

ЛАБОРАТОРНАЯ РАБОТА № 6

Многомерные массивы

Краткие теоретические сведения

В языке C++ все массивы являются статическими; более того, все массивы являются 0-базируемыми. Это означает, что нижняя граница всех индексов массива фиксирована и равна нулю.

В языке C# снято существенное ограничение языка C++ на статичность массивов. Массивы в языке C# являются настоящими динамическими массивами. Они относятся к ссылочным типам и память им отводится динамически в "куче". Однако, не снято ограничение 0-базируемости, хотя во многих задачах гораздо удобнее работать с массивами, у которых нижняя граница не равна нулю.

В языке C++ "классических" многомерных массивов нет. Вместо них введены одномерные массивы и массивы массивов, которые являются более общей структурой данных и позволяют задать не только многомерный куб, но и изрезанную, ступенчатую структуру.

В языке C#, соблюдая преемственность, сохранены одномерные массивы и массивы массивов. В дополнение к ним в язык добавлены многомерные массивы.

Простейшей формой многомерного массива является двумерный массив. Местоположение любого элемента в двумерном массиве обозначается двумя индексами. Такой массив можно предста-

вить в виде таблицы, первый индекс которого указывает на строки данной таблицы, а второй – на ее столбцы.

В следующей строке кода объявляется двумерный массив `integer` размерами `10x15`:

```
int[,] table = new int[10, 15];
```

При объявлении этого массива оба его размера разделяются запятой. В первой части этого объявления синтаксическое обозначение означает, что создается переменная типа ссылки на двумерный массив. Если же память распределяется для массива с помощью оператора `new`, то используется следующее синтаксическое обозначение:

```
int[10, 15]
```

Данное объявление создает массив размерами `10x15`. Для доступа к элементу двумерного массива следует указывать оба индекса, разделив их запятой.

В следующей строке кода элементу массива `myArray` с координатами положения, т. е. индексами `(3,5)` присваивается значение `10`:

```
myArray [3, 5] = 10;
```

В C# допускаются массивы трех и более измерений.

Общая форма объявления многомерного массива.

```
тип[, ..., ] имя_массива = new тип[размер1, размер2,  
... размерN];
```

Например, в приведенном ниже объявлении создается трехмерный целочисленный массив размерами 4x10x3.

```
int [,,] multiArray = new int[4, 10, 3];
```

А в следующем операторе элементу массива `multiArray`, положение которого определяется индексами (2,4,1) присваивается значение 100:

```
multidim[2, 4, 1] = 100;
```

Ниже приведен пример программы, в которой сначала организуется трехмерный массив, содержащий матрицу значений 3x3x3, а затем вычисляется сумма значений диагональных элементов этого массива.

```
// Суммировать значения по одной из диагоналей  
// матрицы 3x3x3.
```

```
using System;  
class Example  
{  
    static void Main()  
    {  
        int [,,] m = new int[3,-3,3];  
        int sum = 0;  
        int n = 1;  
        for (int i = 0; i < 3; i++)  
            for (int j = 0; j < 3; j++)
```

```

        for (int k = 0; k < 3; k++)
            m[i, j, k] = n++;
        sum = m[0, 0, 0] +
            m[1, 1, 1] + m[2, 2, 2];
        Console.WriteLine("Сумма
            значений по диагонали: "
            + sum);
    }
}

```

Результат выполнения этой программы:

Сумма значений по диагонали: 42

Инициализация многомерных массивов

Для инициализации многомерного массива достаточно заключить в фигурные скобки список инициализаторов каждого его размера. Общая форма инициализации двумерного массива имеет следующий вид:

```

тип[, ] имя_массива =
{
    {val, val, val, ..., val},
    {val, val, val, ..., val},
    {val, val, val, ..., val}
};

```

где *val* обозначает инициализирующее значение, а каждый внутренний блок – отдельный ряд. Первое значение в каждом ряду сох-

раняется на первой позиции в массиве, второе значение – на второй позиции и т. д. Блоки инициализаторов разделяются запятыми, а после завершающей эти блоки закрывающей фигурной скобки ставится точка с запятой.

Следующие два примера иллюстрируют возможность инициализации массива непосредственным вводом с клавиатуры и считыванием информации из текстового файла.

