

Массивы в C#

Массивы в C#

В языке C# каждый индекс массива изменяется в диапазоне от 0 до некоторого конечного значения. Массивы в языке C# являются настоящими динамическими массивами. Как следствие этого, массивы относятся к ссылочным типам, память им отводится динамически в "куче". Массивы могут быть одномерными и многомерными.

Объявление одномерного массива

<тип>[] <объявители>;

каждый объявитель может быть именем или именем с инициализацией.

Объявление массива:

- с инициализацией
- с отложенной инициализацией

Объявление одномерного массива

Объявление массива с инициализацией:

- Явная инициализация константным массивом
- Создание массива с помощью операции `new`

Пример явной инициализации:

```
double[] x = {5.5, 6.6, 7.7};
```

в динамической памяти создаётся константный массив с заданными значениями, с которым и связывается ссылка.

Пример создания с помощью операции `new`:

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

массив создаётся в динамической памяти, его элементы получают начальные нулевые значения, и ссылка связывается с этим массивом.

Объявление одномерного массива

Объявление с отложенной инициализацией выполняется в 2 этапа:

1. Объявление массива
2. Инициализация массива

Объявление одномерного массива

Объявление массива

Пример:

```
int[ ] a;
```

При объявлении с отложенной инициализацией сам массив не формируется, а создаётся только ссылка на массив, имеющая неопределённое значение Null. Поэтому пока массив не будет реально создан и его элементы инициализированы, использовать его в вычислениях нельзя!!!

Объявление одномерного массива

Инициализация массива:

Пример:

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

Выражение, задающее границу изменения индексов, в динамическом случае может содержать переменные.

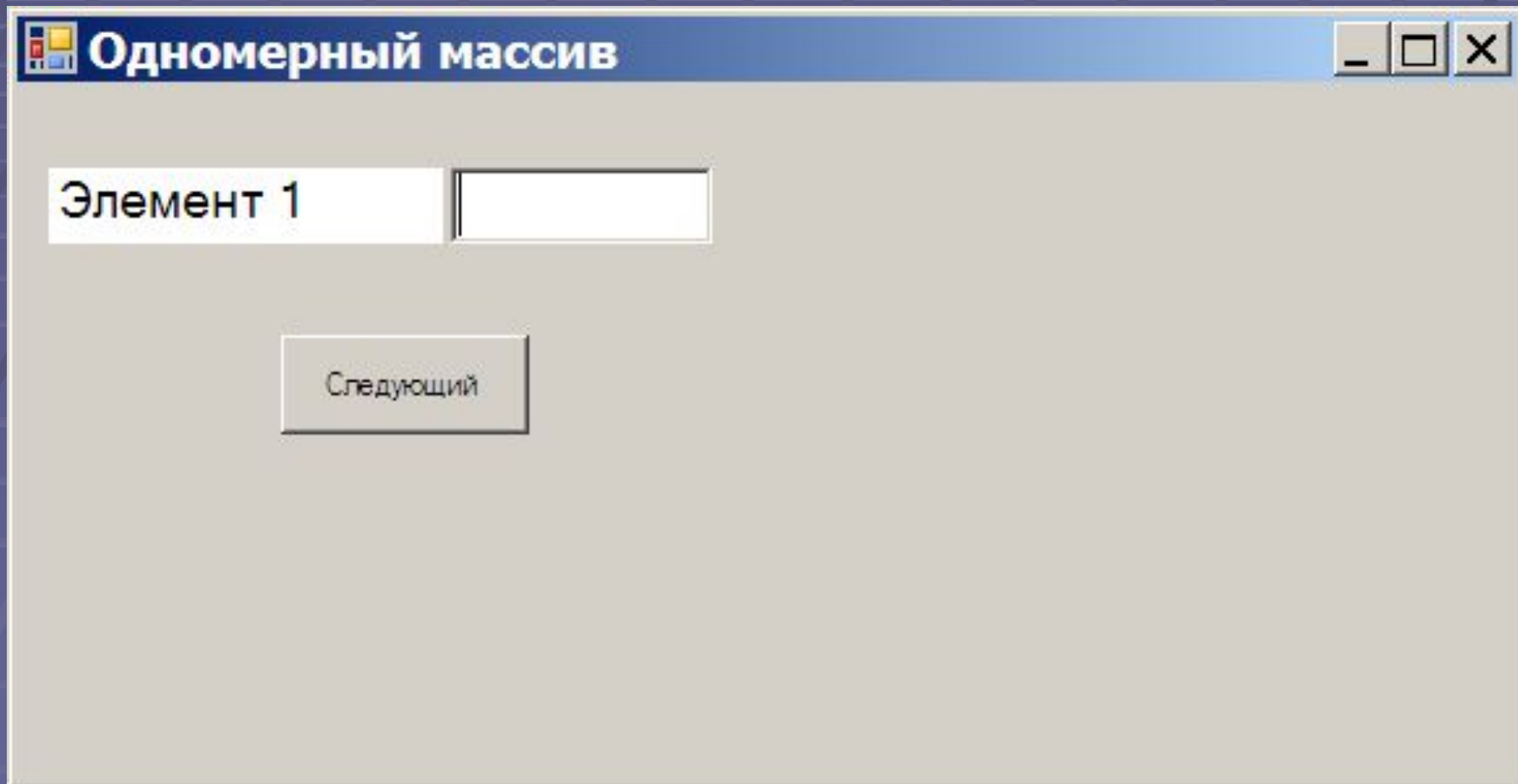
Единственное требование – значения переменных должны быть определены в момент объявления!

