ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

# Основи C#

## Лекція 02

# План

1. Найпростіша програма
2. Типи даних мови C#
3. Оголошення змінних
4. Управляючи конструкції
5. Виведення / введення
6. Специфікатори форматування

# Найпростіша програма на C#

```
using System;   // .NET Framework Namespace
namespace ConsoleApp1  // Namespace Name
{
    class Program  //Class Name
    {
        static void Main(string[] args)   // Keyword;  Return Tape; Metod
        {
            Console.WriteLine("***** My First C# App *****");
            Console.WriteLine("Hello World!"); // Metod to Write the Text
to Console
            Console.WriteLine();
            Console.ReadLine();    // Metod to Read an Input from Console
        }
    }
}
```

# Найпростіша програма на C#

У C# є три типи коментарів:

1. Однорядковий (//);
2. Багаторядковий (/*...*/);
3. XML-документація

```
using System;

/// <summary>
/// .NET Framework Namespace
/// </summary>
namespace ConsoleApp1{
    /// <summary>
    /// Class Name
    /// </summary>
    class Program {
        /// <summary>
        /// Keyword;  Return Tape; Metod
        /// </summary>
        /// <param name="args">Command-line arguments.</param>
        static void Main(string[] args){
            /// <summary>
            /// Metod to Write the Text to Console
            /// </summary>
            Console.WriteLine("***** My First C# App *****");
            Console.WriteLine("Hello World!");

            /// <summary>
            /// Metod to Read an Input from Console
            /// </summary>
            Console.ReadLine();
        }
    }
}
```

# Типи даних мови C#

*Цілі*

| Тип даних | Тип .NET | Діапазон значень | Кількість байтів |
|---|---|---|---|
| byte | System.Byte | 0 … 255 | 1 |
| sbyte | System.SByte | -128 … 127 | 1 |
| short | System.Int16 | -32768 … 32767 | 2 |
| ushort | System.UInt16 | 0 … 65535 | 2 |
| int | System.Int32 | $-2*10^9$ … $2*10^9$ | 4 |
| uint | System.UInt32 | 0 … $4*10^9$ | 4 |
| long | System.Int64 | $-9*10^{18}$ … $9*10^{18}$ | 8 |

# Типи даних мови C#

## *Дробові*

| Тип даних | Тип .NET | Кількість знаків | Кількість байтів |
|---|---|---|---|
| float | System.Single | 7 | 4 |
| double | System.Double | 15-16 | 8 |
| decimal | System.Decimal | 28-29 | 16 |

## *Рядковий*

| Тип даних | Тип .NET |
|---|---|
| string | System.String |

# Типи даних мови C#

## *Символьний*

| Тип даних | Тип .NET | Кількість байтів |
|---|---|---|
| char | System.Char | 2 |

## *Логічний (булевий)*

| Тип даних | Тип .NET | Кількість байтів |
|---|---|---|
| bool | System.Boolean | 1 |

# Оголошення змінних

```
int age = 30;
int z = a + 5;
double pi = 3.14159;
bool test = true;
char firstLetter = 'A';
string name = "John Doe";
```

Тут використовуються вбудовані в мову C# скорочені назви типів даних (наприклад, *int, double, bool, char, string*)

```
System.Int64 r = z * 2;
System.String s = "Some string";
System.Double d = 1.5;
```

Цей спосіб явно вказує на повну назву типу даних (наприклад, *System.Int64*). Такий запис часто використовується для більшої ясності та при роботі з різними фреймворками або бібліотеками, де може бути кілька типів з однаковими назвами.

У C# є поняття неявної типізації (var), коли тип змінної визначається за значенням присвоєння.

# Дробові константи

Дробові константи по замовчуванню мають тип double;

Щоб задати константу типу float треба після числа дописати літеру F:
float somefloatValue = 1.5F;

Щоб задати константу типу decimal треба після числа дописати літеру M:
decimal someDecValue = 1.5M;

Для констант типу double можна вказувати літеру D, але необов’язково:
double someDblValue = 1.5D;

# Дробові константи

```
float floatValue = 3.14159F;
double doubleValue = 2.71828;
decimal decimalValue = 1.6180339887M;

Console.WriteLine("Значення типу float: " + floatValue);
Console.WriteLine("Значення типу double: " + doubleValue);
Console.WriteLine("Значення типу decimal: " + decimalValue);
// різниці в точності
decimal verySmallDecimal = 0.0000000000001M;
double verySmallDouble = 0.0000000000001;

Console.WriteLine("Дуже мале число типу decimal: " + verySmallDecimal);
Console.WriteLine("Дуже мале число типу double: " + verySmallDouble);
```

```
Значення типу float: 3,14159
Значення типу double: 2,71828
Значення типу decimal: 1,6180339887
Дуже мале число типу decimal: 0,0000000000001
Дуже мале число типу double: 1E-14
```

# Оголошення змінних

```
float a = 3.14F;
float b = 30.6f;
decimal c = 1005.8M;
decimal d = 334.8m;
uint a = 10U;
long b = 20L;
ulong c = 30UL;
```

Використання суфіксів

```
int a = 4;
System.Int32 b = 4;
```

Повна назва типу

```
var hello = "Hell to World";
var c = 20;
```

Неявна типізація

# Перетворення типів

**Розширюючі перетворення** – значення змінюють тип на тип з більшим діапазоном;

```
int x = 12;
long y = x * 100;
```

**Звужуючі перетворення** – значення змінюють тип на тип з меншим діапазоном.

```
long x = 12;
int y = x * 100;
```

**помилка компіляції**

# Перетворення типів

**Звужуючі перетворення** викликають помилку компіляції.

Щоб примусово виконати звужуюче перетворення типу, треба явно приводити тип:

```
long x = 12;
int y = (int)(x * 100);
```

При такому перетворенні не будуть виводитися помилки, навіть якщо виникне «переповнення»

# Перетворення типів

Щоб у випадку переповнення генерувалися помилки виконання:

```
long x = 100000000;
int y = checked((int)(x * 100));
```

При виконанні операцій над типом byte результат матиме тип int:

```
byte a = 1;
byte b = 2;
byte res = a + b;
```

**Помилка компіляції: результат має тип int**

# Операції мови C#

*Арифметичні операції:*

+= додати до змінної
-= відняти від змінної
*= помножити змінну
/= поділити змінну

= присвоєння
+ додавання
- віднімання
* множення
/ ділення (для цілих операндів – цілочисельне ділення)
% залишок від ділення (для цілих)

# Операції мови C#

*Префіксний та постфіксний інкремент та декремент*

++x префіксний інкремент
--x префіксний декремент
x-- постфіксний інкремент
x++ постфіксний декремент

```
int i = 1;
int j;
j = i++;      //після обчислень i = 2, а j=1

int i = 1;
int j;
j = ++i;     //після обчислень i = 2 і j=2
```

# Операції мови C#

*Порівняння:*

== дорівнює
!= не дорівнює
> більше
< менше
>= більше або дорівнює
<= менше або дорівнює

*Тернарна операція:*

умова ? вираз1 : вираз2

# Операції мови C#

*Логічні:*

&& логічне І (AND)
|| логічне Або (OR)
! логічне Ні (Not)

*Побітові:*

& побітове І
| побітове Або
^ побітове виключне Або (XOR)
~ побітове Ні
<< побітовий зсув вліво (SHL)
>> побітовий зсув вправо (SHR)

# Операції мови C#

**Відмінність від мови C:** логічні операції та операції порівняння повертають значення типу bool (а не цілі числа, як у мові C)

| *Мова C#* | *Мова C* |
|---|---|
| int x = 1;<br>if (x) { … } | int x = 1;<br>if (x) { … } |