Пример 1.

```
// Заполнить двумерный массив
// непосредственно вводом с клавиатуры
using System;
namespace Example6
{
    class Example6_1
    {
        static void Main()
        {
            int i, j;
            string strValue;
            int[,] iArray = new int[3, 4];
            int[,] jArray = new int[3, 4];
            for (i = 0; i < 3; ++i)
            {
                for (j = 0; j < 4; ++j)
                {
                    // ввод и присваивание значений
```

```

        strValue = Console.ReadLine();
        iArray[i, j] =
            Convert.ToInt32(strValue);
    }
}
// вывод значений массива iArray на экран
for (i = 0; i < 3; ++i)
{
    for (j = 0; j < 4; ++j)
    {
        Console.Write(" iArray[" + i + ", " + j + "] = " + iArray[i, j]);
    }
    Console.WriteLine();
}
Console.WriteLine();
// вывод значений массива jArray на экран
for (i = 0; i < 3; ++i)
{
    for (j = 0; j < 4; ++j)
    {
        jArray[i, j] = iArray[i, j] * 10;
        Console.Write(" jArray[" + i + ", " + j + "] = " + jArray[i, j]);
    }
    Console.WriteLine();
}

```

```

        Console.WriteLine();
    }
}

```

```

C:\WINDOWS\system32\cmd.exe
11
12
13
14
21
22
23
24
31
32
33
34
iArray[0, 0] = 11 iArray[0, 1] = 12 iArray[0, 2] = 13 iArray[0, 3] = 14
iArray[1, 0] = 21 iArray[1, 1] = 22 iArray[1, 2] = 23 iArray[1, 3] = 24
iArray[2, 0] = 31 iArray[2, 1] = 32 iArray[2, 2] = 33 iArray[2, 3] = 34

jArray[0, 0] = 110 jArray[0, 1] = 120 jArray[0, 2] = 130 jArray[0, 3] = 140
jArray[1, 0] = 210 jArray[1, 1] = 220 jArray[1, 2] = 230 jArray[1, 3] = 240
jArray[2, 0] = 310 jArray[2, 1] = 320 jArray[2, 2] = 330 jArray[2, 3] = 340

Для продолжения нажмите любую клавишу . . .

```

Основным методом, используемым для чтения данных с консоли, является метод `ReadLine`. Он читает с консоли строку текста, завершаемую знаком конца строки. Эта строка и является результатом, возвращаемым методом `ReadLine`. Введенная строка с помощью метода `ToInt32` конвертируется в значение типа `int` и это значение присваивается элементу `iArray[i, j]`. Далее в двойном цикле заполняется массив `jArray`

```
jArray[i, j] = iArray[i, j] * 10;
```

и затем оба массива выводятся на экран.

Для ввода данных из файла используется конструктор класса `StreamReader`. В классе `StreamReader` определено несколько конструкторов. Наиболее часто используемый `StreamReader(string имя_файла)`, где *имя_файла* – это имя открываемого файла, включая полный путь к нему. Здесь необходимо отметить, что в имени файла присутствует служебный символ `\`, который обычно на экран не выводится. Если все же необходимо использовать служебные символы в выводимом тексте, следует перед таким символом поставить дополнительный символ `\\` (обратный слеш).

Пример 2.

```
// Двухмерный массив
// Ввод массива в файл и
// вывод массива из файла
using System;
using System.IO;
namespace Example6
{
    class Example6_2
    {
        static void Main()
        {
            int i, j;
            string strValue;
            int[,] iArray1 = new int[3, 4];
            int[,] iArray2 = new int[3, 4];
```



```

StreamReader sRead = new
StreamReader("C:\\C#\\dat.txt");
StreamWriter sWrite = new
StreamWriter("C:\\C#\\res.txt");
for (i = 0; i < 3; i++)
{
    for (j = 0; j < 4; j++)
    {
        strValue = sRead.ReadLine();
        iArray1[i, j] =
        Convert.ToInt32(strValue);
        iArray2[i, j] = iArray1[i, j] *
        100;
        strValue = string.Format("\n {0,
        4:D} {1, 4:D} {2, 6:D} {3, 6:D}",
        i, j, iArray1[i, j], iArray2[i,
        j]);
        Console.WriteLine(strValue);
        Console.WriteLine();
    }
}
for (i = 0; i < 3; i++)
{
    for (j = 0; j < 4; j++)
    {
        sWrite.WriteLine(iArray2[i, j]);
    }
}

```

```

        }
        sWrite.Close();
    }
}

```

Смысл этого примера достаточно прозрачен и отличается только тем, что вместо класса `Console` (Example6_1) используются классы `StreamReader` и `StreamWriter` (Example6_2).

