МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__» _____ 2019 г.

ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ НА ЯЗЫКЕ C#

Методические указания по выполнению лабораторной работы по дисциплинам «Программирование на языках высокого уровня», «Языки программирования» для студентов направлений подготовки 09.03.04 «Программная инженерия», 10.03.01 «Информационная безопасность», 10.05.02 «Информационная безопасность телекоммуникационных систем»

Курск 2019

УДК 681.3.06(071.8)

Составители: Т.М. Белова, В.Г. Белов

Рецензент
Кандидат технических наук, доцент кафедры программной инженерии ЮЗГУ И.Н. Ефремова

Программирование линейных алгоритмов на языке C#: методические указания по выполнению лабораторной работы по дисциплинам «Программирование на языках высокого уровня», «Языки программирования» для студентов направлений подготовки 09.03.04 «Программная инженерия», 10.03.01 «Информационная безопасность», 10.05.02 «Информационная безопасность телекоммуникационных систем»/ Юго-Зап. гос. ун-т; сост. Т.М. Белова, В.Г. Белов. Курск, 2019. – 26 с.

Содержат основные теоретические положения и приемы разработки линейных программ на языке C#, пример решения типовой задачи, индивидуальные задания и контрольные вопросы к защите лабораторной работы.

Методические указания соответствуют требованиям рабочих программ по дисциплинам «Программирование на языках высокого уровня», «Языки программирования».

Предназначены для студентов дневной и заочной форм обучения направлений подготовки 09.03.04 «Программная инженерия», 10.03.01 «Информационная безопасность», 10.05.02 «Информационная безопасность телекоммуникационных систем».

Текст печатается в авторской редакции.

Подписано в печать . Формат 60x84 1/16.
Усл. печ. л. . Уч.-изд. л. . Тираж 100 экз. Заказ . Бесплатно.
Юго-Западный государственный университет
305040, Курск, ул.50 лет Октября, 94.

ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ НА ЯЗЫКЕ C#

Цель работы − изучение и приобретение навыков программирования линейных алгоритмов, освоение оператора присваивания, операторов ввода (компонент *TextBox*) и вывода (компоненты *Label, TextBox* и *Button*), типов данных, арифметических операций и функций языка *C#***.**

Основные понятия

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

Кроме понятия проекта 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, для *C#* программы в ***MS Visual Studio*** создается папка решения, в которой для каждого проекта создается подпапка проекта и создаются другие подпапки с результатами компиляции приложения.

Проект *C#* состоит из форм, модулей, установок параметров проекта, ресурсов и т. д. Вся эта информация размещается в файлах. Многие из этих файлов создаются ***MS Visual Studio*** при разработке приложения. Ресурсы, такие, как битовые матрицы, пиктограммы и т. д., находятся в файлах, которые программист получает из других источников или создает при помощи многочисленных инструментов и редакторов ресурсов. Кроме того, компилятор также создает файлы.

Файлы, создающиеся *MS Visual Studio* при проектировании приложения

Головной файл проекта (.cs)

MS Visual Studio создает этот файл для головной функции ***Main***, которая инициирует программу и запускает ее на выполнение.

Файл опций проекта (.config)

Файлы конфигурации приложения содержат настройки, относящиеся к отдельному приложению.

Файл реализации модуля(.cs)

Этот файл используется для хранения кода модулей. Некоторые модули связаны с формами; в некоторых модулях хранятся только классы с методами.

Файлы с описанием окон формы

Файлы с описанием окон формы имеют расширение *.designer.cs*

Файлы проекта (.csproj)

XML-файл, содержащий всю информацию о проекте.

Файл ресурсов проекта (.resx)

Этот бинарный файл содержит используемую проектом пиктограмму. Этот файл не должен изменяться или создаваться пользователем, так как ***MS Visual Studio*** постоянно модифицирует и пересоздает этот файл.

Файлы решений (.sln)

Файл решения организует проекты, элементы проектов и решений.

Файлы, создающиеся компилятором

Исполняемый файл (.exe)

Это исполняемый файл приложения. Он является автономным исполняемым файлом, для которого больше ничего не требуется, если только не используются библиотеки, содержащиеся в пакетах, *DLL, OCX* и так далее.

Файл базы данных программы (.pdb)

Файл базы данных программы (*PDB*) содержит отладочные данные и сведения о состоянии проекта, позволяющие выполнять

последовательную компоновку отладочной конфигурации программы.

Динамически присоединяемая библиотека (.dll)

Этот файл создается в случае, если разрабатывается собственная динамическая библиотека *DLL*.

Структура приложения

Самую простую структуру имеет консольное приложение. Для создания консольного приложения в ***MS Visual Studio*** необходимо сделать следующее. В меню «*File*» («*Файл*») выбрать пункт «*New*» («*Новый*») и в нем выбрать подпункт «*Project*» («*Проект*»). Далее в появившемся диалоговом окне, в разделе Project types необходимо выбрать пункт *Visual C#* и подпункт *Windows*. В разделе *Templates* выбрать *Console Application*. В нижней части окна необходимо задать имя проекта и папку, где он будет сохраняться. При этом необходимо следить, чтобы папка, в которую сохраняется проект, была доступна для записи.

