

ЛАБОРАТОРНАЯ РАБОТА № 5

Одномерные массивы

Краткие теоретические сведения

Массивом называется последовательная группа переменных одного типа. Массивы служат для размещения набора данных, которые необходимо сохранить и использовать в процессе выполнения программы.

Элементы массива имеют одно и то же имя, но различаются порядковым номером (индексом), что позволяет компактно записывать множество операций с помощью циклов. В языке C#, как и во многих других языках, индексы задаются целочисленным типом.

Число индексов характеризует размерность массива. Каждый индекс изменяется в некотором диапазоне $[a, b]$, который называется граничной парой, где a – нижняя граница, а b – верхняя граница индекса. При объявлении массива границы задаются выражениями. Если все границы заданы константными выражениями, то число элементов массива известно в момент его объявления и ему может быть выделена память еще на этапе трансляции. Такие массивы называются статическими. Если же выражения, задающие границы, зависят от переменных, то такие массивы называются динамическими.

Язык C# поддерживает два вида или две категории типов: типы значений (value types) и типы ссылок (reference types). Элементами массива могут быть как значения, так и ссылки. Массив значимых типов хранит значения, массив ссылочных типов – ссылки на элементы. Всем элементам при создании массива присваиваются значения по умолчанию: нули для значимых типов и `null` – для

ссылочных. Массивы ссылочного типа являются массивами динамическими и память им отводится динамически в области памяти, называемой «кучей» (heap).

Массивами в С# можно пользоваться практически так же, как и в других языках программирования. Тем не менее у них имеется одна особенность: они реализованы в виде объектов. Реализация массивов в виде объектов дает ряд существенных преимуществ, и далеко не самым последним среди них является возможность утилизировать неиспользуемые массивы посредством "сборки мусора".

Поскольку в С# массивы реализованы в виде объектов, то для того чтобы воспользоваться массивом в программе, требуется двух-этапная процедура. Во-первых, необходимо объявить переменную, которая может обращаться к массиву:

```
тип[ ] имя_массива;
```

во-вторых, нужно создать экземпляр массива, используя оператор new:

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

где **тип** объявляет конкретный тип элемента массива, а **размер** определяет число элементов массива. Тип элемента определяет тип данных каждого элемента, составляющего массив. Квадратные скобки указывают на то, что объявляется одномерный массив.

Здесь необходимо отметить, что в отличии от других языков программирования (С, С++, Fortran или VBA), квадратные (или круглые) скобки следуют после названия типа, а не имени массива.

Рассмотрим конкретный пример. В приведенной ниже строке кода создается массив типа `int` из десяти элементов, и переменная `array`, которая является ссылкой на массив типа `int[ ]` с элементами типа `int`.

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

В переменной `array` 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`.

Приведенное выше объявление массива можно разделить на два отдельных оператора. Например:

```
int[] array;           // объявление массива с
                        // именем array
array = new int[10];    // резервирование памяти для
                        // 10 чисел типа int
```

Доступ к отдельному элементу массива осуществляется по индексу. В языке C# индекс первого элемента всех массивов является нулевым. В частности, массив `array` 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 `array` обозначается как `array[0]`, а последний его элемент – как `array[9]`. Ниже приведен пример программы, в которой заполняются все элементы массива `iArray` и массива `chArray`.

Пример 1.

```
// Заполнение массивов
using System;
namespace Example5
{
    class Example5_1
    {
```

```

static void Main()
{
    int j;
    Console.WriteLine("\n\n Одномерный
массив iArray");
    int[] iArray = new int[10];
    for (j = 0; j < 10; j++)
        iArray[j] = j * j;
    // присваивание значений
    // элементам в цикле
    for (j = 0; j < 10; j++)
        // вывод элементов
        Console.WriteLine("\n " + j + " "
+ iArray[j]);
    Console.WriteLine("\n Одномерный
массив chArray с инициализацией");
    char[] chArray =
    { 'a', 'b', 'c', 'd' };
    // Объявление с инициализацией
    j = -1;
    do
    {
        j++;
        Console.WriteLine("\n " +
j + " " + chArray[j]);
    }
    while (chArray[j] != 'd');
}

```

```

        // вывод элементов массива
        Console.WriteLine();
        Console.Write("\n Значения
        присвоены ");
        Console.WriteLine("не всем элемента
        массива iiArray \n");
        int[] iiArray = new int[10];
        for (j = 0; j < 6; j++)
            iiArray[j] = j * j;
        iiArray[9] = 81;
        foreach (int jj in iiArray)
        { Console.Write(" " + jj); }
        Console.WriteLine("\n\n");
        Console.Write(" ");
    }
}
}

```

В начале программы объявлен массив `iArray` из 8 целых чисел. Потом в цикле присваиваются значения элементам. Аналогичный цикл используется и для вывода значений элементов на экран.

Далее объявляется массив символов `chArray` без указания количества элементов с инициализацией, после чего поэлементно выводится в цикле `do-while`.

