МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**ЛАБОРАТОРНА РОБОТА №4**

з курсу «**Конструювання ПЗ**»

на тему: «**СЕМАНТИЧЕСКАЯ ОБРАБОТКА ВЫРАЖЕНИЙ**»

Виконав: студент 3 курсу групи № 631п

спеціальності

6.050103 «Програмна інженерія»

(шифр і назва напряму підготовки (спеціальності))

Кузьмич М. І.

(прізвище й ініціали студента)

Прийняв: к.т.н., доц. Волобуєва Л. О.

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала:

Кількість балів:

Оцінка ECTS:

Харків – 2018

**Цель работы:** реализация семантической обработки распознаваемых конструкций.

**Теоретическое введение**

**Синтаксический LL-анализатор** (**LL parser**) — в информатике нисходящий синтаксический анализатор для некоторого подмножества контекстно-свободных грамматик, известных как LL-грамматики. При этом не все контекстно-свободные грамматики являются LL-грамматиками.

Буквы L в выражении «LL-анализатор» означают, что входная строка анализируется слева направо (left to right), и при этом строится её левосторонний вывод (leftmost derivation).

LL-анализатор называется LL(k)-анализатором, если данный анализатор использует предпросмотр на k токенов (лексем) при разборе входного потока. Грамматика, которая может быть распознана LL(k)-анализатором без возвратов к предыдущим символам, называется LL(k)-грамматикой. Язык, который может быть представлен в виде LL(k)-грамматики, называется LL(k)o-языком. Существуют LL(k+n)-языки, которые не являются LL(k)-языками.

Далее описывается анализатор, в основе которого лежит построение таблиц; альтернативой может служить анализатор, построенный методом рекурсивного спуска, который обычно пишется вручную (хотя существуют и исключения, например, генератор синтаксических анализаторов [ANTLR](https://ru.wikipedia.org/wiki/ANTLR) для LL(\*) грамматик).

LL(1)-грамматики очень распространены, потому что соответствующие им LL-анализаторы просматривают поток только на один символ вперед при принятии решения о том, какое правило грамматики необходимо применить. Языки, основанные на грамматиках с большим значением k, традиционно считались трудными для распознавания, хотя при широком распространении генераторов синтаксических анализаторов, поддерживающих LL(k) грамматики с произвольным k, это замечание уже неактуально.

LL-анализатор называется LL(\*)-анализатором, если нет строгого ограничения для k и анализатор может распознавать язык, если токены принадлежат какому-либо регулярному множеству (например, используя детерминированные конечные автоматы).

Существуют противоречия между так называемой «Европейской школой» построения языков, которая основывается на LL-грамматиках, и «Американской школой», которая предпочитает LR-грамматики. Такие противоречия обусловлены традициями преподавания и деталями описания различных методов и инструментов в конкретных учебниках; кроме того, своё влияние оказал [Н. Вирт](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82,_%D0%9D%D0%B8%D0%BA%D0%BB%D0%B0%D1%83%D1%81) из [ETHZ](https://ru.wikipedia.org/wiki/%D0%A8%D0%B2%D0%B5%D0%B9%D1%86%D0%B0%D1%80%D1%81%D0%BA%D0%B0%D1%8F_%D0%B2%D1%8B%D1%81%D1%88%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%88%D0%BA%D0%BE%D0%BB%D0%B0_%D0%A6%D1%8E%D1%80%D0%B8%D1%85%D0%B0), чьи исследования описывают различные методы оптимизации LL(1) распознавателей и компиляторов.

**LL – грамматики**

LL(k) – грамматикой называется КС – грамматика, в которой выбор правила в ходу левостороннего вывода однозначно определяется не более чем k очередными символами входной цепочки, считываемой слева направо.

Название «LL» происходит от двух слов left: левосторонний вывод при чтении слева. Самыми удобными для распознавания являются LL(1) грамматики, в которых выбор направления распознавания однозначно определяется очередным входным символом.

