МАССИВЫ

Массив — набор элементов одного и того же типа, объединенных общим именем. Массивы в C# можно использовать по аналогии с тем, как они используются в других языках программирования, например, в C++ или Pascal. Однако C#-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 C# как объекты, для которых разработан большой набор методов, реализующих различные алгоритмы обработки элементов массива.

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

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 C# 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 C# 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; 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 C# элементам массива присваиваются начальные значения по умолчанию 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 - четырем. 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 данного 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 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());
        myArray=new int [n];        //выделяем память под массив требуемой длины
        for (int i=0; i<n; i++)     //вводим элементы массива с клавиатуры
        {
                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()
{
   Randomrnd = new Random();      //инициализируем генератор случайных чисел
   int[] myArray;
   int n = rnd.Next(5, 10);      //генерируем случайное число из диапазона [5..10)
   myArray = new int[n];
   for (int i = 0; i < n; i++)
   {
       myArray[i] = rnd.Next(10);      // заполняем массив случайными числами
   }
   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 C# 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.

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.
    //Размерность массива 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. А 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***
BinarySearch	статический метод	Осуществляет двоичный поиск в отсортированном массиве.
Clear	статический метод	Присваивает элементам массива значения, определенные по умолчанию, т.е для арифметических типов нули, для ссылочных типов null.

Copy	статический метод	Копирует элементы одного массива в другой массив.
CopyTo	экземплярный метод	Копирует все элементы текущего одномерного массива в другой массив.
IndexOf	статический метод	Осуществляет поиск первого вхождения элемента в одномерный массив. Если элемент найден, то возвращает его индекс, иначе возвращает значение -1.
LastIndexOf	статический метод	Осуществляет поиск последнего вхождения элемента в одномерный массив. Если элемент найден, то возвращает 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.*

Задания.
1. Добавьте в программу метод Input, предназначенный для ввода с клавиатуры элементов массива. Продемонстрируйте работу данного метода.
2. Самостоятельно изучите остальные члены класса Array и продемонстрируйте их работу.

Использование спецификатора params

Иногда бывает необходимо создавать 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.

Задания.
1. В строке 1 записано условие b.Length!=0.Подумайте, можно ли обойтись без этого условия.
2. Измените метод F так, чтобы в параметр s записывалось 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 определяет номер строки, 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 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 и, в квадратных скобках, номер строки и 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 методуGetLength 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 значения 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: staticint[,] 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}};
```

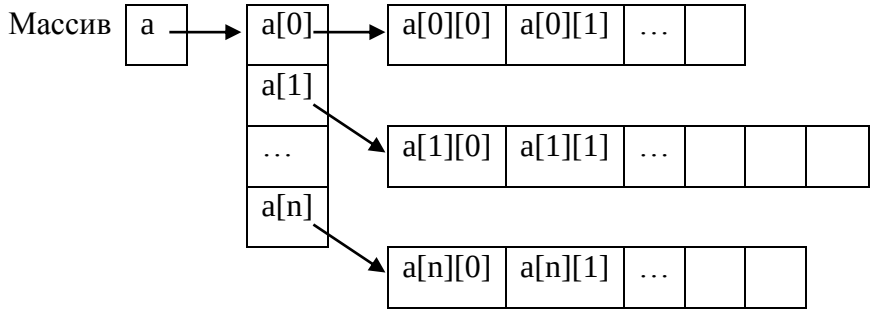
Какое значение будет выведено на экран в результате выполнения команды:

```
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:

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:

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

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: staticint[][] 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};
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?
6. Чему будет равно свойство Rank для рваного 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.

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×m.

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
       for (int i=0; i<a.Length; i++ ) //перебираем все элементы массива
       {
          if (a[i]<min) //если очередной элемент окажется меньше значения min,
          {
            min=a[i]; // то в качестве нового наименьшего значения запоминаем значение
            index=i; // текущего элемента 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

Задания.

1. Измените программу так, чтобы она вычисляла наибольший элемент в одномерном массиве и его номер.
2. Измените программу так, чтобы она работала для двумерного массива n×m.

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);
        constint 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×m.

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.

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 $A_{n\times n}$	Ответ
3	1 2 3	Сумма элементов главной диагонали = 15
	4 5 6	
	7 8 9	

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

6. Дана прямоугольная матрица n×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;
    for (int i = 0; i < (n / 2); i++)  //меняются местами  i-аяи (n-i-1)-ая строки
    {
      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 An×m	Ответ
4	3	1 2 3	0 1 2
		4 5 6	7 8 9
		7 8 9	4 5 6
		0 1 2	1 2 3

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

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.

Указания по решению задачи. *Массив результатов будет одномерный, а его размерность будет совпадать с количеством столбцов исходной матрицы.*

```
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 $A_{n\times m}$	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 его нечетных элементов для каждой строки и записывала полученные данные в новый массив.