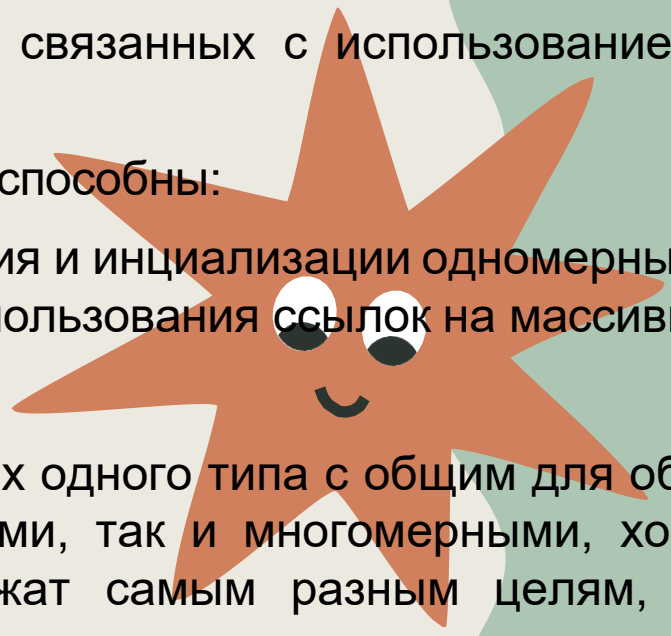
Лекция 6. Одномерные массивы

Цель лекции: Формирование у студентов знаний, связанных с использованием одномерных массивов.

В результате изучения данной лекции студенты будут способны:

- Продемонстрировать знание синтаксиса объявления и инциализации одномерных массивов
- Продемонстрировать понимание особенностей использования ссылок на массивы



Массивы

Массив представляет собой совокупность переменных одного типа с общим для обращения к ним именем. В C# массивы могут быть как одномерными, так и многомерными, хотя чаще всего применяются одномерные массивы. Массивы служат самым разным целям, поскольку они предоставляют удобные средства для объединения связанных вместе переменных. Например, в массиве можно хранить максимальные суточные температуры, зарегистрированные в течение месяца, перечень биржевых курсов или же названия книг по программированию из домашней библиотеки.

Главное преимущество массива — в организации данных таким образом, чтобы ими было проще манипулировать. Так, если имеется массив, содержащий дивиденды, выплачиваемые

по определенной группе акций, то, организовав циклическое обращение к элементам этого массива, можно без особого труда рассчитать средний доход от этих акций. Кроме того, массивы позволяют организовать данные таким образом, чтобы легко отсортировать их.

Массивами в C# можно пользоваться практически так же, как и в других языках программирования. Тем не менее у них имеется одна особенность: они реализованы в виде объектов

Одномерные массивы

Одномерный массив представляет собой список связанных переменных. Такие списки часто применяются в программировании. Например, в одномерном массиве можно хранить учетные номера активных пользователей сети или текущие средние уровни достижений бейсбольной команды.

Для того чтобы воспользоваться массивом в программе, требуется двухэтапная процедура, поскольку в C# массивы реализованы в виде объектов. Во-первых, необходимо объявить переменную, которая может обращаться к массиву. И во-вторых, нужно создать экземпляр массива, используя оператор new. Так, для объявления одномерного массива обычно применяется следующая общая форма:

тип[] *имя_массива* = new *тип*[*размер*];

где *тип* объявляет конкретный тип элемента массива. Тип элемента определяет тип данных каждого элемента, составляющего массив. Обратите внимание на квадратные скобки, которые сопровождают *тип.* Они указывают на то, что объявляется одномерный массив. А *размер* определяет число элементов массива.

Обратимся к конкретному примеру. В приведенной ниже строке кода создается массив типа int, который составляется из десяти элементов и связывается с переменной ссылки на массив, именуемой sample.

```
int[] sample = new int[10];
```

В переменной sample хранится ссылка на область памяти, выделяемой для массива оператором new. Эта область памяти должна быть достаточно большой, чтобы в ней могли храниться десять элементов массива типа int.

