Содержание

[Введение 3](#_Toc23851713)

[1. Обзор методов решения 4](#_Toc23851714)

[2. Блок-схема алгоритма и его описание 8](#_Toc23851715)

[3. Листинг программы 14](#_Toc23851716)

[4. Скриншот работы программы 16](#_Toc23851717)

[Заключение 18](#_Toc23851718)

[Список использованной литературы 19](#_Toc23851719)

# Введение

C# – язык платформы .Net. который постоянно развивается и впитывает в себя все новые возможности. На этом языке можно писать кроссплатформенные приложения под Linux, Mac, Android и iOS.

Язык программирования C# призван реализовать компонентно-ориентированный подход к программированию, который способствует меньшей машинно-архитектурной зависимости результирующего программного кода, большей гибкости, переносимости и легкости повторного использования программ.

Принципиально важным отличием от предшественников является изначальная ориентация на безопасность кода.

Расширенная поддержка событийно-ориентированного программиро­вания.

Язык программирования C# является «родным» для создания приложений в среде Microsoft .NET, поскольку наиболее тесно и эффективно интегрирован с ней.

Достоинства языка программирования C#:

– простота программирования;

– понятный синтаксис;

– подлинная объектная ориентированность;

– компонентно-ориентированное программирование;

– безопасный по сравнению с языками C и C++ код;

– унифицированная система типизации;

– поддержка событийно-ориентированного программирования;

– «родной» язык для создания приложений в среде .NET;

– объединение лучших идей современных языков программирования: Java, C++, Visual Basic и др.

К недостаткам же языка можно отнести следующее:

– приоритетная ориентированность на Windows платформу;

– язык бесплатен только для небольших фирм, индивидуальных программистов, стартапов и учащихся. Крупной компании лицензионную версию этого языка придется покупать;

– в языке осталась возможность использования оператора безусловного перехода.

Однако для реализации поставленной задачи недостатки данного языка не существенны, что позволяет не учитывать их при разработке программного комплекса для автоматизации рабочего места бурового мастера.

Язык программирования C# получил большую распространённость, потому что, имея простой и понятный синтаксис, позволяет более рационально создавать приложения. Язык C# интегрировал в себе преимущества языка Java и С++, что и обуславливает популярность данного языка. При этом в объединенном языке исключены некоторые спорные директивы, макросы, отменены глобальные переменные.

# 1. Обзор методов решения

Массив представляет набор однотипных данных. Объявление массива похоже на объявление переменной за тем исключением, что после указания типа ставятся квадратные скобки:

тип\_переменной[] название\_массива;

Например, определим массив целых чисел:

int[] numbers;

После определения переменной массива мы можем присвоить ей определенное значение:

int[] numbers=new int[4];

Здесь вначале мы объявили массив numbers, который будет хранить данные типа int. Далее используя операцию new, мы выделили память для 4 элементов массива: new int[4]. Число 4 еще называется длиной массива. При таком определении все элементы получают значение по умолчанию, которое предусмотрено для их типа. Для типа int значение по умолчанию - 0.

Также мы сразу можем указать значения для этих элементов:

int[] nums2 = new int[4] { 1, 2, 3, 5 };

int[] nums3 = new int[] { 1, 2, 3, 5 };

int[] nums4 = new[] { 1, 2, 3, 5 };

int[] nums5 = { 1, 2, 3, 5 };

Все перечисленные выше способы будут равноценны.

Для обращения к элементам массива используются индексы. Индекс представляет номер элемента в массиве, при этом нумерация начинается с нуля, поэтому индекс первого элемента будет равен 0. А чтобы обратиться к четвертому элементу в массиве, нам надо использовать индекс 3, к примеру: nums[3]. Используем индексы для получения и установки значений элементов массива:

int[] nums = new int[4];

nums[0] = 1;

nums[1] = 2;

nums[2] = 3;

nums[3] = 5;

Console.WriteLine(nums[3]); // 5

Массивы характеризуются таким понятием как ранг или количество измерений. Выше мы рассматривали массивы, которые имеют одно измерение (то есть их ранг равен 1) - такие массивы можно представлять в виде горизонтального ряда элемента. Но массивы также бывают многомерными. У таких массивов количество измерений (то есть ранг) больше 1.

