Способы передачи аргументов в метод

Аргументы передаются:

- По значению
- По адресу (ссылке)

- *При передаче по значению* метод получает копии значений аргументов, и операторы метода работают с этими копиями.
- *При передаче по ссылке* (*по адресу*) метод получает копии адресов аргументов и осуществляет доступ к аргументам по этим адресам.

Типы параметров

В C# четыре типа параметров:

- параметры-значения - для исходных данных метода;
- параметры-ссылки (**ref**) - для изменения аргумента;
- выходные параметры (**out**) – для формирования аргумента;
- параметры-массивы (**params**) – для переменного кол-ва аргументов.

по адресу

Пример:

```
public int Calculate( int a, ref int b, out int c, params int[] d ) { …
```

параметр-значение

параметр-ссылка

выходной параметр

параметр-массив

Передача аргумента по значению

- При вызове метода на месте параметра, передаваемого по значению, может находиться выражение (а также его частные случаи — переменная или константа).
- Должно существовать неявное преобразование типа выражения к типу параметра.

```
double a = 0.1;
double b = Math.Sin(a);
double c = Math.Sin(b-2*a);
```

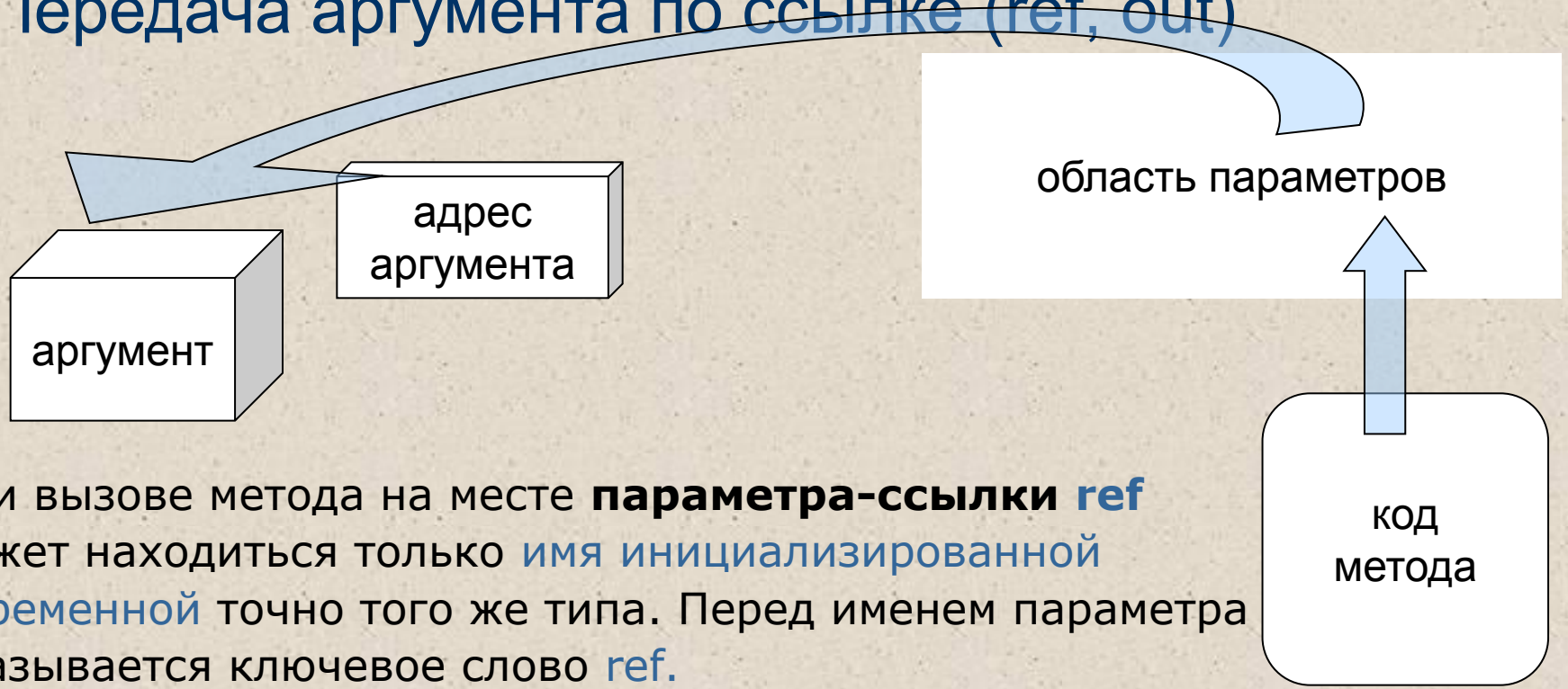
```
static int Max(int a, int b) { … }
…
int x = Max(3, z);
```

