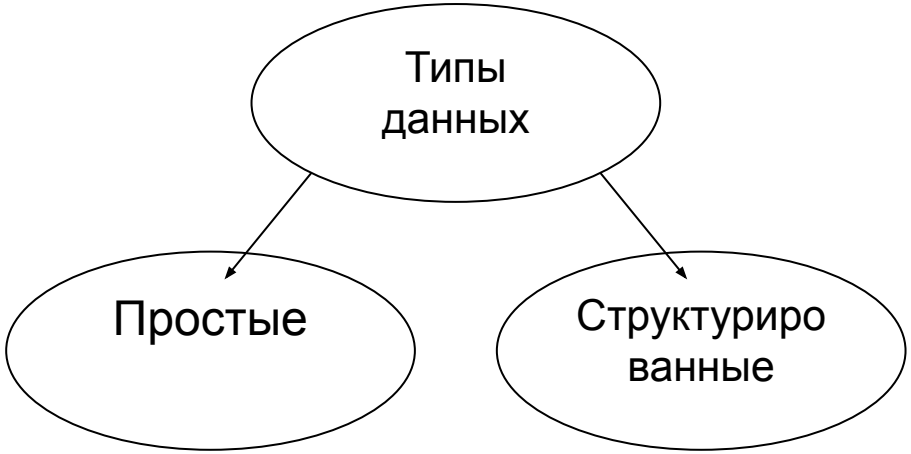
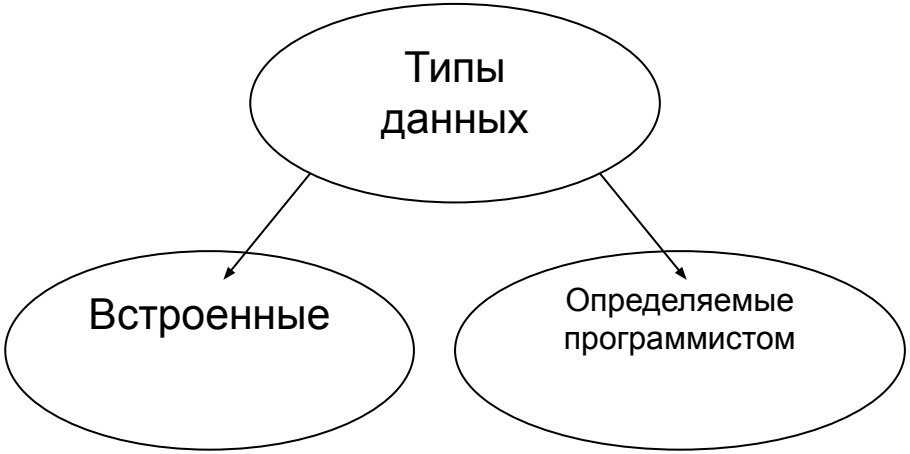
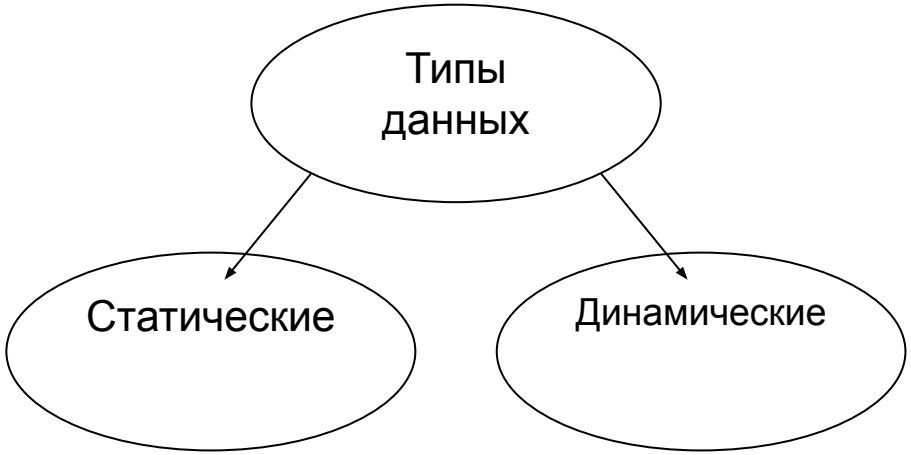
Массивы в C#