Массивы, которые имеют два измерения (ранг равен 2) называют двухмерными. Ниже приведены примеры объявления двумерных массивов.

int[,] nums1;

int[,] nums2 = new int[2, 3];

int[,] nums3 = new int[2, 3] { { 0, 1, 2 }, { 3, 4, 5 } };

int[,] nums4 = new int[,] { { 0, 1, 2 }, { 3, 4, 5 } };

int[,] nums5 = new[,] { { 0, 1, 2 }, { 3, 4, 5 } };

int[,] nums6 = { { 0, 1, 2 }, { 3, 4, 5 } };

Для записи в текстовый файл используется класс StreamWriter. Некоторые из его конструкторов, которые могут применяться для создания объекта StreamWriter:

* StreamWriter(string path): через параметр path передается путь к файлу, который будет связан с потоком;
* StreamWriter(string path, bool append): параметр append указывает, надо ли добавлять в конец файла данные или же перезаписывать файл. Если равно true, то новые данные добавляются в конец файла. Если равно false, то файл перезаписываетсяя заново;
* StreamWriter(string path, bool append, System.Text.Encoding encoding): параметр encoding указывает на кодировку, которая будет применяться при записи.

Свою функциональность StreamWriter реализует через следующие методы:

* int Close(): закрывает записываемый файл и освобождает все ресурсы;
* void Flush(): записывает в файл оставшиеся в буфере данные и очищает буфер;
* Task FlushAsync(): асинхронная версия метода Flush;
* void Write(string value): записывает в файл данные простейших типов, как int, double, char, string и т.д. Соответственно имеет ряд перегруженных версий для записи данных элементарных типов, например, Write(char value), Write(int value), Write(double value) и т.д.;
* Task WriteAsync(string value): асинхронная версия метода Write
* void WriteLine(string value): также записывает данные, только после записи добавляет в файл символ окончания строки;
* Task WriteLineAsync(string value): асинхронная версия метода WriteLine.

Рассмотрим запись в файл на примере:

using System;

using System.IO;

namespace HelloApp

{

class Program

{

static void Main(string[] args)

{

string writePath = @"C:\SomeDir\hta.txt";

string text = "Привет мир!\nПока мир...";

try

{

using (StreamWriter sw = new StreamWriter(writePath, false, System.Text.Encoding.Default))

{

sw.WriteLine(text);

}

using (StreamWriter sw = new StreamWriter(writePath, true, System.Text.Encoding.Default))

{

sw.WriteLine("Дозапись");

sw.Write(4.5);

}

Console.WriteLine("Запись выполнена");

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

}

В данном случае два раза создаем объект StreamWriter. В первом случае если файл существует, то он будет перезаписан. Если не существует, он будет создан. И в нее будет записан текст из переменной text. Во втором случае файл открывается для дозаписи, и будут записаны атомарные данные - строка и число. В обоих случаях будет использоваться кодировка по умолчанию.

По завершении программы в папке C://SomeDir мы сможем найти файл hta.txt, который будет иметь следующие строки:

Привет мир!

Пока мир...

Дозапись

4,5

Для выполнения различных математических операций в библиотеке классов .NET предназначен класс Math. Он является статическим, поэтому все его методы также являются статическими.

* Abs(double value): возвращает абсолютное значение для аргумента value
* Acos(double value): возвращает арккосинус value. Параметр value должен иметь значение от -1 до 1
* Asin(double value): возвращает арксинус value. Параметр value должен иметь значение от -1 до 1
* Atan(double value): возвращает арктангенс value
* BigMul(int x, int y): возвращает произведение x \* y в виде объекта long
* Ceiling(double value): возвращает наименьшее целое число с плавающей точкой, которое не меньше value
* Cos(double d): возвращает косинус угла d
* Cosh(double d): возвращает гиперболический косинус угла d
* DivRem(int a, int b, out int result): возвращает результат от деления a/b, а остаток помещается в параметр result
* Exp(double d): возвращает основание натурального логарифма, возведенное в степень d
* Floor(decimal d): возвращает наибольшее целое число, которое не больше d
* IEEERemainder(double a, double b): возвращает остаток от деления a на b
* Log(double d): возвращает натуральный логарифм числа d
* Log(double a, double newBase): возвращает логарифм числа a по основанию newBase
* Log10(double d): возвращает десятичный логарифм числа d
* Max(double a, double b): возвращает максимальное число из a и b
* Min(double a, double b): возвращает минимальное число из a и b
* Pow(double a, double b): возвращает число a, возведенное в степень b
* Round(double d): возвращает число d, округленное до ближайшего целого числа
* Round(double a, round b): возвращает число a, округленное до определенного количества знаков после запятой, представленного параметром b
* Sign(double value): возвращает число 1, если число value положительное, и -1, если значение value отрицательное. Если value равно 0, то возвращает 0
* Sin(double value): возвращает синус угла value
* Sinh(double value): возвращает гиперболический синус угла value
* Sqrt(double value): возвращает квадратный корень числа value
* Tan(double value): возвращает тангенс угла value
* Tanh(double value): возвращает гиперболический тангенс угла value
* Truncate(double value): отбрасывает дробную часть числа value, возвращая лишь целое значение
* Также класс Math определяет две константы: Math.E и Math.PI.