Пример:

```
n=Convert.ToInt32(Textbox1.Text);
```

```
a = new int[n];
```

Ввод одномерного массива



Одномерный массив

Элемент 1

Следующий

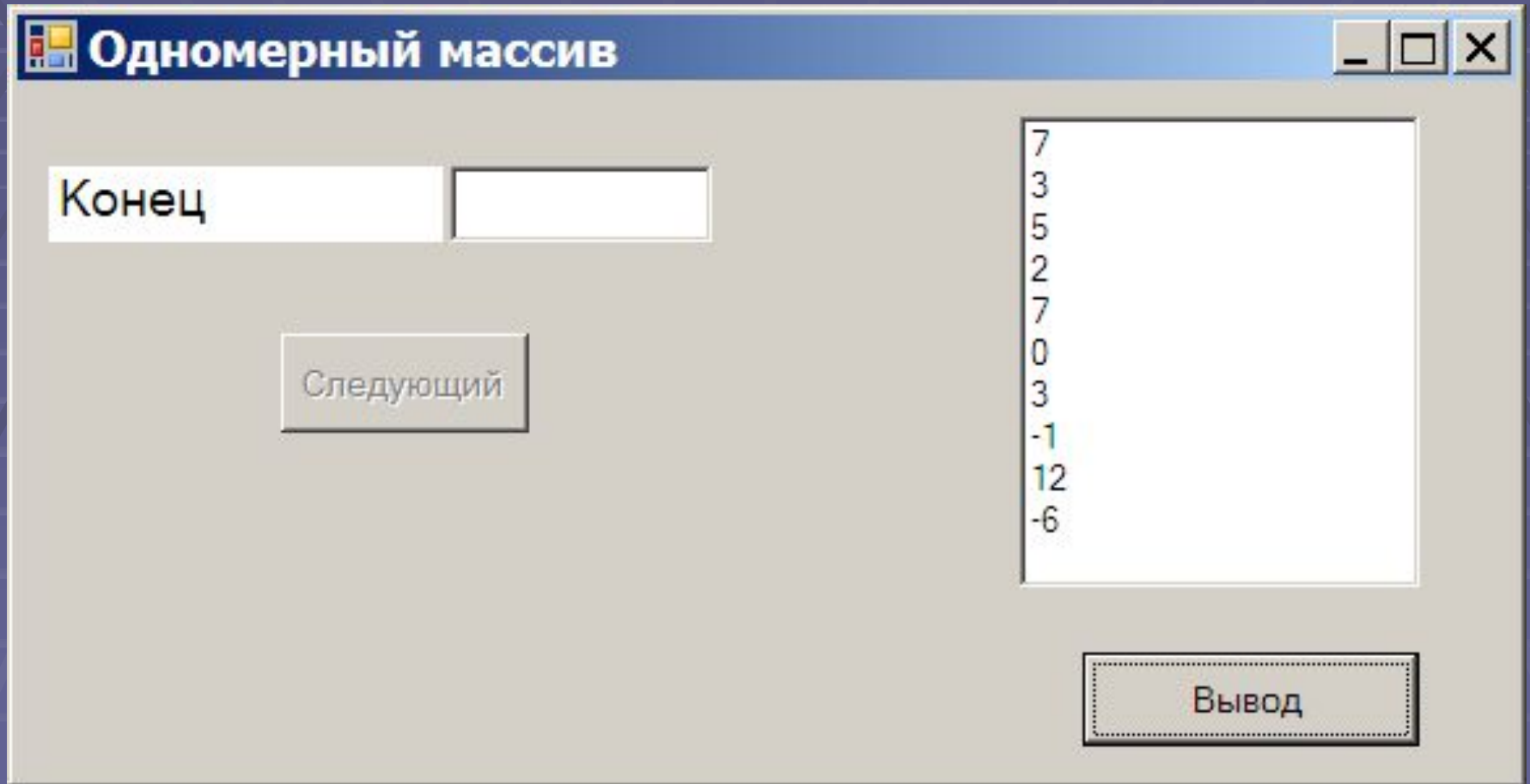
Ввод одномерного массива

```
namespace одномерный_массив
{
    public partial class Form1 : Form
    {
        int[] a;
        int i;
        public Form1()
        {
            InitializeComponent();
            a = new int[10];
            i = 0;
            textBox1.Focus();
            label1.Text = "Элемент " + Convert.ToString(i + 1);
        }
    }
}
```