Ввод данных с клавиатуры осуществляется посредством метода *ReadLine()* класса *Console*. Данный метод возвращает строку, которую ввел с клавиатуры пользователь. Вывод данных на экран осуществляется с помощью метода *WriteLine()* класса *Console*. Код программы следует писать между фигурными скобками (внутри тела) метода *Main*:

```
public static void Main(string[] args) {  }
```

Пример консольного приложения:

```
 using System;
 public class Program
{    public static void Main(string[] args)
     {Console.WriteLine(“Введите свое имя”);
     string st=Console.ReadLine();
     Console.WriteLine(“Привет {0}”,st);
     Console.ReadLine();
} }
```

Пояснения к программе

В разделе подключения пространств имен (каждая строка которого располагается в начале файла и начинается ключевым словом *using*) описываются используемые пространства имён. В данном случае задается пространство имен *System* – основное пространство имен библиотеки *BCL (Base Class Library),* или так называемая *.NET FCL* (англ. *Framework Class Library*), сокращённо *FCL* — стандартная библиотека классов платформы «*.NET Framework*».

Ввод информации с клавиатуры

С клавиатуры вы получаете всегда СТРОКУ. Соответственно, для получения чисел или символов необходимо выполнять преобразования.

```
string s=Console.ReadLine();
```

Для преобразования строки в целочисленный формат:

```
int a=0; a = Convert.ToInt32(s); //получение числа
```

Можно более кратко:

```
int a = Convert.ToInt32(Console.ReadLine());
 //объявление, считывание и преобразование.
```

С помощью класса *Convert* можно преобразовывать ко всем простым типам данных. Для выполнения преобразования после имени *Convert* необходимо поставить точку и в появившемся списке выбрать нужный метод (по имени типа данных).

Вывод информации на экран

Вывод информации на экран осуществляется конструкцией:

```
Console.WriteLine (аргументы);
```

Или:

```
Console.WriteLine(“{0},{1}”, аргументы);
```

Для вывода числовой переменной следует выполнить преобразование в строковый формат.

Пример:

```
int a=999; Console.WriteLine (“Значение a={0}”, a.ToString());
```

Комментарии

Для того чтобы закомментировать строку кода, перед ней ставим два слеша //. Весь текст до конца строки, следующий за

этой парой символов, (например, «//любой текст») воспринимается как комментарий, не влияющий на выполнение программного кода. Началом многострочного комментария является пара символов /*, а концом - */.

Сразу после создания проекта рекомендуется сохранить его в подготовленной папке: *Файл −> Сохранить всё*. При внесении значительных изменений в проект следует еще раз сохранить проект той же командой, а перед запуском программы на выполнение среда обычно сама сохраняет проект на случай какого-либо сбоя. Для открытия существующего проекта используется команда *Файл -> Открыть проект*, либо можно найти в папке файл проекта с расширением ***.csproj*** и сделать на нём двойной щелчок.

При разработке оконного приложения или иначе приложения *Windows Forms* создается следующий код:

```
// Раздел подключенных пространств имен
using System;
// Пространство имен нашего проекта
namespace MyFirstApp
{
// Класс окна
public partial class Form1 : Form
{
// Методы окна
public Form1()
{
InitializeComponent();
}
}
}
```

Пояснения к программе

В разделе подключения пространств имен (каждая строка которого располагается в начале файла и начинается ключевым словом *using*) описываются используемые пространства имён. Каждое пространство имён включает в себя классы, выполняющие определённую работу, например, классы для работы с сетью располага-

ются 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 имён.

Для того чтобы не происходило конфликтов имён 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 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.

Описание данных

Типы данных имеют особенное значение в языке *C#*, поскольку это строго типизированный язык. Это означает, что все операции подвергаются строгому контролю со стороны компилятора на соответствие типов, причем недопустимые операции не компилируются. Такая строгая проверка типов позволяет предотвратить ошибки и повысить надежность программ. Для обеспечения контроля типов все переменные, выражения и значения должны принадлежать к определенному типу. Такого понятия, как "бестиповая" переменная, в данном языке программирования вообще не существует. Более того, тип значения определяет те операции, которые разрешается выполнять над ним. Операция, разрешенная для одного типа данных, может оказаться недопустимой для другого.

В *C#* имеются две общие категории встроенных типов данных: типы значений и ссылочные типы. Они отличаются по содержимому переменной. Концептуально разница между ними состоит в том, что тип значения (*value type*) хранит данные непосредственно, в то время как ссылочный тип (*reference type*) хранит ссылку на значение (рисунок 1).

Эти типы сохраняются в разных местах памяти: типы значений сохраняются в области, известной как стек, а ссылочные типы − в области, называемой управляемой кучей.