*Помилка компіляції, потрібне значення булевого типу*

*У мові C код працює, оскільки істиною є будь-яке ненульове значення*

# Операції мови C#

| *Мова C#* | *Мова C* |
|---|---|
| bool x = true;<br>if (x) { … } | int x = 1;<br>if (x) { … } |

| *Вираз* | *Результат у мові C#* | *Результат у мові C* |
|---|---|---|
| 1 > 2 | false | 0 |
| 100 > 3 | true | 1 |
| (2 > 1) \|\| (3 > 2) | true | 1 |

# Оператори мови C#

Аналогічні мові C:
розгалуження: if, switch
цикли: for, while, do … while (break, continue)

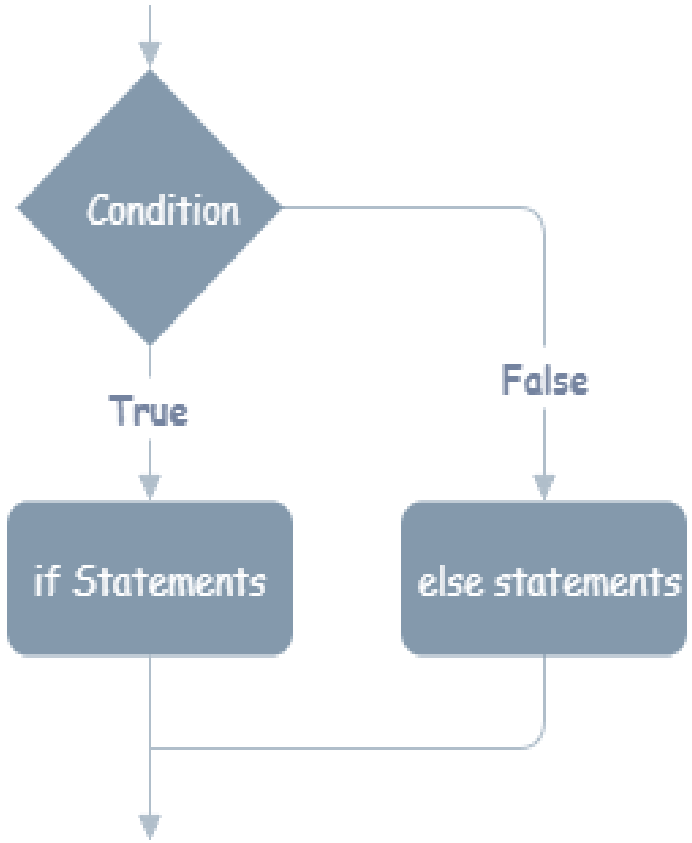
Відмінності:

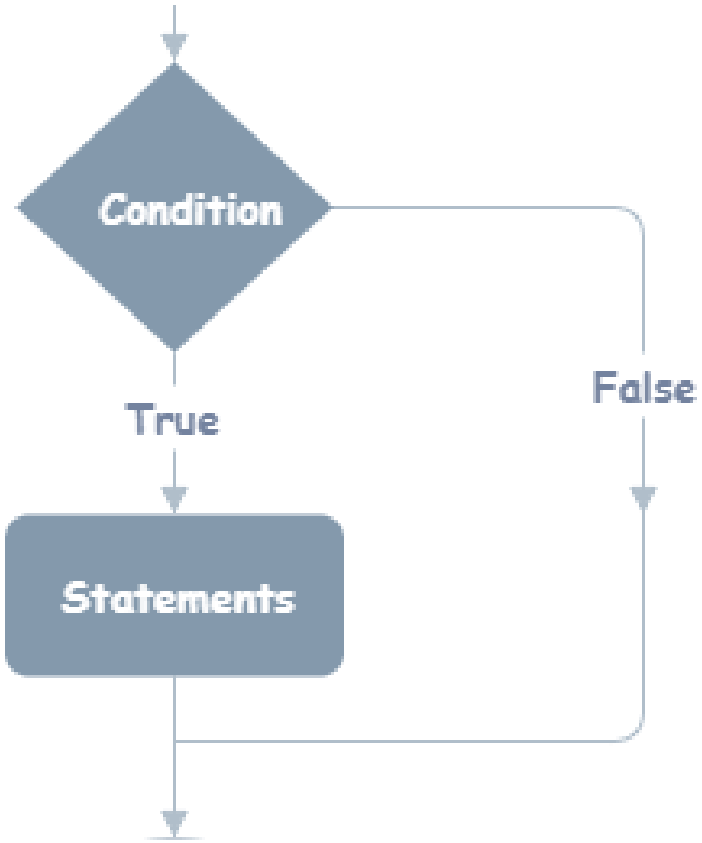
- умова – це значення типу bool
- switch підтримує стандартні типи, у тому числі і string
- у switch оператори break є обов’язковими

