

Массивы

Массив—

- набор элементов одного и того же типа, объединенных общим именем.
- С#-массивы относятся к **ССЫЛОЧНЫМ** типам данных, реализованы как **объекты**.
- **Имя массива** является ссылкой на область кучи (динамической памяти), в которой последовательно размещается набор элементов определенного типа.
- **Выделение памяти** под элементы происходит на этапе инициализации массива.
- **Освобождение памяти** - система сборки мусора - неиспользуемые массивы автоматически утилизируются данной системой.

Одномерные массивы

- Одномерный массив – это фиксированное количество элементов одного и того же типа, объединенных общим именем, где каждый элемент имеет свой номер.
1. Объявляется ссылочная переменная на массив
 2. Выделяется память под требуемое количество элементов базового типа, и ссылочной переменной присваивается адрес нулевого элемента в массиве.

Объявление одномерного массива

Форма записи	Пояснения
<pre>базовый_тип [] имя_массива; <i>Например:</i> int [] a;</pre>	<p>Описана ссылка на одномерный массив, которая в дальнейшем может быть использована:</p> <ol style="list-style-type: none">1) для адресации на уже существующий массив;2) передачи массива в метод в качестве параметра3) отсроченного выделения памяти под элементы массива.
<pre>базовый_тип [] имя_массива = new базовый_тип [размер]; <i>Например:</i> int []a=new int [10];</pre>	<p>Объявлен одномерный массив заданного типа и выделена память под одномерный массив указанной размерности. Адрес данной области памяти записан в ссылочную переменную. Элементы массива равны нулю.</p> <p>Замечание. В C# элементам массива присваиваются начальные значения по умолчанию в зависимости от базового типа.</p> <p>Для арифметических типов – нули, для ссылочных типов – null, для символов - пробел.</p>
<pre>базовый_тип [] имя_массива={список инициализации}; <i>Например:</i> int []a={0, 1, 2, 3};</pre>	<p>Выделена память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации. Адрес этой области памяти записан в ссылочную переменную. Значение элементов массива соответствует списку инициализации.</p>

```
static void Main()
{
    int[] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    int i;
    for (i = 0; i < 10; ++i)
        Console.WriteLine(myArray[i]);
}
```

```
static void Main()
{
    int[] myArray = new int[10];
    int i;
    for (i = 0; i < 10; i++)
        myArray[i] = i * i;
    for (i = 0; i < 10; i++)
        Console.WriteLine(myArray[i]);
}
```

Существующей ссылке на одномерный массив присваивается ссылка на новый массив

```
int [ ] myArray = new int [ ] { 99, 10, 100, 18, 78, 23, 163, 9, 87, 49 };
```

```
static void Main()
```

```
{
```

```
int[] myArray = { 0, 1, 2, 3, 4, 5};
```

```
int i;
```

```
for (i = 0; i < 10; i++)
```

```
    Console.Write(" "+myArray[i]);
```

```
    Console.WriteLine("\nНовый массив: ");
```

```
    myArray = new int[] { 99, 10, 100, 18, 78, 23, 163, 9, 87, 49 };
```

```
    for (i = 0; i < 10; i++)
```

```
        Console.Write(" " + myArray[i]);
```

```
}
```

1. переменная myArray ссылалась на 6-ти элементный массив.

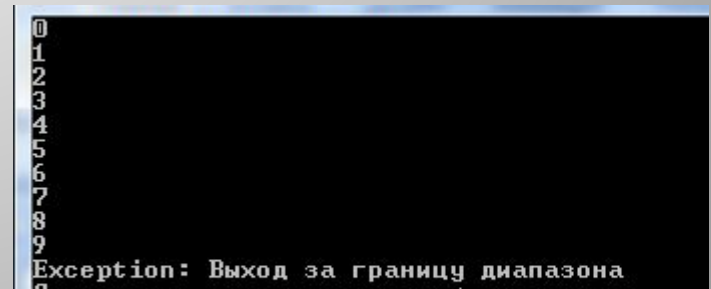
2. переменной myArray была присвоена ссылка на новый 10-элементный массив, в результате чего исходный массив оказался неиспользуемым, т.к. на него теперь не ссылается ни один объект.

