



Управляющие структуры


Лекция №6



Управляющая структура выбора

В языке C# предусмотрены четыре структуры выбора.

- Структура единственного выбора (**if**)
- Структура двойного выбора (**if...else**)
- Структура множественного выбора (**switch**)
- Встроенный условный оператор **?:**



Структура единственного и двойного выбора

- **Оператор if** определяет, какой блок операторов будет выполняться при выполнении условия, заданного выражением **Boolean**.


Структура единственного и двойного выбора

```
string myStr;
Console.WriteLine("Введите строку: ");

myStr = Console.ReadLine();

if (myStr.Length < 5)
    Console.WriteLine("\nВ данной строке меньше 5 символов");
else if ((myStr.Length >= 5) && (myStr.Length <= 12))
    Console.WriteLine("\nВ данной строке {0} символов", myStr.Length);
else Console.WriteLine("\nВ данной строке больше 12 символов");

Console.ReadLine();
```

Структура множественного выбора (**switch**)

- **Оператор switch** — это оператор управления, выбирающий из списка возможных вариантов *раздел переключения*, для выполнения содержащегося в нём кода.

Структура множественного выбора (**switch**)

```
int caseSwitch = 1;
switch (caseSwitch)
{
    case 1:
        Console.WriteLine("Case 1");
        break; // обязательно
    case 2:
        Console.WriteLine("Case 2");
        break;
    default:
        Console.WriteLine("Default case");
        break;}
}
```

Структура множественного выбора (switch)

```
Console.WriteLine("Введите язык (C#, VB или C++)");
string myLanguage = Console.ReadLine(); static void sw1(string s)
{
    switch (s)
    {
        case "C#":
            Console.WriteLine("Вы выбрали язык C#");
            break;
        case "VB":
            Console.WriteLine("Вы выбрали язык Visual Basic");
            break;
        case "C++":
            Console.WriteLine("Вы выбрали язык C++");
            break;
        default:
            Console.WriteLine("Такой язык я не знаю");
            break;
    }
}
sw1(myLanguage);
Console.ReadLine();
```

Встроенный условный оператор

Условный оператор (?:) возвращает одно из двух значений в зависимости от значения логического выражения. Для условного оператора используется следующий синтаксис.

`condition ? first_expression : second_expression;`

Параметр *condition* должен иметь значение **true** или **false**. Если параметр *condition* имеет значение **true**, вычисляется выражение *first_expression* и итог этого вычисления становится результатом. Если параметр *condition* имеет значение **false**, вычисляется выражение *second_expression* и итог этого вычисления становится результатом.


Встроенный условный оператор

```
int input = Convert.ToInt32(Console.ReadLine());  
// if-else construction. if (input > 0) classify =  
    "positive"; else classify = "negative";  
// ?: conditional operator.  
classify = (input > 0) ? "positive" : "negative";
```



Управляющие структуры повторения

- Выражение проверяется в начале цикла (**while**)
- Выражение проверяется в конце цикла (**do...while**)
- Действие выполняется над свойствами объекта или элементами массива (**foreach...in**)
- Повторение, управляемое счетчиком (**for**)



Выражение проверяется в начале цикла (**while**)

- **Оператор while** выполняет оператор или блок операторов, пока определенное выражение не примет значение **false**.

Выражение проверяется в начале цикла (**while**)

```
// Пример возведения числа в несколько степеней  
byte l = 2, i = 0;  
int result = 1;  
  
while (i < 10)  
{  
    i++;  
    result *= l;  
    Console.WriteLine("{0} в степени {1} равно {2}", l, i, result);  
}  
  
Console.ReadLine();
```


Выражение проверяется в конце цикла (**do...while**)


- **Оператор do-while** повторно выполняет оператор или блок операторов, пока определенное выражение не примет значение **false**. Тело цикла должен быть заключен в фигурные скобки, **{}**, если он не состоит из одной инструкции. В этом случае фигурные скобки необязательны.

Выражение проверяется в конце цикла (**do...while**)

```
public class TestDoWhile
{
    public static void Main ()
    { int x = 0;
      do
      {
          Console.WriteLine(x);
          x++;
      } while (x < 5);}}
```


Повторение, управляемое счетчиком (**for**)

- Цикл `for` в C# предоставляет механизм итерации, в котором определенное условие проверяется перед выполнением каждой итерации.
- *for* (инициализатор; условие; итератор) оператор (операторы)



Повторение, управляемое счетчиком (**for**)

- **Инициализатор** это выражение, вычисляемое перед первым выполнением тела цикла (обычно инициализация локальной переменной в качестве счетчика цикла).



Повторение, управляемое счетчиком (**for**)

- **Условие** это выражение, проверяемое перед каждой новой итерацией цикла (должно возвращать true, чтобы была выполнена следующая итерация);
- **Итератор** выражение, вычисляемое после каждой итерации (обычно приращение значения счетчика цикла).

Повторение, управляемое счетчиком (**for**)

```
static void Main()
{
    int i;
    int j = 10;
    for (i = 0, Console.WriteLine("Start: {0}",i); i < j;
        i++, j--, Console.WriteLine("i={0}, j={1}", i, j))
    {
        // Body of the loop.
    }
}
```



Повторение `foreach ... in`

- **Цикл `foreach`** служит для циклического обращения к элементам коллекции, представляющей собой группу объектов. В C# определено несколько видов коллекций, каждая из которых является массивом.



Повторение foreach ... in

- Формально для того, чтобы нечто можно было рассматривать как коллекцию, это нечто должно поддерживать интерфейс IEnumerable. Примерами коллекций могут служить массивы C#, классы коллекций из пространства имен System.Collection, а также пользовательские классы коллекций.



Повторение foreach ... in

- Формально для того, чтобы нечто можно было рассматривать как коллекцию, это нечто должно поддерживать интерфейс IEnumerable. Примерами коллекций могут служить массивы C#, классы коллекций из пространства имен System.Collection, а также пользовательские классы коллекций.

Повторение foreach ... in

```
// Объявляем два массива
int[] myArr = new int[5];
int[,] myTwoArr = new int[5, 6];
int sum = 0;

Random ran = new Random();

// Инициализируем массивы
for (int i = 1; i <= 5; i++)
{
    myArr[i-1] = ran.Next(1, 20);
    for (int j = 1; j <= 6; j++)
        myTwoArr[i - 1, j - 1] = ran.Next(1, 30);
}

// Вычисляем квадрат каждого элемента одномерного массива
foreach (int fVar in myArr)
    Console.WriteLine("{0} в квадрате равно {1}", fVar, fVar*fVar);
```