# Оператори мови C#

## Оператор if - else

```
if (умова) { команда 1; } [ else {команда 2;}]
```




Відмінності від мови C : *умова – це значення типу* ***bool***

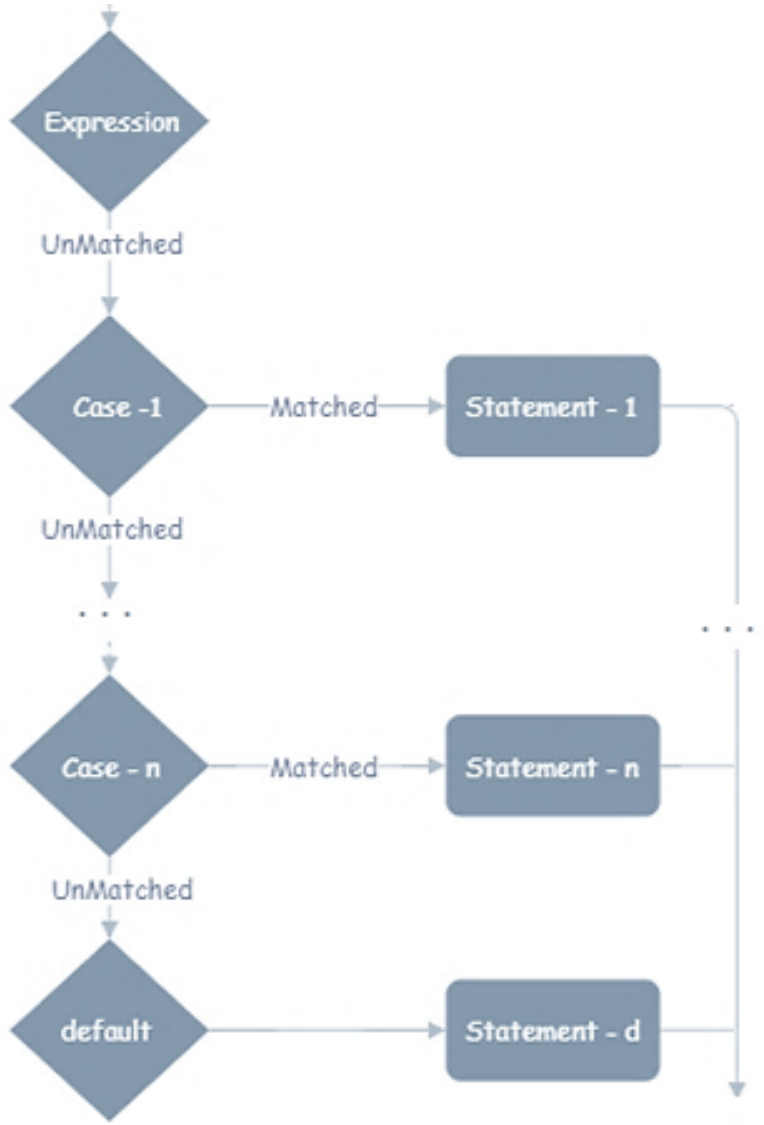
# Оператори мови C#

## Оператор switch

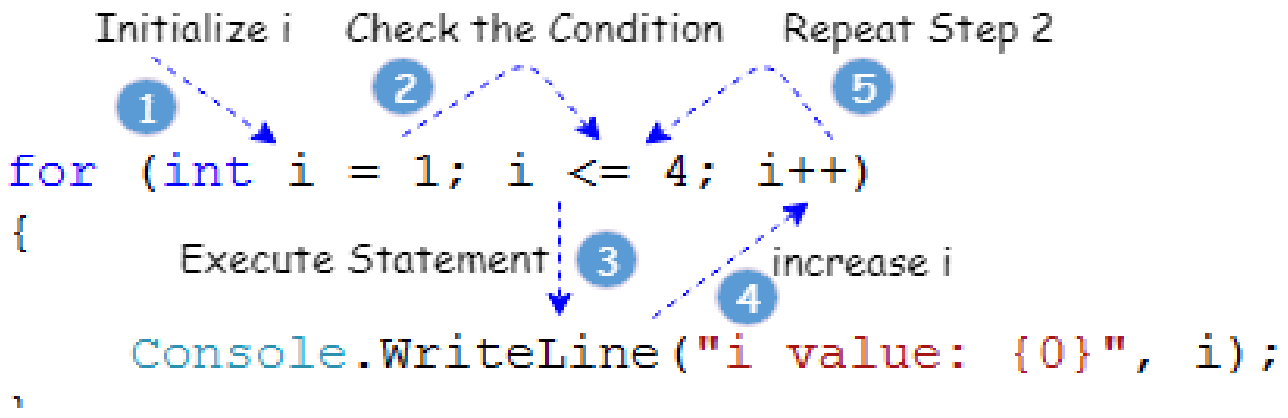
```
switch (< вираз >)
{
    case <ознака 1> : < команда 1 >;
break;
…
    case <ознака N> : < команда N >;
break;
    default: < команда N + 1 >;
break;
}
```

Відмінності від мови C :

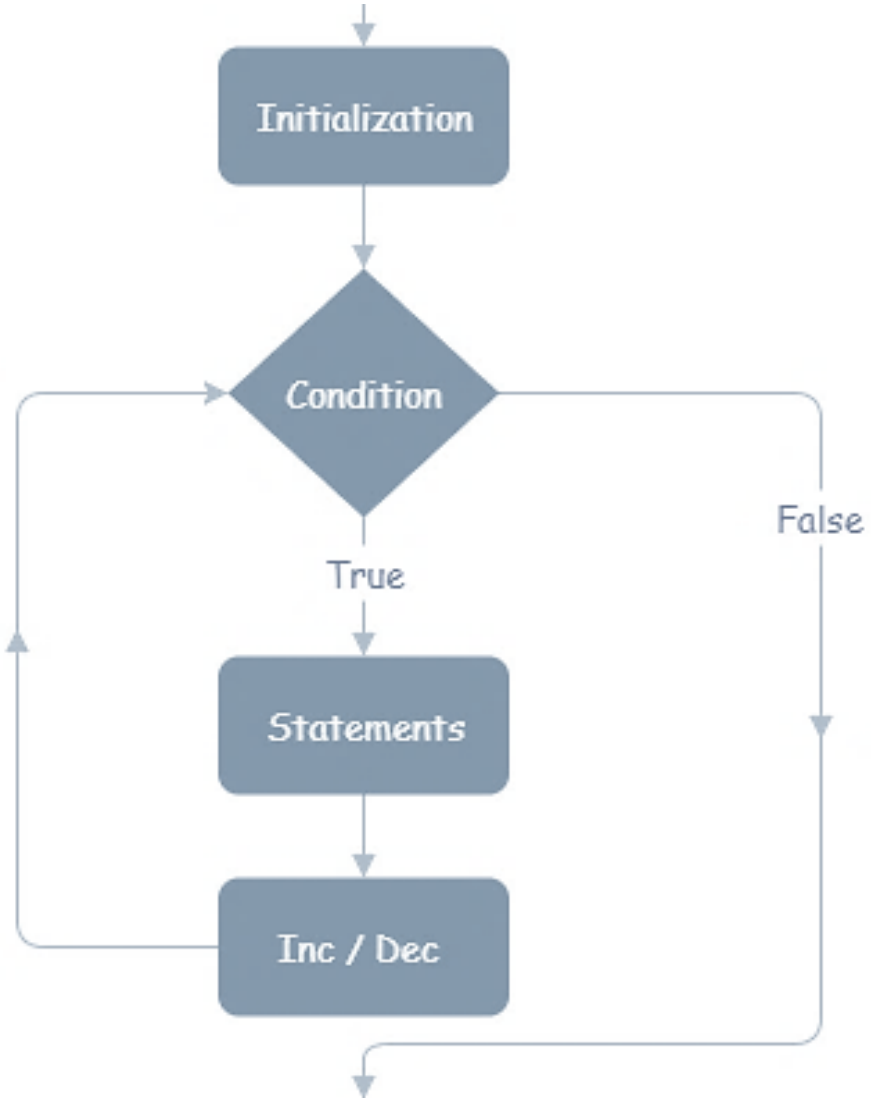
- switch підтримує стандартні типи, у тому числі і string
- у switch оператори break є обов’язковими



# Цикл for



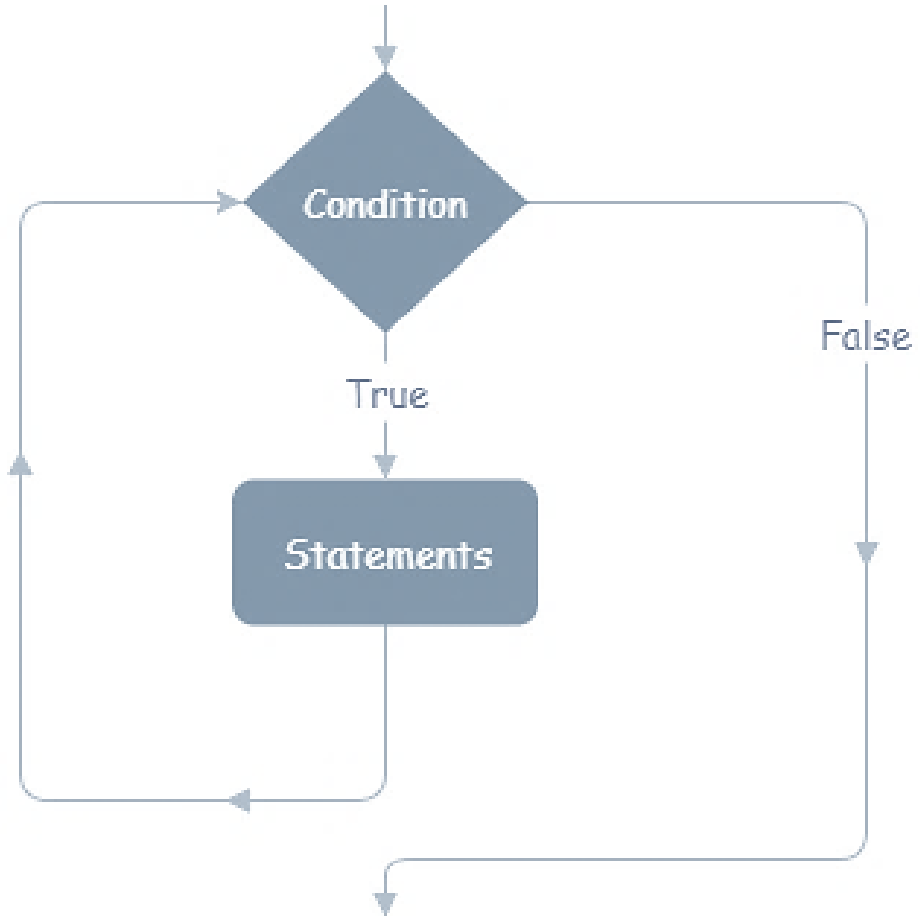

```
for (int i = 1; i <= 4; i++)
{
    Console.WriteLine("i value: {0}", i);
}
```