Кроме описанных ранее, в C# определен цикл `foreach`. Он перебирает подряд все элементы массива. В программе `foreach` применяется к массиву `chArray`. Выражение `foreach(char jj in iiArray){...}`

показывает, что все элементы массива `iiArray` поочередно присваиваются переменной цикла `char`, тип которой должен соответствовать типу массива. На местах элементов, которым не присвоены значения, цикл `foreach` выводит нули, что демонстрируется на примере массива `iiArray`.

```
C:\WINDOWS\system32\cmd.exe

Одномерный массив iiArray
0 0
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81

Одномерный массив chArray с инициализацией
0 a
1 b
2 c
3 d

Значения присвоены не всем элемента массива iiArray
0 1 4 9 16 25 0 0 0 81

Для продолжения нажмите любую клавишу . . .
```

Следует, однако, иметь в виду, что переменная цикла в операторе `foreach` служит только для вывода. Это означает, что, прис-

ваивая этой переменной новое значение, нельзя изменить содержимое массива.

Инициализация массивов

Как уже отмечалось, массив – это структура данных, содержащая несколько элементов одного типа. В следующих примерах показано создание и инициализация одномерных массивов.

```
// Объявление массива
int[ ] array1 = new int[5];
// Объявление и инициализация массива
int[ ] array2 = new int[ ] {1, 3, 5, 7, 9} ;
// Альтернативный синтаксис
int[ ] array3 = {1, 2, 3, 4, 5, 6} ;
// при инициализации массива его размер можно
// указывать явным образом, но этот размер
// должен совпадать с числом инициализаторов
int[ ] array4 = new int[10]
{99, 10, 100, 18, 1, 78, 22, 69};
```

Ввод-вывод массивов

Заполнить массив, т. е. определить значения элементов массива можно следующими способами:

1. при помощи оператора присваивания;
2. непосредственным вводом с клавиатуры;
3. подготовкой и вводом данных из текстового файла;
4. использования датчика случайных чисел;
5. заполнением массива при помощи стандартных функций

Пример 2.

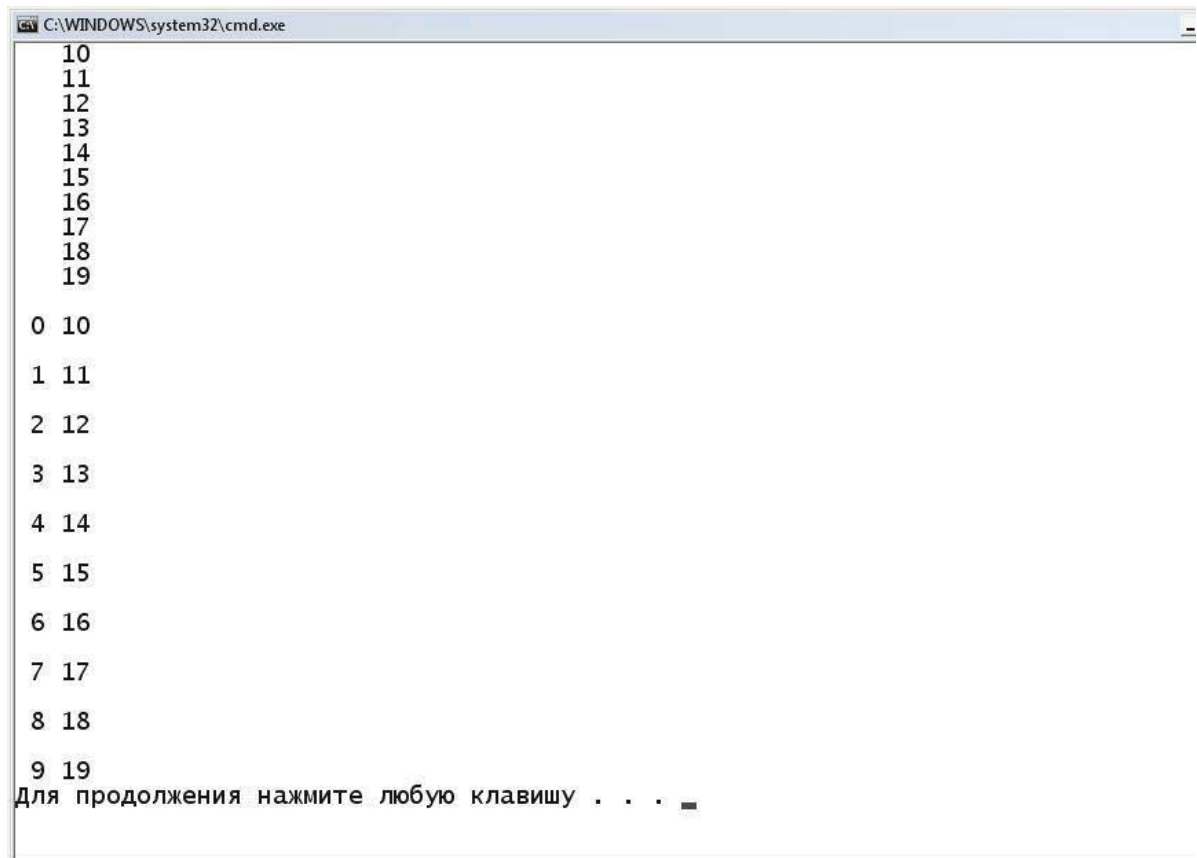
```
// Ввод массива с клавиатуры
using System;
namespace Example5
{
    class Example5_2
    {
        static void Main()
        {
            int j;
            // начальное значение
            string strValue;
            int[] iArray = new int[10];
            for (j = 0; j < 10; j++)
            {
                strValue = Console.ReadLine();
                // ввод и присваивание значений
                iArray[j]= Convert.ToInt32(strValue);
            }
            for (j = 0; j < 10; j++)
                // вывод элементов
                Console.WriteLine("\n " + j + " " +
                    iArray[j]);
        }
    }
}
```


