Массивы. Текстовые файлы

От объектов к массивам. От массивов к “сложным” задачам. Текстовые файлы

[Массивы](#_kdae2flu2io3)

[Одномерные массивы](#_vk8czar7dhb1)

[Массив как параметр](#_a9rou6504ynh)

[Массив как объект](#_fmoa8l44ayhd)

[Двумерные массивы](#_51ahcwjohi45)

[Массив массивов](#_ksbudlfpe548)

[Индексаторы - индексируемые свойства](#_57ggpdwqbxo)

[Класс Array](#_s9qyb131amjk)

[Алгоритмы](#_131ri43ui7f5)

[Линейный поиск](#_5p54cvkmfij7)

[Бинарный поиск](#_wqk7ah2n6jpn)

[Работа с текстовыми файлами](#_5g0owr7vx1nv)

[Обработка исключений при работе с файлами](#_s8qaxg5na8vw)

[Практическая часть урока](#_ztn6o1tyxrp)

[Пример неправильной программы](#_gnve5wj5fipj)

[Задача 1. Правильная программа. Класс “Мой одномерный массив”](#_hm6c3738gcx)

[Задача 2. Массив и файл](#_n4m9vth6zvsu)

[Задача 3. Частотный массив](#_brojlfbpd6wh)

[Задача 4. Класс “Мой двумерный массив”](#_ym2fw2dfix0b)

[Задача 5. Задача на матрицу](#_zhr0icc4ce44)

[Домашнее задание](#_3tfrjxxltv85)

[Дополнительные материалы](#_q7971z70gyms)

[Используемая литература](#_c65ekg1ejcaz)

# 

# 

# Массивы

Массив — 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. Однако С#-массивы имеют существенные отличия: они относятся к ссылочным типам данных, более того - реализованы как объекты. Фактически имя массива является ссылкой на область кучи, в которой последовательно размещается набор элементов определенного типа. Выделение памяти под элементы происходит на этапе инициализации массива. А за освобождением памяти следит система сборки мусора - неиспользуемые массивы автоматически утилизируются данной системой.

Рассмотрим различные типы массивов.

## 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 его элементы будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Одномерный массив в С# является объектом, поэтому он создается в два этапа. Сначала объявляется ссылочная переменная на массив, затем выделяется память под требуемое количество элементов, и ссылочной переменной присваивается адрес на область памяти в которой расположен массив. Тип массива определяет тип данных каждого элемента массива. Количество элементов, которые будут храниться в массиве, определяется размером массива.

В общем случае процесс объявления переменной типа массив и выделение необходимого объема памяти может быть разделен. Кроме того, на этапе объявления массива можно произвести его инициализацию. Поэтому для объявления одномерного массива может использоваться одна из следующих форм записи:

|  |
| --- |
| class TestArraysClass {  static void Main()  {  // Описание объекта типа массив  int[] array0;  // Выделение места под массив  array0=new int[5];  // Объявление одномерного массива из 5 элементов  int[] array1 = new int[5];  // Объявление массива и заполнение его элементами  int[] array2 = new int[] { 1, 3, 5, 7, 9 };  // Альтернативный вариант создания заполненного массива  int[] array3 = { 1, 2, 3, 4, 5, 6 };     }  } |

Обращения к элементам массива происходит с помощью индекса, для этого нужно указать имя массива и в квадратных скобках его номер. Например, a[0], b[10], c[i].

