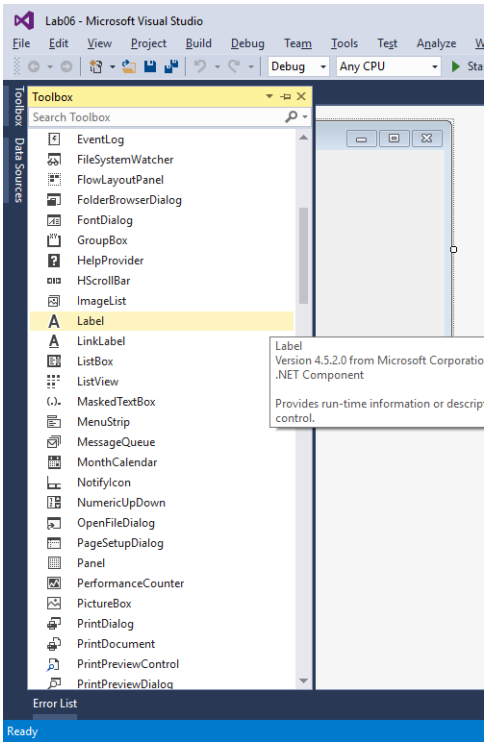


Рис. 7.3. Вікно Toolbox

[1] В загальному випадку назви змінним, які представляють об'єкти, можна давати довільно. Але прийнято, щоб назва містила інформацію про тип об'єкта та його призначення. У назві fMain префікс f означає тип об'єкта (це екземпляр класу Form), а Main означає, що це головна форма програми.

4. Розмістити на формі (у дизайнер форм) три компоненти Label (вони представляють текстові написи – "Змінна X1" та інші). Для цього знайти у вікні Toolbox цей компонент, клікнути на його кнопці, а потім клікнути у потрібному місці форми. На формі з'явиться вибраний компонент. В подальшому його можна перетягувати мишкою.
5. Для кожного елемента Label слід задати властивість Text, відповідно:
   - "Змінна X1"
   - "Змінна X2"
   - "Результат розрахунку Y"

   Для цього спочатку слід вибрати перший компонент, задати його властивість (у вікні Properties), потім вибрати другий і т.д.
6. Напроти трьох міток Label розмістити на формі три компоненти Textbox (вони представляють білі прямокутні поля для текстового вводу). Ці компоненти призначені відповідно для введення значень x_1 та x_2 та виведення значення функції. Попередньо розміщені компоненти Label пояснюють призначення цих полів.
7. Для кожного елемента Textbox слід задати властивість Name (назву), відповідно:
   - tbX1 [2]
   - tbX2
   - tbY
8. Для останнього елемента Textbox слід задати властивість ReadOnly = true (тільки для читання), оскільки його значення користувач не повинен змінювати безпосередньо.
9. У нижній частині форми розмістити три компоненти Button. Їх призначення – ініціювання таких дій програми: розрахунок, очищення полів та вихід.
10. Для кожної кнопки слід задати властивості Name (назва) і Text (напис), відповідно:
    - для першої: btnCalculate [3] та "Обчислити"
    - для другої: btnClear та "Очистити"
    - для третьої: btnExit та "Вихід"
11. У дизайнері форм задати розмір вікна таким, щоб воно охоплювало всі розміщені елементи керування та не містило зайвого порожнього місця з країв. Для цього слід перетягнути правий чи нижній край вікна, або правий нижній кут вікна. Щоб забрати відступ зліва або зверху, потрібно виділити всі компоненти (Ctrl+A у дизайнері форм) та перетягти їх вліво чи вверх.
12. Виконати пробний запуск застосунку, натиснувши клавішу F5. Після запуску відобразиться спроектоване вікно (рис. 7.4). У поля вікна, представлені елементом керування Textbox, можна вводити якийсь текст. На кнопки можна натискати, але при цьому не виконуються ніякі дії (бо нічого ще не запрограмовано). Вікно слід закрити, натиснувши на кнопку з хрестиком у правому верхньому кутку).
    На цьому розробка вікна завершена.

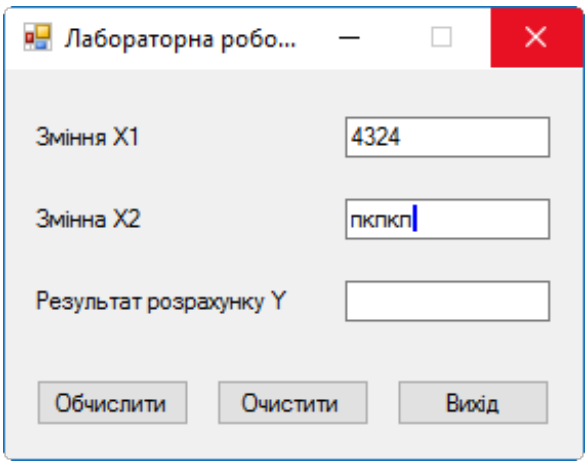


Рис. 7.4. Спроектоване вікно

[2] У назві tbX1 префікс tb означає тип компонента – Textbox, а X1 – призначення: ввід значення X1. Назви інших полів сформовано за цим же принципом.
[3] У назві btnCalc префікс btn означає тип Button, а Calculate – призначення кнопки (calculate – обчислювати). Назви інших кнопок сформовано аналогічно.

Програмування дії обчислення виразу

Для ініціювання обчислення виразу призначена кнопка "Обчислити" (btnCalculation). Щоб запрограмувати дії, які виконуватиме програма при натисканні на кнопку, слід створити метод – оброблювач події Click:

1. У дизайнері форм виділити кнопку "Обчислити".
2. У вікні Properties натиснути на кнопку Events і перейти на сторінку відображення подій.
3. Знайти подію Click і два рази клікнути у полі справа від неї. У середовищі Microsoft Visual Studio відкриється вікно редактора коду з автоматично створеним методом – оброблювачем цієї події:

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

}
```

   Щойно створений метод – оброблювач події не містить жодної інструкції. Код, написаний у цьому методі, автоматично виконуватиметься при кожному натисканні на кнопку "Обчислити" (тобто, при кожному настанні події Click).
4. Записати у метод btnCalculation_Click код з лістингу 7.1.

Лістинг 7.1

```
if (string.IsNullOrEmpty(tbX1.Text) ||
   (String.IsNullOrEmpty(tbX2.Text)))
{
    tbY.Text = "Не введено даних!";
    return;
}

double x1 = double.Parse(tbX1.Text);
double x2 = double.Parse(tbX2.Text);

double y = x1 * x2;

tbY.Text = y.ToString("0.######");
```

Програмування дії очищення полів

Якщо у середовищі Microsoft Visual Studio відкрито редактор коду, то для переходу в дизайнер форм слід вибрати команду View Designer контекстного меню (чи натиснути клавіші Shift+F7), або скористатися відповідною вкладкою середовища. І навпаки: якщо відкрито дизайнер форм, то для переходу в редактор коду слід вибрати команду View Code контекстного меню (чи натиснути клавішу F7), або скористатися відповідною вкладкою середовища.

1. У дизайнері форм виділити кнопку "Очистити".
2. У вікні Properties натиснути на кнопку Events і перейти на сторінку відображення подій.
3. Знайти подію Click і два рази клікнути у полі справа від неї. У середовищі Microsoft Visual Studio відкриється вікно редактора коду з автоматично створеним методом – оброблювачем цієї події:

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

}
```

4. Записати у метод btnClear_Click код з лістингу 7.2.

Лістинг 7.2

```
tbX1.Text = string.Empty;
tbX2.Text = string.Empty;
tbY.Text = string.Empty;
```

Програмування дії завершення роботи застосунку

1. У дизайнері форм виділити кнопку "Вихід".
2. У вікні Properties натиснути на кнопку Events і перейти на сторінку відображення подій.
3. Знайти подію Click і два рази клікнути у полі справа від неї. У середовищі Microsoft Visual Studio відкриється вікно редактора коду з автоматично створеним методом – оброблювачем цієї події:

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

}
```

4. Записати у метод btnExit_Click код з лістингу 7.3.

Лістинг 7.3

```
Application.Exit();
```

Запуск і тестування роботи застосунку

Щоб запустити застосунок, у середовищі Microsoft Visual Studio потрібно натиснути клавішу F5. Далі слід ввести значення для змінних X1, X2, і перевірити функціонал всіх кнопок: для розрахунку, очищення форми та виходу з програми.

Завдання для самостійного опрацювання

1. Вирішити розглянуте раніше завдання для заданої функції $f(x_1, x_2)$ згідно варіанту (таблиця 7.1).
2. Додатково до попереднього завдання, обчислити та вивести на екран:
   - для варіантів, номери яких закінчуються на 0...3 – середнє арифметичне значень x_1, x_2;
   - для варіантів, номери яких закінчуються на 4...6 – менше із значень x_1, x_2;
   - для варіантів, номери яких закінчуються на 7...9 – більше із значень x_1, x_2;
3. При виводі числових значень на екран:
   - для варіантів, номери яких закінчуються на 0...3 – виводити 4 знаки після коми;
   - для варіантів, номери яких закінчуються на 4...6 – виводити 3 знаки після коми;
   - для інших варіантів – виводити значення в експоненційному форматі.

Таблиця 7.1.

Варіант	Завдання	Варіант	Завдання
1.	$y=\frac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000\cdot\sqrt{x_1+x_2^{\,5}}}+65$	**2.**	$y=23\cos^2(x_1^{\,3}x_2^{\,5})+2x_1$
3.	$y=\frac{\sqrt{\cos^3(x_1)+x_2}}{x_1^{\,13}+\sqrt[3]{\cos(x_2)}}$	**4.**	$y=\sin\left(x_1-x_2^{\,3}+\sqrt{x_1}\right)-1.3x_1^{\,3}$
5.	$y=\frac{4\sin(3+x_1x_2)}{34-9x_2^{\,3}}$	**6.**	$y=\sqrt{56x_1+\frac{x_1+x_2+\sin(x_1x_2)}{5-\cos(x_2^{\,2})}}$
7.	$y=\sqrt[5]{0.1x_1\sin x_2\cos x_1^{\,2}+55x_1x_2}$	**8.**	$y=\sqrt{\frac{\cos(2x_2)+x_1/x_2}{16x_2x_1}}$
9.	$y=\cos\left(\sqrt{x_2}+34x_1\right)-4\sin(x_2)$	**10.**	$y=\sin^2\left(x_1\frac{x_2}{x_1+53x_2^{\,2}}\right)$
11.	$y=\frac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000}+65$	**12.**	$y=\exp\left(x_1-x_2^{\,2}\right)+31.55x_2x_1^{\,2}$
13.	$y=\frac{6-\cos(3+x_1)}{34-9x_2^{\,3}+x_2}$	**14.**	$y=\cos^4\left(x_1-\sqrt{\frac{x_2}{x_1+53x_2^{\,2}}}\right)$
15.	$y=23\sin^2(x_1^{\,3}x_2^{\,5})+2x_1+\cos(x_1x_2)$	**16.**	$y=\sqrt{56+\frac{x_1+x_2+\sin(x_1x_2)}{5\cos(x_2^{\,2})}}$
17.	$y=45x_1\sin x_2+\sqrt{9x_2x_1^{\,3}}$	**18.**	$y=\frac{1}{4+x_2}\cdot\sqrt{\cos^2\left(\frac{x_2}{x_1}\right)}$
19.	$y=\frac{\cos^3(x_1)+45+x_2}{x_1^{\,13}+\cos(x_2)}$	**20.**	$y=\frac{5\sqrt{x_1^{\,3}+x_2^{\,5}}-\cos(x_2)}{\exp(x_1)}$
21.	$y=\frac{\cos^2\left(\lg_{10}\frac{x_2}{x_1}\right)}{45+x_2}$	**22.**	$y=45\sin\left(x_1+x_2+\lg_{10}(x_1x_2^{\,2})\right)$
23.	$y=\sqrt{\frac{x_2^{\,2}+x_1/x_2}{16x_2x_1}}$	**24.**	$y=\frac{\ln(x_2)}{\sqrt[5]{0.6x_1\sin x_2\cos x_1^{\,4}}}$
25.	$y=\cos^3\left(\exp\left(\frac{x_1+2x_2+9}{0.666}\right)\right)$	**26.**	$y=\frac{3x_2-x_1^2}{\cos^3\left(\frac{x_1+2x_2+9}{0.37}\right)}$
27.	$y=\cos\left(\sqrt{x_2}+34\cdot\sin(x_1)\right)-4\sin(x_2)$	**28.**	$y=\sqrt{\frac{x_2^{\,2}+x_1/x_2}{\cos(x_1^{\,3}x_2^{\,5})+2x_1}}$
29.	$y=0.1x_1\sin x_2\cos x_1^{\,4}+55$	**30.**	$y=\frac{\sin^3(x_1)+45+x_2}{2x_1^{\,4}+4x_2}$

Лабораторна робота №8

Тема. Розробка багатовіконного застосунку Windows Forms.

Мета. Ознайомлення принципами розробки багатовіконних застосунків на основі технології Windows Forms.

Завдання

Розробити застосунок Windows Forms, який у головному вікні міститиме перелік даних про об'єкти типу "місто" (відповідний клас розроблено протягом лабораторної роботи №5). Крім цього, головне вікно повинне містити елемент керування, при активації якого має відображуватися друге вікно, призначене для внесення повної інформації про об'єкт "місто". Друге вікно має забезпечувати користувачеві вибір: а) підтвердити введені дані (тоді їх слід додати до переліку в головному вікні); б) скасувати операцію. Перелік у головному вікні, крім значень полів об'єкта, має відображувати і результат виклику одного з його методів.

Виконання завдання

В середовищі Microsoft Visual Studio створити новий проект застосунку Windows Forms. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон Windows Forms Application у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab08) та його розташування на диску.

Розробка головного вікна застосунку

1. Змінити назву файла, який описує головне вікно, на fMain.
2. Автоматично створена форма представлятиме головне вікно програми. Для головного вікна задати властивості:
    - Name = fMain
    - Icon->MaximizeBox = false
    - StartPosition = CenterScreen
    - Text = "Лабораторна робота №8"
3. Розмістити на формі компонент Textbox (рис. 8.1). Він буде призначений для виведення текстової інформації про об'єкти. Задати для нього властивості:
    - Name = tbTownsInfo
    - Multiline = true
    - ReadOnly = true
4. Задати розмір компонента достатньо великим для відображення значного масиву тексту.
5. Розмістити у правій частині форми дві кнопки. Одна з них призначена для введення даних про новий об'єкт "місто", а друга – для закінчення роботи застосунку.

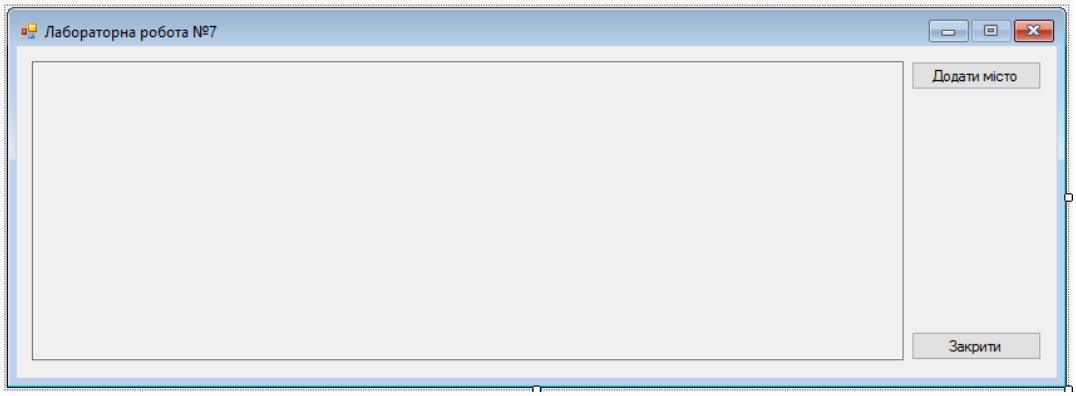


Рис. 8.1. Головна форма застосунку

6. Налаштувати властивості кнопки "Додати місто":
   - Name = btnAddTown
   - Text = "Додати місто"
7. Налаштувати властивості кнопки "Закрити"
   - Name = btnClose
   - Text = "Закрити"

Створення класу, який описує об'єкт "місто"

Клас прийнято описувати в окремому файлі. Щоб створити файл з описом класу:

1. У контекстному меню для елемента, який відповідає проекту (у вікні Solution Explorer), вибрати команду Add\New Item... Далі у вікні, яке з'явиться, в переліку Visual C# Items вибрати пункт Class, ввести назву класу – Town, і натиснути кнопку Add.

2. У файлі Town.cs (він відкриється у редакторі коду) ввести опис класу (його можна скопіювати з попередньої лабораторної). Код класу приведено у лістингу 8.1.

Лістинг 8.1