Для организации ввода необходимо объявить строковую переменную, которой присваивается очередное введенное с клавиатуры число. Следующий оператор

```
iArray[j] = Convert.ToInt32(strValue);
```

преобразует строковую переменную `strValue` в целое 32-разрядное число. Ввод и преобразование происходит в цикле, после завершения которого массив `iArray` содержит исходные данные.

Каждый объект (переменная), каждый операнд при вычислении выражения, в том числе и само выражение характеризуется парой (значение, тип), задающей значение выражения и его тип. В процессе вычислений зачастую возникает необходимость преобразования типов – необходимость преобразовать пару (значение1, тип1) к паре (значение2, тип2). Исходная пара называется ис-



```
C:\WINDOWS\system32\cmd.exe
10
11
12
13
14
15
16
17
18
19
0 10
1 11
2 12
3 13
4 14
5 15
6 16
7 17
8 18
9 19
Для продолжения нажмите любую клавишу . . .
```

точником преобразования, заключительная — целью преобразования. Некоторые преобразования типов выполняются автоматически. Такие преобразования называются неявными, и они часто встречаются при вычислении выражений. Все остальные преобразования называются явными и для них существуют разные способы таких преобразований — операция кастинга (приведение к типу), методы специального класса `Convert`, специальные методы `ToString`, `Parse`.

Все скалярные типы (арифметический, логический, символьный) имеют статический метод `Parse`, аргументом которого является строка, а возвращаемым результатом — объект соответствующего типа. Метод явно выполняет преобразование текстового представления в тот тип данных, который был целью вызова статического метода.

Для преобразований внутри арифметического типа можно использовать кастинг — приведение типа. Для преобразований строкового типа в скалярный тип можно применять метод `Parse`, а в обратную сторону — метод `ToString`.

Однако, во всех ситуациях, когда требуется выполнить преобразование из одного базового встроенного типа в другой базовый тип, можно использовать методы универсального класса `Convert`, встроенного в пространство имен `System`.

Методы класса `Convert` поддерживают общий способ выполнения преобразований между типами. Класс `Convert` содержит 15 статических методов вида (`ToInt16()`, `ToInt32()`, `ToInt64()`, ..., `ToDouble`, `ToDecimal`, ...,). Единственным исключением является тип `object` — метода `ToObject` нет по понятным причинам, поскольку для всех типов существует неявное преобра-

зование к типу `object`. Каждый из этих 15 методов перегружен, и его аргумент может принадлежать к любому из упомянутых типов. С учетом перегрузки с помощью методов этого класса можно осуществить любое из возможных преобразований одного типа в другой.

Пример 3.

```
// Заполнение массива с помощью
// генератора случайных чисел
using System;
namespace Example5
{
    class Example5_3
    {
        static void Main()
        {
            int j, num1, num2;
            string str;
            double db1, db2;
            Random rnd = new Random();
            int[] iArray1 = new int[10];
            int[] iArray2 = new int[10];
            double[] dArray1 = new double[10];
            double[] dArray2 = new double[10];
            for (j = 0; j < 10; j++)
            {
                iArray1[j] = rnd.Next(1, 101);
                iArray2[j] = 50 - rnd.Next(1,
```

```

        101);
    }
    for (j = 0; j < 10; j++)
    {
        num1 = rnd.Next(1, 101);
        db1 = Convert.ToDouble(num1);
        dArray1[j] = db1 +
            Convert.ToDouble(rnd.Next(1,
            101)) / 100;
        num2 = 50 - rnd.Next(1, 101);
        db2 = Convert.ToDouble(num2);
        dArray2[j] = db2 -
            Convert.ToDouble(rnd.Next(1,
            101)) / 100;
    }
    Console.WriteLine("\n  -----
    -----");
    Console.WriteLine("\n  Массивы типа int
    Массивы типа double");
    Console.WriteLine("\n  -----
    -----");
    for (j = 0; j < 10; j++)
    {
        str = string.Format("\n {0, 4:D}
        {1, 6:D} {2, 6:D} {3, 8:D}
        {4, 8:F2} {5, 8:F2}",
        j, iArray1[j],

```

```

        iArray2[j], j, dArray1[j],
        dArray2[j]);
        Console.WriteLine(str);
    }
    Console.WriteLine("\n -----
    -----");
    Console.WriteLine();
}
}
}

```

```

C:\WINDOWS\system32\cmd.exe

-----
Массивы типа int      Массивы типа double
-----
0      41      26      0      12,96      -8,33
1      84      -27     1      4,78      0,16
2      8       0      2      4,06     -14,56
3      33      -12     3      50,16     28,13
4      58      -35     4      25,12     0,64
5      6       12      5      31,63     25,65
6      31      -2      6      3,07     -31,31
7      76      -37     7      92,49     43,90
8      41      -26     8      47,13     3,54
9      96      -4      9      67,51     -24,30
-----
Для продолжения нажмите любую клавишу . . .