Как и при создании экземпляра класса, приведенное выше объявление массива можно разделить на два отдельных оператора. Например:

```
int[] sample;

sample = new int[10];
```

В данном случае переменная sample не ссылается на какой-то определенный физический объект, когда она создается в первом операторе. И лишь после выполнения второго оператора эта переменная ссылается на массив.

Доступ к отдельному элементу массива осуществляется по индексу: *Индекс* обозначает положение элемента в массиве. В языке C# индекс первого элемента всех массивов оказывается нулевым. В частности, массив sample состоит из 10 элементов с индексами от 0 до 9. Для индексирования массива достаточно указать номер требуемого элемента в квадратных скобках. Так, первый элемент массива sample обозначается как sample [0], а последний его элемент — как sample[9]. Ниже приведен пример программы, в которой заполняются все 10 элементов массива sample.

```
// Продемонстрировать одномерный массив.
using System;
class ArrayDemo {
static void Main() {
int[] sample = new int[10];
int i;
for(i = 0; i < 10; i = i+1)
sample[i] = i;
for(i = 0; i < 10; i = i+1)
Console.WriteLine("sample[" + i + "]: " + sample[i]);
}
}
```

При выполнении этой программы получается следующий результат.

sample[0]: 0

sample[1]: 1

sample[2]: 2

sample[3]: 3

sample[4]: 4

sample[5]: 5

sample[6]: 6

sample[7]: 7

sample[8]: 8

sample[9]: 9

Схематически массив sample можно представить таким образом.

Массивы часто применяются в программировании потому, что они дают возможность легко обращаться с большим числом взаимосвязанных переменных. Например, в приведенной ниже программе выявляется среднее арифметическое ряда значений, хранящихся в массиве nums, который циклически опрашивается с помощью оператора цикла for.

```
// Вычислить среднее арифметическое ряда значений.

using System;

class Average {

static void Main() {

int[] nums = new int[10];

int avg = 0;

nums[0] = 99;

nums[1] = 10;

nums[2] = 100;

nums[3] = 18;

nums[4] = 78;

nums[5] = 23;

nums[6] = 63;

nums[7] = 9;

nums[8] = 87;

nums[9] = 49;

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

avg = avg + nums[i];

avg = avg / 10;

Console.WriteLine("Среднее: " + avg);

}

}
```

Результат выполнения этой программы выглядит следующим образом.

```
Среднее: 53
```

Инициализация массива

В приведенной выше программе первоначальные значения были заданы для элементов массива nums вручную в десяти отдельных операторах присваивания. Конечно, такая инициализация массива совершенно правильна, но то же самое можно сделать намного проще. Ведь массивы могут инициализироваться, когда они создаются. Ниже приведена общая форма инициализации одномерного массива:

тип[] *имя_массива* = {*val1, val2, val3, ..., valN*};

где *val1-valN* обозначают первоначальные значения, которые присваиваются по очереди, слева направо и по порядку индексирования. Для хранения инициализаторов массива в C# автоматически распределяется достаточный объем памяти. А необходимость пользоваться оператором new явным образом отпадает сама собой. В качестве примера ниже приведен улучшенный вариант программы, вычисляющей среднее арифметическое.

```
// Вычислить среднее арифметическое ряда значений.

using System;

class Average {

static void Main() {

int[] nums = { 99, 10, 100, 18, 78, 23,

63, 9, 87, 49 };

int avg = 0;
for(int i=0; i < 10; i++)

avg = avg + nums[i];

avg = avg / 10;

Console.WriteLine("Среднее: " + avg);     }    }
```

Любопытно, что при инициализации массива можно также воспользоваться оператором new, хотя особой надобности в этом нет. Например, приведенный ниже фрагмент кода считается верным, но избыточным для инициализации массива nums в упомянутой выше программе.

```
int[] nums = new int[] { 99, 10, 100, 18, 78, 23,

63, 9, 87, 49 };
```

Несмотря на свою избыточность, форма инициализации массива с оператором new оказывается полезной в том случае, если новый массив присваивается уже существующей переменной ссылки на массив. Например:

```
int[] nums;

nums = new int[] { 99, 10, 100, 18, 78, 23,

63, 9, 87, 49 };
```

В данном случае переменная nums объявляется в первом операторе и инициализируется во втором.