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

Пример 1. Вывод массива.

|  |
| --- |
| using System;  class Program  {  static void Main()  {  int[] a = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };  int i;  for (i = 0; i < 10; i++)  Console.Write("{0} ", a[i]);  Console.WriteLine();  }  } |

Пример 2. Возведение каждого элемента массива в квадрат.

|  |
| --- |
| using System;  class Program  {  static void Main()  {  int[] a = new int[10];  int i;  for (i = 0; i < 10; i++) a[i] = i \* i;  for (i = 0; i < 10; i++) Console.WriteLine(a[i]);  Console.WriteLine();  }  } |

## 

## 

## 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, являющимся фактическим параметром.

Рассмотрим пример передачи массива как параметра:

|  |
| --- |
| using System;  class Program  {  static void Print(int n, int[] a)  // n–размерность массива, а–ссылка на массив  {  for (int i = 0; i < n; i++) Console.Write("{0} ", a[i]);  Console.WriteLine();  }  static void Change(int n, int[] a)  {  for (int i = 0; i < n; i++)  if (a[i] < 0) a[i] = -a[i]  // изменяются элементы массива  }  static void Main()  {  int[] myArray = { 0, -1, -2, 3, 4, 5, -6, -7, 8, -9 };  Print(10, myArray);  Change(10, myArray);  Print(10, myArray);  }  } |

## 

## 

## Массив как объект

Мы уже говорили о том, что массивы в С# 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. Преобразуем предыдущий пример:

|  |
| --- |
| class Program  {  static void Print(int[] a) // передаем только ссылку на массив  {  for (int i = 0; i < a.Length; i++) Console.Write("{0} ", a[i]);  Console.WriteLine();  }  static void Change(int[] a)  {  for (int i = 0; i < a.Length; i++)  if (a[i] < 0) a[i] = -a[i];  }  static void Main()  {  int[] A = { 0, -1, -2, 3, 4, 5, -6, -7, 8, -9 };  Print(A);  Change(A);  Print(A);  }  } |

## Двумерные массивы

Для работы с двумерными массива используются два измерения и требуется два индекса. Таблица Excel - хороший аналог двумерного массива, только в двумерных массивах и номер столбца, и номер строки - это числа. Для работы в с двумерными, да и вообще с многомерными массивами, чаще всего используются вложенные циклы

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  // Объявление двумерного массива  int[,] multiDimensionalArray1 = new int[2, 3];  // Объявление и заполнение двумерного массива  int[,] multiDimensionalArray2 = { { 1, 2, 3 }, { 4, 5, 6 } };  }  } |

Примеры обработки двумерного массива смотрите в разделе “Практика”.

## Массив массивов

Так же можно создавать ступенчатые массивы или так называемые массивы массивов. Это одномерный массив. Каждый элемент в этом массиве является ссылкой на другой одномерный массив.

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  // Объявление массива массивов(ступенчатый массив)  int[][] jaggedArray = new int[3][];  // Пример заполнения первого элемента ступенчатого массива  jaggedArray[0] = new int[4] { 1, 2, 3, 4 };  jaggedArray[1] = new int[5] { 1, 2, 3, 4, 5 };  jaggedArray[2] = new int[3] { 1, 2, 3 };  }  } |

## 

## Индексаторы - индексируемые свойства

Так как мы изучаем массивы, отдельно стоит упомянуть специальные свойства - индексаторы. Если вы поняли, для чего нужны свойства, то легко поймете и индексируемые свойства. Например, у нас есть класс, внутри которого есть поле - массив, но поле объявлено приватным. Чтобы получить доступ к элементу массива, можно либо сделать публичный метод, который будет возвращать элемент массива по его номеру, либо сделать индексируемое свойство.

Пример:

|  |
| --- |
| using System;  namespace IndexerSample  {  class MyArray  {  int[] a; // он приватный  public MyArray(int n)  {  a = new int[n];  }  // либо мы делаем метод для получения элемента массива  public int Get(int i)  {  return a[i];  }  // и метод для того, чтобы задать элемент  public void Set(int i, int value)  {  a[i] = value;  }  // либо создаем индексируемое свойство  public int this[int i]  {  get { return a[i]; }  set { a[i] = value; }  }  }  class Program  {  static void Main(string[] args)  {  MyArray array = new MyArray(10);  for (int i = 0; i < 10; i++)  // Для доступа можно использовать либо метод  array.Set(i, i\*10);  for (int i = 0; i < 10; i++)  // Или индексируемое свойство, что более удобно  array[i] = i \* 10;  }  }  } |

# Класс Array

Для работы с массивами существует класс Array. В этом классе собраны методы для обработки массивов. В ООП класс Array является базовым классом для всех массивов в .NET Framework.