```

В данном примере для заполнения массива используется генератор случайных чисел. Для генерирования последовательного ряда

случайных чисел служит класс `Random`. Такие последовательности чисел оказываются полезными в самых разных ситуациях включая имитационное моделирование. Начало последовательности случайных чисел определяется некоторым начальным числом, которое может задаваться автоматически или указываться явным образом.

В классе `Random` определяются два конструктора:

```
public Random ()  
public Random(int seed)
```

Первый конструктор создает объект типа `Random`, использующий системное время определения начального числа. А во втором конструкторе используется начальное значение `seed`, задаваемое явным образом.

В первом цикле заполняются массивы `iArray1` и `iArray2`, причем массив `iArray1` заполняется числами от 0 до 100, массив `iArray2` заполняется числами от -50 до 50. В этих же интервалах находятся и числа `num1` и `num2`, которые в следующих строках преобразуются к типу `double`. Оператор

```
Convert.ToDouble(rnd.Next(1, 101)) / 100;
```

генерирует случайные числа, находящиеся в интервале от 0.0 до 1.0. Таким образом, массивы `dArray1` и `dArray2` будут содержать числа типа `double`. Основные методы класса `Random` представлены в таблице:

Метод	Назначение
<code>Public virtual int Next(int upperBound)</code>	Возвращает следующее случайное целое число, которое будет находиться в пределах от 0 до

	<i>upperBound-1</i> включительно
Public virtual int Next (int <i>LowerBound</i> , int <i>upperBound</i>)	Возвращает следующее случайное целое число, которое будет находи- ться в пределах от <i>LowerBound</i> до <i>upperBound-1</i> включительно
Public virtual double NextDouble (int <i>upperBound</i>)	Возвращает следующее случайное число с плавающей точкой, больше или равно 0,0 и меньше 1,0

Пример 4.

```
// Ввод массива в файл и
// вывод массива из файла
using System;
using System.IO;
namespace Example5
{
    class Example5_4
    {
        static void Main()
        {
            int j;
            string strValue;
            int[] iArray1 = new int[10];
            int[] iArray2 = new int[10];
            StreamReader sRead = new
            StreamReader("C:\\\\dat.txt");
            StreamWriter sWrite = new
```

```

        StreamWriter("C:\\res.txt");
        for (j = 0; j < 10; j++)
        {
            strValue = sRead.ReadLine();
            iArray1[j] =
                Convert.ToInt32(strValue);
            iArray2[j] = 10 * iArray1[j];
            strValue = string.Format("\n {0,
                4:D} {1, 6:D} {2, 6:D}",
                j, iArray1[j], iArray2[j]);
            Console.WriteLine(strValue);
            Console.WriteLine();
            sWrite.WriteLine(iArray2[j]);
        }
        sRead.Close();
        sWrite.Close();
    }
}

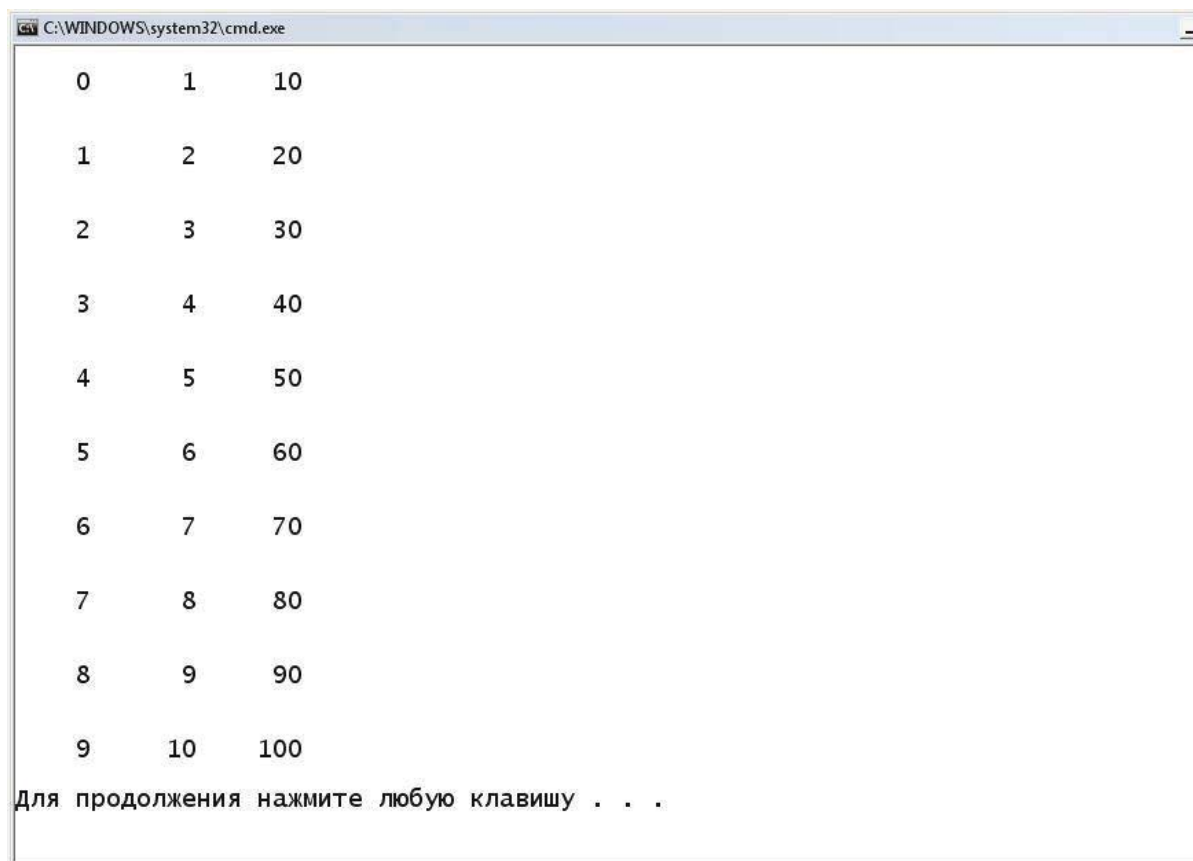
```

