Основы программирования

Утилитные классы

Массивы

Структуры данных

Структуры данных - это совокупность элементов данных и отношений между ними. Элементами данных может быть как простое данное так и структура данных.

- Массив
- Списки односвязный и двусвязный
- Деревья
- Графы
- Стек
- Очередь
- …

Определение массива

Массив представляет собой структуру данных, содержащую ряд переменных, доступ к которым осуществляется с использованием расчетных индексов.

Все переменные, содержащиеся в массиве, которые также называются элементами массива, имеют одинаковый тип, который называется типом элементов массива.

(Спецификация языка C#)

Определение массива

Массив (`array`) — это *коллекция*[1)] переменных одинакового типа, обращение к которым происходит по их номеру с использованием общего для всех имени.

(Шилдт. Полный справочник по C#)

[1)] Коллекция это

Использование

Это удобное средство группирования связанных переменных.

Например, массив можно использовать для хранения:

- значений максимальных дневных температур за месяц;
- списка цен на акции;
- названий книг по программированию из домашней библиотеки.

Особенности

- Массивы реализованы как объекты.
- Одно из преимуществ реализации массивов как объектов состоит в том, что неиспользуемые массивы могут автоматически утилизироваться системой сбора мусора.

Ранг массива

Массив имеет ранг, определяющий количество индексов, связанных с каждым из элементов массива.

Ранг массива указывает количество его измерений.

Массив с рангом, равным единице, называется *одномерным массивом*.

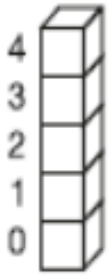
Массив с рангом больше единицы называется *многомерным массивом*. Многомерные массивы конкретного размера часто называются двумерными, трехмерными и так далее.

Массивы бывают

- одномерные;
- многомерные, прямоугольные (двумерные, трехмерные, …);
- ступенчатые, зубчатые, рваные.

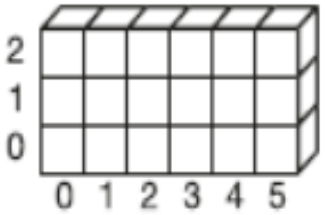
Виды массивов

Одномерный массив

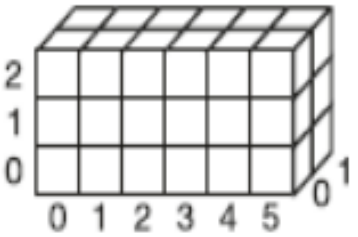


Одномерный массив
int[5]

Многомерные массивы

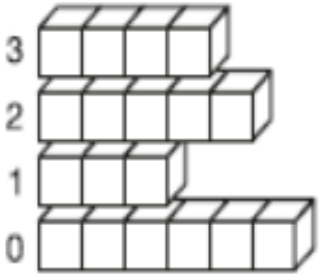


Двухмерный массив
int[3,6]



Трехмерный массив
int[3,6,2]

Зубчатый массив



Зубчатый массив
int[4][]

Одномерные массивы

Одномерный массив – это линейная ограниченная последовательность однотипных данных, имеющих одно имя и предполагающих обращение к элементам последовательности по их номеру (индексу).

Объявление одномерного массива

```
тип[] имя_массива = new тип [размер] {значения};
```

При создании объекта массива его элементам задаются значения по умолчанию:

- «0» для числовых типов;
- «false» для булевого типа;
- «null» для ссылочных типов.

Особенности создания массивов

- При объявлении ссылки на массив объект массива не создается, а ссылке присваивается значение `null`, однако переменная, хранящая ссылку на массив, считается неинициализированной.
- Объект массива создается явно с помощью операции `new`. Тогда в динамической памяти выделяется место для хранения массива, и только на этом этапе задается количество элементов массива (длина массива).

```
int[] arr = new int[5];
```

Количество элементов массива

- Количество элементов массива может быть задано целым положительным числом.
- Также количество элементов массива может быть задано вычислимым выражением, результат которого имеет тип, приводимый к `int`, `uint`, `long` или `ulong`.

Операции с массивами

Переменные типа массив можно присваивать и сравнивать между собой.

Длина массива и нумерация его элементов

- Для задания максимального индекса элемента в массиве используется его свойство `Length`, хранящее количество элементов в массиве.
  Например, `array2.Length`
- Элементы массива всегда нумеруются с `0`, то есть последний элемент массива имеет номер, на единицу меньший, чем длина массива.

Инициализация элементов массива

Элементы массива инициализируются явно, чаще всего, с помощью циклических операторов.
Например:

```
int[] arr = new int[5];
for (int i = 0; i < arr.Length; i++)
{
    arr[i] = i;
}
```

Для полного задания массива должно быть известно:

- тип элементов,
- имя массива,
- количество элементов,
- значения элементов.