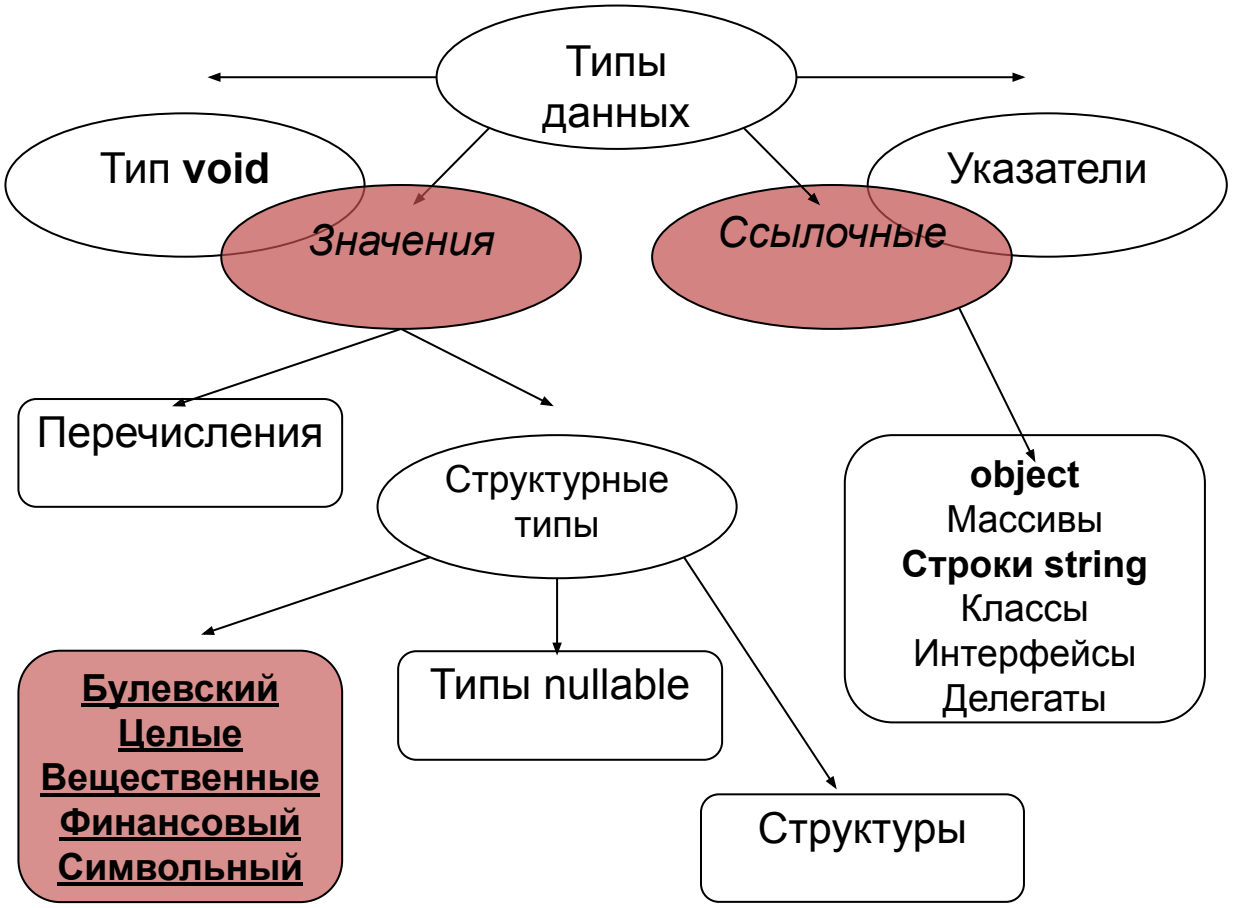
Различные классификации типов данных







Основная классификация типов C#

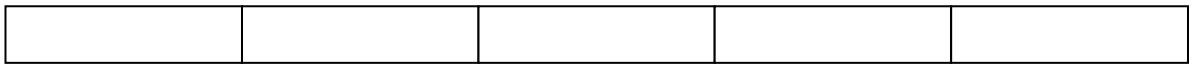


Типы C# разделяют *по способу хранения элементов* на типы-значения и ссылочные типы. Элементы *значимых типов* (value types), представляют собой просто последовательность битов в памяти, необходимый объем которой выделяет компилятор. Величины **значимых типов** хранят свои значения непосредственно. Величина ***ссылочного типа*** хранит не сами данные, а ссылку на них (адрес, по которому расположены данные). Сами данные хранятся в хипе.

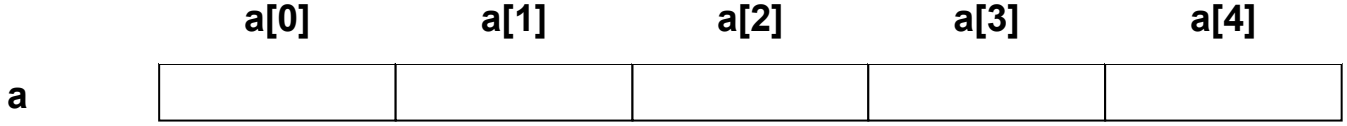
Массивы

- *Массив* — ограниченная совокупность однотипных величин
- Элементы массива имеют одно и то же имя, а различаются по порядковому номеру (*индексу*)

Пять простых переменных (в стеке):



Массив из пяти элементов значимого типа (в хипе):



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 сознания 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!!!

Одномерные массивы

- Варианты описания массива:

тип[] имя;

тип[] имя = new тип [размерность];

тип[] имя = { список_инициализаторов };

тип[] имя = new тип [] { список_инициализаторов };

тип[] имя = new тип [размерность] { список_инициализаторов };

- Примеры описаний (один пример на каждый вариант описания, соответственно):

```
int[] a;                                    //  элементов нет
int[] b = new int[4];                       //  элементы равны 0
int[] c = { 61, 2, 5, -9 };                 //  new подразумевается
int[] d = new int[] { 61, 2, 5, -9 };   // размерность вычисляется
int[] e = new int[4] { 61, 2, 5, -9 };      // избыточное описание
```

Размерность массива

- Количество элементов в массиве (*размерность*) задается при выделении памяти и не может быть изменена впоследствии. Она может задаваться выражением:

short n = ...;

string[] z = new string[2*n + 1];

- Размерность не является частью типа массива.
- Элементы массива нумеруются *с нуля*.

Для *обращения к элементу массива* после имени массива указывается номер элемента в квадратных скобках, например:

w[4] z[i]

- С элементом массива можно делать все, что допустимо для переменных того же типа.
- При работе с массивом автоматически выполняется *контроль выхода за его границы*: если значение индекса выходит за границы массива, генерируется исключение IndexOutOfRangeException.

