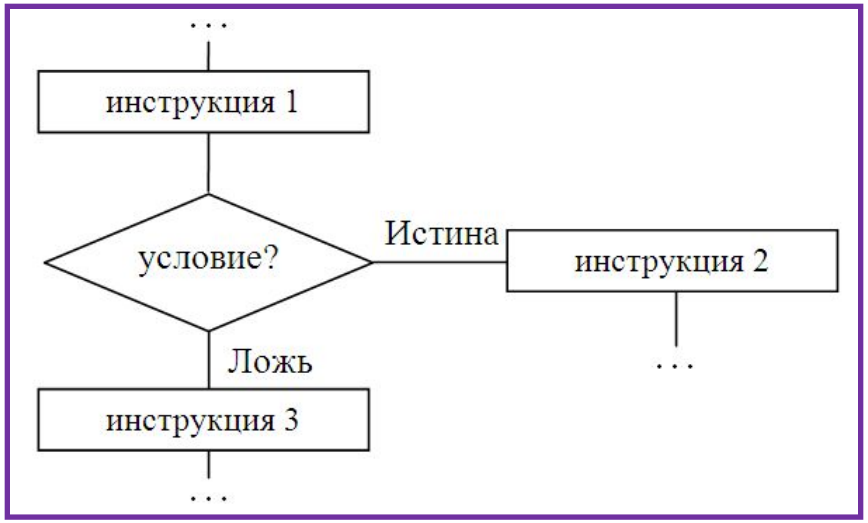
Операторы управления

Типы вычислительных процессов

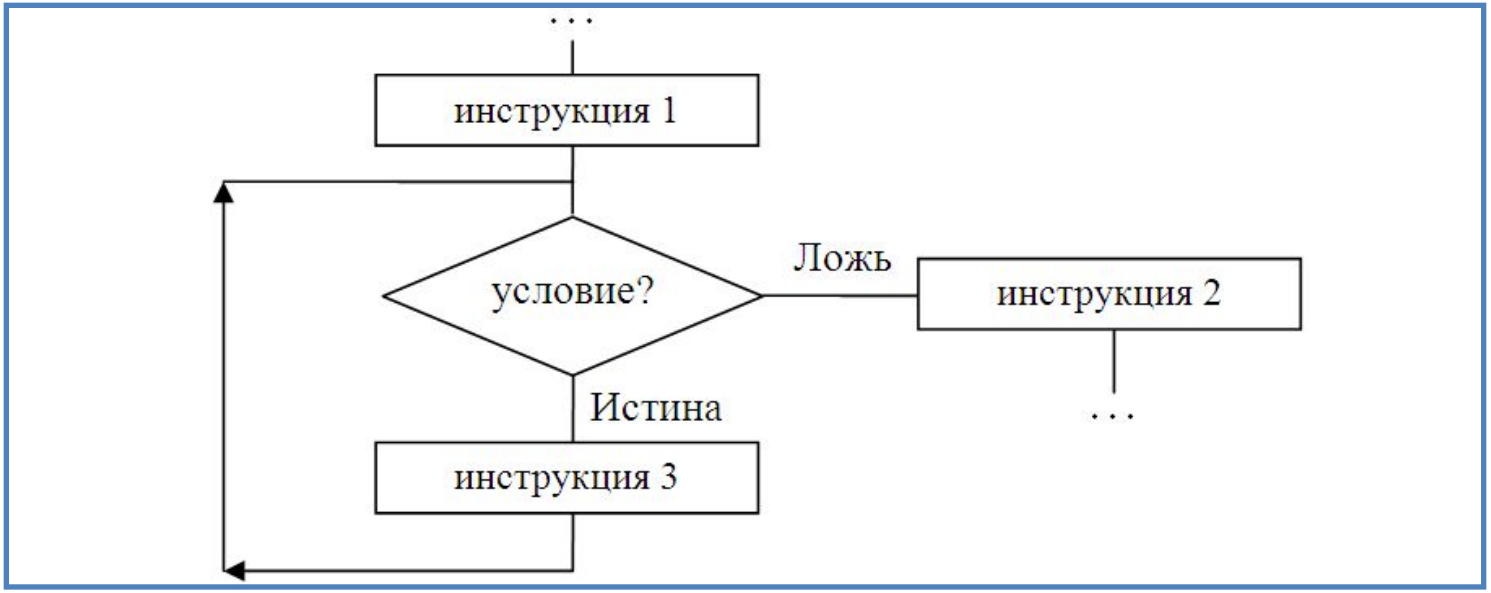
Линейный

Ветвления



Циклический

Циклы в программах составляют, в среднем, 80–85% общего объёма кода



Операторы выбора

Оператор if

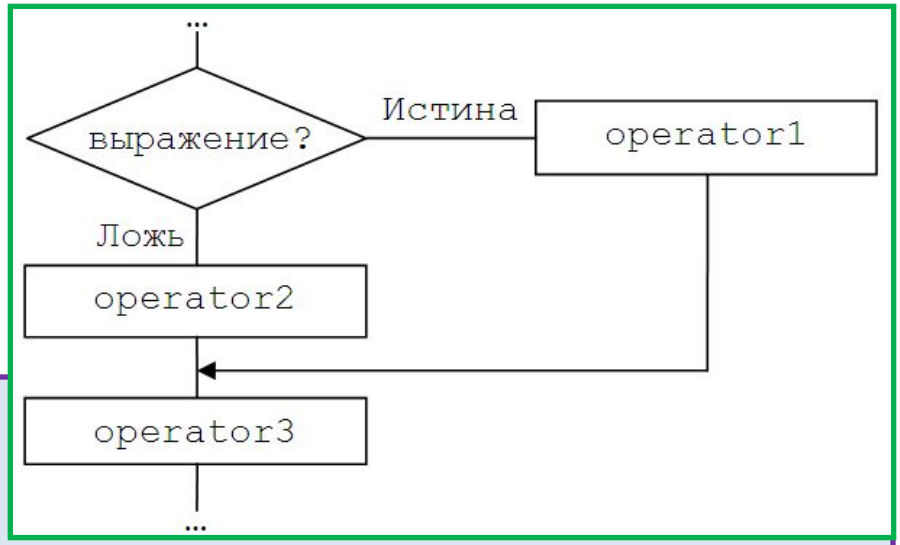
```
if (выражение) operator1;
   operator2;
```