Пример: счетчик

```
class Counter
{    public void Inc(int delta)
     {
          n += delta;
      }
       public override string ToString() { return n.ToString(); }
       int n;
}

class Program
{     static void Main(string[] args)
      {    Counter num = new Counter();
           num.Inc(4);
           int a = 3;
           num.Inc(2*a);
           Console.WriteLine("значение счетчика " + num);
}}
```

Передача по значению

Передача аргумента по ссылке (ref, out)



- При вызове метода на месте **параметра-ссылки ref** может находиться только имя инициализированной переменной точно того же типа. Перед именем параметра указывается ключевое слово ref.
- При вызове метода на месте **выходного параметра out** может находиться только имя переменной точно того же типа. Ее инициализация не требуется. Перед именем параметра указывается ключевое слово out.

```
int SomeMethod(ref int a, out int b) { … }
…
int s = 0; int z;
int x = SomeMethod(ref s, out z);
```

Пример: параметры-значения и ссылки ref

```
using System;
namespace ConsoleApplication1
{   class Class1
    {   static void P( int a, ref int b )
        {
            a = 44; b = 33;
            Console.WriteLine( "внутри метода {0} {1}", a, b );
        }
        static void Main()
        {
            int a = 2, b = 4;
            Console.WriteLine( "до вызова     {0} {1}", a, b );
            P( a, ref b );
            Console.WriteLine( "после вызова  {0} {1}", a, b );
}}}
```

Результат работы программы:
до вызова 2 4
внутри метода 44 33
после вызова 2 33

Пример: выходные параметры out

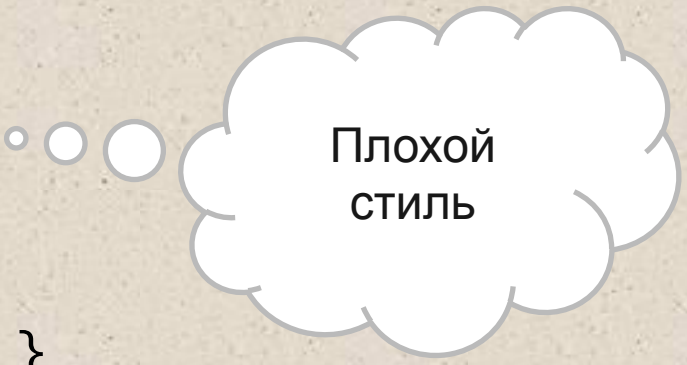
```
using System;
namespace ConsoleApplication1
{   class Class1
    {   static void P( int x, out int y )
        {
            x = 44; y = 33;
            Console.WriteLine( "внутри метода {0} {1}", x, y );
        }
        static void Main()
        {
            int a = 2, b;         // инициализация b не требуется

            P( a, out b );
            Console.WriteLine( "после вызова  {0} {1}", a, b );
}}}
```

Результат работы программы:

внутри метода 44 33
после вызова 2 33

Пример: счетчик (параметр out)

```
class Counter
{     public bool Sync(out int x)
      {    x = n;
           return n==0 ? false : true ;
       }
       public void Set( int start ) { n = start; }
       int n;
        ...
}
class Program
{     static void Main(string[] args)
       {    Counter num1 = new Counter();
            Counter num2 = new Counter();
            ...
            int temp;
            if ( num1.Sync(out temp) ) num2.Set(temp);
            …
}}
```



Summary: Правила применения параметров

1. Для параметров-значений используется передача по значению. Этот способ применяется для исходных данных метода.

- При вызове метода на месте аргумента параметра-значения может быть выражение (а также его частные случаи — переменная или константа). Должно существовать неявное преобразование типа выражения к типу параметра.

2. Параметры-ссылки и выходные параметры передаются по адресу. Этот способ применяется для передачи побочных результатов метода.