Следующая программа содержит пример ввода и вывода массива из файла. Массив `iArray1` содержит три столбца, первый из которых заполнен случайными числами типа `int`, второй – случайными числами типа `double` и третий значениями функции `Sin(x)`. Далее формируется строковая переменная `strValue`, которая форматируется и используется для вывода массива `iArray1` на экран. Следующий оператор

```
iArray2[i, j] = iArray1[i, j] * 100;
```

инициализирует массив `iArray2` и производит запись символического потока в текстовый файл `res3.txt`.

Класс `Array` – это класс, обслуживающий массивы в пространстве имен `System`. Его свойства и методы можно использовать для массивов любого из встроенных типов.

Свойство `Length` доступно только для чтения, имеет тип `int` и содержит количество элементов массива.

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

0  0  11  1100
0  1  12  1200
0  2  13  1300
0  3  14  1400
1  0  21  2100
1  1  22  2200
1  2  23  2300
1  3  24  2400
2  0  31  3100
2  1  32  3200
2  2  33  3300
2  3  34  3400
Для продолжения нажмите любую клавишу . . .
```

Метод `Copy (Array source, Array dest, count)` копирует число элементов, задаваемых параметром *count*, из исходного массива *source* в целевой массив *dest*, начиная с первого элемента массива. Далее происходит вывод массива `iArray1` на экран.

Пример 3.

```
// Заполнить двумерный массив
// и организовать вывод на экран.
// Ввод массива в файл и
// вывод массива из файла
```

```

using System;
using System.IO;
namespace Example6
{
    class Example6_3
    {
        static void Main()
        {
            int i, j, num;
            double x = 0;
            string strValue;
            double[,] iArray1 = new double[10, 3];
            double[,] iArray2 = new double[10, 3];
            Random rnd = new Random();
            StreamWriter sWrite = new
            StreamWriter("C:\\C#\\res3.txt");
            Console.WriteLine("\n -----
            -----");
            for (i = 0; i < 10; i++)
            {
                num = rnd.Next(1, 51);
                iArray1[i, 0] =
                Convert.ToDouble(num);
                num = (rnd.Next(1, 101) - 50) / 10;
                iArray1[i, 1] =
                Convert.ToDouble(num);
                iArray1[i, 2] = Math.Sin(x);
            }
        }
    }
}

```

```

        x = x + 0.314159;
        strValue = string.Format("\n {0, 4:D}
        {1, 10:F0} {2, 10:F2} {3, 10:F2}", i,
        iArray1[i, 0], iArray1[i, 1],
        iArray1[i, 2]);
        Console.WriteLine(strValue);
    }
    Console.WriteLine("\n -----
    -----");
    for (i = 0; i < 10; i++)
    {
        for (j = 0; j < 3; j++)
        {
            iArray2[i, j] = iArray1[i, j] *
            100;
            sWrite.WriteLine(iArray2[i, j]);
        }
    }
    num = iArray2.Length;
    Array.Copy(iArray2, iArray1, num);
    for (i = 0; i < 10; i++)
    {
        strValue = string.Format("\n {0, 4:D}
        {1, 10:F0} {2, 10:F2} {3, 10:F2}", i,
        iArray2[i, 0], iArray2[i, 1],
        iArray2[i, 2]);
        Console.WriteLine(strValue);
    }

```

```

    }
    Console.WriteLine("\n -----
    -----");
    sWrite.Close();
}
}
}

```

The screenshot shows a Windows command prompt window with the title bar "C:\WINDOWS\system32\cmd.exe". The window displays two tables of data, each preceded by a dashed line separator.

Table 1:

0	32	-3,00	0,00
1	12	-3,00	0,31
2	29	0,00	0,59
3	19	-2,00	0,81
4	46	4,00	0,95
5	1	-3,00	1,00
6	39	1,00	0,95
7	22	4,00	0,81
8	39	2,00	0,59
9	20	-4,00	0,31

Table 2:

0	3200	-300,00	0,00
1	1200	-300,00	30,90
2	2900	0,00	58,78
3	1900	-200,00	80,90
4	4600	400,00	95,11
5	100	-300,00	100,00
6	3900	100,00	95,11
7	2200	400,00	80,90
8	3900	200,00	58,78
9	2000	-400,00	30,90

Below the second table, there is a dashed line and the text: "Для продолжения нажмите любую клавишу . . ."

Пример Example6_4 демонстрирует метод вычисления суммы и среднеарифметического всех элементов массива, который заполняется случайными числами. Эти вычисления аналогичны тем, которые проводились для одномерного массива. Разница лишь в том, что в случае одномерного массива использовался одинарный цикл, а для двумерного массива – двойной.

Пример 4.

