**Лабораторная работа №6. Параметры методов**

Метод – это функциональный элемент класса, который реализует вычисления или другие действия, выполняемые классом или экземпляром. Методы определяют поведение класса.

Синтаксис объявления метода:

[атрибуты] [спецификаторы] <тип возвращаемого значения> имя\_метода([параметры])

Параметры используется для обмена информацией с методом. Параметр представляет собой локальную переменную, которая при вызове метода принимает значение соответствующего аргумента. Область действия параметра – весь метод. При вызове метода выполняются следующие действия:

1. Вычисляются выражения, стоящие на месте аргументов.
2. Выделяется память под параметры метода в соответствии с их типом.
3. Каждому из параметров попарно сопоставляется соответствующий аргумент.
4. Выполняется тело метода
5. Если метод возвращает значение, оно передаётся в точку вызова.

В C# для обмена данными между вызывающей и вызываемой функциями предусмотрено четыре типа параметров:

* + параметры-значения;
  + параметры-ссылки – описываются с помощью ключевого слова ref;
  + выходные параметры – описываются с помощью ключевого слова out;
  + параметры-массивы – описываются с помощью ключевого слова params;

Ключевое слово предшествует описанию типа аргумента. Параметр-массив может быть только один и должен располагаться последним в списке.

public int Calculate(

int a,

ref int b,

out int c,

params int[] d)

* 1. ***Задание 1. Параметры-значения.***

В отличие от ссылок на объекты, переменные со значимыми типами передаются в функцию по значению, то есть функции передается значение, содержащееся в этой переменной, но не сама переменная. При такой передаче изменение значения соответствующей переменной внутри функции не вызовет изменения значения переданной переменной в вызывающей программе.

1. Создайте новое консольное приложение.
2. Добавьте в проект класс Example, методы которого будут использоваться для примера передачи параметров-значений.
   1. Создайте открытый статический метод Sqr, принимающий в качестве параметров два значения типа double и не возвращающий значения. Этот метод будет возводить два числа в квадрат и выводить результат на экран (для этого используется статический метод Pow класса Math).

**Методические указания.**

{

a = Math.Pow(a, 2);

b = Math.Pow(b, 2);

Console.WriteLine("В методе Sqr. a = {0}\tb = {1}", a, b);

}

* 1. В методе Main создайте и инициализируйте две переменных типа double. Выведите на консоль значения переменных, затем вызовите метод Sqr, указав их в качестве аргументов, после чего повторите вывод на экран.
  2. Теперь снова вызовите метод Sqr, в параметрах указав удвоенные исходные переменные. Покажите значения переменных в конце функции Main.

**Методические указания.**

Example.Sqr(x \* 2, y \* 2);

Console.WriteLine("В методе Main: a = {0}\tb = {1}", a, b);

* 1. Скомпилируйте программу и посмотрите результаты работы

Как мы ни старались, изменяя значения параметров внутри функции или при передаче их в функцию, присвоить им новые значения не удалось. Дело в том, что при передаче по значению метод получает копии значений аргументов, и операторы метода работают с этими копиями. Доступа к исходным значениям аргументов у метода нет, а следовательно, нет и возможности их изменить.

1. Для примера передачи параметров ссылочного типа воспользуемся уже существующим классом Example, дополнив его:
   1. Объявите в классе два открытых поля типа int.
   2. Создайте открытый конструктор, принимающий один параметр целочисленного типа и инициализирующий одно из полей этим значением.