Некоторые методы класса Array

|  |  |  |
| --- | --- | --- |
| **Имя метода** | **Тип** | **Описание** |
| BinarySearch | Статический, перегруженный | Выполняет поиск значения в отсортированном массиве. |
| Clear | Статический | Присваивает элементам массива значение 0, false или null в зависимости от типа элементов. |
| Copy | Статический, перегруженный | Копирует элементы из одного массива в другой. |
| CopyTo | Не статический, перегруженный | Копирует элементы текущего одномерного массива в другой массив. |
| GetLength | Не статический | Получает целое число, представляющее количество элементов в заданном измерении. |
| IndexOf | Статический, перегруженный | Выполняет поиск указанного объекта внутри всего одномерного массива, и возвращает индекс его первого вхождения. |
| Reverse | Статический, перегруженный | Изменяет порядок элементов во всем одномерном массиве Array на обратный. |
| Sort | Статический, перегруженный | Сортирует элементы во всем одномерном массиве Array |

# Алгоритмы

## Линейный поиск

Линейный алгоритм последовательно просматривает каждый элемент массива. Если элемент массива не соответствует искомому, то алгоритм переходит к следующему элементу, а при достижении конца массива алгоритм сообщает, что искомый элемент не найден. Если элемент найден, алгоритм сообщает индекс найденного элемента.

Для примера рассмотрим массив, содержащий следующие элементы:

45 67 1 12 66 51 93 30 5 52

Допустим, метод ищет значение 51. При использовании линейного алгоритма поиска метод сначала проверяет, совпадает ли 45 с искомым элементом. Совпадения нет, поэтому алгоритм проверяет, совпадает ли 67 с искомым элементом. Последовательное перемещение по массиву продолжается; метод проверяет 1, затем 12 и 66. При достижении значения 51, совпадающего с искомым элементом, метод возвращает индекс 5 — местонахождение 51 в массиве. Если после проверки всех элементов метод определяет, что искомый элемент не совпадает ни с одним элементом в массиве, он возвращает –1. При наличии дубликатов в массиве линейный поиск возвращает индекс первого элемента, совпадающего с искомым элементом.

## Бинарный поиск

Суть бинарного поиска в том, чтобы разделить последовательность на две части. Тогда искомый элемент будет находиться в одной из двух частей. Продолжаем делить последовательность, пока не найдем искомый элемент.

В худшем случае поиск в отсортированном массиве с 1023 элементами при бинарном поиске потребует около 10 сравнений. Многократное деление 1023 на 2 (потому что после каждого сравнения половина массива исключается из рассмотрения) с округлением вниз (так как средний элемент тоже исключается) дает значения 511, 255, 127, 63, 31, 15, 7, 3, 1 и 0. Число 1023 (210 — 1) всего после 10 делений на 2 уменьшилось до 0 (признак того, что элементов для проверки не осталось). Деление на 2 эквивалентно одному сравнению в алгоритме бинарного поиска. Таким образом, для поиска ключа в массиве с 1 048 575 (220 — 1) элементами достаточно всего 20 сравнений, а в массиве с миллиардом элементов (что менее 230 — 1) для поиска ключа потребуется не более 30 сравнений. Это огромный выигрыш по сравнению с линейным поиском: в массиве с миллиардом элементов при линейном поиске потребуется около 500 миллионов сравнений, тогда как с бинарным поиском хватит всего 30!

Правда требуется учесть, что элементы в массиве обязательно должны быть отсортированы, иначе бинарный поиск не будет работать.

# Работа с текстовыми файлами

Для чтения данных из файла в .Net Framework используется класс StreamReader из пространства имен System.IO. Для записи используется класс StreamReader из того же пространства имен.

