

Массивы

Занятие № 4

Массивы

- **Массивом** называется ограниченная совокупность однотипных перенумерованных элементов
- Элементы массива имеют одно и то же **имя**, а различаются **порядковым номером (индексом)**
- Элементы массива нумеруются с **нуля**

Массивы

- Массив характеризуется своим рангом или размерностью, определяющей **число индексов**, которые необходимо указать для доступа к его элементу
- Массив, имеющий ранг, равный единице, называется **одномерным массивом**
- Все массивы, имеющий ранг больше единицы, называются **многомерными массивами**

Массивы

- Массив относится к **ССЫЛОЧНЫМ** типам данных, то есть располагается в динамической области памяти, поэтому создание массива начинается с **выделения памяти** под его элементы
- Элементами массива могут быть величины как **значимых**, так и **ССЫЛОЧНЫХ** типов (в том числе массивы)

Элементы массива не нужно
инициализировать, так как они
инициализируются автоматически по
следующему правилу:

- Если создается массив из чисел, то все его элементы инициализируются **нулями**;
- Если массив содержит элементы логического типа, то все элементы инициализируются в **false**;
- Все ссылочные типы инициализируются **null**.

Одномерные массивы

- Объявление массива:

тип [] имя;

тип [] имя = new тип [размерность];

тип [] имя = {список инициализаторов};

тип [] имя = new тип [] {список
инициализаторов};

Одномерные массивы

- Объявление (примеры):

```
int [ ] a;
```

```
int [ ] b = new int[4];
```

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

```
int [ ] d = new int [ ] {1, 2, 3, 4, 5};
```

- Доступ к элементам:

```
int x = a[0];
```

- Нумерация индексов от 0 до N - 1, где N - размер массива

Работа с массивами: заполнение

```
int [ ] a = new int[7];  
for (int i=0; i<7; i++)  
{  
    a[i] = i*i;  
}
```

Работа с массивами: вычисление суммы элементов

```
int sum = 0;
for (int i=0; i<7; i++)
{
    sum+=a[i];
}
Console.WriteLine("Сумма элементов "+sum);
```

Работа с массивами: вывод массива на экран

```
for (int i=0; i<7; i++)  
{  
    Console.WriteLine( a[i] );  
}
```

Многомерные массивы

- Элементы многомерных массивов идентифицируются набором индексов - "координат" в многомерном пространстве
- Объявление:
`int [,] a0;`
`int [, ,] a1;`
`int [][][] a2;`
- Доступ к элементам:
`int e = a1[0, 1, 2];`

Прямоугольные массивы

- Объявление массива:

тип [,] имя;

тип [,] имя = new тип [разм_1, разм_2];

тип [,] имя = {список инициализаторов};

тип [,] имя = new тип [,] {список
инициализаторов};

Прямоугольные массивы

- Объявление (примеры):

```
int [ , ] a;
```

```
int [ , ] b = new int[2,3];
```

```
int [ , ] c = {{1, 2, 3},{ 4, 5,6}};
```

```
int [ , ] d = new int [2,3 ] {{1, 2, 3},{ 4, 5,6}};
```

- Доступ к элементам:

```
int x = c[0,2];
```

Неровные (ступенчатые) массивы

Неровные массивы – это массивы массивов (ссылок на массивы)

Объявление :

```
int [ ] [ ] a = new int[2][ ];
```

```
a[0] = new int[4];
```

```
a[1] = new int[20];
```

Доступ:

```
int element = a[0][1];
```

! Размеры неровных массивов могут различаться даже в одном измерении. В приведенном примере существует элемент `a[1][15]`, но не существует элемента `a[0][15]`

Оператор цикла *foreach*

Этим оператором обеспечивается повторение множества операторов, составляющих тело цикла, для **каждого** элемента массива или коллекции. После перебора **ВСЕХ** элементов массива или коллекции и выполнения тела цикла для каждого элемента массива или коллекции, управление передаётся следующему оператору.

Оператор цикла *foreach*

`foreach` (тип имя `in` выражение) оператор;

Пример:

```
int [ ] c = {1, 2, 3, 4, 5}; // Объявили и  
    определили массив  
foreach (int i in c)  
    Console.WriteLine( i );
```

Класс **Random**

Класс **Random** определен в пространстве имен **System** и предназначен для генерации исходных данных, заданных случайным образом.

Для получения псевдослучайной последовательности необходимо сначала создать экземпляр класса **Random**.

Пример

```
Random a = new Random();
```

```
Random b = new Random (1);
```

Конструктор без параметров использует начальное значение генератора, вычисленное на основе текущего времени.

Конструктор с параметром типа `int` задает начальное значение генератора, что обеспечивает возможность получения одинаковых последовательностей чисел.