Рекурсивный спуск – это детерминированный метод нисходящего разбора КС – языков, порождаемый LL(1) – грамматиками.

Если в грамматике G существует нетерминал А, для которого , где  - непустая цепочка, грамматика G содержит **левую рекурсию**. Грамматика, содержащая левую рекурсию не может быть LL(1) грамматикой.

**Включение действий в синтаксис**

На основе распознавания структуры входного текста строится и его содержательная обработка, трансляция. Синтаксический анализатор служит основой транслятора, предоставляя возможность выполнить необходимые действия по смысловой (семантической) обработке в нужные моменты в соответствии со структурой входной цепочки.

Встраиваемые в распознаватель действия, предназначенные для выполнения смысловой обработки входного текста, называют **семантическими процедурами.**

Рассмотрим использование СП на примере распознавания целых чисел без знака. Обозначим семантические процедуры через  и  и разметим их на синтаксической диаграмме

Целое:



Процедура  будет выполняться в начале обработки и присваивать : у:=0 некому числу 0. переменная у – это формируемое значение числа.

Процедура  будет добавлять считанный символ (цифру) d к текущему значению у.

: у := 10 \* у + 

**Постановка задачи**

1. Дополнить каждую синтаксическую диаграмму, построенную в лабораторной работе №1 семантическими процедурами в соответствии с приведенными вариантами.
2. Добавить в распознающие процедуры, построенные в лабораторной работе №2, семантическую обработку.
3. В синтаксические таблицы, построенные в лабораторной работе №3 добавить столбец «процедура». Для каждого состояния в столбце «процедура» указать номер вызываемой семантической процедуры, соответственной модифицированным синтаксическим диаграммам (см. п.1).
4. Реализовать альтернативные варианты распознавания и обработки конструкций на основе построенных таблиц и распознающих процедур. Разработать программное обеспечение распознавания заданных конструкций. Анализатор должен либо выдавать сообщение об ошибке с указанием места ее обнаружения; либо сообщать о том, что конструкция записана верно, выводить на экран саму конструкцию и результат ее семантической обработки.

**Вариант №11**

Вычислить коэффициенты плоскости, перпендикулярной заданной и проходящей через ось *Ох*.

Описание построенной грамматики

1. – 2 тип
2. – 3 тип
3. – 2 тип
4. – 2 тип
5. – 3 тип
6. – 3 тип

Общая грамматика 2-го типа.

**Синтаксическая таблица для заданной конструкции**

Синтаксическая таблица представлена Таблицей 1.

Таблица 1 – Синтаксическая таблица

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Состояние** | **Символ** | **Переход** | **Ошибка** | **Читать** | **Сообщение ошибки** | **Процедура** |
| 1 | – | 2 | – | + | – |  |
| 2 | Цифра | 3 | + | – | “Ожидался коэффициент” |  |
| 3 | 0 | 5 | – | + | – |  |
| 4 | Цифра | 4 | – | + | – |  |
| 5 | ‘x’ | 6 | + | + | “Ожидался ‘x’” |  |
| 6 | + | 8 | – | + | – |  |
| 7 | – | 7 | + | + | “Ожидался ‘+’ или ‘–’” |  |
| 8 | Цифра | 9 | + | – | – |  |
| 9 | 0 | 11 | – | + | – |  |
| 10 | Цифра | 10 | – | + | – |  |
| 11 | ‘y’ | 12 | + | + | “Ожидался ‘y’” |  |
| 12 | + | 14 | + | + |  |  |
| 13 | – | 14 | + | + | “Ожидался ‘+’ или ‘–’” |  |
| 14 | Цифра | 15 | + | – | – |  |
| 15 | 0 | 17 | – | + | – |  |
| 16 | Цифра | 16 | – | + | – |  |
| 17 | ‘z’ | 18 | + | + | “Ожидался ‘z’” |  |
| 18 | + | 23 | – | + | – |  |
| 19 | – | 23 | – | + | – |  |
| 20 | = | 21 | + | + | “Ожидался ‘=’” |  |
| 21 | 0 | 22 | + | – | “Ожидался ‘0’” |  |
| 22 | Любой | 0 | – | – | – |  |
| 23 | Цифра | 24 | + | – | “Ожидался коэффициент” |  |
| 24 | 0 | 20 | – | + | – |  |
| 25 | Цифра | 25 | – | + | – |  |
| 26 | Любой | 20 | – | – | – |  |

**Синтаксические диаграммы**

На рисунках 1-6 изображены синтаксические диаграммы правил грамматики.