Задача. Из файла данных на диске считать последовательность чисел и вывести числа на экран. Первым числом в файле стоит количество чисел N. Далее идет последовательность чисел. Каждое число в новой строке.

Например: data.txt

4  
1  
2  
3  
4

|  |
| --- |
| using System;  using System.IO; // Обязательно еще одно пространство имен.  class Program  {  static void Main()  {  // Создаем объект sr и связываем его с файлом data.txt.  StreamReader sr = new StreamReader("..\\..\\data.txt");  // Считаем количество чисел.  int n = int.Parse(sr.ReadLine());  for (int i = 0; i < n; i++)  {  int a = int.Parse(sr.ReadLine());  Console.WriteLine(a);  }  // Освобождаем файл data.txt для использования другими программами.  sr.Close();  }  } |

### 

### 

### Обработка исключений при работе с файлами

Задача. Дан текстовый файл, состоящий из различных данных (числа, строки). Считать файл и найти среднее арифметическое всех чисел.

|  |
| --- |
| using System;  using System.IO;  namespace ReadFromFileWithException  {  class Program  {  static void Main(string[] args)  {  StreamReader sr = new StreamReader("..\\..\\data.txt");  int sum = 0, count = 0;  while (!sr.EndOfStream) // Пока не конец потока (файла)  {  string s = sr.ReadLine();  Console.WriteLine("Считали строку:" + s);  try  {  int a = int.Parse(s);  sum = sum + a;  count++;  Console.WriteLine("{0}.Преобразовали в число:{1}", count, a);  }  // В экземпляре exc класса Exception будет  // храниться информация об ошибке.  catch (Exception exc)  {  Console.WriteLine(exc.Message);  }  }  sr.Close();  Console.WriteLine("Среднее арифметическое:{0:f2}", (double)sum / count);  // Обратите внимание! Если не поставить явное приобразование типов перед sum, то sum/count получит целое число. Попробуйте убрать (double) перед sum.  }  }  } |

# 

# 

# Практическая часть урока

### Задача 1. Правильная программа. Класс “Мой одномерный массив”

Разработать класс для работы с одномерным массивом. Предусмотреть конструктор, заполняющий массив конкретными значениями и конструктор заполняющий массив случайными числами. Сделать методы поиска среднего значения, максимального элемента массива, минимального элемента массива, подсчета количества положительных и метод, возвращающий массив в виде строки.

|  |
| --- |
| using System;  namespace ArrayClass  {  class MyArray  {  int[] a;  // Создание массива и заполнение его одним значением el  public MyArray(int n,int el)  {  a = new int[n];  for (int i = 0; i < n; i++)  a[i] = el;  }  // Создание массива и заполнение его случайными числами от min до max  public MyArray(int n, int min,int max)  {  a = new int[n];  Random rnd = new Random();  for (int i = 0; i < n; i++)  a[i] = rnd.Next(min,max);  }  public int Max  {  get  {  int max = a[0];  for (int i = 1; i < a.Length; i++)  if (a[i] > max) max = a[i];  return max;  }  }  public int Min  {  get  {  int min = a[0];  for (int i = 1; i < a.Length; i++)  if (a[i] < min) min = a[i];  return min;  }  }  public int CountPositiv  {  get  {  int count = 0;  for (int i = 0; i < a.Length; i++)  if (a[i] > 0) count++;  return count;  }  }  public override string ToString()  {  string s = "";  foreach (int v in a)  s=s+v+" ";  return s;  }  }  class Program  {  static void Main(string[] args)  {  MyArray a = new MyArray(10, 0, 100);  Console.WriteLine(a.ToString());  Console.WriteLine(a.Max);  Console.WriteLine(a.Min);  Console.WriteLine(a.CountPositiv);  }  }  } |