В этом примере происходит считывание из файла и запись в файл. Имя файла включает полный путь и, следовательно, содержит служебный знак \ (обратный слеш). Если используемые символы являются служебными, то на экран они не выводятся. Однако, в случае необходимости использовать эти символы в выводимом тексте, следует перед каждым таким символом поставить дополнительный символ \.

Все действия, т. е. считывание из файла и запись в файл, а также вывод на экран организованы в одном цикле. В переменной `strValue` формируется текстовая строка вывода на экран, содержащая форматы, заключенные в фигурные скобки

```
"\n {0, 4:D} {1, 6:D} {2, 6:D}",
```

где в первой позиции расположены цифры 0, 1 и 2, которые являются соответственно указателями на `j`, `iArray1`, `iArray2`. Код форматирования `4:D` определяет число позиций (4), отведенных в строке вывода для данного целого числа (`D` – формат типа `decimal`).



The screenshot shows a Windows command prompt window with the title bar "C:\WINDOWS\system32\cmd.exe". The window displays a table of numbers with three columns. The first column contains integers from 0 to 9. The second column contains integers from 1 to 10. The third column contains integers from 10 to 100. Below the table, there is a prompt message: "Для продолжения нажмите любую клавишу . . .".

0	1	10
1	2	20
2	3	30
3	4	40
4	5	50
5	6	60
6	7	70
7	8	80
8	9	90
9	10	100

Для продолжения нажмите любую клавишу . . .

Пример 5.

// Вычислить сумму и среднеарифметическое всех

```

// элементов одномерного массива,
// состоящего из 15 элементов.
// Заполнение массива происходит при
// помощи генератора случайных чисел
using System;
namespace Example5
{
    class Example5_5
    {
        static void Main()
        {
            int j, num;
            string str;
            string str1 = "Сумма", str2 =
                "Сумма";
            string str3 = "СрАрф", str4 =
                "СрАрф";
            double db1, db2;
            double sum1 = 0, sum2 = 0,
                sum3 = 0, sum4 = 0;
            Random rnd = new Random();
            int[] iArray1 = new int[15];
            int[] iArray2 = new int[15];
            double[] dArray1 = new double[15];
            double[] dArray2 = new double[15];
            for (j = 0; j < 15; j++)
            {

```

```

        iArray1[j] = rnd.Next(1, 101);
        iArray2[j] = 50 - rnd.Next(1,
            101);
        sum1 += iArray1[j];
        sum2 += iArray2[j];
    }
    for (j = 0; j < 15; j++)
    {
        num = rnd.Next(1, 101);
        db1 = Convert.ToDouble(num);
        dArray1[j] = db1 +
            Convert.ToDouble(rnd.Next(1,
                101)) / 100;
        num = 50 - rnd.Next(1, 101);
        db2 = Convert.ToDouble(num);
        dArray2[j] = db2 -
            Convert.ToDouble(rnd.Next(1,
                101)) / 100;
        sum3 += dArray1[j];
        sum4 += dArray2[j];
    }
    Console.WriteLine("\n  -----
    -----
    -----");
    Console.WriteLine("\n
    Массивы типа int                Массивы
    типа double");

```

```

Console.WriteLine("\n -----
-----
-----");
for (j = 0; j < 15; j++)
{
    str = string.Format("\n {0, 10:D}
    {1, 10:D} {2, 10:D} {3, 10:D}
    {4, 10:F2} {5, 10:F2}",
    j, iArray1[j], iArray2[j],
    j, dArray1[j], dArray2[j]);
    Console.WriteLine(str);
}
Console.WriteLine("\n -----
-----
-----");
Console.WriteLine("\n {0, 10} {1, 10}
{2, 10} {3, 10}
{4, 10:F2} {5, 10:F2}",
str1, sum1, sum2,
str2, sum3, sum4);
Console.WriteLine("\n {0, 10}
{1, 10:F2}
{2, 10:F2} {3, 10}
{4, 10:F2} {5, 10:F2}",
str3, sum1 / 15, sum2 /
15, str4, sum3 /
15, sum4 / 15);

