МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ  
  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»  
  
Институт информатики и кибернетики  
Кафедра информационных систем и технологий **ОТЧЕТ**по лабораторной работе

по курсу  
 **Теория автоматов и формальных языков**  
на тему

**Синтаксический анализ автоматных грамматик**

Вариант № 8

Студент: Немченко Д.И.

Группа: 6203-020302D  
  
Преподаватель: Чигарина Е.И.

Самара 2023

**СОДЕРЖАНИЕ**

[1 Задание 3](#_Toc481088979)

[2 Графы состояний 5](#_Toc481088980)

[3 Интерфейс работы программы 6](#_Toc481088985)

[4 Листинг модулей программы 27](#_Toc481088986)

1 ЗАДАНИЕ

**Вариант 8.**

Написать программу синтаксического анализа автоматного языка операторов выбора языка Turbo Pascal, имеющего вид:

**CASE** <идентификатор> **OF** <список выбора> [**ELSE** <оператор присваивания>] **END;**

<список выбора> :: = <константа выбора> [**,**<константа выбора>]**:**<оператор присваи-вания>**;**[<константа выбора>[**,**<константа выбора>]**:**<оператор присваивания>**;**]

<константа выбора> :: = ′<текст>′|<константа любая>

<оператор присваивания> :: = <левая часть> **:=** <правая часть>

<правая часть> :: = <операнд>[<операция><операнд>]

**⎡**⎧ <идентификатор>⎫**⎤**⏐

<операнд> :: = <идентификатор>|<идентификатор> **⎢**⎨ ⎬**⎥**⏐<константа

**⎣**⎩<целая константа>⎭**⎦**⏐ любая>

⎧<идентификатор> ⎫

<левая часть> :: = ⎨ **⎡**⎧<константа целая>⎫**⎤**⎬

⎪<идентификатор> **⎢**⎨ ⎬**⎥**⏐

⎩ **⎣**⎩<идентификатор> ⎭**⎦** ⎭

⎧ + ⎫

⎪ - ⎪

<операция> :: = ⎨ / ⎬

⎪ \* ⎪

⎪ div ⎪

⎩ mod ⎭

<идентификатор> - идентификатор языка Turbo Pascal, начинается с буквы или знака подчеркивания, включает буквы, цифры, подчеркивание, не допускает пробелы и специальные символы, ввести ограничение на длину (не более 8 символов) и не может быть зарезервированным словом (CASE, OF, ELSE, END, DIV, MOD);

<константа> - целое число в диапазоне -32768 ÷ +32767.

<константа любая> - целое число, число с фиксированной точкой, число с плавающей точкой.

Семантика:

Построить и вывести на печать таблицы идентификаторов и констант. Учесть перечисленные выше ограничения на идентификаторы и константы.

Сообщать об ошибках при анализе цепочек языка, указывая курсором место ошибки и ее содержание.

Не должно быть дублирования идентификаторов.