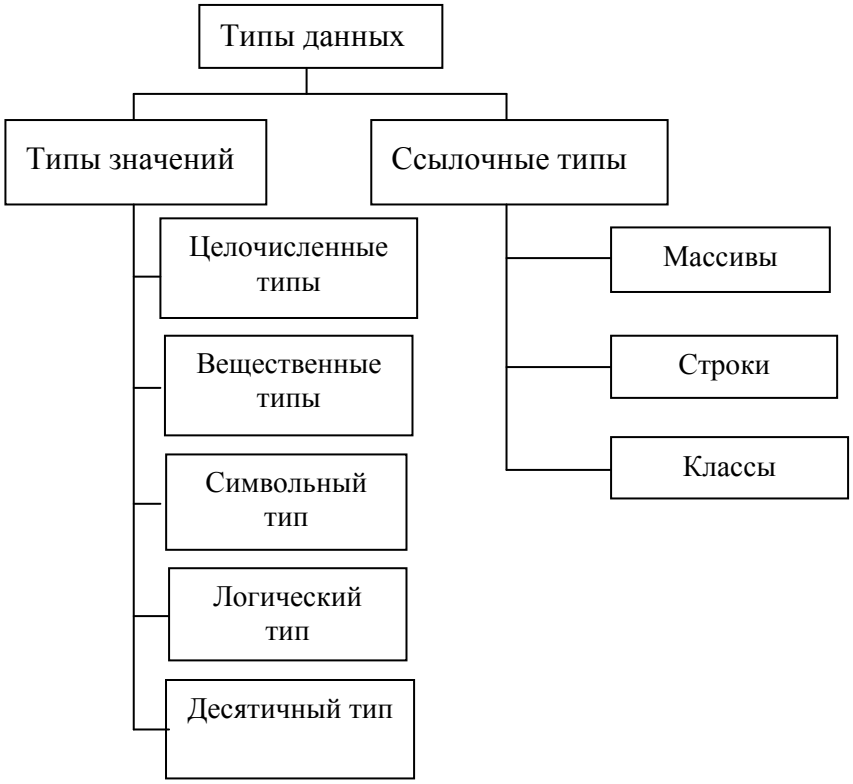

Рисунок 1 – Структура типов данных

Целые типы

В *C#* определены восемь целочисленных типов: *byte*, *sbyte*, *short*, *ushort*, *int*, *uint*, *long* и *ulong* (таблица 1).

Таблица 1 − Целочисленные типы

Тип	Описание	Диапазон	Примеры объявлений
sbyte	8-битовый знаковый целый (1 байт)	-128 : 127	sbyte val = 12;

Тип	Описание	Диапазон	Примеры объявлений
short	16-битовый знаковый целый (2 байта)	-32768 : 32767	short val = 12;
int	32-битовый знаковый целый (4 байта)	-2 147 483 648 : 2 147 483 647	int val = 12;
long	64-битовый знаковый целый (8 байтов)	-9223372036854775808: 9223372036854775807	long val1 = 12; long val2 = 34L;
byte	8-битовый беззнаковый целый (1 байт)	0 : 255	byte val1 = 12;
ushort	16-битовый беззнаковый целый (2 байта)	0 : 65535	ushort val1 = 12;
uint	32- битовый беззнаковый целый (4 байта)	0 : 4 294 967 295	uint val1 = 12; uint val2 = 34U;
ulong	64- битовый беззнаковый целый (8 байтов)	0 : 18446744073709551615	ulong val1 = 12; ulong val2 = 34U; ulong val3 = 56L; ulong val4 = 78UL;

Вещественные типы

Вещественные типы (типы с плавающей точкой) позволяют представлять числа с дробной частью. В *C#* имеются две разновидности типов данных с плавающей точкой: *float* и *double*. Они представляют числовые значения с одинарной и двойной точностью соответственно (таблица 2).

Таблица 2 − Вещественные типы

Тип	Описание	Диапазон	Примеры объявлений
float	Вещественный с плавающей точкой с одинарной точностью (4 байта)	5E-45 до 3,4E+38	float val = 1.23F;
double	Вещественный с плавающей точкой с двойной точностью (8 байтов)	5E-324 до 1,7E+308	double val1 = 1.23; double val2 = 4.56D;

Десятичный тип данных

Для представления чисел с плавающей точкой высокой точности предусмотрен также десятичный тип decimal, который предназначен для применения в финансовых вычислениях. Этот тип имеет разрядность 128 бит для представления числовых значений в пределах от 1E-28 до 7,9E+28. Для обычных арифметических вычислений с плавающей точкой характерны ошибки округления десятичных значений. Эти ошибки исключаются при использовании типа decimal, который позволяет представить числа с точностью до 28 (а иногда и 29) десятичных разрядов. Благодаря этому тип данных decimal способен представлять десятичные значения без ошибок округления, он особенно удобен для расчетов, связанных с финансами

Символы

В *C#* символы представлены не 8-разрядным кодом, как во многих других языках программирования, например *C++*, 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. В *C#* определен тип *char*, представляющий 16-разрядные значения без знака в пределах от 0 до 65 535. При этом стандартный набор символов в 8-разрядном коде *ASCII* является подмножеством юникода в пределах от 0 до 127. Следовательно, символы в коде *ASCII* по-прежнему остаются действительными в *C#.*

Логический тип данных

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

Строки

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. Класс *String* в языке *C#* относится к ссылочным типам. Над строками - объектами этого класса - определен широкий набор операций, соответствующий современному представлению о том, как должен быть устроен строковый тип. По сути, текст хранится в виде последовательной доступной только для чтения коллекции объектов *Char*.