Варианты объявления и инициализации массива

```
int[] arr1 = new int[4];
int[] arr2 = new int[4] {1,2,3,4};
int[] arr3 = new int[] {1,2,3,4};
int[] arr4 = {1, 2, 3, 4};
```

Пример

```
Console.WriteLine(”Введите длину массива”);
int[] arr;
int len = Int32.Parse(Console.ReadLine());
arr = new int[len];
for (int i = 0; i < arr.Length; i)
{
 Console.Write(”Введите {0} элемент массива:”, i);
 arr[i]= Int32.Parse(Console.ReadLine());
}
for (int i = 0; i < arr.Length-1; i++)
{
 Console.Write(”{0}, ”, arr[i]);
}
 Console.WriteLine(arr[arr.Length-1]);
```

Прямоугольный массив

Ранг прямоугольного массива более единицы.

Двумерный массив представляет собой прямоугольник или матрицу.

Трехмерный массив это параллелепипед.

Объявление двумерного массива

Двумерный массив элементов целого типа:

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

Варианты объявления и инициализации двумерного массива

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

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

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

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

Пример работы с массивом 1

```
Console.WriteLine(”Число строк массива”);
int rows = Int32.Parse(Console.ReadLine());

Console.WriteLine(”Число столбцов массива”);
int columns = Int32.Parse(Console.ReadLine());

int[,] arr = new int[rows,columns];
```

Пример работы с массивом 2

```
//построчный ввод массива
for (int i = 0; i < rows; i++)
{
 for (int j = 0; j < columns; j++)
 {
  Console.Write(”Элемент {0} строки {1}:”, j, i);
  arr[i,j]= Int32.Parse(Console.ReadLine());
 }
}
```

Пример работы с массивом 3

```
//построчный вывод массива
for (int i = 0; i < rows; i++)
{
 for (int j = 0; j < columns-1; j++)
 {
  Console.Write(”{0}, ”, arr[i,j]);
 }
 Console.WriteLine(arr[i,columns-1]);
}
```