# 2. Блок-схема алгоритма и его описание

Для решения поставленной задачи, она была разбита на подзадачи, которые в свою очередь выполняются при помощи разработанных методов.

Блок-схема метода ввода массива представлена на рисунке 1.



Рисунок 1 – Блок-схема ввода двумерного массива

Из данной схемы видно, что массив заполняется построчно при помощи ввода с клавиатуры. Ввод данных с клавиатуры позволяет пользователю произвести требуемые расчеты с любым значениями.

На рисунке 2 представлена блок-схема алгоритма для вывода элементов массива.



Рисунок 2 – Блок-схема алгоритма вывода элементов двумерного массива

Метод вывода элементов двумерного массива позволяет выводить любой массив, что делают программу более компактной и не требует повторного написания однотипных алгоритмов. Данная схема является актуальной и при выводе элементов массива в текстовый файл.

Блок-схема алгоритма составления двумерного массива согласно указанной в задании формуле приведена на рисунке 3.

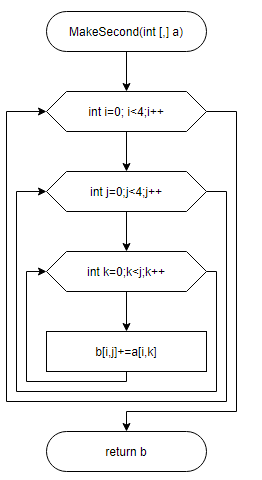


Рисунок 3 – Блок-схема алгоритма составления двумерного массива

Данный метод используется для заполнения второго массива по формуле, данной в задании. Происходит цикл по всем элементам массива, и они заполняются соответствующими значениями произведения.

Поиск суммы элементов четных строк и нечетных столбцов, реализовано при помощи алгоритма, представленного на рисунке 4.

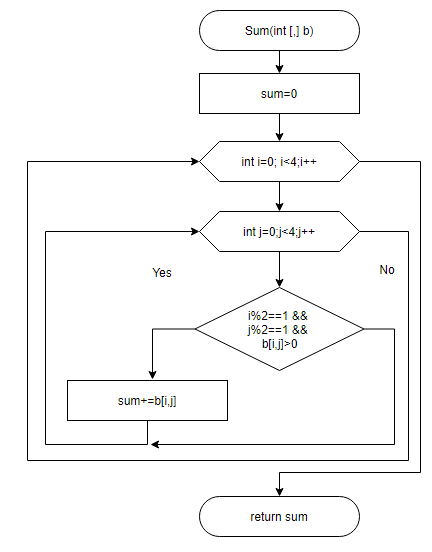


Рисунок 4 – Блок-схема расчета суммы

В данном методе происходит расчет суммы элементов четных строк и нечетных столбцов массива.

Блок-схема алгоритма определения максимального элемента в нечетных столбцах массива показана на рисунке 5.