```

```

        Console.WriteLine();
    }
}
}

```

Массивы типа int			Массивы типа decimal		
0	86	13	0	3,53	3,68
1	86	6	1	88,49	-12,13
2	53	20	2	40,27	-48,47
3	92	29	3	75,48	-49,61
4	69	-38	4	48,65	10,96
5	71	47	5	71,92	-50,40
6	100	-21	6	72,20	-49,98
7	3	2	7	13,22	-23,71
8	81	-5	8	9,82	-35,19
9	55	-35	9	58,68	-34,55
10	21	-17	10	86,69	33,72
11	38	-39	11	88,74	-43,91
12	31	19	12	91,70	-34,92
13	57	29	13	82,09	32,88
14	13	-38	14	53,06	-11,68
Сумма	856	-28	Сумма	884,54	-313,31
СрАрф	57,07	-1,87	СрАрф	58,97	-20,89

Для продолжения нажмите любую клавишу . . .

Массивы типа `int` `iArray1`, `iArray2`, а также типа `double` `dArray1`, `dArray2` заполняются генератором случайных чисел. В первом цикле случайными числами инициализируется массивы це-

лого типа и одновременно подсчитывается сумма элементов обоих массивов – `sum1` и `sum2`. Переменная этого цикла одновременно является индексом массива. На каждом шаге цикла к сумме элементов каждого массива добавляется очередной элемент. По окончании цикла переменные `sum1` и `sum2` будут содержать полную сумму всех элементов массивов `iArray1`, `iArray2`.

В втором цикле инициализируются массивы чисел с плавающей точкой и одновременно по аналогичной схеме подсчитывается сумма элементов массивов `dArray1`, `dArray2` – `sum3` и `sum4`.

Далее формируется заголовок таблицы вывода, которая заполняется в третьем цикле. Для массивов типа `int` использован формат `D`, а для массивов типа `double` – формат `F` (fixed point).

Пример 6.

```
// Вычислить сумму всех четных элементов
// одномерного массива, состоящих из 10 элементов
// Заполнение одномерного массива происходит при
// помощи генератора случайных чисел
using System;
namespace Example5
{
    class Example5_6
    {
        static void Main()
        {
            int j, num, sum = 0;
            Random rnd = new Random();
            int[] iArray = new int[10];
```

```

        for (j = 0; j < 10; j++)
        {
            iArray[j] = rnd.Next(1, 101);
        }
        for (j = 0; j < 10; j++)
        {
            num = Convert.ToInt32(iArray[j]
            % 2);
            if (num == 0) sum += iArray[j];
        }
        foreach (int jj in iArray)
        { Console.Write(" " + jj); }
        Console.WriteLine("\n\n");
        Console.WriteLine("\n Сумма четных
        элементов = " + sum);
        Console.WriteLine();
        Console.Write(" ");
    }
}
}

```

Отличие данного примера от предыдущего заключается в том, что требуется подсчитать сумму не всех элементов массива, а только четных

```

num = Convert.ToInt32(iArray[j] % 2);
if (num == 0) sum += iArray[j];

```

Сначала определяется остаток от деления элемента массива на 2 с помощью выражения `iArray[j] % 2`, результат которого прео-

бразуется к целому типу. Если выполняется условие `num == 0`, то значение элемента `iArray[j]` является величиной четной и происходит суммирование данного элемента с переменной `sum`.



```
C:\WINDOWS\system32\cmd.exe
90 31 16 67 24 98 92 88 98 77

Сумма четных элементов = 506
Для продолжения нажмите любую клавишу . . . _
```

Пример 7.

```
// Определить индексы второго положительного
// и третьего отрицательного элементов
// одномерного массива, состоящих из 20 элементов
// Заполнение массива происходит
// при помощи генератора случайных чисел
using System;
namespace Example5
{
    class Example5_7
    {
        static void Main()
        {
            int j, jnum = 0;
            Random rnd = new Random();
            int[] iArray = new int[20];
            for (j = 0; j < 20; j++)
            {
                iArray[j] = 50 - rnd.Next(1,
```