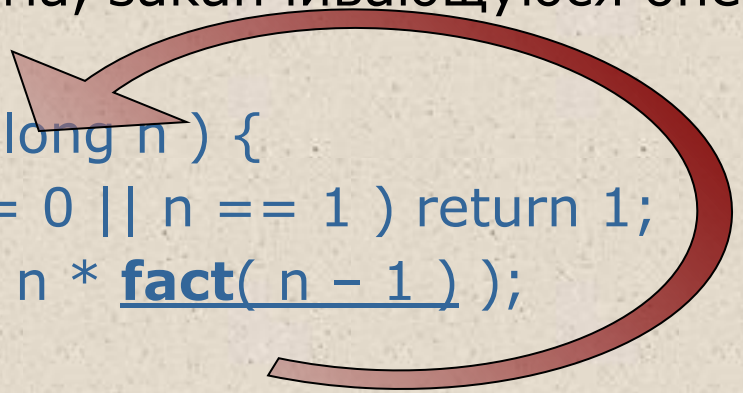
- При вызове метода на месте аргумента параметра-ссылки **ref** может находиться только имя инициализированной переменной точно того же типа. Перед именем параметра и аргумента указывается ключевое слово ref.
- При вызове метода на месте выходного параметра **out** может находиться только имя переменной точно того же типа. Ее инициализация не требуется. Перед именем параметра и аргумента указывается ключевое слово out.

Методы с переменным количеством аргументов

```
class Class1
{    public static double Average( params int[] a )
     {  if ( a.Length == 0 )
                throw new Exception( "Недостаточно аргументов");
        double sum = 0;
        foreach ( int elem in a ) sum += elem;
        return sum / a.Length;
      }
     static void Main()
     {   try
         {  short z = 1, e = 13, w = 4;
            Console.WriteLine( Average( z, e, w ) );        //  6
            byte v = 18;
            Console.WriteLine( Average( z, e, w, v) );      //  9
            int[] b = { -11, -4, 12, 14, 32, -1, 28 };
            Console.WriteLine( Average( b ) );              // 38
            Console.WriteLine( Average() );   // Недостаточно аргументов
         }
         catch( Exception e ) { Console.WriteLine( e.Message ); return; }
         catch                { Console.WriteLine( "ой-кшмр" ); return; }
}}
```

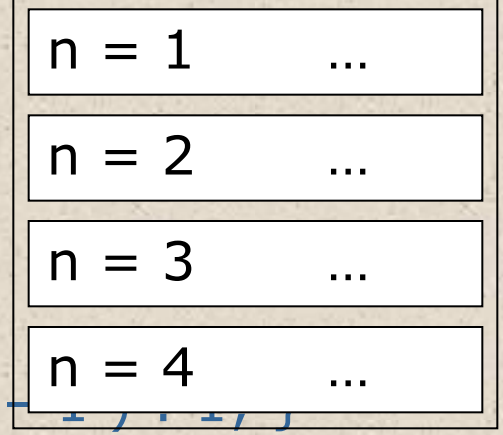
Рекурсивные методы

- *Рекурсивным* называется метод, который вызывает сам себя (*прямая рекурсия)*. *Косвенная* рекурсия - когда два или более метода вызывают друг друга.
- Для завершения вычислений каждый рекурсивный метод должен содержать *хотя бы одну нерекурсивную ветвь* алгоритма, заканчивающуюся оператором возврата.



```
long fact( long n ) {
    if ( n == 0 || n == 1 ) return 1;     // нерекурсивная ветвь
    return ( n * fact( n – 1 ) );          // рекурсивная ветвь
}
… long m = fact(4);
```

стек




```
// или:
long fact( long n ) { return ( n > 1 ) ? n * fact( n – 1 ) : 1; }
```

