

МАССИВЫ

Массив – это набор элементов одного и того же типа, объединенных общим именем. Массивы в С# можно использовать по аналогии с тем, как они используются в других языках программирования, например, в С++, или Pascal. Однако С#-массивы имеют существенные отличия. Во-первых, они относятся к ссылочным типам данных. При этом имя массива является ссылкой на область кучи (динамической памяти), в которой последовательно размещается набор элементов определенного типа. Выделение памяти под элементы происходит на этапе объявления, или инициализации массива, а за освобождением памяти следит сборщик мусора. Во-вторых, массивы реализуются в С# как объекты, для которых разработан большой набор методов обработки элементов массива.

Рассмотрим следующие типы массивов: одномерные, многомерные и ступенчатые (рваные).

Одномерные массивы

Одномерный массив – это фиксированное количество элементов одного и того же типа, объединенных общим именем, где каждый элемент имеет свой номер. Нумерация элементов массива в С# начинается с нуля, то есть, если массив состоит из 10 элементов, то они будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Одномерный массив в С# реализуется как объект, поэтому его создание представляет собой двухступенчатый процесс. Сначала объявляется ссылочная переменная типа массив, затем выделяется память под требуемое количество элементов базового типа, и ссылочной переменной присваивается адрес нулевого элемента в массиве. Базовый тип определяет тип данных каждого элемента массива. Количество элементов, которые будут храниться в массиве, определяется размером массива.

При необходимости, этапы объявления переменной типа массив, и выделения необходимого объема памяти могут быть объединены в один. Кроме того, на этапе объявления массива можно произвести его инициализацию. Поэтому для объявления одномерного массива может использоваться одна из следующих форм записи:

- 1) базовый_тип [] имя_массива;

Например:

```
char [] a;
```

Объявлена ссылка на одномерный массив символов (имя ссылки a), которая в дальнейшем может быть использована для адресации на уже существующий массив, передачи массива в метод в качестве параметра, или отсроченного выделения памяти под элементы массива.

- 2) базовый_тип [] имя_массива = new базовый_тип [размер];

Например:

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

Объявлена ссылка b на одномерный массив целых чисел. Выделена память для 10 элементов целого типа, адрес этой области памяти записан в ссылочную переменную b. Элементы массива инициализируются по умолчанию нулями.

Замечание

Надо отметить, что в С# элементам массива присваиваются начальные значения в зависимости от базового типа. Для арифметических типов – нули, для ссылочных типов – null, для символов – символ с кодом ноль.

3) базовый_тип [] имя_массива={список инициализации};

Например:

```
double [] c={0.3, 1.2, -1.2, 31.5};
```

Объявлена ссылка с на одномерный массив вещественных чисел. Выделена память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации (четыре). Адрес этой области памяти записан в ссылочную переменную с. Значение элементов массива соответствует списку инициализации.

Обращение к элементу массива происходит с помощью индекса: указывается имя массива и, в квадратных скобках, номер элемента. Например,

```
a[0], b[8], c[i]
```

Так как массив представляет собой набор элементов, объединенных общим именем, то обработка массива обычно производится в цикле. Рассмотрим несколько основных примеров работы с одномерными массивами.

Вывод массива на экран

```
static void Main()
{
    int[] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    for (int i = 0; i < 10; i++)
    {
        Console.WriteLine(myArray[i]);
    }
}
```

Задание

Измените программу так, чтобы элементы массива выводились в строчку через пробел.

Для вывода одномерного массива на экран очень удобно использовать оператор `foreach`. Оператор `foreach` применяется для перебора элементов в специальном образом организованной группе данных, в том числе и в массиве. Удобство этого вида цикла заключается в том, что нам не требуется определять количество элементов в группе и выполнять перебор по индексу – мы просто указываем элементы какой группы необходимо перебрать. Синтаксис оператора:

```
foreach (<тип> <имя> in <группа>) <тело цикла>;
```

где *имя* определяет локальную по отношению к циклу переменную, которая будет по очереди перебирать все значения из указанной *группы*; ее *тип* соответствует базовому типу элементов *группы*.

Ограничением оператора `foreach` является то, что с его помощью можно только просматривать значения элементов в группе данных. Никаких изменений ни с самой группой, ни с находящимися в ней данными проводить нельзя.

В нашем случае, осуществить вывод массива на экран можно следующим образом:

```
static void Main()
{
    int[] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    foreach (int i in myArray)
    {
        Console.Write(i + " ");
    }
    Console.WriteLine();
}
```

```
        foreach (int elem in myArray)
        {
            Console.WriteLine(elem);
        }
    }
}
```

Задание

Измените программу так, чтобы на экран выводились квадраты элементов массива в строчку через пробел.

Ввод элементов массива

```
static void Main()
{
    int[] myArray;        //создаем ссылку на массив
    Console.Write("n= ");
    int n=int.Parse(Console.ReadLine());

    //выделяем память под массив требуемой длины
    myArray=new int [n];

    //вводим элементы массива с клавиатуры
    for (int i=0; i<n; i++)
    {
        Console.Write("A[{0}]= ",i);
        myArray[i]=int.Parse(Console.ReadLine());
    }

    foreach (int elem in myArray) //выводим массив на экран
    {
        Console.Write("{0} ",elem);
    }
}
```

Задание

Измените программу так, чтобы на экран выводились только положительные элементы массива.

Заполнение массива случайными элементами

Заполнить массив данными можно с помощью генератора случайных чисел. Для этого используется класс Random:

```
static void Main()
{
    //инициализируем генератор случайных чисел
    Random rnd = new Random();
    int[] myArray;

    //генерируем случайное число из диапазона [5..10)
    int n = rnd.Next(5, 10);
    myArray = new int[n];
}
```

```
for (int i = 0; i < n; i++)
{
    // заполняем массив случайными числами
    myArray[i] = rnd.Next(10);
}
foreach (int elem in myArray) //выводим массив на экран
{
    Console.Write("{0} ", elem);
}
}
```

Задание

Измените программу так, чтобы она работала с массивом вещественных чисел.

Контроль границ массива

Выход за границы массива в С# расценивается как критическая ошибка, которая ведет к завершению работы программы и генерированию стандартного исключения `IndexOutOfRangeException`. Рассмотрим следующий фрагмент программы:

```
int[] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
Console.WriteLine(myArray[10]);
```

В данном случае описан массив из 10 элементов. Так как нумерация элементов массива ведется с нуля, то последний элемент имеет номер 9, и, следовательно, элемента с номером 10 не существует. В этом случае выполнение программы завершится, о чем на консоль будет выдано соответствующее сообщение.

Замечание

Подробно обработка исключений в С# будет рассмотрена позже.

Массив как параметр

Так как имя массива фактически является ссылкой, то он передается в метод по ссылке и, следовательно, все изменения элементов массива, являющегося формальным параметром, отразятся на элементах соответствующего массива, являющегося фактическим параметром. При этом указывать спецификатор `ref` не нужно. Рассмотрим пример передачи массива в качестве параметра метода:

```
class Program
{
    //выводит на экран массив a
    static void Print(int[] a)
    {
        foreach (int elem in a)
        {
            Console.Write("{0} ", elem);
        }
        Console.WriteLine();
    }
    //Заменяет отрицательные элементы массива a на нули.
    //Число элементов в массиве - n.
    static void Change(int[] a, int n)
    {
```

```
        for (int i = 0; i < n; i++)
        {
            if (a[i] < 0)
            {
                a[i] = 0;
            }
        }
    }
}
static void Main()
{
    int[] myArray = { 0, -1, -2, 3, 4, 5, -6, -7, 8, -9 };
    Console.Write("Исходный массив: ");
    Print(myArray);
    Change(myArray, 10);
    Console.Write("Измененный массив: ");
    Print(myArray);
}
}
```

Задание

Измените программу так, чтобы метод `Change` удваивал значения положительных элементов массива.

