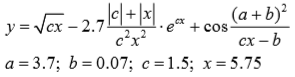
ЛИНЕЙНЫЕ АЛГОРИТМЫ.

**Цель лабораторной работы**: научиться составлять каркас простейшей программы в среде Visual Studio. Написать и отладить программу линейного алгоритма.

**Задание: Task:**



**Теория:**

Структура приложения

Перед началом программирования необходимо создать проект. Проект содержит все исходные материалы для приложения, такие как файлы исходного кода, ресурсов, значки, ссылки на внешние файлы, на которые опирается программа, и данные конфигурации, такие как параметры компилятора.

Кроме понятия проект часто используется более глобальное понятие – решение (solution). Решение содержит один или несколько проектов, один из которых может быть указан как стартовый проект. Выполнение решения начинается со стартового проекта.

Таким образом, при создании простейшей C# программы в Visual Studio создается папка решения, в которой для каждого проекта создается подпапка проекта, а уже в ней – другие подпапки с результатами компиляции приложения.

Проект – это основная единица, с которой работает программист. При создании проекта можно выбрать его тип, а Visual Studio создаст каркас проекта в соответствии с выбранным типом.

Проект в Visual Studio состоит из файла проекта (файл с расширением .csproj), одного или нескольких файлов исходного текста (с расширением .cs), файлов с описанием окон формы ( с расширением .designer.cs), файлов ресурсов (с расширением .resx), а также ряда служебных файлах.

В файле проекта находится информация о модулях, составляющих данный проект, входящих в него ресурсах, а также параметров построения программы. Файл проекта автоматически создается и изменяется средой Visual Studio и не предназначен для ручного редактирования.

Файл исходного текста – программный модуль, предназначен для размещения текстов программ. В этом файле программист размещает текст программы, написанный на языке C#.

В разделе подключения пространств имен (каждая строка которого располагается в начале файла и начинается ключевым словом using) описываются используемые пространства имен. Каждое пространство имен включает в себя классы, выполняющие определенную работу, например, классы для работы с сетью располагаются в пространстве System.Net, а для работы с файлами – в System.IO. Большая часть пространств, которые используются в обычных проектах, уже подключена при создании нового проекта, но при необходимости можно дописать дополнительные пространства имен.

Для того чтобы не происходило конфликтов имен классов и переменных, классы нашего проекта также помещаются в отдельное пространство имен. Определяется оно ключевым словом namespace, после которого следует имя пространства (обычно оно совпадает с именем проекта).

Внутри пространства имен помещаются наши классы – в новом проекте это класс окна, который содержит все методы для управления поведением окна. Обратите внимание, что в определении класса присутствует ключевое слово partial, это говорит о том, что в исходном тексте представлена только часть класса, с которой мы работаем непосредственно, а служебные методы для обслуживания окна скрыты в другом модуле (при желании их тоже можно посмотреть, но редактировать вручную не рекомендуется).

Наконец, внутри класса располагаются переменные, методы и другие элементы программы. Фактически, основная часть программы размещается внутри класса при создании обработчиков событий.

При компиляции программы Visual Studio создает исполняемые .exe-файлы в каталоге bin.

Целочисленные типы

В С# определены девять целочисленных типов: char, byte, sbyte, short, ushort, int, uint, long и ulong. Тип char может хранить числа, но чаще используется для представления символов. Остальные восемь целочисленных типов предназначены для числовых расчетов.

Некоторые целочисленные типы могут хранить как положительные, так и отрицательные значения (sbyte, short, int и long), другие же – только положительные (char, byte, ushort, uint и ulong).

Типы с плавающей точкой

Такие типы позволяют представлять числа с дробной частью. В С# имеются три разновидности типов данных с плавающей точкой: float, double и decimal. Первые два типа представляют числовые значения с одинарной и двойной точностью, вычисления над ними выполняются аппаратно и поэтому быстро. Тип decimal служит для представления чисел с плавающей точкой высокой точности без округления, характерного для типов float и double. Вычисления с использованием этого типа выполняются программно и поэтому более медленны.