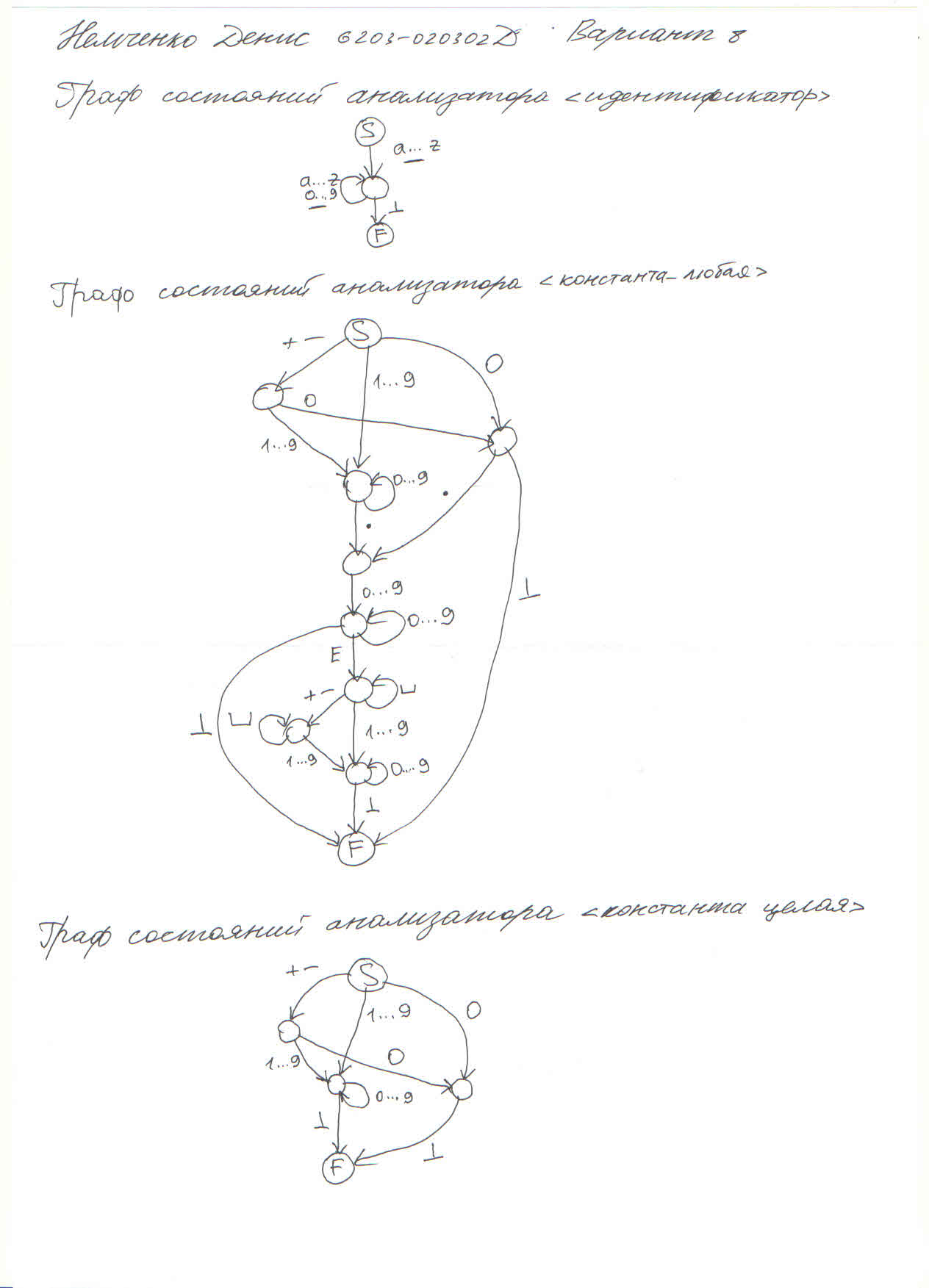
Примеры правильных цепочек:

case \_oper of ‘A’, ‘B’ : C := x + y; ‘L’ : Z := x / y; end;

case A\_B of 12 : A := 14.3 + C; 13 : B := 0.3; end;

case prim of 14.3 : A [-1] := \_A\_L; else C := 8.3E - 3; end;

2 ГРАФ СОСТОЯНИЙ

Рис 1 – «Граф состояний анализатора оператора Case Of»

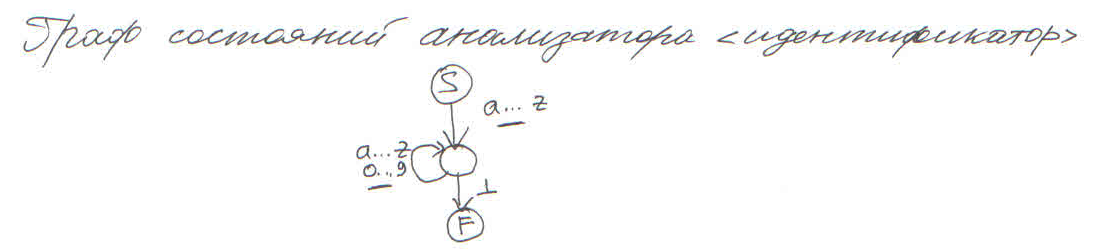


Рис 2 – «Граф состояний анализатора <идентификатор>»