То, что имя массива является ссылкой, следует учитывать при попытке присвоить один массив другому. Рассмотрим следующий пример:

```
class Program
{
    static void Print(int[] a)
    {
        foreach (int elem in a)
        {
            Console.Write("{0} ", elem);
        }
        Console.WriteLine();
    }
}
static void Main()
{
    int[] one = { 1, 2, 3, 4, 5 };
    Console.Write("Первый массив: ");
    Print(one);
    int[] two = { 6, 7, 8, 9 };
    Console.Write("Второй массив: ");
    Print(two);
    one=two;
    two[0]=-100;
    Console.WriteLine("После присвоения ");
    Console.Write("Первый массив: ");
    Print(one);
    Console.Write("Второй массив: ");
    Print(two);
}
}
```

Результат работы программы:

```
Первый массив: 1 2 3 4 5
Второй массив: 6 7 8 9
После присвоения
Первый массив: -100 7 8 9
Второй массив: -100 7 8 9
```

Таким образом, ссылки one и two ссылаются на один массив. Исходный массив, связанный со ссылкой one, оказался потерянным, и будет удален сборщиком мусора.

Массив как объект

Мы уже говорили о том, что массивы в C# реализованы как объекты. Если говорить более точно, то они реализованы на основе базового класса `Array`, определенного в пространстве имен `System`. Данный класс содержит различные свойства и методы. Например, свойство `Length` позволяет определять количество элементов в массиве. Используя данное свойство, внесем изменения в метод `Change`:

```
static void Change(int[] a)
{
    for (int i = 0; i < a.Length; i++)
    {
        if (a[i] > 0)
        {
            a[i] = 0;
        }
    }
}
```

Таким образом, информация о длине массива передается в метод `Change` неявным образом вместе с массивом, и мы избавились от необходимости вводить дополнительную переменную для хранения размерности массива.

Наиболее важные члены класса `Array` приведены в следующей таблице:

Элемент	Вид	Описание
<code>BinarySearch</code>	статический метод	Осуществляет двоичный поиск в отсортированном массиве
<code>Clear</code>	статический метод	Присваивает элементам массива значения, определенные по умолчанию, т.е для арифметических типов нули, для ссылочных типов <code>null</code> .
<code>Copy</code>	статический метод	Копирует элементы одного массива в другой массив.
<code>CopyTo</code>	экземплярный метод	Копирует все элементы текущего одномерного массива в другой массив
<code>IndexOf</code>	статический метод	Осуществляет поиск первого вхождения элемента в одномерный массив. Если элемент найден, то возвращает его индекс, иначе возвращает значение -1.
<code>LastIndexOf</code>	статический	Осуществляет поиск последнего вхождения элемента в одномерный массив. Если элемент найден, то возвращает

Элемент	Вид	Описание
	метод	его индекс, иначе возвращает значение -1.
Length	свойство	Возвращает количество элементов в массиве
Rank	свойство	Возвращает число размерностей массива. Для одномерного массива Rank возвращает 1, для двумерного – 2 и т.д.
Reverse	статический метод	Изменяет порядок следования элементов в массиве на обратный
Sort	статический метод	Упорядочивает элементы одномерного массива

Замечание

Обратите внимание на то, что для перечисленных членов класса *Array* не указываются параметры. Это связано с тем, что большинство из них имеют несколько перегруженных версий, поэтому при их использовании следует обращать внимание на подсказки VS и пользоваться справочной информацией.

Вызов статических методов происходит через обращение к имени класса, например, *Array.Sort(myArray)*. В данном случае мы обращаемся к статическому методу *Sort* класса *Array* и передаем данному методу в качестве параметра объект *myArray* – экземпляр класса *Array*.

Обращение к свойству или вызов экземплярного метода производится через обращение к экземпляру класса, например, *myArray.Length*, или *myArray.GetValue(i)*.

Пример

```
class Program
{
    static void Print(int[] a)
    {
        foreach (int elem in a)
        {
            Console.Write("{0} ", elem);
        }
        Console.WriteLine();
    }
    static void Main()
    {
        int[] one={2,4,6,1,-5,2,9,-2};
        Console.Write("Первый массив:");
        Print(one);
        Console.WriteLine("Первый раз значение 2 встречается в нем в
                           позиции {0}", Array.IndexOf(one,2));
        Console.WriteLine("Последний раз значение 2 встречается в нем
                           в позиции {0}", Array.LastIndexOf(one,2));
        Array.Sort(one);
        Console.Write("отсортирован по возрастанию:");
        Print(one);
        Array.Reverse(one);
        Console.Write("отсортирован по убыванию:");
    }
}
```

```
Print(one);

//создаем новый массив, копируя в него
//все элементы массива myArray
int [] two=new int[one.Length];
Array.Copy(one, two, one.Length);
Console.Write("\nВторой массив: ");
Print(two);

//копируем в середину массива newArray
//фрагмент массива myArray
Array.Copy(one,2, two,4,4);
Console.Write("Вставка элементов в него: ");
Print(two);
Console.WriteLine("\nИтоговые массивы ");
Console.Write("Первый: ");
Print(one);
Console.Write("Второй: ");
Print(two);
    }
}
```

Результат работы программы:

```
Первый массив: 2 4 6 1 -5 2 9 -2
Первый раз значение 2 встречается в нем в позиции 0
Последний раз значение 2 встречается в нем в позиции 5
отсортирован по возрастанию: -5 -2 1 2 2 4 6 9
отсортирован по убыванию: 9 6 4 2 2 1 -2 -5
Второй массив: 9 6 4 2 2 1 -2 -5
Вставка элементов в новый массив: 9 6 4 2 4 2 2 1
Итоговые массивы
Первый: 9 6 4 2 2 1 -2 -5
Второй: 9 6 4 2 4 2 2 1
```

Замечание

Обратите внимание на то, что myArray и newArray это ссылки на разные массивы, а не на один и тот же.

Задание

Добавьте в программу метод Input, предназначенный для ввода с клавиатуры элементов массива. Продемонстрируйте работу данного метода.

Самостоятельно изучите остальные члены класса Array и продемонстрируйте их работу.

Использование спецификатора params

Иногда бывает необходимо создать метод, в который можно передавать различное количество аргументов. Язык C# предоставляет такую возможность. Для этого параметр метода помечается спецификатором params, размещается в списке параметров последним и является массивом требуемого типа неопределенной длины. Следует отметить, что в методе может быть только один параметр помеченный спецификатором params.

Рассмотрим следующий пример:


```
class Program
{
    static int F(params int []a)
    {
        int s=0;
        foreach (int x in a)
        {
            s+=x;
        }
        return s;
    }
    static void Main()
    {
        int a = 1, b = 2, c = 3, d=4;
        Console.WriteLine(F());
        Console.WriteLine(F(a));
        Console.WriteLine(F(a, b));
        Console.WriteLine(F(a, b, c));
        Console.WriteLine(F(a, b, c, d));
    }
}
```

Результат работы программы:

```
Недостаточно аргументов
0
1
3
6
10
```

Как видим, в метод F может быть передано различное количество аргументов, в том числе и нулевое.

Рассмотрим следующий пример:

```
class Program
{
    static void F(int a, out int s, params int []b)
    {
        s = 0;
        if (b.Length != 0) //1
        {
            foreach (int x in b)
            {
                if (x == a)
                {
                    s += x;
                }
            }
        }
    }
    static void Main()
    {
        int a = 1;
```

```

        int []x = {0,1,2,1,0};
        int z;
        F(a, out z, x);
        Console.WriteLine(z);
        int []y = {};
        F(a,out z,y);
        Console.WriteLine(z);
    }
}

```

Результат работы программы:

```

Недостаточно аргументов
2
0

```

В данном случае в метод F должно быть передано три параметра: параметр значение a, выходной параметр s и непустой массив b. Если мы попытаемся передать меньшее количество параметров, то компилятор выдаст сообщение об ошибке.