```
public class Town
{
    public string Name;
    public string Country;
    public string Region;
    public int Population;
    public double YearIncome;
    public double Square;
    public bool HasPort;
    public bool HasAirport;

    public double GetYearIncomePerInhabitant()
    {
        return YearIncome / Population;
    }
}
```

Розробка вікна для вводу даних про об'єкт "місто"

1. Додати до проекту нову форму. Для цього у контекстному меню елемента, який відповідає проекту (у вікні Solution Explorer), вибрати команду Add\New Item... Далі у вікні, яке

з'явиться, в переліку Visual C# Items вибрати пункт Windows Form, ввести назву класу, який описуватиме форму – fTown, і натиснути кнопку Add.

2. Задати для форми fTown властивості:
    - Icon->MaximizeBox = false
    - StartPosition = CenterScreen
    - Text = "Дані про нове місто"
    - ShowInTaskBar = false.
3. Розмістити на формі компонент GroupBox, призначений для візуального групування елементів керування. Задати властивість Text = "Загальні дані". За замовчуванням він матиме назву groupBox1.
4. Збільшити розмір компонента groupBox1 і розмістити на ньому 6 компонентів Label. Задати властивість Text кожного компонента відповідно до рис. 8.2.

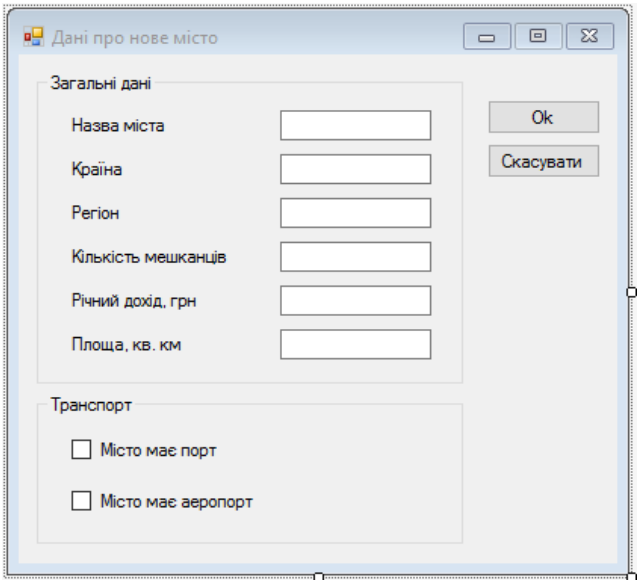


Рис. 8.2. Вікно для введення даних про місто

5. Розмістити на компоненті groupBox1 напроти кожного компонента Label 6 елементів керування TextBox (див. рис. 8.2). Задати їм такі назви (властивості Name):
    - tbName (напроти напису "Назва міста");
    - tbCountry ("Країна");
    - tbRegion ("Регіон");
    - tbPopulation ("Кількість мешканців");
    - tbYearIncome ("Річний дохід, грн");
    - tbSquare ("Площа, кв.км");
6. Розмістити на формі другий компонент GroupBox під першим (див. рис. 8.2). Задати його властивість Text = "Транспорт". За замовчуванням він матиме назву groupBox2.
7. Збільшити розмір компонента groupBox2 і розмістити на ньому 2 компоненти CheckBox. Задати властивість Text кожного компонента відповідно до малюнка.
8. Задати компонентам CheckBox такі назви (властивість Name):
    - chbHasPort (компонент з написом "Місто має порт");
    - chbHasAirport (компонент з написом "Місто має аеропорт")
9. Розмістити на формі 2 компоненти Button (див. рис. 8.2).
10. Для першої кнопки задати властивості:
    - Name = btnOk
    - Text = "Ok"

11. Для другої кнопки задати властивості:
    - Name = btnCancel
    - Text = "Скасувати"
12. Після розміщення кнопок вибрати компонент форми і задати для нього властивості:
    - AcceptButton = btnOk (автоматичне спрацювання кнопки при натисканні Enter)
    - CancelButton = btnCancel (автоматичне спрацювання кнопки при натисканні Esc)
13. Задати розмір вікна так, щоб його площа відповідала площі використаних елементів керування.

Код модуля fTown.cs

1. Перейти до редагування коду модуля fTown.cs. З дизайнера форм це можна зробити за допомогою клавіші F7.
2. У клас fTown додати одне публічне поле (це поле представлятиме об'єкт "місто", у якому міститимуться введені у вікні дані):

```
public Town TheTown;
```

3. Змінити конструктор fTown, надавши йому виду:

```
public fTown(Town t)
{
    TheTown = t;

    InitializeComponent();
}
```

У конструктор передається параметр t класу Town. У цей об'єкт (після натискання кнопки "Ok") буде записано дані, введені користувачем у формі. Поле TheTown зберігає посилання на об'єкт.

4. Створити оброблювач події Click для кнопки btnOk (для цього два рази клацнути на ній у дизайнері форм).
5. У оброблювач записати код з лістингу 8.2 (сірим у лістингу виділено код, автоматично згенерований середовищем, том вручну його вносити не потрібно).

Лістинг 8.2

```
private void btnOk_Click(object sender, EventArgs e)
{
    TheTown.Name = tbName.Text.Trim();
    TheTown.Country = tbCountry.Text.Trim();
    TheTown.Region = tbRegion.Text.Trim();
    TheTown.Population = int.Parse(tbPopulation.Text.Trim());
    TheTown.YearIncome = double.Parse(tbYearIncome.Text.Trim());
    TheTown.Square = double.Parse(tbSquare.Text.Trim());
    TheTown.HasPort = chbHasPort.Checked;
    TheTown.HasAirport = chbHasAirport.Checked;

    DialogResult = DialogResult.OK;
}
```

Тут значення з полів форми записуються у об'єкт TheTown, посилання на який передається у конструкторі fTown при створенні форми.

Поле DialogResult форми містить її модальний результат. Модальний результат діалогового вікна DialogResult.OK у нашому випадку означає, що дані успішно введено і записано в об'єкт TheTown

6. Створити оброблювач події Click для кнопки btnCancel (для цього два рази клацнути на ній у дизайнері форм).
7. У оброблювач записати код з лістингу 8.3.

Лістинг 8.3

```
private void btnCancel_Click(object sender, EventArgs e)
{
    DialogResult = DialogResult.Cancel;
}
```

Поле DialogResult форми містить її модальний результат. Модальний результат діалогового вікна DialogResult.Cancel у нашому випадку означає, що користувач скасовує операцію і введені дані не потрібні (не записуємо їх у об'єкт TheTown).

8. Створити оброблювач події Load для форми (для цього два рази клацнути на ній у дизайнері форм).
9. У оброблювач записати код, який при першому показі вікна відображує дані об'єкта, передані при створенні вікна (лістинг 8.4).

Лістинг 8.4

```
private void fTown_Load(object sender, EventArgs e)
{
    if (TheTown != null)
    {
        tbName.Text = TheTown.Name;
        tbCountry.Text = TheTown.Country;
        tbRegion.Text = TheTown.Region;
        tbPopulation.Text = TheTown.Population.ToString();
        tbYearIncome.Text = TheTown.YearIncome.ToString("0.00");
        tbSquare.Text = TheTown.Square.ToString("0.000");
        chbHasPort.Checked = TheTown.HasPort;
        chbHasAirport.Checked = TheTown.HasAirport;
    }
}
```

Програмування операцій головного вікна

1. Перейти до дизайнера форм з головним вікном (можна через Solution Explorer).
2. Створити оброблювач події Click для кнопки btnAddTown ("Додати місто"). Це можна зробити, два рази клацнувши на кнопці у дизайнері, або через вікно Properties.
3. Привести оброблювач до виду згідно лістингу 8.5.

Лістинг 8.5

```
private void btnAddTown_Click(object sender, EventArgs e)
{
    Town town = new Town();

    fTown ft = new fTown(town);

    if (ft.ShowDialog() == DialogResult.OK)
    {
        tbTownsInfo.Text +=
            string.Format("{0}, {1}, {2}. Мешканців: {3}. Річний дохід:
{4:0.00} грн. Площа: {5:0.000} кв. км. [{6} | {7}] | Річний дохід на
мешканця: {8:0.00} грн\r\n",
```

```
            town.Name, town.Country, town.Region,
            town.Population, town.YearIncome, town.Square,
            town.HasPort ? "Є порт" : "Немає порта",
            town.HasAirport ? "Є аеропорт" : "Немає аеропорта",
            town.GetYearIncomePerInhabitant());
    }
}
```

Тут спочатку створюємо екземпляр класу Town, потім показуємо його, і, якщо у вікні fTown натиснено кнопку "Ok", формуємо рядок на основі введених даних про об'єкт "місто" та додаємо його у перелік міст tbTownsInfo.

4. Перейти до дизайнера форм з головним вікном (можна через Solution Explorer).
5. Створити оброблювач події Click для кнопки btnClose ("Закрити"). Це можна зробити, два рази клацнувши на кнопці у дизайнері, або через вікно Properties.
6. Привести оброблювач до виду згідно лістингу 8.6.

Лістинг 8.6

```
private void btnClose_Click(object sender, EventArgs e)
{
  if (MessageBox.Show("Припинити роботу застосунку?",
      "Припинити роботу", MessageBoxButtons.OKCancel,
      MessageBoxIcon.Question) == DialogResult.OK)
    Application.Exit();
}
```

У цьому коді користувачеві задаємо запитання ("Припинити роботу застосунку?"). Користувач має два варіанти відповіді, представлені кнопками OK та Cancel (параметр MessageBoxButtons.OKCancel). Вказано також тип діалогового вікна (MessageBoxIcon.Question) – вікно з запитанням. Якщо при перепитуванні користувач натиснув кнопку Ok, закриваємо застосунок.

Запуск і тестування роботи застосунку

Щоб запустити застосунок, потрібно натиснути клавішу F5. Слід перевірити весь функціонал програми (див. схему на рис. 8.3). Приклад головного вікна програми з введеними даними про міста приведено на рис. 8.4.

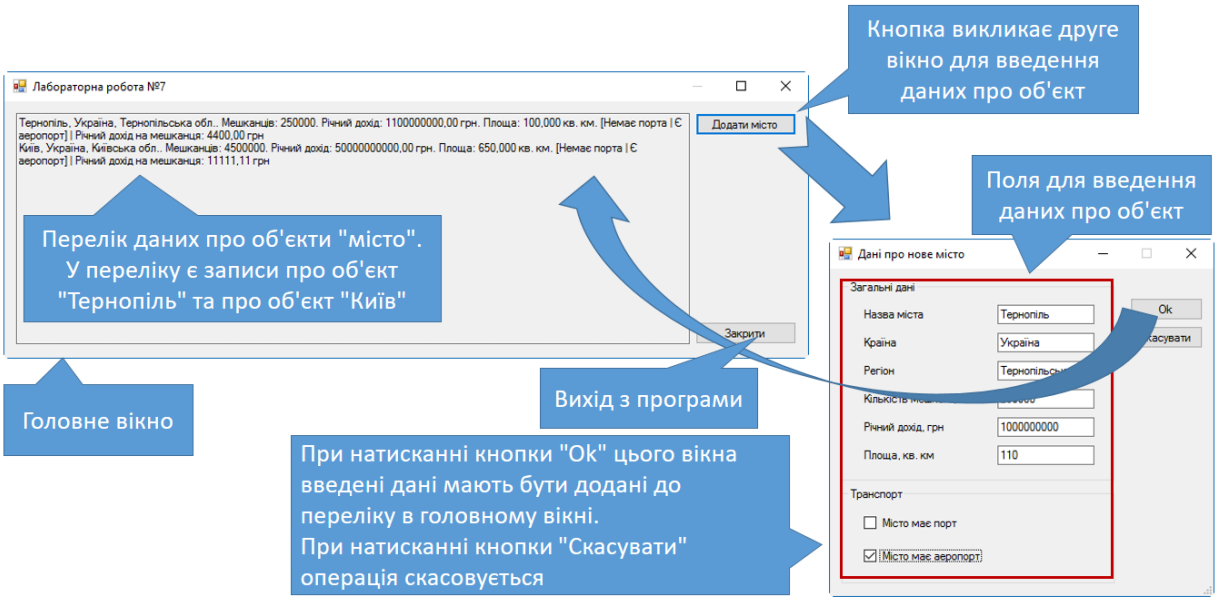


Рис. 8.3. Ілюстрація роботи застосунку

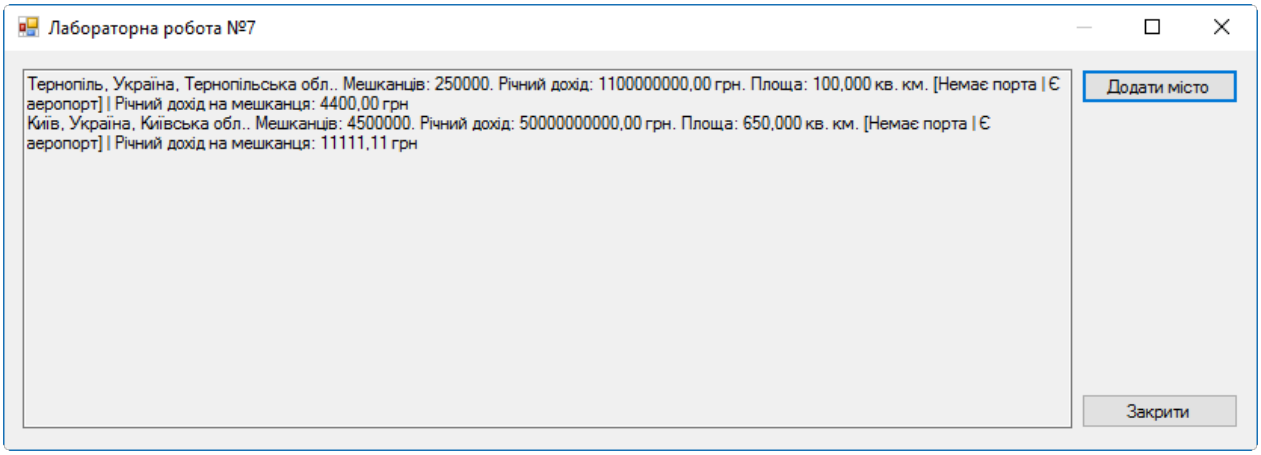


Рис. 8.4. Приклад головного вікна застосунку з введеними даними

Завдання для самостійного опрацювання

Вирішити розглянуте раніше завдання щодо предметної області, вказаної у завданні до лабораторної роботи №5. Для опису об'єкта використати клас, розроблений при виконанні лабораторної роботи №5.

Лабораторна робота №9

Тема. Відображення довільних даних у табличній формі за допомогою компонента DataGridView.

Мета. Ознайомлення принципами виведення масиву даних за допомогою класу DataGridView.

Завдання

Розробити застосунок Windows Forms, який розраховуватиме заданий математичний вираз з двома змінними $f(x_1,x_2)=x_1+x_2$ в певному діапазоні для обох змінних. Програма повинна забезпечувати ввід мінімальних $x1_{min}$, $x2_{min}$ та максимальних $x1_{max}$, $x2_{max}$ значень, а також кроку зміни *dx1, dx2*. Всі розраховані значення виразу слід вивести у вікно в табличній формі за допомогою елемента керування DataGridView.

Виконання завдання

В середовищі Microsoft Visual Studio створити новий проект застосунку Windows Forms. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон Windows Forms Application у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab09) та його розташування на диску.

Розробка головного вікна застосунку

1. Змінити назву файла, який описує головне вікно, на fMain.
2. Автоматично створена форма представлятиме головне вікно програми. Привести форму до вигляду згідно рис. 9.1. Для цього розмістити на ній:
    - 6 компонентів Label
    - 6 компонентів Textbox
    - компонент DataGridView
    - 3 компоненти Button
3. Задати назви (властивість Name) елементів керування TextBox відповідно до призначення:
    - tbx1min
    - tbx1max
    - tbdx1
    - tbx2min
    - tbx2max
    - tbdx2
4. Задати назву (властивість Name) елемента керування DataGridView:
    - gv

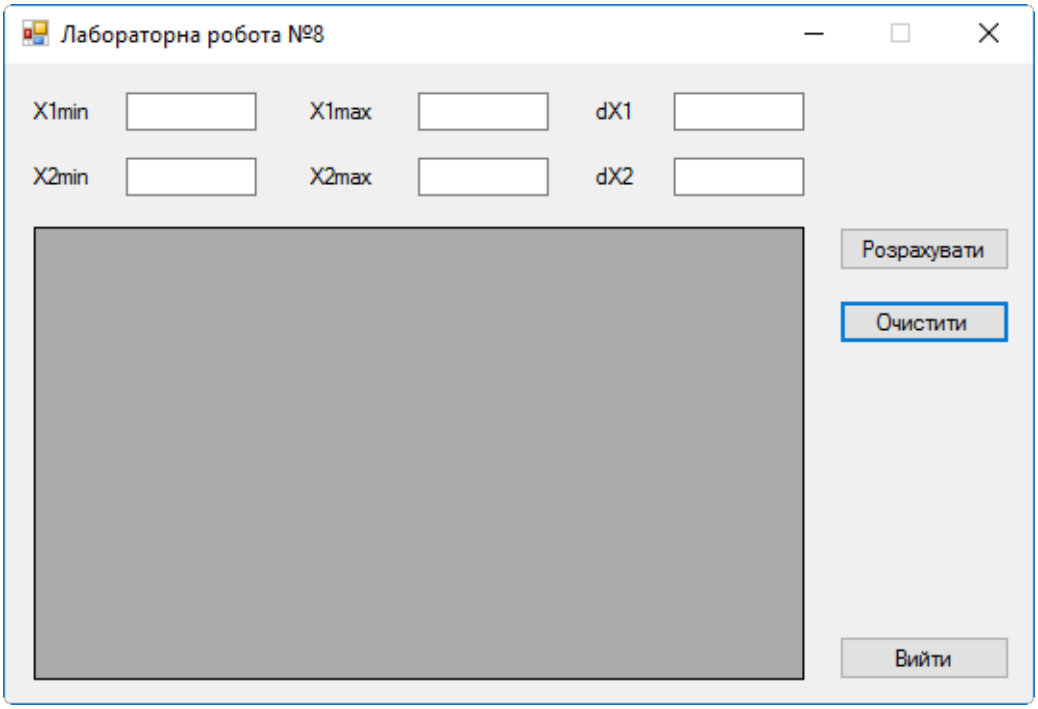


Рис. 9.1. Головна форма застосунку

5. Задати назви (властивість Name) елементів керування Button:
   - btnCalc (Розрахувати)
   - btnClear (Витерти)
   - btnExit (Вийти)
6. Продумати алгоритм програми. Алгоритм для виконання завдання такий:
   1) Отримати значення x1min, x1max, dx1, x2min, x2max, dx2.
   2) Відповідно до введених даних, обчислити кількість стовпчиків та рядків таблиці.
   3) У циклі пройти по всіх комірках стовпчика із заголовками рядків та заповнити його значеннями змінної x1.
   4) У циклі пройти по всіх комірках рядка із заголовками стовпчиків та заповнити його значеннями змінної x2.
   5) rw = 0 (змінна rw представляє поточний індекс рядка).
   6) x1 = x1min (змінна x1 представляє поточне значення x1).
   7) У циклі, поки x1 <= x1max:
      - x2 = x2min (змінна x2 представляє поточне значення x2).
      - cl = 0 (змінна cl представляє поточний індекс стовпчика).
      - У циклі, поки x2 <= x2max:
        - Розрахувати значення функції для поточних значень x1 та x2 і вивести його у комірку.
        - Збільшити x2 на dx2, збільшити cl на 1.
      - Збільшити x1 на dx1, збільшити rw на 1.
7. У оброблювач події Click для кнопки btnCalc ("Розрахувати") записати код з лістингу 9.1.

Лістинг 9.1