Ввод одномерного массива

```
private void button1_Click(object sender, EventArgs e)
{
    a[i] = Convert.ToInt32(textBox1.Text);
    i++;
    if (i < 10)
    {
        label1.Text = "Элемент " + Convert.ToString(i + 1);
        textBox1.Focus();
    }
    else
    {
        button1.Enabled = false;
        label1.Text = "Конец";
    }
    textBox1.Text = "";
}
```

Вывод одномерного массива



Вывод одномерного массива

Первый вариант:

```
private void button2_Click(object sender,  
    EventArgs e)  
    { listBox1.Items.Clear();  
      for(int k=0; k<10; k++)  
  
        listBox1.Items.Add(Convert.ToString(a[k]));  
    }
```

Вывод одномерного массива

Новый оператор цикла:

**foreach(тип идентификатор in массив)
оператор**

Цикл работает в полном соответствии со своим названием – тело цикла выполняется для каждого элемента в контейнере. Тип идентификатора должен быть согласован с типом элементов, хранящихся в массиве данных. На каждом шаге цикла идентификатор, задающий текущий элемент массива, получает значение очередного элемента в соответствии с порядком, установленным на элементах массива. С этим текущим элементом и выполняется тело цикла - выполняется столько раз, сколько элементов находится в массиве. Цикл заканчивается, когда полностью перебраны все элементы массива.

Вывод одномерного массива

Второй вариант:

```
private void button2_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear();
    foreach (int item in a)
        listBox1.Items.Add(Convert.ToString(item));
}
```

Многомерные массивы

Объявление многомерных массивов

<тип>[, ... ,] <объявители>;