Задание 1

В строке 1 записано условие `b.Length!=0`. Подумайте, можно ли обойтись без этого условия.

Задание 2

Измените метод F так, чтобы в параметр s записывалась количество элементов, равных a, стоящих на нечетных позициях. Подумайте, можно ли теперь обойтись без условия, записанного в строке 1.

Двумерные массивы

Многомерные массивы имеют более одного измерения. Чаще всего используются двумерные массивы, которые представляют собой таблицы.

Замечание

Работу с другими видами многомерных массивов рассмотрите самостоятельно.

Каждый элемент массива имеет два индекса. Первый индекс определяет номер строки, а второй – номер столбца, на пересечении которых находится элемент. Нумерация строк и столбцов начинается с нуля.

Объявить двумерный массив можно одним из предложенных способов:

- 1) базовый_тип [,] имя_массива;

Например:

```
int [,] a;
```

Объявлена ссылка на двумерный массив целых чисел (имя ссылки a), которая в дальнейшем может быть использована: для адресации на уже существующий массив; передачи массива в метод в качестве параметра; отсроченного выделения памяти под элементы массива.

- 2) базовый тип [,] имя_массива = new базовый_тип [размер1, размер2];

Например

```
float [,] a= new float [3, 4];
```

Объявлена ссылка **b** на двумерный массив вещественных чисел. Выделена память для 12 элементов вещественного типа, адрес данной области памяти записан в ссылочную переменную **b**. Элементы массива инициализируются по умолчанию нулями.

- 3) базовый_тип [,] имя_массива=={{элементы 1-ой строки}, ... , {элементы n-ой строки}};

Например:

```
int [,] a= new int [,]{{0, 1, 2}, {3, 4, 5}};
```

Объявлена ссылка **c** на двумерный массив целых чисел. Выделена память под двумерный массив, размерность которого 2×3. Адрес этой области памяти записан в ссылочную переменную **c**. Значение элементов массива соответствует списку инициализации.

Обращение к элементу массива происходит с помощью индексов: указывается имя массива **i**, в квадратных скобках, номер строки **i**, через запятую, номер столбца, на пересечении которых находится данный элемент. Например, **a[0, 0]**, **b[2, 3]**, **c[i, j]**.

Так как массив представляет собой набор элементов, объединенных общим именем, то обработка массива обычно производится с помощью вложенных циклов. Заметим также, что при обращении к свойству **Length** для двумерного массива мы получим общее количество элементов в массиве. Чтобы получить количество строк нужно обратиться к методу **GetLength** с параметром 0. Чтобы получить количество столбцов – к методу **GetLength** с параметром 1.

В качестве примера рассмотрим программу, в которой сразу будем учитывать два факта:

- 1) двумерные массивы относятся к ссылочным типам данных;
- 2) двумерные массивы реализованы как объекты.

```
class Program
{
    static void Print(int[,] a)
    {
        for (int i = 0; i < a.GetLength(0); i++)
        {
            for (int j = 0; j < a.GetLength(1); j++)
            {
                Console.Write("{0} ", a[i, j]);
            }
            Console.WriteLine();
        }
    }
    static void Input(out int[,] a)
    {
        Console.Write("n= ");
        int n = int.Parse(Console.ReadLine());
        Console.Write("m= ");
        int m = int.Parse(Console.ReadLine());
        a = new int[n,m];
        for (int i = 0; i < a.GetLength(0); i++)
        {
            for (int j = 0; j < a.GetLength(1); j++)
            {
                Console.Write("a[{0},{1}]= ", i, j);
                a[i,j] = int.Parse(Console.ReadLine());
            }
        }
    }
}
```

```
    }  
  }  
  static void Change(int[,] a)  
  {  
    for (int i = 0; i < a.GetLength(0); i++)  
      for (int j = 0; j < a.GetLength(1); j++)  
        if (a[i, j] % 2 == 0)  
        {  
          a[i, j] = 0;  
        }  
  }  
  static void Main()  
  {  
    int [,]a;  
    Input(out a);  
    Console.WriteLine("Исходный массив:");  
    Print(a);  
    Change(a);  
    Console.WriteLine("Измененный массив:");  
    Print(a);  
  }  
}
```

Результат работы программы:

```
n=2  
m=3  
a[0,0]=1  
a[0,1]=2  
a[0,2]=3  
a[1,0]=4  
a[1,1]=5  
a[1,2]=6  
Исходный массив:  
1 2 3  
4 5 6  
Измененный массив:  
1 0 3  
0 5 0
```

Задания

- 1) Объясните, что произойдет, если в качестве *n* и *m* ввести значение 3 и 4 соответственно, и обратиться к элементу массива следующим образом `a[3, 4]`.
- 2) Подумайте, можно ли в методе `Print` вместо вложенных циклов `for` использовать один цикл `foreach`. Как в этом случае массив будет выводиться на экран?
- 3) Объясните, почему в методе `Input` массив объявлен как выходной параметр. Измените метод так, чтобы его заголовок выглядел следующим образом: `static int[,] Input()`
- 4) Измените метод `Change` так, чтобы положительные элементы массива заменялись на противоположные значения.
- 5) Пусть массивы объявлены следующим образом:

```
int [,] a= {{0, 1}, {2, 3}};  
int [,] b= {{3, 4}, {5, 6}};
```

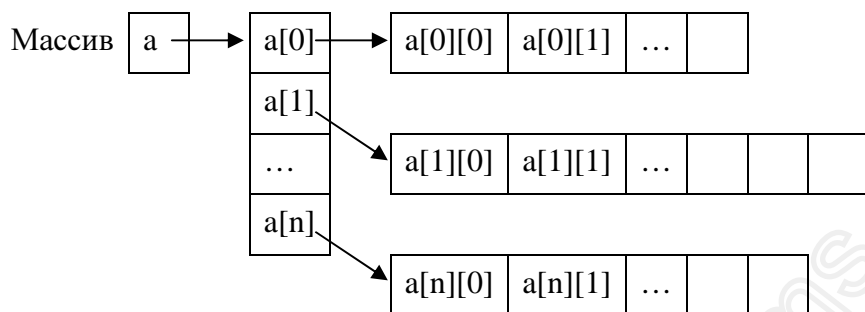
Какое значение будет выведено на экран в результате выполнения команды:

```
Console.WriteLine(a[0,1]);
```

если перед этим была выполнена команда $a=b$ и почему?

Ступенчатые массивы

В ступенчатых (или рваных) массивах количество элементов в разных строках может быть различным. В памяти ступенчатый массив хранится в виде массива массивов. Структура двумерного ступенчатого массива:



Объявление ступенчатого массива производится следующим образом:

```
тип [][] имя_массива;
```

Например:

```
int [][]a;
```

Фактически, мы объявили одномерный массив ссылок на целочисленные одномерные массивы. При таком описании потребуется не только выделять память под одномерный массив ссылок, но и под каждый из целочисленных одномерных массивов. Такое распределение памяти позволяет определять произвольную длину каждой строки массива (отсюда и произошло название массива – ступенчатый).

Например:

```
// Создаем три строки
int [][] a = new int [3][];
// 0-ая строка ссылается на 2-х элементный одномерный массив
a[0] = new int [2];
// 1-ая строка ссылается на 3-х элементный одномерный массив
a[1] = new int [3];
// 2-ая строка ссылается на 10 элементный одномерный массив
a[2] = new int [10];
```

Другой способ выделения памяти:

```
int [][] a = {new int [2], new int [3], new int [10]};
```

Так как каждая строка ступенчатого массива фактически является одномерным массивом, то с каждой строкой можно работать как с экземпляром класса `Array`. Это является преимуществом ступенчатых массивов перед двумерными массивами.