Числа, входящие в выражения, C# по умолчанию считает целочисленными. Поэтому следующее выражение будет иметь значение 0, ведь если 1 нацело разделить на 2, то получится как раз 0:

double x = 1 / 2;

Чтобы этого не происходило, в подобных случаях нужно явно указывать тип чисел с помощью символов-модификаторов: f для float и d для double. Приведенный выше пример правильно будет выглядеть так:

double x = 1d / 2d;

Иногда в программе возникает необходимость записать числа в экспоненциальной форме. Для этого после мантиссы числа записывается символ «e» и сразу после него – порядок. Например, число 2,5·10–2 будет записано в программе следующим образом:

2.5e‐2

Символьные типы

В С# символы представлены не 8-разрядным кодом, как во многих других языках программирования, а 16-разрядным кодом, который называется юникодом (Unicode). В юникоде набор символов представлен настолько широко, что он охватывает символы практически из всех естественных языков на свете.

Основным типом при работе со строками является тип string, задающий строки переменной длины. Тип string представляет последовательность из нуля или более символов в кодировке Юникод. По сути, текст хранится в виде последовательной доступной только для чтения коллекции объектов char.

Логический тип данных

Тип bool представляет два логических значения: «истина» и «ложь». Эти логические значения обозначаются в С# зарезервированными словами true и false соответственно. Следовательно, переменная или выражение типа bool будет принимать одно из этих логических значений.

Рассмотрим самые популярные данные – переменные и константы. Переменная – это ячейка памяти, которой присвоено некоторое имя, и это имя используется для доступа к данным, расположенным в данной ячейке.

Для каждой переменной задается тип данных – диапазон всех возможных значений для данной переменной. Объявляются переменные непосредственно в тексте программы. Лучше всего сразу присвоить им начальное значение с помощью знака присвоения «=» (переменная = значение):

int a;  // Только объявление

int b = 7; // Объявление и инициализация

Для того чтобы присвоить значение символьной переменной, достаточно заключить это значение (т. е. символ) в одинарные кавычки:

char ch;    // Только объявление  char symbol = 'Z'; // Объявление и инициализация

Частным случаем переменных являются константы. Константы – это переменные, значения которых не меняются в процессе выполнения программы. Константы описываются как обычная переменная, только с ключевым словом const впереди:

const int c = 5;

Ввод/вывод данных в программу

Рассмотрим один из способов ввода данных через элементы, размещенные на форме. Для ввода данных чаще всего используют элемент управления TextBox, через обращение к его свойству Text. Свойство Text хранит в себе строку введенных символов. Поэтому данные можно считать таким образом:

private void button1\_Click(object sender,EventArgs e)  {    string s = textBox1.Text;  }

Однако со строкой символов трудно производить арифметические операции, поэтому лучше всего при вводе числовых данных перевести строку в целое или вещественное число. Для этого у типов int и double существуют методы Parse для преобразования строк в числа. С этими числами можно производить различные арифметические действия. Таким образом, предыдущий пример можно переделать следующим образом:

private void button1\_Click(object sender, EventArgs e)  {    string s = textBox1.Text;

int a = int.Parse(s);

int b = a \* a;  }

Перед выводом числовые данные следует преобразовать назад в строку. Для этого у каждой переменной существует метод ToString(), который возвращает в результате строку с символьным представлением значения. Вывод данных можно осуществлять в элементы TextBox или Label, используя свойство Text. Например:

private void button1\_Click(object sender, EventArgs e)  {    string s = textBox1.Text;

int a = int.Parse(s);

int b = a \* a;

label1.Text = b.ToString();  }

Арифметические действия и стандартные функции

При вычислении выражения, стоящего в правой части оператора присвоения, могут использоваться арифметические операции:

 умножение (×);

 сложение (+);

 вычитание (–);

 деление (/);

 остаток от деления (%).

Для задания приоритетов операций могут использоваться круглые скобки ( ). Также могут использоваться стандартные математические функции, представленные методами класса Math:

 Math.Sin(a) – синус;

 Math.Sinh(a) – гиперболический синус;

Math.Cos(a) – косинус (аргумент задается в радианах);

 Math.Atan(a) – арктангенс (аргумент задается в радианах);

 Math.Log(a) – натуральный логарифм;

 Math.Exp(a) – экспонента;

 Math.Pow(x, y) – возводит переменную x в степень y;

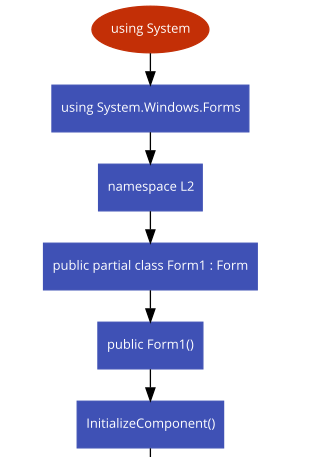
 Math.Sqrt(a) – квадратный корень;