значение в скобках булевого типа


```
using System;
{
      static void Main()
      {
         int  i, j, k;
         string s;
         Console.WriteLine("Задайте первое число!");
         s = Console.ReadLine();
         i = Convert.ToInt32(s);
         Console.WriteLine("Задайте второе число!");
         s = Console.ReadLine();
         j = Convert.ToInt32(s);
         if (i < j ) k = j;
         if (i > j ) k = i;
        Console.WriteLine("Максимум из заданных чисел = {0}",k);
       }
  }
```

Оператор if - else

```
if  (выражение) operator1;
    else operator2;
operator3;
```

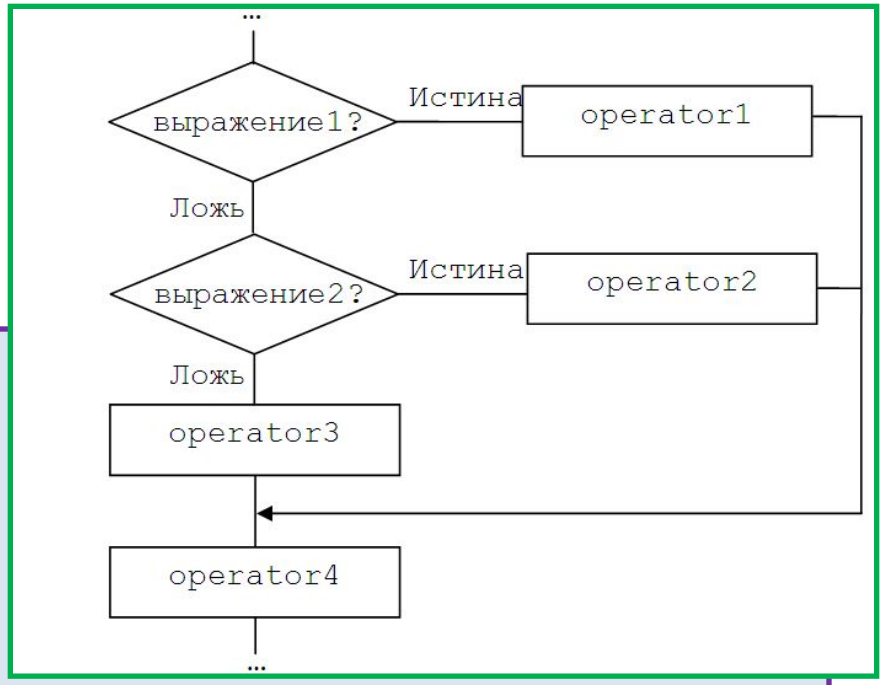



```
using System;
{
      static void Main()
      {
         int  i, j, k;
         string s;
         Console.WriteLine("Задайте первое число!");
         s = Console.ReadLine( );
         i = Convert.ToInt32 ( s );
         Console.WriteLine ("Задайте второе число!" );
         s = Console.ReadLine( );
         j = Convert.ToInt32 ( s );
         if (i > j ) k = i;
             else k = j;
          Console.WriteLine ("Максимум из заданных чисел = {0}", k );
       }
  }
```

«простой» if

Оператор if-else-if

```
if (выражение1) operator1;
    else if (выражение2) operator2;
         else operator3;
operator4;
```




```
using System;
class Primer
{
   static void Main()
   {
    int x, f;
    string s;
    Console.WriteLine("\t\tРаботает сигнальная функция");
      Console.WriteLine("Задайте значение x!");
      s = Console.ReadLine();
      x = Convert.ToInt32(s);
      if (x < 0) f = -1;
        else if (x > 0) f = 1;
              else f = 0;
       Console.WriteLine("Значение сигнальной функции = {0}", f);
   }
}
```

$$f(x) = \begin{cases} 1, & \text{для всех } x > 0 \\ 0, & \text{для } x = 0 \\ -1, & \text{для всех } x < 0 \end{cases}$$

```
                Работает сигнальная функция
Задайте значение x!
100
Значение сигнальной функции = 1
```

Блочный оператор

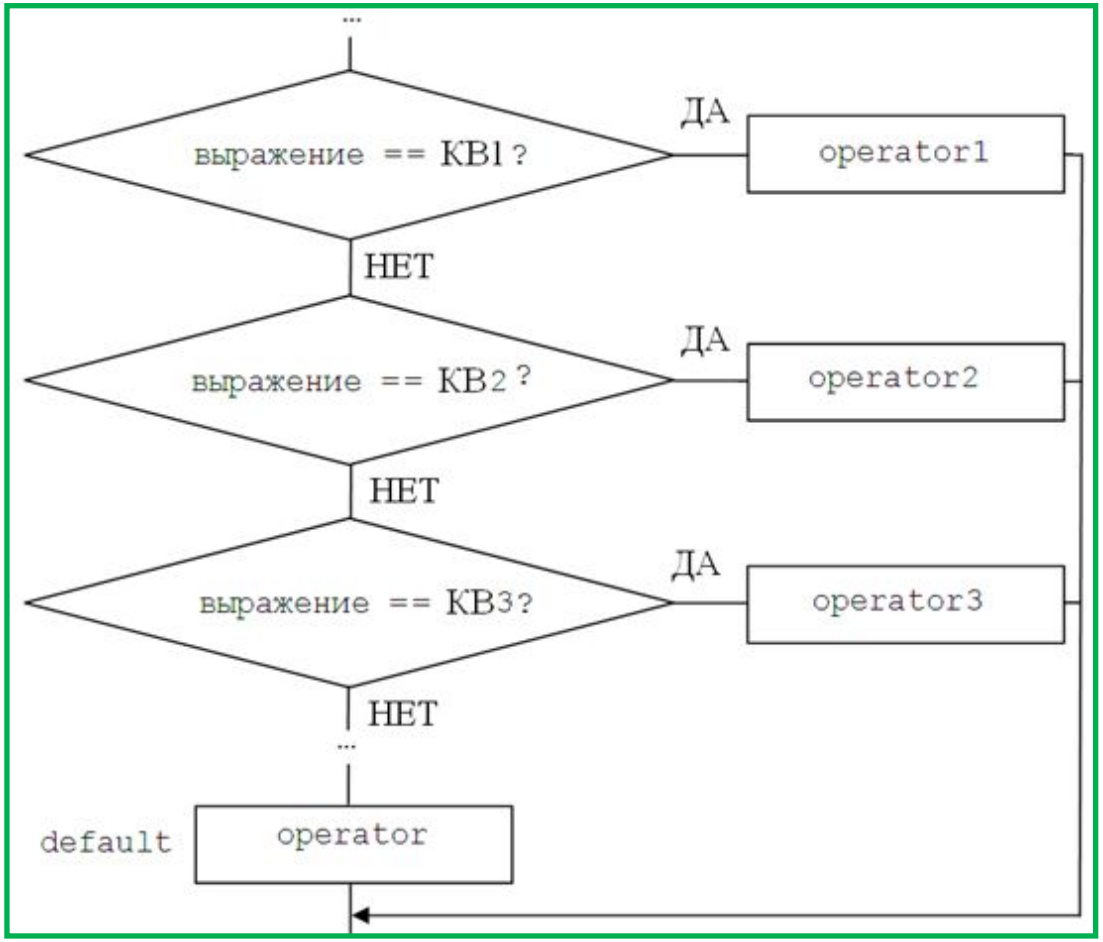
1. ***Блочный оператор (или блок) – это инструкции (операторы), размещённые внутри парных фигурных скобок.***
2. ***Самостоятельное значение имеют блоки в операторах выбора и циклов.***
3. ***В операторах выбора*** *(а также – циклов)* ***на месте operator может находиться блок операторов, инструкции в котором будут выполнены в соответствии с рассматриваемой логикой***

```
using System;
class Primer
{   static void Main()
    {   int i, j, max, min;
        string s;
        Console.WriteLine("Задайте первое число!");
        s = Console.ReadLine();
        i = Convert.ToInt32(s);
        Console.WriteLine("Задайте второе число!");
        s = Console.ReadLine();
        j = Convert.ToInt32(s);
         if (i > j) { max = i; min = j; }
                    else { max = j; min = i; }
        Console.WriteLine("Максимальное = {0}, минимальное = {1} ",max, min);
    }
 }
```

