

Лабораторна робота № 1

МОВА ПРОГРАМУВАННЯ C#. ТИПИ ДАНИХ.

РЕАЛІЗАЦІЯ ОСНОВНИХ ТИПІВ АЛГОРИТМІВ

Мета роботи: Набути умінь і навичок створення простих консольних додатків мовою C# у середовищі Microsoft Visual Studio 2022.

Короткі теоретичні відомості

У мові C# підтримується стандартний набір типів даних. Однак кожен тип даних є об'єктом, що не передбачається в стандартному наборі. Для вирішення цього протиріччя в мові розрізняють: *типи значень (value type)* і *типи посилань (reference type)*. Типи значень включають в себе дані простого типу, а також переліковий тип даних і структури. До простих типів відносяться, наприклад, такі типи як char, int, float.

До типів посилань відносяться типи класів, інтерфейсів, масивів та делегатів.

До стандартних «простих» типів у мові C# відносяться наступні:

Тип	Опис	Область значень
object	Базовий клас для всіх інших типів	
string	Рядковий тип, послідовність символів Unicode	
sbyte	8-розрядне ціле число з знаком	-128 до 127
short	16-розрядне ціле число з знаком	-32768 до 32767
int	32-розрядне ціле число з знаком	-2147483648 до 2147483647
long	64-розрядне ціле число з знаком	-9223372036854775808 до 9223372036854775807
byte	8-розрядне ціле число без знака	0 до 255
ushort	16-розрядне ціле число без знака	0 до 65535

uint	32-розрядне ціле число без знака	0 до 4294967295
ulong	64-розрядне ціле число без знака	0 до 18446744073709551615
float	Число з плаваючою крапкою 4 байти, точність — 7 розрядів	$\pm 1,5 \cdot 10^{-45}$ до $\pm 3,4 \cdot 10^{33}$
double	Число з плаваючою крапкою 8 байт, точність — 16 розрядів	$\pm 5 \cdot 10^{-324}$ до $\pm 1,7 \cdot 10^{306}$
bool	Логічний тип	true або false
char	Тип символу Unicode	U+0000 до U+ffff
decimal	Тип десяткового числа 12 байт, точність — 28 розрядів	(Від $-7,9 \times 10^{28}$ до $7,9 \times 10^{28}$) / (10^{0-28})

Кожен із наведених типів є скороченням деякого системного типу. Наприклад, тип `Int` є альтернативною назвою структури `System.Int32`. В більшості випадків ці імена повністю рівноправні.

Переліковний тип це тип, який задається повним переліком своїх значень.

Значеннями перелікового типу можуть бути константи типів `byte`, `sbyte`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`. Якщо при описанні перелікового типу в явному вигляді не вказано тип його констант, то вважається, що константи типу `int`.

Всі константи повинні бути одного типу, який називається *базовим типом*.

Тип `char` не може використовуватися в якості базового типу. В переліковному типі можна задавати довільну кількість іменованих констант.

Загальне правило опису:

<атрибути> <модифікатори> `enum` <ім'я типу> [: <базовий тип>]

{

 <атрибути> <конст.1> [= <знач.1>],

 <атрибути> <конст.2> [= <знач.2>],

<атрибути> <конст.N> [= <знач.N>] }

Приклад:

enum Week : Byte

```
{  
    Monday,    // =0  
    Tuesday,   // =1  
    Wednesday, // =2  
    Thursday,  // =3  
    Friday,    // =4  
    Saturday,  // =5  
    Sunday     // =6  
}
```

У арифметичних виразах використовуються наступні *бінарні* арифметичні операції:

Операція	Позначення	Приклад
+	додавання	$z=x+y$
—	віднімання	$z=x-y$
*	множення	$z=x*y$
/	ділення	$z=x/y$
%	остача від ділення	$z=x\%y$

У мові C# є можливість здійснювати побітові операції над розрядами аргументів.

Операція	Позначення	Приклад
&	побітове «і»	int x=10; //x=1010 ₍₂₎ int y=7; //y=0111 ₍₂₎ z=x&y // z=2=0010 ₍₂₎
	побітове «або»	int x=10; //x=1010 ₍₂₎ int y=7; //y=0111 ₍₂₎ z=x y //z=15=1111 ₍₂₎

\wedge	Побітове «виключаюче або»	int x=10; //x=1010 ₍₂₎ int y=7; //y=0111 ₍₂₎ z=x^y //z=13=1101 ₍₂₎
----------	------------------------------	---

Якщо при виконанні бінарної операції результат зберігається у змінній, що є першим аргументом, то можна використати так звані операції присвоювання.