 Math.Abs(a) – модуль числа;

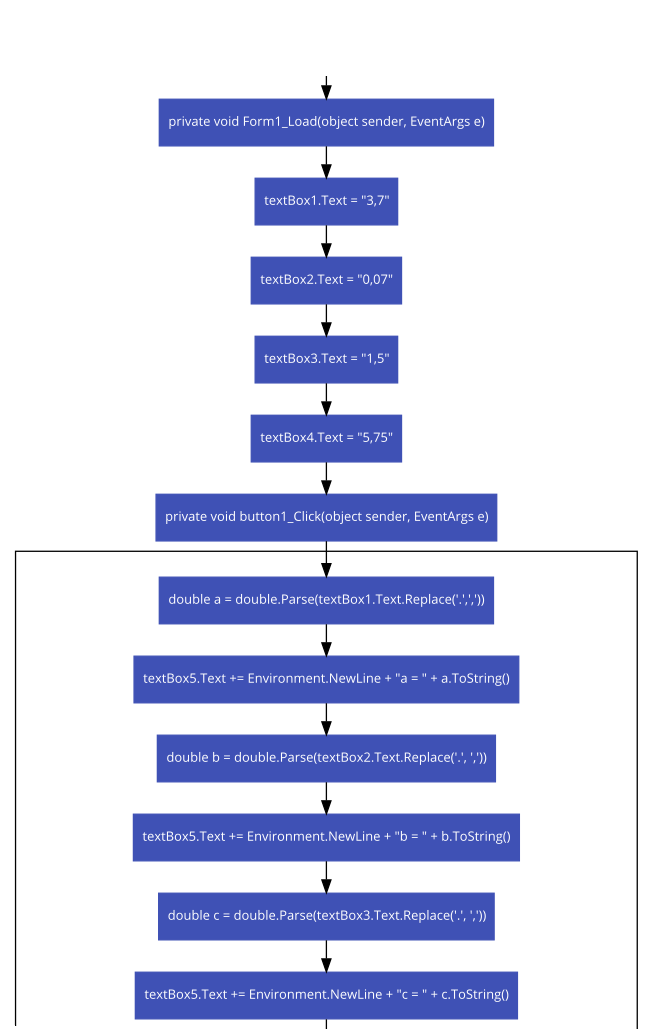
Math.Truncate(a) – целая часть числа;

 Math.Round(a) – округление числа.

В тригонометрических функциях все аргументы задаются в радианах.



1

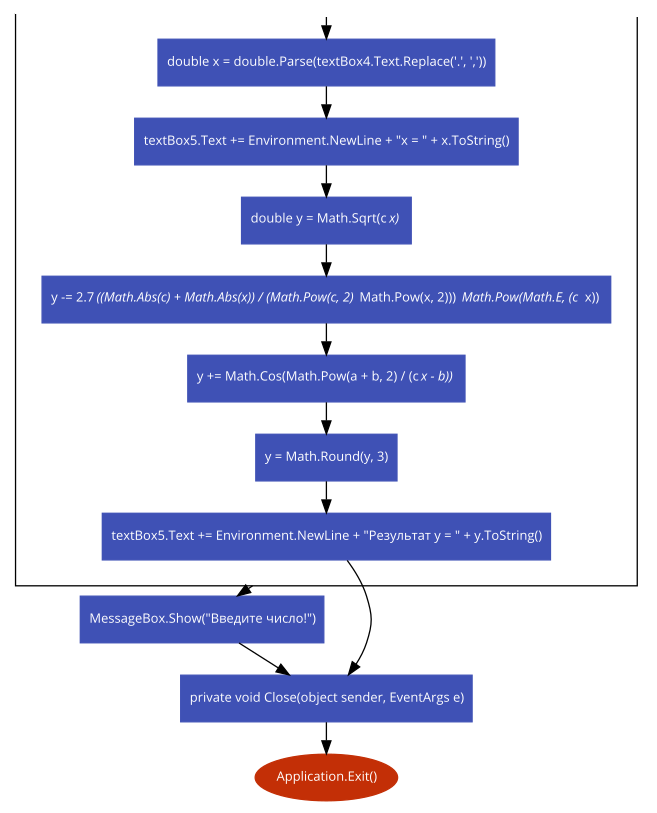


4

3

2

1



4

2

3

**Код программы:**

using System;

using System.Windows.Forms;

namespace L2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

textBox1.Text = "3,7";

textBox2.Text = "0,07";

textBox3.Text = "1,5";

textBox4.Text = "5,75";