### 

### 

### Задача 2. Массив и файл

Дана последовательность целых чисел, записанная в текстовый файл. Требуется считать данные из файла в массив, найти среднее арифметическое элементов и вывести минимальный и максимальный элементы массива на экран. Отсортировать массив.

|  |
| --- |
| using System;  using System.IO;  namespace SeriesN\_SortMaxMinMiddle  {  class MyArray  {  int[] a;  public MyArray(string filename)  {  StreamReader sr = new StreamReader("..\\..\\data.txt");  // Считываем количество элементов массива  int N = int.Parse(sr.ReadLine());  a = new int[N];  // Считываем массив  for (int i = 0; i < N; i++)  {  a[i] = int.Parse(sr.ReadLine());  }  sr.Close();  }  public int Length  {  get  {  return a.Length;  }  }  public double Sum  {  get  {  double sum = 0;  foreach (int el in a)  sum += el;  return sum;  }  }  public int Max  {  get  {  // Находим максимальный элемент  int max = a[0];  for (int i = 1; i < a.Length; i++)  if (a[i] > max) max = a[i];  return max;  }  }  public int Min  {  get  {  // Находим минимальный элемент  int min = a[0];  for (int i = 1; i < a.Length; i++)  if (a[i] < min) min = a[i];  return min;  }  }  public void BubbleSort()  {  // Сортируем используя сортировку методом "пузырька"  for (int i = 0; i < a.Length; i++)  for (int j = 0; j < a.Length - 1; j++)  if (a[j] > a[j + 1])//Сравниваем соседние элементы  {  // Обмениваем элементы местами  int t = a[j];  a[j] = a[j + 1];  a[j + 1] = t;  }  }  public void Print()  {  foreach (int el in a)  Console.Write("{0,4}", el);  }  public void Print(string msg)  {  Console.WriteLine(msg);  Print();  }  }  class Program  {  static void Main(string[] args)  {  MyArray a = new MyArray("data.txt");  a.Print();  Console.WriteLine("\nMax:{0}", a.Max);  Console.WriteLine("Min:{0}", a.Min);  Console.WriteLine("Middle:{0}\n", a.Sum/a.Length);  a.BubbleSort();  a.Print("Отсортированный массив");  Console.WriteLine();  }  }  } |

### 

### 

### Задача 3. Частотный массив

Дана последовательность натуральных чисел от 0 до 99. Найти какое число встречается чаще всего. Если таких чисел несколько, то вывести все числа.

|  |
| --- |
| using System;  // В частотном массиве мы как бы выворачиваем массив наизнанку.  // В частотном массиве индексы массива соответствуют элементам массива,  //а значения - это количество этих элементов.  // Поэтому нужно понять, что размер частотного массива связан с диапазоном чисел, которые мы //подсчитываем  namespace SeriesN\_ЧастотныйМассив  {  class Program  {  static void Main(string[] args)  {  Random rnd = new Random();  int N = 10;  int[] a = new int[N];  // Заполняем массив случайными числами  for(int i=0;i< N;i++)  a[i] = rnd.Next(0, 100);  // Выводим массив на экран  foreach(var v in a)  Console.Write(v + " ");  // Создаем частотный массив от 0 до 99  int[] mass = new int[100];  // Подсчитываем вхождение элементов  foreach (var v in a) mass[v]++;//Элемент массива a является номером в частотном массиве mass  // Находим максимальный элемент в частотном массиве  int imax = 0;  for(int i=0;i<mass.Length;i++)  if (mass[i] > mass[imax]) imax = i;  // Выводим все элементы, которые в частотном массиве встречались тоже количество раз, что и imax количество раз  for (int i = 0; i < mass.Length; i++)  if (mass[i]==mass[imax]) Console.WriteLine("\n"+i);  }  }  } |

### 

### 

### Задача 4. Класс “Мой двумерный массив”

Разработать класс для работы с двумерным массивом. Сделать методы поиска среднего значения, максимального элемента массива, минимального элемента массива, подсчета количества положительных, вывода массива на экран и метод, возвращающий массив в виде строки.

