Основы программирования

Самойлов Михаил Юрьевич

Массивы

Массив представляет набор однотипных переменных. Объявление массива похоже на объявление переменной за тем исключением, что после указания типа ставятся квадратные скобки:

тип_переменной[] название_массива;

Массивы

Например, определим массив целых чисел:

```
int[] numbers;
```

После определения переменной массива мы можем присвоить ей определенное значение:

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

Здесь вначале мы объявили массив nums, который будет хранить данные типа int. Далее используя операцию new, мы выделили память для 4 элементов массива: new int[4]. Число 4 еще называется длиной массива. При таком определении все элементы получают значение по умолчанию, которое предусмотренно для их типа. Для типа int значение по умолчанию - 0.

Массивы

Также мы сразу можем указать значения для этих элементов:

```
int[] nums2 = new int[4] { 1, 2, 3, 5 };
int[] nums3 = new int[] { 1, 2, 3, 5 };
int[] nums4 = new[] { 1, 2, 3, 5 };
int[] nums5 = { 1, 2, 3, 5 };
```

Все перечисленные выше способы будут равноценны.

Элементы массива

Для обращения к элементам массива используются индексы. Индекс представляет номер элемента в массиве, при этом нумерация начинается с нуля, поэтому индекс первого элемента будет равен 0. А чтобы обратиться к четвертому элементу в массиве, нам надо использовать индекс 3, к примеру: nums[3]. Используем индексы для получения и установки значений элементов массива:

Элементы массива

Элементы массива

```
int[] nums = new int[4];
nums[0] = 1;
nums[1] = 2;
nums[2] = 3;
nums[3] = 5;
Console.WriteLine(nums[3]);  // 5
```

И так как у нас массив определен только для 4 элементов, то мы не можем обратиться, например, к шестому элементу: nums[5] = 5;. Если мы так попытаемся сделать, то мы получим исключение IndexOutOfRangeException.

Свойство Length

Все массивы являются объектами и у них есть некоторые свойства. Самым полезным для нас будет свойство Length, которое возвращает количество элементов в массиве (во всех размерностях)

```
int[] numbers = new int[5];
int size = numbers.Length; // size = 5
```

Перебор массивов. Цикл foreach

Цикл foreach предназначен для перебора элементов в контейнерах, в том числе в массивах. Формальное объявление цикла foreach:

```
foreach (тип_данных название_переменной in контейнер)
{
    // действия
}
```

Перебор массивов. Цикл foreach

Например:

```
int[] numbers = new int[] { 1, 2, 3, 4, 5 };
foreach (int i in numbers)
{
    Console.WriteLine(i);
}
```

Здесь в качестве контейнера выступает массив данных типа int. Поэтому мы объявляем переменную с типом int

Перебор массивов. Цикл for

Подобные действия мы можем сделать и с помощью цикл for:

```
int[] numbers = new int[] { 1, 2, 3, 4, 5 };
for (int i = 0; i < numbers.Length; i++)
{
    Console.WriteLine(numbers[i]);
}
```

Перебор массивов. Цикл for

Цикл for более гибкий по сравнению с foreach. Если foreach последовательно извлекает элементы контейнера и только для чтения, то в цикле for мы можем перескакивать на несколько элементов вперед в зависимости от приращения счетчика, а также можем изменять элементы:

```
int[] numbers = new int[] { 1, 2, 3, 4, 5 };
for (int i = 0; i < numbers.Length; i++)
{
    numbers[i] = numbers[i] * 2;
    Console.WriteLine(numbers[i]);
}
```

Задачи с массивами

Рассмотрим пару задач для работы с массивами.