Рисунок 1 – Синтаксическая диаграмма нетерминала S

P1: s = 0, P2: s = -KxZKyZKzZK=0, P3: s = KxZKyZKzZK=0



Рисунок 2 – Синтаксическая диаграмма нетерминала Z

P4: Z = 0, P5: Z = - , P6: Z = +



Рисунок 3 – Синтаксическая диаграмма нетерминала K

P7: K = 0, P8: K = C, P9: K = C1(H.Count +1) +H



Рисунок 4 – Синтаксическая диаграмма нетерминала H

P10: H = 0, P11: H += H\*10+C



Рисунок 5 – Синтаксическая диаграмма нетерминала C

P12: C = 0, P13: C = d



Рисунок 6 – Синтаксическая диаграмма нетерминала C1

P14: C1 = 0, P15: C1 = 9

**Экранные формы работы ПО синтаксического анализатора**

Демонстрация работы программы представлена на рисунках 7-10.

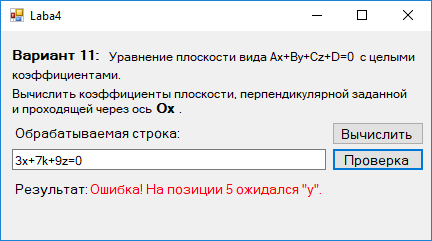


Рисунок 7 – Тестовый пример №1

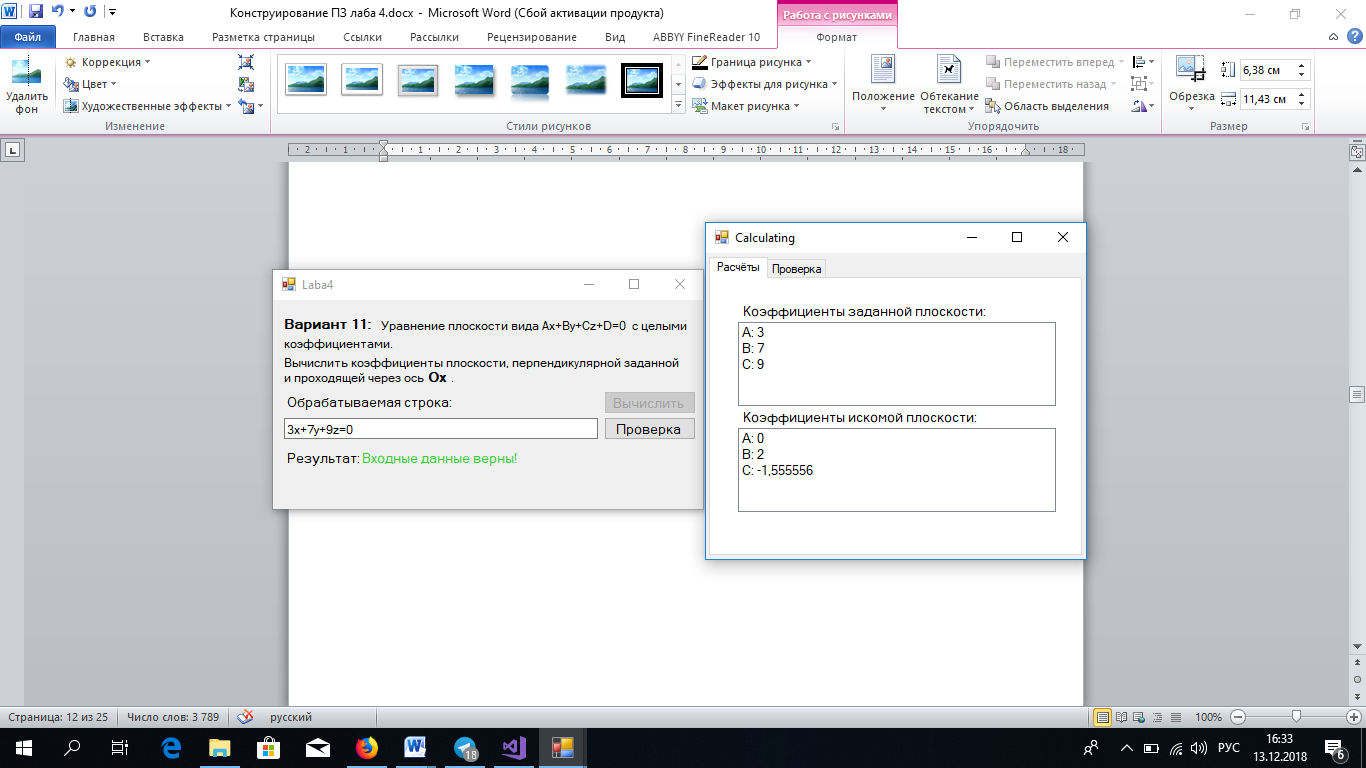


Рисунок 8 – Тестовый пример №2

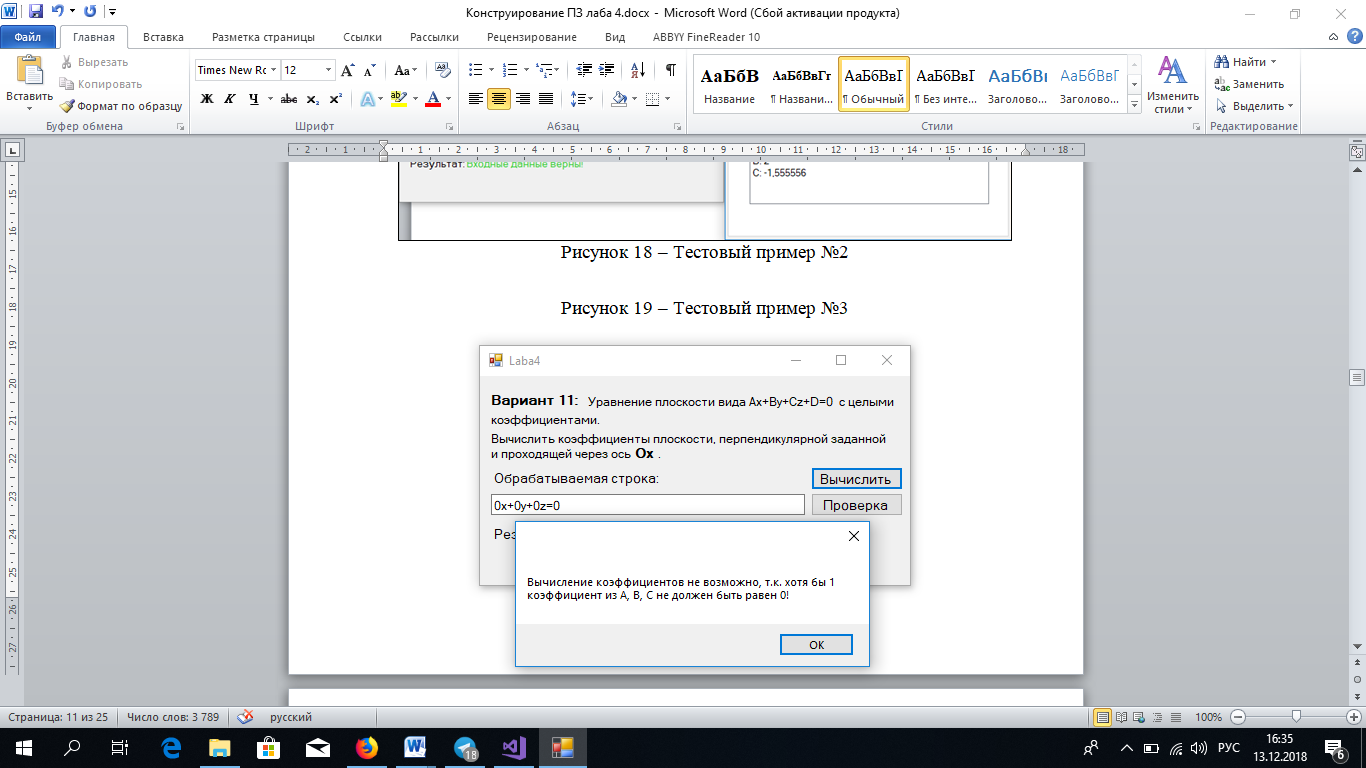


Рисунок 9 – Тестовый пример №3

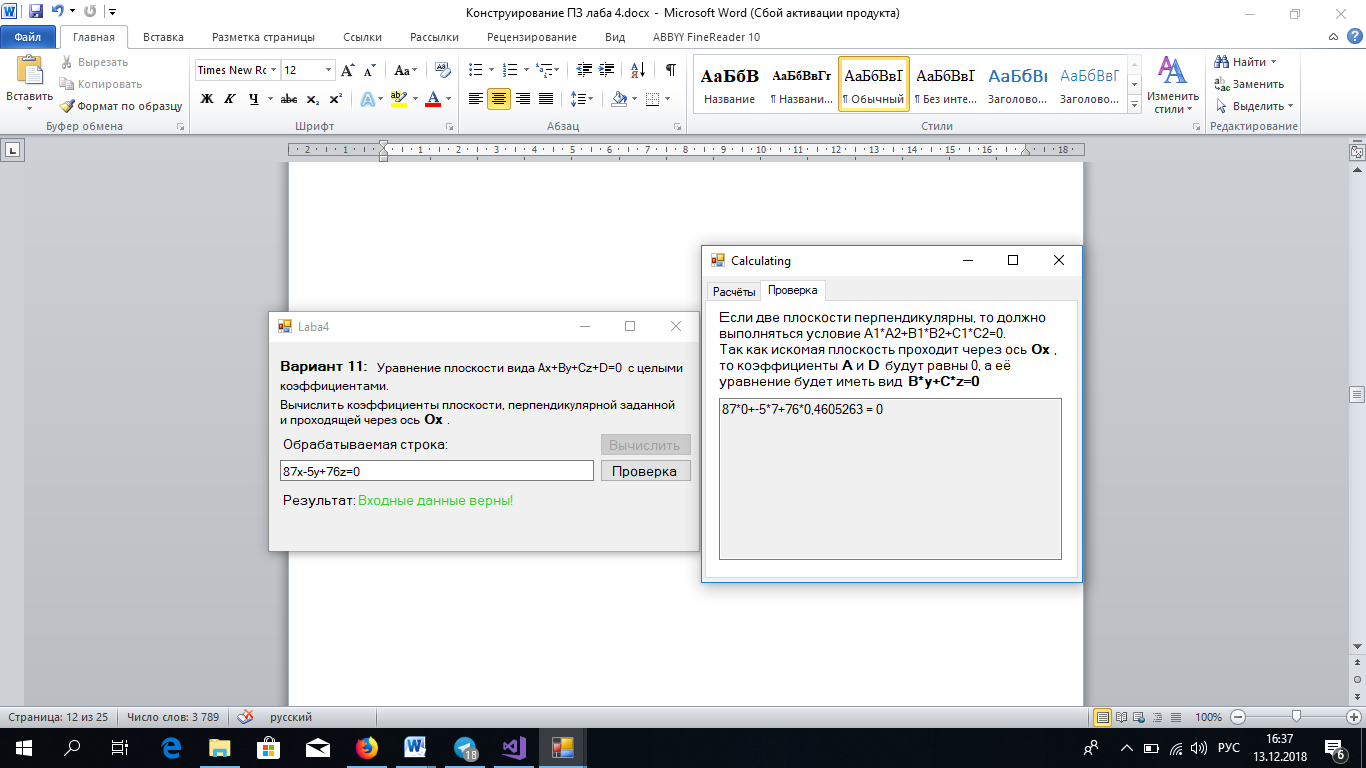


Рисунок 10 – Тестовый пример №4