```
// Перетворення текстових рядків, які ввів користувач,
// у змінні числового типу
double x1min = double.Parse(tbx1min.Text);
double x1max = double.Parse(tbx1max.Text);
double x2min = double.Parse(tbx2min.Text);
```

```
double x2max = double.Parse(tbx2max.Text);
double dx1 = double.Parse(tbdx1.Text);
double dx2 = double.Parse(tbdx2.Text);

// Обчислення кількості рядків та стовпчиків таблиці
gv.ColumnCount = (int)Math.Truncate((x2max - x2min) / dx2) + 1;
gv.RowCount = (int)Math.Truncate((x1max - x1min) / dx1) + 1;

// Вивід заголовків рядків та стовпців таблиці
for (int i = 0; i < gv.RowCount; i++)
    gv.Rows[i].HeaderCell.Value = (x1min + i * dx1).ToString("0.000");
gv.RowHeadersWidth = 80;

for (int i = 0; i < gv.ColumnCount; i++)
{
    gv.Columns[i].HeaderCell.Value = (x2min + i *
dx2).ToString("0.000");
    gv.Columns[i].Width = 60;
}

// Для автоматичного підлаштування розмірів стовпчиків та рядків
// можна використовувати ці методи
//gv.AutoResizeColumns();
//gv.AutoResizeRows();

int cl, rw;
double x1, x2, y;

// Розрахунок і вивід результатів
rw = 0;
x1 = x1min;
while (x1 <= x1max)
{
    x2 = x2min;
    cl = 0;
    while (x2 <= x2max)
    {
        y = x1 + x2;
        gv.Rows[rw].Cells[cl].Value = y.ToString("0.000");
        x2 += dx2;
        cl++;
    }
    x1 += dx1;
    rw++;
}
```

8. У оброблювач події Click для кнопки btnClear ("Очистити") записати наступний код з лістингу 9.2.

Лістинг 9.2

```
tbx1min.Text = "";
tbx1max.Text = "";
tbx2min.Text = "";
tbx2max.Text = "";
tbdx1.Text = "";
tbdx2.Text = "";

gv.Rows.Clear();
for (int Cl = 0; Cl < gv.ColumnCount; Cl++)
    gv.Columns[Cl].HeaderCell.Value = "";
```

9. У оброблювач події Click для кнопки btnExit ("Вихід") записати код з лістингу 9.3.

Лістинг 9.3

```
if (MessageBox.Show("Закрити програму?", "Вихід з програми",
    MessageBoxButtons.OKCancel, MessageBoxIcon.Question) ==
DialogResult.OK)
        Application.Exit();
```

10. Запустити застосунок (F5) і перевірити його функціонал. Приклад вікна програми з розрахунковими даними приведено на рис. 9.2.

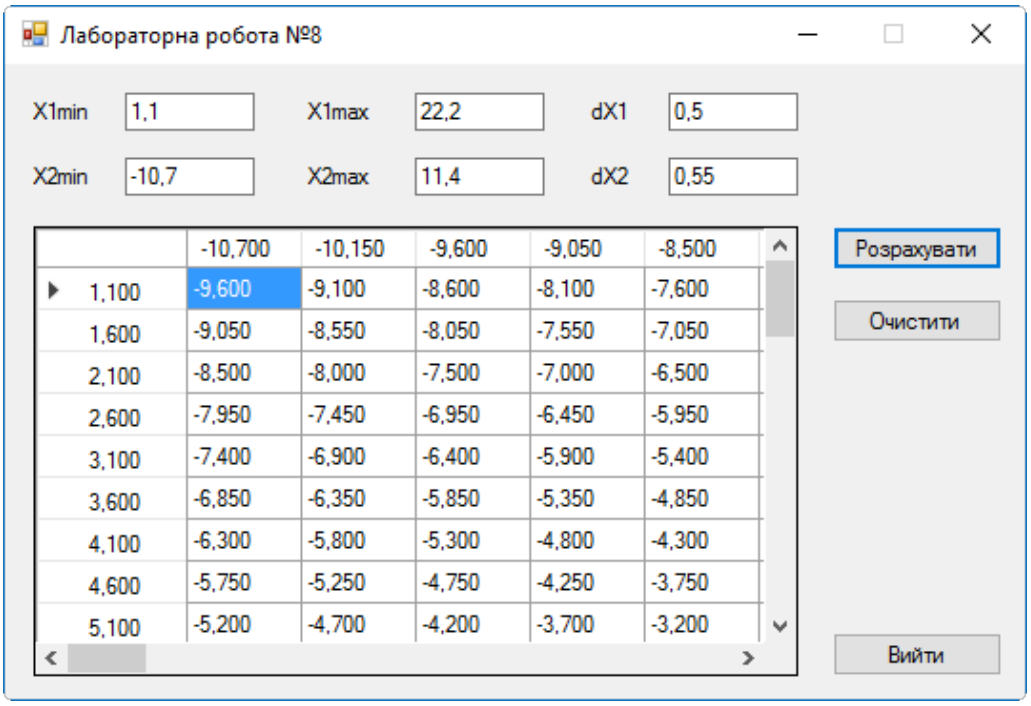


Рис. 9.2. Приклад вікна програми з розрахунковими даними

Завдання для самостійного опрацювання

1. Вирішити розглянуте раніше завдання для заданої функції $f(x_1, x_2)$ згідно варіанту.
2. Додатково до попереднього завдання, обчислити та вивести на екран:
   - для варіантів, номери яких закінчуються на 0 чи 5 – суму всіх від'ємних розрахованих проміжних значень $f(x_1, x_2)$;
   - для варіантів, номери яких закінчуються на 1 чи 6 – добуток всіх додатних розрахованих проміжних значень $f(x_1, x_2)$;
   - для варіантів, номери яких закінчуються на 2 чи 7 – суму від'ємних синусів всіх розрахованих проміжних значень $f(x_1, x_2)$;
   - для варіантів, номери яких закінчуються на 3 чи 8 – суму додатних косинусів всіх розрахованих проміжних значень $f(x_1, x_2)$;
   - для варіантів, номери яких закінчуються на 4 чи 9 – суму квадратів всіх від'ємних розрахованих проміжних значень $f(x_1, x_2)$;
3. В кінці до таблиці з результатами додати порожній рядок (знизу) та стовпчик (справа) і заповнити їх:
   - для парних варіантів – сумами відповідних стовпчиків та рядків;
   - для непарних варіантів – середніми значеннями відповідних стовпчиків та рядків.

Таблиця 9.1.

Варіант	Завдання	Варіант	Завдання
31.	$y=\frac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000\cdot\sqrt{x_1+x_2^{\,5}}}+65$	**32.**	$y=23\cos^2(x_1^{\,3}x_2^{\,5})+2x_1$
33.	$y=\frac{\sqrt{\cos^3(x_1)+x_2}}{x_1^{\,13}+\sqrt[3]{\cos(x_2)}}$	**34.**	$y=\sin\left(x_1-x_2^{\,3}+\sqrt{x_1}\right)-1.3x_1^{\,3}$
35.	$y=\frac{4\sin(3+x_1x_2)}{34-9x_2^{\,3}}$	**36.**	$y=\sqrt{56x_1+\frac{x_1+x_2+\sin(x_1x_2)}{5-\cos(x_2^{\,2})}}$
37.	$y=\sqrt[5]{0.1x_1\sin x_2\cos x_1^{\,2}+55x_1x_2}$	**38.**	$y=\sqrt{\frac{\cos(2x_2)+x_1/x_2}{16x_2x_1}}$
39.	$y=\cos\left(\sqrt{x_2}+34x_1\right)-4\sin(x_2)$	**40.**	$y=\sin^2\left(x_1\frac{x_2}{x_1+53x_2^{\,2}}\right)$
41.	$y=\frac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000}+65$	**42.**	$y=\exp\left(x_1-x_2^{\,2}\right)+31.55x_2x_1^{\,2}$
43.	$y=\frac{6-\cos(3+x_1)}{34-9x_2^{\,3}+x_2}$	**44.**	$y=\cos^4\left(x_1-\sqrt{\frac{x_2}{x_1+53x_2^{\,2}}}\right)$
45.	$y=23\sin^2(x_1^{\,3}x_2^{\,5})+2x_1+\cos(x_1x_2)$	**46.**	$y=\sqrt{56+\frac{x_1+x_2+\sin(x_1x_2)}{5\cos(x_2^{\,2})}}$
47.	$y=45x_1\sin x_2+\sqrt{9x_2x_1^{\,3}}$	**48.**	$y=\frac{1}{4+x_2}\cdot\sqrt{\cos^2\left(\frac{x_2}{x_1}\right)}$
49.	$y=\frac{\cos^3(x_1)+45+x_2}{x_1^{\,13}+\cos(x_2)}$	**50.**	$y=\frac{5\sqrt{x_1^{\,3}+x_2^{\,5}}-\cos(x_2)}{\exp(x_1)}$
51.	$y=\frac{\cos^2\left(\lg_{10}\frac{x_2}{x_1}\right)}{45+x_2}$	**52.**	$y=45\sin\left(x_1+x_2+\lg_{10}(x_1x_2^{\,2})\right)$
53.	$y=\sqrt{\frac{x_2^{\,2}+x_1/x_2}{16x_2x_1}}$	**54.**	$y=\frac{\ln(x_2)}{\sqrt[5]{0.6x_1\sin x_2\cos x_1^{\,4}}}$
55.	$y=\cos^3\left(\exp\left(\frac{x_1+2x_2+9}{0.666}\right)\right)$	**56.**	$y=\frac{3x_2-x_1^2}{\cos^3\left(\frac{x_1+2x_2+9}{0.37}\right)}$
57.	$y=\cos\left(\sqrt{x_2}+34\cdot\sin(x_1)\right)-4\sin(x_2)$	**58.**	$y=\sqrt{\frac{x_2^{\,2}+x_1/x_2}{\cos(x_1^{\,3}x_2^{\,5})+2x_1}}$
59.	$y=0.1x_1\sin x_2\cos x_1^{\,4}+55$	**60.**	$y=\frac{\sin^3(x_1)+45+x_2}{2x_1^{\,4}+4x_2}$

Лабораторна робота №10

Тема. Відображення даних про об'єкти одного класу в табличній формі.

Мета. Ознайомлення принципами виведення даних про множину об'єктів за допомогою класу DataGridView.

Завдання

Розробити застосунок Windows Forms, який забезпечує внесення даних про об'єкти типу "місто" та відображення цих даних у табличній формі за допомогою елемента керування DataGridView. Крім цього, застосунок має забезпечувати редагування даних про об'єкт типу "місто", видалення даних про вибраний об'єкт та загальне очищення списку. Керування роботою програми реалізувати через панель інструментів.

Виконання завдання

В середовищі Microsoft Visual Studio створити новий проект застосунку Windows Forms. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон Windows Forms Application у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab10) та його розташування на диску.

Проектування інтерфейсу застосунку

Для виконання всіх перелічених задач слід спроектувати інтерфейс вікна. На рис. 10.1 приведено вигляд вікна, розробка якого буде описана далі. Дії, які виконуватиме програма:

1) Додавання нового запису.
2) Редагування вибраного запису.
3) Видалення вибраного запису.
4) Відображення списку записів у табличній формі.
5) Очищення таблиці (видалення всіх записів).
6) Вихід з програми.

Для виклику всіх цих дій використаємо компонент ToolStrip.

Розробка головного вікна

1. Задати властивості головного вікна:
    - Name = fMain (змінити у Solution Explorer)
    - Text = "Лабораторна робота №10"
    - StartPosition = CenterScreen

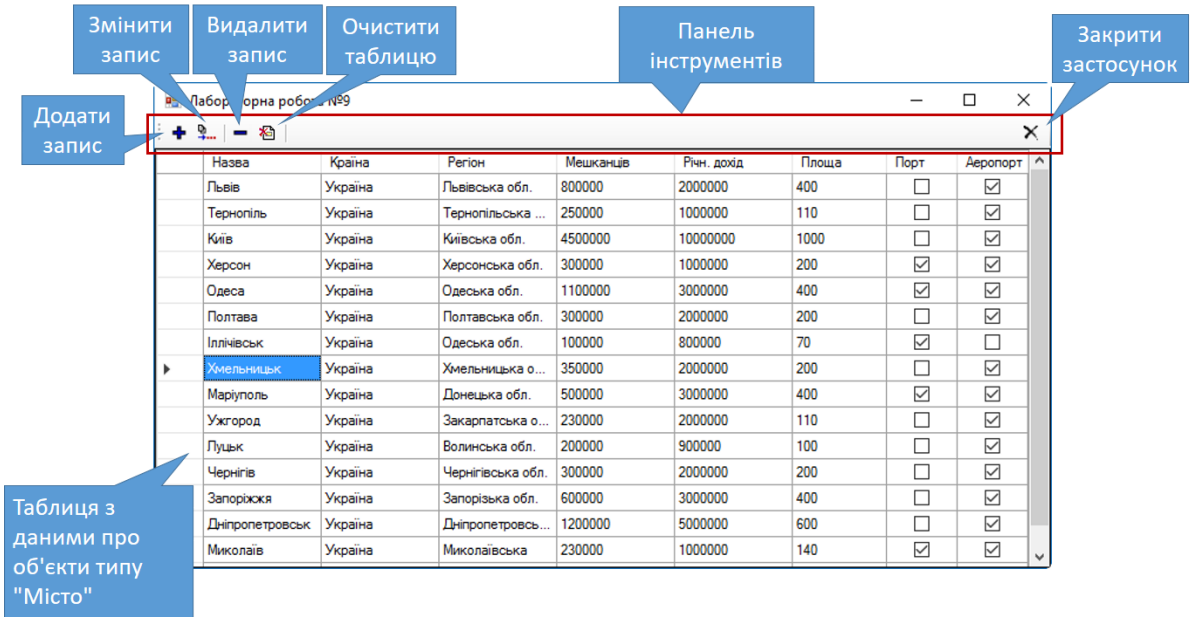


Рис. 10.1. Головне вікно застосунку

2. На головній формі розмістити компонент ToolStrip.
3. За допомогою спеціальної кнопки компонента ToolStrip розмістити на панелі інструментів такі елементи керування (рис. 10.2, але спочатку вони будуть без піктограм):
    - кнопку (Button) – для додавання запису
    - кнопку (Button) – для зміни запису
    - розділювач (Separator)
    - кнопку (Button) – для видалення запису
    - кнопку (Button) – для очищення таблиці
    - розділювач (Separator)
    - кнопку (Button) – для виходу з програми

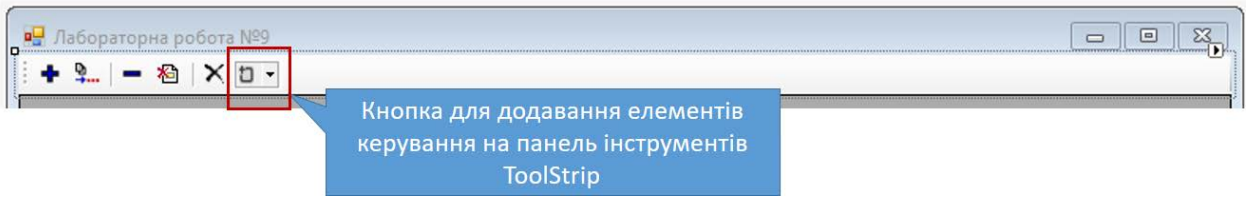


Рис. 10.2. Панель Toolstrip

4. Задати властивості кнопки для додавання нового запису:
    - Name = btnAdd
    - Text = "Додати запис про місто"
    - Image – задати піктограму, вибравши відповідний фал. Для цього натиснути на кнопку з трикрапкою біля цієї властивості у вікні Properties
5. Задати властивості кнопки для зміни поточного запису:
    - Name = btnEdit
    - Text = "Редагувати запис"
    - Image – задати піктограму, вибравши відповідний фал. Для цього натиснути на кнопку з трикрапкою біля цієї властивості у вікні Properties
6. Задати властивості кнопки для видалення поточного запису:

- Name = btnDel
- Text = "Видалити запис"
- Image – задати піктограму, вибравши відповідний фал. Для цього натиснути на кнопку з трикрапкою біля цієї властивості у вікні Properties

7. Задати властивості кнопки для очищення таблиці:
   - Name = btnClear
   - Text = "Очистити дані"
   - Image – задати піктограму, вибравши відповідний фал. Для цього натиснути на кнопку з трикрапкою біля цієї властивості у вікні Properties
8. Задати властивості кнопки для виходу з програми:
   - Name = btnExit
   - Text = "Вийти з програми"
   - Image – задати піктограму, вибравши відповідний фал. Для цього натиснути на кнопку з трикрапкою біля цієї властивості у вікні Properties
9. Для розділювачів задати властивість:
   - Name = tsSeparator1 та tsSeparator2 відповідно
10. Зручно, щоб кнопка "Вийти з програми" була не разом зі всіма іншими кнопками, а окрема, біля правого краю панелі інструментів. Тому потрібно збільшити відступ зліва для цієї кнопки (відступи задає властивість Margin кнопки). Для цього в оброблювач події Resize форми fMain записати код з лістингу 10.1.

Лістинг 10.1