```
// Заполнить двухмерный массив случайными числами.
// Вычислить сумму и среднеарифметическое всех
// элементов
using System;
using System.IO;
namespace Example6
{
    class Example6_4
    {
        static void Main()
        {
            int i, j;
            int num, sum = 0;
            int[,] iArray = new int[5, 6];
            int[] iA = new int[6];
            Random rnd = new Random();
            string strValue = "\n -----
            -----";
            Console.Write("\n ");
            // номера столбцов
```

```

for (j = 0; j < 6; j++)
    Console.Write("{0, 5}", j);
Console.WriteLine(strValue);
// заполнение массива; сумма и
// среднеарифметическое всех элементов
for (i = 0; i < 5; i++)
{
    for (j = 0; j < 6; j++)
    {
        num = rnd.Next(1, 101);
        iArray[i, j] =
            Convert.ToInt32(num);
        iA[j] = iArray[i, j];
        sum = sum + iArray[i, j];
    }
}
// вывод массива
for (i = 0; i < 5; i++)
{
    // номер строки
    Console.Write("\n " + i);
    for (j = 0; j < 6; j++)
    {
        // элементы строки
        Console.Write("{0, 5}",
            iArray[i, j]);
    }
}

```



```

    }
    Console.WriteLine(strValue);
    Console.WriteLine();
    Console.WriteLine(" Сумма элементов
массива:    " + sum);
    Console.WriteLine(" Среднеарифметическое:
" + sum / 30);
    Console.WriteLine();
}
}
}

```

```

C:\WINDOWS\system32\cmd.exe

    0    1    2    3    4    5
-----
0  36    9   26   98   41   82
1  95   35   24   59   44   85
2  55   17   38   35   74   77
3  61   34   97   76   91   45
4  94   53   45   34   91   74
-----

Сумма элементов массива:    1725
Среднеарифметическое:      57

Для продолжения нажмите любую клавишу . . .

```

Следующий пример демонстрирует перестановку двух произвольных строк в двумерном массиве **A**, который инициализируется случайными числами. Все происходит аналогично перестановке любых двух элементов одномерного массива. Разница, как и раньше, лишь в том, что в случае одномерного массива использовался одинарный цикл, а для двумерного массива — двойной.

Пример 5.

```
// Заполнить двухмерный массив A(M, N)
// случайными числами.
// Переставить в этом массиве строки 2 и M - 2
using System;
namespace Example6
{
    class Example6_5
    {
        static void Main()
        {
            int M = 6, N = 8;
            int i, j, temp;
            int[,] iArray1 = new int[M, N];
            int[,] iArray2 = new int[M, N];
            int[ ] iArray3 = new int[N];
            Random rnd = new Random();
            string strValue = "\n -----
            -----";
            Console.Write("\n ");
            Console.WriteLine("\n Исходный массив:
            ");
            Console.WriteLine(strValue);
            // заполнение массива
            for (i = 0; i < M; i++)
            {
                for (j = 0; j < N; j++)
```

```

        {
            temp = rnd.Next(1, 101);
            iArray1[i, j] =
            Convert.ToInt32(temp);
            iArray2[i, j] =
            Convert.ToInt32(temp);
            iArray3[j] = iArray1[i, j];
        }
        foreach (int jj in iArray3)
        {
            Console.Write("{0, 5}", jj);
        }
        Console.Write("\n");
    }
    Console.WriteLine(strValue);
    Console.WriteLine("\n Массив после
    перестановки: ");
    Console.WriteLine(strValue);
    for (j = 0; j < N; j++)
    {
        temp = iArray2[2, j];
        iArray2[2, j] = iArray2[M - 2, j];
        iArray2[M - 1, j] = temp;
    }
    for (i = 0; i < M; i++)
    {
        for (j = 0; j < N; j++)

```

```

        {
            iArray3[j] = iArray2[i, j];
        }
        foreach (int jj in iArray3)
        {
            Console.Write("{0, 5}", jj);
        }
        Console.Write("\n");
    }
    Console.WriteLine(strValue);
    Console.WriteLine();
    Console.WriteLine();
}
}
}

```

```

C:\WINDOWS\system32\cmd.exe

Исходный массив:
-----
48  77  81  21  66  71  57  34
87  39  92  26  95  34  44   4
60  52  64  16  37  30  14  17
66  44  40  79  77  94  12  98
60  25  74  10  15  24  92   3
62  58  13  40  83  50  60  93
-----

Массив после перестановки:
-----
48  77  81  21  66  71  57  34
87  39  92  26  95  34  44   4
60  25  74  10  15  24  92   3
66  44  40  79  77  94  12  98
60  25  74  10  15  24  92   3
60  52  64  16  37  30  14  17
-----

Для продолжения нажмите любую клавишу . . .

```

Пример 6.