3. он автоматически будет удален сборщиком мусора.

Массивы и исключения

```
static void Main()
{
    int[] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    int i;
    try
    {
        for (i = 0; i <= 10; i++) Console.WriteLine(myArray[i]);
    }
    catch (IndexOutOfRangeException)
    {
        Console.WriteLine("Exception: Выход за границу диапазона");
    }
}
```

Выход за границы массива в C# расценивается как ошибка, в ответ на которую генерируется исключение - `IndexOutOfRangeException`



Массив как параметр

- Так как имя массива фактически является ссылкой, то он передается в метод по ссылке
- Все изменения элементов массива, являющегося формальным параметром, отразятся на элементах соответствующего массива, являющимся фактическим параметром.

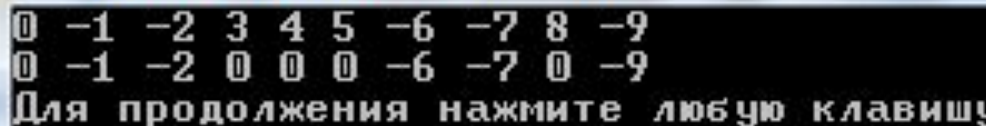

```

class Program
{
    static void Print(int n, int[] a) //n – размерность массива, a – ссылка на
    МАССИВ
    {
        for (int i = 0; i < n; i++) Console.Write("{0} ", a[i]);
        Console.WriteLine();    }

    static void Change(int n, int[] a)
    {
        for (int i = 0; i < n; i++)
            if (a[i] > 0) a[i] = 0; // изменяются элементы массива
    }

    static void Main()
    {
        int[] myArray = { 0, -1, -2, 3, 4, 5, -6, -7, 8, -9 };
        Print(10, myArray);
        Change(10, myArray);
        Print(10, myArray);    }
}

```



```

0 -1 -2 3 4 5 -6 -7 8 -9
0 -1 -2 0 0 0 -6 -7 0 -9
Для продолжения нажмите любую клавишу

```

Массив как объект

Элемент	Вид	Описание
Length	свойство	Количество элементов массива (по всем размерностям)
BinarySearch	статический метод	Двоичный поиск в отсортированном массиве
Clear	статический метод	Присваивание элементам массива значений по умолчанию
Copy	статический метод	Копирование заданного диапазона элементов одного массива в другой
CopyTo	экземплярный метод	Копирование всех элементов текущего одномерного массива в другой массив
GetValue	экземплярный метод	Получение значения элемента массива
IndexOf	статический метод	Поиск первого вхождения элемента в одномерный массив
LastIndexOf	статический метод	Поиск последнего вхождения элемента в одномерный массив
Reverse	статический метод	Изменение порядка следования элементов на обратный
SetValue	экземплярный метод	Установка значения элемента массива
Sort	статический метод	Упорядочивание элементов одномерного массива

- Вызов статических методов происходит через обращение к имени класса

Например:

*/*Обращение к статическому методу Sort класса Array и передача данному методу в качестве параметра объект myArray - экземпляр класса Array*/*

Array.Sort(myArray)

- Обращение к свойству или вызов экземплярного метода производится через обращение к экземпляру класса

