

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  
}
```