```
// Заполнить двухмерный массив A(5, 6)
// случайными числами.
// Найти в этом массиве строку,
// сумма элементов которой
// является максимальной, а в ней
// минимальный по величине элемент
using System;
namespace Example6
{
    class Example6_6
    {
        static void Main()
        {
            int im = 0, jm = 0, i, j;
            int num, sum = 0;
            int max = -100, min = 100;
            int[,] iArray = new int[5, 6];
            int[ ] iA = new int[6];
            Random rnd = new Random();
            string strValue = "\n -----
            -----";
            Console.Write("\n ");
            // номера столбцов
            for (j = 0; j < 6; j++)
                Console.Write("{0, 5}", j);
            Console.WriteLine(strValue);
```

```

// заполнение массива; сумма строк
for (i = 0; i < 5; i++)
{
    sum = 0;
    for (j = 0; j < 6; j++)
    {
        num = rnd.Next(1, 101);
        iArray[i, j] =
            Convert.ToInt32(num);
        iA[j] = iArray[i, j];
        sum = sum + iArray[i, j];
    }
    if (sum > max)
    {
        max = sum;
        im = i;
    }
}
for (j = 0; j < 5; j++)
{
    if (iArray[im, j] < min)
    {
        min = iArray[im, j];
        jm = j;
    }
}
// вывод массива

```

```

        for (i = 0; i < 5; i++)
        {
            // номер строки
            Console.Write("\n " + i);
            for (j = 0; j < 6; j++)
            {
                // элементы строки
                Console.Write("{0, 5}",
                    iArray[i, j]);
            }
            Console.WriteLine(strValue);
            Console.WriteLine();
            Console.WriteLine("\n i = " + im);
            Console.WriteLine("\n j = " + jm);
            Console.WriteLine();
        }
    }
}

```

Сумма элементов строки вычисляется во внутреннем цикле по j . На каждом шаге цикла к сумме элементов строки i добавляется очередной элемент. По окончании цикла переменная `sum` будет содержать полную сумму всех элементов данной строки. Максимальная сумма определяется во внешнем цикле. Для этого используется следующая группа операторов

```

if (sum > max)

```

```
{
    max = sum;
    im = i;
}
```

которая определяет максимальную сумму и номер строки *im*. Другая группа операторов

```
if (iArray[im, j] < min)
{
    min = iArray[im, j];
    jm = j;
}
```

используется для поиска минимального элемента и его номера *jm* в найденной строке.

The screenshot shows a Windows command prompt window with the title bar "C:\WINDOWS\system32\cmd.exe". Inside the window, a 6x6 array of numbers is displayed, with columns indexed 0 to 5 and rows indexed 0 to 4. Below the array, the values of variables *i* and *j* are shown as *i* = 2 and *j* = 3. At the bottom, there is a prompt "Для продолжения нажмите любую клавишу . . .".

	0	1	2	3	4	5
0	69	43	1	51	26	5
1	24	26	49	74	9	3
2	39	92	47	32	74	29
3	59	13	79	43	23	18
4	7	50	71	10	73	25

i = 2
j = 3
 Для продолжения нажмите любую клавишу . . .

Генерация таблицы умножения достаточно тривиальная задача. Каждый элемент таблицы определяется произведением номера

строки на номер столбца $k = i * j$. Единственная трудность заключается в организации вывода таблицы на экран.

Конструкция выбора

