ТЕМА 4: ОПЕРАТОРИ. СТРУКТУРИ. ПЕРЕРАХУВАННЯ

План

Функції складаються з операторів, що виконуються один за одним. Блок операторів – це послідовність операторів, що знаходяться в фігурних дужках {}.

4.1. Оператори оголошення.

```
char a = '5';
const int b= 5;
b++; //помилка на етапі компіляції
```

Локальні змінні. Областю видимості локальної змінної або локальної константи є поточний блок {} та всі вкладені блоки.

```
static void Main()
{
      int x;
      {
            int y;
            int x; //помилка
      }
      {
            int y; //помилки немає – y не знаходиться в області видимості
      }
      Console.Write(y); //помилка - y знаходиться поза областю видимості
}
```

4.2. Оператори-вирази.

Являють собою вирази. Вони мають або змінити стан, або викликати щось, що може змінити стан. Наприклад:

- вирази присвоєння;
- вирази виклику методів;
- вирази створення екземплярів об’єктів.

4.3. Оператори вибору.

Для умовного керування потоком виконання програми. В C# існує декілька операторів, що дозволяють змінювати потік команд:

- оператори розгалуження і вибору (`if`, `switch`);

- умовна операція (? :);
- оператори циклу (while, do..while, for, foreach).

Оператор розгалуження – це оператор, що виконує команду або групу команд в залежності від певної умови. Умова в умовному операторі подається у вигляді виразу логічного типу. Має скорочену та повну форми запису.

```
if (x>y)
    maxx=x;
else
    maxx=y;
```

Приклади логічних виразів та груп операторів.

Вкладені умовні оператори.

Оператор вибору – це оператор, що виконує одну команду із заданого набору в залежності від значення виразу.

```
int x, y;
x=5;y=2;
     switch (x)
     {
          case 1:
          case 2:
          case 3:
                    Console.WriteLine("від 1 до 3");
                    break;
          case 5: Console.WriteLine(555);
                    break;
          default: Console.WriteLine(123);
                    break;
     }
```

В операторі switch можна використовувати лише такі значення, які можуть бути статично обчисленими (цілочисельний, bool, enum, string). Якщо один і той самий код має виконуватися для декількох значень, застосовується послідовний запис конструкції case.

4.4. Оператори ітерацій.

Цикл while виконується до тих пір, поки умова істина.

```
int x=0;
     while (x<3)
     {
          Console.WriteLine(++x);
     }    //123
```

Цикл do..while відрізняється від циклу while лише тим, що умова перевіряється лише після виконання блоку операторів.

```
int x=0;
do
     {
```

```
            Console.Write(x);
            x++;
      }
while (x<3);  //012
```

Цикл `for` включає спеціальні конструкції (не обов'язкові) для ініціалізації та ітерації змінної циклу:

`for` (*конструкція ініціалізації*; *конструкція умови*; *конструкція ітерації*) *оператор або блок операторів*;

Наприклад:

```
int x,y;
for (x=0,y=0;x<3;x++,y++)
     Console.Write(x+y); //024
```

Вкладені цикли.

Оператор `foreach` забезпечує проходження за всіма елементами контейнера (масив, рядок, колекція):

```
foreach (char c in "example")
           Console.Write(c+" "); //e x a m p l e
```

4.5. Оператори переходу

в C# є `break`, `continue`, `goto`, `return`, `throw`.

Оператор `break` припиняє виконання тіла ітерації або оператора `switch`.

Оператор `continue` пропускає оператори, що залишились після цього оператора в тілі циклу, і розпочинає наступну ітерацію.

Звужуючі та розширюючі перетворення типів.

4.6. Тип Enum (перерахування).

Символічні імена, що відображають числові значення.

```
enum EmpType
{
Manager,           //   =    0
Grunt,             //   =    1
Contractor,        //   =    2
VicePresident //   =    3
}
```

За замовчуванням першому елементу присвоюється значення 0, всім наступним *n*+1. Нумерувати елементи перерахування можна як завгодно:

```
// Начать нумерацию со значения 102.
enum EmpType
{
Manager = 102,
Grunt,             // =103
Contractor,        // =104
VicePresident // =105
```

```
}
// Значення елементів у перерахуванні не обов’язково мають бути
послідовними!
enum EmpType
{
Manager = 10,
Grunt = 1,
Contractor = 100,
VicePresident = 9
}
```

Перерахування, для економії пам’яті, може базуватись на іншому, ніж `int`, типі: (`byte`, `short` або `long`):

```
// На.этот раз EmpType отображается на тип byte.
enum EmpType : byte
{
Manager = 10,
Grunt = 1,
Contractor = 100,
VicePresident = 9
}
```

Оголощення змінної

```
EmpType emp = EmpType.Contractor;
```

Метод `ToString()` повертає назву значення. Для того, щоб отримати числове значення, що стоїть в основі перерахування, потрібно перетворити тип:

```
// Выводит строку "Contractor = 100".
Console.WriteLine("{0} = {1}", emp.ToString(), (byte)emp);
```

Метод `GetValues()` повертає масив, кожен елемент якого члену перерахування:

```
System.Enum e;
Array enumData = Enum.GetValues(e.GetType());
for(int i = 0; i < enumData.Length; i++)
{
Console.WriteLine("Name: {0}, Value: {0:D}",
enumData.GetValue(i));
}
```

4.7. Структури.

В C# структури визначаються за допомогою ключового слова `struct`:

```
struct Point
{
// Поля структуры,
public int X;
public int Y;
// Добавить 1 к позиции (X, Y)
public void Increment()
{
```

```
X++; Y++;
}
// Вычесть 1 к позиции (X, Y)
public void Decrement()
{
X--; Y--;
}
// Отобразить текущую позицию,
public void Display()
{
Console.WriteLine("X = {0}, Y = {1}", X,Y);
}
}
```

Звертання до елементів структури можливе через «.».

```
// Создать начальный экземпляр Point.
Point myPoint;
myPoint.X = 349;
myPoint.Y = 76;
myPoint.Display();
// Скорректировать значения X и Y.
myPoint.Increment();
myPoint.Display();
Console.ReadLine();
```

Перед використанням структури всім відкритим полям потрібно присвоїти значення:

```
/ Ошибка! Полю Y не присвоено значение.
Point pl;
pl.X = 10;
pl.Display ();
// Все в порядке! Обоим полям присвоены значения перед
использованием.
Point p2;
p2.X = 10;
p2.Y = 10;
p2.Display ();
```

Інший варіант ініціалізації полів структури стандартними значеннями:

```
// Установить все поля в стандартные значения, используя
стандартный конструктор.
Point pl = new Point() ;
pl.Display();          // Выводит X=0, Y=0
```