**Вывод**

В ходе выполнения данной лабораторной работы было разработано приложение, которое выполняет семантическую обработку конструкций, разработанных в лабораторной работе №1. Также приложение предусматривает возможность вычисления коэффициентов плоскости, перпендикулярной заданной и проходящей через ось *Ох.*

**ПРИЛОЖЕНИЕ А**

Листинг программы

**Класс Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ConstruirovanieLaba3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

btn\_Calc.Enabled = false;

}

public float[] coefficients;

string[] str;

private void btn\_check\_Click(object sender, EventArgs e)

{

Table t = new Table(label\_result);

t.Check(inp\_str\_text.Text);

str = inp\_str\_text.Text.Split('x', 'y', 'z', '=', '+');

btn\_Calc.Enabled = true;

}

private void btn\_Calc\_Click(object sender, EventArgs e)

{

int index = 0;

for (int i = 0; i < str.Length - 1; i++)

{

if (str[i] == "") index++;

}

coefficients = new float[str.Length - index - 1];

int j = 0;

for (int i = 0; i < str.Length - 1; i++)

{

if (str[i] == "") continue;

else

{

coefficients[j] = Int32.Parse(str[i]);

j++;

}

}

int kol = 0;

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

{

if (coefficients[i] == 0) kol++;

}

if(kol==coefficients.Length)

{