Пример:

```
class Program
{
    static void Print(int[][] a)
    {
        for (int i = 0; i < a.Length; i++)
        {
            for (int j = 0; j < a[i].Length; j++)
            {
                Console.Write("{0} ", a[i][j]);
            }
            Console.WriteLine();
        }
    }
    static void Input( out int[][]a)
    {
        Console.Write("n= ");
        int n = int.Parse(Console.ReadLine());
        a = new int[n][];
        for (int i = 0; i < a.Length; i++)
        {
            Console.Write("введите количество элементов в {0} строке:",
                           i);
            int j = int.Parse(Console.ReadLine());
            a[i] = new int[j];
            for (j = 0; j < a[i].Length; j++)
            {
                Console.Write("a[{0}][{1}]= ", i, j);
                a[i][j] = int.Parse(Console.ReadLine());
            }
        }
    }
    static void Change (int [][]a)
    {
        for (int i = 0; i < a.Length; i++)
        {
            Array.Sort(a[i]);
        }
    }
    static void Main()
    {
        int [][]a;
        Input(out a);
        Console.WriteLine("Исходный массив:");
        Print(a);
        Change(a);
        Console.WriteLine("Измененный массив:");
        Print(a);
    }
}
```

Результат работы программы:

```
n=3
введите количество элементов в 0 строке: 2
a[0,0]=1
a[0,1]=3
введите количество элементов в 1 строке: 4
a[1,0]=7
a[1,1]=3
a[1,2]=2
a[1,3]=5
введите количество элементов в 2 строке: 3
a[2,0]=0
a[2,1]=5
a[2,2]=2
Исходный массив:
1 3
7 3 2 5
0 5 2
Измененный массив:
1 3
2 3 5 7
0 2 5
```

Задания

- 1) Подумайте, можно ли в методе *Print* вместо вложенных циклов *for* использовать один цикл *foreach*. Как в этом случае массив будет выводиться на экран?
- 2) Объясните, почему в методе *Input* массив объявлен как выходной параметр. Измените метод так, чтобы его заголовок выглядел следующим образом: `static int[][] Input()`
- 3) Измените метод *Change* так, чтобы элементы каждой строки были отсортированы по убыванию.
- 4) Создайте метод `static int[][] Make(int n)`, который создает массив, в котором *n* строк, а количество элементов в каждой строке больше номера строки в два раза. При этом каждый элемент равен сумме номеров строки и столбца, в котором он находится. Продемонстрируйте работу данного метода.
- 5) Пусть массив был объявлен следующим образом:

```
int [][] a= new int [3][];
a[0]= new int[] {0, 1};
a[1]= new int[] {2, 3, 4};
a[2]= new int[] {5, 6};
```

- 6) Какое значение будет выведено на экран в результате выполнения команды:

```
Console.WriteLine(a[0][2]);
```

если перед этим была выполнена команда `a[0]=a[1]`, и почему?

- 7) Чему будет равно свойство *Rank* для враного массива? Ответ объясните.

Примеры использования массивов

При изучении массивов мы рассмотрели основные приемы работы с ними. Теперь приведем примеры использования массивов для решения практических задач.

Пример 1

Дан массив из n целых чисел. Написать программу для подсчета суммы этих чисел.

```
class Program
{
    static int[] Input()
    {
        Console.Write("n= ");
        int n=int.Parse(Console.ReadLine());
        int []a=new int[n];
        for (int i=0;i<a.Length;++i)
        {
            Console.Write("a[{0}]= ", i);
            a[i]=int.Parse(Console.ReadLine());
        }
        return a;
    }
    static int Sum(int[] a)
    {
        int sum=0;
        foreach (int elem in a )
        {
            sum+=elem;
        }
        return sum;
    }
    static void Main()
    {
        int[] a = Input();
        Console.WriteLine("Сумма элементов массива={0}", Sum(a));
    }
}
```

Результат работы программы:

n	Исходные данные	Ответ
5	1 2 3 4 5	15

Задание

Измените программу так, чтобы она работала для двумерного массива $n \times m$.

Пример 2

Дан массив из n целых чисел. Написать программу, которая определяет наименьший элемент в массиве и его порядковый номер.

```
class Program
{
    static int[] Input()
    {
        Console.Write("n= ");
        int n = int.Parse(Console.ReadLine());
        int[] a = new int[n];
```



```
        for (int i = 0; i < a.Length; i++)
        {
            Console.Write("a[{0}]= ", i);
            a[i] = int.Parse(Console.ReadLine());
        }
        return a;
    }
    static int Min(int[] a, out int index)
    {
        //в качестве наименьшего значения полагаем
        //нулевой элемент массива
        int min = a[0];
        index = 0; //и запоминаем его номер

        //перебираем все элементы массива
        for (int i = 0; i < a.Length; i++)
        {
            //если очередной элемент окажется меньше значения min,
            //то в качестве
            if (a[i] < min)
            {
                //нового наименьшего значения запоминаем
                //значение текущего элемента
                min = a[i];
                index = i; //и фиксируем его номер
            }
        }
        return min;
    }
    static void Main()
    {
        int[] a = Input();
        int index;
        int min = Min(a, out index);
        Console.WriteLine("Наименьший элемент {0} имеет номер {1}",
                           min, index);
    }
}
```

Результат работы программы:

n	Исходные данные	Наименьшее значение	Его номер
5	1 3 7 -41 9	-41	3

Задание 1

Измените программу так, чтобы она вычисляла наибольший элемент в одномерном массиве и его номер.

Задание 2

Измените программу так, чтобы она работала для двумерного массива $n \times m$.

Пример 3

Дан массив из n целых чисел. Написать программу, которая все наименьшие элементы увеличивает в два раза.

```
class Program
{
    static int[] Input()
    {
        Console.Write("n= ");
        int n = int.Parse(Console.ReadLine());
        int[] a = new int[n];
        for (int i = 0; i < a.Length; i++)
        {
            Console.Write("a[{0}]= ", i);
            a[i] = int.Parse(Console.ReadLine());
        }
        return a;
    }
    static void Print(int[] a)
    {
        foreach (int elem in a)
        {
            Console.Write("{0} ", elem);
        }
        Console.WriteLine();
    }
    static int Min(int[] a)
    {
        int min = a[0];
        for (int i = 0; i < a.Length; i++)
        {
            if (a[i] < min)
            {
                min = a[i];
            }
        }
        return min;
    }
    static void Change(int[] a, int x, int y)
    {
        for (int i = 0; i < a.Length; i++)
        {
            if (a[i] == x)
            {
                a[i] *= y;
            }
        }
    }
    static void Main()
    {
        int[] a = Input();
```

```
        Console.WriteLine("Исходный массив:");  
        Print(a);  
        int min = Min(a);  
        const int n = 2;  
        Change(a, min,n);  
        Console.WriteLine("Измененный массив:");  
        Print(a);  
    }  
}
```

Результат работы программы:

n	Исходные данные	Измененные данные
6	3 5 7 3 9 3	6 5 7 6 9 6

Задание

Измените программу так, чтобы она работала для двумерного массива $n \times m$.