```
private void fMain_Resize(object sender, EventArgs e)
{
    int buttonsSize = 5 * btnAdd.Width + 2 * tsSeparator1.Width + 30;
    btnExit.Margin = new Padding(Width - buttonsSize, 0, 0, 0);
}
```

11. Розмістити на формі компонент BindingSource. Після розміщення він з'явиться у нижній частині дизайнера форм. Для нього слід задати властивість:
    - Name = bindSrcTowns
12. Розмістити на формі компонент DataGridView. Він призначений для відображення даних про об'єкти. Для компонента DataGridView задати властивості:
    - Name = gvTowns
    - DataSource = bindSrcTowns (цим вказуємо, що джерелом даних для таблиці буде раніше розміщений на формі bindSrcTowns)
    - Dock = Fill (вибрати з випадаючого списку)
    - AllowUserToAddRows = false
    - AllowUserToDeleteRows = false
    - ReadOnly = true

    Властивості DataSource, AllowUserToAddRows, AllowUserToDeleteRows та ReadOnly компонента DataGridView можна задати також у дизайнері форм. Для цього слід клікнути на маленькій кнопці зі стрілкою зверху справа компонента DataGridView (рис. 10.3).

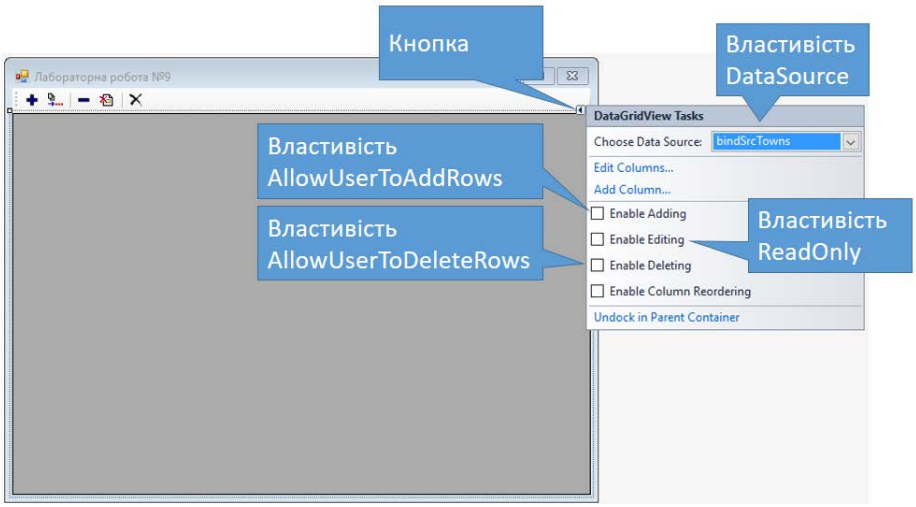


Рис. 10.3. Налаштування компонента DataGridView у дизайнері форм

Створення класу, який описує об'єкт "місто"

За завданням програма має працювати з об'єктами типу "місто". Такий об'єкт опишемо у класі Town. Щоб створити новий файл з описом класу:

1. У контекстному меню для елемента, який відповідає проекту (у вікні Solution Explorer), вибрати команду Add\New Item... У вікні, яке з'явиться, в переліку Visual C# Items вибрати пункт Class, ввести назву класу – Town, і натиснути кнопку Add.

2. У файлі Town.cs (він відкриється у редакторі коду) ввести опис класу (його можна скопіювати з лабораторної №8 і підредагувати). Код класу приведено у лістингу 10.2. Від прикладу в лабораторній роботі №8 цей код відрізняється тим, що замість полів використано властивості. Справа у тому, що компонент BindingSource працює з властивостями, а не полями. При цьому використано властивості з автоматичною реалізацією (методів доступу та підтримуючого поля не задано).

Лістинг 10.2

```
public class Town
{
    public string Name { get; set; }
    public string Country { get; set; }
    public string Region { get; set; }
    public int Population { get; set; }
    public double YearIncome { get; set; }
    public double Square { get; set; }
    public bool HasPort { get; set; }
    public bool HasAirport { get; set; }

    public double GetYearIncomePerInhabitant()
    {
        return YearIncome / Population;
    }

    public Town()
    {

    }

    public Town(string name, string country, string region,
        int population, double yearIncome, double square,
        bool hasPort, bool hasAirport)
    {
```

```
            Name = name;
            Country = country;
            Region = region;
            Population = population;
            YearIncome = yearIncome;
            Square = square;
            HasPort = hasPort;
            HasAirport = hasAirport;
        }
    }
```

Конструктор без параметрів створює об'єкт, не ініціалізуючи його властивості, а другий конструктор створює об'єкт та присвоює його властивостям задані значення.

3. Перед першим показом головного вікна (в оброблювачі події Load головної форми) ініціалізуємо BindingSource та DataGridView (лістинг 10.3).

Лістинг 10.3

```
gvTowns.AutoGenerateColumns = false;

DataGridViewColumn column = new DataGridViewTextBoxColumn();
column.DataPropertyName = "Name";
column.Name = "Назва";
gvTowns.Columns.Add(column);

column = new DataGridViewTextBoxColumn();
column.DataPropertyName = "Country";
column.Name = "Країна";
gvTowns.Columns.Add(column);

column = new DataGridViewTextBoxColumn();
column.DataPropertyName = "Region";
column.Name = "Регіон";
gvTowns.Columns.Add(column);

column = new DataGridViewTextBoxColumn();
column.DataPropertyName = "Population";
column.Name = "Мешканців";
gvTowns.Columns.Add(column);

column = new DataGridViewTextBoxColumn();
column.DataPropertyName = "YearIncome";
column.Name = "Річн. дохід";
gvTowns.Columns.Add(column);

column = new DataGridViewTextBoxColumn();
column.DataPropertyName = "Square";
column.Name = "Площа";
column.Width = 80;
gvTowns.Columns.Add(column);

column = new DataGridViewCheckBoxColumn();
column.DataPropertyName = "HasPort";
column.Name = "Порт";
column.Width = 60;
gvTowns.Columns.Add(column);

column = new DataGridViewCheckBoxColumn();
column.DataPropertyName = "HasAirport";
column.Name = "Аеропорт";
column.Width = 60;
gvTowns.Columns.Add(column);

bindSrcTowns.Add(new Town("Львів", "Україна", "Львівська обл.", 800000,
   2000000, 400, false, true));
```

```
EventArgs args = new EventArgs();
OnResize(args);
```

4. Запустити програму (F5). На цьому етапі розробки у вікні має відобразитися таблиця з одним записом, який додається в оброблювачі події Load форми (рис. 10.4). Закрити застосунок.

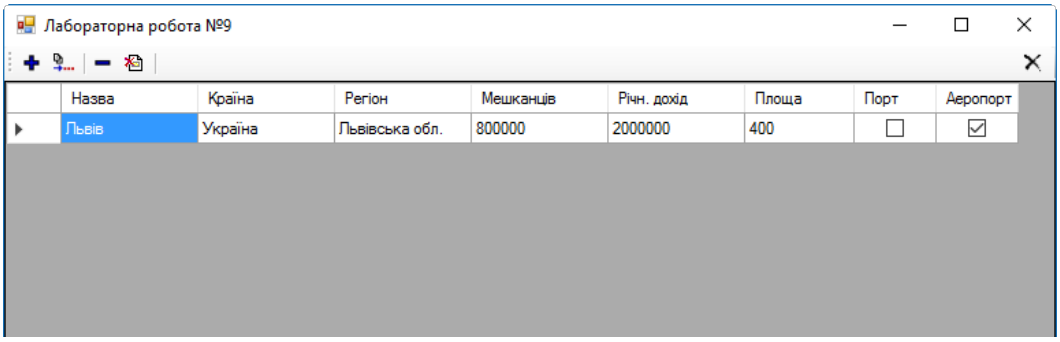


Рис. 10.4. Вікно з одним записом, який додається при запуску програми

5. Для внесення даних про місто при додаванні нового запису або при зміні існуючого запису використаємо форму fTown, розроблену протягом лабораторної роботи №8 (рис. 10.5). Щоб використати розроблене раніше вікно, слід:

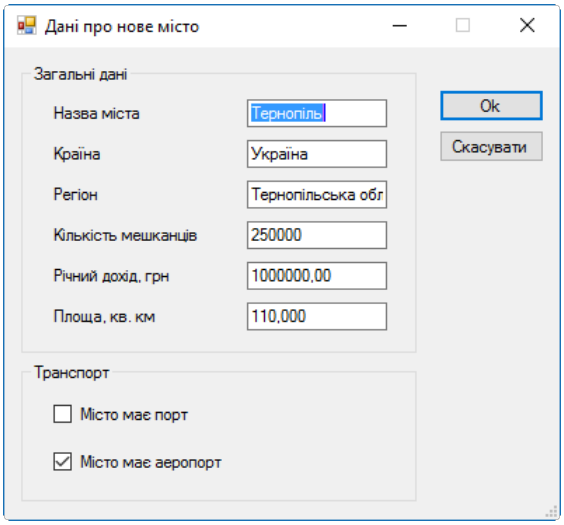


Рис. 10.5. Вікно для введення даних про місто

   1) З папки "старого" проекту слід скопіювати в папку Lab10\Lab10 нового проекту такі файли: fTown.cs, fTown.Designer.cs та fTown.resx.

   2) З контекстного меню Solution Explorer, викликаного на пункті , який відповідає проекту, вибрати команду Add\Existing Item...

   3) У вікні, що з'явиться, вибрати раніше скопійований файл fTown.cs, і натиснути кнопку Add. Додані елементи мають відобразитися в Solution Explorer.

   4) У файлах fTown.cs та fTown.Designer.cs змінити назву простора імен на той, який використовує даний проект: namespace Lab10 замість старого namespace Lab08.

6. Створити оброблювач події Click для кнопки btnAdd ("Додати запис про місто") і записати код з лістингу 10.4.

Лістинг 10.4

```
private void btnAdd_Click(object sender, EventArgs e)
{
    Town town = new Town();

    fTown ft = new fTown(ref town);
    if (ft.ShowDialog() == DialogResult.OK)
    {
        bindSrcTowns.Add(town);
    }
}
```

7. Створити оброблювач події Click для кнопки btnEdit ("Редагувати запис") і записати код з лістингу 10.5.

Лістинг 10.5

```
private void btnEdit_Click(object sender, EventArgs e)
{
    Town town = (Town)bindSrcTowns.List[bindSrcTowns.Position];

    fTown ft = new fTown(ref town);
    if (ft.ShowDialog() == DialogResult.OK)
    {
        bindSrcTowns.List[bindSrcTowns.Position] = town;
    }
}
```

8. Створити оброблювач події Click для кнопки btnDel ("Видалити запис") і записати код з лістингу 10.6.

Лістинг 10.6

```
private void btnDel_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Видалити поточний запис?",
           "Видалення запису", MessageBoxButtons.OKCancel,
           MessageBoxIcon.Warning) == DialogResult.OK)
    {
        bindSrcTowns.RemoveCurrent();
    }
}
```

9. Створити оброблювач події Click для кнопки btnClear ("Очистити дані") і записати код з лістингу 10.7.

Лістинг 10.7

```
private void btnClear_Click(object sender, EventArgs e)
{
    if (MessageBox.Show(
        "Очистити таблицю?\n\nВсі дані будуть втрачені",
        "Очищення даних", MessageBoxButtons.OKCancel,
        MessageBoxIcon.Question) == DialogResult.OK)
    {
        bindSrcTowns.Clear();
    }
}
```

10. Створити оброблювач події Click для кнопки btnExit ("Вийти з програми") і записати код з лістингу 10.8.

Лістинг 10.8

```
private void btnExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Закрити застосунок?", "Вихід з програми",
       MessageBoxButtons.OKCancel,
        MessageBoxIcon.Question) == DialogResult.OK)
    {
        Application.Exit();
    }
}
```

11. Запустити застосунок і протестувати його роботу. Слід перевірити функціонал всіх кнопок панелі інструментів: додавання запису, його зміна, видалення, очищення таблиці, вихід з програми (див. рис. 10.6).

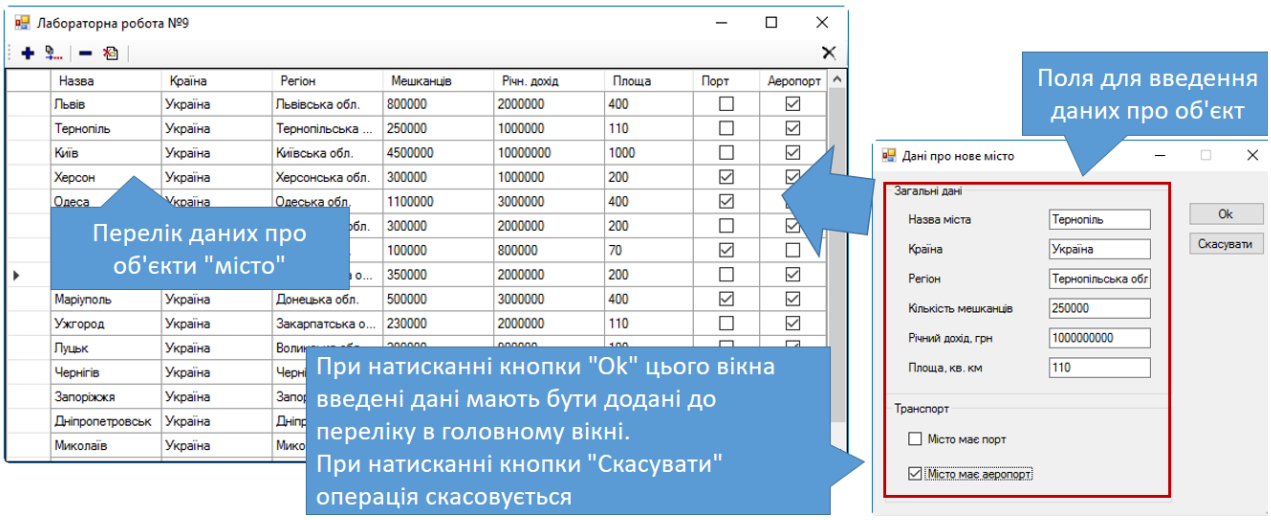


Рис. 10.6. Схема функціонування застосунку

Завдання для самостійного опрацювання

Вирішити розглянуте раніше завдання щодо предметної області, вказаної у завданні до лабораторної роботи №5. Для опису об'єкта використати клас, розроблений при виконанні лабораторної роботи №8.

Лабораторна робота №11

Тема. Проектування класу за принципами інкапсуляції.

Мета. Ознайомлення принципами інкапсуляції та їх застосування на практиці.

Завдання

Розробити клас, який описує об'єкт "коло" та забезпечує виконання об'єктом такого переліку дій: відображення об'єкта на екрані, приховування об'єкта, переміщення об'єкта по екрану, та збільшення/ зменшення розмірів об'єкта. Розробити застосунок, який демонструватиме всі перелічені можливості роботи з об'єктом "коло".

Виконання завдання

В середовищі Microsoft Visual Studio створити новий проект застосунку Windows Forms. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон Windows Forms Application у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab11) та його розташування на диску.

Створення класу, який описує об'єкт "коло" - CCircle

За завданням програма має працювати з об'єктами типу "голо". Такий об'єкт опишемо у класі CCircle. Щоб створити новий файл з описом класу:

1. У контекстному меню для елемента, який відповідає проекту (у вікні Solution Explorer), вибрати команду Add\New Item... У вікні, яке з'явиться, в переліку Visual C# Items вибрати пункт Class, ввести назву класу – CCircle, і натиснути кнопку Add.
2. У файлі CCircle.cs (він відкриється у редакторі коду) ввести оголошення класу з лістингу 11.1.

Лістинг 11.1

```
using System.Drawing;

class CCircle
{
    // Константи
    const int DefaultRadius = 50; // Радіус кола за замовчуванням, пікс

    // Поля
    private Graphics graphics;
    private int _radius;    // Підтримуюче поле для властивості Radius

    // Властивості
    public int X { get; set; }    // Координата X центра кола
    public int Y { get; set; }    // Координата Y центра кола
    public int Radius {             // Радіус кола
        get
        {
```

```
            return _radius;
        }
        set
        {
            _radius = value >= 200 ? 200 : value;
            _radius = value <= 5  ? 5  : value;
        }
    }

    // Конструктор, створює об'єкт кола (для заданої поверхні
    // малювання GDI+) з заданими координатами.
    // Радіус кола приймається за замовчуванням
    public CCircle(Graphics graphics, int X, int Y)
    {
        this.graphics = graphics;
        this.X = X;
        this.Y = Y;
        this.Radius = DefaultRadius;
    }