Например:

myArray.Length

ИЛИ

myArray.GetValue(i)

```

class Program {
    static void Main() {
        try {
            int[] MyArray;
            Console.WriteLine("Введите размерность массива: ");
            int n = int.Parse(Console.ReadLine());
            MyArray = new int[n];
            for (int i = 0; i < MyArray.Length; ++i)
            {
                Console.WriteLine("a[{0}]=", i);
                MyArray[i] = int.Parse(Console.ReadLine()); }
            PrintArray("ИСХОДНЫЙ МАССИВ:", MyArray);
            Array.Sort(MyArray);
            PrintArray("массив отсортирован по возрастанию", MyArray);
            Array.Reverse(MyArray);
            PrintArray("массив отсортирован по убыванию", MyArray); }
        catch (FormatException) {
            Console.WriteLine("неверный формат ввода данных"); }
        catch (OverflowException)
        {
            Console.WriteLine("переполнение"); }
        catch (OutOfMemoryException) {
            Console.WriteLine("недостаточно памяти для создания нового объекта"); } }
    static void PrintArray(string a, int[] mas) { Console.WriteLine(a);
        for (int i = 0; i < mas.Length; i++) Console.WriteLine("{0} ", mas[i]);
        Console.WriteLine(); } } }

```

```

Введите размерность массива: i
неверный формат ввода данных

```

```

Введите размерность массива: 10000000000000000
переполнение

```

```

Введите размерность массива: 7
a[0]=1
a[1]=5
a[2]=66
a[3]=-8
a[4]=0
a[5]=-88
a[6]=9
исходный массив:
1 5 66 -8 0 -88 9
массив отсортирован по возрастанию
-88 -8 0 1 5 9 66
массив отсортирован по убыванию
66 9 5 1 0 -8 -88
Для продолжения нажмите любую клавишу . . .

```

Многомерные массивы

тип [,] имя__массива;

тип [,] имя__массива = new тип [размер1, размер2];

тип [,] имя__массива={{элементы 1-ой строки}, ... , {элементы n-ой строки}};

тип [,] имя__массива= new тип [,]{{элементы 1-ой строки}, ... , {элементы n-ой строки}};

Например:

int [,] a;

int [,] a= new int [3, 4];

int [,] a={{0, 1, 2}, {3, 4, 5}};

int [,] a= new int [,]{{0, 1, 2}, {3, 4, 5}};

```

class Program
{
    static void PrintArray(string a, int[,] mas)
    {
        Console.WriteLine(a);
        for (int i = 0; i < mas.GetLength(0); i++)
        {
            for (int j = 0; j < mas.GetLength(1); j++)
            {
                Console.Write("{0} ", mas[i, j]);
                Console.WriteLine();
            }
        }
    }

    static void Change(int[,] mas)
    {
        for (int i = 0; i < mas.GetLength(0); i++)
            for (int j = 0; j < mas.GetLength(1); j++)
                if (mas[i, j] % 2 == 0) mas[i, j] = 0;
    }

    static void Main()
    {
        try {
            int[,] MyArray = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
            PrintArray("ИСХОДНЫЙ МАССИВ:", MyArray);
            Change(MyArray);
            PrintArray("ИТОГОВЫЙ МАССИВ", MyArray);
        }
        catch (FormatException)
        {
            Console.WriteLine("неверный формат ввода данных");
        }
        catch (OverflowException)
        {
            Console.WriteLine("переполнение");
        }
        catch (OutOfMemoryException)
        {
            Console.WriteLine("недостаточно памяти для создания нового объекта");
        }
    }
}

```

```

ИСХОДНЫЙ МАССИВ :
1 2 3
4 5 6
7 8 9
ИТОГОВЫЙ МАССИВ
1 0 3
0 5 0
7 0 9
Для продолжения нажмите

```

Рваные массивы

тип[][] имя = new тип[размер][];

jagged[0][0]	jagged[0][1]	jagged[0][2]	jagged[0][3]	
jagged[1][0]	jagged[1][1]	jagged[1][2]		
jagged[2][0]	jagged[2][1]	jagged[2][2]	jagged[2][3]	jagged[2][4]

```
int [ ][ ] jagged = new int [ 3 ] [ ];  
jagged [0] = new int [ 4 ] ;  
jagged [1] = new int [ 3 ] ;  
jagged [2] = new int [ 5 ] ;
```

Так как каждая строка ступенчатого массива фактически является одномерным массивом, то с каждой строкой можно работать как с экземпляром класса Array.