Пример 4

Дан массив из n целых чисел. Написать программу, которая подсчитывает количество пар соседних элементов массива, для которых предыдущий элемент равен последующему.

```
class Program  
{  
    static int[] Input()  
    {  
        Console.Write("n= ");  
        int n = int.Parse(Console.ReadLine());  
        int[] a = new int[n];  
        for (int i = 0; i < a.Length; i++)  
        {  
            Console.Write("a[{0}]= ", i);  
            a[i] = int.Parse(Console.ReadLine());  
        }  
        return a;  
    }  
    static int F(int[] a)  
    {  
        int k=0;  
        for (int i = 0; i < a.Length-1; i++)  
        {  
            if (a[i] == a[i+1])  
            {  
                ++k;  
            }  
        }  
        return k;  
    }  
    static void Main()  
    {  
        int[] a = Input();  
        Console.WriteLine("k={0}", F(a));  
    }  
}
```

Результат работы программы:

n	Исходные данные	Ответ
6	1 2 3 4 5 6	k=0
6	1 1 2 2 3 3	k=3

Замечание

Обратите внимание на то, что в последнем цикле параметр i принимает значения от 0 до $n-2$, а не до $n-1$. Это связано с тем, что для $i=n-2$ существует пара с номерами $(n-2, n-1)$, а для $i=n-1$ пары с номерами $(n-1, n)$ не существует, так как в массиве всего n элементов и последний элемент имеет номер $n-1$.

Задание

Измените программу так, чтобы она подсчитывала количество пар соседних элементов массива, для которых предыдущий элемент больше последующего.

Пример 5

Дана квадратная матрица, элементами которой являются вещественные числа. Подсчитать сумму элементов главной диагонали.

Указания по решению задачи. Для элементов, стоящих на главной диагонали, характерно то, что номер строки совпадает с номером столбца. Этот факт будем учитывать при решении задачи.

```
class Program
{
    static void Print(int[,] a)
    {
        for (int i = 0; i < a.GetLength(0); i++)
        {
            for (int j = 0; j < a.GetLength(1); j++)
            {
                Console.Write("{0} ", a[i, j]);
            }
            Console.WriteLine();
        }
    }
    static void Input(out int[,] a)
    {
        Console.Write("n= ");
        int n = int.Parse(Console.ReadLine());
        a = new int[n, n];
        for (int i = 0; i < a.GetLength(0); i++)
        {
            for (int j = 0; j < a.GetLength(1); j++)
            {
                Console.Write("a[{0},{1}]= ", i, j);
                a[i, j] = int.Parse(Console.ReadLine());
            }
        }
    }
    static int F (int[,] a)
    {
        int k = 0;
```

```

        for (int i = 0; i < a.GetLength(0); i++)
        {
            k += a[i, i];
        }
        return k;
    }
    static void Main()
    {
        int[,] a;
        Input(out a);
        Console.WriteLine("Исходный массив:");
        Print(a);
        Console.WriteLine("Сумма элементов на главной диагонали {0}",
                           F(a));
    }
}

```

Результат работы программы:

n	Массив	Ответ
3	1 2 3 4 5 6 7 8 9	Сумма элементов главной диагонали = 15

Задание

Измените программу так, чтобы она подсчитывала сумму элементов, стоящих на побочной диагонали.

Пример 6

Дана прямоугольная матрица $n \times m$, элементами которой являются целые числа. Поменять местами ее строки следующим образом: первую строку с последней, вторую с предпоследней и т.д.

Указания по решению задачи. Если в массиве n строк, то 0-ую строку нужно поменять с $n-1$, 1-ую строку – с $n-2$, i -ую строку – с $n-i-1$. При этом перебираются не все строки, а только половина. Следует отметить, что хотя по условию дана прямоугольная матрица, т.е., двумерный массив, но для решения данной задачи удобнее использовать ступенчатый массив. Подумайте, почему.

```

class Program
{
    static void Print(int[][] a)
    {
        for (int i = 0; i < a.Length; i++)
        {
            for (int j = 0; j < a[i].Length; j++)
            {
                Console.Write("{0} ", a[i][j]);
            }
            Console.WriteLine();
        }
    }
}

```

```
static void Input(out int[][] a)
{
    Console.Write("n= ");
    int n = int.Parse(Console.ReadLine());
    Console.Write("m= ");
    int m = int.Parse(Console.ReadLine());
    a = new int[n][];
    for (int i = 0; i < a.Length; i++)
    {
        a[i] = new int[m];
        for (int j = 0; j < a[i].Length; j++)
        {
            Console.Write("a[{0}][{1}]= ", i, j);
            a[i][j] = int.Parse(Console.ReadLine());
        }
    }
}

static void Change(int[][] a)
{
    int[] z;
    int n = a.Length;
    //меняются местами i-ая и (n-i-1)-ая строки
    for (int i = 0; i < (n / 2); i++)
    {
        z = a[i];
        a[i] = a[n - i - 1];
        a[n - i - 1] = z;
    }
}

static void Main()
{
    int[][] a;
    Input(out a);
    Console.WriteLine("Исходный массив:");
    Print(a);
    Change(a);
    Console.WriteLine("Измененный массив:");
    Print(a);
}
```

Результат работы программы:

n	m	Массив	Ответ
4	3	1 2 3	0 1 2
		4 5 6	7 8 9
		7 8 9	4 5 6
		7 8 9	1 2 3

Задание

Измените программу так, чтобы она работала с двумерным, а не со ступенчатым массивом.

Пример 7

Дана прямоугольная матрица, элементами которой являются целые числа. Для каждого столбца подсчитать среднее арифметическое его нечетных элементов и записать полученные данные в новый массив.

```
class Program
{
    static void Print(double[] a)
    {
        foreach (double elem in a)
        {
            Console.Write("{0} ", elem);
        }
        Console.WriteLine();
    }
    static void Print(int[,] a)
    {
        for (int i = 0; i < a.GetLength(0); i++)
        {
            for (int j = 0; j < a.GetLength(1); j++)
            {
                Console.Write("{0,5:f2} ", a[i, j]);
            }
            Console.WriteLine();
        }
    }
    static void Input(out int[,] a)
    {
        Console.Write("n= ");
        int n = int.Parse(Console.ReadLine());
        a = new int[n, n];
        for (int i = 0; i < a.GetLength(0); i++)
        {
            for (int j = 0; j < a.GetLength(1); j++)
            {
                Console.Write("a[{0},{1}]= ", i, j);
                a[i, j] = int.Parse(Console.ReadLine());
            }
        }
    }
    static double[] F(int[,] a)
    {
        double[] b = new double[a.GetLength(1)];
        for (int j = 0; j < a.GetLength(1); j++)
        {
            int k = 0;
            for (int i = 0; i < a.GetLength(0); i++)
            {
                if (a[i, j] % 2 == 1)
                {
                    b[j] += a[i, j];
                    k++;
                }
            }
        }
    }
}
```

```

    }
    if (k != 0)
    {
        b[j] /= k;
    }
}
return b;
}
static void Main()
{
    int[,] a;
    Input(out a);
    Console.WriteLine("Исходный двумерный массив:");
    Print(a);
    double[] b=F(a);
    Console.WriteLine("Искомый одномерный массив:");
    Print(b);
}
}

```

Результат работы программы:

n	m	Массив	Ответ
3	3	1 2 3	4.00 5.00 6.00
		4 5 6	
		7 8 9	

Задание

Измените программу так, чтобы она подсчитывала среднее арифметическое нечетных элементов массива для каждой его строки и записывала полученные данные в новый массив.

Вставка и удаление элементов в массивах

При объявлении массива мы определяем его максимальную размерность, которая в дальнейшем изменена быть не может. Однако с помощью вспомогательной переменной можно контролировать текущее количество элементов, которое не может быть больше максимального.

Замечание

В пространстве имен *System.Collection* реализована коллекция *ArrayList* – массив, динамически изменяющий свой размер. Мы будем рассматривать его позже.

Пример

Рассмотрим фрагмент программы:

```

int []a = new int [10];
int n = 5;
for (int i = 0; i < n; i++)
{
    a[i] = i*i;
}

```

В этом случае массив можно представить следующим образом:

n=5	0	1	2	3	4	5	6	7	8	9
a	0	1	4	9	16	0	0	0	0	0

Во время описания был определен массив из 10 элементов, заполненных нулями, а затем заполнены только первые 5. В оставшихся элементах сохраняются значения по-умолчанию.

