СОДЕРЖАНИЕ

1. Лабораторная работа №1…………...………………………………………….2

2. Лабораторная работа №2………………………………………………………5

3. Лабораторная работа №3………………………………………………………9

4. Лабораторная работа №4……………………………………………………..17

5. Лабораторная работа №5……………………………………………………..22

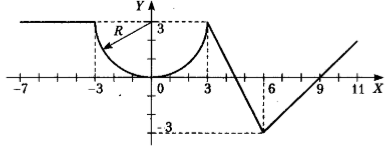
6. Лабораторная работа №6……………………………………………………..25

**Лабораторная работа №2**

**Тема: «Разветвляющиеся вычислительные процессы »**

**Цель работы:** Изучение операторов ветвления на языке C#.

**Задание:**  Часть 1. Написать программу, вычисляющую значение кусочной функции, заданной на графике (Рисунок 1).

  
Рисунок 1 – График кусочной функции

Часть 2. Написать программу, которая определяет, принадлежит ли пара (х,у) закрашенной области (Рисунок 2).

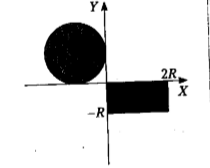


Рисунок 2 – Изображение закрашенной области

**Краткая теория**

Для организации условного ветвления язык C# унаследовал от С и С++ конструкцию if...else. Синтаксис:

if (условие)

оператор (операторы)

else

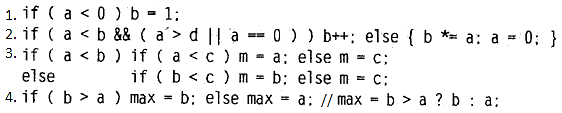
оператор (операторы)

Если по каждому из условий нужно выполнить более одного оператора, эти операторы должны быть объединены в блок с помощью фигурных скобок {...}.

Стоит обратить внимание, что в отличие от языков С и С++, в C# условный оператор if может работать только с булевскими выражениями, но не с произвольными значениями вроде -1 и 0.

В операторе if могут применяться сложные выражения, и он может содержать операторы else, обеспечивая выполнение более сложных проверок. Синтаксис похож на применяемый в аналогичных ситуациях в языках С (С++) и Java. При построении сложных выражений в C# используется вполне ожидаемый набор логических операторов.

Примеры использования условного оператора if:



Вторым оператором выбора в C# является оператор switch, который обеспечивает многонаправленное ветвление программы. Следовательно, этот оператор позволяет сделать выбор среди нескольких альтернативных вариантов дальнейшего выполнения программы. Несмотря на то, что многонаправленная проверка может быть организована с помощью последовательного ряда вложенных операторов if, во многих случаях более эффективным оказывается применение оператора switch. Этот оператор действует следующим образом. Значение выражения последовательно сравнивается с константами выбора из заданного списка. Как только будет обнаружено совпадение с одним из условий выбора, выполняется последовательность операторов, связанных с этим условием. Ниже приведена общая форма оператора switch:

switch(выражение) {

case константа1:

последовательность операторов

break;

case константа2:

последовательность операторов

break;

case константаЗ:

последовательность операторов

break;

default: //Введено что-то другое

последовательность операторов

break;

}

Оператор switch...case в C# он немного безопаснее, чем его эквивалент С++. В частности, он запрещает "сквозные" условия почти во всех случаях. Это значит, что если часть case вызывается в начале блока, то фрагменты кода за последующими частями case не могут быть выполнены, если только не используется явно оператор goto для перехода к ним. Компилятор обеспечивает это ограничение за счет того, что требует, чтобы за каждой частью case следовал оператор break, в противном случае он выдает ошибку.

Важно отметить, что заданное выражение в операторе switch должно быть целочисленного типа (char, byte, short или int), перечислимого или же строкового. А выражения других типов, например с плавающей точкой, в операторе switch не допускаются. Зачастую выражение, управляющее оператором switch, просто сводится к одной переменной. Кроме того, константы выбора должны иметь тип, совместимый с типом выражения. В одном операторе switch не допускается наличие двух одинаковых по значению констант выбора.

**Практическая часть**

Код программы:

using System;

namespace The\_saga

{

class Program

{

public static void Main(string[] args)

{

double x, y=0,R;

Console.WriteLine("Первая часть задания");

do

{

Console.Write("Введите x=");

x = Convert.ToDouble(Console.ReadLine());

if (x < -3)

y = 3;

else if (-3 <= x && x <= 3)

y = 3 - Math.Sqrt(9 - Math.Pow((x ), 2));

else if (3 < x && x < 6)

y = -x+1;

else if (6 <= x)

y = x+1;

Console.WriteLine("y=" +y);

Console.WriteLine("Приступить ко второй части задания? (Да - любой символ, Нет - нет)");

}

while (Console.ReadLine() == "нет");

