

# Оператори та інструкції мови C#

Клакович Л.М.

# Оператори C#

Category of Operator	Operators
Primary	(x), x.y, f(x), a[x], x++, x--, new, typeof, sizeof, checked, unchecked
Unary	+, -, !, ~, ++x, --x, (T)x
Multiplicative	*, /, %
Additive	+, -
Shift	<<, >>
Relational	<, >, <=, >=, is, as
Equality	==
Logical AND	&
Logical XOR	^
Logical OR	
Conditional AND	&&
Conditional OR	
Conditional	?:
Assignment	=, *=, /=, %=, +=, -=, <<=, >>=, &=, ^=,  =

# Арифметичні оператори

Name	Example
Identity (unary plus)	<code>+x</code>
Negation (unary minus)	<code>-x</code>
Post-increment	<code>x++</code>
Post-decrement	<code>x--</code>
Pre-increment	<code>++x</code>
Pre-decrement	<code>--x</code>
Multiplication	<code>x * y</code>
Division	<code>x / y</code>
Remainder	<code>x % y</code>
Addition	<code>x + y</code>
Subtraction	<code>x - y</code>

➤ Результат дії операторів залежить від типів операндів

```
x = 3 / 5 ;  
x = 3.0 / 5 ;  
x = 3.0 % 5;
```

```
0  
0.6  
error
```

➤ Pre і Post оператори

```
int a = 4 ;  
a ++ ;  
int b = a++;  
b = ++a;
```

```
a = 4  
a = 5  
a = 6 b = 5  
a = 7 b = 7
```

# Логічні оператори і оператори порівняння

Name	Example
Logical negation (also known as NOT)	<code>!x</code>
Conditional AND	<code>x &amp;&amp; y</code>
Conditional OR	<code>x    y</code>

```
bool rez;  
int x;  
// read value of x  
rez = ( x > 0 ) && ( ( x % 2 ) == 0 );  
  
rez = ( x > 0 ) || ( ( x % 2 ) == 0 );  
  
rez = ! ( x == 0 );
```

```
rez = ( x != 0 );
```

Name	Example
Less than	<code>x &lt; y</code>
Greater than	<code>x &gt; y</code>
Less than or equal	<code>x &lt;= y</code>
Greater than or equal	<code>x &gt;= y</code>
Equal	<code>x == y</code>
Not equal	<code>x != y</code>

# Умовний (тернарний) Operator (?:)

test ? expression1 : expression2

обчислитися якщо **test** є *true*

обчислитися якщо **test** є *false*

```
string answer = (a < 0) ? "negative" : "positive" ;
```

```
string answer = (a < 0) ? "negative" : (a > 0) ? "positive" : "zero" ;
```

# Оператори присвоєння

$x += 5$  еквівалентне до  $x = x + 5$

$x /= 2$  еквівалентне до  $x = x / 2$

Operator	Description
<b>+=</b>	<b>add / assign</b>
<b>-=</b>	<b>subtract / assign</b>
<b>*=</b>	<b>multiply / assign</b>
<b>/=</b>	<b>divide / assign</b>
<b>%=</b>	<b>remainder / assign</b>
<b>&lt;&lt;=</b>	<b>left shift / assign</b>
<b>&gt;&gt;=</b>	<b>right shift / assign</b>
<b>&amp;=</b>	<b>bitwise and / assign</b>
<b> =</b>	<b>bitwise or / assign</b>
<b>^=</b>	<b>bitwise xor / assign</b>

# Завдання 2. Оператори C#

Розв'язок задач записати без використання інструкцій, тільки оператори!

- Ввести два цілих числа **day** та **month** і перевірити чи можуть вони представляти день та місяць. Вивести true чи false
- Ввести дійсне число `number` і отримати 2 перші цифри після коми цього числа. Вивести суму цих цифр. Напр.: 3.456->4+5=9
- Ввести ціле число `h` , яке представляє час доби (годину).  
В залежності від часу доби вивести привітання  
(«Доброго ранку!», «Доброго дня!», «Доброго вечора!», «Доброї ночі!»)

