ТРЕТЬЕ ЗАНЯТИЕ

МЕТОДЫ

```
<модификаторы> <тип> название_метода (<параметры>)
{
    // тело метода
}
```

БЕЗ ВОЗВРАЩАЕМОГО ЗНАЧЕНИЯ

```
static void Method1()
{
    Console.WriteLine("Method1");
}
```

С возвращаемым значением

```
static int Method2()
{
    Console.WriteLine("Method2");
    return 25;//Должны вернуть значение
}
```

Передача параметров

```
static int Squeare(int a)
{
    Console.WriteLine("Вычесляем квадрат");
    return A * A;
}
```

Модификатор out

- Выходные параметры должны присваиваться вызываемым методом (и, следовательно, передаваться по ссылке). Если параметрам out в вызываемом методе значения не присвоены, компилятор сообщит об ошибке

```
static void main(string[] args)
{
    int val; //Можно не инициализировать
    Sum(1, 3, out val);
    Console.WriteLine(val);
}
static void Sum(int x, int y, out int a)
{
    a = x + y;//Обязаны присвоить значение
}
```

Модификатор ref

- Это значение первоначально присваивается вызывающим кодом и при желании может повторно присваиваться в вызываемом методе (поскольку данные также передаются по ссылке). Если параметрам ref в вызываемом методе значения не присвоены, компилятор никакой ошибки генерировать не будет

```
static void Main(string[] args)
{
    int a = 3;
    int x = 0;//обязаны инициализировать
    Addition(ref x, a);
}
static void Addition(ref int x, int y)
{
    x += y;
    //Но можем x не трогать
}
```

Модификатор params

- Этот модификатор позволяет передавать в виде одного логического параметра *переменное* количество аргументов. В каждом методе может присутствовать только один модификатор params и он должен обязательно указываться последним в списке параметров. В реальности необходимость в использовании модификатора params возникает не особо часто, однако он применяется во многих методах внутри библиотек базовых классов

```
static void Main(string[] args)
{
    Sum(1, 2, -5);
    Sum();
}
static void Sum(params int[] integers)
{
    int result = 0;
    for (int i = 0; i < integers.Length; i++)
    {
        result += integers[i];
    }
    Console.WriteLine(result);
}
```

НЕОБЯЗАТЕЛЬНЫЕ ПАРАМЕТРЫ

```
static int OptionalParam(int x, int y, int z=5, int s=4)
{
    return x + y + z + s;
}//Можно вызвать: OptionalParam(3, 4);
```

ИМЕНОВАННЫЕ ПАРАМЕТРЫ

```
// Необязательный параметр z
// использует значение по умолчанию
//  OptionalParam(y:2,x:3,s:10);
static int NamedParam(int x, int y, int z=5, int s=4)
{
    return x + y + z + s;
}
```


РЕКУРСИЯ
Вызов метода из этого самого метода

Перечисления

представляют собой удобную программную конструкцию, которая позволяет группировать данные в пары "имя-значение". Например, предположим, что требуется создать приложение видеоигры, в котором игроку бы позволялось выбирать персонажа одной из трех следующих категорий: Wizard (маг), Fighter (воин) или Thief (вор).

```
enum <Название>
{
    <конкретное перечисление>,
    <конкретное перечисление>,
    <конкретное перечисление>
}
```

УСЛОВНАЯ КОНСТРУКЦИЯ SWITCH

```
switch (number)
{
    case 1:
        Console.WriteLine("case 1");
        goto case 5; // переход к case 5
    case 3:
        Console.WriteLine("case 3");
        break;
    case 5:
        Console.WriteLine("case 5");
        break;
    default:
        Console.WriteLine("default");
        break;
}
```

```
IF (<УСЛОВИЕ>)
{
    //ЕСЛИ УСЛОВИЕ ИСТИННО
}
ELSE
{
    //ЕСЛИ УСЛОВИЕ ЛОЖНО
}
```