Console.WriteLine("Вторая часть задания");

do

{

Console.Write("Введите R=");

R = Convert.ToDouble(Console.ReadLine());

Console.Write("Введите x=");

x = Convert.ToDouble(Console.ReadLine());

Console.Write("Введите у=");

y = Convert.ToDouble(Console.ReadLine());

if ((y <= Math.Pow((x+R)+(y-R),2)||(y<=2\*R\*(-R))))

Console.WriteLine("Пара (х;у) принадлежит области");

else

Console.WriteLine("Пара (х;у) не принадлежит области");

Console.WriteLine("Завершить задание? (Да - любой символ, Нет - нет)");

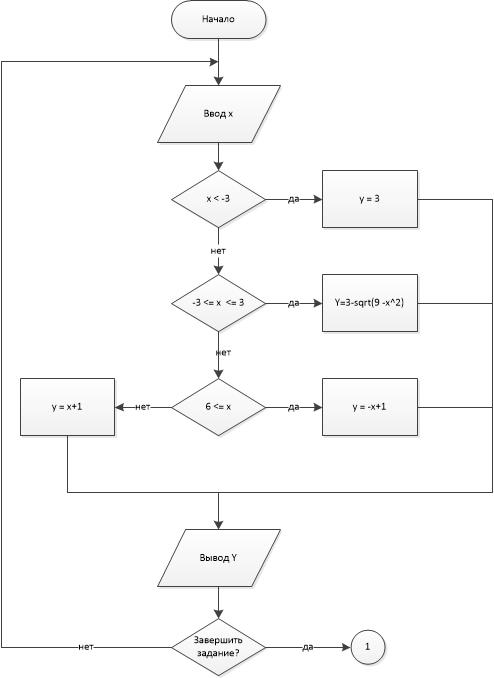
}

while (Console.ReadLine() == "нет");

}

}

}



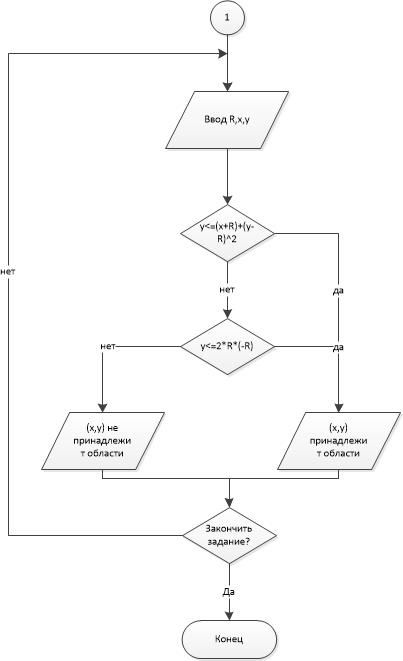


Рисунок 3 – Блок-схема реализации программы

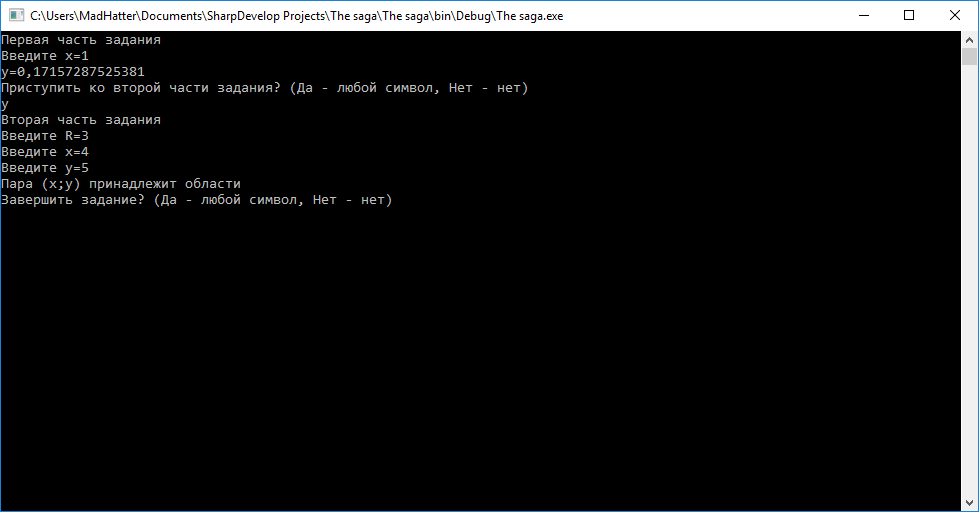


Рисунок 4 – Пример выполнения программы

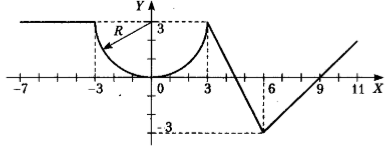
**Вывод:** Получены навыки написания программ с ветвлениями на языке C# в среде программирования Microsoft Visual Studio.