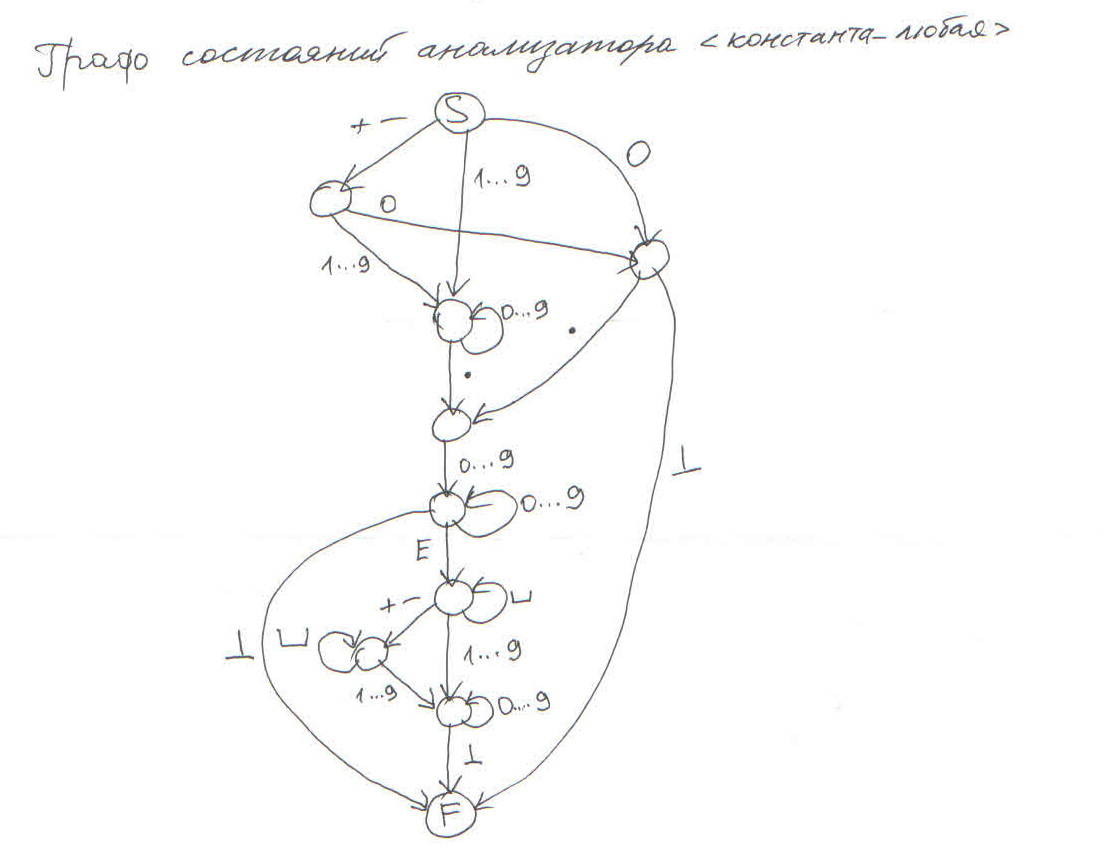


Рис 3 – «Граф состояний анализатора <константа любая>»

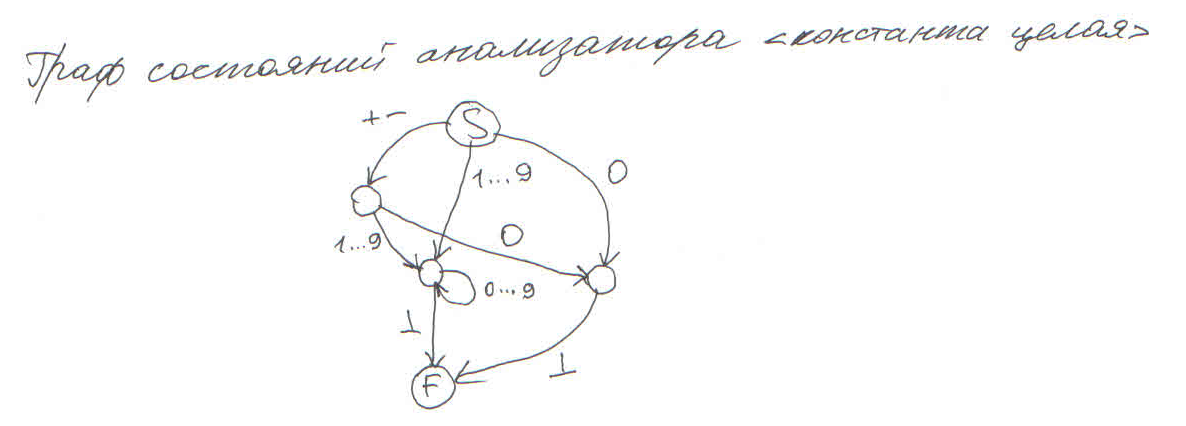


Рис 4 – «Граф состояний анализатора <константа целая>»