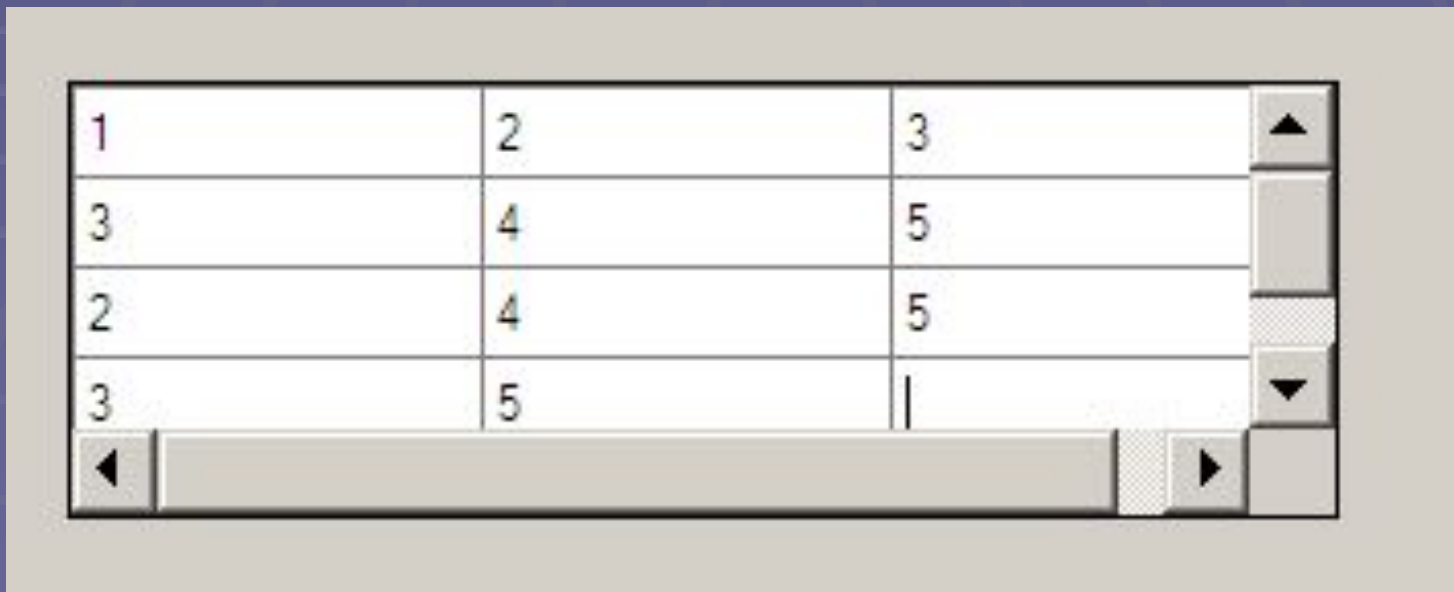
Число запятых, увеличенное на единицу, задаёт размерность массива. Всё, что сказано для одномерных массивов, справедливо и для многомерных.

Пример явной инициализации двумерного массива:

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

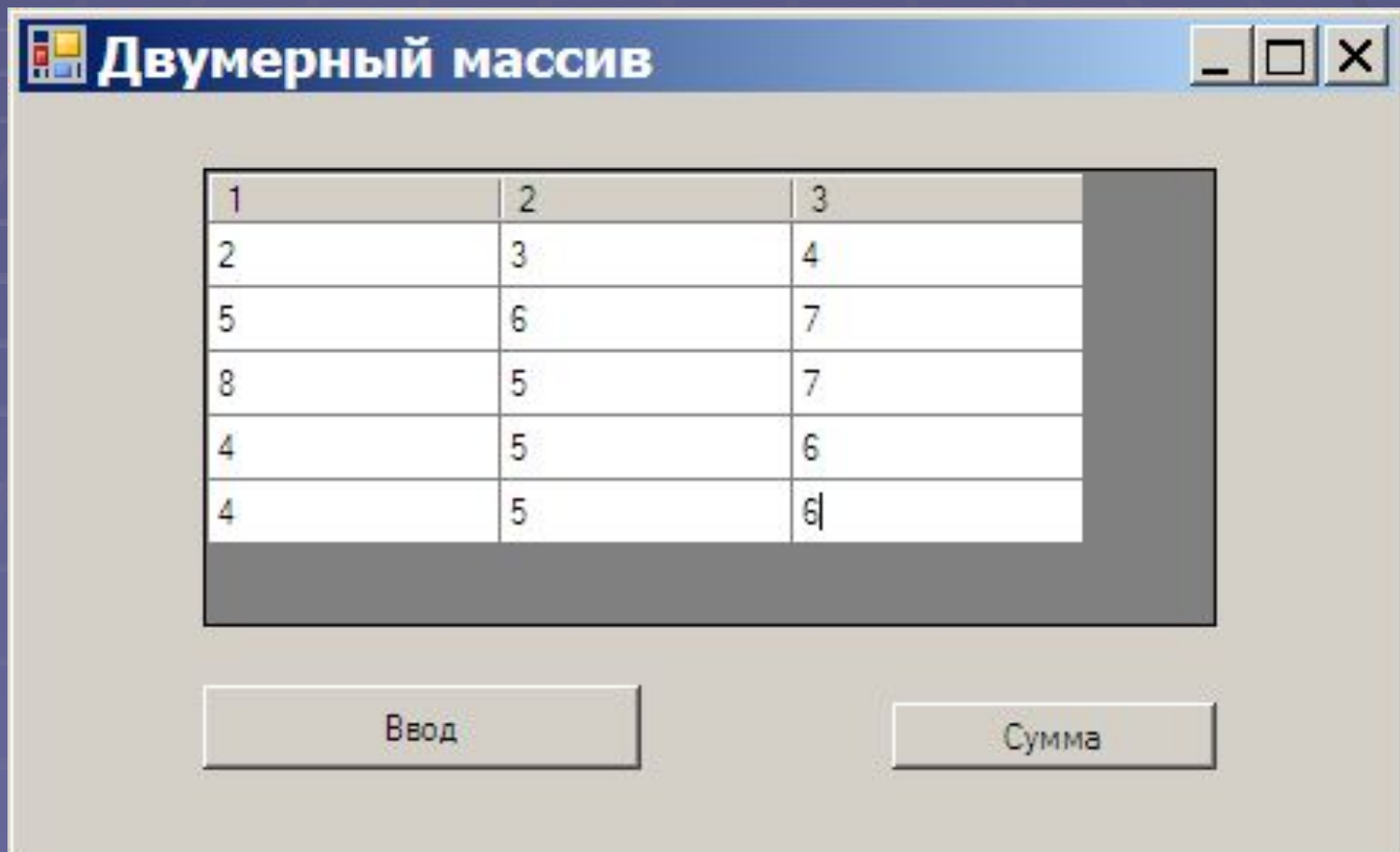
Многомерные массивы

Визуальный компонент для ввода/вывода двумерного массива: *DateGridView* из вкладки *Data*



Многомерный массив

Пример ввода двумерного массива:



The screenshot shows a window titled "Двумерный массив" with a standard Windows title bar. Inside the window is a 6x3 grid of text boxes containing the following values:

1	2	3
2	3	4
5	6	7
8	5	7
4	5	6
4	5	6

Below the grid are two buttons: "Ввод" on the left and "Сумма" on the right.

Многомерный массив

```
namespace Двумерный_массив
{
    public partial class Form1 : Form
    {
        static int n=5,m=3;
        int[,] matr=new int[n, m];
        public Form1()
        {
            InitializeComponent();
            dataGridView1.ColumnCount = m;
            dataGridView1.RowCount = n;
            for(int k=0;k<3;k++)
                dataGridView1.Columns[k].HeaderText = Convert.ToString(k+1);
        }
    }
}
```

Многомерный массив

```
private void button1_Click(object sender, EventArgs e)
{
    for(int i=0;i<n;i++)
        for(int j=0;j<m;j++)

            matr[i,j]=Convert.ToInt32(dataGridView1.Rows[i].Cells[j].Value
);
}
```

Многомерный массив

```
private void button2_Click(object sender, EventArgs e)
{
    int sum = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            sum = sum + matr[i, j];
    MessageBox.Show(Convert.ToString(sum));
}
```

Класс Array

Все классы, являющиеся массивами, имеют много общего, поскольку все они являются потомками класса

`System.Array` из библиотеки FCL .

Класс `Array` имеет довольно большое число собственных методов и свойств.

Класс Array

Методы класса Array:

1. Copy – позволяет копировать весь массив или его часть в другой массив.
2. IndexOf, LastIndexOf – определяют индексы первого и последнего вхождения образца в массив, возвращая значение -1, если такового вхождения не обнаружено.
3. Reverse – выполняет обращение массива, переставляя элементы в обратном порядке.
4. Sort – осуществляет сортировку массива.
5. BinarySearch – определяет индекс первого вхождения образца в отсортированный массив, используя алгоритм двоичного поиска.
6. Clear – обнуление элементов массива в заданном диапазоне

Класс Array

Примеры использования методов:

1. `Array.Copy(a, b, 4);`
2. `int k= Array.IndexOf(a,100);`
3. `Array.Reverse(a);`
4. `Array.Sort(a);`
5. `int k = Array.BinarySearch(a, 100);`
6. `Array.Clear(a, 3, 4);`