# Цикл while

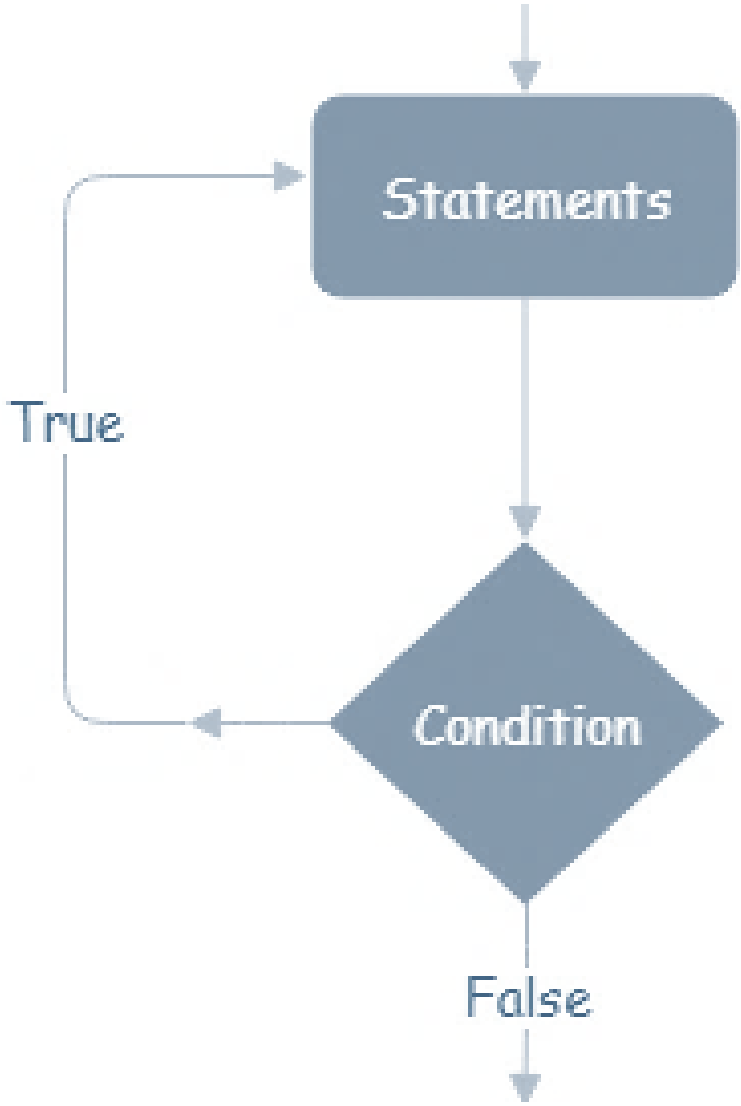
```
while (< умова >)
{
    < тіло циклу >
}
```

```
int i = 1;
while (i <= 4)
{
  Console.WriteLine("i value: {0}", i);
  i++;
 }
```



# Цикл do-while

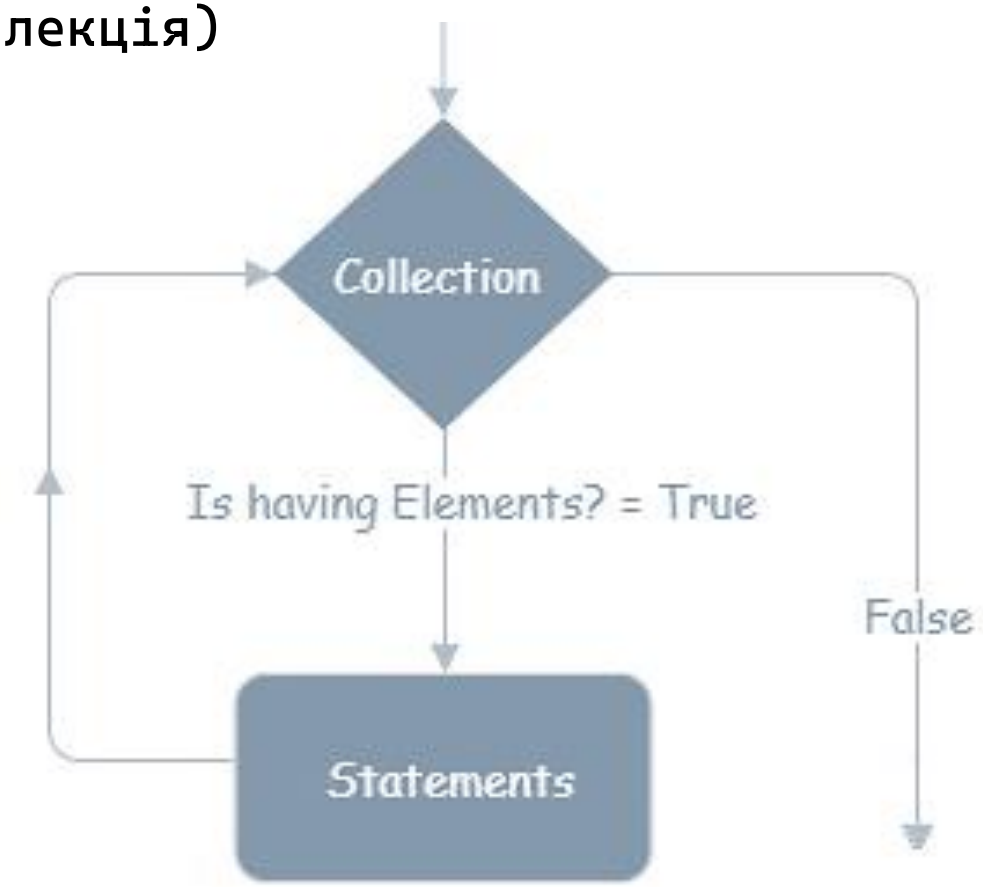
```
do
{
    < тіло циклу >
}
while (< умова >);
```




```
int i = 1;
do
{
 Console.WriteLine("i value: {0}", i);
 i++;
} while (i <= 4);
```

# Цикл foreach

```
foreach (тип_елементу змінна_елементу in колекція)
{
    // Тіло циклу
    // В тілі циклу можна працювати з
    //кожним елементом колекції
}
```




```
int[] numbers = { 1, 2, 3, 4, 5 };

foreach (int number in numbers)
{
    Console.WriteLine(number);
}
```