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 задаётся 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 значением
```

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 значением
```

Несмотря на то что тип char определен в C# как целочисленный, его не следует путать со всеми остальными целочисленными типами.

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;
```

В *C#* литералами называются постоянные значения, представленные в удобной для восприятия форме. Например, число 100 является литералом. У литералов должен быть также конкретный тип, поскольку *C#* является строго типизированным языком. По умолчанию литералы с фиксированной точкой относятся к типу int.

Для указания типа long к литералу присоединяется суффикс *l* или *L*. Например, 12 — это литерал типа *int*, а *12L* — литерал типа *long*. Для указания целочисленного типа без знака к литералу присоединяется суффикс *u* или *U*. Следовательно, 100 — это литерал типа *int*, а 100U — литерал типа *uint*. А для указания длинного целочисленного типа без знака к литералу присоединяется суффикс *ul* или *UL*. Например, 984375UL — это литерал типа *ulong*.

Кроме того, для указания типа *float* к литералу присоединяется суффикс *F* или *f*. Например, 10.19F — это литерал типа *float*. Можете даже указать тип *double*, присоединив к литералу суффикс *d* или *D*, хотя это излишне. Ведь, как упоминалось выше, по умолчанию литералы с плавающей точкой относятся к типу *double*.

И, наконец, для указания типа *decimal* к литералу присоединяется суффикс *m* или *M*. Например, 9.95M — это десятичный литерал типа *decimal*.

Математические функции

Для работы с данными могут использоваться математические функции, представленные ниже (таблица 3). Для тригонометрических функций угол задается в радианах.

Таблица 3 − Стандартные математические функции

Функция	Описание	Примеры
Math.Abs	Возвращаем абсолютное число, имеет 7 перегрузок, то есть метод принимает разные типы переменных.	int i = Math.Abs(x);
Math.Acos	Арккосинус. Определяется угол, косинус которого равен указанному числу.	double i =Math.Acos(0.5);
Math.Asin	Арксинус. Определяется угол, синус которого равен указанному числу.	double i = Math.Asin(0.5);
Math.Atan	Арктангенс. Возвращает угол, тангенс которого равен указанному числу.	double i = Math.Atan(0.5);
Math.Cos	Возвращает косинус угла.	double x =Math.Cos(1.4);
Math.Exp	Экспоненциальная функция e^x	double x = Math.Exp(2);

Функция	Описание	Примеры
Math.Log	Вычисление логарифма. X - аргумент, Osn - основание логарифма.	double y = Math.Log(X,Osn);
Math.Log10	Вычисление десятичного логарифма.	Double x = Math.Log10(10)
Math.Max	Возвращает из 2-х чисел большее число. Имеет 11 перегруженых методов.	Int x = Math.Max(10,20);
Math.Min	Возвращает из 2-х чисел меньшее число. Имеет 11 перегруженых методов.	int x = Math.Min(10,20);
Math.PI	Возвращает число Пи	Double pi = Math.PI;
Math.Pow	Вычисляет число, возведенное в степень: a^x	double i = Math.Pow(a, x);
Math.Sin	Возвращает синус угла.	double p = Math.Sin(0.5);
Math.Sqrt	Возвращает квадратный корень.	double r = Math.Sqrt(7);
Math.Tan	Возвращает тангенс угла.	Double p = Math.Tan(1.04);
Math.Floor	Возвращает наибольшее целое число, которое меньше или равно заданному числу с плавающей запятой двойной точности.	double p = Math. Floor (7.04); Ответ : 7
Math.Ceiling	Возвращает наименьшее целое число, которое больше или равно указанному числу.	double p = Math.Ceiling (7.04); Ответ : 8

Оператор присваивания

Оператор присваивания обозначается одиночным знаком равенства (=). В *C#* оператор присваивания действует таким же образом, как и в других языках программирования. Ниже приведена его общая форма:

имя_переменной = выражение;

Здесь имя_переменной должно быть совместимо с типом выражения. У оператора присваивания имеется одна интересная особенность, о которой полезно знать: он позволяет создавать цепочку операций присваивания. Рассмотрим следующий фрагмент кода:

int x, y, z;

```
x = y = z = 10; // присвоить значение 10 переменным x, y и z
```

В приведенном выше фрагменте кода одно и то же значение 10 задается для переменных x, y и z с помощью единственного оператора присваивания. Это значение присваивается сначала переменной z, затем переменной y и, наконец, переменной x. Такой способ присваивания "по цепочке" удобен для задания общего значения целой группе переменных.

Укороченные операторы присваивания

В *C#* предусмотрены специальные *укороченные* **(**составные) операторы (таблица 4) присваивания, упрощающие программирование некоторых операций присваивания. Обратимся сначала к простому примеру:

```
x = x + 3;
// Можно переписать следующим образом
x += 3;
```

Пара операторов += указывает компилятору на то, что переменной x должно быть присвоено ее первоначальное значение, увеличенное на 3.

Таблица 4 − Укороченные операторы присваивания

Оператор	Аналог
x += 1	x = x + 1;
x -= 1	x = x - 1;
x *= 1	x = x * 1;
x /= 1	x = x / 1;
x %= 1	x = x % 1;
x |= 1	x = x | 1;
x ^= 1	x = x ^ 1;

У укороченных операторов присваивания имеются два главных преимущества. Во-первых, они более компактны, чем их "несокращенные" эквиваленты. И, во-вторых, они дают более эффективный исполняемый код, поскольку левый операнд этих операторов вычисляется только один раз. Именно по этим причинам составные операторы присваивания чаще всего применяются в программах, профессионально написанных на *C#*.

Операции присваивания возвращают как результат присвоенное значение, поэтому допускается сцепление

Операции инкремента + + и декремента - -

Операции инкремента (+ +) и декремента (- -) сводятся к увеличению (+ +) или уменьшению (- -) операнда на единицу. Эти операции выполняются быстрее, чем обычное сложение и вычитание.

Если операция инкремента или декремента помещена перед переменной, говорят о префиксной форме записи (++n или - -n). Если операция инкремента или декремента записана после переменной, то говорят о постфиксной форме записи (n++ или n- -). При префиксной форме переменная сначала увеличивается или уменьшается на единицу, а затем ее новое значение используется. При постфиксной форме в выражении используется текущее значение переменной, а после ее значение изменяется на единицу.

Пример:

1) int i=1;
int j = i+ + *5;

Результат:

j = 1*5 = 5
i = 2

2) int i=1;
int j = + +i * 5;

Результат:

j = 2*5 = 10
i = 2

Рассмотрим приоритеты операций (таблица 5).

Таблица 5 − Приоритеты операций

Приоритет	Категория	Операции	Порядок
0	Первичные	(expr); x.y; f(x); a[x]; x++; x new; sizeof(t); typeof(t); checked(expr); unchecked(expr)	Слева направо
1	Унарные	+ - ! ~ ++x --x (T)x	Слева направо
2	Мультипликативные (Умножение)	- * / %	Слева направо
3	Аддитивные (Сложение)	+ -	Слева направо
4	Сдвиг	<< >>	Слева направо
5	Отношения, проверка типов	< > <= >= is as	Слева направо

Приоритет	Категория	Операции	Порядок
6	Эквивалентность	= = !=	Слева направо
7	Логическое И	&	Слева направо
8	Логическое исключающее ИЛИ (XOR)	^	Слева направо
9	Логическое ИЛИ (OR)	|	Слева направо
10	Условное И	&&	Слева направо
11	Условное ИЛИ	||	Слева направо
12	Условное выражение	? :	Справа налево
13	Присваивание	= *= /= %= += -= <<= >>= &= ^= |=	Слева направо

Пример программирования линейного алгоритма

Задание. Даны действительные числа x, y, z. Составить программу вычисления для заданных значений x, y, z арифметического выражения

$$u = \tan{(x+y)^2} - e^{y-z}\sqrt{\cos(x^2) + \sin(z^2)}$$

1. На рисунке 2 – разработка алгоритма:
– входные данные: x, y, z – действительные числа;
– выходные данные: u– действительное число.

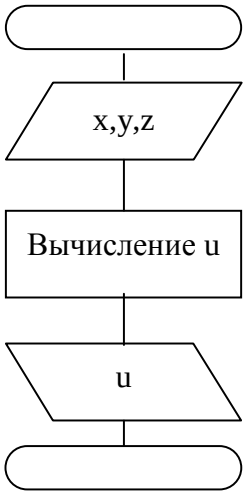

Рисунок 2 − Схема алгоритма решения задачи

2. На рисунке 3 – разработка формы.

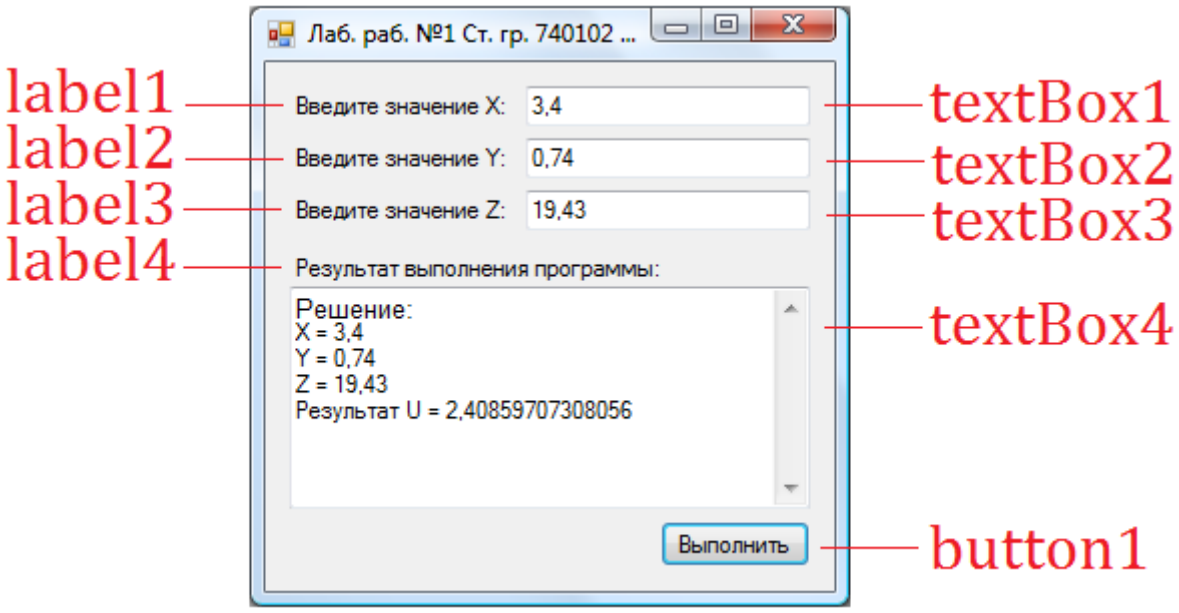

Рисунок 3 − Внешний вид формы

Для вывода результатов работы программы используется текстовое окно, которое представлено элементом управления *textBox*. После установки свойства *Multiline* в *true* появляется возможность растягивать элемент управления не только по горизонтали, но и по вертикали. А после установки свойства *ScrollBars* в значение *Both* в окне появится вертикальная, а при необходимости и горизонтальная полосы прокрутки.

Информация, которая отображается построчно в окне, находится в массиве строк *Lines*, каждая строка которого имеет тип *string*. Однако напрямую нельзя обратиться к этому свойству для добавления новых строк, поскольку размер массивов в *C#* определяется в момент их инициализации. Для добавления нового элемента используется свойство *Text*, к текущему содержимому которого можно добавить новую строку:

```
textBox4.Text += Environment.NewLine + "Привет";
```

В этом примере к текущему содержимому окна добавляется символ перевода курсора на новую строку (который может отличаться в разных операционных системах и потому представлен свойством класса *Environment*) и сама новая строка. Если добавляется числовое значение, то его предварительно нужно привести в символьный вид методом *ToString()*.

Работа с программой происходит следующим образом. Нажмите (щелкните мышью) кнопку “Выполнить”. В окне *textBox4* появляется результат. Измените исходные значения x, y, z в окнах *textBox1 – textBox3* и снова нажмите кнопку ”Выполнить” - появится новые результаты.

Текст программы имеет следующий вид:

```
using System;
using System.Windows.Forms;
namespace MyFirstApp
{
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
}
private void Form1_Load(object sender, EventArgs e)
{
// Вывод строки в многострочный редактор
textBox4.Text = "Решение:";
}
private void button1_Click(object sender, EventArgs e)
{
// Считывание значения x
double x = double.Parse (textBox1.Text);
// Вывод значения x в окно
textBox4.Text += Environment.NewLine +
"X = " + x.ToString();
// Считывание значения y
double y = double.Parse (textBox2.Text);
// Вывод значения y в окно
textBox4.Text += Environment.NewLine +
"Y = " + y.ToString();
// Считывание значения Z
double z = double.Parse (textBox3.Text);
// Вывод значения Z в окно
textBox4.Text += Environment.NewLine +
"Z = " + z.ToString();
// Вычисляем арифметическое выражение
double a = Math.Tan (x + y) * Math.Tan (x + y);
double b = Math.Exp (y - z);
```