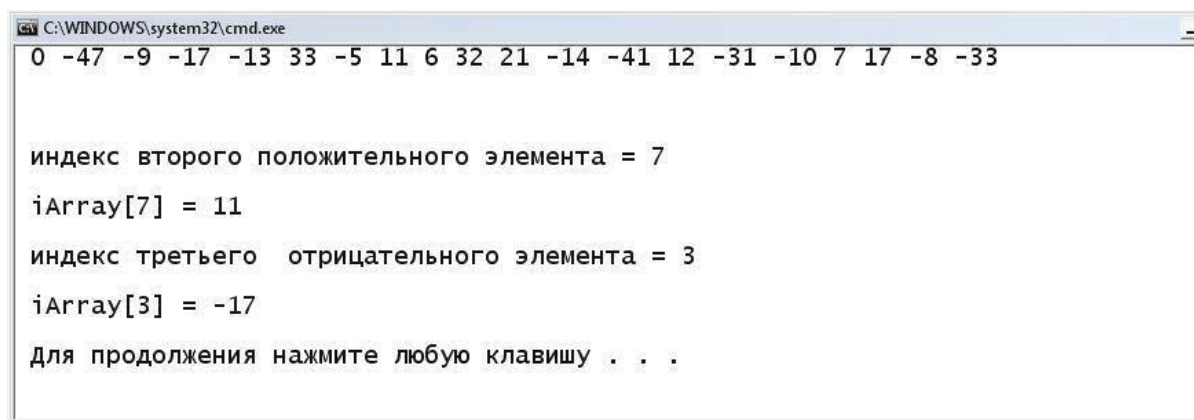
```

        101);
    }
    for (j = 0; j < 20; j++)
    {
        if (iArray[j] > 0) jnum += 1;
        if (jnum == 2) break;
    }
    foreach (int jj in iArray)
    { Console.Write(" " + jj); }
    Console.WriteLine("\n\n");
    Console.WriteLine("\n индекс второго
    положительного элемента = " + j);
    Console.WriteLine("\n iArray[" +
    j + "] = " + iArray[j]);
    jnum = 0;
    for (j = 0; j < 20; j++)
    {
        if (iArray[j] < 0) jnum += 1;
        if (jnum == 3) break;
    }
    Console.WriteLine("\n индекс третьего
    отрицательного элемента = " + j);
    Console.WriteLine("\n iArray[" +
    j + "] = " + iArray[j]);
    Console.WriteLine();
    Console.Write(" ");
}

```

```
}  
}
```

Поиск второго положительного элемента массива `iArray` происходит в цикле. Переменная `jnum` определяет количество положительных элементов, которые встретились при выполнении цикла. Как только значение выражения `jnum == 2` будет равно `true` оператор `break` прерывает выполнение цикла.



```
C:\WINDOWS\system32\cmd.exe  
0 -47 -9 -17 -13 33 -5 11 6 32 21 -14 -41 12 -31 -10 7 17 -8 -33  
  
индекс второго положительного элемента = 7  
iArray[7] = 11  
индекс третьего отрицательного элемента = 3  
iArray[3] = -17  
Для продолжения нажмите любую клавишу . . .
```

Аналогичный алгоритм используется при нахождении третьего отрицательного элемента.

Пример 8.

```
// Задан одномерный массив размером N.  
// Сформировать два массива, включив  
// в первый - четные элементы исходного  
// массива, а во второй - нечетные элементы.  
// Отсортировать массивы в порядке возрастания.  
// Заполнение массива происходит при помощи  
// генератора случайных чисел
```

```

using System;
namespace Example5
{
    class Example5_8
    {
        static void Main()
        {
            int jnum = 0, N = 20;
            int jAA = 0, jBB = 0;
            int j, k, temp;
            Random rnd = new Random();
            int[] iArray = new int[N];
            int[] jA = new int[N];
            int[] jB = new int[N];
            for (j = 0; j < N; j++)
            {
                iArray[j] = rnd.Next(1, 101);
            }
            Console.WriteLine("\n исходный массив
\n\n");
            foreach (int jj in iArray)
            { Console.Write(" " + jj); }
            Console.WriteLine("\n\n");
            for (j = 0; j < N; j++)

```

```

{
    jnum = iArray[j] / 2;
    iArray[j] =
    Convert.ToInt32(iArray[j]);
    if (iArray[j] == jnum * 2)
    {
        jA[jAA] = iArray[j];
        jAA += 1;
    }
    else
    {
        jB[jBB] = iArray[j];
        jBB += 1;
    }
}

Console.WriteLine("\n массив A[]
\n\n");
foreach (int jj in jA)
{ Console.Write(" " + jj); }
Console.WriteLine("\n\n");
Console.WriteLine("\n массив B[]
\n\n");
foreach (int jj in jB)
{ Console.Write(" " + jj); }

```

```

Console.WriteLine("\n\n");
// Сортировка массива A
jAA -= 1;
for (k = 0; k < jAA; k++)
{
    for (j = 0; j < jAA; j++)
    {
        if (jA[j + 1] < jA[j])
        {
            temp = jA[j];
            jA[j] = jA[j + 1];
            jA[j + 1] = temp;
        }
    }
}
// Сортировка массива B
jBB -= 1;
for (k = 0; k < jBB; k++)
{
    for (j = 0; j < jBB; j++)
    {
        if (jB[j + 1] < jB[j])
        {
            temp = jB[j];

```

```

        jB[j] = jB[j + 1];
        jB[j + 1] = temp;
    }
}
}
Console.WriteLine("\n отсортированный
массив A[] \n\n");
foreach (int jj in jA)
{ Console.Write(" " + jj); }
Console.WriteLine("\n\n");
Console.WriteLine("\n отсортированный
массив B[] \n\n");
foreach (int jj in jB)
{ Console.Write(" " + jj); }
Console.WriteLine("\n\n");
}
}
}

```

Для определения четности или нечетности очередного элемента массива. Если результат данного преобразования равен исходному значению, то данный элемент массива содержит четное число, В противном случае – число нечетное. По окончании данной процедуры значения всех элементов массива jA – четные числа, массива jA – нечетные числа.