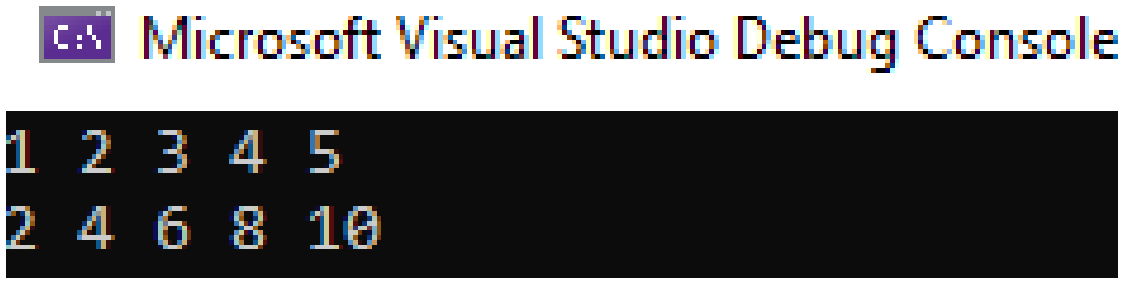
# Приклад

```
int[] numbers = { 1, 2, 3, 4, 5 };

foreach (int number in numbers)
{
    Console.Write(number + " ");
}
numbers = numbers.Select(x => x * 2).ToArray();

Console.Write("\n");

foreach (int number in numbers)
{
    Console.Write(number + " ");
}
```



# Виведення у консоль

Console.Write() – виводить і залишає курсор у тому ж рядку

Console.WriteLine() – виводить і переводить курсор на наступний рядок

**Синтаксис:**

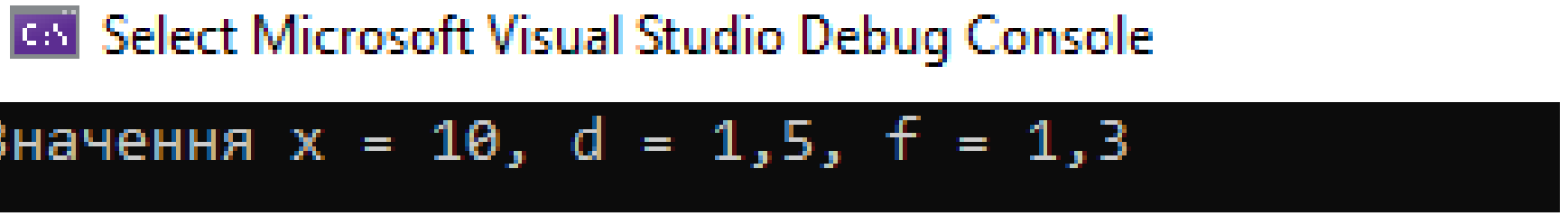
```
Console.Write("рядок формату"+зм1+зм2+...);
```

Рядок формату може містити специфікатори форматування

# Виведення у консоль

Специфікатори форматування задаються у фігурних дужках, де вказується номер змінної:

```
int x = 10;
double d = 1.5;
float f = 1.3f;
Console.WriteLine("Значення x = {0}, d = {1}, f = {2}", x, d, f);
```



# Виведення у консоль

Дробові числа можуть виводитись з роздільником «кома». Щоб встановити роздільник «крапка» використовуйте код:

```
static void Main(string[] args)
{
    System.Globalization.CultureInfo customCulture = (System.Globalization.CultureInfo)
                        System.Threading.Thread.CurrentThread.CurrentCulture.Clone();
    customCulture.NumberFormat.NumberDecimalSeparator = ".";
    System.Threading.Thread.CurrentThread.CurrentCulture = customCulture;

    int x = 10;
    double d = 1.5;
    float f = 1.3f;
    Console.WriteLine("Значення x = {0}, d = {1}, f = {2}", x, d, f);
}
```

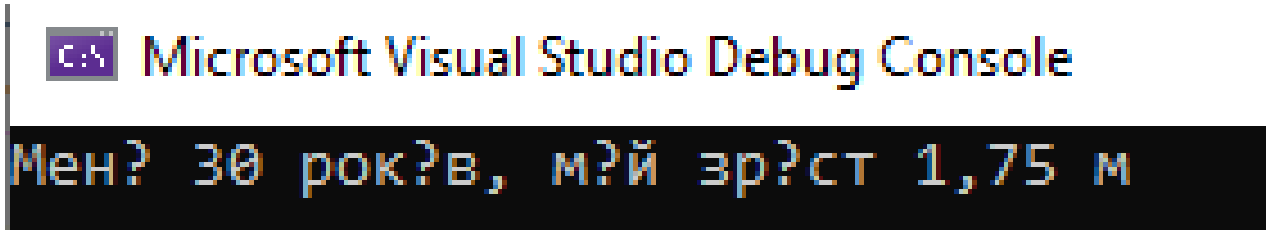
# Виведення у консоль

Для підтримки української абетки треба додати наступний код:

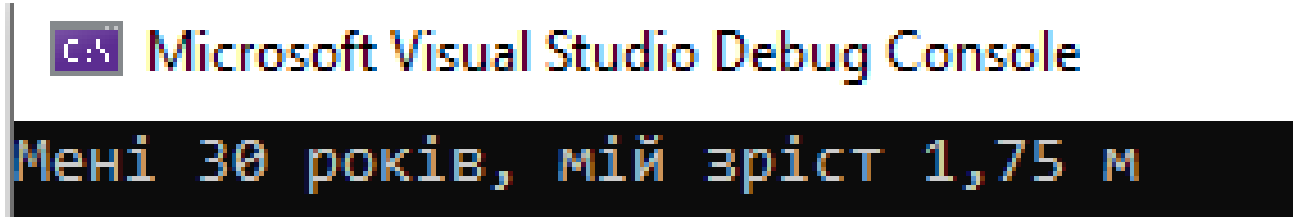
```
Console.OutputEncoding = Encoding.Unicode;
Console.InputEncoding = Encoding.Unicode;
```

# Приклад

```
int age = 30;
double height = 1.75;
Console.WriteLine("Мені {0} років, мій зріст {1} м", age, height);
```




```
Console.OutputEncoding = Encoding.Unicode;
Console.InputEncoding = Encoding.Unicode;
int age = 30;
double height = 1.75;
Console.WriteLine("Мені {0} років, мій зріст {1} м", age, height);
```



# Читання даних з консолі

Мова C# дозволяє читати лише **рядки** або **окремі символи**.
Неможливо прочитати з клавіатури значення інших типів (наприклад, цілих).

- Console.ReadKey() – читає один символ
- Console.ReadLine() – читає рядок

# Читання даних з консолі

Щоб працювати з цілими або дробовими числами виконують такі дії:

1. Читають з клавіатури рядок (тип string)

2. Виконують перетворення типу зі string до потрібного (наприклад, int) типу

# Читання даних з консолі

Перетворення типу можна виконати такими способами:

```
1. тип x = тип.Parse(Console.ReadLine());
2. тип x = Convert.ToТипDotNet(Console.ReadLine());
```

Приклад:

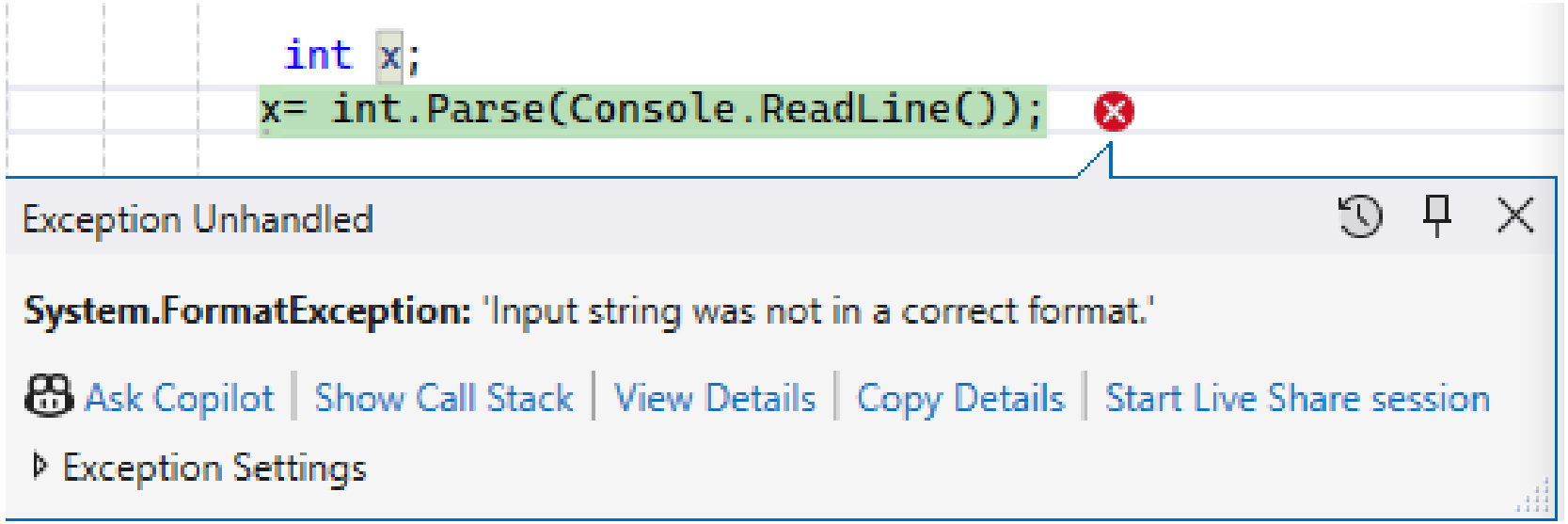
```
1. int x = int.Parse(Console.ReadLine());
2. int x = Convert.ToInt32(Console.ReadLine());
```

# Приклад

```
class Program
{
    static void Main(string[] args)
    {
      int x;
      Console.WriteLine("Введіть x = ");
      x = int.Parse(Console.ReadLine());
    }
}
```

# Читання даних з консолі

При введенні некоректних даних отримаємо помилку:




# Читання даних з консолі

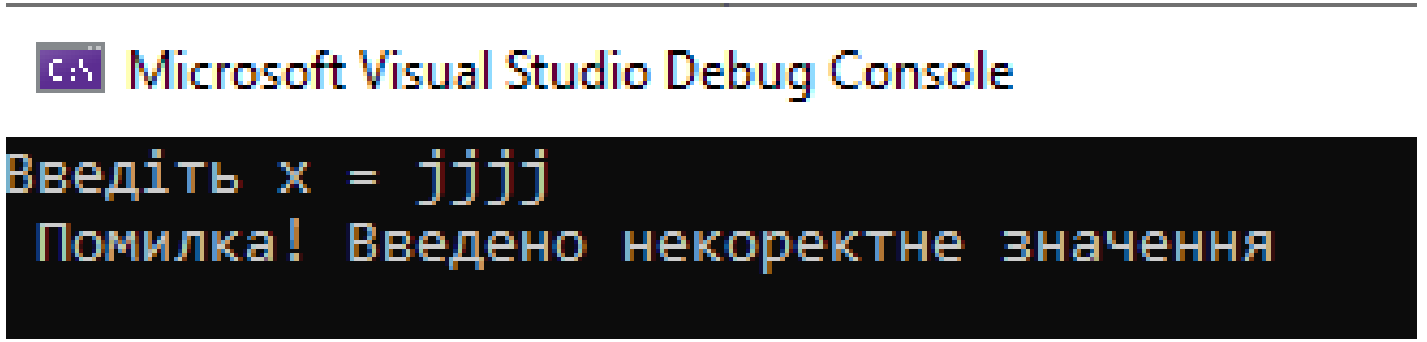
Щоб не отримувати помилки потрібно використовувати метод **TryParse**

```
int x;
Console.Write("Введіть x = ");
if (int.TryParse(Console.ReadLine(), out x))
{
    … x введено правильно …
}
```

# Читання даних з консолі

При введенні некоректних даних отримаємо помилку:

```
int x;
Console.Write("Введіть x = ");
if (int.TryParse(Console.ReadLine(), out x))
{
    Console.Write(" x введено правильно");
}
else
{
    Console.WriteLine(" Помилка! Введено некоректне значення");
}
```



# Математичні функції  Math.<ім'я функції>

| | | | |
|---|---|---|---|
| Max(x,y) | максимум з двох чисел | Acos(x) | |
| Min(x,y) | мінімум з двох чисел | Asin(x) | |
| Pi | число пі | Atan(x) | |
| Pow(x,y) | піднесення x до степеня  y | Ceiling(x) | округлення до більшого |
| Round(x) | округлення за звичайним правилом | Cos(x) | |
| Sign(x) | знак числа x  (+ / -  значення відповідно +1, -1, 0) | Cosh(x) | |
| Sin(x) | | E | число Ейлера |
| Sinh(x) | | Exp(x) | |
| Sqrt(x) | | Floor(x) | округлення до меншого |
| Tan(x) | | Log(x) | |
| Tanh(x) | | Log10(x) | |
| Abs(x) | | BigMul(int x, int y) | повертає добуток x * y у вигляді об'єкта long |

# Окремі математичні функції

DivRem(int a, int b, out int result) - повертає результат від ділення a / b, а залишок поміщається в параметр result

IEEERemainder(double a, double b) - повертає залишок від ділення a на b

Truncate(double value) - відкидає дробову частину числа value, повертає лише ціле значення

# Математичні функції Math.<ім'я функції>

Math.Abs Повертаємо абсолютне число. int i = Math.Abs(x);
Math.Acos АркКосинус. double i = Math.Acos(0.5);
Math.Asin АркСинус. double i = Math.Asin(0.5);
Math.Atan АркТангенс. double i = Math.Atan(0.5);
Math.Cos double x = Math.Cos(1.04);
Math.Exp Експонента. double x = Math.Exp(2);

Math.Log Обчислення логарифма.X - число яке потрібно знайти, Osn - підстава логарифма.
double x = Math.Log(X, Osn);

Math.Log10 Обчислення десяткового логарифма. double x = Math.Log10(10)
Math.Max Повертає з 2 чисел більше число. int x = Math.Max(10, 20);
Math.Min Повертає з 2 чисел менше число. int x = Math.Min(10, 20);
Math.PI Повертає число Пі. double pi = Math.PI;
Math.Pow Обчислює зведене до ступіню. double i = Math.Pow(a, x);
Math.Sin Повертає синус кута. double p = Math.Sin(0.5);
Math.Sqrt Повертає квадратний корінь. double r = Math.Sqrt(7);
Math.Tan Повертає тангенс кута. double p = Math.Tan(1.04);

# Приклад

```
double gradus = 30;
double radian = gradus * Math.PI / 180;
double x = 1;

double x2 = Math.Sqrt(3);

Console.WriteLine("Cos: " + Math.Cos(radian));
Console.WriteLine("Sin: " + Math.Sin(radian));
Console.WriteLine("Tan: " + Math.Tan(radian));
Console.WriteLine("cTan: " + 1 / Math.Tan(radian));
Console.WriteLine("ACos: " + Math.Acos(x) * 180 / Math.PI);
Console.WriteLine("ASin: " + Math.Asin(x) * 180 / Math.PI);
Console.WriteLine("ATan: " + Math.Atan(x2) * 180 / Math.PI);
Console.WriteLine("ACTan:" + 1/(Math.Atan(x2)*180/ Math.PI));
```