**Лабораторная работа №3**

**Тема: «Организация циклов»**

**Цель работы:** Изучения операторов циклов на языке C#.

**Задание:**  Вычислить и вывести на экран в виде таблицы значения функции, заданной графически на интервале от хтч до хт„ с шагом dx. Интервал и шаг задать таким образом, чтобы проверить все ветви программы. (Рисунок 1).

  
Рисунок 1 – График кусочной функции

Часть 2. Для десяти выстрелов, координаты которых задаются с клавиатуры, вывести текстовые сообщения о попадании в мишень (Рисунок 2).

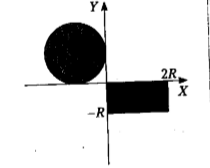


Рисунок 2 – Изображение закрашенной области

Часть 3. Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда Тейлора, на интервале от дггач до #кон с шагом dx с точностью е. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.(Рисунок 3)



Рисунок 3 – Функция, заданная с помощью ряда Тейлора

**Краткая теория**

В C# имеются четыре различных вида циклов (for, while, do...while и foreach), позволяющие выполнять блок кода повторно до тех пор, пока удовлетворяется определенное условие.

Цикл for в C# предоставляет механизм итерации, в котором определенное условие проверяется перед выполнением каждой итерации. Синтаксис этого оператора показан ниже:

for (инициализатор; условие; итератор)

оператор (операторы)

Здесь:

* инициализатор

это выражение, вычисляемое перед первым выполнением тела цикла (обычно инициализация локальной переменной в качестве счетчика цикла). Инициализация, как правило, представлена оператором присваивания, задающим первоначальное значение переменной, которая выполняет роль счетчика и управляет циклом;

* условие

это выражение, проверяемое перед каждой новой итерацией цикла (должно возвращать true, чтобы была выполнена следующая итерация);

* итератор

выражение, вычисляемое после каждой итерации (обычно приращение значения счетчика цикла).

Подобно for, while также является циклом с предварительной проверкой. Синтаксис его аналогичен, но циклы while включают только одно выражение:

while(условие)

оператор (операторы);

где оператор — это единственный оператор или же блок операторов, а условие означает конкретное условие управления циклом и может быть любым логическим выражением. В этом цикле оператор выполняется до тех пор, пока условие истинно. Как только условие становится ложным, управление программой передается строке кода, следующей непосредственно после цикла.

Как и в цикле for, в цикле while проверяется условное выражение, указываемое в самом начале цикла. Это означает, что код в теле цикла может вообще не выполняться, а также избавляет от необходимости выполнять отдельную проверку перед самим циклом.позволяет сделать выбор среди нескольких альтернативных вариантов дальнейшего выполнения программыЦикл do...while в C# — это версия while с постпроверкой условия. Это значит, что условие цикла проверяется после выполнения тела цикла. Следовательно, циклы do...while удобны в тех ситуациях, когда блок операторов должен быть выполнен как минимум однажды. Ниже приведена общая форма оператора цикла do-while:

do {

операторы;

} while (условие);

При наличии лишь одного оператора фигурные скобки в данной форме записи необязательны. Тем не менее они зачастую используются для того, чтобы сделать конструкцию do-while более удобочитаемой и не путать ее с конструкцией цикла while. Цикл do-while выполняется до тех пор, пока условное выражение истинно. В качестве примера использования цикла do-while можно привести следующую программу, расчитывающую факториал числа:

**Практическая часть**

Код программы:

using System;

namespace TA\_Lab\_3

{

class Program

{

public static void Main(string[] args)

{

double x = -7, y = -1, dx, R;

Console.WriteLine("Zadanie 1");

Console.WriteLine("Tablica znachenij funkcii:");

Console.WriteLine(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine(" | X | Y |");

Console.WriteLine(" |\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|");

for (dx = 1; x < 18; x = x + dx)

{

if (x < -3)

y = 3;

else if (-3 <= x && x <= 3)

y = 3 - Math.Sqrt(9 - Math.Pow((x), 2));

else if (3 < x && x < 6)

y = -2 \* x + 9;

else if (6 <= x)

y = x - 9;

Console.WriteLine(" |" + String.Format("{0,4:0.#}", x) + " |" + String.Format("{0,5:0.##}", y) + " |");

Console.WriteLine(" |\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|");

}

Console.WriteLine("Zadanie 2");

for (int i = 1; i <= 10; i++)

{

Console.Write("Vvedite R=");

R = Convert.ToDouble(Console.ReadLine());

Console.Write("Vvedite x=");

x = Convert.ToDouble(Console.ReadLine());

Console.Write("Vvedite у=");

y = Convert.ToDouble(Console.ReadLine());

if ((y <= Math.Pow((x + R) + (y - R), 2)) || (y <= 0 && y >= -R && x >= 0 && x <= 2 \* R))