Что значит *удалить из одномерного массива* элемент с номером 3? Удаление должно привести к физическому «уничтожению» элемента с номером 3 из массива, при этом общее количество элементов должно быть уменьшено. В этом понимании удаления элемента итоговый массив должен выглядеть следующим образом

	0	1	2	4	5	6	7	8	9	недопустимое состояние
a	0	1	4	16	0	0	0	0	0	

Такое удаление для массивов *невозможно*, поскольку элементы массива располагаются в памяти последовательно друг за другом.

Однако «удаление» можно смоделировать сдвигом элементов влево и уменьшением значения переменной, которая отвечает за текущее количество элементов в массиве, на единицу:

n=4	0	1	2	3	4	5	6	7	8	9
a	0	1	4	16	0	0	0	0	0	0

В общем случае, если мы хотим удалить элемент массива с номером k (всего в массиве n элементов, а последний элемент имеет индекс $n-1$), то нам необходимо произвести сдвиг элементов, начиная с $(k+1)$ -го на одну позицию влево. Т.е., на k -ое место поставить $(k+1)$ -й элемент, на место $(k+1)$ -го – $(k+2)$ -й элемент, ..., на место $(n-2)$ – $(n-1)$ -й элемент. После этого значение n необходимо уменьшить на 1. В этом случае число элементов в массиве не изменится, изменится лишь текущее количество элементов, и у нас создается ощущение, что элемент с номером k удален. Рассмотрим данный алгоритм на примере:

```
class Program
{
    static int[] Input()
    {
        Console.WriteLine("n= ");
        int n = int.Parse(Console.ReadLine());
        int[] a = new int[n];
        for (int i = 0; i < a.Length; i++)
        {
            Console.WriteLine("a[{0}]= ", i);
            a[i] = int.Parse(Console.ReadLine());
        }
        return a;
    }
    static void Print(int[] a, int n)
    {
        for (int i = 0; i < n; i++)
        {
            Console.WriteLine("{0} ", a[i]);
        }
        Console.WriteLine();
    }
}
```

```
static void Delete(int[] a, ref int n, int m)
{
    for (int i = m; i < n - 1; i++)
    {
        a[i] = a[i + 1];
    }
    --n;
}
static void Main()
{
    int[] a = Input();
    int n = a.Length;
    Console.WriteLine("Исходный массив:");
    Print(a, n);
    Console.WriteLine("Введите номер элемента для удаления:");
    int m = int.Parse(Console.ReadLine());
    Delete(a, ref n, m);
    Console.WriteLine("Измененный массив:");
    Print(a, n);
}
```

Задание

Измените метод Delete так, чтобы он удалял из массива элемент со значением m, при условии, что все элементы в массиве встречаются ровно один раз.

Рассмотрим теперь операцию *удаления в двумерном массиве*. Число элементов в двумерном массиве также зафиксировано на этапе объявления массива. Однако при необходимости можно «смоделировать» удаление целой строки в массиве, выполняя сдвиг всех строк, начиная с k-той на единицу вверх. В этом случае реальное число строк также не изменится, а текущее количество строк будет уменьшено на единицу. В качестве примера удалим из двумерного массива строку с номером k.

```
class Class
{
    static int[,] Input (out int n, out int m)
    {
        Console.WriteLine("введите размерность массива");
        Console.Write("n = ");
        n = int.Parse(Console.ReadLine());
        Console.Write("m = ");
        m = int.Parse(Console.ReadLine());
        int[,] a = new int[n, m];
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
            {
                Console.Write("a[{0},{1}]= ", i, j);
                a[i, j] = int.Parse(Console.ReadLine());
            }
        return a;
    }
}
```

```
static void Print(int[,] a, int n, int m)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
            Console.Write("{0,5} ", a[i, j]);
        Console.WriteLine();
    }
}
static void Delete(int[,] a, ref int n, int m, int k)
{
    for (int i = k; i < n-1; i++)
        for (int j = 0; j < m; j++)
            a[i, j] = a[i+1, j];
    --n;
}
static void Main()
{
    int n, m;
    int[,] a = Input(out n, out m);
    Console.WriteLine("Исходный массив:");
    Print(a, n, m);
    Console.WriteLine("Введите номер строки для удаления:");
    int k = int.Parse(Console.ReadLine());
    Delete(a, ref n, m, k);
    Console.WriteLine("Измененный массив:");
    Print(a, n, m);
}
```

Задание

Измените программу так, чтобы она удаляла k-тый столбец в двумерном массиве.

Рассмотрим модификацию предыдущей программы для случая, когда используется ступенчатый массив.

```
class Class
{
    static int[][] Input(out int n, out int m)
    {
        Console.WriteLine("Введите размерность массива");
        Console.Write("n = ");
        n = int.Parse(Console.ReadLine());
        Console.Write("m = ");
        m = int.Parse(Console.ReadLine());
        int[][] a = new int[n][];
        for (int i = 0; i < n; i++)
        {
            a[i] = new int[m];
            for (int j = 0; j < m; j++)
            {
                Console.Write("a[{0},{1}]= ", i, j);
                a[i][j] = int.Parse(Console.ReadLine());
            }
        }
    }
}
```

```

    }
    return a;
}
static void Print(int[][] a, int n, int m)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            Console.Write("{0,5} ", a[i][j]);
        }
        Console.WriteLine();
    }
}
static void Delete(int[][] a, ref int n, int k)
{
    for (int i = k; i < n - 1; i++) //производим сдвиг ссылок
    {
        a[i] = a[i + 1];
    }
    --n;
}
static void Main()
{
    int n, m;
    int[][] a = Input(out n, out m);
    Console.WriteLine("Исходный массив:");
    Print(a, n, m);
    Console.WriteLine("Введите номер строки для удаления:");
    int k = int.Parse(Console.ReadLine());
    Delete(a, ref n, k);
    Console.WriteLine("Измененный массив:");
    Print(a, n, m);
}
}

```

Задание

Измените программу так, чтобы она удаляла *k*-ый столбец в двумерном массиве.

Вернемся к массиву, определенному в самом первом примере. Что значит *добавить элемент в одномерный массив* в позицию с номером *k*? В этом случае все элементы, начиная с *k*-ого, должны быть сдвинуты вправо на одну позицию. Однако сдвиг нужно начинать с конца, т.е., на первом шаге на *n*-е место поставить (*n*–1)-ый элемент, потом на (*n*–1)-ое место поставить (*n*–2)-й элемент, ..., и, наконец, на (*k*+1) место вставить *k*-й элемент. Таким образом, копия *k*-го элемента будет на (*k*+1)-м месте, и на *k*-е место можно поставить новый элемент. Затем необходимо увеличить текущее количество элементов на 1.

Рассмотрим массив из примера 1, и в качестве *k* зададим значение равное 3. В этом случае массив будет выглядеть следующим образом:

k=3	0	1	2	3	4	5	6	7	8	9
a	0	1	4	9	9	16	0	0	0	0

Теперь в позицию с номером 3 можно поместить новое значение, а текущее количество элементов в массиве станет равным 6.

Задание

Подумайте, почему сдвиг нужно выполнять с конца массива, а не с начала, как мы это делали в случае удаления элемента из массива.