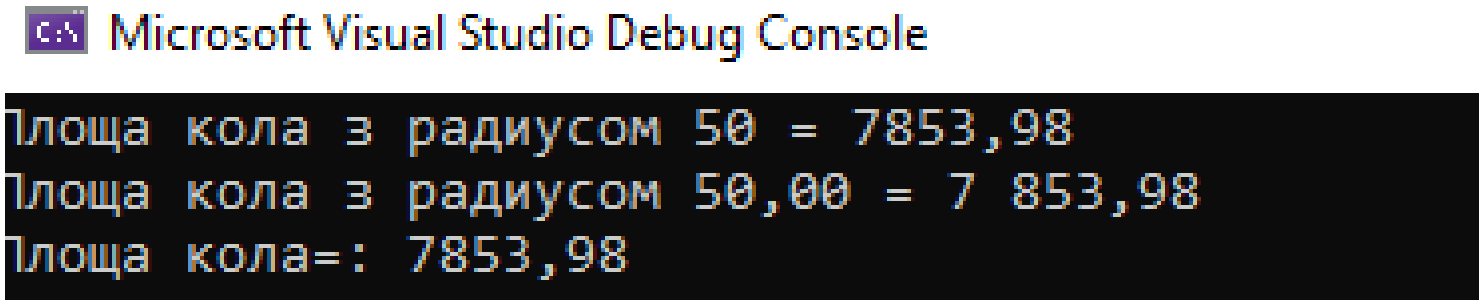
# Приклад

```
double radius = 50;
double area = Math.PI * Math.Pow(radius, 2);

Console.WriteLine($"Площа кола з радиусом {radius} = {Math.Round(area, 2)}");

Console.WriteLine($"Площа кола з радиусом {radius:F2} = {area:N2}");

Console.WriteLine("Площа кола=: " + Math.Round(area, 2));
```

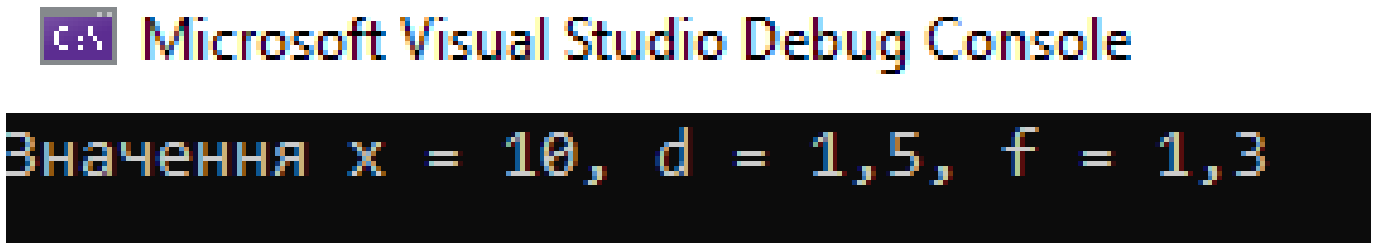


# Специфікатори форматування

Специфікатори форматування задаються у фігурних дужках, де вказується номер змінної:

```
int x = 10;
double d = 1.5;
float f = 1.3f;
Console.WriteLine("Значення x= {0}, d= {1}, f= {2}", x, d, f);
```

У результаті виконання цього рядка в консолі буде виведено наступний текст:



# Специфікатори форматування

Синтаксис:

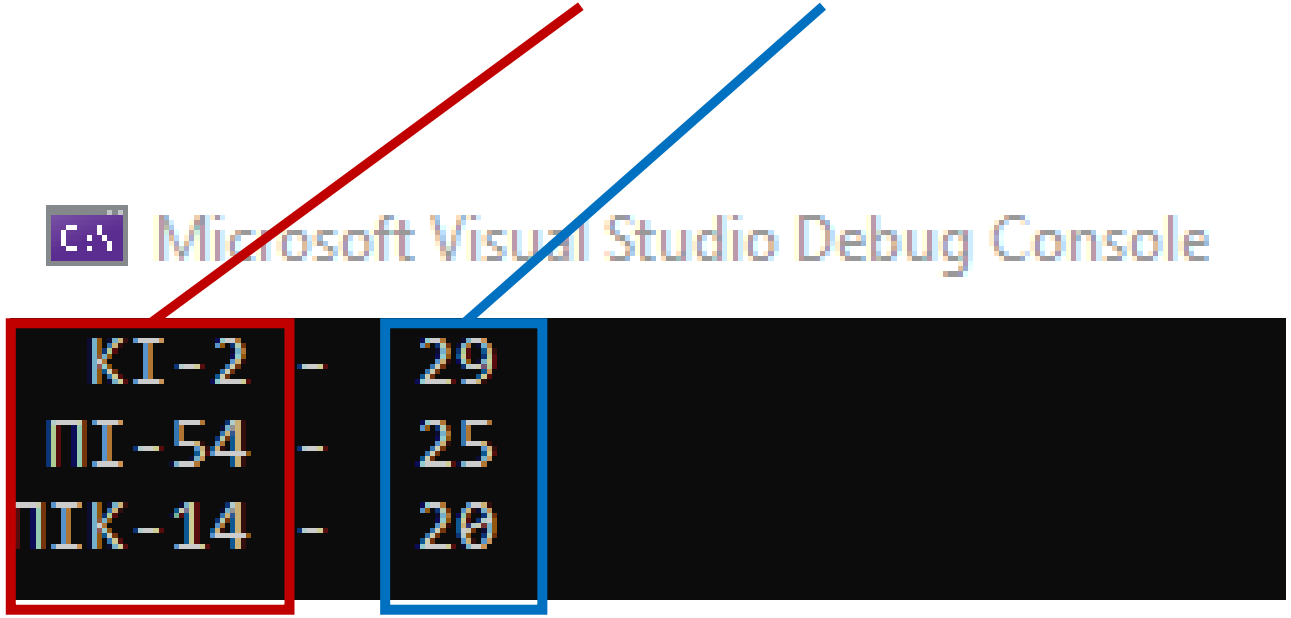
{номер[,розмірПоля][:формат]}

Квадратні дужки означають можливість відсутності відповідної компоненти (самі квадратні дужки не пишуться):

{номер,розмірПоля:формат}
{номер:формат}
{номер,розмірПоля}

# Специфікатори форматування

```
string group1 = "КІ-2",
       group2 = "ПІ-54",
       group3 = "ПІК-14";
int count1 = 29, count2 = 25, count3 = 20;
Console.WriteLine("{0,6} - {1,3}", group1, count1);
Console.WriteLine("{0,6} - {1,3}", group2, count2);
Console.WriteLine("{0,6} - {1,3}", group3, count3);
```




Якщо значення займає менше символів, ніж розмір поля, то додаються пробіли. Це дозволяє вертикально вирівнювати інформацію