**Методические указания.**

public int x;

public int y;

public Example(int x)

{

this.x = x;

}

* 1. Добавьте в класс Example статический метод с именем Init, принимающий в качестве параметра экземпляр этого же класса и не возвращающий ничего в вызывающую функцию. Метод должен присваивать обоим полям объекта-параметра значения, равные 1. В теле метода так же предусмотрите вывод на консоль значений полей объекта-параметра после их изменения.
  2. В методе Main создайте объект класса Example, передав его конструктору неединичное значение;
  3. Выведите на консоль значения полей созданного объекта;
  4. Передайте только что созданный объект статическому методу Init после чего посмотрите значения его полей с помощью метода Console.WriteLine();

**Методические указания.**

Console.WriteLine("В методе Main: Obj.x = {0}\tObj.y = {1}", Obj.x, Obj.y);

Example.Init(Obj);

Console.WriteLine("В методе Main: Obj.x = {0}\tObj.y = {1}", Obj.x, Obj.y);

* 1. Скомпилируйте измененную программу и посмотрите результат её работы.

Значения полей объекта изменились, несмотря на то, что он был передан функции по значению. Причина в том, что переменная-объект на самом деле хранит ссылку на данные, расположенные в динамической памяти, и именно эта ссылка передаётся в метод. Метод получает в своё распоряжение фактический адрес данных и, следовательно, может их изменить.

1. Измените метод Init таким образом, чтобы он создавал новый объект класса Example, и присваивал его передаваемому в метод параметру. После этого, как и раньше, полям объекта должны присваиваться новые значения, которые будут выведены на консоль в теле метода.
   1. ***Задание 2. Параметры-ссылки***

Если в методе требуется изменить значение каких-либо передаваемых в него величин, используются параметры-ссылки. Признаком параметра-сслыки является ключевое слово ref перед описанием параметра:

ref int x

При вызове метода в область параметров копируется не значение аргумента, а его адрес, и метод через него имеет доступ к ячейке, в которой хранится аргумент. Метод работает непосредственно с переменной из вызывающей функции и, следовательно, может её изменить. При вызове метода перед именем параметра указывается ключевое слово ref.

1. Добавьте в решение (Solution) из первого задания новый проект консольного приложения.
   1. В меню выберите пункт File->Add->Create project.
   2. Далее в окне выбора проекта укажите тип проекта Console Application.
   3. Укажите имя проекта и нажмите OK.
2. Скопируйте код класса Example и функции Main из первого проекта и вставьте его во второй.
3. В методе Sqr измените объявление так, чтобы оба параметра передавались в метод по ссылке.

**Методические указания.**

public static void Sqr(ref double a, ref double b)

1. В функции Init так же измените тип параметра с параметра-значения на параметр-ссылку.
2. Скомпилируйте проект.
   1. При необходимости, измените код функции Main.
   2. Если какой-либо из вызовов функций не корректен для параметров-ссылок, закомментируйте его.
   3. Посмотрите результаты работы программы.
3. Вызовите метод Sqr с неинициализированной переменной типа double.
   1. В теле функции Main объявите, но не инициализируйте новую переменную нужного типа.
   2. Добавьте вызов метода Sqr, передав в качестве второго аргумента только что объявленную переменную.

**Методические указания.**

double c;

Example.Sqr(ref a, ref c);

* 1. Попробуйте скомпилировать проект.
  2. ***Задание 3. Выходные параметры.***

Довольно часто возникает необходимость в методах, которые формируют несколько величин, например, если в методе создаются объекты или инициализируются ресурсы. В этом случае становится неудобным ограничение параметров-ссылок: необходимость присваивания значения аргументу до вызова метода. Это ограничение снимает спецификатор out. Параметру, имеющему этот спецификатор, должно быть обязательно присвоено значение внутри метода, компилятор за этим следит. Зато в вызывающем коде можно ограничиться описанием переменной без инициализации.

1. Добавьте в существующее решение новый консольный проект так же, как в задании 2.
2. Создайте статический метод, принимающий параметр вещественного типа и возвращающий информацию о нём: знак числа, является ли число целым, модуль числа, а так же квадрат числа. При этом само число по выходу из функции измениться не должно.
3. Объявите функцию с одним параметром-значением типа double и выходными параметрами нужных значений: int для знака числа, bool для проверки на целочисленность, double для модуля числа и для квадрата числа. Функция не возвращает значения.
4. В теле функции присвойте соответствующим параметрам необходимые значения в зависимости от входного параметра. Для этой цели удобно использовать статические методы класса Math из пространства имён System.

