СОДЕРЖАНИЕ

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

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

3. Лабораторная работа №3……………………………………………………..10

4. Лабораторная работа №4……………………………………………………..16

5. Лабораторная работа №5……………………………………………………..21

6. Лабораторная работа №6……………………………………………………..24

**Лабораторная работа №1**

**Тема: «Линейные программы»**

**Цель работы: «Изучить основы линейных программ на языке C#»**

Вариант 8

**Теоретическая часть:**

Pow - математическая функция в С# которая возвращает значение, то есть возводит число x в степень a.

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

Ключевое слово double обозначает простой тип, используемый для хранения 64-разрядных значений с плавающей запятой. В следующей таблице представлен приблизительный диапазон и точность для типа double.

**Практическая часть:**

z1= 1-1/4\*(sin(2\*a))^2+cos(2\*a)

z2= (cos(a))^2+(cos(a))^4

Блок-схема:



Код программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

double z1, z2, a;

Console.Write("Введите а=");

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

z1 = 1 - (1.0/4) \* Math.Pow(Math.Sin(2\*a), 2) + Math.Cos(2 \* a);

z2 = Math.Pow(Math.Cos(a),2);

z2 = z2 + z2 \* z2;

Console.WriteLine("z1=" +z1);

Console.WriteLine("z2=" +z2);

Console.ReadKey(true);

}

}

}

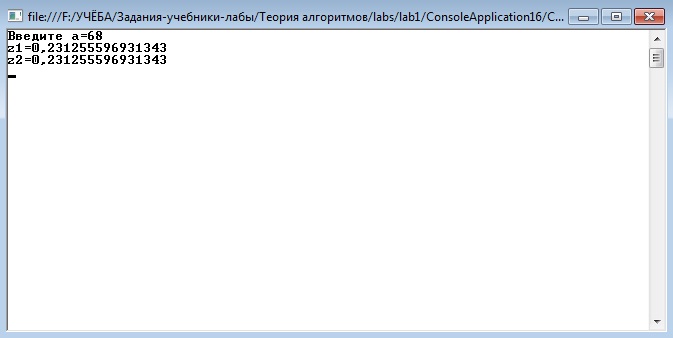


Рисунок 1 – Выполнение программы

**Вывод:** в процессе выполнения лабораторной работы была создана программа, вычисляющая два уравнения.

**Лабораторная работа №2**

**Тема: «Разветвляющиеся вычислительные процессы»**

**Цель работы:** «Изучение операторов ветвлений на языке C#»

**Теоретическая часть**

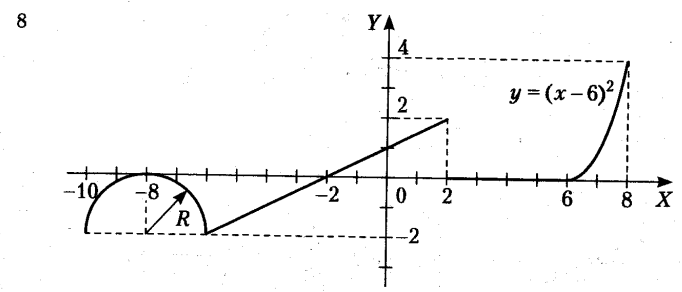
Оператор if обеспечивает передачу управления на одну из двух ветвей вы-числений. Сначала вычисляется логическое выражение. Если оно имеет значение true, выполняется первый оператор, иначе- второй. После этого управление передается на оператор, следующий за условным. В C# условный оператор if может работать 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.

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. Оператор switch...case 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**

Вариант 8

Задание 1. Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



Задание 2. Написать программу, которая определяет, попадает ли точка с заданными координатами в область, закрашенную серым цветом.

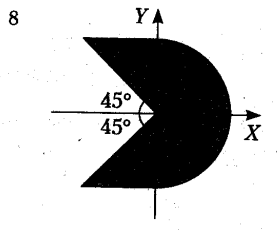
****





Рисунок 1 – Блок-схема реализации программы

Код программы:

namespace Lab\_2\_Kur

{

class Program

{

static void Main(string[] args)

{

double x, y = 0;

// Задание 1

Console.WriteLine("Задание 1");

Console.Write("x=");

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

if (-10 <= x && x < -6)

y = -Math.Sqrt(4 - Math.Pow(x + 8, 2)) - 2;

else if (-6 <= x && x <= 2)

y = 0.5 \* x + 1;

else if (2 < x && x < 6)

y = 0;

else if (6 <= x)

y = Math.Pow(x - 6, 2);

else

Console.WriteLine("Неопределена");

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

Console.ReadKey(true);

// Задание 2

Double R;

Console.WriteLine("\nЗадание 2");Console.Write("R=");

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

Console.Write("x=");

