Методы

Метод – это набор операторов, организованный в соответствии с синтаксисом:

[Модификаторы] Тип Имя *(список входных параметров)* ***{ Тело метода }***

Модификаторы – это ключевые слова, обеспечивающие режим использования метода. Пока *(в данном разделе)* будет использован только модификатор **static**, т.к. он позволяет использовать метод без необходимости создания объекта.

Тип *(метода)* – это всегда **тип единственного** значения, которое метод ***может*** возвращать в точку вызова. Конкретное возвращаемое значение определяется аргументом оператора **return имя**. Если тип метода **void**, то оператор **return** *(без аргумента)* в теле метода может ***отсутствовать***, а возврат в точку вызова произойдёт после прохождения «потоком выполнения» ***закрывающей*** фигурной скобки тела метода.

Список входных параметров представляет собой перечисление ***объявлений*** объектов, используемых для передачи данных в метод. Он определяет типы, количество и порядок следования переменных и называется **сигнатурой.** Эти переменные локализуются в теле метода. В отличие от обычных объявлений в списке входных аргументов после каждого ***типа*** должно быть только ***одно имя***.

Тело метода – это фрагмент логически завершённого кода. Тело метода может иметь произвольное количество операторов **return**, каждый из которых должен возвращать значение, соответствующее ***типу метода***.

Передача управления методу *(для начала его работы)* осуществляется **оператором вызова**:

Имя_объекта = Имя_метода *(список* ***фактических*** *аргументов);*

Элементами списка **фактических аргументов** могут быть переменные или константы, тип и порядок следования которых ***точно*** соответствует ***сигнатуре***. Если метод объявлен ***void***, или если возвращаемое значение не используется, часть оператора вызова «**Имя_объекта=**» опускается.

Пример на объявление, вызов и перегрузка методов

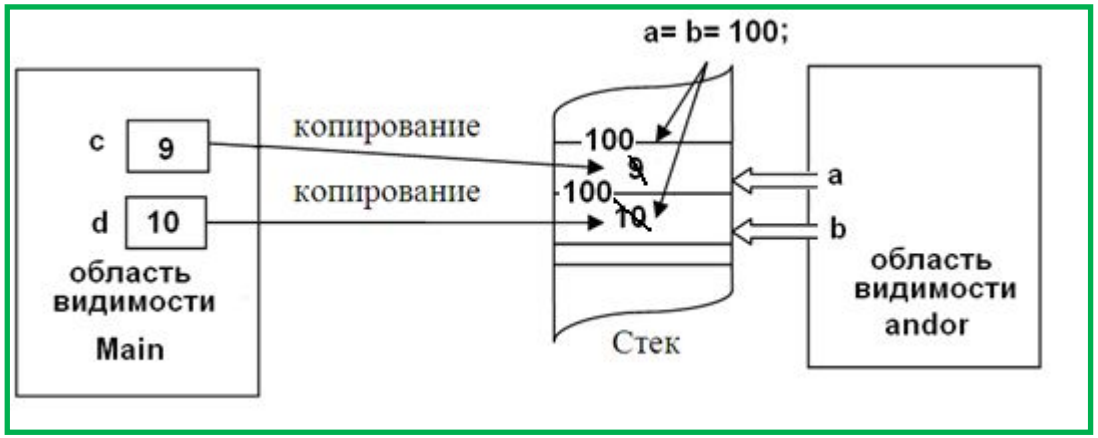
```
using System;
class Методы
{static void Main()
 {
  bool a = true, b = false;
  int c = 9, d = 10;
  Console.WriteLine("До методов:\na={0},b={1},c={2},d={3}",a,b,c,d);
  andor(a, b);
  andor(c, d);
  Console.WriteLine("После методов:\na={0},b={1},c={2},d={3}",a,b,c,d);
 }
static void andor(bool a, bool b)
 {
  Console.WriteLine("В методе andor(bool,bool)");
  Console.WriteLine("ИЛИ {0}", a || b);
  Console.WriteLine("И   {0}", a && b);
  a = false; b = true;
  Console.WriteLine("a={0},b={1}",a, b);
 }
static void andor(int a, int b)
 {
  Console.WriteLine("В методе andor(int,int)");
  Console.WriteLine("ИЛИ {0}", a | b);
  Console.WriteLine("И   {0}", a & b);
  a =  b = 100;
  Console.WriteLine("a={0},b={1}",a,b);
 }
}
```

```
До методов:
a=True,b=False,c=9,d=10
В методе andor(bool,bool)
ИЛИ True
И   False
a=False,b=True
В методе andor(int,int)
ИЛИ 11
И   8
a=100,b=100
После методов:
a=True,b=False,c=9,d=10
```