Оператор множественного выбора switch

- выполняет те же действия, что и многоступенчатый if else, но более наглядным образом

```
switch (выражение)
{
case КВ1 : operator1; break;
case КВ2 : operator2; break;
case КВ3 : operator3; break;
…
default: operator; break;
}
```



КВ – константное выражение. Обычно вместо него используется целочисленная или строковая **константа**.

На месте любого оператора может быть последовательность операторов (раздел). Заканчиваться раздел должен оператором перехода. Обычно используют ***break***, действием которого является передача управления за границу блока **switch**

Метка default помечает раздел, который выполняется, если не было вхождения в какую-либо метку с **КВ**

Примеры на switch

```
using System;
class Primer2
{
   static void Main()
   {
      int  x= 0xd;
      switch(x)
      {
         default: x += 1; goto case 3;
         case 1:  x += 2; break;
         case 2:  x += 3; goto case 1;
         case 3:
         case 4:  x += 4; goto case 2;
         case 5:  x += 5; break;
         case 6:  x += 6; break;
      }
      Console.WriteLine("x=  {0} ", x);
   }
}
```

x = 23

метку **default:** можно размещать в любом месте **switch,** но только не после пустой метки !
(в этом примере нельзя после case 3:)

```
using System;
class Primer1
{
   static void Main()
   {
      int x=3;
      switch(x)
      {
         case 2:  x+=2;   break;
         case 1:  x+=1;   break;
         case 3:
         case 4:
         case 5:  x+=5;   break;
         default: x-=10;  break;
      }
      Console.WriteLine("x = {0}", x);
   }
}
```

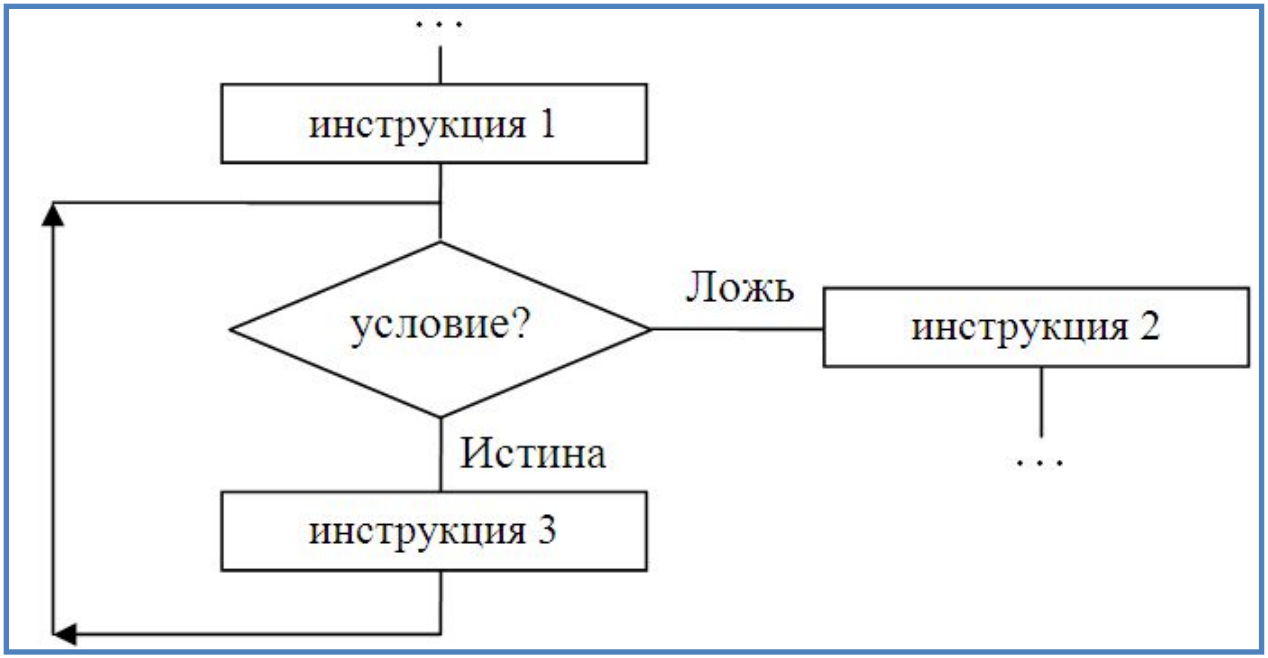
x = 8

Оператор **goto** осуществляет переход на **метку**, имя которой указывается в качестве его параметра.
Метка может располагаться или в **том же**, или в более **внешнем блоке**.