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

Console.Write("y=");

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

if ((x >= 0 && x <= R && x \* x + y \* y <= R \* R) || (x >= -R && x <= 0 && ((y >= -x && y <= R)||(y <= x && y >= -R))))

Console.WriteLine("Пара (x,y) принадлежит заданной области");

else Console.WriteLine("Пара (x,y) не принадлежит заданной области");

Console.ReadKey(true);

}

}

}

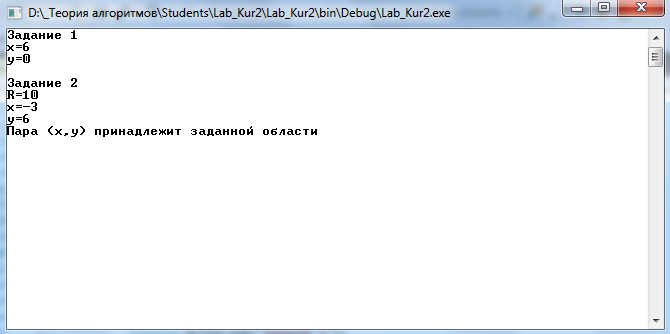


Рисунок 2 – Пример выполнения программы

**Вывод:** были изучены операторы ветвлений на языке C# и написаны программы вычисляющие значение функции и определяющие нахождение точки в графике.

**Лабораторная работа №3**

**Тема: «Организация циклов»**

**Цель работы:** Ознакомиться с конструкциями циклов на языке C#.

**Задание:**  Вариант 8.

Часть 1. Написать программу, выводящую значения на некотором интервале с заданным шагом, кусочной функции, посчитанной в предыдущей работе.

Часть 2. Организовать цикл для второй части задания предыдущего задания для десяти итераций программы.

Часть 3. Вычислить и вывести на экран в виде таблицы значения функции заданной с помощью ряда Тейлора, на некотором интервале с заданной точностью.

**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

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).

Эти три основные части оператора цикла for должны быть разделены точкой с запятой. Выполнение цикла for будет продолжаться до тех пор, пока проверка условия дает истинный результат. Как только эта проверка даст ложный результат, цикл завершится, а выполнение программы будет продолжено с оператора, следующего после цикла for.

Цикл while

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.

Цикл do. . . while

Цикл 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.

**Практическая часть**

Код программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_Kur3

{

class Program

{

static void Main(string[] args)

{

Console.Write("Выборка заданий: \n\tЧасть 1 - 1 \n\tЧасть 2 - 2 \n\tЧасть 3 - 3\nВыберете задание: ");

switch (Console.ReadLine())

{

case "1":

Console.WriteLine("\nЧасть-1");

double x = -10, y = 8, dx;

Console.WriteLine("Таблица значений кусочной функции");

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

Console.WriteLine(" | x | y |");

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

for (dx = 1; x < 9; x = x + dx)

{

if (-10 <= x && x < -6)

y = Math.Sqrt(4 - Math.Pow(x + 8, 2)) - 2;

else if (-6 <= x && x <= 2)

y = 0.5 \* x + 1;

else if (2 < x && x < 6)

y = 0;

else if (6 <= x)

y = Math.Pow(x - 6, 2);

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

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

}

break;

case "2":

Console.WriteLine("\nЧасть-2");

Double R;

Console.WriteLine("Попадания в мишень");

for (int i = 0; i < 10; i++)

{

while (true)

{

try

{

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());

}

catch (FormatException) { Console.WriteLine("\nОшибка формата данных данных\n"); continue; }

break;

}

if ((x >= 0 && x <= R && x \* x + y \* y <= R \* R) || (x >= -R && x <= 0 && ((y >= -x && y <= R)||(y <= x && y >= -R))))

Console.WriteLine("Есть попадание!");

else

Console.WriteLine("Мимо!");

}

break;

case "3":

Console.WriteLine("\nЧасть-3");

Double eps;

Console.WriteLine("Таблица значений функции");

y = 0;

eps = 0.0001;

Console.WriteLine("");

Console.WriteLine(String.Format("{0,5:0.#}", "X") + String.Format("{0,10}", "Y") + String.Format("{0,50}", "Кол-во просуммированных членов ряда"));

Console.WriteLine("");

for (x = 2; x < 20; x += 1)

{

int n = 0;

double b = 0;

do

{

y = Math.Pow((-1), n + 1) / ((2 \* n + 1) \* Math.Pow(x, 2 \* n + 1));

b = y + b;

n += 1;

} while (Math.Abs(y) > eps);

b = Math.PI / 2 + b;

Console.WriteLine(String.Format("{0,5:0.#}", x) + String.Format("{0,20:0.###############}", b) + String.Format("{0,10}", n));

