

**УО «МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ А.А. КУЛЕШОВА»  
СОЦИАЛЬНО-ГУМАНИТАРНЫЙ КОЛЛЕДЖ**



**Дисциплина  
«Конструирование программ и языки программирования»**

**Разработка программ с использованием массивов  
(4 часа)**

Методические рекомендации к лабораторной работе № 6

Могилев

Понятие «Массив». Методические указания по лабораторной работе № 6 по дисциплине «Конструирование программ и языки программирования». Для учащихся 3 курса очной формы обучения специальности 2–40 01 01 «Программное обеспечение информационных технологий».

## Оглавление

1 Цель работы .....	4
2 Ход работы.....	5
3 Краткие теоретические сведения .....	6
3.1 Краткие теоретические сведения.....	6
3.2 Инициализация массивов .....	9
3.3 Ввод-вывод массивов.....	9
3.4 Многомерные массивы .....	21
3.5 Массивы массивов.....	22
4 Задания .....	24
5 Контрольные вопросы .....	30

## **1 Цель работы**

выработать умение разрабатывать программы с использованием массивов.

## **2 Ход работы**

1. Изучение теоретического материала.
2. Выполнение практических индивидуальных заданий по вариантам (вариант уточняйте у преподавателя).
3. Оформление отчета.
  - 3.1. Отчет оформляется индивидуально каждым студентом. Отчет должен содержать задание, алгоритм и листинг программы.
  - 3.2. Отчет по лабораторной работе выполняется на листах формата А4. В состав отчета входят:
    - 1) титульный лист;
    - 2) цель работы;
    - 3) текст индивидуального задания;
    - 4) выполнение индивидуального задания.
4. Контрольные вопросы.

### 3 Краткие теоретические сведения

#### 3.1 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 C# 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 C# массивы реализованы в виде объектов, то для того чтобы воспользоваться массивом в программе, требуется двухэтапная 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 отличие от других языков программирования (C, C++, 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\nОдномерный 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 `iiArray` 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 С# 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`.



```

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

```

Следует, однако, иметь в виду, что переменная цикла в операторе `foreach` служит только для вывода. Это означает, что, присваивая этой переменной новое значение, нельзя изменить содержимое массива.

### 3.2 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};

```

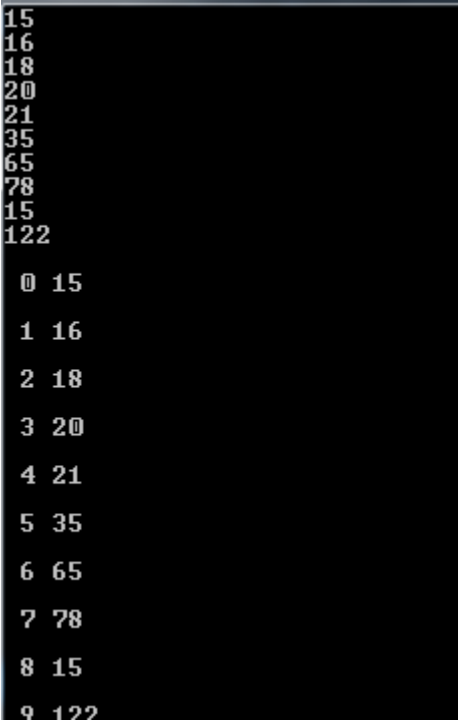
### 3.3 Ввод-вывод массивов

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]);
            }
        }
    }
}
```



```
15
16
18
20
21
35
65
78
15
122

0 15
1 16
2 18
3 20
4 21
5 35
6 65
7 78
8 15
9 122
```

Для организации ввода необходимо объявить строковую переменную, которой присваивается очередное введенное с клавиатуры число. Следующий оператор `iArray[j] = Convert.` вычислений зачастую возникает необходимость преобразования типов - необходимость преобразовать пару (значение1, тип1) к паре (значение2, тип2). Исходная пара называется источником преобразования, `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.. 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 преобразование 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();
Console.ReadLine();
}
}
}

```

-----					
Массивы типа int			Массивы типа double		
0	88	-10	0	13,87	43,95
1	25	43	1	37,91	38,66
2	18	35	2	26,56	41,32
3	83	25	3	27,03	30,54
4	84	14	4	19,33	24,17
5	84	-45	5	46,30	-8,05
6	79	0	6	98,30	32,18
7	86	7	7	8,61	41,14
8	14	45	8	79,26	-14,01
9	21	-28	9	54,20	-20,27

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

#### Пример 4.

