Массивы в C#

Массивы в C#

- Массивы в C# с точки зрения синтаксиса практически не отличаются от массивов в C, C++ и Java.
- Однако внутренне массив в C# устроен как тип, производный от класса System.Array.
- Формально массив определяется как набор элементов, доступ к которым производится с помощью числового индекса.

Объявление массивов

Объявление одномерных массивов

<тип>[] <объявители>;

Пример:

int[] k;

При объявлении с отложенной инициализацией сам массив не формируется, а создается только ссылка на массив, имеющая неопределенное значение Null.

double[] x= {5.5, 6.6, 7.7}; //явная инициализация с константным массивом

int[] d= new int[5]; // создание и инициализация массива выполняется в объектном стиле с вызовом конструктора массива

Массив символьных строк с 10 элементами {0, 1,…, 9}

```
string[] booksOnCOM;
booksOnCOM = new string[10];
```

Массив символьных строк с 2 //элементами {0, 1}

```
string[] booksOnPLI = new string[2];
```

Массив символьных строк из 100 //элементов {0, 1,…, 99}

```
string[] booksOnDotNet = new string[100];
```

Замечание

- Если массив объявляется без инициализации, то создается только висячая ссылка со значением void.
- Если инициализация выполняется конструктором, то в динамической памяти создается сам массив, элементы которого инициализируются константами соответствующего типа (ноль для арифметики, пустая строка для строковых массивов), и ссылка связывается с этим массивом.
- Если массив инициализируется константным массивом, то в памяти создается константный массив, с которым и связывается ссылка.

- Нумерация элементов массива **идет с нуля.**
- Таким образом в

int[] k = new int [3];

- начальный элемент массива – это k[0], а последний – k[2].
- Элемента k[3]нет.

//Задаем элементы массива

- k[0]=-5;
- k[1]=4;
- k[2]=55;

```
Console.WriteLine("сколько элементов? ");
n = Convert.ToInt32(Console.ReadLine());
a=new int [n];
for (int i = 0; i < n; i++)
{
Console.WriteLine(" элемент # "+i);
a[i] = Convert.ToInt32(Console.ReadLine());
}
```

Массив из случайных элементов

```
int[] a;
int n;
Console.WriteLine("сколько элементов? ");
n = Convert.ToInt32(Console.ReadLine());
a=new int [n];
Random r = new Random();
for (int i = 0; i < n; i++)
    a[i] = r.Next(10, 100);

foreach (int k in a) Console.Write(k + " ");
```

```
for(int i=0; i<3; i++)    Console.Write(k[i]+" ");
```

```
//Выводим третий элемент массива
Console.WriteLine(k[2].ToString());
```

Вывод массива

- foreach (int i in k)
  {
  Console.WriteLine(i.ToString());
  }
  }

```
foreach (int k in a) Console.WriteLine(k);
for (int i = 0; i < n; i++) Console.WriteLine(a[i]);
```

- String[] a = { "red", "GREEN", "YELLOW", "BLUE", "purple", "black", "orange" };

Важное различие между массивами C++ и C#

- В C# элементам массива автоматически присваиваются значения по умолчанию в зависимости от используемого для них типа данных.