    // Конструктор, створює об'єкт кола (для заданої поверхні
    // малювання GDI+) з заданими координатами та радіусом
    public CCircle(Graphics graphics, int X, int Y, int Radius)
    {
        this.graphics = graphics;
        this.X = X;
        this.Y = Y;
        this.Radius = Radius;
    }

    // Малює коло на поверхні малювання GDI+.
    // Параметри кола задає перо pen
    private void Draw(Pen pen)
    {
        Rectangle rectangle = new Rectangle(X - Radius, Y - Radius,
            2 * Radius, 2 * Radius);
        graphics.DrawEllipse(pen, rectangle);
    }

    // Показує коло (малює на поверхні малювання GDI+ кольором
    // переднього плану)
    public void Show()
    {
        Draw(Pens.Red);
    }

    // Приховує коло (малює на поверхні малювання GDI+ кольором фону)
    public void Hide()
    {
        Draw(Pens.White);
    }

    // Розширює коло: збільшує радіус на один піксель
    public void Expand()
    {
        Hide();
        Radius++;
        Show();
    }

    // Розширює коло: збільшує радіус на dR пікселів
    public void Expand(int dR)
    {
        Hide();
        Radius = Radius + dR;
        Show();
    }
```

```
    // Стискає коло: зменшує радіус на один піксель
    public void Collapse()
    {
        Hide();
        Radius--;
        Show();
    }

    // Стискає коло: зменшує радіус на dR пікселів
    public void Collapse(int dR)
    {
        Hide();
        Radius = Radius - dR;
        Show();
    }

    // Переміщує коло
    public void Move(int dX, int dY)
    {
        Hide();
        X = X + dX;
        Y = Y + dY;
        Show();
    }

}  // Кінець оголошення класу
```

Розробка головного вікна

Загальний вигляд головного вікна застосунку приведено на рис. 11.1.

1. Для головного вікна та класу, який його описує, задати назву. З цією метою викликати контекстне меню на відповідному елементі Solution Explorer, вибрати команду Rename та ввести назву fMain.

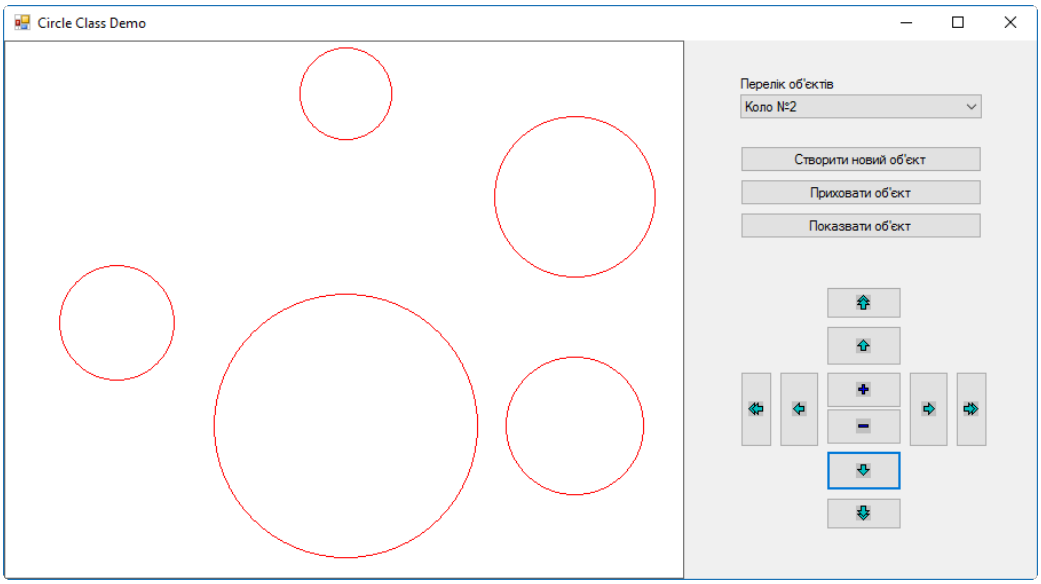


Рис. 11.1. Головне вікно програми для керування об'єктами "коло"

2. Для форми задати такі властивості:
    - StartPosition = CenterScreen
    - Text = "Лабораторна робота №11"

3. Всю область вікна поділимо на дві частини: перша (ліва) буде призначена для відображення графічного подання об'єктів (кіл), а друга (права) міститиме елементи керування програмою (див. рис. 11.1). Для цього розмістимо у вікні два компоненти Panel. Обидві панелі слід розташувати відповідно та задати розміри, достатні для розміщення компонентів.
4. Для лівої панелі (призначена для малювання) слід задати властивості:
    - Name = pnMain
    - BackColor = White (білий буде кольором фону)
    - BorderStyle = FixedSingle
5. Для правої панелі (призначена для керування) слід задати властивості:
    - Name = pnTools
6. Розмістити на правій панелі елементи керування роботою програми та задати їх властивості відповідно до рис. 11.2. Використано один компонент Label, один компонент Combobox, і три компоненти Button.

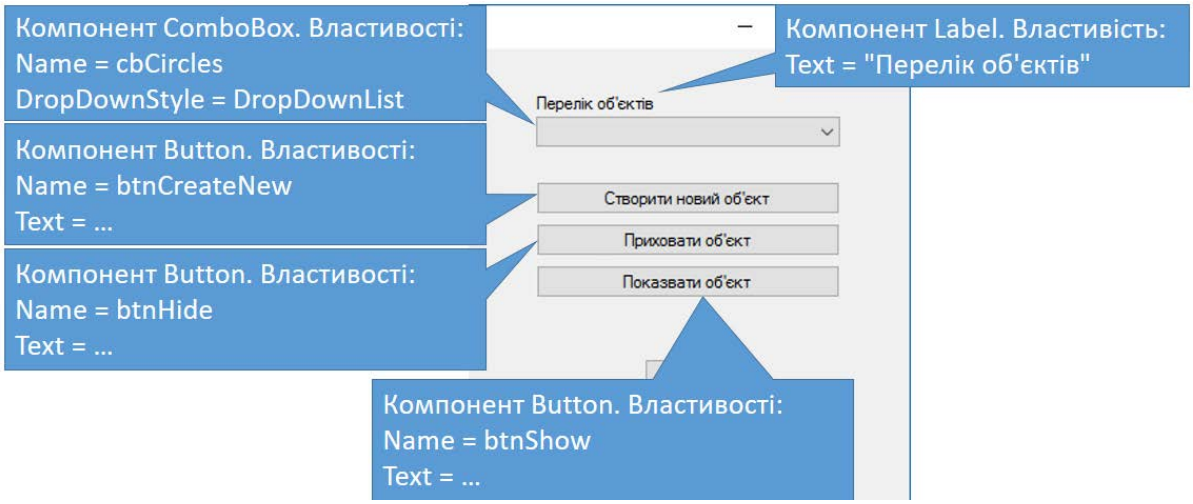


Рис. 11.2. Елементи керування програмою

7. Розмістити на правій панелі кнопки для маніпулювання об'єктами "коло" (рис. 11.3). Кнопкам слід задати назву (властивість Name) та малюнок (Image). Кнопки з однією стрілкою призначені для переміщення кола на 10 пікс. Кнопки з двома стрілками – для переміщення на більшу відстань. Кнопка зі знаком "+" призначена для збільшення розміру об'єкта, а кнопка з "-" для зменшення розміру об'єкта. Піктограми для кнопок можна задати на свій розсуд, але вони мають відповідати призначенню.
8. Після того, як на правій панелі розташовано всі потрібні компоненти і задано її підходящий розмір, для неї слід задати властивість Dock = Right. Потім для лівої панелі задати властивість Dock = Fill. Тоді при зміні розмірів вікна права панель завжди матиме фіксовану ширину і буде прив'язана до лівого краю вікна, а ліва займатиме весь інший простір вікна.
9. У класі головного вікна оголосити поле – масив circles, який міститиме перелік посилань на об'єкти кіл (екземпляри класу CCircle):

```
CCircle[] circles;
```

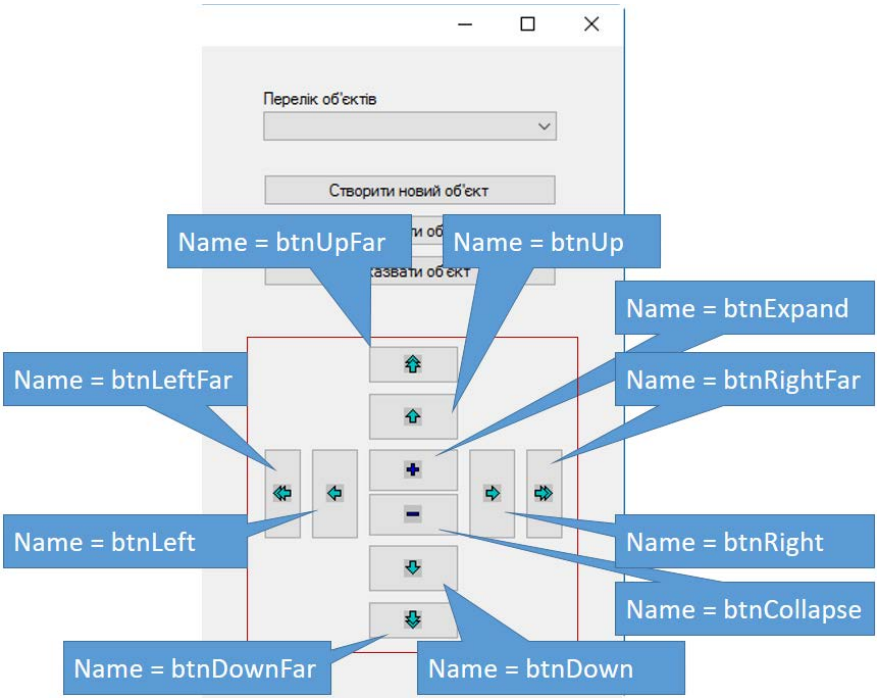


Рис. 11.3. Елементи для маніпулювання об'єктами "коло"

10. У класі головного вікна оголосити поле CircleCount, яке міститиме кількість створених об'єктів – кіл.

```
int CircleCount = 0;
```

11. У класі головного вікна оголосити поле CurrentCircleIndex, яке міститиме індекс поточного кола, з яким ведеться робота.

```
int CurrentCircleIndex;
```

12. У конструкторі класу форми дописати виділений фрагмент з лістингу 11.2.

Лістинг 11.2

```
public fMain()
{
    InitializeComponent();

    // Створення масиву для об'єктів - кіл
    circles = new CCircle[100];
}
```

13. Створити оброблювач події Click кнопки btnCreateNew (лістинг 11.3).

Лістинг 11.3

```
private void btnCreateNew_Click(object sender, EventArgs e)
{
    if (CircleCount >= 99)
    {
        MessageBox.Show("Досягнуто межі кількості об'єктів!");
        return;
    }
    Graphics graphics = pnMain.CreateGraphics();
    CurrentCircleIndex = CircleCount;

    // Створення нового об'єкта - екземпляра класу CCircle
    circles[CurrentCircleIndex] =
        new CCircle(graphics, pnMain.Width / 2,
        pnMain.Height / 2, 50);
    circles[CurrentCircleIndex].Show();
```

```
    CircleCount++;

    cbCircles.Items.Add("Коло №" + (CircleCount - 1).ToString());
    cbCircles.SelectedIndex = CircleCount - 1;
}
```

14. Створити оброблювач події Click кнопки btnHide (лістинг 11.4).

Лістинг 11.4

```
private void btnHide_Click(object sender, EventArgs e)
{
    CurrentCircleIndex = cbCircles.SelectedIndex;
    if ((CurrentCircleIndex > CircleCount) || (CurrentCircleIndex < 0))
        return;

    // ------------------------------------------------------------------
    // Приховування поточного об'єкта - екземпляра класу CCircle
    circles[CurrentCircleIndex].Hide();
    // ------------------------------------------------------------------
}
```

15. Створити оброблювач події Click кнопки btnShow (лістинг 11.5).

Лістинг 11.5

```
private void btnShow_Click(object sender, EventArgs e)
{
    CurrentCircleIndex = cbCircles.SelectedIndex;
    if ((CurrentCircleIndex > CircleCount) || (CurrentCircleIndex < 0))
        return;

    // ------------------------------------------------------------------
    // Відображення поточного об'єкта
    circles[CurrentCircleIndex].Show();
    // ------------------------------------------------------------------
}
```

16. Створити оброблювач події Click кнопки btnExpand (лістинг 11.6).

Лістинг 11.6

```
private void btnExpand_Click(object sender, EventArgs e)
{
    CurrentCircleIndex = cbCircles.SelectedIndex;
    if ((CurrentCircleIndex > CircleCount) || (CurrentCircleIndex < 0))
        return;

    // ------------------------------------------------------------------
    // Розширення поточного об'єкта - екземпляра класу CCircle
    circles[CurrentCircleIndex].Expand(5);
    // ------------------------------------------------------------------
}
```

17. Створити оброблювач події Click кнопки btnCollapse (лістинг 11.7).

Лістинг 11.7

```
private void btnCollapse_Click (object sender, EventArgs e)
{
    CurrentCircleIndex = cbCircles.SelectedIndex;
    if ((CurrentCircleIndex > CircleCount) || (CurrentCircleIndex < 0))
        return;
```

```
    // ------------------------------------------------------------
    // Стискання поточного об'єкта - екземпляра класу CCircle
    circles[CurrentCircleIndex].Collapse(5);
    // ------------------------------------------------------------
}
```

18. Створити оброблювач події Click кнопки btnUp (лістинг 11.8).

Лістинг 11.8

```
private void btnUp_Click(object sender, EventArgs e)
{
    CurrentCircleIndex = cbCircles.SelectedIndex;
    if ((CurrentCircleIndex > CircleCount) || (CurrentCircleIndex < 0))
        return;

    // ------------------------------------------------------------
    // Переміщення вверх поточного об'єкта
    circles[CurrentCircleIndex].Move(0, -10);
    // ------------------------------------------------------------
}
```

19. Створити оброблювач події Click кнопки btnDown (лістинг 11.9).

Лістинг 11.9

```
private void btnDown_Click(object sender, EventArgs e)
{
    CurrentCircleIndex = cbCircles.SelectedIndex;
    if ((CurrentCircleIndex > CircleCount) || (CurrentCircleIndex < 0))
        return;

    // ------------------------------------------------------------
    // Переміщення вниз поточного об'єкта
    circles[CurrentCircleIndex].Move(0, 10);
    // ------------------------------------------------------------
}
```

20. Створити оброблювач події Click кнопки btnRight (лістинг 11.10).

Лістинг 11.10

```
private void btnRight_Click(object sender, EventArgs e)
{
    CurrentCircleIndex = cbCircles.SelectedIndex;
    if ((CurrentCircleIndex > CircleCount) || (CurrentCircleIndex < 0))
        return;

    // ------------------------------------------------------------
    // Переміщення вправо поточного об'єкта
    circles[CurrentCircleIndex].Move(10, 0);
    // ------------------------------------------------------------
}
```

21. Створити оброблювач події Click кнопки btnLeft (лістинг 11.11).

Лістинг 11.11

```
private void btnLeft_Click(object sender, EventArgs e)
{
    CurrentCircleIndex = cbCircles.SelectedIndex;
    if ((CurrentCircleIndex > CircleCount) || (CurrentCircleIndex < 0))
        return;
```

```
    // ------------------------------------------------------------
    // Переміщення вліво поточного об'єкта
    circles[CurrentCircleIndex].Move(-10, 0);
    // ------------------------------------------------------------
}
```

22. Створити оброблювач події Click кнопки btnRightFar (лістинг 11.12).

Лістинг 11.12

```
private void btnRightFar_Click(object sender, EventArgs e)
{
    CurrentCircleIndex = cbCircles.SelectedIndex;
    if ((CurrentCircleIndex > CircleCount) || (CurrentCircleIndex < 0))
        return;

    // ------------------------------------------------------------
    // Переміщення поточного об'єкта вправо на більшу відстань
    for (int i = 0; i < 100; i++)
    {
        circles[CurrentCircleIndex].Move(1, 0);
        System.Threading.Thread.Sleep(5);
    }
    // ------------------------------------------------------------
}
```

23. Створити оброблювач події Click кнопки btnLeftFar (лістинг 11.13).

Лістинг 11.13

```
private void btnLeftFar_Click(object sender, EventArgs e)
{
    CurrentCircleIndex = cbCircles.SelectedIndex;
    if ((CurrentCircleIndex > CircleCount) || (CurrentCircleIndex < 0))
        return;

    // ------------------------------------------------------------
    // Переміщення поточного об'єкта вліво на більшу відстань
    for (int i = 0; i < 100; i++)
    {
        circles[CurrentCircleIndex].Move(-1, 0);
        System.Threading.Thread.Sleep(5);
    }
    // ------------------------------------------------------------
}
```

24. Створити оброблювач події Click кнопки btnUpFar (лістинг 11.14).

Лістинг 11.14

```
private void btnUpFar_Click(object sender, EventArgs e)
{
    CurrentCircleIndex = cbCircles.SelectedIndex;
    if ((CurrentCircleIndex > CircleCount) || (CurrentCircleIndex < 0))
        return;

    // ------------------------------------------------------------
    // Переміщення поточного об'єкта вверх на більшу відстань
    for (int i = 0; i < 100; i++)
    {
        circles[CurrentCircleIndex].Move(0, -1);
        System.Threading.Thread.Sleep(5);
    }
```

```
    // ------------------------------------------------------------------
}
```

25. Створити оброблювач події Click кнопки btnDownFar (лістинг 11.15).

Лістинг 11.15