```
// 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_4
    {
        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, dAr-
ray1[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.ReadLine();
    }
}
}

```

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

Во 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 5.

```
// 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_5
    {
        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(" ");
            Console.ReadLine();
        }
    }
}
```

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



```
29 80 45 60 57 93 41 54 74 41
```

```
Сумма четных элементов = 268
```

#### Пример 6.

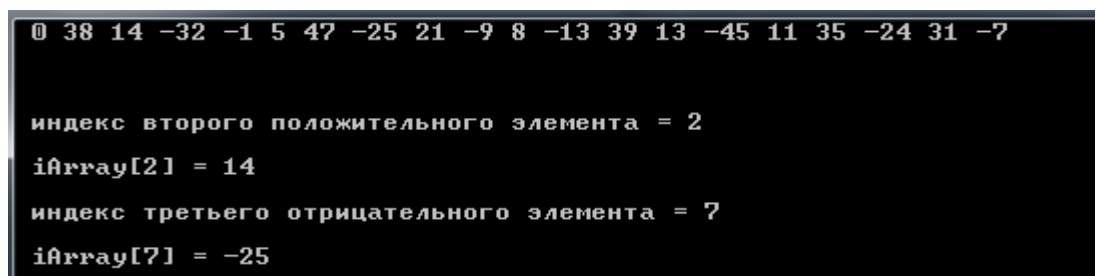
```
// Определить индексы второго положительного
// и третьего отрицательного элементов
// одномерного массива, состоящих из 20 элементов
// Заполнение массива происходит
// при помощи генератора случайных чисел
using System;
namespace Example5
{
    class Example5_6
    {
        static void Main()
        {
            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 элемен-
та = " + j);
```

```

        Console.WriteLine("\n iArray[" + j + "] = " + iArray[j]);
        Console.WriteLine();
        Console.Write(" ");
        Console.ReadLine();
    }
}
}

```

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.



```

0 38 14 -32 -1 5 47 -25 21 -9 8 -13 39 13 -45 11 35 -24 31 -7

индекс второго положительного элемента = 2
iArray[2] = 14
индекс третьего отрицательного элемента = 7
iArray[7] = -25

```

Аналогичный алгоритм используется при нахождении третьего отрицательного элемента.

#### Пример 7.

```

// Задан одномерный массив размером N.
// Сформировать два массива, включи 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_7
    {
        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");
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

```

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

```

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

```

исходный массив
45 32 93 27 13 98 7 82 44 80 67 32 75 18 41 24 31 88 15 11

массив A[]
32 98 82 44 80 32 18 24 88 0 0 0 0 0 0 0 0 0 0 0

массив B[]
45 93 27 13 7 67 75 41 31 15 11 0 0 0 0 0 0 0 0 0

отсортированный массив A[]
18 24 32 32 44 80 82 88 98 0 0 0 0 0 0 0 0 0 0 0

отсортированный массив B[]
7 11 13 15 27 31 41 45 67 75 93 0 0 0 0 0 0 0 0 0

```

### 3.4 Многомерные массивы

Разделение массивов на одномерные и многомерные носит исторический характер. Никакой принципиальной разницы между ними нет. Одномерные массивы – это частный случай многомерных. Можно говорить и по-другому: многомерные массивы являются естественным обобщением одномерных. Одномерные массивы позволяют задавать такие математические структуры как векторы, двумерные – матрицы, трехмерные – кубы данных, массивы большей размерности – многомерные кубы данных.