И последнее замечание: при инициализации массива его размер можно указывать явным образом, но этот размер должен совпадать с числом инициализаторов. В качестве примера ниже приведен еще один способ инициализации массива nums.

```
int[] nums = new int[10] { 99, 10, 100, 18, 78, 23,

63, 9, 87, 49 );
```

В этом объявлении размер массива nums задается равным 10 явно.

Соблюдение границ массива

Границы массива в C# строго соблюдаются. Если границы массива не достигаются или же превышаются, то возникает ошибка при выполнении. Для того чтобы убедиться в этом, попробуйте выполнить приведенную ниже программу, в которой намеренно превышаются границы массива.

```
// Продемонстрировать превышение границ массива.

using System;

class ArrayErr {

static void Main() {

int[] sample = new int[10];

int i;

// Воссоздать превышение границ массива.

for(i = 0; i < 100; i = i+1)

sample[i] = i;

}

}
```

Как только значение переменной i достигает 10, возникнет исключительная ситуация типа IndexOutOfRangeException, связанная с выходом за пределы индексирования массива, и программа преждевременно завершится.

Присваивание ссылок на массивы

Присваивание значения одной переменной ссылки на массив другой переменной, по существу, означает, что обе переменные ссылаются на один и тот же массив, и в этом отношении массивы ничем не отличаются от любых других объектов. Такое присваивание не приводит ни к созданию копии массива, ни к копированию содержимого одного массива в другой. В качестве примера рассмотрим следующую программу.

```
// Присваивание ссылок на массивы.
using System;
class AssignARef {
static void Main() {
int i;
int[] nums1 = new int[10];
int[] nums2 = new int [10];
for(i=0; i < 10; i++) nums1[i] = i;
for(i=0; i < 10; i++) nums2[i] = -i;
Console.Write("Содержимое массива nums1: ");
for(i=0; i < 10; i++)
Console.Write(nums1[i] + " ");
Console.WriteLine();
```

```
Console.Write("Содержимое массива nums2: ");
for(i=0; i < 10; i++)
Console.Write(nums2[i] + " ");
Console.WriteLine();
nums2 = nums1; // теперь nums2 ссылается на nums1
Console.Write("Содержимое массива nums2\n" + "после присваивания: ");
for(i=0; i < 10; i++)
Console.Write(nums2[i] + " ");
Console.WriteLine();
// Далее оперировать массивом nums1 посредством
// переменной ссылки на массив nums2.
nums2[3] = 99;
Console.Write("Содержимое массива nums1 после изменения\n" +
"посредством переменной nums2: ");
for(i=0; i < 10; i++)
Console.Write(numsl[i] + " ");
Console.WriteLine();
}
```

Выполнение этой программы приводит к следующему результату.

Содержимое массива nums1: 0 1 2 3 4 5 6 7 8 9

Содержимое массива nums2: 0 -1 -2 -3 -4 -5 -6 -7 -8 -9

Содержимое массива nums2

после присваивания: 0 1 2 3 4 5 6 7 8 9

Содержимое массива nums1 после изменения

посредством переменной nums2: 0 1 2 99 4 5 6 7 8 9

Как видите, после присваивания переменной nums2 значения переменной nums1 обе переменные ссылки на массив ссылаются на один и тот же объект.