**Методические указания.**

sign = Math.Sign(value);

isInteger = value % 1 == 0 ? true : false;

abs = Math.Abs(value);

sqr = value \* value;

1. В функции Main создайте две переменных типа double и инициализируйте их значениями 42 и -12.67.
2. Объявите, но не инициализируйте ещё две переменных типа double и по одной переменной типа int и bool.
3. Дважды вызовите метод, формирующий информацию о числе с различными инициализированными переменными. Не забудьте указать в качестве выходных параметров переменные нужного типа. После каждого вызова выводите на консоль результаты работы функции.
4. Скомпилируйте программу и посмотрите на результат её работы.
   1. ***Задание 4. Параметры-массивы.***

Иногда бывает удобно создать метод, в который можно передавать разное число аргументов. Язык C# предоставляет такую возможность с помощью ключевого слова params. Параметр, помеченный этим ключевым словом, может быть только один и должен располагаться последним в списке. Он обозначает массив заданного типа неопределенной длины. Внутри метода к параметрам обращаются, как к обычным элементам массива. Параметр-массив поддерживает все свойства и методы, характерные для массивов.

1. Добавьте четвертый проект консольного приложения к уже сушествующим.
2. Создайте статический метод Avarge, принимающий параметр-массив arr целочисленного типа и возвращающий среднее значение элементов массива.

**Методические указания.**

{

double av = 0;

foreach (int elem in arr)

av += elem;

return (av/arr.Length);

}

1. В функции Main определите массив типа int, состоящий из 4 элементов. С помощью функции Avarge вычислите среднее значение элементов массива.
2. Объявите и инициализируйте две переменные типа short и одну переменную типа byte. Передайте их в качестве параметров функции Avarge.

**Методические указания.**

short z = 1, e = 12;

byte v = 107;

Console.WriteLine(Avarge(z, e, v));

1. Вызовите метод Avarge без параметров.
2. Скомпилируйте проект.
   1. ***Задание 5. Узкие места передачи параметров в функцию.***
3. Создайте проекты, иллюстрирующие следующие особенности передачи параметров в функцию.
   * + - 1. Отличие параметров <тип>[] от params <тип>[]. Передача массивов в функцию.
         2. params – по значению или по ссылке.
         3. Передача в методы строк. Изменение строк в методе.
         4. Object как формальный параметр функции. Определение типа переданного в функцию аргумента.
     1. **Контрольные вопросы**
4. Объясните результат работы последнего пункта программы, иллюстрирующей передачу параметров-значений.
5. Чем отличается передача аргумента ссылочного типа по значению от передачи по ссылке?
6. Почему при передаче в метод в качестве параметра-ссылки неинициализированного значения, проект не компилируется?
7. В чём отличие между параметрами-ссылками и выходными параметрами методов?
8. Можно ли в качестве выходного параметра передавать в функцию инициализированную переменную?
9. Аргументы каких типов можно передавать в функцию со следующим объявлением:

int Example(params double[] x)?

1. Можно ли изменить параметр типа string в теле функции? Изменится ли исходная строка после возврата в точку вызова?
2. Какие модификаторы (ключевые слова) могут использоваться для передачи параметров в функцию?
3. Возможно ли передать один и тот же параметр по ссылке в одну функцию несколько раз? А по значению?
4. Можно ли передавать ссылочный тип с модификатором ref?
5. Необходимо ли инициализировать переменную, которая будет переданна в функцию с модификатором out?
6. Что случиться, если не изменить значение переменной, переданной в функцию с модификатором out?
7. Какое из объявлений функции являеться верным:

static void MyFunction(int a,params int[] values);

или

static void MyFunction(params double[] values, int a);?