Функции с побочным эффектом. Перегрузка методов

Функции с побочным эффектом

Функция называется функцией с побочным эффектом, если помимо результата, вычисляемого функцией и возвращаемого ею в операторе **return,** она имеет выходные параметры с зарезервированными словами **ref** и **out.**

Хороший стиль ОО-программирования не рекомендует использование таких функций. Если по каким-либо причинам функция должна возвращать несколько значений, то логичнее оформить ее в виде процедуры с соответствующим набором выходных параметров.

Под **перегрузкой методов** понимается использование в одном классе **нескольких методов с одним именем**. Перегруженные методы, имея одинаковое имя, должны отличаться либо числом аргументов, либо их типами, либо ключевыми словами.

Приведем пример (`TestMethod`), в котором создадим класс `StrMethod`, где опишем три метода с одним именем, каждый из которых выполняет различные действия:

Приведем пример (TestMethod), в котором создадим класс StrMethod, где опишем три метода с одним именем, каждый из которых выполняет различные действия:

```
class StrMethod
  {
      static void Str()
      {
         Console.WriteLine("My test ");
      }

      static void Str(string mystring)
      {
         Console.WriteLine("My test string" + mystring);
      }

      static void Str(string mystr1, string mystr2)
      {
        Console.WriteLine("My test string 1 is"+mystr1);
        Console.WriteLine("My test string 2 is" + mystr2);
      }

  }
```

Эти методы компилятор различает по списку параметров, анализируя их тип и метод передачи.

Если два метода отличаются только возвращаемым значением, то это вызовет ошибку компиляции.

Если два метода отличаются только именами параметров, то для компилятора они идентичны.

Примеры

Пример 1

Рассмотрим **простейший пример** реализации метода **вычисления суммы двух чисел,** используя разные способы передачи параметров

.

Первый способ

```
class Program
    {
        static void Main(string[] args)
        {
            int a = 2, c = 3;
            Console.WriteLine(p(a, c));
            Console.ReadKey();
        }
        static int p(int a1, int c1)
        {
            int s = a1 + c1;
            return s;

        }
    }
```

В методе p вычисляется сумма двух переменных целого типа. Метод возвращает одно значение s, которое вычисляется в этом методе.

Если метод возвращает значение, то имя переменной, в которую помещается возвращаемое значение, указывается после ключевого слова return, присутствие которого в данном случае обязательно.

Второй способ

Те же вычисления выполняются в методе, не возвращающем значения. Если метод не возвращает значения, он имеет тип void..

```
class Program
    {
        static void Main(string[] args)
        {
            int a = 2, c = 3, x;
             ps(a, c, out x);
            Console.WriteLine(x);
            Console.ReadKey();
         }
        static void ps(int a1, int c1, out int s)
        {
            s = a1 + c1;
        }
    }
```

В списке параметров метода перечислены **входные (a1, c1) и выходной (s)** параметры.

Ключевое слово **out** перед выходным параметром, имеющим тип значения, означает передачу параметра по ссылке, т.е. при обращении к методу на место выходного параметра передается адрес аргумента, фигурирующего в обращении к методу.

В примере передается не значение переменной x, а адрес переменной x. Параметр s не является типом int; он является ссылкой на тип int, в данном случае ссылкой на переменную x.

Поэтому после вызова метода значение переменной x изменяется.

Третий способ

Для передачи по ссылке можно использовать также ключевое слово **ref**, но в этом случае аргумент, передаваемый по ссылке, должен быть инициализирован до обращения к методу (при использовании ключевого слова out это необязательно).

Ниже приводится вариант программы с использованием **ref:**

```
class Program
    {
        static void Main(string[] args)
       {
          int a = 2, c = 3, x = 0;
          //x присвоено фиктивное значение 0
          p(a, c, ref x);
          Console.WriteLine(x);
          Console.ReadKey();
        }
          static void p(int a1, int c1, ref int s)
        {
            s = a1 + c1;
        }

    }
```

Пример 2.
Вычислить число сочетаний из n по m по формуле:
$C = n!/(m!(n-m)!)$ (в программе cnm).

Вычисление факториала оформить в виде метода, возвращающего значение (функции)

```
class Program
    {
        static int fact(int n)
        {
            int f = 1;
            for (int i = 2; i <= n; i++)
            {
                f = f * i;
            }
            return f;
        }
        static void Main()
        {
        int n = 5, m = 3 ; int cnm;
        cnm = fact(n)/ (fact(m) * fact(n - m));
        Console.WriteLine("{0}", cnm);
        Console.ReadKey();
        }

    }
```