МАССИВЫ

Массив – это структура данных, которая позволяет хранить набор элементов одного типа под одним именем

Каждый элемент массива имеет свой уникальный индекс, начиная с нуля. Это означает, что если массив состоит из пяти элементов, то их индексы будут: 0, 1, 2, 3, 4.

Зачем нужны массивы ?

Массивы упрощают работу с набором однотипных данных, обеспечивая быстрый доступ к каждому элементу по индексу.

Представьте, что вам нужно сохранить список учеников класса. Вместо того чтобы объявлять переменные для каждого ученика (ученик1, ученик2, ..., ученикN), вы можете создать массив и обращаться к каждому ученику по его номеру в списке.

Индексация массивов?

Например, если у вас есть массив students, то students[0] будет первым учеником, students[1] – вторым и так далее.

Этот массив состоит из 5 элементов

Но индекс начинается с нуля

Одномерный массив

Одномерный массив представляет собой линейную последовательность элементов одинакового типа. Каждый элемент имеет свой уникальный индекс, начиная с нуля.

ЧИСЛА

```
// одномерный массив чисел. Создание и инициализация массива
int[] массивЧисел = { 50, 22, 30, 50, 70 };
Console.WriteLine($"Обычный массив чисел: {массивЧисел[1]}");
```

СТРОКИ

```
//одномерный массив строк. Создание и инициализация массива
string[] массивСлов = { "привет", "пока", "До свидания" };
Console.WriteLine($"Обычный массив строк: {массивСлов[1]}");
```

Вывод на консоль

```
// вывод на консоль всех элементов массива
Console.WriteLine($"Все элементы массива: {string.Join(", ", массивЧисел)}");
```

Многом. (двумерный) массив

Многомерный массив – это массив, который содержит несколько измерений. Например, двумерный массив можно представить как таблицу, где каждый элемент определяется двумя индексами: строкой и столбцом.

СТРОКИ

```
// Создаем многомерный массив строк размером 3x3
string[,] многомерныйМассив = new string[3, 3];
// Заполняем массив значениями
многомерныйМассив[0, 0] = "A1";
многомерныйМассив[0, 1] = "B1";
многомерныйМассив[0, 2] = "C1";
многомерныйМассив[1, 0] = "A2";
многомерныйМассив[1, 1] = "B2";
многомерныйМассив[1, 2] = "C2";
многомерныйМассив[2, 0] = "A3";
многомерныйМассив[2, 2] = "B3";
многомерныйМассив[2, 2] = "C3";
```

A1	B1	C1
B1	B2	B3
C1	C2	C3

ЧИСЛА

```
//многомерный массив
int[,] многоМассив = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9},
};
```

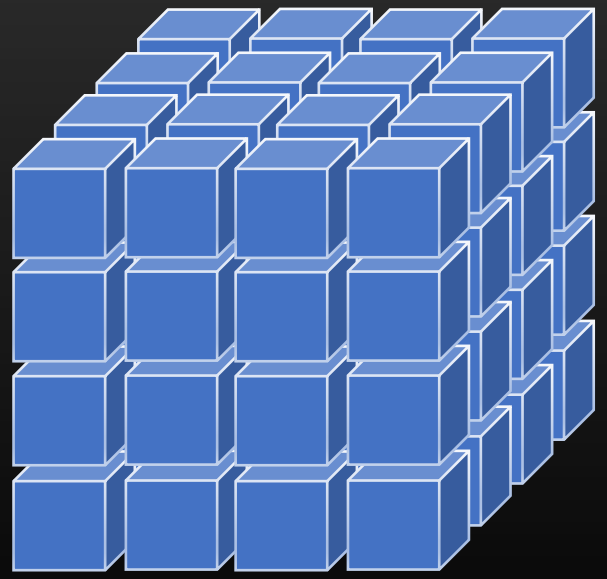
1	2	3
4	5	6
7	8	9

Трёхмерный массив

Трёхмерный массив — это структура данных, представляющая собой набор элементов, организованных в виде трёх измерений.

```
// Создаём трёхмерный массив размером 4x4x4
        int[,,] трехмерныйМассив = new int[4, 4, 4]
            {
            {
                { 1, 2, 3, 4 },
                { 5, 6, 7, 8 },
                { 9, 10, 11, 12 },
                { 13, 14, 15, 16 }
            },
            {
                { 17, 18, 19, 20 },
                { 21, 22, 23, 24 },
                { 25, 26, 27, 28 },
                { 29, 30, 31, 32 }
            },
            {
                { 33, 34, 35, 36 },
                { 37, 38, 39, 40 },
                { 41, 42, 43, 44 },
                { 45, 46, 47, 48 }
            },
            {
                { 49, 50, 51, 52 },
                { 53, 54, 55, 56 },
                { 57, 58, 59, 60 },
                { 61, 62, 63, 64 }
            }
            };
            Console.WriteLine(трехмерныйМассив[0, 3, 2]); //число 15
```

Трёхмерный массив — это структура данных, представляющая собой набор элементов, организованных в виде трёх измерений. Его можно представить как куб или сетку, где каждый элемент имеет три координаты: одна для первого измерения (слой), вторая для второго измерения (строка) и третья для третьего измерения (столбец).