```
if (k < 10) Console.Write("  ");  
else      Console.Write(" ");
```

вставляет перед числом $k < 10$ два пробела, а во всех остальных случаях – один пробел. На нижеследующем рисунке представлен результат вывода, полученный при выполнении данной программы.

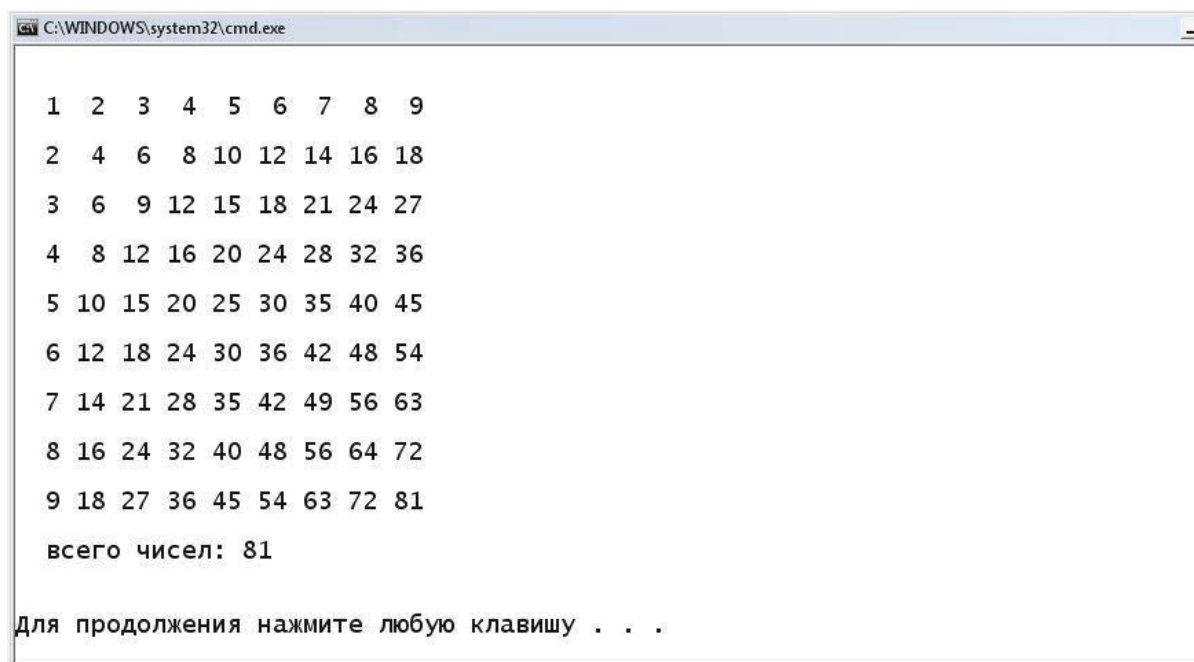
Пример 7.

```
// Рассчитать и вывести на экран  
// таблицу умножения. Подсчитать общее  
// количество чисел в данной таблице  
using System;  
namespace Example6  
{  
    class Example6_7  
    {  
        static void Main()  
        {  
            int i, j, k, m = 0;  
            for (i = 1; i < 10; i++)  
            {  
                Console.WriteLine("\n");  
                for (j = 1; j < 10; j++)  
                {  
                    k = i * j;
```

```

        m++;
        if (k < 10) Console.Write(" ");
        else          Console.Write(" ");
        Console.Write(k);
    }
}
Console.WriteLine("\n\n  всего чисел: "
+ m);
Console.WriteLine("\n");
}
}
}

```



```

C:\WINDOWS\system32\cmd.exe

1 2 3 4 5 6 7 8 9
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
всего чисел: 81
Для продолжения нажмите любую клавишу . . .

```

Программа данного примера выводит на экран верхнюю и нижнюю треугольные матрицы. Для управления выводом используются операторы `break` и `continue`. Оператор `break` прерывает

выполнение цикла при выполнении заданного условия. Оператор `continue` также при выполнении заданного условия приводит к пропуску следующих за ним операторов блока, но без выхода из блока.

Пример 8.

```
// Рассчитать таблицу (матрицу) умножения.
// Вывести на экран верхнюю треугольную и нижнюю
// треугольную матрицы. Подсчитать общее
// количество чисел (элементов) в каждой из матриц.
using System;
namespace Example6
{
    class Example6_8
    {
        static void Main()
        {
            // верхняя треугольная матрица
            int i, j, k, m = 0;
            for (i = 1; i < 10; i++)
            {
                Console.WriteLine("\n");
                for (j = 1; j < 10; j++)
                {
                    k = i * j;
                    if (j > 10 - i) break;
                    m++;
                    if (k < 10) Console.Write(" ");
                }
            }
        }
    }
}
```

```

        else Console.Write(" ");
        Console.Write(k);
    }
}
Console.WriteLine("\n\n  всего чисел: "
+ m);
Console.WriteLine("\n");
m = 0;
for (i = 1; i < 10; i++)
{
    Console.WriteLine("\n");
    for (j = 1; j < 10; j++)
    {
        k = i * j;
        if (k < i * (10 - i))
            Console.Write("  ");
        else
        {
            if (k < 10) Console.Write
                ("  " + k);
            else Console.Write(" " + k);
            m++;
        }
    }
}
Console.WriteLine("\n\n  всего чисел: "
+ m);

```