Далее происходит сортировка элементов массивов jA и jB по возрастанию. Процесс сортировки происходит следующим образом: если элемент с индексом j больше элемента с индексом $j+1$, то выполняется процедура перестановки

```
C:\WINDOWS\system32\cmd.exe

исходный массив

24 96 28 67 12 50 91 11 93 61 87 60 60 15 79 86 45 51 48 13

массив A[]

24 96 28 12 50 60 60 86 48 0 0 0 0 0 0 0 0 0 0 0

массив B[]

67 91 11 93 61 87 15 79 45 51 13 0 0 0 0 0 0 0 0 0

отсортированный массив A[]

12 24 28 48 50 60 60 86 96 0 0 0 0 0 0 0 0 0 0 0

отсортированный массив B[]

11 13 15 45 51 61 67 79 87 91 93 0 0 0 0 0 0 0 0 0

Для продолжения нажмите любую клавишу . . .
```

```
temp = jB[j];
jB[j] = jB[j + 1];
jB[j + 1] = temp;
```

и завершении цикла массивы jA и jB будут отсортированы по возрастанию.

2. Практическая часть

Задание к лабораторной работе

Для заданного условия составить процедуру и придумать несколько наборов тестовых данных для отладки. Возможно использование как статических массивов, так и динамических. Ввод исходных данных осуществить из файла или с клавиатуры.

1. Задан одномерный массив размером N . Определить количество отрицательных чисел, количество положительных чисел и среднее арифметическое всех чисел.
2. Задан одномерный массив размером N . Сформировать два массива размером $N/2$, включая в первый элементы исходного массива с четными индексами, а во второй – с нечетными. Вычислить суммы элементов каждого из массивов.
3. Определить среднее арифметическое значение первого отрицательного и последнего положительного элементов одномерного массива, размер которого равен M .
4. Определить число отрицательных элементов, расположенных перед наибольшим положительным элементом одномерного массива, размер которого равен M .
5. В заданном одномерном массиве размером N поменять местами первый и последний положительные элементы.
6. Написать программу для вычисления суммы и среднего арифметического значения всех элементов заданного одномерного массива A , состоящего из 10-ти элементов.
7. Написать программу для вычисления суммы положительных элементов, заданного массива A , состоящего из 20-ти элементов.

8. Написать программу для вычисления суммы четных элементов заданного массива A , состоящего из 20-ти элементов.
9. Написать программу для определения количества положительных элементов, заданного массива A , состоящего из 20-ти элементов.
10. Написать программу для определения индексов положительных элементов, заданного массива A , состоящего из 20-ти элементов.
11. Написать программу для вычисления среднего арифметического значения всех элементов заданного массива D . Для отрицательных элементов использовать их абсолютные значения.
12. Написать программу для поиска в заданном массиве B , состоящем из 10-ти элементов, третьего положительного элемента и его индекса. Известно, что хотя бы один положительный элемент в массиве B имеется.
13. Написать программу поиска в заданном массиве B , состоящем из 20-ти элементов, третьего положительного элемента и его индекса.
14. Написать программу для поиска в заданном массиве $A(15)$ наибольшего значения элемента и его индекса.
15. Написать программу, в которой производится перестановка четных и нечетных элементов, заданного массива C .
16. Для заданного массива A , состоящего не более чем из 50-ти элементов, найти наименьший элемент и переставить его со вторым по порядку отрицательным элементом массива.
17. Написать программу по упорядочению элементов заданного массива B в следующем порядке: сначала идут положительные числа, потом – нули и в конце – отрицательные.

18. Написать программу сортировки по возрастанию заданного массива В, состоящего из 10-ти элементов.
19. Написать программу. Для заданного массива В, состоящего из 10-ти элементов, изменить порядок следования его элементов на обратный.
20. Написать программу, в которой для заданного массива В, состоящего из 10-ти элементов, его элементы перемещались бы например на 7 позиций вправо. При этом 7 элементов из конца массива перемещаются в начало.
21. Задан массив А. Поместить положительные элементы этого массива в массив В, а отрицательные элементы – в массив С.
22. В заданном векторе (одномерном массиве) найти сумму первого и последнего отрицательного элемента
23. В заданном векторе (одномерном массиве) найти: разность первого и последнего нечетного элементов
24. В заданном векторе (одномерном массиве) найти: индексы наименьшего и наибольшего из элементов
25. В заданном векторе (одномерном массиве) найти: произведение трех наименьших элементов вектора
26. В заданном векторе (одномерном массиве) найти: сумму элементов вектора с четными индексами
27. В заданном векторе (одномерном массиве) найти: разность первого положительного и последнего отрицательного элемента
28. В заданном векторе (одномерном массиве) найти: число отрицательных элементов с нечетными индексами
29. В заданном векторе (одномерном массиве) найти: число элементов, расположенных после его наибольшего отрицательного элемента.

30. В заданном векторе (одномерном массиве) найти: наибольший отрицательный и наименьший положительные элементы.