Применение свойства Length

Реализация в C# массивов в виде объектов дает целый ряд преимуществ. Одно из них заключается в том, что с каждым массивом связано свойство Length, содержащее число элементов, из которых может состоять массив. Следовательно, у каждого массива имеется специальное свойство, позволяющее определить его длину. Ниже приведен пример программы, в которой демонстрируется это свойство.

```
// Использовать свойство Length массива.
using System;
class LengthDemo {
static void Main() {
int[] nums = new int[10];
Console.WriteLine("Длина массива nums равна " + nums.Length);
// Использовать свойство Length для инициализации массива nums.
for (int i=0; i < nums.Length; i++)
nums[i] = i * i;
// А теперь воспользоваться свойством Length
// для вывода содержимого массива nums.
Console.Write("Содержимое массива nums: ");
for(int i=0; i < nums.Length; i++)
Console.Write(nums[i] + " ");
Console.WriteLine();
} }
```

При выполнении этой программы получается следующий результат.

```
Длина массива nums равна 10
Содержимое массива nums: 0 1 4 9 16 25 36 49 64 81
```

Заключение

В рамках данной лекции были изучены основы работы с одномерными массивами в языке C#: принципы их организации как объектов, синтаксис объявления и выделения памяти, способы инициализации при создании, правила индексации элементов, контроль границ массива, а также особенности присваивания ссылок на массивы.

Использование одномерных массивов позволяет эффективно хранить и обрабатывать большие объемы однотипных данных с помощью циклических конструкций, существенно сокращая объем программного кода. Понимание того, что в C# массивы являются ссылочными объектами со строгой проверкой границ (начиная с нулевого индекса), помогает предотвращать ошибки времени выполнения (такие как `IndexOutOfRangeException`) и корректно управлять данными при передаче ссылок между переменными.

Вопросы для самопроверки

1. **Понятие массива:** Что представляет собой массив в языке C# и в чем заключается его главное преимущество при обработке данных? Какая важная архитектурная особенность отличает массивы в C# от массивов в классических процедурных языках (например, в C)?

2. **Объявление и создание:** Из каких двух этапов состоит процесс создания массива? Приведите общий синтаксис объявления одномерного массива и выделения под него памяти с помощью оператора `new`.

3. **Индексация:** С какого числа начинается индексация элементов любого массива в C#? Какой индекс будет у первого и последнего элементов массива, состоящего из 10 элементов?

4. **Способы инициализации:** Как можно проинициализировать одномерный массив начальными значениями непосредственно при его объявлении? В каком случае при инициализации списком значений обязательно использовать оператор `new`?

5. **Контроль границ:** Что произойдет во время выполнения программы, если попытаться обратиться к элементу массива по индексу, выходящему за его допустимые пределы (например, по индексу `10` для массива размером `10`)? Какое исключение при этом генерируется?

6. **Ссылки на массивы:** Что происходит в памяти, когда одной переменной ссылки на массив присваивается значение другой переменной ссылки на массив (например, `nums2 = nums1`)? Изменятся ли данные в `nums1`, если модифицировать элемент через `nums2`?

Список используемых источников

1. **Шилдт Г.** C# 4.0: полное руководство. — М.: Вильямс, 2019. — 1056 с. *(Глава 7: Массивы и строки — подробный разбор одномерных массивов, инициализации и ссылок на массивы).*

2. **Троелсен Э., Джепикс Ф.** Язык программирования C# и платформа .NET. — СПб.: Диалектика, 2022. — 1312 с. *(Раздел: Работа с массивами в C#).*

3. **Албахари Д., Албахари Б.** C#. Справочник. Полное описание языка. — СПб.: БХВ-Петербург, 2023. — 1088 с. *(Спецификация массивов, правила индексации и контроля границ).*

4. **Павловская Т. А.** C#. Программирование на языке высокого уровня: Учебник для вузов. — СПб.: Питер, 2020. — 432 с. *(Глава: Одномерные и многомерные массивы).*

5. **Официальная документация Microsoft Learn:** Одномерные массивы (руководство по программированию на C#) [Электронный ресурс]. — URL: microsoft.com (дата обращения: 13.11.2026).