MessageBox.Show("Вычисление коэффициентов не возможно, т.к. хотя бы 1 коэффициент из A, B, C не должен быть равен 0!");

btn\_Calc.Enabled = false;

return;

}

Form2 f = new Form2();

f.Show();

btn\_Calc.Enabled = false;

}

}

}

**Класс Row.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConstruirovanieLaba3

{

class Row

{

public int State { get; set; }

public string Symbol { get; set; }

public int Transition { get; set; }

public bool Error { get; set; }

public bool Read { get; set; }

public string Exception { get; set; }

public Row(int State, string Symbol, int Transition, bool Error, bool Read, string Exception)

{

this.State = State;

this.Symbol = Symbol;

this.Transition = Transition;

this.Error = Error;

this.Read = Read;

this.Exception = Exception;

}

}

}

**Класс Table.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ConstruirovanieLaba3

{

class Table

{

public List<Row> table;

private Label label;

public Table(Label label)

{

table = new List<Row>

{

new Row(1,"-",2,false,true,""),

new Row(2, "Number", 3, true, false,"ожидался коэффициент."),

new Row(3, "0", 5, false, true,""),

new Row(4, "Number", 4, false, true,""),

new Row(5, "x", 6, true, true,"ожидался \"x\"."),

new Row(6, "+", 8, false, true,""),

new Row(7, "-",8,true,true,"ожидался \"+\" или \"-\"."),

new Row(8, "Number", 9, true, false,"ожидался коэффициент."),

new Row(9, "0", 11, false, true,""),

new Row(10, "Number", 10, false, true,""),

new Row(11, "y", 12, true, true,"ожидался \"y\"."),

new Row(12, "+", 14, false, true,""),

new Row(13, "-",14, true, true,"ожидался \"+\" или \"-\"."),

new Row(14, "Number", 15, true, false,"ожидался коэффициент."),

new Row(15, "0", 17, false, true,""),

new Row(16, "Number", 16, false, true,""),

new Row(17, "z", 18, true, true,"ожидался \"z\"."),

new Row(18, "+",23,false,true,""),

new Row(19, "-", 23, false, true,""),

new Row(20, "=", 21, true, true,"ожидалось \"=\"."),

new Row(21, "0", 22, true, false,"ожидался \"0\"."),

new Row(22, "Any", 0, false, false,""),

new Row(23, "Number", 24, true, false,"ожидался коэффициент."),

new Row(24, "0", 20, false, true,""),

new Row(25, "Number", 25, false, true,""),

new Row(26, "Any",20, false, false,"")