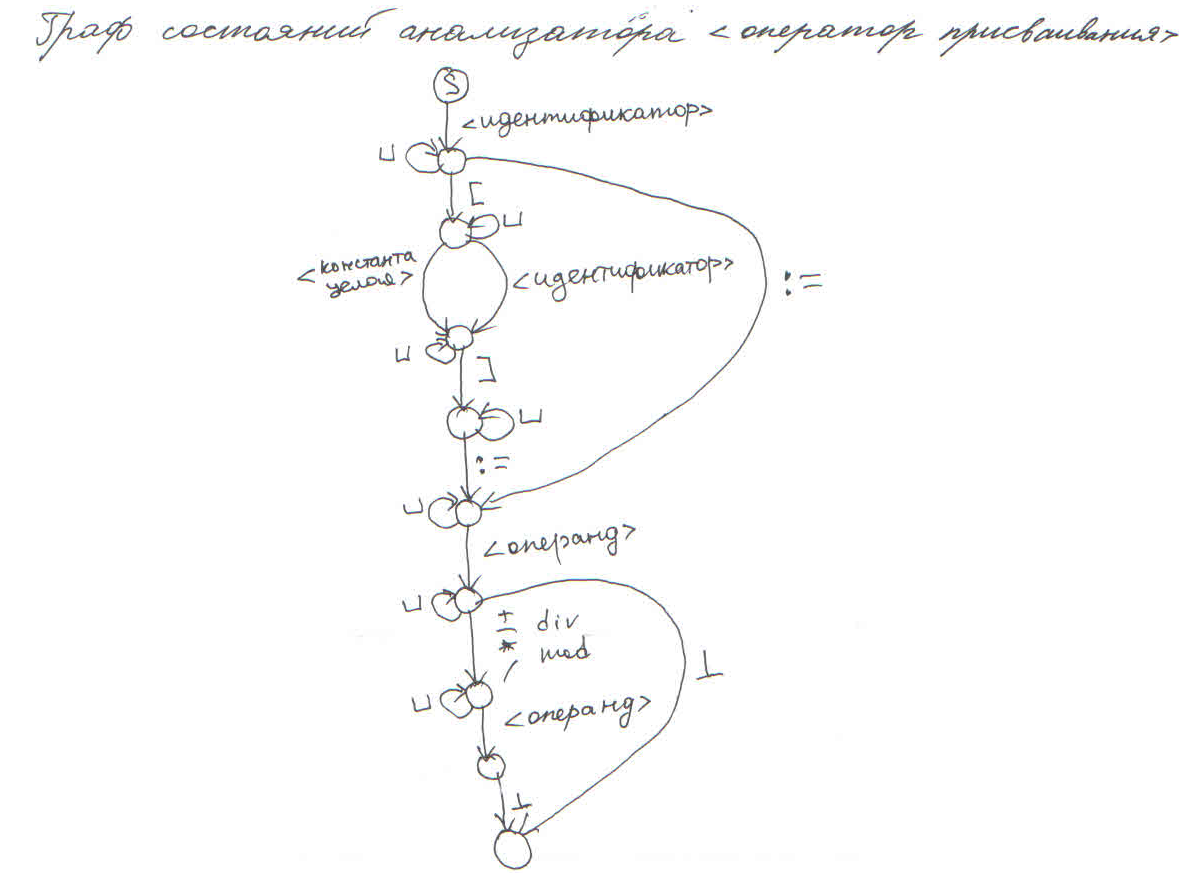


Рис 5 – «Граф состояний анализатора <оператор присваивания>»