В чем особенность объявления многомерного массива? Как в типе указать размерность массива? Это делается достаточно просто, за счет использования запятых. Вот как выглядит объявление многомерного массива в общем случае:

```
<тип>[, ... ,] <объявители>;
```

Число запятых, увеличенное на единицу, и задает размерность массива. Что касается объявителей, то все, что сказано для одномерных массивов, справедливо и для многомерных. Можно лишь отметить, что хотя явная инициализация с использованием многомерных константных массивов возможна, но применяется редко из-за громоздкости такой структуры. Проще инициализацию реализовать программно, но иногда она все же применяется. Вот пример:

```

public void TestMultiArr()
{
    int[,] matrix = { { 1, 2 }, { 3, 4 } };
    int [,] dM;
    int n = Convert.ToInt32(Console.ReadLine()); int m = Con-
    vert.ToInt32(Console.ReadLine());
    dM = new int[n,m];
    for(int i = 0; i < n; i++)
    {

```

```

for(int j = 0; j < m; j++)
{
    dM[i, j] = Convert.ToInt32(Console.ReadLine());
}
}
int max = dM[0, 0];
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < m; j++)
    {
        if (dM[i, j] > max)
        {
            max = dM[i, j];
        }
    }
}
Console.WriteLine("Max. element - {0}", max);
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < m; j++)
    {
        Console.Write("{0} ", dM[i, j]);
    }
    Console.WriteLine();
}
}

```

### 3.5 Массивы массивов

Еще одним видом массивов C# являются массивы массивов, называемые также изрезанными массивами (jagged arrays). Такой массив массивов можно рассматривать как одномерный массив, элементы которого являются массивами, элементы которых, в свою очередь, снова могут быть массивами, и так может продолжаться до некоторого уровня вложенности.

В каких ситуациях может возникать необходимость в таких структурах данных? Эти массивы могут применяться для представления деревьев, у которых узлы могут иметь произвольное число потомков. Таковым может быть, например, генеалогическое дерево. Вершины первого уровня - Fathers, представляющие отцов, могут задаваться одномерным массивом, так что Fathers[i] – это i-й отец. Вершины второго уровня представляются массивом массивов – Children, так что Children[i] – это массив детей i-го отца, а Children[i][j] – это j-й ребенок i-го отца. Для представления внуков понадобится третий уровень, так что GrandChildren [i][j][k] будет представлять k-го внука j-го ребенка i-го отца.

Есть некоторые особенности в объявлении и инициализации таких массивов. Если при объявлении типа многомерных массивов для указания размерности использовались запятые, то для изрезанных массивов применяется более ясная символика – совокупности пар квадратных скобок; например, int[][] задает массив, элементы которого – одномерные

массивы элементов типа `int`.

Сложнее с созданием самих массивов и их инициализацией. Здесь нельзя вызвать конструктор `new int[3][5]`, поскольку он не задает изрезанный массив. Фактически нужно вызывать конструктор для каждого массива на самом нижнем уровне. В этом и состоит сложность объявления таких массивов. Пример:

```
int[][] jagger = new int[3][]  
{  
    new int[] {5,7,9,11},  
    new int[] {2,8},  
    new int[] {6,12,4}  
};
```

Массив `jagger` имеет всего два уровня. Можно считать, что у него три элемента, каждый из которых является массивом. Для каждого такого массива необходимо вызвать конструктор `new`, чтобы создать внутренний массив. В данном примере элементы внутренних массивов получают значение, будучи явно инициализированы константными массивами. Конечно, допустимо и такое объявление:

```
int[][] jagger1 = new int[3][]  
{  
    new int[4], new int[2], new int[3]  
};
```

В этом случае элементы массива получают при инициализации нулевые значения. Реальную инициализацию нужно будет выполнять программным путем. Стоит заметить, что в конструкторе верхнего уровня константу 3 можно опустить и писать просто `new int[][]`.

А вот конструкторы нижнего уровня необходимы. Еще одно важное замечание – динамические массивы возможны и здесь. В общем случае, границы на любом уровне могут быть выражениями, зависящими от переменных. Более того, допустимо, чтобы массивы на нижнем уровне были многомерными.

## 4 Задания

### 1. Выполнить задание по вариантам.

#### Вариант 1

В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить:

- сумму отрицательных элементов массива;
- произведение элементов массива, расположенных между максимальным и минимальным элементами.

Упорядочить элементы массива по возрастанию.

#### Вариант 2

В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить:

- сумму положительных элементов массива;
- произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

Упорядочить элементы массива по убыванию.