Console.WriteLine("Popadanie");

else

Console.WriteLine("Mimo");

}

Console.WriteLine("Zadanie 3");

double xn, xk, gx, e = 1E-15;

Console.Write("X nachalnoe: ");

xn= Convert.ToDouble(Console.ReadLine());

Console.Write("X konechnoe: ");

xk = Convert.ToDouble(Console.ReadLine());

Console.Write("dX: ;");

gx = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("\n X Тейлор N Math");

Console.WriteLine(new string('\_', 55));

for (x = xn; x <= xk; x += gx)

if (Math.Abs(x) <= 1)

{

int n = 1, mul = -1, d = 1;

double sum = Math.PI / 2, x\_n = x;

double el = sum, oldel;

do

{

oldel = el;

el = x\_n / d;

sum += mul \* el;

x\_n \*= x \* x;

n++;

mul = -mul;

d += 2;

} while (Math.Abs(el - oldel) > e);

Console.WriteLine("{0,6:F3} {1,18:F15} {2,9} {3,18:F15}", x, sum, n, Math.PI / 2 - Math.Atan(x));

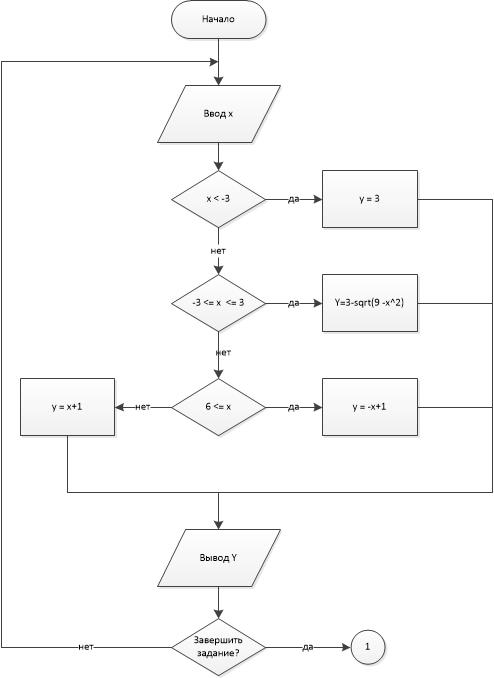
}

Console.ReadKey();

Console.ReadKey(true);

}

}

}

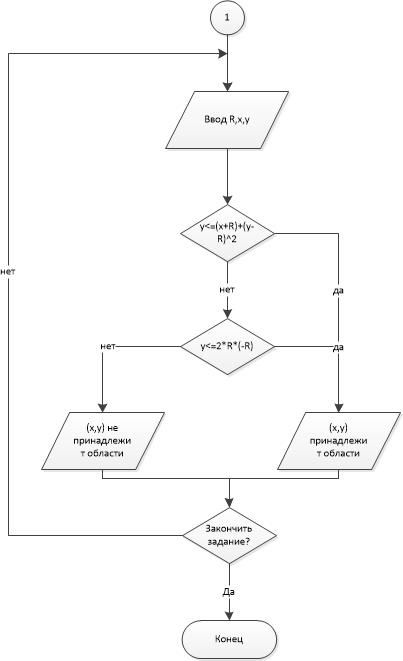


Рисунок 3 – Блок-схема реализации программы

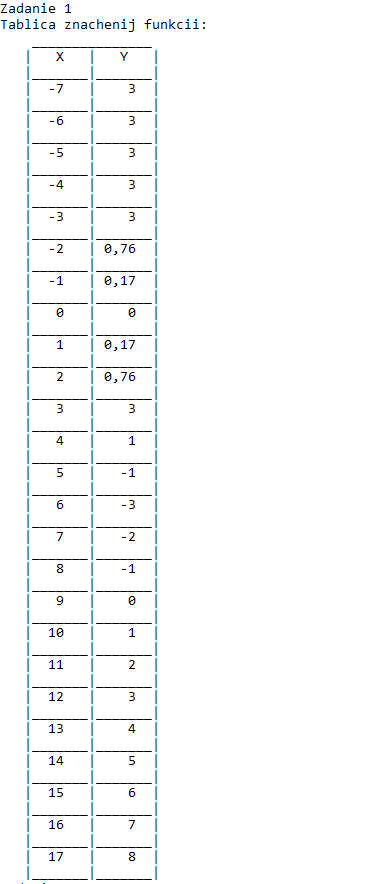


Рисунок 4 – Пример выполнения программы

**Вывод:** Получены навыки написания программ с организацией циклов на языке C# в среде программирования Microsoft Visual Studio.