};

this.label = label;

this.label.Visible = false;

}

public void Check(string str)

{

str.Trim();

str = str.Replace(" ", "");

str += ";";

int i = 0, j = 0;

while (true)

{

if (table[i].Transition == 0)

{

break;

}

char k = str[j];

switch (table[i].Symbol.Length)

{

case 1:

if (k.ToString() == table[i].Symbol)

{

Processing(ref i, ref j);

continue;

}

if (table[i].Error)

{

ErrorShow(i, j, table[i].Exception);

return;

}

break;

case 3:

i = table[i].Transition - 1;

continue;

case 6:

if (Char.IsDigit(k))

{

Processing(ref i, ref j);

continue;

}

if (table[i].Error)

{

ErrorShow(i, j, table[i].Exception);

return;

}

break;

default:

break;

}

i++;

}

if (j == str.Length - 2)

{

label.Visible = true;

label.ForeColor = Color.LimeGreen;

label.Text = "Входные данные верны!";

}

else

{

label.Visible = true;

label.ForeColor = Color.Red;

label.Text = "Ошибка! Правая часть выражения содержит недопустимый символ на позиции " + (j + 2) + ".";

}

}

private void ErrorShow(int i, int j, string message)

{

label.Visible = true;

label.ForeColor = Color.Red;

label.Text = "Ошибка! На позиции " + (j + 1) + " " + message;

}

private void Processing(ref int i, ref int j)

{

if (table[i].Read)

{

j++;

}

i = table[i].Transition - 1;

}

}

}

**Класс Form2.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ConstruirovanieLaba3

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

string[] coef\_str = new string[] { "A", "B", "C", "D" };

float[] coefficients = new float[4];

float res;

private void Form2\_Load(object sender, EventArgs e)

{

Output(lb\_plane1, (Application.OpenForms[0] as Form1).coefficients);

coefficients = Calculate((Application.OpenForms[0] as Form1).coefficients);

Output(lb\_plane2, coefficients);

res = Check\_res(ref (Application.OpenForms[0] as Form1).coefficients, ref coefficients);

if ((Application.OpenForms[0] as Form1).coefficients.Length == 3)

check\_tb.Text = $"{(Application.OpenForms[0] as Form1).coefficients[0]}\*{coefficients[0]}+{(Application.OpenForms[0] as Form1).coefficients[1]}\*{coefficients[1]}+{(Application.OpenForms[0] as Form1).coefficients[2]}\*{coefficients[2]} = {(int)res}";

else check\_tb.Text = $"{(Application.OpenForms[0] as Form1).coefficients[0]}\*{coefficients[0]}+{(Application.OpenForms[0] as Form1).coefficients[1]}\*{coefficients[1]}+{(Application.OpenForms[0] as Form1).coefficients[2]}\*{coefficients[2]}+{(Application.OpenForms[0] as Form1).coefficients[3]}\*{coefficients[3]} = {(int)res}";

}

float Check\_res(ref float[] arr1, ref float[] arr2)

{

float res = 0;

for (int i = 0; i < arr1.Length; i++)

{

res += arr1[i] \* arr2[i];

}

return res;

}

private void Output(ListBox lb, float[] arr)

{

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

{

lb.Items.Add(coef\_str[i] + ": " + arr[i]);

}

}

private float[] Calculate(float[] arr)

{

float[] coefficients\_1 = new float[arr.Length];

Random m = new Random();

coefficients\_1[1] = m.Next(1, 10);

coefficients\_1[2] = (-coefficients\_1[1] \* arr[1]) / arr[2];

return coefficients\_1;

}

}

}