|  |
| --- |
| using System;  namespace ArrayTwoDimensionClass  {  class MyArrayTwoDim  {  int[,] a;  public MyArrayTwoDim(int n,int el)  {  a = new int[n, n];  for (int i = 0; i < n; i++)  for (int j = 0; j < n; j++)  a[i, j] = el;  }  public MyArrayTwoDim(int n, int min,int max)  {  a = new int[n, n];  Random rnd = new Random();  for (int i = 0; i < n; i++)  for (int j = 0; j < n; j++)  a[i, j] = rnd.Next(min,max);  }  public int Min  {  get  {  int min = a[0, 0];  // Находим минимальный элемент  // В двухмерном массиве для получения размерности нужно использовать  // метод GetLength, а в скобках указывать измерение  for (int i = 0; i < a.GetLength(0); i++)  for (int j = 0; j < a.GetLength(1); j++)  if (a[i, j] < min) min = a[i, j];  return min;  }  }  public int Max  {  get  {  int max = a[0, 0];  for (int i = 0; i < a.GetLength(0); i++)  for (int j = 0; j < a.GetLength(1); j++)  if (a[i, j] > max) max = a[i, j];  return max;  }  }  // Свойство - подсчет количества положительных  public int CountPositive  {  get  {  int count = 0;  for (int i = 0; i < a.GetLength(0); i++)  for (int j = 0; j < a.GetLength(1); j++)  if (a[i, j] > 0) count++;  return count;  }  }  // Свойство - подсчет среднего арифметического  public double Average  {  get  {  double sum = 0;  for (int i = 0; i < a.GetLength(0); i++)  for (int j = 0; j < a.GetLength(1); j++)  sum+=a[i,j];  return sum/a.GetLength(0)/a.GetLength(1);  }  }  // Метод - который возвращает массив в виде строки  public string ToString()  {  string s="";  for (int i = 0; i < a.GetLength(0); i++)  {  for (int j = 0; j < a.GetLength(1); j++)  s += a[i, j] + " ";  s += "\n"; // Переход на новую строчку  }  return s;  }  }  class Program  {  static void Main(string[] args)  {  MyArrayTwoDim a = new MyArrayTwoDim(2, 0, 10);  Console.WriteLine(a.ToString());  Console.WriteLine("\nМаксимальный элемент:" + a.Max);  Console.WriteLine("Минимальный элемент:" + a.Min);  Console.WriteLine("Среднее значение элементов:" + a.Average);  }  }  } |

### 

### 

### Задача 5. Задача на матрицу

Дан прямоугольный массив целых положительных чисел N х M, заполненный случайными числами. Описать на русском языке или на одном из языков программирования алгоритм поиска строки с наименьшей суммой элементов. Вывести на печать номер строки и сумму ее элементов. Если таких строк несколько, вывести информацию о каждой строке.

Пример:

|  |  |
| --- | --- |
| **Input** | **Output** |
| -1 -4 4 3 1 3 -1 3 -2 -5 -5 0 -5 2 0 2 0 -5 -3 -1  4 -4 4 -2 1 -3 3 -5 0 -1 3 0 3 -2 -2 -4 2 2 -3 -3  -4 4 -5 -3 0 -5 -1 1 -4 -2 -4 -3 -2 2 -5 -3 -3 1 -3 -2  -5 -2 -2 3 -3 -1 1 2 -1 -2 4 3 3 -5 -2 -5 -4 1 2 1  3 -3 4 -4 -2 0 -4 -4 -2 -5 4 -4 0 3 -3 3 1 -4 -5 -5  -2 2 -4 3 -1 -2 -2 -2 -1 4 -4 0 3 0 0 3 -2 -4 0 1  -3 0 -2 -1 -2 3 -2 -5 3 -1 -1 4 1 2 -3 -4 -5 -4 2 -5  -4 3 2 -2 3 -2 -3 4 -2 -2 -1 -1 0 -2 2 3 -2 2 -1 1  -5 -2 0 4 0 -3 4 -4 -2 -1 2 -3 -1 0 0 -2 3 -5 0 -5  0 -1 2 -1 -5 4 -2 -3 4 -1 3 -3 1 -5 -5 3 -2 2 2 1 | 3 -41 |

Напишем решение в стиле ООП.