Операція	Приклад	Аналог з бінарними операціями
+=	int i = 5; i += 3; // i=8	int i = 5; i=i+3; // i=8
-=	int i = 5; i -= 3; // i=2	int i = 5; i=i-3; // i=2
*=	int i = 5; i *= 3; // i=15	int i = 5; i=i*3; // i=15
/=	int i = 6; i /= 3; // i=2	int i = 6; i=i/3; // i=2
>>=	int i = 5; i >>= 1; // i=2	int i = 5; i=i>>1; // i=2
<<=	int i = 5; i <<= 1; // i=10	int i = 5; i=i<<1; // i=10
&=	int i = 5; int j = 7; i &= j; // i=5	int i = 5; int j = 7; i=i&j; // i=5
^=	int i = 5; int j = 7; i ^= 1; // i=2	int i = 5; int j = 7; i=i^j; // i=2
=	int i = 5; int j = 7; i = 1; // i=7	int i = 5; int j = 7; i=i j; // i=7

Арифметичні вирази можуть містити різні математичні функції.

Назва	Опис	Результат	Пояснення
Тригонометричні функції			

Sin	Синус	double	Math.Sin(double x)
Cos	Косинус	double	Math.Cos(double x)
Tan	Тангенс	double	Math.Tan(double x)
Обернені тригонометричні функції			

ASin	Арксинус	double	Math.ASin(double x)
ACos	Арккосинус	double	Math.ACos(double x)
ATan	Арктангенс	double	Math.ATan(double x)
ATan2	Арктангенс	double	Math.ATan2(double x, double y) – кут, тангенс якого є результатом ділення y на x

Гіперболічні функції			
Tanh	Тангенс гіперболічний	double	Math.Tanh(double x)
Sinh	Синус гіперболічний	double	Math.Sinh(double x)
Cosh	Косинус гіперболічний	double	Math.Cosh(double x)

Експонента і логарифмічні функції			
Exp	Експонента	double	Math.Exp(x)
Log	Логарифм натуральний	double	Math.Log(x)
Log10	Логарифм десятковий	double	Math.Log10(x)

Abs	Модуль	Залежить від типу аргументу	Math.Abs(x)
Sqrt	Квадратний корінь	double	Math.Sqrt(x)
Sign	Знак числа	int	Math.Sign(x)
Ceiling	Заокруглення до більшого цілого	double	Math.Ceiling(double x)
Floor	Заокруглення до меншого цілого	double	Math.Floor(double x)
Round	Заокруглення	Залежить від типу аргументу	Math.Round(x)
Min	Мінімум двох чисел	Залежить від типу аргументу	Math.Min(x,y)
Max	Максимум двох чисел	Залежить від типу аргументу	Math.Max(x,y)
Pow	Піднесення до степеня	double	Math.Pow(x,y)
IEEERemainder	Остача від ділення	double	Math.IEEERemainder(double x, double y)
BigMul	Добуток	long	Math.BigMul(int x,int y)
DivRem	Ділення і остача	Залежить від типу аргументу	Math.DivRem(x,y, rem)

E	База натурального логарифма	double	Math.E
PI	Значення числа π	double	Math.PI

Логічним виразом або умовою називається такий вираз, внаслідок обчислення якого одержується логічне значення типу bool. Логічний вираз може містити операції відношення.

Операція	Позначення	Приклад
==	рівність	$x==y$
>	більше	$x>y$
<	менше	$x<y$
>=	більше або рівно	$x>=y$
<=	менше або рівно	$x<=y$
!=	не рівно	$x!=y$

У логічному виразі можуть також використовуватися логічні операції.

x	y	$x \& \& y$ (логічне «і»)	$x \parallel y$ (логічне «або»)	$!x$ (заперечення)
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

Для введення та виведення даних у консольних додатках використовуються методи класу Console.

Завдання для лабораторної роботи

Виконати завдання 1, 2 до даної лабораторної роботи. Номер варіанту обрати згідно номеру у журналі групи.

Завдання 1.

Варіант 10. Трикутник задається координатами своїх вершин на площині:

A(X1,Y1), B(X2,Y2), C(x3, y3). Скласти програму для знаходження периметра.

```
using System;
```

```
namespace ConsoleApp1
```

```
{
```

```
    class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            Console.WriteLine("Enter x1, y1, x2, y2, x3, y3: ");
```

```
            float x1 = float.Parse(Console.ReadLine());
```

```
            float y1 = float.Parse(Console.ReadLine());
```

```
            float x2 = float.Parse(Console.ReadLine());
```

```
            float y2 = float.Parse(Console.ReadLine());
```

```
            float x3 = float.Parse(Console.ReadLine());
```

```
            float y3 = float.Parse(Console.ReadLine());
```

```
            float perimeter = triangle_point_perimeter(x1, y1, x2, y2,
x3, y3);
```

```
            Console.WriteLine("Perimeter: " + perimeter);
```

```
        }
```

```
        public static float triangle_point_perimeter(float x1, float y1,
float x2, float y2, float x3, float y3)
```

```
        {
```

```
            float side1, side2, side3, semiperimeter, perimeter;
```

```
            side1 = MathF.Sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 -
y1));
```

```
            side2 = MathF.Sqrt((x3 - x2) * (x3 - x2) + (y3 - y2) * (y3 -
y2));
```

```
            side3 = MathF.Sqrt((x1 - x3) * (x1 - x3) + (y1 - y3) * (y1 -
y3));
```

```
            semiperimeter = (side1 + side2 + side3) / 2;
```