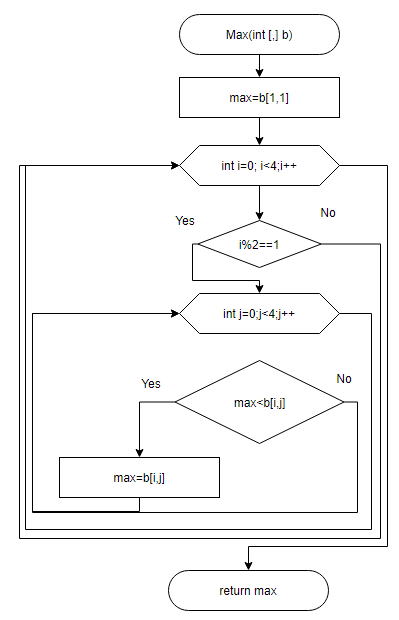


Рисунок 4 – Блок-схема поиска максимального элемента

В данном методе происходит поиск максимального элемента среди элементов нечетных столбцов массива.

На рисунке 6 приведена блок-схема метода расчета математической формулы.

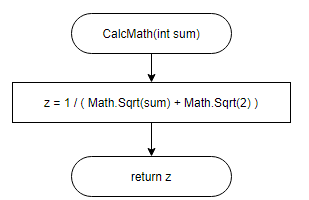


Рисунок 6 – Блок-схема расчета математической формулы

Данный метод реализует расчет математической формулы при помощи методов класса Math.

Блок-схема запуска программы приведена на рисунке 7.



Рисунок 7 – Блок-схема запуска программы

На данной схеме показан метод main, который является точкой запуска программы, отсюда запускаются методы, алгоритмы которых показаны выше.

# 3. Листинг программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace Kurs

{

class Program

{

// ввод первого массива

public static int[,] Input()

{

int[,] a = new int[4, 4];

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] = Convert.ToInt32(Console.ReadLine());

}

return a;

}

//создание второго массива

public static int[,] MakeSecond(int [,]a)

{

int[,] b = new int[4, 4];

for (int i = 0; i < b.GetLength(0); i++)

for (int j = 0; j < b.GetLength(1); j++)

b[i, j] = 0;

//заполнение второго массива значениями

for (int i = 0; i < b.GetLength(0); i++)

for (int j = 0; j < b.GetLength(1); j++)

for (int k = 0; k <= j; k++)

b[i, j] += a[i, k];

return b;

}

//высчитывание значения суммы положительных элементов четных строк и четных столбцов

public static int Sum(int [,]b)

{

int sum = 0;

for (int i = 0; i < b.GetLength(0); i ++)

for (int j = 0; j < b.GetLength(1); j++)

//если четные и положительные

if(i%2==1 && j%2==1 && b[i,j]>0)

sum += b[i, j];

return sum;

}

//поиск максимального элемента четных строк

public static int Max(int [,]b)

{

int max = b[1,1];

//пробегаем четные строки

for (int i = 0; i < b.GetLength(0); i++)

if(i%2==1)

//каждый столбец четной строки

for (int j = 0; j < b.GetLength(1); j ++)

if (max < b[i, j])

max = b[i, j];

return max;

}

//вывод массива на экран

public static void OutPut(int [,]a)

{

for (int i = 0; i < a.GetLength(0); i++)

{

for (int j = 0; j < a.GetLength(1); j++)

Console.Write(a[i, j] + "\t");

Console.WriteLine();

}

}

//сохранение массива в файл

public static void OutPut(int[,] a,StreamWriter wr)

{

for (int i = 0; i < a.GetLength(0); i++)

{

for (int j = 0; j < a.GetLength(1); j++)

wr.Write(a[i, j] + "\t");

wr.WriteLine();

}

}

public static double CalcMath(int sum)

{

try

{

return 1 / (Math.Sqrt(sum) + Math.Sqrt(2));

}

catch (Exception e) {

Console.WriteLine("Ошибка при расчете математического выражения");

return -1;

}

}

//точка входа в програму

static void Main(string[] args)

{

Console.WriteLine("Введите элементы первого массива");

//заполняем первый массив

int[,] a = Input();

//делаем второй массив

int[,] b = MakeSecond(a);

int sum = Sum(b);

int max = Max(b);

double z = CalcMath(sum);

Console.WriteLine("Элементы первого массива");

OutPut(a);

Console.WriteLine("Элементы второго массива");

OutPut(b);

Console.WriteLine("Сумма положительных элементов четных строк и четных столбцов равна "+sum);

Console.WriteLine("Максимальный элемент четных строкравен "+max);

Console.WriteLine("Результат математического выражения {0:f2}",z);