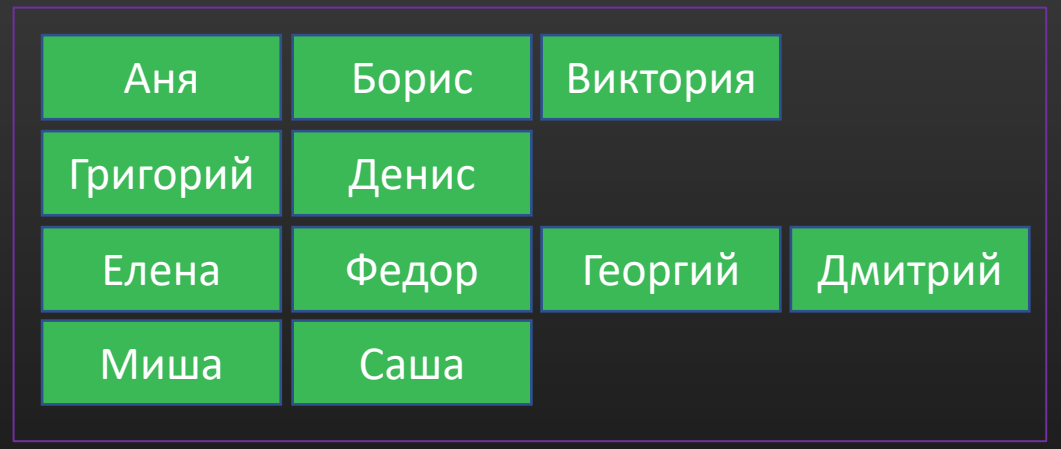
Трёхмерный массив — это расширение концепции двумерного массива, добавляющее третье измерение. Он может быть использован для хранения и обработки данных, которые имеют три оси или параметра.

Зубчатый массив

Зубчатый массив (или массив массивов) – это структура данных, где каждый элемент основного массива является другим массивом. Это позволяет хранить данные разной длины в каждом элементе.

```
// Создаем зубчатый массив, состоящий из четырёх массивов строк
string[][] зубчатыйМассив = new string[4][];

// Инициализируем каждый внутренний массив
зубчатыйМассив[0] = new string[] { "Аня", "Борис", "Виктория" };
зубчатыйМассив[1] = new string[] { "Григорий", "Денис" };
зубчатыйМассив[2] = new string[] { "Елена", "Федор", "Георгий", "Дмитрий" };
зубчатыйМассив[3] = new string[] { "Миша", "Саша" };
```

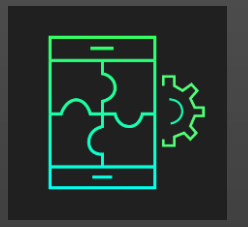



```
// Выводим все элементы зубчатого массива,
объединяя их в одну строку
Console.WriteLine(
    $"Все элементы зубчатого массива:\n" +
    string.Join("\n",
    зубчатыйМассив.Select(внутреннийМассив =>
    string.Join(", ", внутреннийМассив)))
);
```

- Здесь используется метод LINQ Select, который применяет определённую операцию ко всем элементам коллекции.
- В данном случае коллекция — это наш зубчатыйМассив.
- Каждый элемент зубчатого массива (каждый внутренний массив) передаётся в лямбда-выражение внутреннийМассив => String.Join(", ", внутреннийМассив):

-Внутри этого лямбда-выражения вызывается метод String.Join(", ", внутреннийМассив), который объединяет все элементы внутреннего массива в одну строку, разделяя их запятой и пробелом (", ").

Методы работы с массивами



Вот некоторые методы для работы с массивами
(хотя есть ещё Sort, Reverse, Clear и др.)

```
// Выводит индекс элемента
int индексМассива = Array.IndexOf(массивЧисел, 50);
Console.WriteLine($"Вывод индекса массива: {индексМассива}");
```

Метод Array.IndexOf ищет первое вхождение указанного элемента в массиве и возвращает его индекс.

```
// Выводит индекс последнего вхождения элемента
int обратныйИндексМассива = Array.LastIndexOf(массивЧисел, 50);
Console.WriteLine($"Вывод обратного индекса массива: {обратныйИндексМассива}");
```

Метод Array.LastIndexOf ищет последнее вхождение указанного элемента в массиве и возвращает его индекс.

```
// Вывод скопированных элементов
int[] новыйМассив = new int[3];
Array.Copy(массивЧисел, 1, новыйМассив, 0, 3);
Console.WriteLine($"Вывод скопированных элементов: {string.Join(", ", новыйМассив)}");
```

Метод Array.Copy копирует часть массива в другой массив.

Метод string.Join выведет все элементы массива последовательно и объединит все элементы массива в одну строку, разделяя символом.