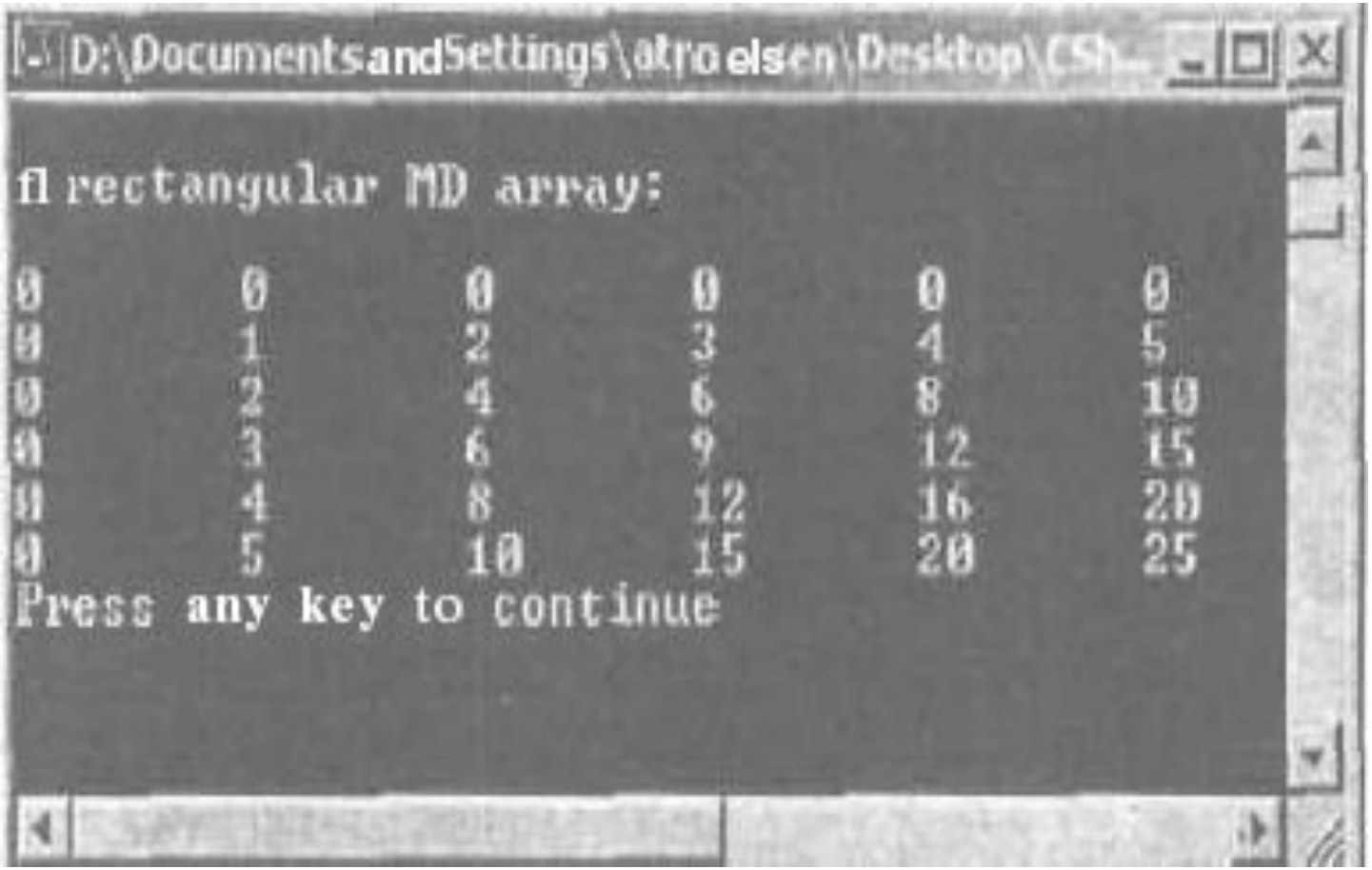
  Например, для массива целых чисел всем элементам будет изначально присвоено значение 0, для массива объектов — значение NULL и т. д.

Многомерные массивы

- Помимо массивов с одним измерением в C# поддерживаются также две основные разновидности многомерных массивов.
- Первую разновидность многомерных массивов иногда называют “прямоугольным массивом”. Такой тип массива образуется простым сложением нескольких измерений. При этом все строки и столбцы в данном массиве будут одинаковой длины.

```
// Прямоугольный многомерный массив
int[] myMatrix;
myMatrix = new int[6, 6];
// Заполняем массив 6 на 6
for (int i = 0; i < 6; i++)
    for (int j = 0; j < 6; j++)
        myMatrix[i, j] = i*j;
// Выводим элементы многомерного массива на
//системную консоль
for (int i = 0 ; i < 6; i++)
{
    for (int j = 0; j < 6; j++)
    {
        Console.Write(myMatrix[i, j] +"\t");
    }
    Console.WriteLine();
}
```

Результат работы программы



многомерные массивы

int[,] k = new int [2,3];

Обратите внимание, что пара квадратных скобок только одна.

Естественно, что в нашем примере у массива 6 (=2*3) элементов (k[0,0] – первый, k[1,2] – последний).

- пример трехмерного массива:
- **int[,,] k = new int [10,10,10];**

- А вот так можно сразу инициализировать многомерные массивы:
- int[,] k = {{2,-2},{3,-22},{0,4}};

```
int[,] a;
int n,m;
Console.WriteLine("сколько столбцов? ");
m = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("сколько строк? ");
n = Convert.ToInt32(Console.ReadLine());
a=new int [m,n];
Random r = new Random();
for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
    a[i,j] = r.Next(10, 100);

foreach (int k in a) Console.Write(k + " ");
Console.WriteLine();
for (int j = 0; j < n; j++,  Console.WriteLine())
    for (int i = 0; i < m; i++) Console.Write(a[i,j] + " ");
Console.ReadLine();
```

Функции класса System.Array

- Массивы в C# основаны на классе System.Array. У этого класса, как и у любого другого, есть разные полезные методы.