Найдем количество положительных чисел в массиве:

```
int[] numbers = { -4, -3, -2, -1, 0, 1, 2, 3, 4 };
int result = 0;
foreach(int number in numbers)
{
    if(number > 0)
    {
        result++;
    }
}
Console.WriteLine(result);
```

Задачи с массивами

Вторая задача - инверсия массива, то есть переворот его в обратном порядке:

```
int[] numbers = { -4, -3, -2, -1,0, 1, 2, 3, 4 };

int n = numbers.Length; // длина массива
int k = n / 2;          // середина массива
int temp;               // вспомогательный элемент для обмена значениями
```

Задачи с массивами

Вторая задача - инверсия массива, то есть переворот его в обратном порядке:

```
for(int i=0; i < k; i++)
{
    temp = numbers[i];
    numbers[i] = numbers[n - i - 1];
    numbers[n - i - 1] = temp;
}
```

Задачи с массивами

Вторая задача - инверсия массива, то есть переворот его в обратном порядке:

```
foreach(int i in numbers)
{
    Console.WriteLine(i);
}
```

Задачи с массивами

Вторая задача - инверсия массива, то есть переворот его в обратном порядке:

Поскольку нам надо изменять элементы массива, то для этого используется цикл for. Алгоритм решения задачи подразумевает перебор элементов до середины массива, которая в программе представлена переменной k, и обмен значений элемента, который имеет индекс i, и элемента с индексом n-i-1.

Array

Все массивы в C# построены на основе класса **Array** из пространства имен System. Этот класс определяет ряд свойств и методов, которые мы можем использовать при работе с массивами. Основные свойства и методы:

- Свойство **Length** возвращает длину массива
- Свойство **Rank** возвращает размерность массива
- Статический метод **BinarySearch()** выполняет бинарный поиск в отсортированном массиве
- Статический метод **Clear()** очищает массив, устанавливая для всех его элементов значение по умолчанию
- Статический метод **Copy()** копирует часть одного массива в другой массив
- Статический метод **Exists()** проверяет, содержит ли массив определенный элемент

Array

Все массивы в C# построены на основе класса **Array** из пространства имен System. Этот класс определяет ряд свойств и методов, которые мы можем использовать при работе с массивами. Основные свойства и методы:

- Статический метод **Find()** находит элемент, который удовлеворяет определенному условию
- Статический метод **FindAll()** находит все элементы, которые удовлеворяет определенному условию
- Статический метод **IndexOf()** возвращает индекс элемента
- Статический метод **Resize()** изменяет размер одномерного массива
- Статический метод **Reverse()** располагает элементы массива в обратном порядке
- Статический метод **Sort()** сортирует элементы одномерного массива

Array

Разберем самые используемые методы. Например, изменим порядок элементов:

```
int[] numbers = { -4, -3, -2, -1,0, 1, 2, 3, 4 };

// расположим в обратном порядке
Array.Reverse(numbers);

foreach(int number in numbers)
{
    Console.Write(number);
}
```

Array

Метод **Copy** копирует часть одного массива в другой:

```
int[] numbers = { -4, -3, -2, -1,0, 1, 2, 3, 4 };
int[] numbers2 = new int[5];

// копируем из numbers с 2-го индекса 5 элементов
// и поместим их в массив numbers2, начиная с 0-го индекса
Array.Copy(numbers, 2, numbers2, 0, 5);

foreach(int number in numbers2)
{
    Console.Write(number);
}
```

Array

Отсортируем массив с помощью метода **Sort()**:

```
int[] numbers = { -3, 10, 0, -5, 12, 1, 22, 3};

Array.Sort(numbers);

foreach(int number in numbers)
{
    Console.WriteLine(number);
}
```

Задачи

- Задать массив из N случайных чисел, вывести на экран элементы массива через пробел.
- Задать массив из N чисел с клавиатуры, переставить местами первый и последний элемент и вывести элементы массива на экран.
- Задать массив из N случайных чисел, вывести на экран в первой строке положительные элементы массива через пробел, во второй отрицательные.
- Задать массив из N случайных чисел, вывести на экран элементы массива через пробел, найти максимальный элемент и сколько раз он встречается.
- Задать массив из N случайных чисел, вывести на экран элементы массива через пробел, найти количество положительных и отрицательных чисел.