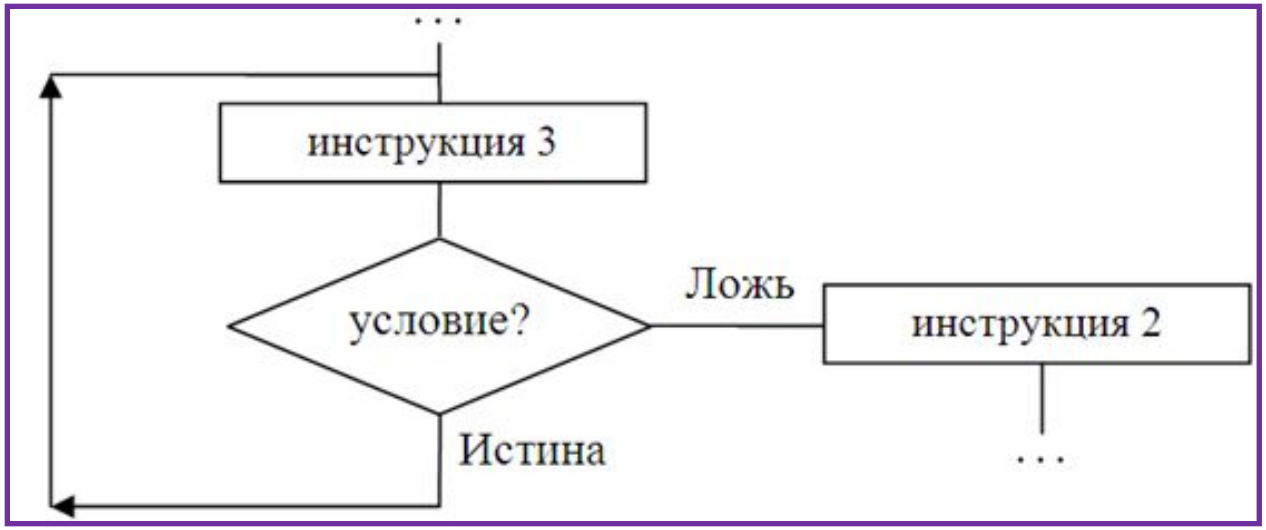
Операторы циклов

Типы циклов

с предусловием



с постусловием



инструкция3 – это т.н. «тело цикла»

Оператор цикла while

while (выражение) оператор1;
оператор2;

на месте «оператор1» может быть:
- простой,
- пустой,
- блочный оператор.

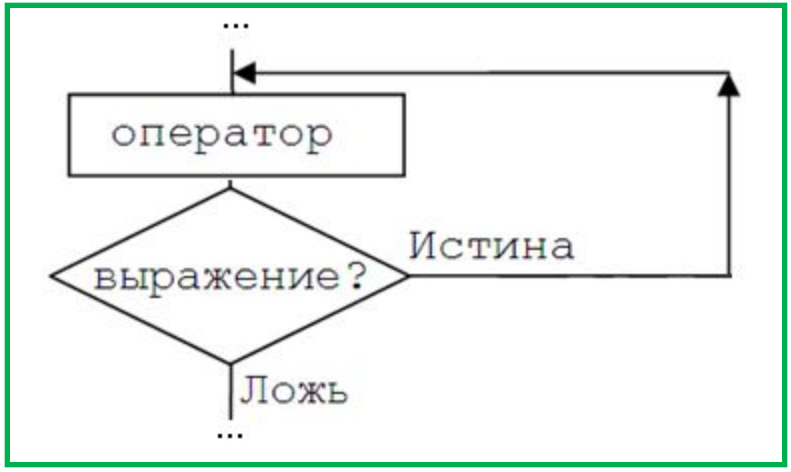
```
using System;
class Primer
{ static void Main()
   {  uint  i,  j = 1 ;
      uint   f = 1;
      string s;
      Console.WriteLine("Задайте натуральное число!");
      s = Console.ReadLine();
      i = Convert.ToUInt32(s);
      while ( j  <=  i)
            { f *= j;
              j++;
            }
    Console.WriteLine("Факториал  от  {0}  = {1}", i, f);
   }
}
```

```
Задайте натуральное число!
18
Факториал  от  18  = 3396534272
Задайте натуральное число!
19
Факториал  от  19  = 109641728
```

Справочно: max_{uint} = 4294967295

Оператор цикла do while

```
do оператор;
   while выражение;
```




```
using System;
class Primer
{
   static void Main()
   {
      int num= 12345, next;
      do
      {
         next = num % 10;
         Console.Write( next );
         num = num / 10;
      }
      while ( num > 0 );
   }
}
```

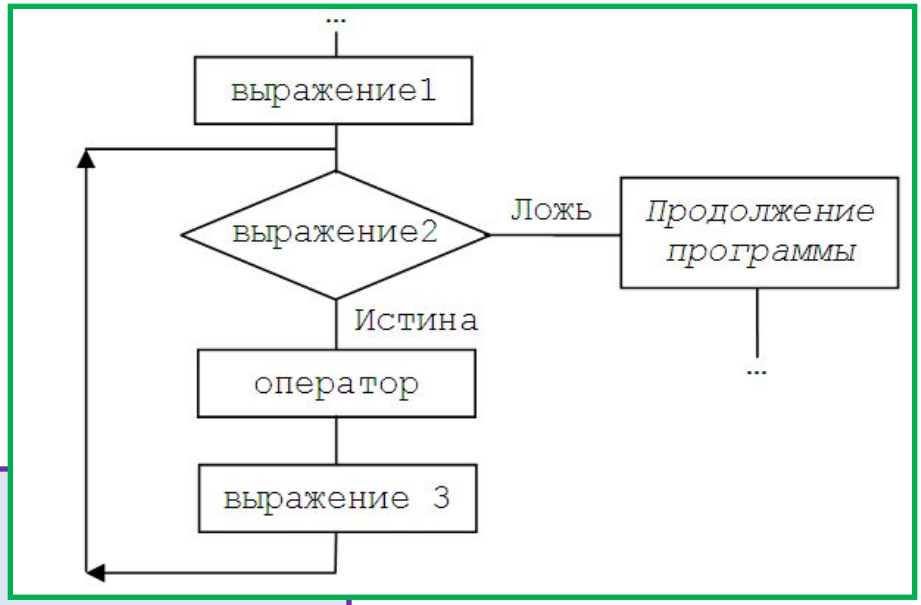
54321

Оператор цикла for

for (выражение1; выражение2; выражение3) оператор ;