# Інструкції C#

## Лінійні алгоритми

- Визначення (оголошення)
- `expr;` інструкція-вираз
- `{ }` складена інструкція
- `;` пуста інструкція

## Галужені алгоритми

- `if` умовна інструкція
- `switch` перемикач
- `continue` інструкція продовження
- `break` інструкція виходу
- `return` інструкція повернення

## Циклічні алгоритми

- `for` , `foreach` покроковий цикл
- `while` цикл з передумовою
- `do-while` цикл з післяумовою



# Умовна інструкція **if**

```
if (condition)
{
    then-statement;
}
```

```
if (condition)
{
    then-statement;
}
else
{
    else-statement;
}
```

**Особливості:** Вираз *condition* повинен бути тільки булівського типу

```
a=3;
if (a < 5)
{
    b += 27;
}
//error
if (a = 5)
{...}

//error
if (a)
{...}
```

# Перемикач **switch**

```
switch (expression)
{
    case const-expr1 : statement(s)1; break;
    case const-expr2 : statement(s)2; break;
    default : statement(s); break;
}
```

## Особливості:

- **expression** в операторі **switch** може бути не тільки цілочисловим виразом, а й рядком (string).
- Кожен вибір **case** повинен закінчуватися **break** чи **goto**.

# Перемикач **switch**

```
Console.WriteLine ( "Do you enjoy C# ? (yes/no/maybe)" );  
string input = Console.ReadLine();  
  
switch ( input.ToLower ( ) )  
{  
    case "yes":  
    case "maybe":  
        Console.WriteLine ( " Great!" );  
        break;  
    case "no":  
        Console.WriteLine ( "Too bad!" );  
        break;  
}
```

# Цикли while та do while

- в циклах **while** та **do while** умовний вираз повинен бути булівського типу

```
while ( condition )  
{  
    statements;  
}
```

```
do  
{  
    statement(s);  
}  
while( condition );
```

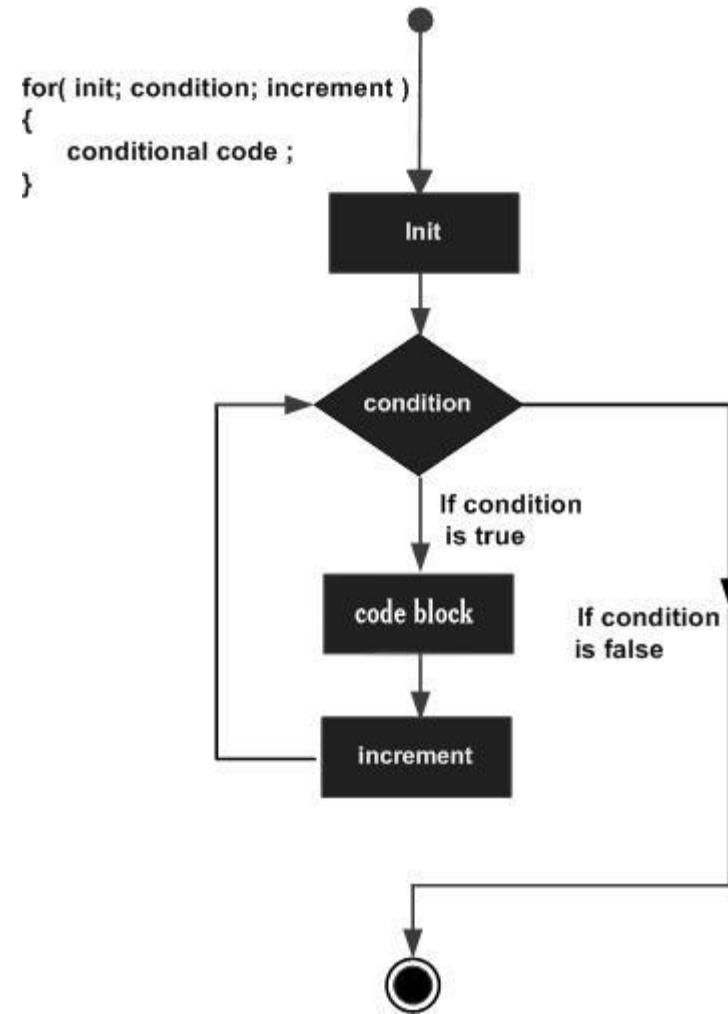
```
int number = 0;  
while ( number < 5 )  
{  
    Console.WriteLine(number);  
    number = number + 1;  
}
```

```
int number = 0;  
do  
{  
    Console.WriteLine(number);  
    number = number + 1;  
}  
while ( number < 5);
```