```
private void btnDownFar_Click(object sender, EventArgs e)
{
    CurrentCircleIndex = cbCircles.SelectedIndex;
    if ((CurrentCircleIndex > CircleCount) || (CurrentCircleIndex < 0))
        return;

    // ------------------------------------------------------------------
    // Переміщення поточного об'єкта вниз на більшу відстань
    for (int i = 0; i < 100; i++)
    {
        circles[CurrentCircleIndex].Move(0, 1);
        System.Threading.Thread.Sleep(5);
    }
    // ------------------------------------------------------------------
}
```

26. Запустити застосунок (наприклад, натиснувши клавішу F5) та перевірити функціонал всіх кнопок програми.

Завдання для самостійного опрацювання

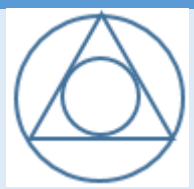
Розробити клас CEmblem, який описує об'єкт "емблема" (згідно варіанту) та забезпечує виконання об'єктом такого переліку дій: відображення об'єкта на екрані, приховування об'єкта, переміщення об'єкта по екрану, та збільшення/ зменшення розмірів об'єкта. Розробити застосунок, який демонструватиме всі перелічені можливості роботи з об'єктом "емблема".

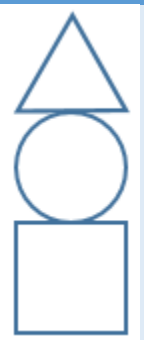
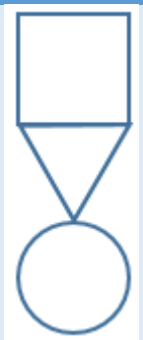
Примітки:

1) Всі об'єкти "емблема" складаються з базових фігур: квадрата, кола та рівностороннього трикутника.
2) Фігура, розташована в іншій, є геометрично вписаною.
3) Кути повороту фігур кратні 45^0.

Таблиця 11.1.

Варіант	Завдання	Варіант	Завдання
1.		**2.**	
3.		**4.**	

Варіант	Завдання	Варіант	Завдання
5.		6.	
7.		8.	
9.		10.	
11.		12.	
13.		14.	
15.		16.	
17.		18.	
19.		20.	
21.		22.	
23.		24.	

Варіант	Завдання	Варіант	Завдання
25.		26.	
27.		28.	
29.		30.	

Лабораторна робота №12

Тема. Спадкування класів.

Мета. Ознайомлення з принципами спадкування класів та їх застосування на практиці. Вивчення принципів використання абстрактних класів.

Завдання

Розробити абстрактний клас, який описує поведінку об'єкта "фігура" із сукупності об'єктів "геометричні фігури", та забезпечує виконання об'єктами такого переліку дій: відображення об'єкта на екрані, приховування об'єкта, переміщення об'єкта по екрану, та збільшення/ зменшення розмірів об'єкта. Розробити сукупність похідних класів, які описують об'єкти "коло", "прямокутник" та "трикутник". Розробити застосунок, який демонструватиме всі перелічені можливості роботи зі всіма розглянутими об'єктами сукупності "геометричні фігури".

Примітка. Сукупність "геометричні фігури" містить об'єкти "коло", "прямокутник", "трикутник" та інші прості фігури.

Виконання завдання

В середовищі Microsoft Visual Studio створити новий проект застосунку Windows Forms. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон Windows Forms Application у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab12) та його розташування на диску.

Створення абстрактного класу, який описує об'єкт "фігура" - CFigure

Загальні риси поведінки геометричних фігур опишемо у абстрактному класі CFigure. Щоб створити новий файл з описом класу:

1. у контекстному меню для елемента, який відповідає проекту (у вікні Solution Explorer), вибрати команду Add\New Item... У вікні, яке з'явиться, в переліку Visual C# Items вибрати пункт Class, ввести назву класу – CFigure, і натиснути кнопку Add.
2. у файлі CFigure.cs (він відкриється у редакторі коду) ввести оголошення класу (лістинг 12.1).

Лістинг 12.1

```
using System.Drawing;

abstract class CFigure
{
    // Поля
    protected Graphics graphics;

    // Властивості
    public int X { get; set; }      // Координата X центра фігури
```

```
    public int Y { get; set; }      // Координата Y центра фігури

    // Абстрактний метод: малює фігуру на поверхні малювання GDI+
    abstract protected void Draw(Pen pen);

    // Показує фігуру (малює на поверхні малювання GDI+ кольором
    // переднього плану)
    public void Show()
    {
        Draw(Pens.Red);
    }

    // Приховує фігуру (малює на поверхні малювання GDI+ кольором фону)
    public void Hide()
    {
        Draw(Pens.White);
    }

    // Переміщує фігуру
    public void Move(int dX, int dY)
    {
        Hide();

        X = X + dX;
        Y = Y + dY;

        Show();
    }

    // Розширює фігуру
    abstract public void Expand(int dR);

    // Стискає фігуру
    abstract public void Collapse(int dR);

}  // Кінець опису класу
```

Абстрактний клас CFigure в загальному (абстрактно) описує потрібну поведінку геометричної фігури. Геометрична фігура:

- має координати центра (властивості X, Y);
- може малюватися на екрані (метод Draw);
- може бути показана чи прихована (методи Show та Hide);
- може переміщуватися (метод Move);
- може розширюватися та стискатися (методи Expand та Collapse).

При цьому:

- метод Draw є абстрактним, бо промальовування різних видів геометричних фігур робиться по-різному і буде реалізоване у похідних класах. Методу Draw передаються параметри пера (колір), за якими буде намальовано фігуру;
- методи Show і Hide не є абстрактними, оскільки просто викликають метод Draw з параметрами пера для переднього плану (Draw) чи фону (Hide);
- методи Expand та Collapse є абстрактними, бо зміна розмірів фігур різних видів робиться по-різному (для кола міняємо радіус, для прямокутника – довжину сторони тощо);

- метод Move не є абстрактним, так як його робота основана на елемента, визначених у класі CFigure (фігура приховується, змінюється значення координат центра фігури, і фігура показується знову).

Створення класу, який описує об'єкт "коло" - CCircle

Об'єкт "коло" входить у множину об'єктів "геометричні фігури". Тому клас, який його описує – CCircle – слід створити на базі класу CFigure. Щоб створити новий файл з описом класу:

1. У контекстному меню для елемента, який відповідає проекту (у вікні Solution Explorer), вибрати команду Add\New Item... У вікні, яке з'явиться, в переліку Visual C# Items вибрати пункт Class, ввести назву класу – CCircle, і натиснути кнопку Add.
2. У файлі CCircle.cs (він відкриється у редакторі коду) ввести оголошення класу (лістинг 12.2).

Лістинг 12.2

```
using System.Drawing;

class CCircle : CFigure
{
    // Поля
    private int _radius;  // Підтримуюче поле для властивості Radius

    // Властивості
    public int Radius {   // Радіус кола
        get
        {
            return _radius;
        }
        set
        {
            _radius = value >= 200 ? 200 : value;
            _radius = value <= 5  ? 5  : value;
        }
    }

    // Конструктор, створює об'єкт кола (для заданої поверхні
    // малювання GDI+) з заданими координатами та радіусом
    public CCircle(Graphics graphics, int X, int Y, int Radius)
    {
        this.graphics = graphics;
        this.X = X;
        this.Y = Y;
        this.Radius = Radius;
    }

    // Малює коло на поверхні малювання GDI+
    protected override void Draw(Pen pen)
    {
        Rectangle rectangle = new Rectangle(X - Radius, Y - Radius,
            2 * Radius, 2 * Radius);
        graphics.DrawEllipse(pen, rectangle);
    }

    // Розширює коло: збільшує радіус на dR пікселів
    override public void Expand(int dR)
    {
        Hide();
        Radius = Radius + dR;
        Show();
    }
```

```
    // Стискає коло: зменшує радіус на dR пікселів
    override public void Collapse(int dR)
    {
        Hide();
        Radius = Radius - dR;
        Show();
    }
}  // Кінець оголошення класу
```

Створення класу, який описує об'єкт "прямокутник" - CRectangle

Об'єкт "прямокутник" входить у множину об'єктів "геометричні фігури". Тому клас, який його описує – CRectangle – також слід створити на базі класу CFigure.

Щоб створити новий файл з описом класу:

1. У контекстному меню для елемента, який відповідає проекту (у вікні Solution Explorer), вибрати команду Add\New Item... У вікні, яке з'явиться, в переліку Visual C# Items вибрати пункт Class, ввести назву класу – CRectangle, і натиснути кнопку Add.
2. У файлі CRectangle.cs (він відкриється у редакторі коду) ввести оголошення класу (лістинг 12.3).

Лістинг 12.3

```
using System.Drawing;

class CRectangle : CFigure
{
    // Поля
    private int _sideA;   // Підтримуюче поле для властивості SideA
    private int _sideB;   // Підтримуюче поле для властивості SideB

    // Властивості
    public int SideA
    {
        get
        {
            return _sideA;
        }
        set
        {
            _sideA = value >= 200 ? 200 : value;
            _sideA = value <= 5 ? 5 : value;
        }
    }

    public int SideB
    {
        get
        {
            return _sideB;
        }
        set
        {
            _sideB = value >= 200 ? 200 : value;
            _sideB = value <= 5 ? 5 : value;
        }
    }



    // Конструктор, створює об'єкт прямокутника (для заданої поверхні
    // малювання GDI+) з заданими координатами та довжиною сторін
```

```
    public CRectangle(Graphics graphics, int X, int Y, int SideA,
        int SideB)
    {
        this.graphics = graphics;
        this.X = X;
        this.Y = Y;
        this.SideA = SideA;
        this.SideB = SideB;
    }

    // Малює прямокутник на поверхні малювання GDI+
    protected override void Draw(Pen pen)
    {
        Rectangle rectangle = new Rectangle(X - SideA / 2,
            Y - SideB / 2, SideA, SideB);
        graphics.DrawRectangle(pen, rectangle);
    }

    // Розширює прямокутник: збільшує довжину стрін на dX пікселів
    override public void Expand(int dX)
    {
        Hide();
        SideA = SideA + dX;
        SideB = SideB + dX;
        Show();
    }

    // Стискає прямокутник: зменшує довжину сторін на dX пікселів
    override public void Collapse(int dX)
    {
        Hide();
        SideA = SideA - dX;
        SideB = SideB - dX;
        Show();
    }

}  // Кінець оголошення класу
```

Створення класу, який описує об'єкт "трикутник" - CTriangle

Об'єкт "трикутник" входить у множину об'єктів "геометричні фігури". Тому клас, який його описує – CTriangle – слід створити на базі класу CFigure.

Щоб створити новий файл з описом класу:

1. У контекстному меню для елемента, який відповідає проекту (у вікні Solution Explorer), вибрати команду Add\New Item... У вікні, яке з'явиться, в переліку Visual C# Items вибрати пункт Class, ввести назву класу – CTriangle, і натиснути кнопку Add.
2. У файлі CTriangle.cs (він відкриється у редакторі коду) ввести оголошення класу (лістинг 12.4).

Лістинг 12.4

```
using System;
using System.Drawing;

class CTriangle : CFigure
{
    // Поля
    private int _side;   // Підтримуюче поле для властивості Side
    // Властивості
    public int Side
    {
        get
```

```
        {
            return _side;
        }
        set
        {
            _side = value >= 200 ? 200 : value;
            _side = value <= 5 ? 5 : value;
        }
    }


    // Конструктор, створює об'єкт трикутника (для заданої поверхні
    // малювання GDI+) з заданими координатами та довжиною сторони
    public CTriangle(Graphics graphics, int X, int Y, int Side)
    {
        this.graphics = graphics;
        this.X = X;
        this.Y = Y;
        this.Side = Side;
    }

    // Малює трикутник на поверхні малювання GDI+
    protected override void Draw(Pen pen)
    {
        double r = (Side / 2) / Math.Sin(Math.PI / 3);
        Point p1 = new Point(X, Y - (int)r);
        Point p2 = new Point(X - (int)(r * Math.Cos(Math.PI / 6)),
            Y + (int)(r * Math.Sin(Math.PI / 6)));
        Point p3 = new Point(X + (int)(r * Math.Cos(Math.PI / 6)),
            Y + (int)(r * Math.Sin(Math.PI / 6)));
        Point[] triangle = { p1, p2, p3 };
        graphics.DrawPolygon(pen, triangle);
    }

    // Розширює трикутник: збільшує довжину стрін на dX пікселів
    override public void Expand(int dX)
    {
        Hide();
        Side = Side + dX;
        Show();
    }

    // Стискає трикутник: зменшує довжину сторін на dX пікселів
    override public void Collapse(int dX)
    {
        Hide();
        Side = Side - dX;
        Show();
    }

}  // Кінець оголошення класу
```

Розробка головного вікна

Загальний вигляд головного вікна застосунку приведено на рис. 12.1.

1. Для головного вікна та класу, який його описує, задати назву. З цією метою викликати контекстне меню на відповідному елементі Solution Explorer, вибрати команду Rename та ввести назву fMain.

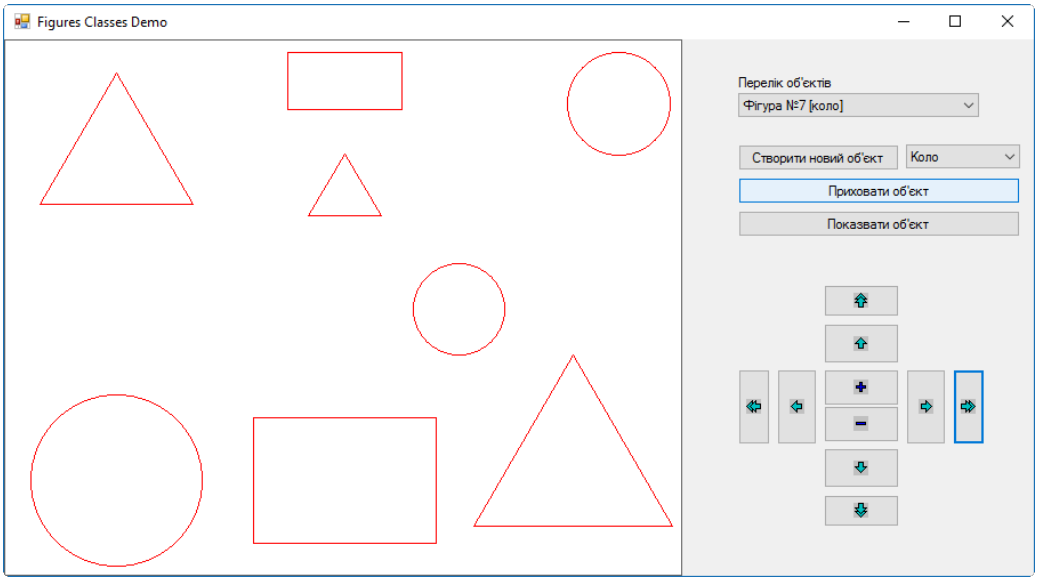


Рис. 12.1. Головне вікно програми

2. Для форми задати такі властивості:
   - StartPosition = CenterScreen
   - Text = "Лабораторна робота №12"
3. Всю область вікна поділимо на дві частини: перша (ліва) буде призначена для відображення графічного подання об'єктів (фігур), а друга (права) міститиме елементи керування програмою. Для цього розмістимо у вікні два компоненти Panel. Обидві панелі слід розташувати відповідно та задати розміри, достатні для розміщення компонентів.
4. Для лівої панелі (призначена для малювання) слід задати властивості:
   - Name = pnMain
   - BackColor = White (білий буде кольором фону)
   - BorderStyle = FixedSingle
5. Для правої панелі (призначена для керування) слід задати властивості:
   - Name = pnTools
6. Розмістити на правій панелі елементи керування роботою програми та задати їх властивості відповідно до рис. 12.2. Використано один компонент Label, два компоненти Combobox, і три компоненти Button.

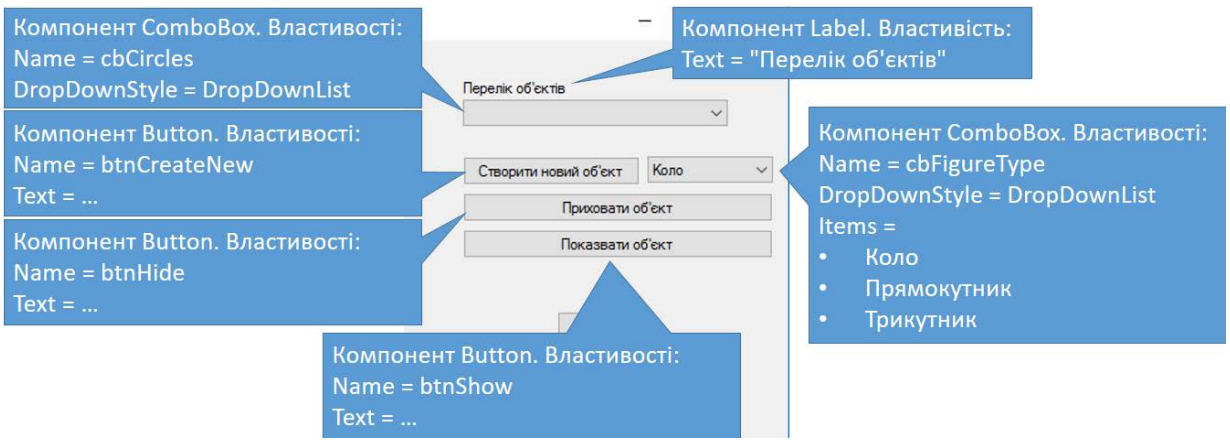


Рис. 12.2. Елементи керування програмою

7. Розмістити на правій панелі кнопки для маніпулювання об'єктами "фігура" (рис. 12.3). Кнопкам слід задати назву (властивість Name) та малюнок (Image). Кнопки з однією стрілкою призначені для переміщення кола на 10 пікс. Кнопки з двома стрілками – для переміщення на більшу відстань. Кнопка зі знаком "+" призначена для збільшення розміру об'єкта, а кнопка з "-" для зменшення розміру об'єкта. Піктограми для кнопок можна задати на свій розсуд, але вони мають відповідати призначенню.

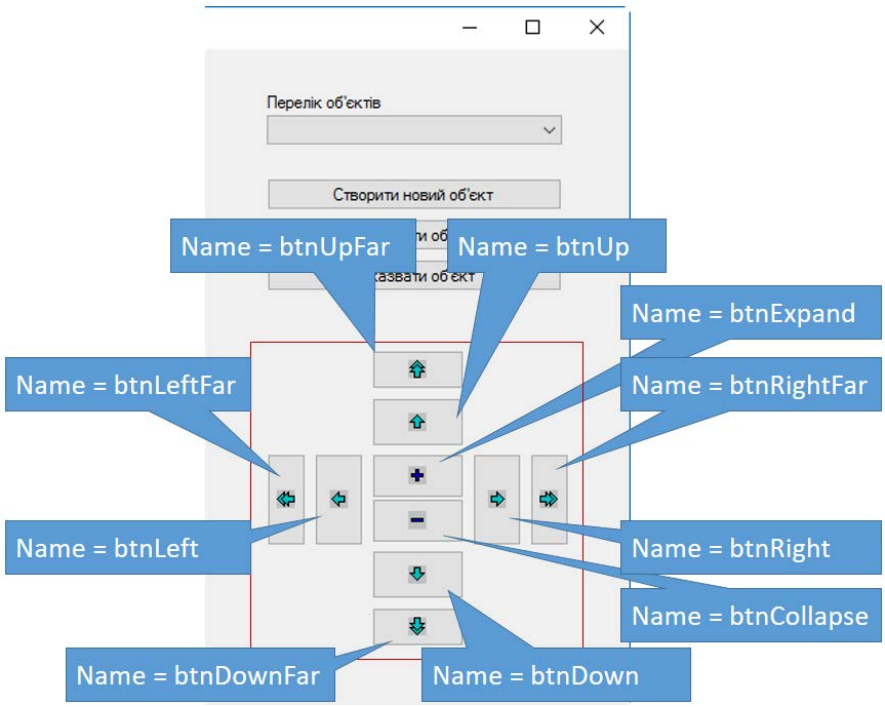


Рис. 12.3. Елементи для маніпулювання об'єктами "фігура"

8. Після того, як на правій панелі розташовано всі потрібні компоненти і задано її підходящий розмір, для неї слід задати властивість Dock = Right. Потім для лівої панелі задати властивість Dock = Fill. Тоді при зміні розмірів вікна права панель завжди матиме фіксовану ширину і буде прив'язана до лівого краю вікна, а ліва займатиме весь інший простір вікна.
9. У класі головного вікна описати поле – масив figures, який міститиме перелік посилань на об'єкти геометричних фігур:

```
CFigure[] figures;
```

10. У класі головного вікна описати поле FiguresCount, яке міститиме кількість створених об'єктів – фігур.

```
int FiguresCount = 0;
```

11. У класі головного вікна оголосити поле CurrentFigureIndex, яке міститиме індекс поточного кола, з яким ведеться робота.

```
int CurrentFigureIndex;
```

12. У конструкторі класу форми дописати виділений фрагмент з лістингу 12.5.

Лістинг 12.5

```
public fMain()
{
    InitializeComponent();

    // Створення масиву для об'єктів - фігур
    figures = new CFigure[100];
```

```
    cbFigureType.SelectedIndex = 0;
}
```

13. Створити оброблювач події Click кнопки btnCreateNew (лістинг 12.6).

Лістинг 12.6

```
private void btnCreateNew_Click(object sender, EventArgs e)
{
    if (FiguresCount >= 99)
    {
        MessageBox.Show("Досягнуто межі кількості об'єктів!");
        return;
    }
    Graphics graphics = pnMain.CreateGraphics();
    CurrentFigureIndex = FiguresCount;
    // Створення нового об'єкта - фігури
    if (cbFigureType.SelectedIndex == 0)      // Створюємо коло
    {
        figures[CurrentFigureIndex] =
            new CCircle(graphics, pnMain.Width / 2, pnMain.Height / 2,
                50);
        cbFigures.Items.Add("Фігура №" + (FiguresCount).ToString() +
            " [коло]");
    }

    else if (cbFigureType.SelectedIndex == 1) // Створюємо прямокутник
    {
        figures[CurrentFigureIndex] = new CRectangle(graphics,
            pnMain.Width / 2, pnMain.Height / 2, 100, 50);
        cbFigures.Items.Add("Фігура №" + (FiguresCount).ToString() +
            " [прямокутник]");
    }
    else if (cbFigureType.SelectedIndex == 2) // Створюємо трикутник
    {
        figures[CurrentFigureIndex] = new CTriangle(graphics,
            pnMain.Width / 2, pnMain.Height / 2, 100);
        cbFigures.Items.Add("Фігура №" + (FiguresCount).ToString() +
            " [трикутник]");
    }
    figures[CurrentFigureIndex].Show();
    FiguresCount++;
    cbFigures.SelectedIndex = FiguresCount - 1;
}
```

14. Створити оброблювач події Click кнопки btnHide (лістинг 12.7).

Лістинг 12.7

```
private void btnHide_Click(object sender, EventArgs e)
{
    CurrentFigureIndex = cbFigures.SelectedIndex;
    if ((CurrentFigureIndex > FiguresCount) ||
       (CurrentFigureIndex < 0))
        return;
    // ------------------------------------------------------------
    // Приховування поточного об'єкта
    figures[CurrentFigureIndex].Hide();
    // ------------------------------------------------------------
}
```

15. Створити оброблювач події Click кнопки btnShow (лістинг 12.8).

Лістинг 12.8

```
private void btnShow_Click(object sender, EventArgs e)
{
    CurrentFigureIndex = cbFigures.SelectedIndex;
    if ((CurrentFigureIndex > FiguresCount) ||
       (CurrentFigureIndex < 0))
        return;

    // ------------------------------------------------------------
    // Відображення поточного об'єкта
    figures[CurrentFigureIndex].Show();
    // ------------------------------------------------------------
}
```

16. Створити оброблювач події Click кнопки btnExpand (лістинг 12.9).

Лістинг 12.9

```
private void btnExpand_Click(object sender, EventArgs e)
{
    CurrentFigureIndex = cbFigures.SelectedIndex;
    if ((CurrentFigureIndex > FiguresCount) ||
       (CurrentFigureIndex < 0))
        return;
    // ------------------------------------------------------------
    // Розширення поточного об'єкта
    figures[CurrentFigureIndex].Expand(5);
    // ------------------------------------------------------------
}
```

17. Створити оброблювач події Click кнопки btnCollapse (лістинг 12.10).

Лістинг 12.10

```
private void btnCollapse_Click (object sender, EventArgs e)
{
    CurrentFigureIndex = cbFigures.SelectedIndex;
    if ((CurrentFigureIndex > FiguresCount) ||
       (CurrentFigureIndex < 0))
        return;
    // ------------------------------------------------------------
    // Розширення поточного об'єкта - екземпляра класу CCircle
    figures[CurrentFigureIndex].Collapse(5);
    // ------------------------------------------------------------
}
```

18. Створити оброблювач події Click кнопки btnUp (лістинг 12.11).

Лістинг 12.11

```
private void btnUp_Click(object sender, EventArgs e)
{
    CurrentFigureIndex = cbFigures.SelectedIndex;
    if ((CurrentFigureIndex > FiguresCount) ||
       (CurrentFigureIndex < 0))
        return;
    // ------------------------------------------------------------
    // Переміщення вверх поточного об'єкта
    figures[CurrentFigureIndex].Move(0, -10);
    // ------------------------------------------------------------
}
```

19. Створити оброблювач події Click кнопки btnDown (лістинг 12.12).

Лістинг 12.12

```
private void btnDown_Click(object sender, EventArgs e)
{
    CurrentFigureIndex = cbFigures.SelectedIndex;
    if ((CurrentFigureIndex > FiguresCount) ||
       (CurrentFigureIndex < 0))
        return;
    // ---------------------------------------------------------------
    // Переміщення вниз поточного об'єкта
    figures[CurrentFigureIndex].Move(0, 10);
    // ---------------------------------------------------------------
}
```

20. Створити оброблювач події Click кнопки btnRight (лістинг 12.13).

Лістинг 12.13

```
private void btnRight_Click(object sender, EventArgs e)
{
    CurrentFigureIndex = cbFigures.SelectedIndex;
    if ((CurrentFigureIndex > FiguresCount) ||
       (CurrentFigureIndex < 0))
        return;
    // ---------------------------------------------------------------
    // Переміщення вправо поточного об'єкта
    figures[CurrentFigureIndex].Move(10, 0);
    // ---------------------------------------------------------------
}
```

21. Створити оброблювач події Click кнопки btnLeft (лістинг 12.14).

Лістинг 12.14

```
private void btnLeft_Click(object sender, EventArgs e)
{
    CurrentFigureIndex = cbFigures.SelectedIndex;
    if ((CurrentFigureIndex > FiguresCount) ||
       (CurrentFigureIndex < 0))
        return;
    // ---------------------------------------------------------------
    // Переміщення вліво поточного об'єкта
    figures[CurrentFigureIndex].Move(-10, 0);
    // ---------------------------------------------------------------
}
```

22. Створити оброблювач події Click кнопки btnRightFar (лістинг 12.15).

Лістинг 12.15

```
private void btnRightFar_Click(object sender, EventArgs e)
{
    CurrentFigureIndex = cbFigures.SelectedIndex;
    if ((CurrentFigureIndex > FiguresCount) ||
       (CurrentFigureIndex < 0))
       return;
    // ---------------------------------------------------------------
    // Переміщення поточного об'єкта вправо на більшу відстань
    for (int i = 0; i < 100; i++)
    {
        figures[CurrentFigureIndex].Move(1, 0);
        System.Threading.Thread.Sleep(5);
    }
    // ---------------------------------------------------------------
}
```

23. Створити оброблювач події Click кнопки btnLeftFar (лістинг 12.16).

Лістинг 12.16

```
private void btnLeftFar_Click(object sender, EventArgs e)
{
    CurrentFigureIndex = cbFigures.SelectedIndex;
    if ((CurrentFigureIndex > FiguresCount) ||
       (CurrentFigureIndex < 0))
        return;
    // ---------------------------------------------------------------
    // Переміщення поточного об'єкта вліво на більшу відстань
    for (int i = 0; i < 100; i++)
    {
        figures[CurrentFigureIndex].Move(-1, 0);
        System.Threading.Thread.Sleep(5);
    }
    // ---------------------------------------------------------------
}
```

24. Створити оброблювач події Click кнопки btnUpFar (лістинг 12.17).

Лістинг 12.17

```
private void btnUpFar_Click(object sender, EventArgs e)
{
    CurrentFigureIndex = cbFigures.SelectedIndex;
    if ((CurrentFigureIndex > FiguresCount) ||
       (CurrentFigureIndex < 0))
        return;
    // ---------------------------------------------------------------
    // Переміщення поточного об'єкта вверх на більшу відстань
    for (int i = 0; i < 100; i++)
    {
        figures[CurrentFigureIndex].Move(0, -1);
        System.Threading.Thread.Sleep(5);
    }
    // ---------------------------------------------------------------
}
```

25. Створити оброблювач події Click кнопки btnDownFar (лістинг 12.18).

Лістинг 12.18

```
private void btnDownFar_Click(object sender, EventArgs e)
{
    CurrentFigureIndex = cbFigures.SelectedIndex;
    if ((CurrentFigureIndex > FiguresCount) ||
       (CurrentFigureIndex < 0))
        return;
    // ---------------------------------------------------------------
    // Переміщення поточного об'єкта вниз на більшу відстань
    for (int i = 0; i < 100; i++)
    {
        figures[CurrentFigureIndex].Move(0, 1);
        System.Threading.Thread.Sleep(5);
    }
    // ---------------------------------------------------------------
}
```

26. Запустити застосунок (наприклад, натиснувши клавішу F5) та перевірити функціонал всіх кнопок програми.

Завдання для самостійного опрацювання

Розробити клас CEmblem, похідний від класу CFigure, який описує об'єкт "емблема" (згідно варіанту, див. таблицю 11.1 завдання до лабораторної роботи №11) та забезпечує виконання об'єктом такого переліку дій: відображення об'єкта на екрані, приховування об'єкта, переміщення об'єкта по екрану. Розробити застосунок, який демонструватиме всі перелічені можливості роботи з об'єктом "Емблема".

Примітки:

1) Всі об'єкти "Емблема" складаються з базових фігур: квадрата, кола та рівностороннього трикутника.
2) Фігура, розташована в іншій, є геометрично вписаною.
3) Кути повороту фігур кратні 45^0.

Лабораторна робота №13

Тема. Робота з файлами.

Мета. Вивчення принципів читання та запису у текстові та бінарні файли за допомогою файлових потоків.

Завдання

Доповнити застосунок (розроблений протягом лабораторної роботи №10), який забезпечує внесення даних про об'єкти типу "місто" та відображення цих даних у табличній формі за допомогою елемента керування DataGridView. Для цього додати до панелі інструментів кнопки, при натисканні яких дані з таблиці зберігатимуться на диску в текстовому та двійковому вигляді, а також читатимуться і відображатимуться у вікні.

Виконання завдання

Імпорт даних з розробленого застосунку

1. В середовищі Microsoft Visual Studio створити новий проект застосунку Windows Forms. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон Windows Forms Application у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab13) та його розташування на диску.
2. З проекту лабораторної роботи №10 (з папки Lab10\Lab10) в папку нового проекту (в папку Lab13\Lab13) скопіювати такі файли:
    - fMain.cs, fMain.Designer.cs, fMain.resx
    - fTown.cs, fTown.Designer.cs, fTown.resx
    - Town.cs
3. Вибрати команду Project\ Add Existing Item, у діалоговому вікні вибрати три попередньо скопійовані файли: fMain.cs, fTown.cs та Town.cs і натиснути кнопку Add.
4. В Solution Explorer виділити елемент Form1.cs, викликати на ньому контекстне меню, вибрати команду Delete і підтвердити видалення автоматично згенерованої форми.
5. Відкрити файл Program.cs і змінити параметр виклику метода Application.Run – з "`new Form1()`" на "`new fMain()`"
6. Відкрити код для форми fMain.cs (клацнути в Solution Explorer на назву файлу і натиснути F7) і змінити назву простору імен на той, що вказано при створенні проекту (наприклад, з "`namespace Lab10`" на "`namespace Lab13`").
7. Відкрити код файлу fMain.Designer.cs і змінити назву простору імен (як у п.6).
8. Відкрити код файлу fTown.cs і змінити назву простору імен (як у п.6).
9. Відкрити код файлу fTown.Designer.cs і змінити назву простору імен (як у п.6).
10. Відкрити форму fMain у дизайнері форм та змінити її заголовок (властивість Text) на "Лабораторна робота №13"

11. Перевірити роботоздатність проекту, запустивши його (F5). Якщо все виконано вірно, запуститься застосунок, аналогічний до розробленого на лабораторній роботі №10. Якщо при компіляції будуть помилки, слід ще раз переглянути виконання попередніх пунктів (пп.2 – 11)

Розробка головної форми

1. На панель інструментів (представлену компонентом toolBar) перед кнопкою для виходу з програми додати чотири кнопки (рис. 13.1):

   - Для збереження вмісту таблиці в текстовому форматі.
   - Для збереження вмісту таблиці в бінарному форматі.
   - Для читання збережених текстових даних.
   - Для читання збережених бінарних даних.

   Після останньої кнопки додати розділювач (Separator).
   Задати для розміщених кнопок назви (властивість Name) – btnSaveAsText, btnSaveAsBinary, btnOpenFromText, btnOpenFromBinary та текст підказки (властивість ToolTipText). Наприклад, для першої кнопки ToolTipText можна задати текст "Зберегти у текстовому форматі" і т.д.

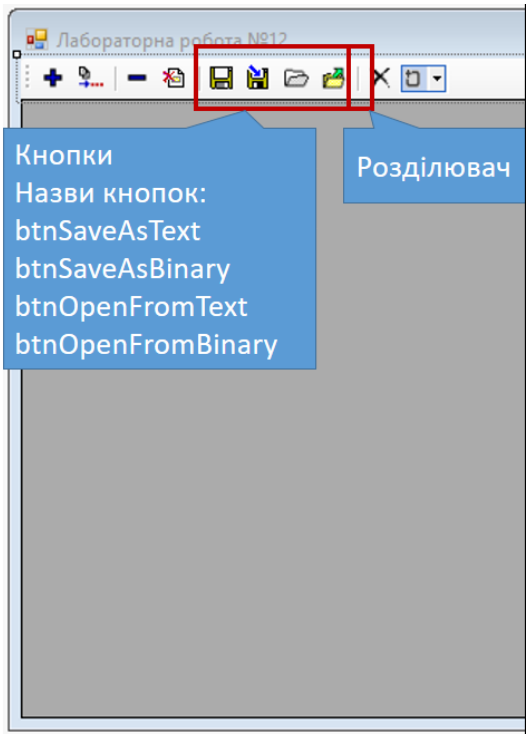


Рис. 13.1. Кнопки для файлових операцій

2. Змінити оброблювач події Resize головного вікна. Тут виконується припасування положення кнопки btnExit для виходу з програми до правого краю вікна. Для цього слід задати відповідний відступ зліва для кнопки btnExit. Оскільки на панель інструментів додано ще чотири кнопки, то відступ буде інший, ніж для лабораторної роботи №10. Код оброблювача події Resize форми fMain приведено у лістингу 13.1:

Лістинг 13.1