Является универсальным и представляет собой своего рода шаблон для типичного цикла

for (инициализация; условие; итерация)
оператор ;




```
using System;
class Primer
{ static void Main()
   {   uint  i, j;
       uint  f;
       string s;
       Console.WriteLine("Задайте натуральное число!");
       s = Console.ReadLine();
       i = Convert.ToUInt32( s );
       for (f = j = 1; j <= i; j++) f *= j;
       Console.WriteLine("{0}!  = {1}", i, f);
   }
}
```

формально: пустой цикл

```
for (f = j = 1; j <= i; f*=j++);
```

бесконечный и пустой цикл:
for (;;);

Цикл for *(пример)*

```
using System;
class Primer
{
   static void Main()
   {
      int  i, sum = 0 ;
      for (i = 1; i < 10; i+=2, sum++) ;
      Console.WriteLine("i={0}, sum = {1}", i, sum);
   }
}
```

```
i=11, sum = 5
```

Цикл for (пример)

```
using System;
class Primer
{
    static void Main()
    {
     int  i, j, k = 0 ;
     for ( i = 0, j = 10; i <= j; i++, j--) k++;
     Console.WriteLine("i={0}, j = {1},  k = {2}", i, j, k );
    }
}
```

i= 6, j = 4, k= 6

Операторы goto, break *и* continue *в циклах* (*пример1*)

```
using System;
class Primer
{  static void Main( )
   {   short   i, j;
       short f ;
       string s;
       Console.WriteLine("Задайте натуральное число!");
       s = Console.ReadLine( );
       i = Convert.ToInt16 ( s );
       for (f = j = 1; j <= i; j++)
       {
          f  *=  j;
          if ( j == 7 ) break;
       }
       Console.WriteLine("{0}!  = {1}", i, f );
   }
}
```

```
Задайте натуральное число!
6
6!   = 720
Задайте натуральное число!
7
7!   = 5040
Задайте натуральное число!
8
8!   = 5040
```

8! = 40320

$max_{short} = 2^{15} - 1 = 32767$

Операторы goto, break *и* continue *в циклах* *(продолжение примера1)*

```
using System;
class Primer
{
   static void Main ( )
   {
      short i, j;
      short f;
      string s;
      Console.WriteLine( "Задайте натуральное число!" );
      s = Console.ReadLine ( );
      i = Convert.ToInt16 ( s );
      for (f = j = 1; j <= i; j++)
      {
         f *= j;
         if (j == 7) goto m1;
      }
      Console.WriteLine("{0}!  = {1}", i, f);
      return;
m1:  Console.WriteLine("Наибольшее  {0}!  = {1}", j, f);
   }
}
```

```
Задайте натуральное число!
6
6!   = 720
```

```
Задайте натуральное число!
7
Наибольшее   7!   = 5040
```

```
Задайте натуральное число!
10
Наибольшее   7!   = 5040
```

Операторы goto, break и continue в циклах (ещё один вариант примера1)

```
using System;
class Primer
{
   static void Main()
   {
      short i, j;
      short f;
      string s;
      Console.WriteLine ( "Задайте натуральное число!" );
      s = Console.ReadLine ( );
      i = Convert.ToInt16 ( s );
      for (f = j = 1; j <= i; j++)
      {
         if ( j > 7 ) continue;
         f *= j;
      }
   Console.WriteLine("{0}!  = {1}",  (i<=7) ? i : (short)7, f);
   }
}
```

```
Задайте натуральное число!
10
7!   = 5040
```

Операторы goto, break *и* continue *в циклах* *(пример2)*

```
using System;
class Primer
{  static void Main()
   {   short i, j;
       short f =0;
       string s;
       Console.WriteLine("Задайте натуральное число!");
       s = Console.ReadLine();
       i = Convert.ToInt16(s);
       for ( j = 1; j <= i; j++)
       {
          if ( j%2 == 0) continue;
          f += j;
       }
Console.WriteLine("Сумма нечётных интервала 1 - {0}= {1}", j, f);
   }
}
```

```
Задайте натуральное число!
10
Сумма нечётных интервала  1 - 11   = 25
```

Вложенные циклы

```
using System;
class Primer
{
   static void Main()
   {
      for (short j = 1; j <= 5; Console.WriteLine( ), j++)
         for (short i = 1; i < 5; i++)
            Console.Write(" \t{0}", i * j );
   }
}
```

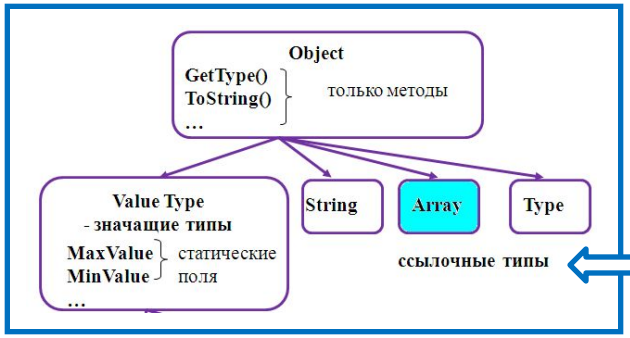
```
1   2   3   4
2   4   6   8
3   6   9   12
4   8   12  16
5   10  15  20
```

```
using System;
class Primer
{
   static void Main()
   {
     for (short j = 1; j <= 5; Console.WriteLine(),  j++)
         for (short i = 1; i <= j; i++)
            Console.Write("{0,5}", i * j);
    }
}
```

```
1
2    4
3    6    9
4    8    12   16
5    10   15   20   25
```

Массивы

- агрегированные объекты, состоящие из *заданного* количества *однотипных* элементов.



Массивы в C# – это ссылочные типы, производные от базового класса *System.Array;*

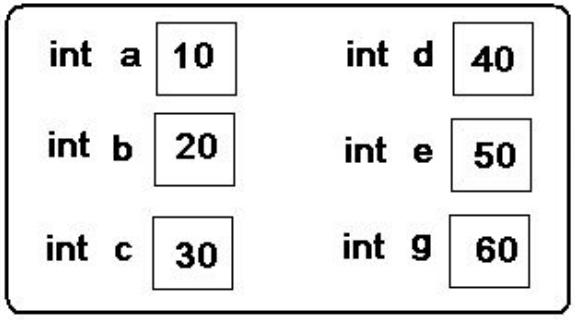
Массивы различают по *размерности* и *типу* составляющих их элементов.