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];
```

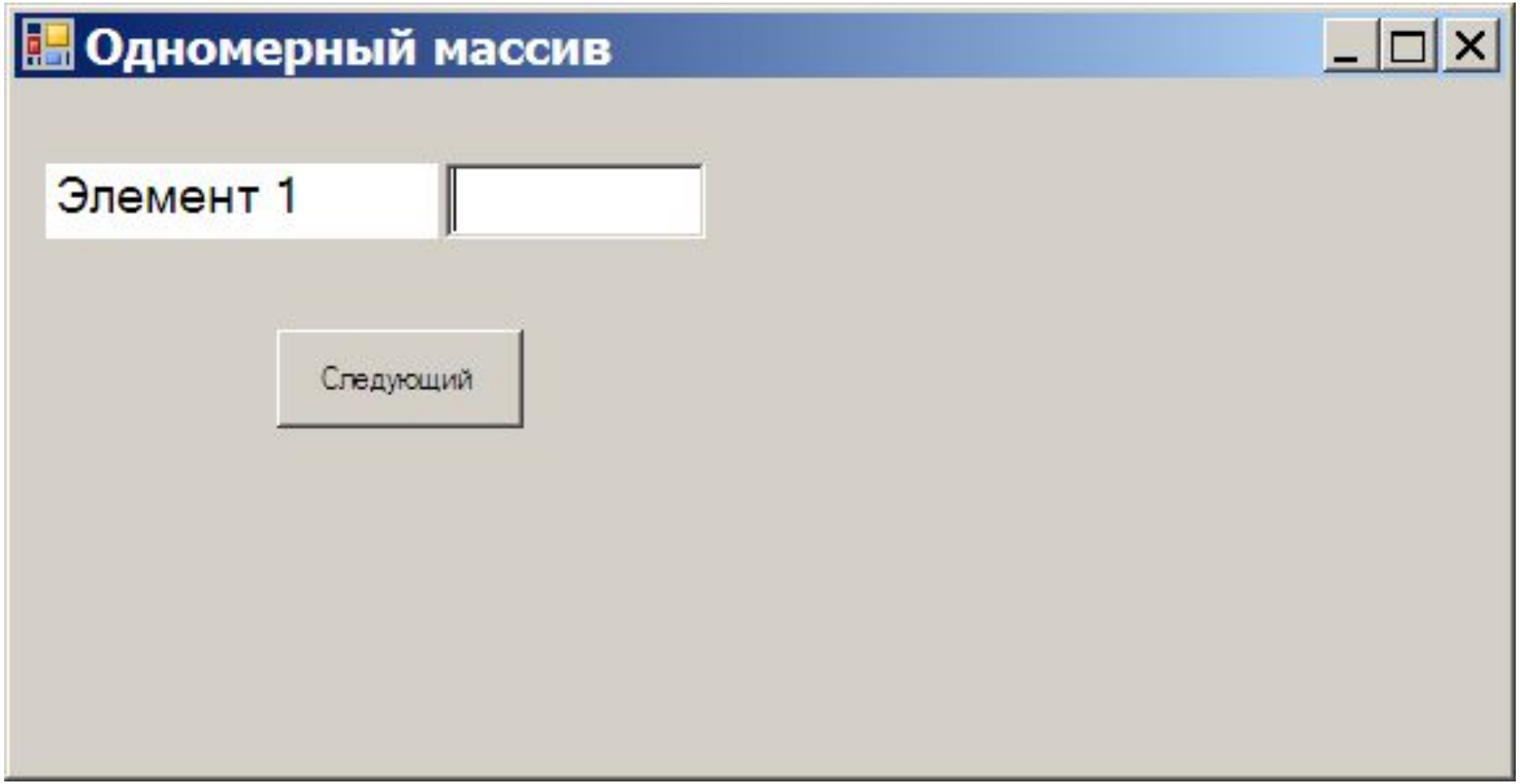
Пример 1. *С клавиатуры ввести элементы массива из 20-ти значений. Вывести элементы массива с указанием для каждого его индекса (порядкового номера в массиве)*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            double[] a = new double[20];
            string b;

            Console.WriteLine("Введите числа в массив");
            for (int i = 0; i <= 19; i++)
            {
                Console.Write("a[{0}]=",i);
                a[i] = Convert.ToDouble(Console.ReadLine());
            }
            Console.Write("Массив заполнен. Вывести значения на экран (y/n)?");
            b = Console.ReadLine();
            if (b == "y")
                for (int i = 0; i <= 19; i++)
                    Console.WriteLine("a[{0}]={1}", i, a[i]);
            Console.ReadKey();
        }
    }
}
```