```
double c = Math.Sqrt (Math.Cos(x * x) + Math.Sin (z * z));
double u = a - b * c;
// Выводим результат в окно
textBox4.Text += Environment.NewLine +
"Результат U = " + u.ToString();
}
}
}
```

Индивидуальные задания

Решите две задачи из первого и второго уровней сложности.

Задачи первого уровня сложности

Вычислите значения f и g, если:

Вариант	f	g
1	$3\sin(2^x)+0.35x^3-3.8$	$\ln(x)-x+1.8$
2	$0.25\ x^4+3x+2.5$	$x^8 * \operatorname{tg} x-1/3$
3	$\sin(\ln\|x\|)-\cos(\ln\|x\|)$	$\operatorname{tg}(x/2)-\operatorname{ctg}(x/2)+x$
4	$x-2+\sin(1/x)$	$0.4x^{12}+\operatorname{arctg}(\sqrt[3]{x})-x$
5	$e^{2x+4}+\ln\|x\|-10\ x^5+3x$	$1-2x^7-\cos(3x)$
6	$\cos(2x)-5x^9+1$	$3.6^x-2.3x^5-3$
7	$1-x^7+\sin x-\ln\|1+x\|$	$3x-\sqrt{3.7x}+e^{x+5}$
8	$3\ln(x^2)+6\ln\|x\|-6$	$x+3.6x-\sin x^7$
9	$x-2x*\sin x-\cos x^9$	$\cos(1.3x)+\sqrt[3]{x}*\ln x$
10	$x^2-\ln\|1+x\|-3$	$\ln(x^2+5)-\cos^3 x$
11	$x-1/(3+\sin 3.6x)$	$1/(x^6+13x)-\operatorname{arctg} x$
12	$0.1x^4-x*\ln\|x\|$	$\cos(\operatorname{arctg}^5 x)-3$
13	$\operatorname{tg} x-3\operatorname{tg}^2 x+0.2\operatorname{tg}^3 x$	$3*\ln(x^4+2.8)-x$
14	$\arccos x-\sqrt{\|1-0.3x\|}$	$1/\operatorname{arctg} x+\cos x-4.2$
15	$3x^5-4\ln\|x-5\|$	$(1-2x)/(x^3+5.4)$

Вариант	f	g
16	$\cos(2/x)-2\sin(1/x)$	$\operatorname{arctg} x-\sin^2 x+5$
17	$e^x-e^{-x+2}+5\ln\lvert x-3\rvert$	$13x^6+13\sin x+13\cos x$
18	$\sin(\ln\lvert x\rvert)+\sqrt[3]{x}$	$\ln 0.7x-\cos^7 x$
19	$1+\sin^2(x+y)$	$2+\lvert x-2x/(1+x^y)\rvert$
20	$\cos^2(\operatorname{arctg}(1/y))$	$2\lvert\cos(x-\pi/6)\rvert$
21	$1/(2x)+\sin(y)$	$\sqrt[5]{x+\ln x}$
22	$y^2+\lvert x^2/y+x/4\rvert$	$\sqrt[3]{5x+\lg x}+8.2$
23	$5\lg(x^3)+3.2\lg\lvert x\rvert-2$	$7.2^x-5x-9.8$
24	$\sqrt[3]{x+\lg\lvert x\rvert}$	$\lvert y^5-x+\sin x^2\rvert$
25	$\sqrt[3]{2x+\sin\lvert x\rvert}$	$5*\lg(x^3+9.8)-x^7$
26	$\sqrt[7]{e^{3x+5}+\lvert\operatorname{tg}x^3\rvert}$	$2\lvert\sin(x^3-\pi/3)\rvert$
27	$2x^5+\lvert x+7x/(\operatorname{tg}^3 x+x^y)\rvert$	$\sqrt[5]{4x^3+\lvert\cos^9 x^7\rvert}$
28	$\sqrt[9]{3x^2+\sqrt{\lvert 3\operatorname{tg}x\rvert}}$	$\lg(3x^2+6x+81)-y^8$
29	$\ln(xy^2+6^x+31x)+71.5^x$	$\sqrt[3]{4x^5\cdot\lg\sqrt{\lvert x\rvert}}$
30	$5\ln(x^3)+7.2\log_2\lvert x\rvert/3$	$\sqrt[3]{5e^{9x+5}+\lvert\lg^7 x^3\rvert}$

Задачи второго уровня сложности

1. Вычислите дробную часть среднего геометрического трех заданных положительных целых чисел.

2. По заданным коэффициентам и правым частям системы уравнений $\begin{cases} a_1x+b_1y=c_1 \\ a_2x+b_2y=c_2 \end{cases}$ найдите ее решение в предположении, что определитель системы не равен нулю.

3. По координатам вершин некоторого треугольника найдите его площадь и периметр.

4. По длинам двух сторон некоторого треугольника и углу (в градусах) между ними найдите длину третьей стороны и площадь этого треугольника.

5. Найдите произведение цифр заданного четырехзначного числа.

6. Определите число, полученное выписыванием в обратном порядке цифр заданного трехзначного числа.

7. Присвойте целой переменной d первую цифру из дробной части положительного вещественного числа x (так, если x=32.597, то d=5).

8. Целой переменной присвойте сумму цифр трехзначного целого числа k.

9. Идет k-я секунда суток. Определите, сколько полных часов (h) и полных минут (m) прошло к этому моменту (например, h=3 и m=40, если k=3*3600+40*60+57).

10. Даны катеты прямоугольного треугольника. Найдите его периметр и площадь.

11. Даны два действительных числа. Найдите среднее арифметическое и среднее геометрическое этих чисел.

12. Даны гипотенуза и катет прямоугольного треугольника. Найдите второй катет и радиус вписанной окружности.

13. Известна длина окружности. Найдите площадь круга, ограниченного этой окружностью.

14. Найдите площадь кольца, внутренний радиус которого равен 20, а внешний заданному числу r (r>20).

15. Даны основания и высота равнобедренной трапеции. Найдите ее периметр и площадь.

16. Вычислите дробную часть среднего арифметического трех заданных положительных чисел.

17. По координатам вершин некоторого прямоугольника найдите его площадь и периметр.

18. Дана сторона равностороннего треугольника. Вычислите площадь и периметр треугольника.

19. Дана сторона квадрата. В квадрат вписана окружность. Найдите сторону и площадь квадрата, вписанного в эту окружность.

20. Дан радиус окружности. В окружность вписан квадрат. Найдите площади окружности и квадрата.

21. Равносторонний треугольник задан координатами вершин. Найдите площадь и периметр треугольника.

22. Вычислите целую часть среднего геометрического трех заданных положительных действительных чисел.

23. Вычислите целую часть среднего арифметического четырех заданных действительных чисел.