Создадим класс Matrix. Массив с данными, а также вспомогательный массив с суммой строк сделаем полями этого класса. Создадим конструктор, в котором будем заполнять эти поля. Добавим несколько методов для работы с этими полями.

|  |
| --- |
| using System;  namespace EGE\_Matrix  {  class Matrix  {  int[,] a; // Матрица  int[] Rows; // Сумма строк этой матрицы  public Matrix(int n, int m)  {  a = new int[n, m];  Random rnd = new Random();  Rows = new int[n];  for (int i = 0; i < n; i++)  {  int s = 0;  for (int j = 0; j < m; j++)  {  a[i, j] = rnd.Next(0, 10);  // Подсчет сумм каждой строки  s += a[i, j];  }  // Сохранение суммы для каждой строки  Rows[i] = s;  }  }  // Вывод матрицы на экран  public void Print()  {  for (int i = 0; i < a.GetLength(0); i++)  {  for (int j = 0; j < a.GetLength(1); j++)  Console.Write("{0,4}", a[i, j]);  Console.WriteLine();  }  }  // Метод Search находит минимальную сумму и возвращает количество таких строк  public int Search(out int count)  {  int min = int.MaxValue;  count = 0; // count описывать не нужно  foreach (int e in Rows)  {  if (e < min) min = e;  }  foreach (int e in Rows)  {  if (e == min) count++;  }  return min;  }  // Находим все строки с такой же суммой  public void SearchRows()  {  int countRow;// Количество минимальных элементов  int min = Search(out countRow); ;  for (int i = 0; i < Rows.Length; i++)  {  if (Rows[i] == min)  {  Console.WriteLine("\n{0} {1}", i, Rows[i]);  }  }  }  }  class Program  {  static void Main(string[] args)  {  Matrix a = new Matrix(5, 2);  a.Print();  a.SearchRows();  Console.ReadKey();  }  }  } |

### 

### 

### Задача 6

Разработать класс для работы с одномерным массивом. Создать конструктор для заполнения массива случайными числами и конструктор для заполнения массива из файла. Создать свойство возвращающее максимальный элемент. Реализовать индексируемое свойство.

|  |
| --- |
| using System;  using System.Linq;  using System.IO;  namespace CoolArray  {  class CoolArray  {  private int[] a;  Random rnd = new Random();  public CoolArray(int n)  {  a = new int[n];  for (int i = 0; i < n; i++)  a[i] = rnd.Next(1, 101);  }  public CoolArray(string filename)  {  //Если файл существует  if (File.Exists(filename))  {  //Считываем все строки в файл  string[] ss = File.ReadAllLines(filename);  a = new int[ss.Length];  //Переводим данные из строкового формата в числовой  for (int i = 0; i < ss.Length; i++)  a[i] = int.Parse(ss[i]);  }  else Console.WriteLine("Error load file");  }  public int Max  {  get  {  return a.Max();  }  }  public int this[int i]  {  get { return a[i]; }  set { a[i] = value; }  }  public void Print()  {  foreach (int el in a)  Console.Write("{0,4}", el);  }  }  class Program  {  static void Main(string[] args)  {  CoolArray array = new CoolArray(5);  array.Print();  Console.WriteLine();  Console.WriteLine("Максимальный элемент: " + array.Max);  Console.WriteLine("array[3]: " + array[3]);  Console.ReadKey();  }  }  } |