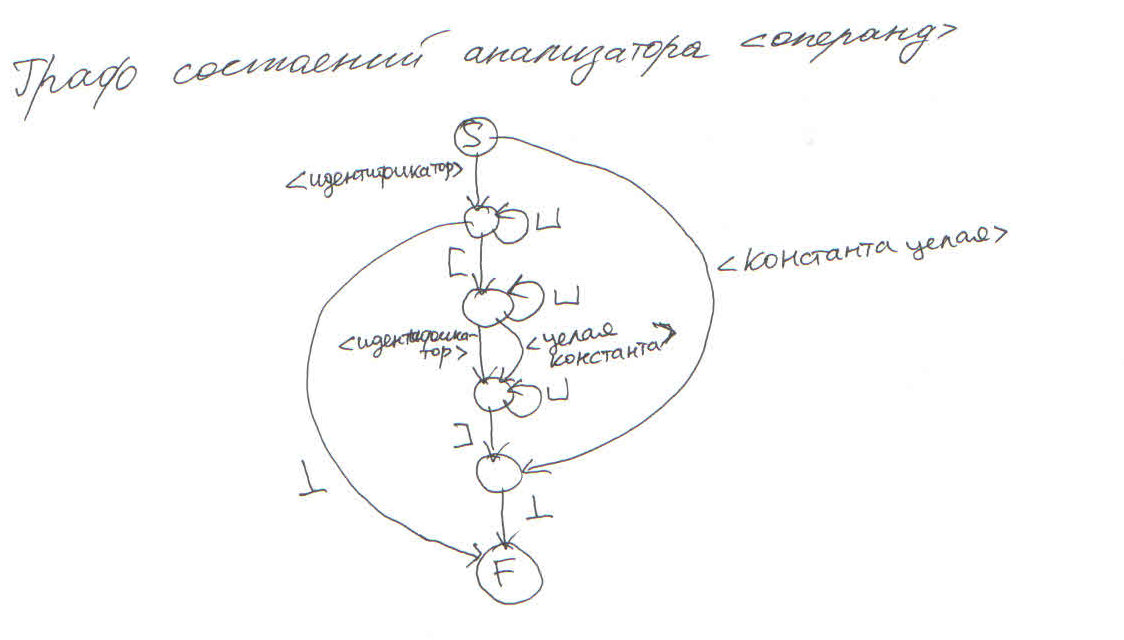


Рис 6 – «Граф состояний анализатора <операнд>»