# Цикл for

```
for ( init; condition; increment )  
{  
    statement(s);  
}
```

```
for (int a = 10; a < 20; a++)  
{  
    if (a==15) continue;  
    Console.WriteLine( " value of a: {0}", a);  
}
```



# Масиви

- Масиви забезпечують ефективне зберігання послідовності елементів
- Знайомий синтаксис інших мов програмування
- Незмінний розмір масиву – підчас оголошення
- Масиви в С# - це тип, похідний від класу **System.Array**
  - утворюються оператором **new**
  - необхідно вказати тип елементів та їх кількість
  - доступ до елементів через оператор **[ ]**
  - кількість елементів в масиві визначається властивістю **Length**
  - **IndexOutOfRangeException** генерується при невірному індексі

створення →

```
int[] a = new int[5];  
int [,] myMatrix=new int [6,8];
```

Доступ до елементів →

```
a[0] = 17;  
a[1] = 32;  
int x = a[1];  
a[5] = 32; //IndexOutOfRangeException
```

Кількість елементів →

```
int l = a.Length;
```

# Масиви

- Елементом масиву автоматично присвоюється значення за замовчуванням
  - 0 для цілих типів
  - **false** для **bool**
  - '**\x0000**' для **char**
  - **null** для посилань
- Можлива ініціалізація елементів при оголошенні масиву

Всі елементи **false** → `bool[] a = new bool[10];`

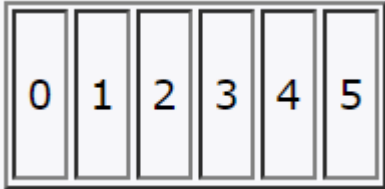
Всі елементи 0 → `int[] b = new int[5];`

ініціалізація → `int[] c = new int[5] { 48, 2, 55, 17, 7 };`

`int [] ages={5,6,8,9,2,0};`

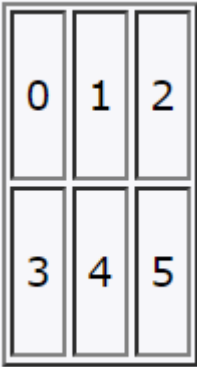
# Багатовимірні масиви

```
int[] nums1 = new int[] { 0, 1, 2, 3, 4, 5 };
```



0	1	2	3	4	5
---	---	---	---	---	---

```
int[,] nums2 = { { 0, 1, 2 }, { 3, 4, 5 } };
```



0	1	2
3	4	5

```
int[,,] nums3 = new int[2, 3, 4];
```



# Цикл foreach

- цикл `foreach` повторює інструкції для кожного елемента в масиві або для кожного об'єкта колекції .
- Елементи масиву стають доступними тільки для читання (не можливо їх змінювати).

```
int[] data = new int[5] { 48, 2, 55, 17, 7 };  
int sum = 0;  
foreach (int x in data)  
{  
    sum += x;  
    //x=x+4;  
}
```

Diagram illustrating the components of the `foreach` loop:

- `type` points to `int`
- `value` points to `x`
- `collection` points to `data`

```
string str="dhfhsdfdfhgsdj";  
  
foreach (char ch in str)  
    Console.WriteLine(ch+"-");  
  
for(int i=0;i<str.Count;i++)  
{  
    if(str[i]=='d')  
        Console.WriteLine(str[i]+"*");  
}
```

# Завдання 3. Інструкції

- 1) Ввести рядок символів. Видрукувати кожен другий символ.
- 2) Введіть послідовність додатніх цілих чисел (до першого від'ємного).  
Обчисліть середнє арифметичне значення введених чисел.
- 3) Введіть номер місяця року і виведіть скільки в цьому місяці днів (вважати рік не високосним)