Методы с одинаковыми **именами**, но различной **сигнатурой** называются **перегруженными**

Механизм передачи входных аргументов

- по списку входных аргументов метода выделяется соответствующее количество ячеек **в стеке**;
- в каждую из выделенных ячеек **<u>копируется</u>** значение соответствующего входного аргумента;
- выделенная ячейка стека становится доступной в теле метода под **именем**, заданным в списке **входных** аргументов;
- по завершению выполнения ***метода*** ячейки с копиями входных аргументов **удаляются** из стека




```
До методов:
a=True,b=False,c=9,d=10
В методе andor(bool,bool)
ИЛИ True
И   False
a=False,b=True
В методе andor(int,int)
ИЛИ 11
И   8
a=100,b=100
После методов:
a=True,b=False,c=9,d=10
```

Модификатор ref

- это способ передачи вызываемому методу **ссылок на объекты**, поименованные в списке входных аргументов *(а не копий их значений)*. Модификатор **ref** указывается дважды, и в ***точке вызова***, и в ***заголовке метода***:

```
using System;
class Методы
{
  static void Main()
    {
    int a = 1, b = 5, c = 10;
    Console.WriteLine("Аргументы до вызова = {0},  {1},  {2}",a, b,c);
    среднее(ref a, ref b, ref c);
    Console.WriteLine("Аргументы после вызова={0}, {1}, {2}",a,b,c);
    }
    static void среднее(ref int a, ref  int b, ref  int c)
    {
      Console.WriteLine("Ср.арифм. значение = {0:f5}",(a+b+c)/3.0);
      a = b = c = 0;
      Console.WriteLine("Аргументы в методе={0}, {1}, {2}",a,b,c);
    }
}
```

```
Аргументы до вызова = 1,  5,  10
Среднеарифметическое значение = 5,33333
Аргументы в методе = 0,  0,  0
Аргументы после вызова = 0,  0,  0
```

Входной параметр метода можно пометить также модификатором **out** *(выходной)*. При этом параметр может **не получать** значения **до вызова** метода, но обязательно должен **получить значение в теле метода**

Массив в качестве входного аргумента метода

Передать **ссылку** *(адрес)* вместо ***копии значения*** объекта можно и в случае, когда в качестве входного аргумента используется **массив,** т.к. массив – это изначально ссылочный объект и его значением является адрес памяти, где реально размещены элементы массива

```
using System;
class Методы
{    static void Main()
{
 int[] a = { 1, 5, 10 };
 float cp_ap;
 среднее(a, out cp_ap);
 Console.WriteLine("Среднеарифметическое значение={0:f5}",cp_ap);
 Console.WriteLine("a[0] = {0}", a[0]);
}
static void среднее(int[] mass, out float cp_ap)
   {
     cp_ap = 0;
     foreach( int i in mass)cp_ap += i;
     cp_ap /= mass.Length;
     mass[0] = 0;
   }
}
```

```
Среднеарифметическое значение = 5,33333
a[0] = 0
```

Модификатор params

В случае, когда в **точке вызова** метода используется несколько входных аргументов **одного типа**, в самом методе они могут быть определены, как элементы массива. Для этого в заголовке метода объявляется массив соответствующего типа с модификатором **params**:

```
using System;
class Методы
{
    static void Main()
    {
     int a = 1, b = 5, c = 10;
     float cp_ap;
     среднее( out cp_ap, a, b, c);
     Console.WriteLine("Среднеарифметическое значение={0:f5}",cp_ap);
    }
    static void среднее(out float cp_ap, params int[] mass)
    {
     cp_ap = 0;
     foreach( int i in mass)
              cp_ap += i;
     cp_ap /= 3;
    }
}
```

Массив, объявленный с модификатором **params,**
должен размещаться ***в конце*** списка входных аргументов

Метод с возвращаемым значением

```
using System;
class Методы
{ static void Main()
    {
     int a =  1   , b= 5, c= 10 ;
     float cp_ap;
     cp_ap =  среднее( a, b, c);
     Console.WriteLine("Среднеарифметическое значение={0:f5}",cp_ap);
     Console.WriteLine(a);
    }
  static float среднее(params int[] mass)
    {
     float cp_ap = 0;
     foreach( int i in mass)
             cp_ap += i;
     cp_ap /= mass.Length;
     mass[0] = 0;
     return cp_ap;
    }
}
```

```
Среднеарифметическое значение = 5,33333
1
```

Обнуление нулевого элемента массива **mass** в методе ***среднее()*** **не изменило** содержимого переменной **a** в методе **Main**:
в массиве **mass,** по-прежнему, находятся **копии объектов**