```

        Console.WriteLine("\n");
    }
}

```

```

C:\WINDOWS\system32\cmd.exe

1 2 3 4 5 6 7 8 9
2 4 6 8 10 12 14 16
3 6 9 12 15 18 21
4 8 12 16 20 24
5 10 15 20 25
6 12 18 24
7 14 21
8 16
9
всего чисел: 45

                                     9
                                16 18
                           21 24 27
                      24 28 32 36
                25 30 35 40 45
            24 30 36 42 48 54
        21 28 35 42 49 56 63
    16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
всего чисел: 45

Для продолжения нажмите любую клавишу . . .

```

В приведенных выше примерах применения двумерного массива, по существу создавался так называемый прямоугольный массив. Двумерный массив можно представить в виде таблицы, в кото-

рой длина каждой строки остается неизменной по всему массиву. Но в С# можно также создавать специальный тип двумерного массива, называемый ступенчатым массивом. Ступенчатый массив представляет собой массив массивов, в котором длина каждого массива может быть разной. Следовательно, ступенчатый массив можно использовать для составления таблицы из строк разной длины.

Ступенчатые массивы объявляются с помощью ряда квадратных скобок, в которых указывается их размерность. Например, для объявления двумерного ступенчатого массива служит следующая общая форма:

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

где *размер* обозначает число строк в массиве. Память для самих строк распределяется индивидуально, и поэтому длина строк может быть разной. Память сначала распределяется для его первого измерения автоматически, а затем для второго измерения вручную.

После создания ступенчатого массива доступ к его элементам осуществляется по индексу, указываемому в отдельных квадратных скобках. Например, в следующей строке кода элементу массива *jagged*, находящемуся на позиции с координатами (2,1), присваивается значение 10:

```
jagged[2][1] = 10;
```

Обратите внимание на синтаксические отличия в доступе к элементу ступенчатого и прямоугольного массива.

В приведенном ниже примере программы демонстрируется создание двумерного ступенчатого массива. Здесь следует обратить внимание, что язык C# позволяет использовать в качестве имен переменных и массивов русские имена.

Пример 9.

```
// Инициализировать и вывести на экран
// ступенчатый массив.
using System;
namespace Example6
{
    class Example6_9
    {
        static void Main()
        {
            int j, max = 0;
            int[] Массив = new int[5];
            for (j = 0; j < 5; j++)
            {
                Массив[j] = j + 2;
                if (Массив[j] > max) max = Массив[j];
            }
            // вспомогательный массив, определяющий
            // границы каждой из строк ступенчатого
            // массива
            int размер0 = 5;
            int размер1 = max;
            int размер2 = Convert.ToInt32((4 * max
```

```

- 1) / 3);
int[] размер = { размер0, размер1,
размер2 };
int[ ][ ] стМассив = new int[3][ ];
стМассив[0] = new int[размер[0]];
стМассив[1] = new int[размер[1]];
стМассив[2] = new int[размер[2]];
Console.WriteLine();
Console.Write("                ");
for (j = 0; j < размер2; j++)
Console.Write("{0,5}", j);
Console.WriteLine("\n\t\t -----
----- ");
Console.Write(" стМассив[0][5]");
for (j = 0; j < размер0; j++)
стМассив[0][j] = j + 1;
foreach (int jj in стМассив[0])
Console.Write("{0,5}", jj);
Console.WriteLine();
Console.Write(" стМассив[1][6]");
for (j = 0; j < размер1; j++)
стМассив[1][j] = j + 11;
foreach (int jj in стМассив[1])
Console.Write("{0,5}", jj);
Console.WriteLine();
Console.Write(" стМассив[2][7]");
for (j = 0; j < размер2; j++)

```



```

        стМассив[2][j] = j + 111;
        foreach (int jj in стМассив[2])
            Console.WriteLine("{0,5}", jj);
        Console.WriteLine("\n\t\t -----
        ----- ");
        Console.WriteLine();
    }
}
}

```

```

C:\WINDOWS\system32\cmd.exe

        0    1    2    3    4    5    6
-----
стМассив[0][5]  1    2    3    4    5
стМассив[1][6] 11   12   13   14   15   16
стМассив[2][7] 111  112  113  114  115  116  117
-----

для продолжения нажмите любую клавишу . . .

```

Ступенчатые массивы находят полезное применение не во всех, а лишь в некоторых случаях. Так, если требуется очень длинный двумерный массив, который заполняется не полностью, т.е. такой массив, в котором используются не все, а лишь отдельные его элементы, то для этой цели идеально подходит ступенчатый массив.

Следует отметить, что ступенчатые массивы представляют собой массивы массивов, и поэтому они не обязательно должны состоять из одномерных массивов. Так в приведенной ниже строке кода создается массив двумерных массивов.