**Лабораторная работа №4**

**Тема: «Одномерные массивы»**

**Цель работы:** Научиться работать с одномерными массивами на языке C#.

**Задание:**  Вариант 7. В одномерном массиве, состоящем из п целочисленных элементов, вычислить:

* номер максимального элемента массива;
* произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине — элементы, стоявшие в четных позициях.

**Краткая теория**

В C# массивы фактически являются объектами, а не только адресуемыми областями непрерывной памяти, как в C и C++. Array является абстрактным базовым типом всех типов массивов. Можно использовать свойства и другие члены класса, которые имеет Array. В примере используется свойство Length для получения длины массива. В следующем коде длина массива numbers, равная 5, присваивается переменной lengthOfNumbers:

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

int lengthOfNumbers = numbers.Length;

Класс Array позволяет использовать много других полезных методов и свойств для выполнения сортировки, поиска и копирования массивов.

Инициализация массива

Массив можно инициализировать при объявлении. В этом случае спецификация ранга не нужна, поскольку она уже предоставлена по числу элементов в списке инициализации. Примеры.

int[] array1 = new int[] { 1, 3, 5, 7, 9 };

Строковый массив можно инициализировать таким же образом. Ниже приведено объявление строкового массива, в котором каждый элемент инициализируется названием дня:

string[] weekDays = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };

При инициализации массива при объявлении можно использовать следующие сочетания клавиш:

int[] array2 = { 1, 3, 5, 7, 9 };

string[] weekDays2 = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };

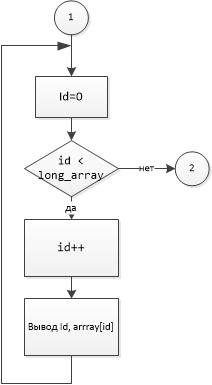
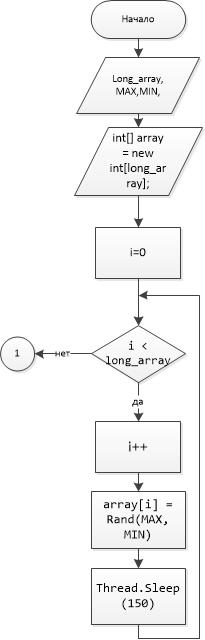
Можно объявить переменную массива без инициализации, но при присвоении массива этой переменной нужно использовать оператор new. Примеры.

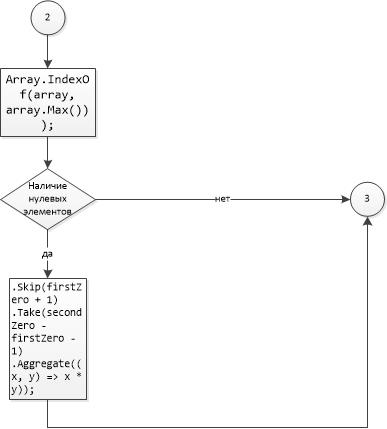
int[] array3;

array3 = new int[] { 1, 3, 5, 7, 9 }; // OK

//array3 = {1, 3, 5, 7, 9}; // Error

**Практическая часть**





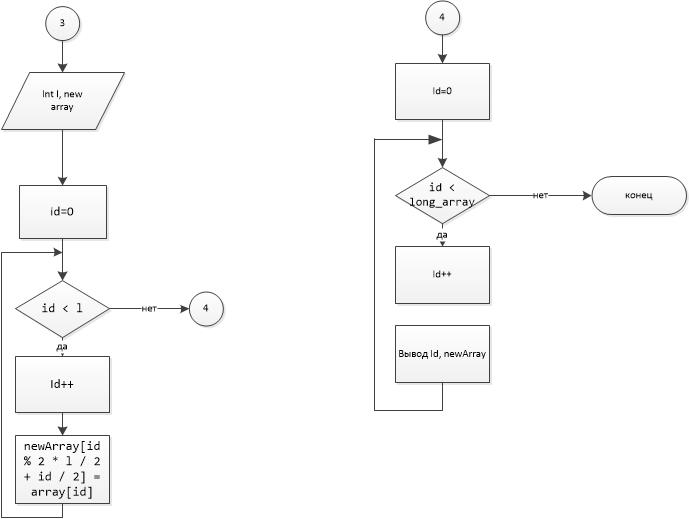


Рисунок 1 – Блок-схема реализации программы

Код программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

namespace Massive

{

class Program

{

static void Main(string[] args)

{

int long\_array = 11, MAX = 100, MIN = -100;

int[] array = new int[long\_array];

for (int i = 0; i < long\_array; i++)

{

array[i] = Rand(MAX, MIN);

Thread.Sleep(150);

}

Console.WriteLine(" Содержание массива");

Console.WriteLine(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine(" | ID |array[ID]|");

Console.WriteLine(" |\_\_\_\_\_|\_\_\_\_\_\_\_\_\_|");

for (int id = 0; id < long\_array; id++)