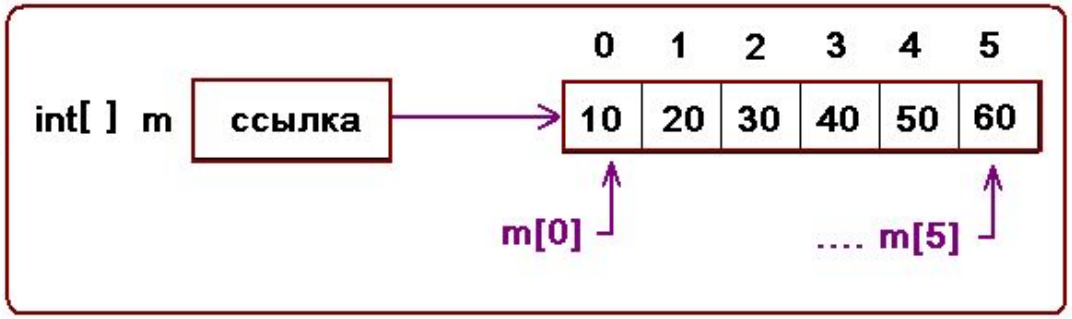
Основное преимущество – возможность выбора *элемента массива* по его *(т.е.элемента)* индексу.

Индекс элемента массива - это его *порядковый номер*.

Индексация элементов массива в C# всегда начинается с *нуля*.

Массив представляет собой простейшую коллекцию

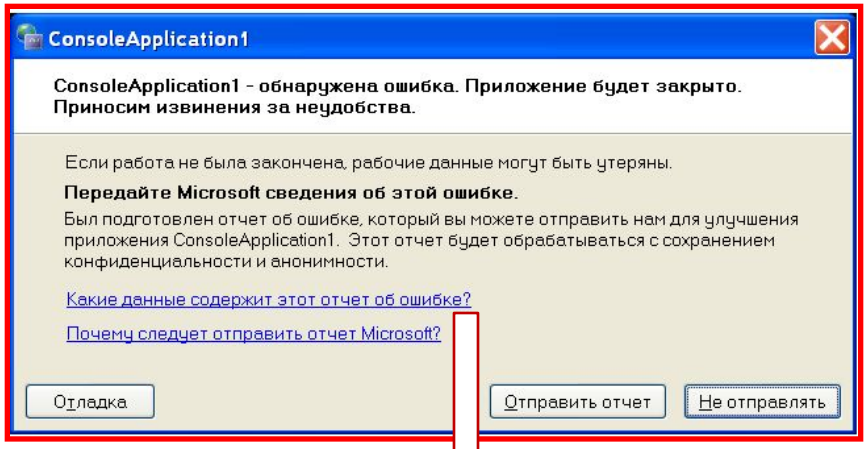




Нарушение границ массива

Так как индексация элементов массива (по умолчанию) начинается с нуля, то заканчивается она на элементе с индексом N-1, если N – количество элементов массива.

Система осуществляет контроль за соблюдением границ массива:




```
Необработанное исключение:
System.IndexOutOfRangeException:
Индекс находился вне границ массива.
в Массивы.Main() в C:\Proects\Primer\пример.cs:строка 6
```

Одномерные массивы

тип[ ] ИмяМассива = new тип [КоличествоЭлементов] {Блок инициализаторов};

```
using System;
class Массивы
      {
         static void Main()
          {
            int [ ]  m1= new int [ 4 ], m2 = new   int [ ]  { 2, 4, 6, 8}, m3 = {1, 3, 5, 7 };
            for (short j = 0; j < 4;) m1 [ j ] = ++j;
            int сумма = 0 ;
            for (short i = 0 ; i <= 3; i++) сумма += m1[ i ] + m2[ i ] + m3[ i ];
           Console.WriteLine("{0:d}", сумма);
          }
      }
```

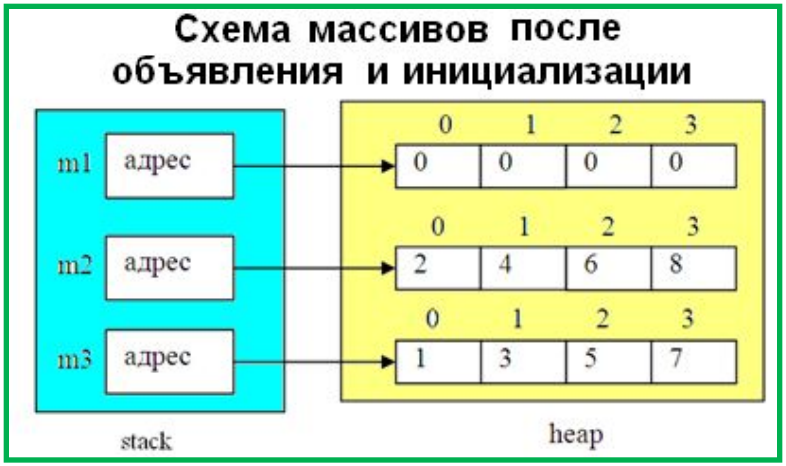
1

2

блок инициализации

3

46



1 - массиву **m1** выделяется память для четырех элементов, которые инициализируются нулём;
2 - для массива **m2** количество элементов определяется по блоку инициализации;
3 - для создания и инициализации массива **m3** использована предельно краткая запись

Оператор цикла для работы с коллекциями **foreach**

foreach (Элемент **in** ИмяМассива) оператор ;