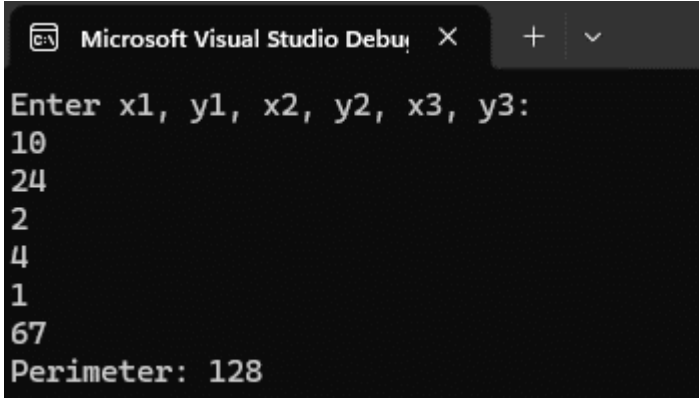
```
            perimeter = MathF.Sqrt(semiperimeter * (semiperimeter -
side1) * (semiperimeter - side2) * (semiperimeter - side3));
```

```
            return perimeter;
```

```
        }
```

```
    }
```


}



```
Microsoft Visual Studio Debug Console
Enter x1, y1, x2, y2, x3, y3:
10
24
2
4
1
67
Perimeter: 128
```

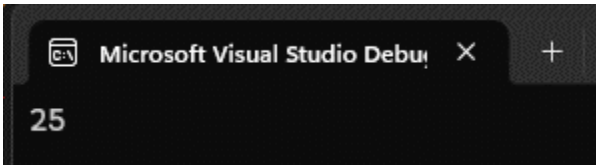
Рисунок 1.1 – Результат розрахунку периметра трикутника по точкам

Завдання 2

Варіант 10. Дано три дійсних числа: a, b, c. Знайти $(\min(a, b, c))^2$.

```
static double square_min_three_nums(int num1, int num2, int num3)
{
    int min = Math.Min(num1, Math.Min(num2, num3));
    return Math.Pow(min, 2);
}
```

```
Console.WriteLine(square_min_three_nums(10, 15, 5));
```



```
Microsoft Visual Studio Debug Console
25
```

Рисунок 1.2 – Результат знаходження квадрату мінімального числа з трьох

Питання для самоконтролю

1. Основні типи даних у мові C#: цілі числа (int, long), числа з рухомою комою (float, double), символи (char), логічні значення (bool), рядки (string).

2. Операції, що використовуються у мові C#: арифметичні (+, -, *, /), порівняння (==, !=, <, >), логічні (&&, ||, !), операції присвоєння (=), операції збільшення/зменшення (++, --), бітові операції (&, |, ^, <<, >>).

3. Загальний вигляд оператора розгалуження if:

```
if (умова)
```

```
{
```

```
}
```

```
else
```

```
{
```

```
}
```

4. Загальний вигляд оператора вибору switch:

```
switch (вираз)
```

```
{
```

```
    case значення1:
```

```
        break;
```

```
    case значення2:
```

```
        break;
```

```
    default:
```

```
        break;
```

5. Загальний вигляд оператора циклу while:

```
while (умова)
```

```
{
```

```
}
```

6. Загальний вигляд оператора циклу do-while:

```
do
```

```
{
```

```
    } while (умова);
```

7. Відмінності у роботі операторів циклу while та do-while: оператор while перевіряє умову перед входом у цикл, тоді як оператор do-while перевіряє умову після кожного виконання циклу, тобто код у do-while завжди виконається принаймні один раз.

8. Загальний вигляд оператора for:

```
for (ініціалізація; умова; інкремент)
```

```
{
```

```
}
```

9. Оператор for виконується за алгоритмом: спочатку виконується ініціалізація, потім перевіряється умова, якщо умова істинна, виконується тіло циклу, після чого виконується інкремент або декремент, і цей процес повторюється до тих пір, поки умова стає хибною.

10. Призначення оператора break - зупинка виконання циклу або виходу із комутатора (switch).

11. Призначення оператора continue - перехід до наступної ітерації циклу, обходячи решту коду у цьому циклі.