{

Console.WriteLine(" |" + String.Format("{0,5:0.#}", id) + "|" + String.Format("{0,9:0.##}", array[id]) + "|");

Console.WriteLine(" |\_\_\_\_\_|\_\_\_\_\_\_\_\_\_|");

}

Console.WriteLine("\n\tНомер максимального элемента массива: {0}",

Array.IndexOf(array, array.Max()));

try

{

int firstZero = Array.IndexOf(array, 0);

int secondZero = Array.IndexOf(array, 0, firstZero + 1);

Console.WriteLine("произведение: {0}", array

.Skip(firstZero + 1)

.Take(secondZero - firstZero - 1)

.Aggregate((x, y) => x \* y));

}

catch (Exception)

{

Console.WriteLine("\n\tНет нулевых значений");

}

int l = array.Length;

int[] newArray = new int[l];

for (int id = 0; id < l; id++)

{

newArray[id % 2 \* l / 2 + id / 2] = array[id];

}

Console.WriteLine("\n\tИзмененный массив");

Console.WriteLine(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine(" | ID |array[ID]|");

Console.WriteLine(" |\_\_\_\_\_|\_\_\_\_\_\_\_\_\_|");

for (int id = 0; id < long\_array; id++)

{

Console.WriteLine(" |" + String.Format("{0,5:0.#}", id) + "|" + String.Format("{0,9:0.##}",newArray[id]) + "|");

Console.WriteLine(" |\_\_\_\_\_|\_\_\_\_\_\_\_\_\_|");

}

Console.ReadKey(true);

}

public static int Rand(int max, int min)

{

Random rand = new Random();

return rand.Next(min, max);

}

}

}

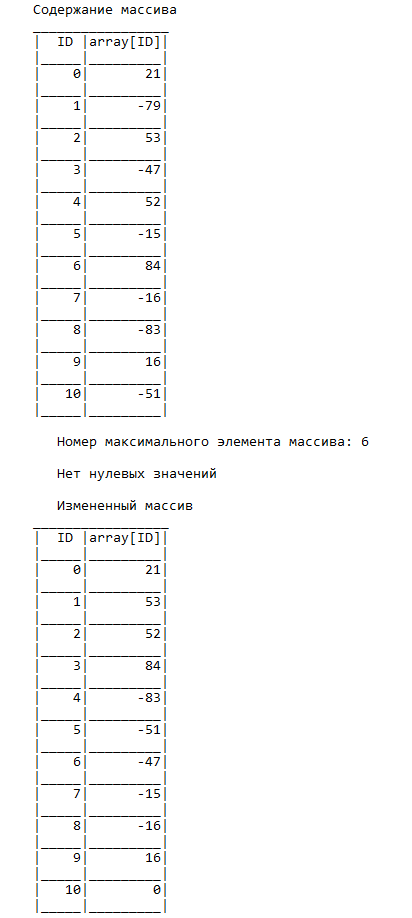


Рисунок 4 – Пример выполнения программы

**Вывод:** Изучена работа с одномерными массивами на языке C# в среде программирования Microsoft Visual Studio.

**Лабораторная работа №5**

**Тема: «Двумерные массивы»**

**Цель работы:** Научиться работать с двумерными массивами на языке C#.

**Задание:**  Вариант 7. Для заданной матрицы размером 8x8 найти такие к, при которых k-я строка матрицы совпадает с k-м столбцом. Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.

**Краткая теория**

Двумерные массивы

Простейшей формой многомерного массива является двумерный массив. Местоположение любого элемента в двумерном массиве обозначается двумя индексами. Такой массив можно представить в виде таблицы, на строки которой указывает один индекс, а на столбцы — другой.

// Объявляем двумерный массив

int[,] myArr = new int[a,b]; //a,b – высота и ширина массива

**Практическая часть**

Код программы:

using System;

using System.IO;

using System.Collections.Generic;

using System.Globalization;

namespace TA5

{

class Program

{

static void Main(string[] args)

{

List<int> matchKNumbers = new List<int>();

Console.WriteLine("Квадратная матрица");

int[,] matrix = new int[8, 8];

Random rand = new Random();

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

matrix[i, j] = rand.Next(-8, 8);

}

}

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

Console.Write("\_\_\_\_\_\_");

}

Console.WriteLine();

for (int j = 0; j < 8; j++)

{

Console.Write("|" + String.Format("{0,5:0.#}", matrix[i, j]));

}

Console.WriteLine("|");

for (int j = 0; j < 8; j++)

{

Console.Write("|\_\_\_\_\_");

}

Console.WriteLine("|");

}

Console.WriteLine("\n");

for (int i = 0; i < 8; i++)

{

bool doesMatch = true;

for (int j = 0; j < 8; j++)

if (matrix[i, j] != matrix[j, i])

doesMatch = false;

if (doesMatch)

matchKNumbers.Add(i);