Ввод одномерного массива в Windows Forms



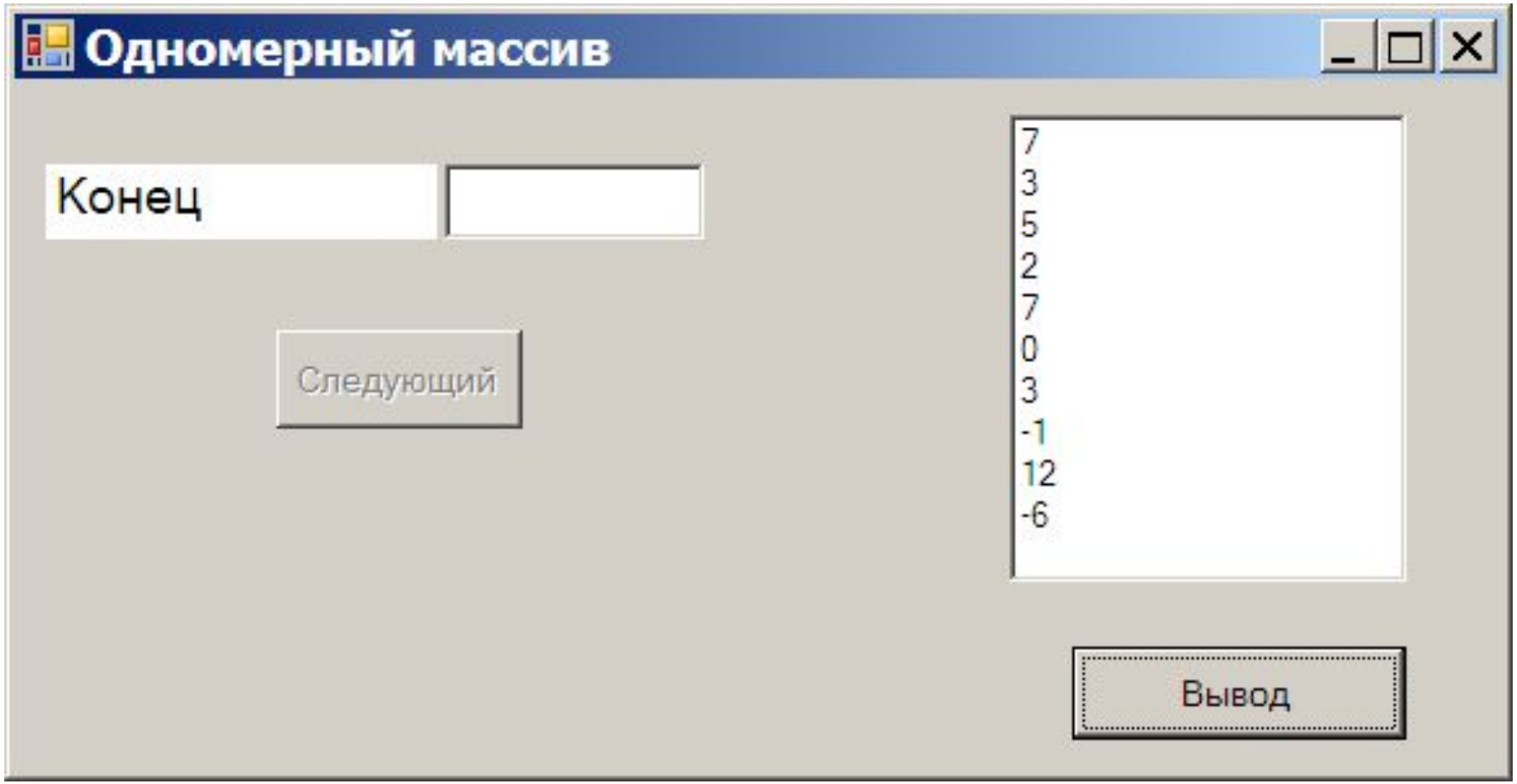
Ввод одномерного массива в Windows Forms

```
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 в Windows Forms

```
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));
    }
```

Действия с массивами

- Массивы одного типа можно *присваивать* друг другу. При этом происходит присваивание ссылок, а не элементов:

```
int[] a = new int[10];
int[] b = a;        // b и a указывают на один и тот же массив
```

- Для удобства работы с массивами в C# существует базовый класс Array, в котором собраны основные методы обработки массивов.
- В лабораторной работе 1 методы класса Array **не используются**. Цель работы – написание собственных методов для выполнения требуемых функций.

Пример (не лучший способ)

Для массива, состоящего из 6 целочисленных элементов, программа определяет:

- сумму и количество отрицательных элементов;
- максимальный элемент.

Программа

```
const int n = 6;
int[] a = new int[n] { 3, 12, 5, -9, 8, -4 };

Console.WriteLine( "Исходный массив:" );
for ( int i = 0; i < n; ++i ) Console.Write( "\t" + a[i] );
Console.WriteLine();

long sum = 0;            // сумма отрицательных элементов
int  num = 0;            // количество отрицательных элементов
for ( int i = 0; i < n; ++i )
      if ( a[i] < 0 ) {
       sum += a[i];  ++num;
      }
 Console.WriteLine( "Сумма  отрицательных = " + sum );
 Console.WriteLine( "Кол-во отрицательных = " + num );

 int max = a[0];          // максимальный элемент
 for ( int i = 0; i < n; ++i )
         if ( a[i] > max ) max = a[i];
 Console.WriteLine( "Максимальный элемент = " + max );
```

Оператор foreach (упрощенно)

- Применяется для перебора элементов массива. Синтаксис:

foreach (тип имя **in** имя_массива **) тело_цикла**

- *Имя* задает локальную по отношению к циклу переменную, которая будет по очереди принимать все значения из массива, например:

```
int[] massiv =  { 24, 50, 18, 3, 16, -7, 9, -1 };
foreach ( int x in massiv ) Console.WriteLine( x );
```

Оператор foreach служит только для перебора и просмотра значений в некоторой коллекции. Изменять данные в коллекции с помощью него нельзя.

Программа с использованием foreach

```
int[] a = { 3, 12, 5, -9, 8, -4 };
Console.WriteLine( "Исходный массив:" );
foreach ( int elem in a )
    Console.Write( "\t" + elem );
Console.WriteLine();

long sum = 0;      // сумма отрицательных элементов
int  num = 0;      // количество отрицательных элементов
foreach ( int elem in a )
      if ( elem < 0 ) {
          sum += elem;  ++num;
      }
Console.WriteLine( "sum = " + sum );
Console.WriteLine( "num = " + num );

int max = a[0];          // максимальный элемент
foreach ( int elem in a )
     if ( elem > max ) max = elem;
Console.WriteLine( "max = " + max );
```

```
for ( int i = 0; i < n; ++i )
      if ( a[i] < 0 ) {
      sum += a[i];  ++num;
      }
```

Работа с массивом, размерность которого вводится в процессе работы программы

Пример 5. Даны два n -мерных вектора. Найти сумму этих векторов.

```
namespace Vektora
{
    class Program
    {
        static void Main(string[] args)
        {
            int n;

            Console.Write("Введите размерность массивов:");
            n = Convert.ToInt32(Console.ReadLine());

            int[] a = new int[n];
            int[] b = new int[n];
            int[] c = new int[n];

            for (int i = 0; i <= n-1; i++)
            {
                Console.Write("a[{0}]=", i);
                a[i] = Convert.ToInt32(Console.ReadLine());
            }
            for (int i = 0; i <= n - 1; i++)
            {
                Console.Write("b[{0}]=", i);
                b[i] = Convert.ToInt32(Console.ReadLine());
            }
            for (int i = 0; i <= n - 1; i++)
            {
                c[i] = a[i] + b[i];
                Console.WriteLine(c[i]);
            }
            Console.ReadKey();
        }
    }
```

Формирование значений элементов массива случайным образом

```
Random random = new Random();

   for (int i=0;i<=n-1;i++)/*пробегая последовательно элементы  массива*/
            X[i]= random.Next(1,50); /* запишем туда случайное число, кото
                                рое сформирует компьютер в диапазоне [1..50]*/
```

Обмен значений двух переменных