}

Console.WriteLine("\nВыполнение окончено.");

Console.ReadKey(true);

break;

}

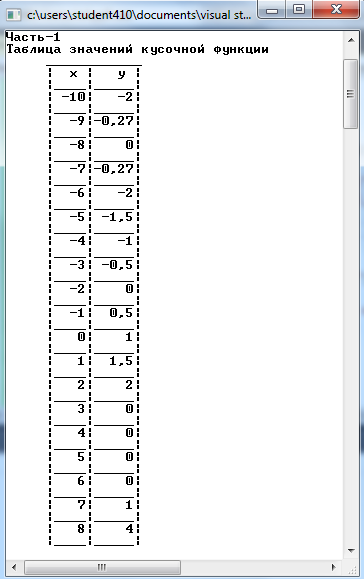
}

}

}



Рисунок 3 – Блок-схема реализации программы



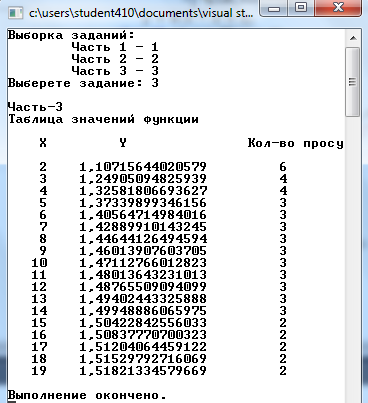


Рисунок 4 – Пример выполнения программы

**Вывод:** Изучена организация 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 8. В одномерном массиве, состоящем из вещественных элементов, вычислить: номер минимального элемента массива, сумму элементов массива, расположенных между первым и вторым отрицательными элементами. Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает единицу, а потом – все остальные.

**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;

namespace ConsoleApp6

{

class Program

{

static void Main(string[] args)

{

try

{

// Номер минимального элемента массива

Console.WriteLine("введите количество элементов массива");

Console.Write("N = ");

int n = int.Parse(Console.ReadLine());

float[] array = new float[n];

int min = 0;

int a = -1, b = -1;

Console.WriteLine("введите элементы массива");

for (int i = 0; i < array.Length; i++)

{

Console.Write("array[{0}] = ", i);

array[i] = float.Parse(Console.ReadLine());

if (array[min] > array[i])

min = i;

if (array[i] < 0)

if (a == -1)

a = i;

else if (b == -1)

b = i;

}

// Сумму элементов массива, расположенных между первым и вторым отрицательными элементами

Console.WriteLine("Номер минимального элемента = {0}", min);

if (a == -1 || b == -1)

Console.WriteLine("В массиве нет хотя бы двух отрицательных чисел: сумму не посчитать!");

else

{

float sum = 0;

for (int i = a + 1; i < b; i++)

sum += array[i];

Console.WriteLine("Сумма между двумя отрицательными числами = {0}", sum);

}

// Преобразовать массив таким образом, чтобы сначала располагались все элементы,

// модуль которых не превышает 1, а потом все остальные

for (int i = 0, j = 0; i < array.Length; i++)

if (Math.Abs(array[i]) <= 1)

{

float temp = array[i];

array[i] = array[j];

array[j] = temp;

j++;

}

Console.WriteLine("Результат преобразования массива:");

for (int i = 0; i < array.Length; i++)

Console.Write("{0} ", array[i]);

Console.WriteLine();

Console.ReadLine();

}

catch

{

Console.WriteLine("Ошибка");

Console.WriteLine("Приложение будет закрыто после нажатия клавиши Enter");

Console.ReadLine();

}

}

}

}

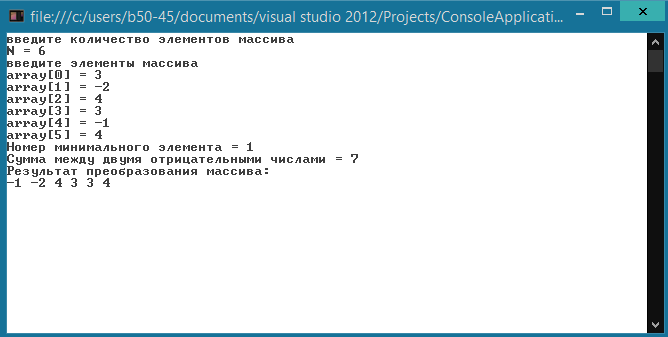


Рисунок 2 – Пример выполнения программы

**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 8. Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик. Найти сумму элементов в тех столбцах, 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.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp6

{

class Program

{

static void Main(string[] args)

{

Random rand = new Random();

Console.WriteLine("Введите число");

int n = Convert.ToInt32(Console.ReadLine());

int[,] c = new int[n, n];