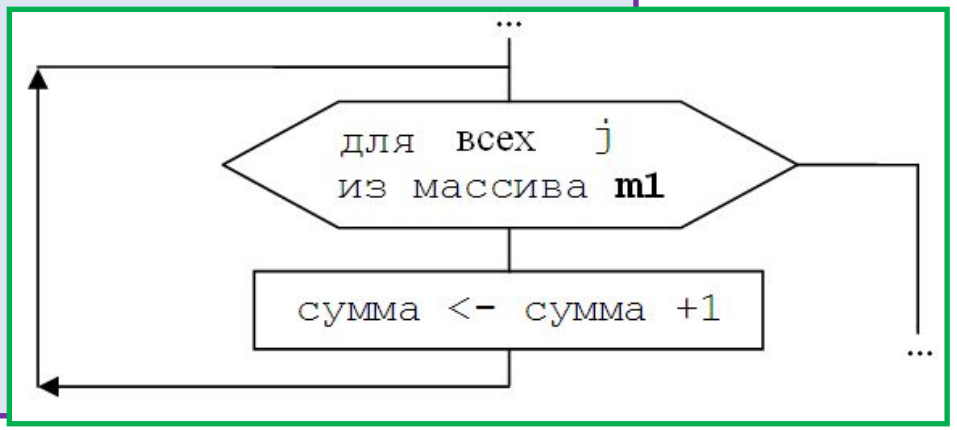
тип «Элемента» должен совпадать с типом массива

foreach может быть использован только для ***чтения*** значений элементов массива, инициализировать или изменять значения **он не умеет**

```
using System;
class Массивы
{
   static void Main()
    {
      int[ ] m1 = new int [4], m2 = new int [ ] {2,4,6,8}, m3 = {1,3,5,7};
      for (short j = 0; j < 4; ) m1[ j ] = ++j;
      int сумма = 0 ;
      foreach (int j in m1) сумма += j;
      foreach (int j in m2) сумма += j;
      foreach (int j in m3) сумма += j;
      Console.WriteLine("{0:d}", сумма);
    }
   }
```

1

2



Примечание 1,2: переменная может быть объявлена в самом операторе цикла и тогда её область видимости ограничена оператором или блоком цикла

Ещё один пример на foreach

```
using System;
class Массивы
{
   static void Main()
   {
      int[ ]  m2 = new int[ ] { 1, 1, 2, 2 }, m3 = { 1, 3, 5, 7 };
      int сумма = 0;
      foreach ( int j in m2 ) сумма += m3 [ j ];
      Console.WriteLine("{0:d}", сумма);
   }
}
```

16

В данном случае j – целочисленный и поэтому может быть использован в качестве индекса массива m3

Но при попытке изменить значение переменной j компилятором фиксируется ошибка

Инициализация массивов датчиком случайных чисел

```
using System;
class Массивы
{
  static void Main()
  {
    Random Gen = new Random();
    int[ ] m1 = new int[10];
    for (int i = 0; i < 10; i++) m1[ i ] = Gen.Next(100);
    int Счётчик = 0 ;
    foreach (int j in m1)
          if ( j % 2 == 0) Счётчик++;
    Console.WriteLine("Массив случайных значений");
    for (int i = 0; i < 10; i++ )
         Console.WriteLine("m1[{0}] = {1:d}", i, m1[ i ]);
    Console.WriteLine("Количество четных = {0}", Счётчик);
  }
```

```
Массив случайных значений
m1[0] = 24
m1[1] = 13
m1[2] = 48
m1[3] = 42
m1[4] = 8
m1[5] = 37
m1[6] = 20
m1[7] = 98
m1[8] = 22
m1[9] = 78
Количество четных = 8
```

```
Массив случайных значений
m1[0] = 19
m1[1] = 41
m1[2] = 4
m1[3] = 63
m1[4] = 20
m1[5] = 67
m1[6] = 9
m1[7] = 21
m1[8] = 14
m1[9] = 6
Количество четных = 4
```

Основные методы класса **Random** *(нестатические)*:

- **int Next ( )** возвращает очередное псевдослучайное целое число в диапазоне от *0 до 0x7FFFFFFF*;
- **int Next (int Max)** возвращает очередное псевдослучайное целое число в диапазоне от *0 до* ***Max***;
- **int Next (int Min, int Max)** возвращает очередное псевдослучайное целое число в диапазоне *от* ***Min*** *до* ***Max***;
- **double NextDouble ( )** возвращает очередное псевдослучайное вещественное число в диапазоне от *0,0 до 1,0*.

Экстремальные значения и статистические характеристики элементов массива

```
using System;
class Массивы
{   static void Main()
  {   string s;
      Console.WriteLine("Задайте количество элементов массива");
      s = Console.ReadLine ( );
      int i = Convert.ToInt32 ( s );
      int[ ]  mas = new int [ i ];
      Random Gen = new Random( );
      for (int k = 0; k < mas.Length; k++)
          mas[ k ] = Gen.Next(1,10);
      Console.WriteLine("Элементы массива\n");
      foreach ( int j in mas)
                Console.Write("{0,8}", j);
        int max = mas[ 0 ], min = mas [ 0 ];
        foreach ( int j in mas)  {   if (max < j) max = j;     if (min > j) min = j;   }
        Console.WriteLine ("\n Максимум= {0}, Минимум ={1} ", max, min);
      double  среднее = 0, дисперсия = 0;
      foreach (int j in mas)  среднее += j;
      среднее /= mas.Length;
      Console.WriteLine("\nСреднее арифметическое= {0:f5}",среднее);
        foreach ( int j  in  mas)  дисперсия +=  ( j - среднее ) * ( j - среднее );
        дисперсия /= mas.Length;
        Console.WriteLine("Дисперсия = {0:f5}", дисперсия);
        Console.WriteLine("Среднеквадратичное отклонение= {0:f5} ", Math.Sqrt(дисперсия));
  }
}
```

double Math.Sqrt(double d)
Возвращает квадратный корень из указанного числа

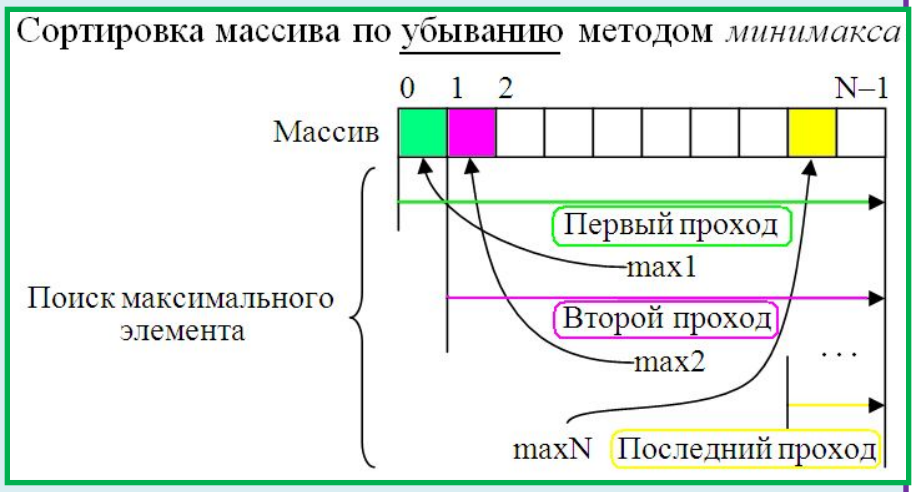
Вызов статического метода

```
Задайте количество элементов массива
20
Элементы массива

       9       2       7       9       3       4       1       9       2       7
       7       7       6       4       5       9       3       7       6       3

 Максимум= 9, Минимум =1

Среднее арифметическое= 5,50000
Дисперсия = 6,45000
Среднеквадратичное отклонение= 2,53968
```