# 

# Домашнее задание

1. Дан целочисленный массив из 20 элементов. Элементы массива могут принимать целые значения от –10 000 до 10 000 включительно. Написать программу, позволяющую найти и вывести количество пар элементов массива, в которых хотя бы одно число делится на 3. В данной задаче под парой подразумевается два подряд идущих элемента массива. Например, для массива из пяти элементов: 6; 2; 9; –3; 6 – ответ: 4.
2. а) Дописать класс для работы с одномерным массивом. Реализовать конструктор, создающий массив заданной размерности и заполняющий массив числами от начального значения с заданным шагом. Создать свойство Sum, которые возвращают сумму элементов массива, метод Inverse, меняющий знаки у всех элементов массива, Метод Multi, умножающий каждый элемент массива на определенное число, свойство MaxCount, возвращающее количество максимальных элементов. В Main продемонстрировать работу класса.  
   б)\*Добавить конструктор и методы, которые загружают данные из файла и записывают данные в файл.
3. Решить задачу с логинами из предыдущего урока, только логины и пароли считать из файла в массив.
4. \*а) Реализовать класс для работы с двумерным массивом. Реализовать конструктор, заполняющий массив случайными числами. Создать методы, которые возвращают сумму всех элементов массива, сумму всех элементов массива больше заданного, свойство, возвращающее минимальный элемент массива, свойство, возвращающее максимальный элемент массива, метод, возвращающий номер максимального элемента массива (через параметры, используя модификатор ref или out)  
   \*\*б) Добавить конструктор и методы, которые загружают данные из файла и записывают данные в файл.

Дополнительные задачи

1. \*\*Существует алгоритмическая игра “Удвоитель”. В этой игре человеку предлагается какое-то число, а человек должен, управляя роботом “Удвоитель”, достичь этого числа за минимальное число шагов. Робот умеет выполнять несколько команд: увеличить число на 1, умножить число на 2 и сбросить число до 1. Начальное значение удвоителя равно 1.   
    Реализовать класс “Удвоитель”. Класс хранит в себе поле current - текущее значение, finish - число, которого нужно достичь, конструктор, в котором задается конечное число. Методы: увеличить число на 1, увеличить число в два раза, сброс текущего до 1, свойство Current, которое возвращает текущее значение, свойство Finish,которое возвращает конечное значение. Создать с помощью этого класса игру, в которой компьютер загадывает число, а человек. выбирая из меню на экране, отдает команды удвоителю и старается получить это число за наименьшее число ходов. Если человек получает число больше положенного, игра прекращается.
2. \*\*\*Написать игру “Верю. Не верю”. В файле хранятся некоторые данные и правда это или нет. Например: “Шариковую ручку изобрели в древнем Египте”, “Да”.  
   Компьютер загружает эти данные, случайным образом выбирает 5 вопросов и задает их игроку.  
   Игрок пытается ответить правда или нет, то что ему загадали и набирает баллы. Список вопросов ищите во вложении или можно воспользоваться Интернетом.

Достаточно решить 3 задачи. Старайтесь разбивать программы на подпрограммы. Переписывайте в начало программы условие и свою фамилию. Все программы сделать в одном решении.

# 

# 

# Дополнительные материалы

1. [Индексаторы в C#](https://msdn.microsoft.com/ru-ru/library/6x16t2tx.aspx);
2. Пол Дейтел, Харви Дейтел, “Как программировать на Visual C# 2012”. Сортировка и поиск.

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Пол Дейтел, Харви Дейтел, “Как программировать на Visual C# 2012”, Питер 2014 г.;
2. Т.А. Павловская. “Программирование на языке высокого уровня”, 2009 г. ;
3. [Демоверсии ГИА по информатике](http://gia.edu.ru/ru/graduates_classes/demonstration/);
4. [Дистанционная подготовка к олимпиадам по информатике](http://informatics.mccme.ru/moodle/);
5. Демоверсии ЕГЭ по информатике;
6. [MSDN](https://msdn.microsoft.com/ru-ru/default.aspx).