Рассмотрим программную реализацию данного алгоритма:

```
class Program
{
    static int[] Input(out int n)
    {
        Console.WriteLine("Введите размерность массива");
        n = int.Parse(Console.ReadLine());
        //выделяем памяти больше чем требуется
        int[] a = new int[2 * n];
        for (int i = 0; i < n; i++)
        {
            Console.Write("a[{0}]= ", i);
            a[i] = int.Parse(Console.ReadLine());
        }
        return a;
    }
    static void Print(int[] a, int n)
    {
        for (int i = 0; i < n; i++)
        {
            Console.Write("{0} ", a[i]);
        }
        Console.WriteLine();
    }
    static void Add(int[] a, ref int n, int m)
    {
        for (int i = n; i >= m; i--)
        {
            a[i] = a[i - 1];
        }
        ++n;
        Console.WriteLine("Введите значение нового элемента");
        a[m] = int.Parse(Console.ReadLine());
    }
    static void Main()
    {
        int n;
        int[] a = Input(out n);
        Console.WriteLine("Исходный массив:");
        Print(a, n);
        Console.WriteLine("Введите номер элемента для вставки:");
        int m = int.Parse(Console.ReadLine());
        Add(a, ref n, m);
        Console.WriteLine("Измененный массив:");
    }
}
```

```
        Print(a, n);  
    }  
}
```

Задание

Измените метод *Add* так, чтобы новый элемент добавлялся в массив после элемента со значением *m* (все элементы в массиве встречаются ровно один раз).

Теперь рассмотрим добавление строки в двумерный массив. Для этого все строки после строки с номером *k* передвигаем на 1 строку вниз. Затем, увеличиваем количество строк на 1. После этого копия строки с номером *k* будет находиться в столбце с номером (*k*+1). Следовательно, *k*-тый столбец можно заполнить новыми значениями. Рассмотрим программную реализацию алгоритма:

```
class Class  
{  
    static int[,] Input(out int n, out int m)  
    {  
        Console.WriteLine("введите размерность массива");  
        Console.Write("n = ");  
        n = int.Parse(Console.ReadLine());  
        Console.Write("m = ");  
        m = int.Parse(Console.ReadLine());  
        //выделяем памяти больше чем необходимо  
        int[,] a = new int[2 * n, m];  
        for (int i = 0; i < n; i++)  
        {  
            for (int j = 0; j < m; j++)  
            {  
                Console.Write("a[{0},{1}]= ", i, j);  
                a[i, j] = int.Parse(Console.ReadLine());  
            }  
        }  
        return a;  
    }  
    static void Print(int[,] a, int n, int m)  
    {  
        for (int i = 0; i < n; i++)  
        {  
            for (int j = 0; j < m; j++)  
            {  
                Console.Write("{0,5} ", a[i, j]);  
            }  
            Console.WriteLine();  
        }  
    }  
    static void Add(int[,] a, ref int n, int m, int k)  
    {  
        for (int i = n; i >= k; i--)  
        {  
            for (int j = 0; j < m; j++)  
            {  
                a[i + 1, j] = a[i, j];  
            }  
        }  
    }  
}
```

```
    }
    }
    ++n;
    Console.WriteLine("Введите элементы новой строки");
    for (int j = 0; j < m; j++)
    {
        Console.Write("a[{0},{1}]= ", k, j);
        a[k, j] = int.Parse(Console.ReadLine());
    }
}
static void Main()
{
    int n, m;
    int[,] a = Input(out n, out m);
    Console.WriteLine("Исходный массив:");
    Print(a, n, m);
    Console.WriteLine("Введите номер строки для добавления:");
    int k = int.Parse(Console.ReadLine());
    Add(a, ref n, m, k);
    Console.WriteLine("Измененный массив:");
    Print(a, n, m);
}
}
```

Задание

Измените программу так, чтобы она добавляла k-ый столбец в двумерный массив.

Рассмотрим модификацию предыдущей программы для случая, когда используется ступенчатый массив.

```
class Class
{
    static int[][] Input(out int n, out int m)
    {
        Console.WriteLine("введите размерность массива");
        Console.Write("n = ");
        n = int.Parse(Console.ReadLine());
        Console.Write("m = ");
        m = int.Parse(Console.ReadLine());
        //выделяем памяти больше чем необходимо
        int[][] a = new int[2 * n][];
        for (int i = 0; i < n; i++)
        {
            a[i] = new int[m];
            for (int j = 0; j < m; j++)
            {
                Console.Write("a[{0}][{1}]= ", i, j);
                a[i][j] = int.Parse(Console.ReadLine());
            }
        }
        return a;
    }
}
```

```
static void Print(int[][] a, int n, int m)
{
    for (int i = 0; i < n; ++i, Console.WriteLine())
    {
        for (int j = 0; j < m; ++j)
        {
            Console.Write("{0,5} ", a[i][j]);
        }
    }
}

static void Add(int[][] a, ref int n, int m, int k)
{
    for (int i = n; i >= k; i--) //выполняем сдвиг ссылок
    {
        a[i + 1] = a[i];
    }
    ++n;
    a[k] = new int[m]; //создаем новую строку
    Console.WriteLine("Введите элементы новой строки");
    for (int j = 0; j < m; j++)
    {
        Console.Write("a[{0}][{1}]=", k, j);
        a[k][j] = int.Parse(Console.ReadLine());
    }
}

static void Main()
{
    int n, m;
    int[][] a = Input(out n, out m);
    Console.WriteLine("Исходный массив:");
    Print(a, n, m);
    Console.WriteLine("Введите номер строки для добавления:");
    int k = int.Parse(Console.ReadLine());
    Add(a, ref n, m, k);
    Console.WriteLine("Измененный массив:");
    Print(a, n, m);
}
```

Задание

Измените программу так, чтобы в каждую строку ступенчатого массива добавлялся новый элемент в указанную позицию.

Практикум №6

Задание 1

Для заданной последовательности целых чисел

- 1) заменить все положительные элементы противоположными им числами;
- 2) заменить все элементы, меньшие заданного числа, этим числом;
- 3) заменить все элементы, попадающие в интервал [a, b], нулем;
- 4) заменить все отрицательные элементы, не кратные 3, противоположными им числами;
- 5) все элементы, меньшие заданного числа, увеличить в два раза;

- 6) подсчитать среднее арифметическое элементов;
- 7) подсчитать среднее арифметическое отрицательных элементов;
- 8) подсчитать количество нечетных элементов;
- 9) подсчитать сумму элементов, попадающих в заданный интервал;
- 10) подсчитать сумму элементов, кратных 9;
- 11) подсчитать количество элементов, не попадающих в заданный интервал;
- 12) подсчитать сумму квадратов четных элементов;
- 13) вывести на экран номера всех элементов, больших заданного числа;
- 14) вывести на экран номера всех нечетных элементов;
- 15) вывести на экран номера всех элементов, которые не делятся на 7;
- 16) вывести на экран номера всех элементов, не попадающих в заданный интервал;
- 17) определить, является ли произведение элементов трехзначным числом;
- 18) определить, является ли сумма элементов двухзначным числом;
- 19) вывести на экран элементы с четными индексами (для двумерного массива сумма индексов должна быть четной);
- 20) вывести на экран положительные элементы с нечетными индексами (для двумерного массива первый индекс должен быть нечетным).

Замечание

Задачи из данного пункта решить двумя способами, используя одномерный массив, а затем двумерный. Размерность массива вводится с клавиатуры.

Задание 2

Для заданной последовательности из n действительных чисел

- 1) подсчитать количество максимальных элементов;
- 2) вывести на экран номера всех минимальных элементов;
- 3) заменить все максимальные элементы нулями;
- 4) заменить все минимальные элементы на противоположные;
- 5) поменять местами максимальный элемент и первый;
- 6) вывести на экран номера всех элементов, не совпадающих с максимальным;
- 7) найти номер первого минимального элемента;
- 8) найти номер последнего максимального элемента;
- 9) подсчитать сумму элементов, расположенных между максимальным и минимальным элементами (минимальный и максимальный элементы в массиве единственные); если максимальный элемент встречается позже минимального, то выдать сообщение об этом;
- 10) найти номер первого максимального элемента;
- 11) найти номер последнего минимального элемента;
- 12) подсчитать сумму элементов, расположенных между первым максимальным и последним минимальными элементами; если максимальный элемент встречается позже минимального, то выдать сообщение об этом;
- 13) поменять местами первый минимальный и последний максимальный элементы;
- 14) найти максимум из отрицательных элементов;
- 15) найти минимум из положительных элементов;
- 16) найти максимум из модулей элементов;
- 17) найти количество пар соседних элементов, разность между которыми равна заданному числу;

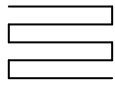
- 18) подсчитать количество элементов, значения которых больше значения предыдущего элемента;
- 19) найти количество пар соседних элементов, в которых предыдущий элемент кратен последующему;
- 20) найти количество пар соседних элементов, в которых предыдущий элемент меньше последующего.