Основные методы

Название	Описание
Next()	Возвращает целое положительное число во всем положительном диапазоне int
Next(max)	Возвращает целое положительное число в диапазоне [0, max]
Next(min,max)	Возвращает целое положительное число в диапазоне [min, max]
NextBytes(массив)	Возвращает массив чисел в диапазоне [0, 255]
NextDouble()	Возвращает вещественное положительное число в диапазоне [0, 1)

Пример

```
Random a = new Random();  
int[] mas1 = new int[10];  
byte[] mas2 = new byte[10];  
double[] mas3 = new double[10];
```

```
Console.WriteLine("Массив целых чисел в  
диапазоне -50..50");  
for (int i = 0; i < 10; i++)  
    mas1[i] = (a.Next(-50,50));  
foreach (int m in mas1)  
    Console.Write("{0} ", m);  
Console.WriteLine();
```

```
Console.WriteLine("Массив целых чисел в  
диапазоне 0..255");  
a.NextBytes(mas2);  
foreach (int m in mas2)  
    Console.Write("{0} ", m);  
Console.WriteLine();
```

```
Console.WriteLine("Массив вещественных чисел в  
диапазоне -50..50");  
for (int i = 0; i < 10; i++)  
    mas3[i] = (a.NextDouble() - 0.5) * 100;  
foreach (double m in mas3)  
    Console.Write("{0:00.00} ", m);  
Console.WriteLine();
```

Результат

```
D:\WINDOWS\system32\cmd.exe
Массив целых чисел в диапазоне -50..50
19 33 -6 30 -28 40 7 48 0 45
Массив целых чисел в диапазоне 0..255
111 212 131 86 78 188 109 27 211 88
Массив вещественных чисел в диапазоне -50..50
-41,87 30,58 -09,69 46,08 07,44 -30,02 06,47 -06,68 -19,51 -01,56
Press any key to continue . . .
```

Класс **System.Array**

Все массивы в C# построены на основе базового класса **System.Array**, который содержит полезные для программиста свойства и методы.

Свойство Length

Свойство **Length** определяет количество элементов массива (по всем размерностям)

Пример:

```
for (int i=0; i< m.Length; i++)  
{  
    Console.WriteLine(m[i]);  
}
```

Свойство Rank

Свойство **Rank** показывает размерность массива

```
int [,] mas = new int [2,3];  
Console.WriteLine(mas.Rank);
```

Результат: 2

Метод Clear()

Метод `Clear()` позволяет очистить указанный диапазон элементов (числовые элементы приобретут значения 0, ссылки на объекты — `null`, логические элементы - `false`).

Первым параметром этого метода является имя массива, вторым — индекс, с которого происходит очистка, третьим — число элементов.

```
int [ ] c = {1, 2, 3, 4, 5};  
Array.Clear(c, 0, c.Length);
```

Метод `GetLength()`

Метод `GetLength()` используется для определения количества элементов в указанном измерении массива.

```
int [,] c = {{1,2,3},{4,5,6}};  
int dim0 = c.GetLength(0); // == 2  
int dim1 = c.GetLength(1); // == 3
```

Метод IndexOf()

Метод IndexOf() возвращает номер первого вхождения указанного элемента. Если элемент не найден, то возвращается -1.

```
int [ ] c = {12, 32, 3, 54, 15, 6};  
int w = Array.IndexOf(c, 54); //==3
```

Метод LastIndexOf()

Метод LastIndexOf() возвращает номер последнего вхождения указанного элемента. Если элемент не найден, то возвращается -1.

```
int [ ] c = {12, 32, 3, 54, 12, 6};  
int w = Array.LastIndexOf(c, 12);  
//==4
```

Метод Sort()

Метод Sort() сортирует одномерный массив встроенных типов данных, причем массив передается как параметр.

```
int [ ] c = {12, 32, 3, 54, 15, 6};  
Array.Sort(c);
```

Результат :

3 6 12 15 32 54

Метод Reverse()

Метод Reverse() позволяет расставить элементы одномерного массива в обратном порядке, причем массив передается как параметр.

```
int [ ] c = {12, 32, 3, 54, 15, 6};  
Array.Reverse(c);
```

Результат :

6 15 54 3 32 12

Метод BinarySearch()

Метод BinarySearch() выполняет двоичный поиск в отсортированном массиве.
Возвращает индекс элемента.

```
int[] c = { 12, 32, 3, 54, 15, 6 };
```

```
Array.Sort(c);
```

```
Console.WriteLine(Array.BinarySearch(c, 12));
```

Метод CopyTo()

Метод CopyTo() используется для копирования элементов из исходного массива в массив назначения.

```
int[ ] c = { 12, 32, 3, 54, 15, 6 };
```

```
int[ ] d = new int[6];
```

```
c.CopyTo(d, 0);
```

Метод Copy()

Метод Copy() используется для копирования заданного диапазона элементов из исходного массива в массив назначения.

```
int[] c = { 12, 32, 3, 54, 15, 6 };
```

```
int[] d = new int[6];
```

```
Array.Copy(c, d, 3); // 12 32 3 0 0 0
```

```
Array.Copy(c, 1, d, 2, 3); // 0 0 32 3 54 0
```