```
int[ ][,] jArray = new int[5][,];
```

2. Практическая часть

Задания к лабораторной работе

Для заданных 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. Написать процедуру вычисления суммы всех элементов для заданного двумерного массива A состоящего из 5-ти строк и 6-ти столбцов.
2. Написать процедуру вычисления для заданного массива $A(5, 6)$ среднего арифметического элементов значения его положительных элементов. Известно, что хотя бы один элемент массива имеет положительное значение
3. Для заданного массива $A(4, 6)$ вычислить среднее арифметическое значение положительных элементов каждой строки. Результаты поместить в одномерный массив $B(4)$. Известно, что в каждой строке массива хотя бы один элемент имеет положительное значение.
4. Вычислить для заданного массива $B(20, 30)$ наибольший элемент каждого столбца. Результаты поместить в одномерный массив $BM(30)$.
5. Написать процедуру поиска для заданного массива $A(4, 5)$, строки с наибольшим средним арифметическим значением положительных элементов. Результат поместить в одномерный массив $D(5)$. Считаем, что хотя бы один положительный элемент в каждой строке имеется.

6. Написать процедуру. Для двумерного массива $A(6, 4)$, переставить местами 2-ю и 4-ю строки.
7. Написать процедуру. Для заданного массива $B(4, 5)$, переставить местами столбец с наибольшим количеством нулевых элементов и столбец последний по порядку следования в массиве B .
8. Написать процедуру для подсчета в заданном массиве $A(5, 5)$ количества элементов превышающих заданное число B и лежащих на главной диагонали и выше нее.
9. Написать процедуру для вывода на печать отрицательных элементов, лежащих на главной диагонали заданного массива $A(4, 4)$.
10. Написать процедуру перестановки элементов заданного массива $B(4, 4)$, лежащих на главной и побочной главной диагонали.
11. Написать процедуру пересылки двумерного массива $A(5, 6)$ в одномерный массив $B(30)$ того же размера, по строкам с сохранением порядка следования элементов.
12. Написать процедуру по транспонированию заданной квадратной матрицы A максимальная размерность которой 100.
13. Написать процедуру перемножения матрицы A на вектор B .
14. Написать процедуру перемножения матрицы $A(5, 10)$ на матрицу $B(10, 4)$.
15. Задан массив размером $N \times N$. Разделить элементы каждого столбца на элемент главной диагонали расположенный в соответствующем столбце.

16. Задан двумерный массив размером $N \times N$. Сформировать из его диагональных элементов одномерный массив.
17. Задан массив размером $N \times N$. Определить максимальный и минимальный элементы главной диагонали и переставить местами столбцы в которых лежат эти элементы.
18. Просуммировать элементы заданного двумерного массива размером $N \times N$, расположенные в его верхней части, ограниченной главной и побочной диагоналями, включая элементы, расположенные на этих диагоналях.
19. Для заданного двумерного массива размером $N \times N$ просуммировать элементы, расположенные на диагоналях, параллельных главной. Результаты поместить в одномерный массив.
20. В заданном двумерном массиве размером $N \times M$ поменять местами элементы первого и второго столбца, третьего и четвертого и т. д.
21. Задан массив размером 16. Сформировать из него массив размером 4×4 по строкам.
22. В заданном двумерном массиве размером $N \times M$ переместить нулевые элементы каждого столбца в конец столбца.
23. Задан двумерный массив размером $N \times N$. Максимальный элемент каждой строки поменять местами с диагональным элементом соответствующей строки.
24. Поместить в одномерный массив элементы, расположенные по периметру заданного двумерного массива размером $N \times N$. Использовать один цикл.
25. Заданный двумерный массив размером $N \times N$ повернуть вправо на 90° , без использования вспомогательных массивов.

26. В заданном двумерном массиве размером $N \times N$ поменять местами элементы, расположенные в верхней и нижней частях массива, между главной и побочной диагоналями за исключением элементов, расположенных на этих диагоналях.
27. Двумерный массив размером $N \times N$ задан в виде одномерного массива по столбцам. Вывести на печать верхний треугольник массива по строкам, включая элементы главной диагонали.
28. Написать процедуру, задающую треугольную матрицу произвольного размера. Число, определяющее размер матрицы вводится с клавиатуры во время выполнения процедуры
29. Сформировать двумерный массив, у которого элементы равны произведению своих индексов
30. Найти сумму элементов главной и произведение элементов побочной диагоналей квадратной матрицы