#### Вариант 3

В одномерном массиве, состоящем из  $n$  целочисленных элементов, вычислить:

- произведение элементов массива с четными номерами;
- сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом – все отрицательные (элементы, равные нулю, считать положительными).

#### Вариант 4

В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить:

- сумму элементов массива с нечетными номерами;
- сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Сжать массив, удалив из него все элементы, модуль которых не превышает единицу.

Освободившиеся в конце массива элементы заполнить нулями.

#### Вариант 5

В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить:

- максимальный элемент массива;
- сумму элементов массива, расположенных до последнего положительного элемента.

Сжать массив, удалив из него все элементы, модуль которых находится в интервале  $[a, b]$ . Освободившиеся в конце массива элементы заполнить нулями.

#### Вариант 6

В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить:

- минимальный элемент массива;
- сумму элементов массива, расположенных между первым и последним положительными



элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом – все остальные.

#### **Вариант 7**

В одномерном массиве, состоящем из  $n$  целочисленных элементов, вычислить:

- номер максимального элемента массива;
- произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине – элементы, стоявшие в четных позициях.

#### **Вариант 8**

В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить:

- номер минимального элемента массива;
- сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает единицу, а потом – все остальные.

#### **Вариант 9**

В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить:

- максимальный по модулю элемент массива;
- сумму элементов массива, расположенных между первым и вторым положительными элементами.

Преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

#### **Вариант 10**

В одномерном массиве, состоящем из  $n$  целочисленных элементов, вычислить:

- минимальный по модулю элемент массива;
- сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине – элементы, стоявшие в нечетных позициях.

#### **Вариант 11**

В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить:

- номер минимального по модулю элемента массива;
- сумму модулей элементов массива, расположенных после первого отрицательного элемента.

Сжать массив, удалив из него все элементы, величина которых находится в интервале  $[a, b]$ . Освободившиеся в конце массива элементы заполнить нулями.

### Вариант 12

В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить:

- номер максимального по модулю элемента массива;
- сумму элементов массива, расположенных после первого положительного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале  $[a, b]$ , а потом — все остальные.

### Вариант 13

В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить:

- количество элементов массива, лежащих в диапазоне от  $A$  до  $B$ ;
- сумму элементов массива, расположенных после максимального элемента.

Упорядочить элементы массива по убыванию модулей.

### Вариант 14

В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить:

- количество элементов массива, равных нулю;
- сумму элементов массива, расположенных после минимального элемента.

Упорядочить элементы массива по возрастанию модулей.

### Вариант 15

В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить:

- количество отрицательных элементов массива;
- сумму модулей элементов массива, расположенных после минимального по модулю элемента.

Заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию

## 2. Выполнить задание по вариантам.

### Вариант 1

Дана целочисленная прямоугольная матрица. Определить:

- количество строк, не содержащих ни одного нулевого элемента;
- максимальное из чисел, встречающихся в заданной матрице более одного раза.

### Вариант 2

Дана целочисленная прямоугольная матрица. Определить количество столбцов, не содержащих ни одного нулевого элемента.

Характеристикой строки целочисленной матрицы назовем сумму ее положительных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик.

### Вариант 3

Дана целочисленная прямоугольная матрица. Определить:

- количество столбцов, содержащих хотя бы один нулевой элемент;
- номер строки, в которой находится самая длинная серия одинаковых элементов.

#### **Вариант 4**

Дана целочисленная квадратная матрица. Определить:

- произведение элементов в тех строках, которые не содержат отрицательных элементов;
- максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

#### **Вариант 5**

Дана целочисленная квадратная матрица. Определить:

- сумму элементов в тех столбцах, которые не содержат отрицательных элементов;
- минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

#### **Вариант 6**

Для заданной матрицы размером  $8 \times 8$  найти такие  $k$ , при которых  $k$ -я строка матрицы совпадает с  $k$ -м столбцом.

Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.

#### **Вариант 7**

Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик.

Найти сумму элементов в тех столбцах, которые содержат хотя бы один отрицательный элемент.

#### **Вариант 8**

Уплотнить заданную матрицу, удаляя из нее строки и столбцы, заполненные нулями.