}

foreach (int number in matchKNumbers)

Console.WriteLine("k = {0}", number);

for (int i = 0; i < 8; i++)

{

bool hasNegativeItem = false;

for (int j = 0; j < 8; j++)

if (matrix[i, j] < 0.0)

hasNegativeItem = true;

if (hasNegativeItem)

{

double sum = 0.0;

for (int j = 0; j < 8; j++)

sum += matrix[i, j];

Console.WriteLine("Сумма строки {0} равна {1}", i, sum);

Console.ReadKey(true);

}

}

}

}

}

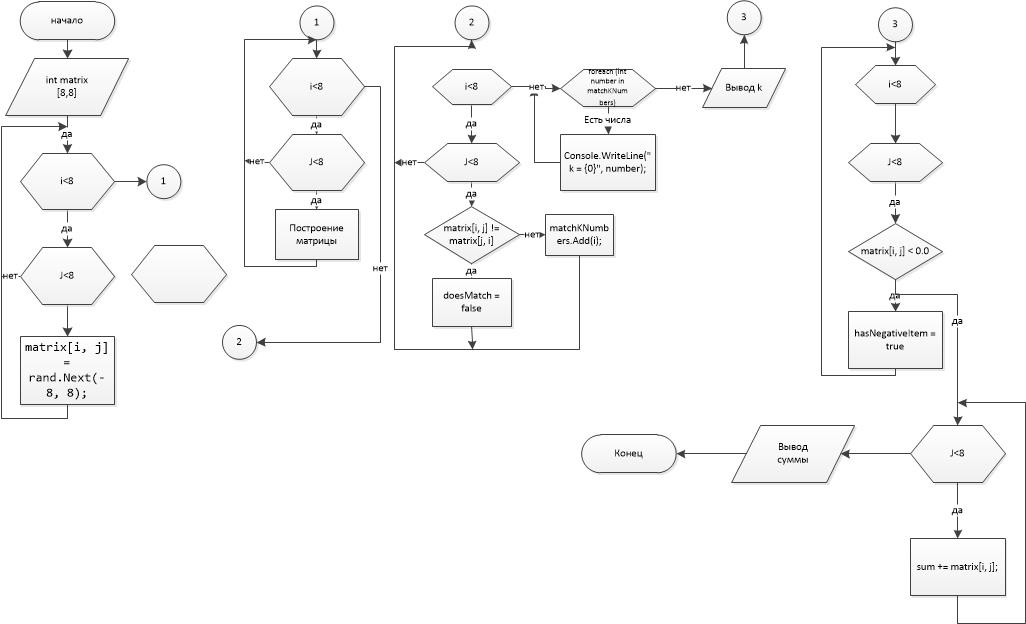
****

Рисунок 1 – Блок-схема реализации программы

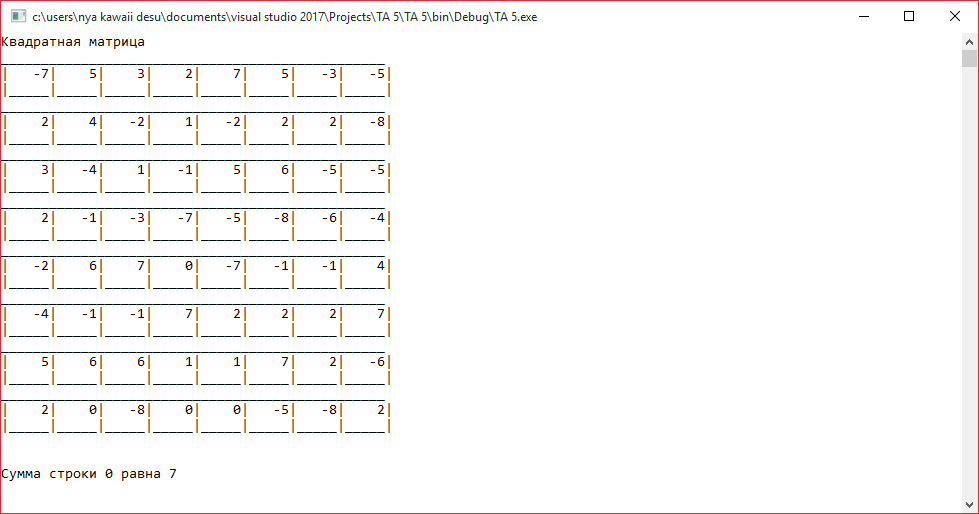


Рисунок 4 – Пример выполнения программы

**Вывод:** Изучена работа с двумерными массивами на языке C# в среде программирования Microsoft Visual Studio.

**Лабораторная работа №6**

**Тема: «Строки»**

**Цель работы:** Изучение операторов строк на языке C#.

**Задание:**  Написать программу, которая считывает английский текст из файла и выводит на экран слова текста, начинающиеся и оканчивающиеся на гласные буквы.