```
private void fMain_Resize(object sender, EventArgs e)
{
    int buttonsSize = 9 * btnAdd.Width + 3 * tsSeparator1.Width
    btnExit.Margin = new Padding(Width - buttonsSize, 0, 0, 0);
}
```

3. Додати у дизайнер форм компонент SaveFileDialog для вибору користувачем назви та розміщення файлу, в який буде записано дані. Задати його властивість:
   - Name = saveFileDialog
4. Для збереження даних у текстовому форматі використаємо такий алгоритм:
   1) Показати стандартне вікно для вибору файлу, в який будуть записані дані. Якщо користувач підтвердив операцію збереження, то:
      - Створити потік для текстового запису в файл StreamWriter

- У циклі пройти по всіх рядках, в межах кожного рядка пройти по всіх його комірках і записати значення кожної комірки (додаючи символ табуляції для розділення двох послідовно записаних текстових значень) у потік.
- Закрити потік запису в файл.

5. Створити оброблювач події Click кнопки btnSaveAsText і записати код з лістингу 13.2.

Лістинг 13.2

```
saveFileDialog.Filter =
    "Текстові файли (*.txt)|*.txt|All files (*.*)|*.*";
saveFileDialog.Title = "Зберегти дані у текстовому форматі";
saveFileDialog.InitialDirectory = Application.StartupPath;

StreamWriter sw;

if (saveFileDialog.ShowDialog() == DialogResult.OK)
{
    sw = new StreamWriter(saveFileDialog.FileName, false,
        Encoding.UTF8);
    try
    {
        foreach (Town town in bindSrcTowns.List) {
          sw.Write(town.Name + "\t" + town.Country + "\t" +
            town.Region + "\t" + town.Population + "\t" +
            town.YearIncome + "\t" + town.Square + "\t" +
            town.HasPort + "\t" + town.HasAirport + "\t\n");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Сталась помилка: \n{0}", ex.Message,
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        sw.Close();
    }
}
```

6. Перевірити функціональність кнопки btnSaveAsText. Для цього запустити програму, ввести у вікні програми кілька записів про міста і зберегти їх у текстовому файлі. Перевірити його вміст: він матиме приблизно такий вид (конкретні дані можуть бути різними):

```
Львів  Україна      Львівська обл.      800000 2000000      182    False  True
Київ   Україна      Київська обл.3800000       5000000000   890    True   True
Тернопіль    Україна      Тернопільська обл.  215000 2000000      72     False  True
Хмельницький Україна      Хмельницька обл.    262000 20000000     90     False  True
```

7. Для збереження даних у бінарному форматі використаємо такий алгоритм:
   2) Показати стандартне вікно для вибору файлу, в який будуть записані дані. Якщо користувач підтвердив операцію збереження, то:
   - Створити потік для запису в файл BinaryWriter
   - У циклі пройти по всіх рядках, в межах кожного рядка пройти по всіх його комірках і записати значення кожної комірки згідно його типу у потік.
   - Закрити потік запису в файл.
8. Створити оброблювач події Click кнопки btnSaveAsBinary і записати код з лістингу 13.3.

Лістинг 13.3

```
saveFileDialog.Filter =
    "Файли даних (*.towns)|*.towns|All files (*.*)|*.*";
saveFileDialog.Title = "Зберегти дані у бінарному форматі";
saveFileDialog.InitialDirectory = Application.StartupPath;
BinaryWriter bw;
if (saveFileDialog.ShowDialog() == DialogResult.OK)
{
    bw = new BinaryWriter(saveFileDialog.OpenFile());
    try
    {
        foreach (Town town in bindSrcTowns.List)
        {
            bw.Write(town.Name);
            bw.Write(town.Country);
            bw.Write(town.Region);
            bw.Write(town.Population);
            bw.Write(town.YearIncome);
            bw.Write(town.Square);
            bw.Write(town.HasPort);
            bw.Write(town.HasAirport);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Сталась помилка: \n{0}", ex.Message,
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        bw.Close();
    }
```

9. Перевірити функціональність кнопки btnSaveAsBinary. Для цього запустити програму, ввести у вікні програми кілька записів про міста і зберегти їх у бінарному файлі. Перевірити його вміст: він матиме приблизно такий вид (конкретні дані можуть бути різними):

```
Львів□Україна□Львівська обл. 5♠     □□>A     ◆@ □□Київ□Україна□Київська
обл._9      _ Ъ   Ъ@□□□Тернопіль□Україна"Тернопільська
обл.□    □□>A     R@ □□Хмельницький□УкраїнаХмельницька обл.р◆ BsA     □V@ □
```

10. Для читання даних у текстовому форматі використаємо такий алгоритм:
    1) Показати стандартне вікно для вибору файлу, з якого мають бути прочитані дані. Якщо користувач підтвердив операцію відкривання, то:
        - Створити потік StreamReader для читання текстового файлу
        - Доки не досягнуто кінця файлу:
            - Прочитати з файлу рядок.
            - Розділити його за символами табуляції на окремі значення.
            - Створити на основі отриманих значень об'єкт Town і додати його до джерела даних bindSrcTowns.
        - Закрити потік запису в файл.
11. Створити оброблювач події Click кнопки btnOpenFromText і записати код з лістингу 13.4.

Лістинг 13.4

```
openFileDialog.Filter = "Текстові файли (*.txt)|*.txt|All files
(*.*)|*.*";
openFileDialog.Title = "Прочитати дані у текстовому форматі";
openFileDialog.InitialDirectory = Application.StartupPath;
```

```
StreamReader sr;
if (openFileDialog.ShowDialog() == DialogResult.OK)
{
    bindSrcTowns.Clear();
    sr = new StreamReader(openFileDialog.FileName, Encoding.UTF8);
    string s;

    try
    {
        while ((s = sr.ReadLine()) != null) {
            string[] split = s.Split('\t');
            Town town = new Town(split[0], split[1], split[2],
              int.Parse(split[3]), double.Parse(split[4]),
              double.Parse(split[5]), bool.Parse(split[6]),
              bool.Parse(split[7]));
            bindSrcTowns.Add(town);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Сталась помилка: \n{0}", ex.Message,
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        sr.Close();
    }
}
```

12. Перевірити функціональність кнопки btnOpenFromText. Для цього запустити програму, очистити вміст таблиці і відкрити попередньо збережений текстовий файл. Програма має правильно відобразити збережені дані у таблиці.
13. Для читання даних у бінарному форматі використаємо такий алгоритм:
    1) Показати стандартне вікно для вибору файлу, з якого мають бути прочитані дані. Якщо користувач підтвердив операцію відкривання, то:
        1. Створити потік BinaryReader для читання файлу
        2. Доки не досягнуто кінця файлу:
            - Прочитати дані для кожного поля об'єкта "Місто" у тому ж порядку, в якому вони були записані.
            - Створити на основі отриманих значень об'єкт Town і додати його до джерела даних bindSrcTowns.
        3. Закрити потік запису в файл.
14. Створити оброблювач події Click кнопки btnOpenFromBinary і записати код з лістингу 13.5.

Лістинг 13.5

```
openFileDialog.Filter = "Файли даних (*.towns)|*.towns|All files
(*.*)|*.*";
openFileDialog.Title = "Прочитати дані у бінарному форматі";
openFileDialog.InitialDirectory = Application.StartupPath;
BinaryReader br;
if (openFileDialog.ShowDialog() == DialogResult.OK)
{
    bindSrcTowns.Clear();
    br = new BinaryReader(openFileDialog.OpenFile());
    try
    {
        Town town;
        while (br.BaseStream.Position < br.BaseStream.Length)
        {
```

```
                town = new Town();
                for (int i = 1; i <= 8; i++)
                {
                    switch (i)
                    {
                        case 1:
                            town.Name = br.ReadString();
                            break;
                        case 2:
                            town.Country = br.ReadString();
                            break;
                        case 3:
                            town.Region = br.ReadString();
                            break;
                        case 4:
                            town.Population = br.ReadInt32();
                            break;
                        case 5:
                            town.YearIncome = br.ReadDouble();
                            break;
                        case 6:
                            town.Square = br.ReadDouble();
                            break;
                        case 7:
                            town.HasPort = br.ReadBoolean();
                            break;
                        case 8:
                            town.HasAirport = br.ReadBoolean();
                            break;
                    }
                }
                bindSrcTowns.Add(town);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Сталась помилка: \n{0}", ex.Message,
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            br.Close();
        }
    }
}
```

15. Перевірити функціональність кнопки btnOpenFromBinary. Для цього запустити програму, очистити вміст таблиці і відкрити попередньо збережений бінарний файл. Програма має правильно відобразити збережені дані у таблиці.

Завдання для самостійного опрацювання

Вирішити попередньо розглянуте завдання щодо застосунку, розробленого згідно самостійного завдання для лабораторної роботи №10.

Лабораторна робота №14

Тема. Робота з колекціями та інтерфейсами.

Мета. Вивчення принципів роботи з колекціями типу "список". Ознайомлення з принципами використання інтерфейсів.

Завдання

Розробити консольний застосунок, який читатиме дані з бінарного файлу, що містить інформацію про об'єкти "місто" (файл записує програма, розроблена протягом лабораторної роботи №13). Дані слід подати у виді об'єктів класу Town і розмістити у колекції (за основу використати попередньо розроблений клас Town). Існуючий клас доповнити таким чином, щоб він відповідав вимогам інтерфейсу IComparable – це дозволить сортувати об'єкти колекції.

Програма повинна:

- відобразити на екрані прочитані з файлу несортовані дані колекції;
- відсортувати колекцію по зростанню значень;
- відобразити на екрані відсортовані дані колекції;
- додати у колекцію ще один об'єкт "Місто";
- відобразити на екрані відсортовані дані доповненої колекції;
- видалити з колекції довільний елемент (наприклад, останній);
- відобразити на екрані дані кінцевої колекції.

Виконання завдання

Імпорт даних з розробленого застосунку

1. В середовищі Microsoft Visual Studio створити новий проект консольного застосунку. Для цього вибрати команду меню File/New/Project. У вікні New Project вибрати шаблон Console Application у розділі Templates/ Visual C#/ Windows. Далі вказати назву проекту (наприклад, Lab14) та його розташування на диску.
2. З лабораторної роботи №13 (папка Lab13\Lab13) скопіювати файл Town.cs у проект для даної лабораторної роботи (папка Lab14\Lab14).
3. Додати скопійований файл до нового проекту. Для цього вибрати команду меню Project\Add Existing Item..., вибрати файл Town.cs і натиснути кнопку Add.

4. Скопіювати бінарний файл, отриманий після збереження даних у лабораторній роботі №13 (файл *.towns – з папки Lab13\Lab13\bin\Debug) у папку, де формується виконавчий файл нового проекту (папка Lab14\Lab14\bin\Debug). З цього файлу читатимуться дані для колекції.

Реалізація інтерфейсу IComparable у класі Town

Інтерфейс IComparable визначає правила сортування двох об'єктів довільного типу. Його реалізація у класі дозволить сортувати об'єкти – екземпляри цього класу. Щоб вказати, що клас спадкує певний інтерфейс, його назву слід вказати у переліку базових класів. Для внесення змін у клас Town слід відкрити файл Town.cs у редакторі коду. Код, який слід додати до опису класу Town, виділено синім кольором у лістингу 14.1.

Лістинг 14.1

```
using System;

public class Town : IComparable
{
    public string Name { get; set; }
    public string Country { get; set; }

    ...  // Фрагмент коду пропущено

    public string Info()
    {
        return Name + ", " + Country + ", " + Region;
    }
    public int CompareTo(object obj)
    {
        Town t = obj as Town;
        return string.Compare(this.Name, t.Name);
    }
}
```

Робота з колекцією

1. У класі Program (у файлі Program.cs), який представляє нашу програму, слід оголосити статичне поле – колекцію:

```
static List<Town> towns;
```

2. У класі Program оголосити метод, який виводитиме текстові дані про об'єкт "місто" (лістинг 14.2).

Лістинг 14.2

```
static void PrintTowns()
{
    foreach (Town town in towns)
    {
        Console.WriteLine(town.Info().Replace('i', 'i'));
    }
    Console.WriteLine();
}
```

3. У метод Main() ввести код з лістингу 14.3.

Лістинг 14.3

```
towns = new List<Town>();
FileStream fs = new FileStream("UATowns.towns", FileMode.Open);
BinaryReader reader = new BinaryReader(fs);
try
{
    Town town;
    Console.WriteLine("    Читаємо дані з файлу...\n");
    while (reader.BaseStream.Position < reader.BaseStream.Length)
    {
        town = new Town();
        for (int i = 1; i <= 8; i++)
        {
            switch (i)
            {
                case 1:
                    town.Name = reader.ReadString();
                    break;
                case 2:
                    town.Country = reader.ReadString();
                    break;
                case 3:
                    town.Region = reader.ReadString();
                    break;
                case 4:
                    town.Population = reader.ReadInt32();
                    break;
                case 5:
                    town.YearIncome = reader.ReadDouble();
                    break;
                case 6:
                    town.Square = reader.ReadDouble();
                    break;
                case 7:
                    town.HasPort = reader.ReadBoolean();
                    break;
                case 8:
                    town.HasAirport = reader.ReadBoolean();
                    break;
            }
        }
        towns.Add(town);
    }
}
catch (Exception ex)
{
    Console.WriteLine("Сталась помилка: {0}", ex.Message);
}
finally
{
    reader.Close();
}
Console.WriteLine("    Несортований перелік міст: {0}", towns.Count);
PrintTowns();
towns.Sort();
Console.WriteLine("    Сортований перелік міст: {0}", towns.Count);
PrintTowns();
Console.WriteLine("    Додаємо новий запис: Одеса");
Town townOdesa = new Town("Одеса", "Україна", "Одеська обл.", 997000,
    4000000, 237, true, true);
towns.Add(townOdesa);
towns.Sort();
Console.WriteLine("    Перелік міст: {0}", towns.Count);
PrintTowns();
Console.WriteLine("    Видаляємо останнє значення");
towns.RemoveAt(towns.Count - 1);
Console.WriteLine("    Перелік міст: {0}", towns.Count);
```

```
PrintTowns();
Console.ReadKey();
```

4. Запустити застосунок (F5) і перевірити його функціонал. Звернути увагу на те, як відбувається сортування переліку. Приклад консольного виводу розробленого застосунку приведено на рис. 14.1.

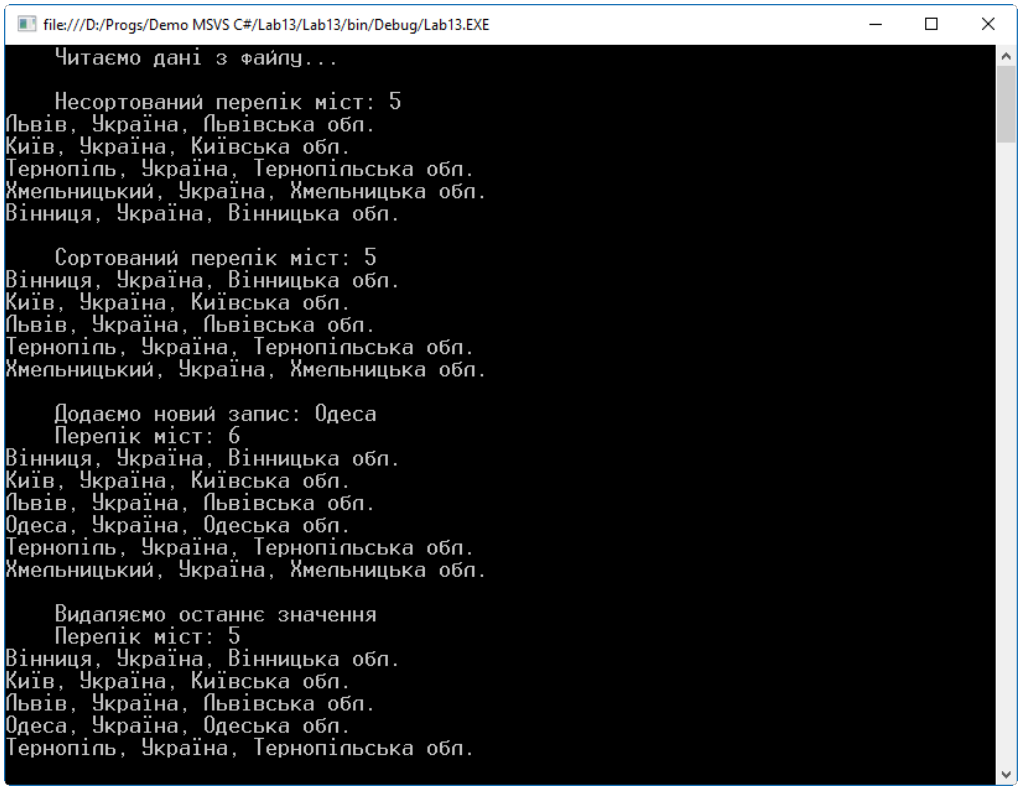


Рис. 14.1. Вивід переліку об'єктів "місто"

Завдання для самостійного опрацювання

Вирішити розглянуте раніше завдання щодо предметної області, вказаної у завданні до лабораторної роботи №5. Файл з даними для колекції, взяти, виконавши самостійне завдання для лабораторної роботи №10.