24. Дано целое трехзначное число. Найдите вторую цифру дробной части среднего арифметического цифр этого числа.

25. Дано целое четырехзначное число. Вычислите дробную часть среднего арифметического цифр этого числа.

26. Дано целое четырехзначное число. Вычислите дробную часть среднего геометрического цифр этого числа.

27. Дано целое пятизначное число. Вычислите среднее геометрическое и среднее арифметическое 1, 3, 5 цифр этого числа.

28. Дано целое пятизначное число. Вычислите сумму и произведение 1, 3, 5 цифр этого числа.

29. Дано целое пятизначное число. Вычислите среднее арифметическое и среднее геометрическое 2, 4 цифр этого числа.

30. Даны x, y, z. Вычислите a, b, если:

$$a = \frac{3 + e^{y-1}}{1 + x\,|y - \mathrm{tg}\ z|}; \quad b = 1 + |y-x| + \frac{(y-x)^2}{2} + \frac{|y-x|^3}{3}.$$

Примечания:

1. Сторона треугольника $c = \sqrt{a^2 + b^2 - 2 \times a \times b \times \cos(g)}$, где a, b – стороны треугольника; g – угол между сторонами a и b.

2. Площадь треугольника $s = \sqrt{p \times (p-a) \times (p-b) \times (p-c)}$, где a, b, c – стороны треугольника; p – полупериметр.

3. Радиус вписанной в прямоугольный треугольник окружности r = s/p, где s – площадь треугольника; p – полупериметр треугольника.

4. Площадь окружности $s = \pi r^2$, длина окружности $l = 2\pi r$, где r – радиус окружности.

Контрольные вопросы к защите лабораторной работы

1. Какую структуру имеет программа на языке *C#*?
2. Верно ли, что в программе, написанной на языке *C#*, надо описывать все используемые в ней переменные?
3. Какие существуют варианты описания целых типов на языке *C#*?
4. Какие существуют варианты описания вещественных типов на языке *C#*?
5. Каким образом описать на языке *C#* переменную символьного типа?
6. Каким образом описать на языке *C#* л переменную огического типа?
7. Запишите на языке *C#* следующие числа:

   $-25{,}8*10^{-7}$; 10^6; $0{,}5*10^6$; 7,48; 2; 4/1000.

8. Запишите следующие числа без десятичного порядка:

   -0.00027E+4; 759E-3; 1E1.

9. Запишите на языке ***C#*** следующие формулы:

   $a+bx+cyz$; $(1+x)^2$; (1+x)0.5;
   $\cos 3x^2$; $|a+bx|$; sin 8; $\log_2 0.4x$;
   arcctg 10^3; tg x; arcsin x; x^5; ln (x+3);
   $x\sqrt{2}$; $\sqrt[3]{|x|}$; x^{-2}; $e^{|x-y|}$; ln (1+3.3x).

10. Вычислите значения выражения:

    ceil(6.3); max(2.3,9.6); pow(3,2);
    floor(2.7); min(0,5); sqrt(81).

11. Вычислите значение выражений:

    20 / 5; 20 % 7;
    2 / 5; 2 % 7.

12. Укажите порядок выполнения операций в выражении

    a % b + a /b*c/a;

13. Какое значение будет иметь переменная x после выполнения операторов? x=10; x+=3;

14. Какое значение будет иметь переменная x после выполнения операторов? y=3; x=++y * 3; x=y++ * 2;

15. Какое значение будет иметь переменная y после выполнения операторов? x=5; y =--x * 7; y = x -- + 3;

16. С помощью каких средств языка *C#* можно осуществить ввод данных в приложениии *Windows Forms*?

17. С помощью каких средств языка *C#* можно осуществить вывод данных в приложениии ***Windows Forms****?*

18. Каким образом можно отредактировать форму, предложенную *C#* с тем, чтобы создать свой проект?

19. С помощью каких средств языка *C#* можно осуществить ввод данных в консольном приложении?

20. С помощью каких средств языка *C#* можно осуществить вывод данных в консольном приложении?

Содержание отчета

Отчет по лабораторной работе включает:

- титульный лист;
- условие задания;
- алгоритм решения задачи;
- текст программы;
- результаты тестирования программы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Подбельский В.В. Язык C#. Базовый курс. [Текст]/ В.В. Подбельский. – М.: Финансы и статистика, 2013. – 427 с.

2. Троелсен, Эндрю. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е изд. [Текст]/ Эндрю Троелсен – М.: ООО «И.Д. Вильямс», 2013. – 1312 с.

3. Макконнелл, Стив. Совершенный код. Мастер-класс. [Текст]/ Пер. с англ. – М. : Издательство «Русская редакция», 2010. – 896 с.

4. Техническая документация, материалы по API и примеры кода [Электронный ресурс]// Режим доступа – https://docs.microsoft.com (дата обращения: 26.05.2019).

5. METANIT.COM – Сайт о программировании [Электронный ресурс]// Режим доступа – https://metanit.com (дата обращения: 6.06.2019).

6. MSDN – сеть разработчиков Microsoft [Электронный ресурс]// Режим доступа – https://msdn.microsoft.com (дата обращения: 27.05.2019).

7. Шилдт, Г. C# 4.0: полное руководство. Пер. с англ. [Текст]/ Герберт Шилдт. – М.: ООО "И.Д. Вильяме", 2011. - 1056 с.

8. Википедия – свободная энциклопедия [Электронный ресурс]// Режим доступа – https://ru.wikipedia.org/wiki/Заглавная_страница (дата обращения: 24.05.2019).