**Краткая теория**

C# — это язык программирования, предназначенный для разработки самых разнообразных приложений, предназначенных для выполнения в среде .NET Framework.

В данном случае using подключает различные области имен:

using System- область имен с системными базовыми классами.

using System.Collections.Generic- дженерики и коллекции.

using System.Text- для работы с текстом.

System.Linq- содержит классы и интерфейсы, которые поддерживают LINQ.

System.Threading.Tasks- пространство имен предоставляет типы, которые упрощают задачу написания параллельного и асинхронного кода.

name space - пространство имен.

static void Main() статик говорит, что майн - локальная функция для .cpp файла и вне файла к ней получить доступ просто так не получится. В скобках пишутся аргументы функции.

Double- Обозначает простой тип, используемый для хранения 64-разрядных значений с плавающей запятой.

Console.Write - Записывает текстовое представление заданного значения или значений в стандартный выходной поток.

Console.WriteLine- Записывает текущий признак конца строки в стандартный выходной поток.

Console.ReadLine- Считывает следующую строку символов из стандартного входного потока.

Класс Convert- Преобразует значение одного базового типа данных к другому базовому типу данных.

Convert.ToDouble- Преобразует заданное строковое представление числа в эквивалентное число с плавающей запятой двойной точности.

Конструкция if/else проверяет истинность некоторого условия и в зависимости от результатов проверки выполняет определенный код. После ключевого слова if ставится условие. И если это условие выполняется, то срабатывает код, который помещен далее в блоке if после фигурных скобок.

Класс Math- Предоставляет константы и статические методы для тригонометрических, логарифмических и иных общих математических функций.

Math.PI- Представляет отношение длины окружности к ее диаметру, определяемое константой π.

# Math.Sqrt- Возвращает квадратный корень из указанного числа.

# Math.Sin- Возвращает синус указанного угла.

Math.Pow- Возвращает указанное число, возведенное в указанную степень.

Console.ReadKey- Получает следующий нажатый пользователем символ или функциональную клавишу.

Значение **true**, чтобы не отображать нажатую клавишу  в окне консоли. В противном случае — значение **false**.

**Практическая часть**

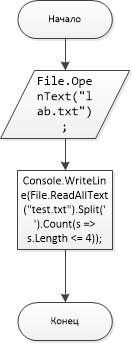


Рисунок 1 – Блок-схема реализующая алгоритм выполнения вычислений

Код программы:

namespace TA\_6

{

public class Program

{

private static void Main(string[] args)

{

StreamReader f = File.OpenText("test.txt");

string text = f.ReadToEnd();

Console.WriteLine("Исходный текст: ");

Console.WriteLine(text);

Console.WriteLine("\n");

Console.WriteLine(File.ReadAllText("test.txt").Split(' ').Count(s => s.Length <= 4));

Console.ReadKey();

}

}

}

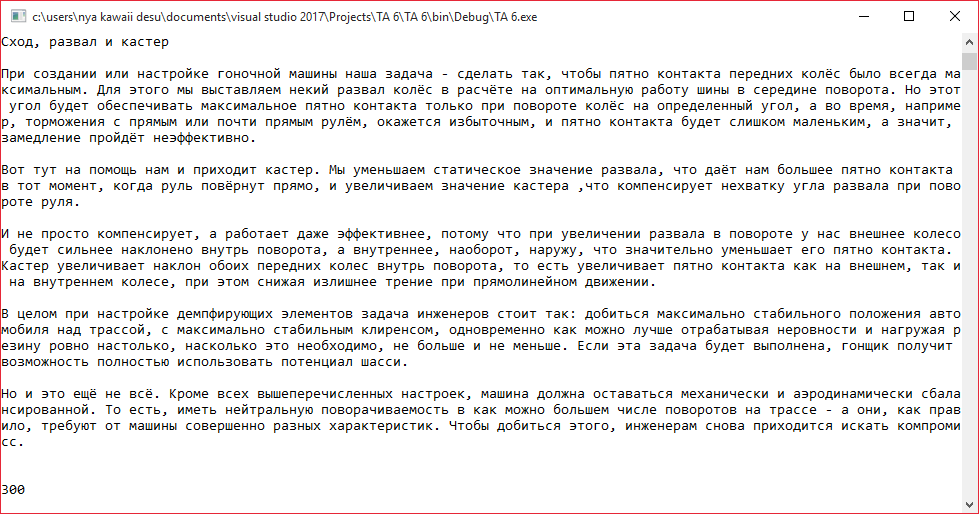


Рисунок 2 – Пример выполнения программы

**Вывод:** Получены навыки написания программ с помощью операторов ветвления на языке C# в среде программирования Microsoft Visual Studio.