Таблица

Член класса	Назначение
BinarySearch()	Этот статический метод можно использовать только тогда, когда массив реали-зует интерфейс IComparer, в этом случае метод позволяет найти элемент массива.
Clear()	Этот статический метод позволяет очистить диапазон

	элементов (числовые элементы приобретут значения 0, а ссылки на на объекты – null)
CopyTo()	Используется для копирования элементов из исходного массива в массив назначения
GetEnumerato()	Возвращает интерфейс IEnumerator для указанного массива.

GetLengt().Lenght	Метод GetLength() используется для определения количества элементов в указанном измерении массива. Length - это свойство только для чтения, с помощью которого можно получить количество элементов массива
GetLowerBound() GetUpperBound()	Эти методы используются для определения верхней и нижней границы

	выбранного вами измерения массива.
GetValue() SetValue()	Возвращает или устанавливает значение указанного индекса для массива. Этот метод перегружен для нормальной работы как с одномерными, так и с многомерными массивами
Reverse()	Этот статический метод позволяет расставить

	элементы одномерного массива в обратном порядке.
Sort()	Сортирует одномерный массив встроенных типов данных. Если элементы массива поддерживают интерфейс IComparer, то с помощью этого метода вы сможете производить сортировку и ваших пользовательских типов данных.

- BinarySearch(Array, Object)
- Выполняет поиск заданного элемента во всем отсортированном одномерном массиве **Array**, используя для этого интерфейс IComparable, реализуемый каждым элементом массива **Array** и заданным объектом.

Clear

Задает диапазон элементов массива **Array** равным нулю, **false** или **Nothing** в зависимости от типа элемента.

```
Array.Clear(num, 0, 5);
Console.WriteLine("Обнуленный массив");
foreach (int i in num)
{ Console.WriteLine(i.ToString()); }
```

- Copy(Array, Array, Int32)
- Копирует диапазон элементов из массива **Array**, начиная с первого элемента, и вставляет его в другой массив **Array**, также начиная с первого элемента. Длина задается как 32-разрядное целое число.

- Find(Of T)
- Выполняет поиск элемента, удовлетворяющего условиям указанного предиката, и возвращает первое найденное вхождение в пределах всего списка **Array**.

IndexOf

Статический метод IndexOf предназначенный для поиска элемента в массиве

```
int k=-5;
Console.WriteLine("Число {0} находится на {1} месте.", k, Array.IndexOf(num, k));
```

Этот метод возвращает индекс искомого элемента (нумерация с нуля). Если такого элемента нет, то возвращается -1.

```
int[] a;
int n;
Console.WriteLine("сколько элементов? ");
n = Convert.ToInt32(Console.ReadLine());
a=new int [n];
Random r = new Random();
for (int i = 0; i < n; i++)
    a[i] = r.Next(10, 100);

foreach (int k in a) Console.Write(k + " ");
Console.WriteLine();

int t;
Console.WriteLine("введи число для поиска ");
t=Convert.ToInt32(Console.ReadLine());
if (Array.IndexOf(a, t) != -1)
    Console.WriteLine("Число {0} находится на {1} месте.", t, Array.IndexOf(a, t));
else Console.WriteLine("Число {0} нет в массиве", t);
```

- FindAll(Of T)
- Извлекает все элементы, удовлетворяющие условиям указанного предиката.

- Resize(Of T)
- Изменяет количество элементов в массиве до указанной величины.

Reverse(Array)

Изменяет порядок элементов во всем одномерном массиве **Array** на обратный.

```
Console.WriteLine("Перевернутый массив");
Array.Reverse(num);
foreach (int i in num)
 {       Console.WriteLine(i.ToString());        }
```

Sort(Array)

Сортирует элементы во всем одномерном массиве **Array**, используя реализацию интерфейса IComparable каждого элемента массива **Array**.

```
Array.Sort(num);
Console.WriteLine("Отсортированный массив");
foreach (int i in num)
{           Console.WriteLine(i.ToString());          }
```

Sort(Array, Array)

Сортирует пару одномерных объектов **Array** (один содержит ключи, а другой — соответствующие элементы) по ключам в первом массиве **Array**, используя реализацию интерфейса IComparable каждого ключа.

http://msdn.microsoft.com/ru-ru/library/85y6y2d3.aspx