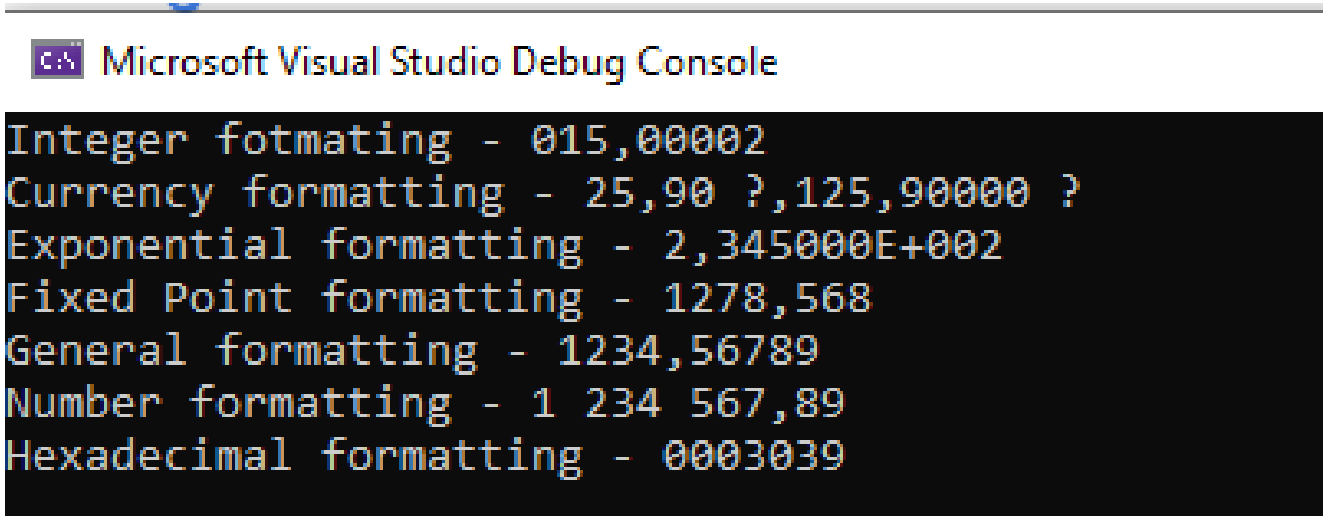
# Специфікатори форматування

| Літера для позначення формату | Що означає | Числове значення після літери |
|---|---|---|
| C | Відображення значення зі знаком валюти з використанням прийнятого символу | |
| D | Ціле число | Мінімальна кількість цифр |
| N | Відображення коми | |
| E | Експоненціальна форма дробового числа | Кількість дробових знаків |
| F | Звичайна форма дробового числа | |
| G | Найкоротший з форматів E або F | |
| P | Відсотковий формат | |
| X | Число  у шістнадцятковій системі | |

# Приклад

```
Console.WriteLine("Integer fotmating - {0:D3},{1:D5}",15,2);
Console.WriteLine("Currency formatting - {0:C},{1:C5}", 25.9, 125.9);
Console.WriteLine("Exponential formatting - {0:E}", 234.5);
Console.WriteLine("Fixed Point formatting - {0:F3}", 1278.56789);
Console.WriteLine("General formatting - {0:G}", 1234.56789);
Console.WriteLine("Number formatting - {0:N}", 1234567.89);
Console.WriteLine("Hexadecimal formatting - {0:X7}", 12345);
```

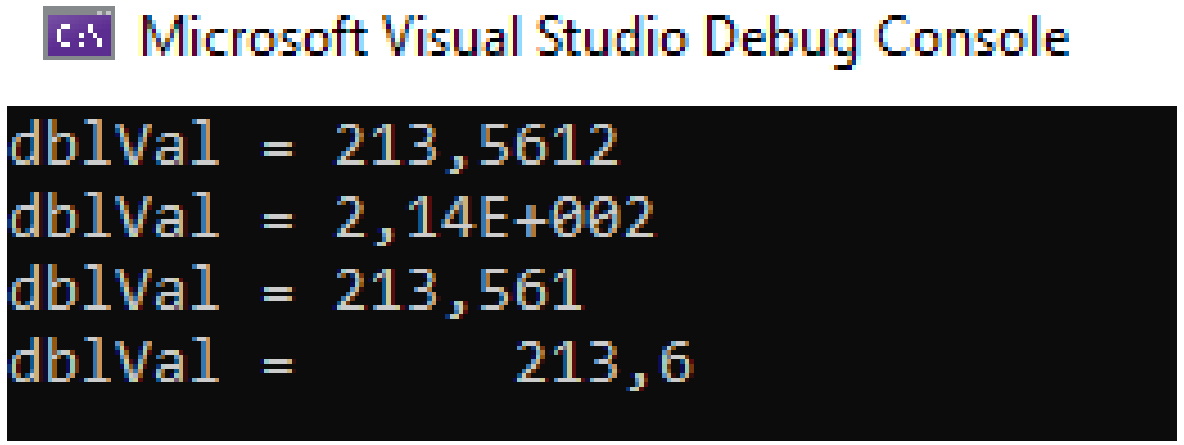
У результаті виконання цього рядка в консолі буде виведено наступний текст:



# Приклад

```
double dblVal = 213.5612;
Console.WriteLine("dblVal = {0}", dblVal);
Console.WriteLine("dblVal = {0:E2}", dblVal);
Console.WriteLine("dblVal = {0:F3}", dblVal);
Console.WriteLine("dblVal = {0,10:N1}", dblVal);
```

У результаті виконання цього рядка в консолі буде виведено наступний текст:



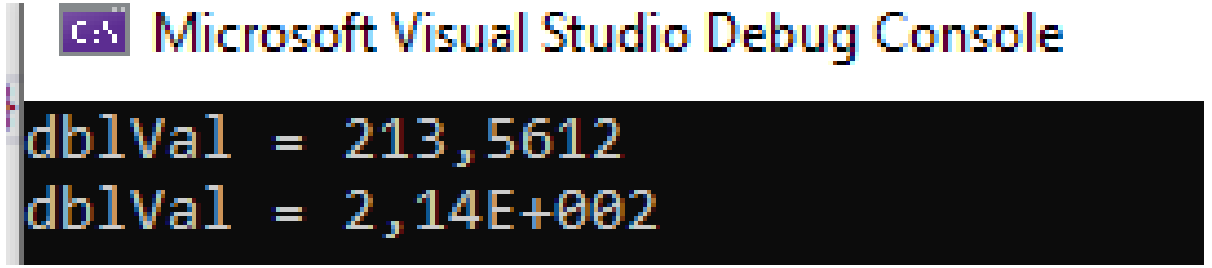
# Специфікатори форматування

Результат можна записати у новий рядок, замість того, щоб виводити на екран:

```
string newStr;
newStr = string.Format("рядок формату",зм1,зм2,...);
```

Приклад:

```
string resultString;
double dblVal = 213.5612;
resultString = string.Format("dblVal = {0}\ndblVal = {0:E2}\n", dblVal,
dblVal);
Console.WriteLine(resultString);
```



# Специфікатори форматування

Починаючи з Visual Studio 2015 додана можливість **інтерполяції** рядків. Вона дозволяє використовувати замість номера змінної - назви змінних та вирази.

Інтерполяція рядків – це зручний спосіб для вставки змінених та виразів у текстові рядки. Це робить код більш читабельним та спрощує конкатенацію рядка.

Перед початком інтерпольованого рядка поставити символ $.

# Приклад

```
Console.OutputEncoding = Encoding.Unicode;
Console.InputEncoding = Encoding.Unicode;
string name = "Іван";
int age = 30;
double pi = Math.PI;
// Простий приклад
Console.WriteLine($"Мене звати {name}, мені {age} років.");
// Форматування чисел
Console.WriteLine($"Значення числа Pi: {pi:F3}");
// Виклик методів
DateTime now = DateTime.Now;
Console.WriteLine($"Поточна дата і час: {now:dd.MM.yyyy HH:mm:ss}");
// Багаторядкові рядки
string message =
    $@"Це багаторядковий
    текст з інтерполяцією: {name}";
Console.WriteLine(message);
```