Характеристики рекурсии

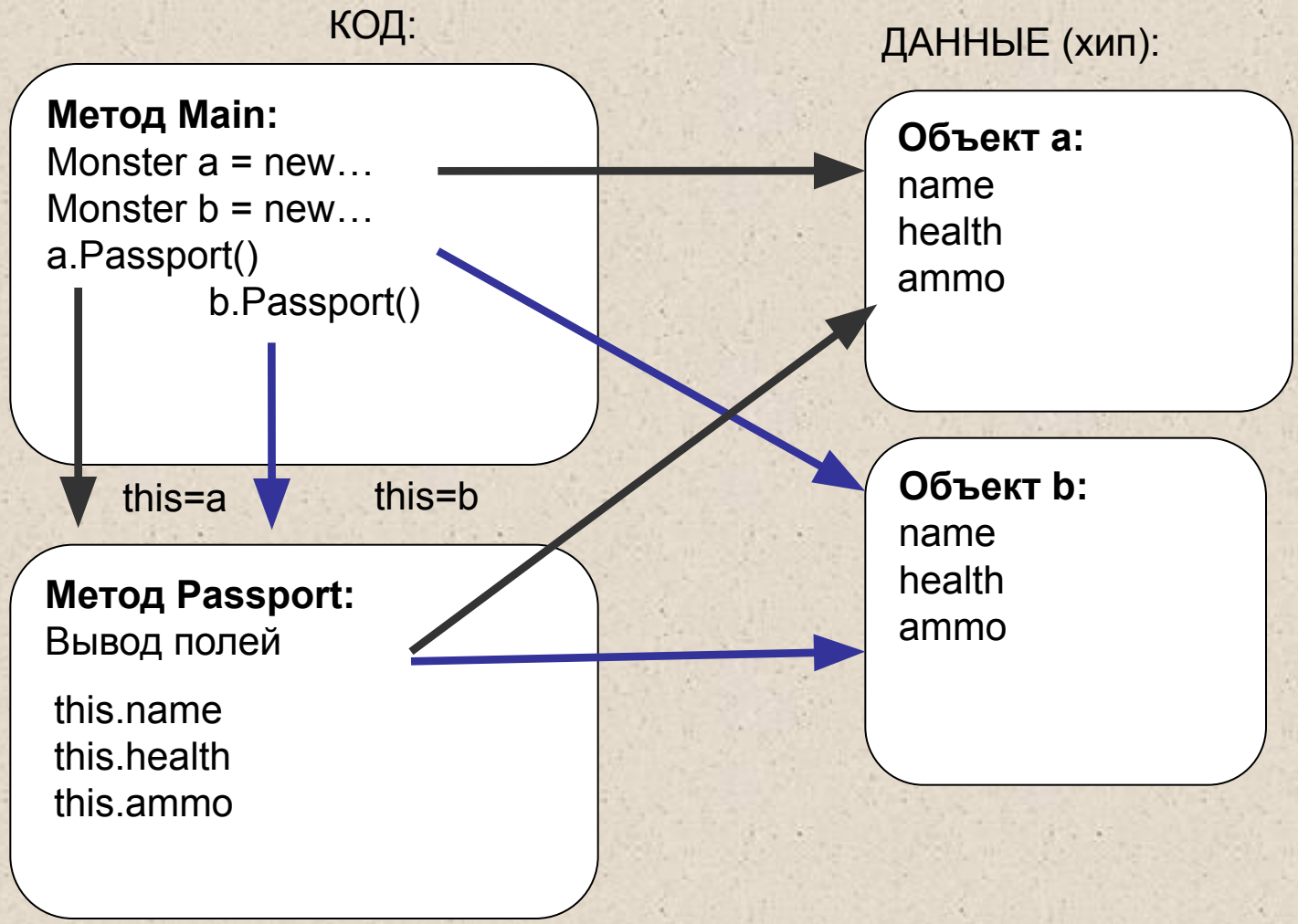
Достоинство рекурсии: компактность записи.

Недостатки: опасность переполнения стека; расход времени и памяти на повторные вызовы метода и передачу ему копий параметров.

Рекурсивные методы используются в основном для работы с рекурсивными структурами данных, например, бинарными деревьями.

Ключевое слово this

Чтобы обеспечить работу метода с полями того объекта, для которого он был вызван, в метод автоматически передается скрытый параметр this, в котором хранится ссылка на вызвавший функцию объект.



Использование явного this

В явном виде параметр this применяется:

1) чтобы возвратить из метода ссылку на вызвавший объект:

```
class Demo
{       double y;
        public Demo T()     { return this; }

// 2) для идентификации поля, если его имя совпадает с
//      именем параметра метода:

        public void Set_y( double y ) { this.y = y; }
}
```

Пример: счетчик (this)

```
class Counter
{     public bool Sync(out int x)
      {    x = n;
           return n==0 ? false : true ;
       }
       public void Set( int start ) { n = start; }
       int n;
        ...
}
class Program
{     static void Main(string[] args)
       {    Counter num1 = new Counter();
            Counter num2 = new Counter();
            ...
            int temp;
            if ( num1.Sync(out temp) ) num2.Set(temp);
            ...
}}
```

public void Set(int n) { this.n = n; }