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

double a = double.Parse(textBox1.Text.Replace('.',','));

textBox5.Text += Environment.NewLine + "a = " + a.ToString();

double b = double.Parse(textBox2.Text.Replace('.', ','));

textBox5.Text += Environment.NewLine + "b = " + b.ToString();

double c = double.Parse(textBox3.Text.Replace('.', ','));

textBox5.Text += Environment.NewLine + "c = " + c.ToString();

double x = double.Parse(textBox4.Text.Replace('.', ','));

textBox5.Text += Environment.NewLine + "x = " + x.ToString();

double y = Math.Sqrt(c \* x);

y -= 2.7 \* ((Math.Abs(c) + Math.Abs(x)) / (Math.Pow(c, 2) \* Math.Pow(x, 2))) \* Math.Pow(Math.E, (c \* x)); ;

y += Math.Cos(Math.Pow(a + b, 2) / (c \* x - b));

y = Math.Round(y, 3);

textBox5.Text += Environment.NewLine + "Результат y = " + y.ToString();

}

catch

{

MessageBox.Show("Введите число!");

}

}

private void Close(object sender, EventArgs e)

{

Application.Exit();

}

}

}

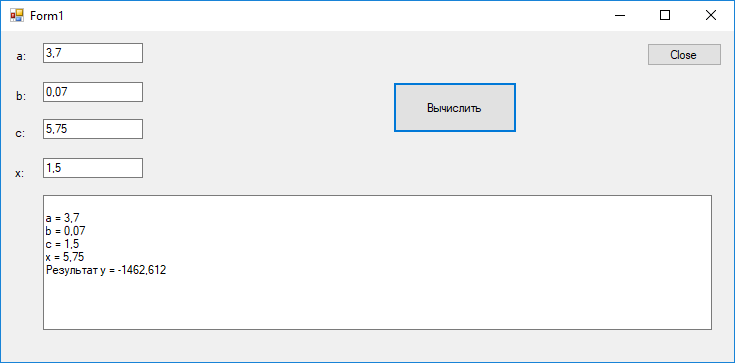


Рис.1 Работа программы со стандартными значениями.

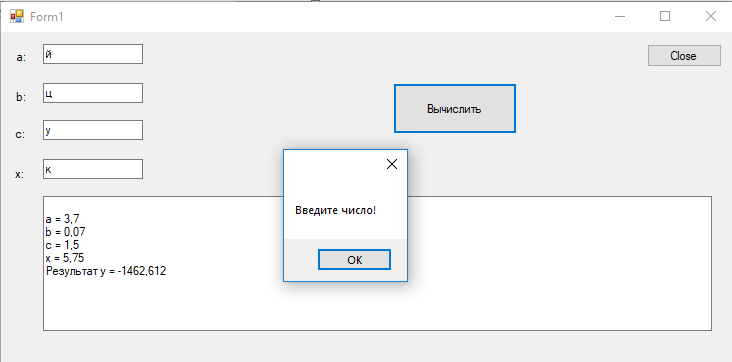


Рис.2 Ввод символов вместо чисел

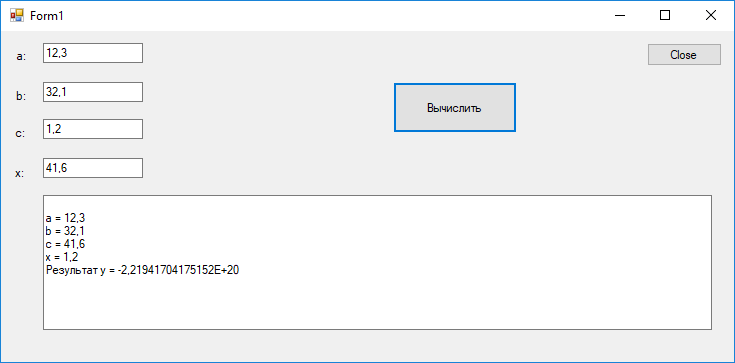


Рис.3 Ввод чисел с запятой вместо точки

**Вывод:**

Научились составлять карка простейшей программы в среде Visual Studio. Написали и отладили программу по заданным заданиям.