Найти номер первой из строк, содержащих хотя бы один положительный элемент.

#### **Вариант 9**

Осуществить циклический сдвиг элементов прямоугольной матрицы на  $n$  элементов вправо или вниз (в зависимости от введенного режима),  $n$  может быть больше количества элементов в строке или столбце.

#### **Вариант 10**

Дана целочисленная прямоугольная матрица. Определить номер первого из столбцов, содержащих хотя бы один нулевой элемент.

Характеристикой строки целочисленной матрицы назовем сумму ее отрицательных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с убыванием характеристик.

#### **Вариант 11**

Упорядочить строки целочисленной прямоугольной матрицы по возрастанию количества одинаковых элементов в каждой строке.

Найти номер первого из столбцов, не содержащих ни одного отрицательного элемента.

### **Вариант 12**

Дана целочисленная прямоугольная матрица. Определить:

- количество строк, содержащих хотя бы один нулевой элемент;
- номер столбца, в котором находится самая длинная серия одинаковых элементов.

### **Вариант 13**

Дана целочисленная квадратная матрица. Определить:

- сумму элементов в тех строках, которые не содержат отрицательных элементов;
- минимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

### **Вариант 14**

Дана целочисленная прямоугольная матрица. Определить:

- количество отрицательных элементов в тех строках, которые содержат хотя бы один нулевой элемент;
- минимальное из чисел, встречающихся в заданной матрице более одного раза.

### **Вариант 15**

Упорядочить строки целочисленной прямоугольной матрицы по убыванию количества одинаковых элементов в каждой строке.

Найти номер первого из столбцов, не содержащих ни одного отрицательного элемента.

**Задание 3.** . Выполнить предложенный вариант задания.

### **Вариант 1.**

Организовать ступенчатый массив, который инициализирует названиям мастей игральных карт (черви, пики, трефы, бубны). Вывести полученный массив на экран.

### **Вариант 2.**

Организовать ступенчатый массив, который инициализирует названиям дней недели. Вывести полученный массив на экран.

### **Вариант 3.**

Организовать ступенчатый массив, который инициализирует названиям месяцев года. Вывести полученный массив на экран.

### **Вариант 4.**

Организовать ступенчатый массив, который инициализирует названиям знаков зодиака. Вывести полученный массив на экран.

### **Вариант 5.**

Организовать ступенчатый массив, который инициализирует названиям времен года. Вывести полученный массив на экран.

### **Вариант 6.**

Организовать ступенчатый массив, который инициализирует названия областных центров республики Беларусь. Вывести полученный массив на экран.

**Вариант 7.**

Организируйте ступенчатый массив, который инициализирует фамилии студентов вашей подгруппы. Выведите полученный массив на экран.

**Вариант 8.**

Организируйте ступенчатый массив, который инициализирует названиям цветов радуги. Выведите полученный массив на экран.

**Вариант 9.**

Организируйте ступенчатый массив, который инициализирует названия стран восточной Европы. Выведите полученный массив на экран.

**Вариант 10.**

Организируйте ступенчатый массив, который инициализирует названия 5 учебных дисциплин, которые вы сейчас изучаете. Выведите полученный массив на экран.

**Вариант 11.**

Организируйте ступенчатый массив, который инициализирует названия вузов города Могилева. Выведите полученный массив на экран.

**Вариант 12.**

Организируйте ступенчатый массив, который инициализирует страны-соседи Республики Беларусь. Выведите полученный массив на экран.

**Вариант 13.**

Организируйте ступенчатый массив, который инициализирует расписание занятий, который проходят в понедельник. Выведите полученный массив на экран.

**Вариант 14.**

Организируйте ступенчатый массив, который инициализирует районный центры Могилевской области. Выведите полученный массив на экран.

**Вариант 15.**

Организируйте ступенчатый массив, который заполняется случайными числами. Выведите полученный массив на экран.

## **5 Контрольные вопросы**

1. Что такое массив?
2. Что такое ступенчатый (зубчатый) массив?
3. Как инициализировать одномерный массив? Многомерный? Ступенчатый?
4. Какой класс служит для генерирования последовательного ряда случайных чисел?
5. Как заполнить массив случайными числами?