Это является преимуществом ступенчатых массивов перед двумерными массивами.


```

class Program {
    static void Main() {
        try {
            int[][] MyArray; Console.WriteLine("Введите количество строк: ");
            int n = int.Parse(Console.ReadLine()); MyArray = new int[n][];
            for (int i = 0; i < MyArray.Length; i++)
            {
                Console.WriteLine("Введите количество элементов в {0} строке: ", i);
                int j = int.Parse(Console.ReadLine()); MyArray[i] = new int[j];
                for (j = 0; j < MyArray[i].Length; j++)
                {
                    Console.WriteLine("a[{0}][{1}] = ", i, j); MyArray[i][j] = int.Parse(Console.ReadLine()); }
                PrintArray("Исходный массив:", MyArray);
                for (int i = 0; i < MyArray.Length; i++) Array.Sort(MyArray[i]);
                PrintArray("Измененный массив", MyArray); }
            catch (FormatException)
            {
                Console.WriteLine("Неверный формат ввода данных"); }
            catch (OverflowException)
            {
                Console.WriteLine("Переполнение"); }
            catch (OutOfMemoryException)
            {
                Console.WriteLine("Недостаточно памяти для создания нового объекта"); }
        }
        static void PrintArray(string a, int[][] mas) {
            Console.WriteLine(a);
            for (int i = 0; i < mas.Length; i++)
            {
                for (int j = 0; j < mas[i].Length; j++) Console.WriteLine("{0} ", mas[i][j]); Console.WriteLine(); } } }
    }
}

```


C:\Windows\system32\cmd.exe

```
Введите количество строк: 5
введите количество элементов в 0 строке: 2
a[0][0]= 1
a[0][1]= 2
введите количество элементов в 1 строке: 3
a[1][0]= 1
a[1][1]= 2
a[1][2]= 3
введите количество элементов в 2 строке: 8
a[2][0]= 5
a[2][1]= 6
a[2][2]= 7
a[2][3]= 8
a[2][4]= 9
a[2][5]= 10
a[2][6]= 11
a[2][7]= 12
введите количество элементов в 3 строке: 2
a[3][0]= 4
a[3][1]= 5
введите количество элементов в 4 строке: 4
a[4][0]= 2
a[4][1]= 0
a[4][2]= 2
a[4][3]= 0
исходный массив :
1 2
1 2 3
5 6 7 8 9 10 11 12
4 5
2 0 2 0
измененный массив
1 2
1 2 3
5 6 7 8 9 10 11 12
4 5
0 0 2 2
Для продолжения нажмите любую клавишу . . .
```

Оператор foreach

- Применяется для перебора элементов в специальном образом организованной группе данных, в том числе и в массиве.
- Удобство заключается в том, что не требуется определять количество элементов в группе и выполнять перебор по индексу – просто указываем на необходимость перебрать все элементы группы.

foreach (<тип> <имя> in <группа>) <тело цикла>

где *имя* определяет локальную по отношению к циклу переменную, которая будет по очереди принимать все значения из указанной *группы*, а *тип* соответствует базовому типу элементов *группы*.

Ограничение: с его помощью **можно** только **просматривать** значения элементов в группе данных, но **нельзя их изменять**.

Использование оператора foreach

1) для работы с одномерными массивами:

```
static void PrintArray(string a, int [] mas)
{
    Console.WriteLine(a);
    foreach (int x in mas) Console.Write("{0} ", x);
    Console.WriteLine();    }
```

2) для работы с двумерными массивами:

```
static int Sum (int [,] mas)
{
    int s=0;
    foreach (int x in mas) s += x;
    return s;    }
```

3) для работы со ступенчатыми массивами:

```
static void PrintArray3(string a, int[][] mas)
{
    Console.WriteLine(a);
    for (int i = 0; i < mas.Length; i++)
    {
        foreach (int x in mas[i]) Console.Write("{0} ", x);
        Console.WriteLine();    }    }
```