StreamWriter wr = new StreamWriter("Answer.txt");

wr.WriteLine("Элементы первого массива");

OutPut(a, wr);

wr.WriteLine("Элементы второго массива");

OutPut(b, wr);

wr.WriteLine("Сумма положительных элементов четных строк и четных столбцов равна " + sum);

wr.WriteLine("Максимальный элемент четных строкравен " + max);

wr.WriteLine("Результат математического выражения {0:f2}", z);

wr.Close();

Console.WriteLine("Нажмите любую клавишу для выхода...");

Console.ReadKey();

}

}

}

# 4. Скриншот работы программы

На рисунке 8 представлен скриншот работы программы.

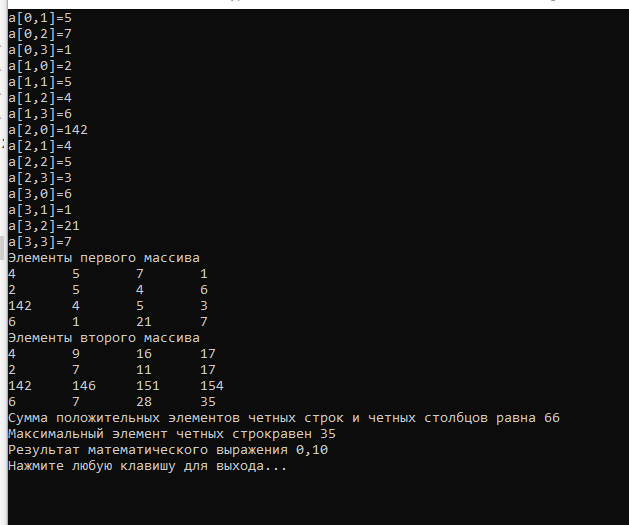


Рисунок 8 – Скриншот работы программы

На рисунке 9 приведен текстовый файл, который сформировала программа.

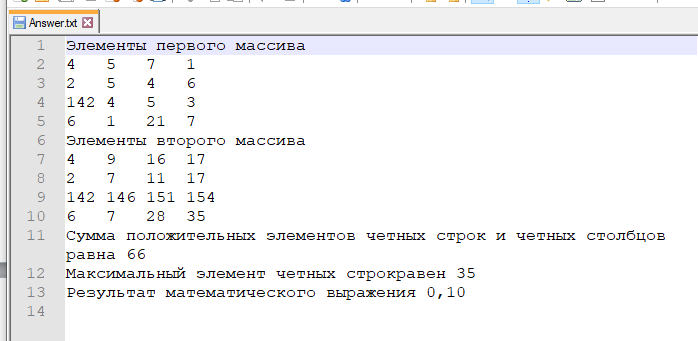


Рисунок 9 – Текстовый файл, сформированный программой

Как видно из рисунков на экран и в файл выводятся сведения, которые были указаны в задании.

# Заключение

В результате выполнения данной курсовой работе была разработана программа с использованием языка программирования C#.

Выполнение курсовой работы было разделено на два этапа:

– теоретический анализ поставленной задачи, включающий в себя постановку задачи, определение методов решения задачи, выделение основных действий и алгоритмов их реализации, определение математического аппарата, необходимого при решении задачи;

– программная реализация задачи средствами C#, включающая в себя реализацию основных действий: работы с массивами, текстовыми файлами и математическими формулами.

В ходе выполнения курсовой работы были изучены такие пространства имен как console, array, math.

Исходя из приведенных скриншотов программы, можно сделать вывод о том, что поставленная задача была решена верно, и в полном объеме.

# Список использованной литературы

1. Зиборов, В.В. Visual C# 2010 на примерах / В.В. Зиборов. – СПб.: БХВ – Санкт-Петербург, 2011. – 432 с.
2. Пугачев С., Шериев А., Кичинский К. Разработка приложений для Windows 8 на языке C#; БХВ-Петербург, 2013. - 416 c.
3. Мартин Р. С., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C#; Символ-Плюс, 2011. - 768 c.
4. Стенли, Б. Язык программирования C# / Б. Стенли. – М.: Вильямс, 2014. – 1120 с.
5. Шилдт, Г. Полный справочник по C# / Г. Шилдт. – М.: Вильямс, 2004. – 744 с.