Структура программы в C#. Ввод и вывод данных. Оператор присваивания.

Инструкции

Базовым элементом программы являются **инструкции** (операторы). Инструкция представляет некоторое действие, например, арифметическую операцию, вызов метода, объявление переменной и присвоение ей значения. В конце каждой инструкции в C# ставится точка с запятой (;). Данный знак указывает компилятору на конец инструкции. Например:

```
Console.WriteLine("Привет");
```

Данная строка представляет вызов метода Console.WriteLine, который выводит на консоль строку. В данном случае вызов метода является инструкцией и поэтому завершается точкой с запятой.

Набор инструкций может объединяться в блок кода. Блок кода заключается в фигурные скобки, а инструкции помещаются между открывающей и закрывающей фигурными скобками:

```
{    Console.WriteLine("Привет");
Console.WriteLine("Добро пожаловать в C#");}
```

В данном блоке кода две инструкции, которые выводят на консоль определенную строку.

Одни блоки кода могут содержать другие блоки:

```
{
  Console.WriteLine("Первый_блок");
     {       Console.WriteLine("Второй блок");
      }
 }
```

Метод Main

Точкой входа в программу на языке C# является метод Main.

При создании проекта консольного приложения в Visual Studio автоматически создается метод Main:

```
class Program
{
    static void Main(string[] args)
    {
        // здесь помещаются выполняемые инструкции
    }
}
```

По умолчанию метод Main размещается в классе Program. В общем случае название класса может быть любым. Но метод Main является обязательной частью консольного приложения. Если название изменить, то программа не скомпилируется.

И класс, и метод представляют блок кода: блок метода помещается в блок класса. Внутри блока метода Main располагаются выполняемые в программе инструкции.

Регистрозависимость

C# является регистрозависимым языком.

Например, название обязательного метода Main начинается именно с большой буквы.

Если мы назовем метод main или MAIN, то программа не скомпилируется, так как метод, представляющий стартовую точку в приложение, обязательно должен называться Main.

Комментарии

Важной частью программного кода являются комментарии.

Комментарии делают код программы более понятным. Они служат для пояснения кода программы. При компиляции они игнорируются.

В C# есть два типа комментариев: однострочный и многострочный.

Однострочный комментарий размещается на одной строке после двойного слеша //.

Многострочный комментарий заключается между символами /* текст комментария */. Он может размещаться на нескольких строках.

```
1.  using System;
2.
3.  namespace HelloApp
4.  {
5.      /*
6.          программа, которая спрашивает у пользователя имя
7.          и выводит его на консоль
8.      */
```

```
9.      class Program
10.     {
11.        // метод Main - стартовая точка приложения
12.         static void Main(string[] args)
13.         {
14.             Console.Write("Введите свое имя: ");
15.             string name = Console.ReadLine();     // ввод имени
16.             Console.WriteLine($"Привет {name}"); // вывод
    имени на консоль
17.             Console.ReadKey();
18.         }
19.     }
20. }
```

Операторы присваивания

Операторы присваивания устанавливают значение. В них участвуют два операнда.

Как и в других языках программирования, в C# имеется базовая операция присваивания **=**, которая присваивает значение правого операнда левому операнду:

```
int number = 23;
```

Также можно выполнять множественно присвоение сразу нескольких переменным одновременно:

```
int a, b, c;
a = b = c = 34;
```

Операция присваивания имеет низкий приоритет. В начале будет вычисляться значение правого операнда и только потом будет идти присвоение этого значения левому операнду. Например:

```
int a, b, c;
a = b = c = 34 * 2 / 4; // 17
```

Кроме базовой операции присвоения в C# есть еще ряд операций:

+= присваивание после сложения.

A += B равнозначно выражению **A = A + B**

-= присваивание после вычитания.

A-=B эквивалентно **A = A - B**

***=** присваивание после умножения.

A *= B эквивалентно **A = A * B**

/= присваивание после деления.

A /= B эквивалентно **A = A / B**

%= присваивание остатка от деления по модулю.

A %= B эквивалентно **A = A % B**

&= присваивание после поразрядной конъюнкции.

A &= B эквивалентно **A = A & B**

|= присваивание после поразрядной дизъюнкции. :

A |= B эквивалентно **A = A | B**

^= присваивание после операции исключающего ИЛИ.

A ^= B эквивалентно **A = A ^ B**

<<=: присваивание после сдвига разрядов влево.

A <<= B эквивалентно **A = A << B**

>>=: присваивание после сдвига разрядов вправо.

A >>= B эквивалентно **A = A >> B**

```
int a = 10;
a += 10;        // 20
a -= 4;         // 16
a *= 2;         // 32
a /= 8;         // 4
a <<= 4;        // 64
a >>= 2;        // 16
```

Операции присвоения являются правоассоциативными, то есть выполняются справа налево. Например:

```
int a = 8;
int b = 6;
int c = a += b -= 5;    // 9
```

В данном случае выполнение выражения будет идти следующим образом:

b -= 5 (6-5=1)

a += (b-=5) (8+1 = 9)

c = (a += (b-=5)) (c = 9)

Ввод и вывод данных

Ввод и вывод данных в C# выполняется помощью методов.

Для ввода используют метод Console.ReadLine();

Например:

```
string name = Console.ReadLine();
```

Для вывода на экран используют метод Console.Write();

Например:

```
Console.WriteLine($"Привет {name}");
```

После этого, чтобы программа не закрылась, и можно было увидеть результат, используют метод Console.ReadKey();

Этот метод указывает системе на ожидание нажатия клавиши пользователем. После нажатия клавиши приложение соответственно закрывается.