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

{

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

{

c[i, j] = rand.Next(-100, 100);

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

}

Console.WriteLine();

}

int[] index = new int[n];

int[] num = new int[n];

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

{

int temp = 0;

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

{

if (c[i, j] < 0) temp += Math.Abs(c[i, j]);

}

num[i] = temp;

index[i] = i;

}

Array.Sort(num, index);

int[,] d = new int[n, n];

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

{

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

{

d[i, j] = c[index[i], j];

}

}

Console.WriteLine();

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

{

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

{

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

}

Console.WriteLine();

}

Console.ReadKey();

}

}

}



Рисунок 1 – Блок-схема реализации программы

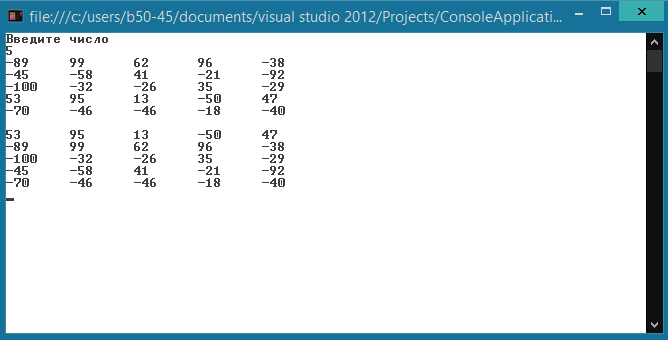


Рисунок 4 – Пример выполнения программы

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

**Лабораторная работа №7**

**Тема: «Строки»**

**Цель работы:** Научиться работать со строками на языке C#.

**Задание:**  Вариант 8. Написать программу, которая считывает текст из файла и выводит на экран только цитаты, то есть предложения, заключенные в кавычки.

**Краткая теория**

Строка — это объект типа String, значением которого является текст. Внутренне текст хранится в виде упорядоченной коллекции объектов Char только для чтения. В конце строки C# нет нуль-символов. Поэтому строка C# может содержать любое число внедренных нуль-символов ('\0'). Свойство Length строки представляет число объектов Char, содержащихся в ней, но не число символов Юникода.

String.Split - метод (Char[])

Возвращаемое значение

Тип: System.String[]

Массив, элементы которого содержат подстроки данного экземпляра, разделенные одним или более знаками из separator (массивом Char[]). Каждый элемент в *separator* определяет отдельный знак-разделитель.

Метод String.Replace (String, String)

Возвращает новую строку, в которой все вхождения заданной строки в текущем экземпляре заменены другой заданной строкой.

Класс StreamReader

Реализует объект TextReader, который считывает символы из потока байтов в определенной кодировке. Метод класса ReadToEnd() считывает все символы, начиная с текущей позиции до конца потока. (Переопределяет TextReader.ReadToEnd().)

**Практическая часть**

Код программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_Kur6

{

class Program

{

static void Main(string[] args)

{

string s = "123 \"мир\" 234\"труд\" 777 \"май\""; //строка из которой будем брать цитаты

Console.WriteLine(s);

bool citata = false; // логическое значение, определяет читаются символы из цитаты

// или из обычного текста

for (int i = 1; i < s.Length; i++) // перебираем последовательно все символы в строке

{

if (s[i] == '"' && (!citata)) // если встретилась кавычка, значит началась цитата

{

citata = true;

continue; // завершает текущую итерацию цикла

// (чтобы не выводилась на экран кавычка в начале цитаты)

}

else if (s[i] == '"' && (citata)) // если встретилась кавычка и значение citata было true,

// то это конец цитаты

{

citata = false;

Console.WriteLine();

}

if (citata) Console.Write(s[i]); // выводить на экран символы, если они относятся к цитате

}

Console.ReadLine();

}

}

}



Рисунок 1 – Блок-схема реализации программы

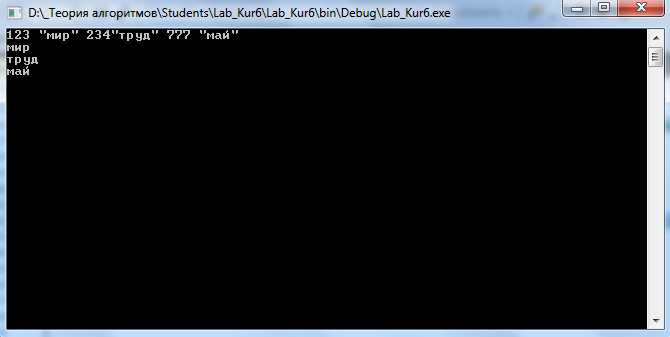


Рисунок 1 – Пример выполнения программы

**Вывод:** Изучена работа со строками на языке C# в среде программирования Microsoft Visual Studio.