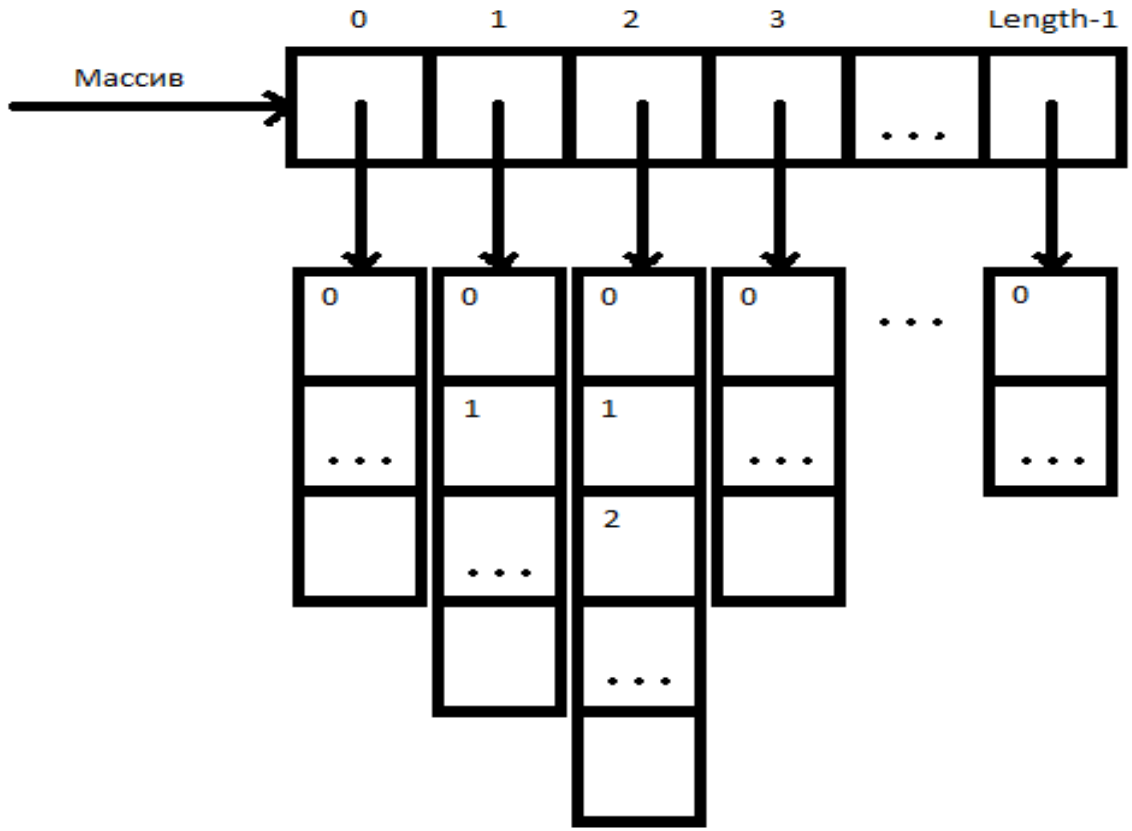
Ступенчатые массивы

Ступенчатый массив – это одномерный массив ступенчатых массивов, размерность которых на единицу меньше:

- двумерный ступенчатый массив – это одномерный массив одномерных массивов,
- трехмерный ступенчатый массив – это одномерный массив двумерных ступенчатых массивов, каждый из которых является одномерным массивом и т.д.

Еще один термин «массив массивов».

Общий вид двумерного ступенчатого массива



Объявление ступенчатого массива

Объявление ссылки на двумерный ступенчатый массив элементов целого типа:

```
int[][] arr;
```

Объявление и создание объекта двумерного ступенчатого массива:

```
int[][] arr = new int[3][];
```

Объявление двумерного ступенчатого массива

```
int[ ][ ] arr = new int[3][ ];

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

Свойство Length

```
int[] mas1 = new int[3]  { 1, 2, 3};
Console.WriteLine("Свойство Length mas1:  " +
mas1.Length);

int[,] mas2 = new int[2, 3] {{1,2,3},{4,5,6}};
Console.WriteLine("Свойство Length mas2:  " +
mas2.Length);
```

Метод GetLength()

```
int[,] mas2 = new int[2, 3] {{1,2,3},{4,5,6}};

Console.WriteLine("Метод GetLength mas2:  " +
mas2.GetLength(0));

Console.WriteLine("Метод GetLength mas2:  " +
mas2.GetLength(1));
```

Цикл foreach

Используется для опроса элементов *коллекции.*

Коллекция — это группа объектов.

C# определяет несколько типов коллекций, и одним из них является массив.

Синтаксис:

```
foreach (тип имя_переменной in коллекция)
{     тело_цикла}
```

Пример использования foreach

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

foreach (int elem in arr)
{
Console.Write(elem + ” ”);
}
Console.Write(”\n”); //перевод на новую строку
```

Массивы объектов

Массив может содержать объекты не только значащего типа.

В массиве могут содержаться элементы класса, описанного пользователем.

Такой массив называют массивом объектов.

Пример массива объектов

```
class MyClass
{
…
}
…

MyClass[] arrayOfMyClass = new MyClass[3];
arrayOfMyClass[0] = new MyClass();
arrayOfMyClass[1] = new MyClass();
arrayOfMyClass[2] = new MyClass();
```

Класс Array

Является базовым для всех массивов.

То есть все массивы обладают функциональностью, описанной в этом классе.

Основные свойства класса Array

- свойство **Length** возвращает длину массива.
- свойство **Rank** возвращает размерность массива.

Статические методы класса Array

- **Copy()** - позволяет копировать весь массив или его часть в другой массив.
- **IndexOf()**, **LastIndexOf()** - определяют индексы первого и последнего вхождения образца в массив, возвращая -1, если такового вхождения не обнаружено.
- **Reverse()** - выполняет обращение массива, переставляя элементы в обратном порядке.
- **Sort()** - осуществляет сортировку массива.
- **BinarySearch()** - определяет индекс первого вхождения образца в отсортированный массив, используя алгоритм двоичного поиска.

Статические методы класса Array

- **Clear()** очищает массив, устанавливая для всех его элементов значение по умолчанию.
- **Exists()** проверяет, содержит ли массив определенный элемент.
- **Find()** находит элемент, который удовлеворяет определенному условию.
- **FindAll()** находит все элементы, которые удовлеворяет определенному условию.
- **Resize()** изменяет размер одномерного массива.

Класс Array

Класс `Array` является базовым для всех массивов, значит, любой массив неявно преобразуем к типу `Array`.

Но класс `Array` не поддерживает индексацию. То есть, если на массив есть только ссылка типа `Array`, то к элементам массива нет доступа с помощью операции «[ ]», а только с помощью методов `GetValue` и `SetValue`,

Источники

1. METANIT.COM. C#/.Net: Полное руководство по языку программирования C# 8.0 и платформе .NET Core 3. Массивы. https://metanit.com/sharp/tutorial/2.4.php
2. Массивы (Руководство по программированию на C#) https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/arrays/