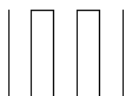
Замечание

Задачи из данного пункта решить, используя одномерный массив.

Задание 3

Для заданного массива А размером $n \times n$, элементы которого целые числа,

- 1) подсчитать среднее арифметическое нечетных элементов, расположенных выше главной диагонали;
- 2) подсчитать среднее арифметическое четных элементов, расположенных ниже главной диагонали;
- 3) подсчитать сумму элементов, расположенных на побочной диагонали;
- 4) подсчитать среднее арифметическое ненулевых элементов, расположенных над побочной диагональю;
- 5) подсчитать среднее арифметическое элементов, расположенных под побочной диагональю;
- 6) поменять местами столбцы по правилу: первый с последним, второй с предпоследним и т.д.;
- 7) поменять местами две средних строки, если количество строк четное, и первую со средней строкой, если количество строк нечетное;
- 8) поменять местами два средних столбца, если количество столбцов четное, и первый со средним столбцом, если количество столбцов нечетное;
- 9) если количество строк в массиве четное, то поменять строки местами по правилу: первую строку со второй, третью – с четвертой и т.д.; если количество строк в массиве нечетное, то оставить массив без изменений.
- 10) если количество столбцов в массиве четное, то поменять столбцы местами по правилу: первый столбец со вторым, третий – с четвертым и т.д.; если количество столбцов в массиве нечетное, то оставить массив без изменений.
- 11) вычислить A^n , где n – натуральное число;
- 12) подсчитать норму матрицы по формуле $\|A\| = \sum_i \max_j a_{i,j}$;
- 13) подсчитать норму матрицы по формуле $\|A\| = \sum_j \max_i a_{i,j}$;
- 14) вывести элементы матрицы в следующем порядке: ;
- 15) выяснить, является ли матрица симметричной относительно главной диагонали;
- 16) заполнить матрицу числами от 1 до n (где $n = m \times k$, m – количество строк, а k – количество столбцов прямоугольной матрицы) следующим образом:



- 17) определить, есть ли в данном массиве строка, состоящая только из положительных элементов;
- 18) определить, есть ли в данном массиве столбец, состоящий только из отрицательных элементов;
- 19) в каждой строке найти максимум и заменить его на противоположный элемент;
- 20) в каждом столбце найти минимум и заменить его нулем.

Замечание

При решении задач из данного пункта использовать или двумерные, или ступенчатые массивы. Свой выбор обосновать.

Задание 4

Для заданного массива размером $n \times n$, элементы которого являются целыми числами

- 1) найти максимальный элемент в каждой строке и записать данные в новый массив;
- 2) найти минимальный элемент в каждом столбце и записать данные в новый массив;
- 3) четные столбцы таблицы заменить на вектор X ;
- 4) нечетные строки таблицы заменить на вектор X ;
- 5) поменять местами элементы главной и побочной диагонали;
- 6) для каждой строки подсчитать количество положительных элементов и записать данные в новый массив;
- 7) для каждого столбца подсчитать сумму отрицательных элементов и записать данные в новый массив;
- 8) для каждого столбца подсчитать сумму четных положительных элементов и записать данные в новый массив;
- 9) для каждой строки подсчитать количество элементов, больших заданного числа, и записать данные в новый массив;
- 10) для каждого столбца найти первый положительный элемент и записать данные в новый массив;
- 11) для каждой строки найти последний четный элемент и записать данные в новый массив;
- 12) для каждого столбца найти номер последнего нечетного элемента и записать данные в новый массив;
- 13) для каждой строки найти номер первого отрицательного элемента и записать данные в новый массив;
- 14) для каждой строки найти сумму элементов с номерами от k_1 до k_2 и записать данные в новый массив;
- 15) для каждого столбца найти произведение элементов с номерами от k_1 до k_2 и записать данные в новый массив;
- 16) для каждой строки подсчитать сумму элементов, не попадающих в заданный интервал, и записать данные в новый массив;
- 17) подсчитать сумму элементов каждой строки и записать данные в новый массив; найти максимальный элемент нового массива;
- 18) подсчитать произведение элементов каждого столбца и записать данные в новый массив; найти минимальный элемент нового массива;
- 19) для каждой строки найти номер первой пары неравных элементов; данные записать в новый массив;
- 20) для каждого столбца найти номер первой пары одинаковых элементов; данные записать в новый массив.

Замечание

Для хранения массива использовать или двумерный, или ступенчатый массив. Свой выбор обосновать.

Задание 5

В одномерном массиве, элементы которого являются целыми числами, произвести следующие действия:

- 1) удалить из массива все четные числа;
- 2) удалить из массива все максимальные элементы;
- 3) удалить из массива все числа, значения которых попадают в данный интервал;
- 4) удалить из массива все элементы, последняя цифра которых равна данной;
- 5) удалить из массива элементы с номера k_1 по номер k_2 ;
- 6) вставить новый элемент перед первым отрицательным элементом;
- 7) вставить новый элемент после последнего положительного;
- 8) вставить новый элемент перед всеми четными элементами;
- 9) вставить новый элемент после всех элементов, которые заканчиваются на данную цифру;
- 10) вставить новый элемент после всех элементов, кратных своему номеру;
- 11) удалить из массива все элементы, в записи которых все цифры различны;
- 12) удалить из массива повторяющиеся элементы, оставив только их первые вхождения;
- 13) вставить новый элемент после всех максимальных;
- 14) вставить новый элемент перед всеми элементами, в записи которых есть данная цифра;
- 15) вставить новый элемент между всеми парами элементов, имеющими разные знаки.

Задание 6

В массиве размером $n \times n$, элементы которого являются целыми числами, произвести следующие действия:

- 1) вставить новую строку после строки, в которой находится первый встреченный минимальный элемент;
- 2) вставить новый столбец после столбца, в котором нет ни одного отрицательного элемента;
- 3) вставить новую строку после всех строк, в которых нет ни одного четного элемента;
- 4) вставить новый столбец перед всеми столбцами, в которых встречается заданное число;
- 5) вставить строку из нулей после всех строк, в которых нет ни одного нуля;
- 6) удалить все строки, в которых нет ни одного четного элемента;
- 7) удалить все столбцы, в которых первый элемент больше последнего;
- 8) удалить все строки, в которых сумма элементов не превышает заданного числа;
- 9) удалить все столбцы, в которых четное количество нечетных элементов;
- 10) удалить все строки, в которых каждый элемент попадает в заданный интервал;
- 11) удалить все столбцы, в которых все элементы положительны;
- 12) удалить все строки, в которых среднее арифметическое элементов является двузначным числом;
- 13) удалить строку и столбец, на пересечении которых стоит минимальный элемент (минимальный элемент встречается в массиве только один раз);
- 14) удалить из массива k -тую строку и j -тый столбец, если их значения совпадают;
- 15) уплотнить массив, удалив из него все строки и столбцы, состоящие из одних нулей.

Замечание

Для хранения массива $n \times n$ использовать или двумерный, или ступенчатый массив. Свой выбор обосновать.

EPAM Systems