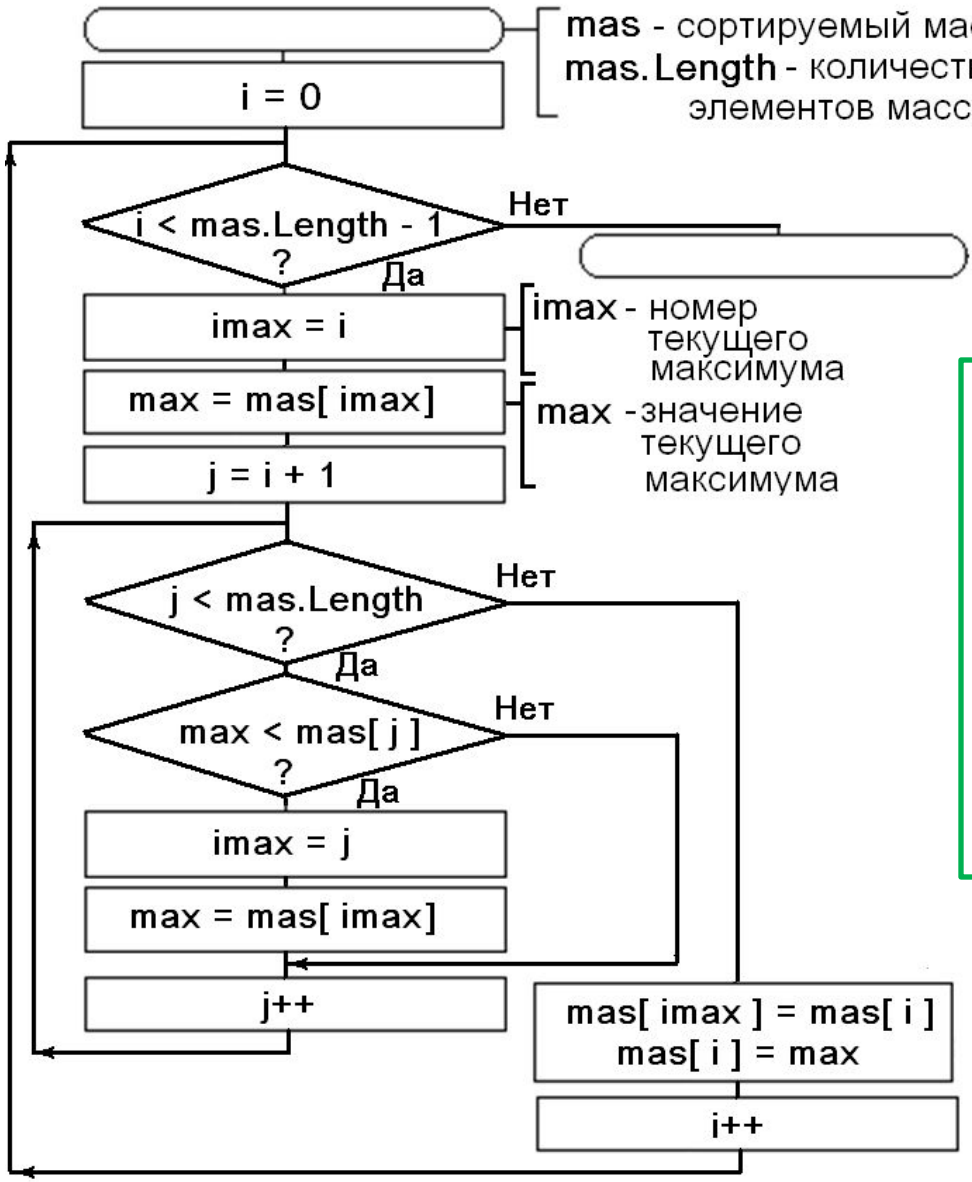
Сортировка одномерного массива

```
Задайте количество элементов массива
30
Элементы массива
      54       1      84      15      85      28       7      20      79      85
      11      42      85      97      86      92      33      71      43      47
      11      92      36      77       2      95      55      39      88      97

Элементы массива после сортировки по убыванию
      97      97      95      92      92      88      86      85      85      85
      84      79      77      71      55      54      47      43      42      39
      36      33      28      20      15      11      11       7       2       1
```

```
using System;
class Сортировка
{  static void Main()
   {string s;
    Console.WriteLine("Задайте количество элементов массива");
    s = Console.ReadLine ( );
    int k = Convert.ToInt32 ( s );
    int[ ] mas = new int [ k ];
    Random Gen = new Random( );
    for (int i = 0; i < mas.Length; i++)mas[ i ] = Gen.Next ( 1,100);
    Console.WriteLine ( "Элементы массива" );
    foreach (int j in mas) Console.Write( "{0,8}", j );
    int max , imax;
    for (int i = 0; i < mas.Length - 1; i++)
    {
       max = mas[ imax = i ];
       for ( int j = i + 1 ; j < mas.Length; j++)
          if (max < mas[ j ]) max = mas[imax = j];
       mas[ imax ] = mas[ i ];
       mas[ i ] = max;
    }
    Console.WriteLine ("\nЭлементы массива после сортировки по убыванию");
    foreach (int j in mas) Console.Write("{0,8}", j);
   }
}
```



Блок-схема сортировки методом минимакса




```
…
int max , imax;
for (int i = 0; i < mas.Length - 1; i++)
 {
   max = mas[ imax = i ];
   for ( int j = i + 1 ; j < mas.Length; j++)
      if (max < mas[ j ]) max = mas[imax = j];
   mas[ imax ] = mas[ i ];
   mas[ i ] = max;
 }
…
```