# 3 Примеры работы

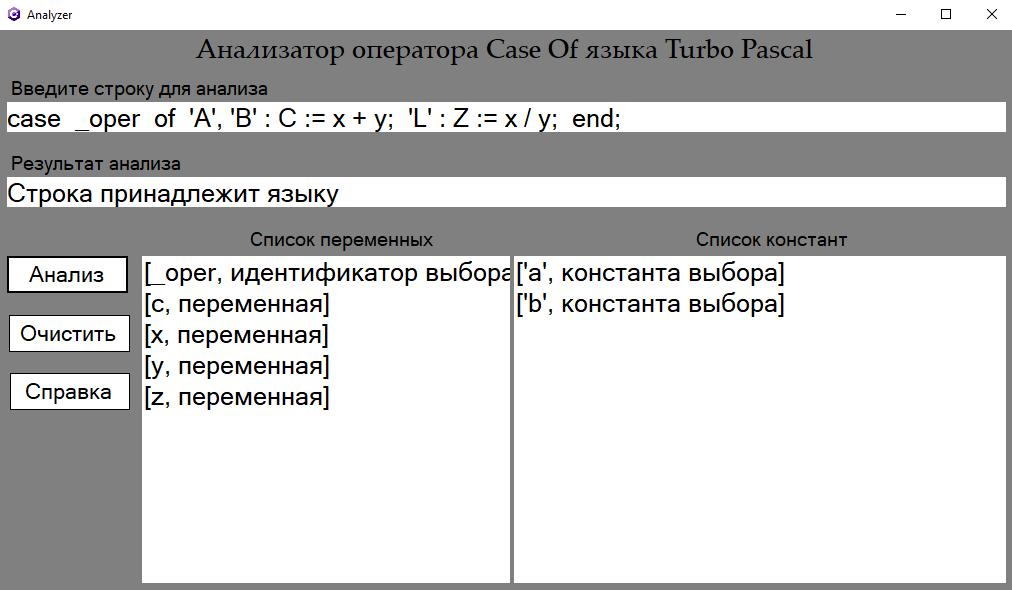


Рисунок 3.1 – Пример правильного выражения

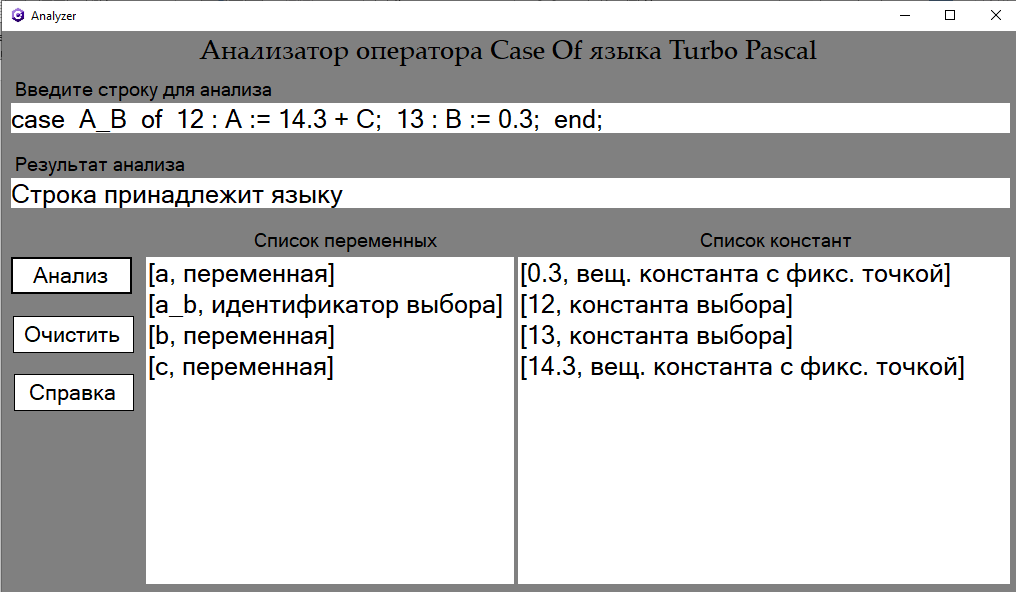


Рисунок 3.2 – Пример правильного выражения

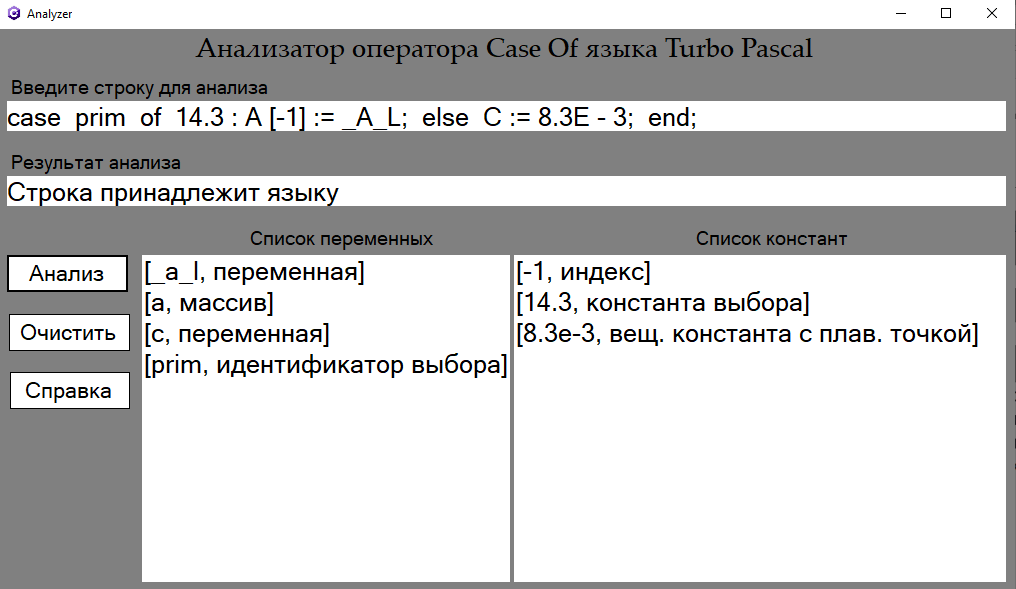


Рисунок 3.3 – Пример правильного выражения

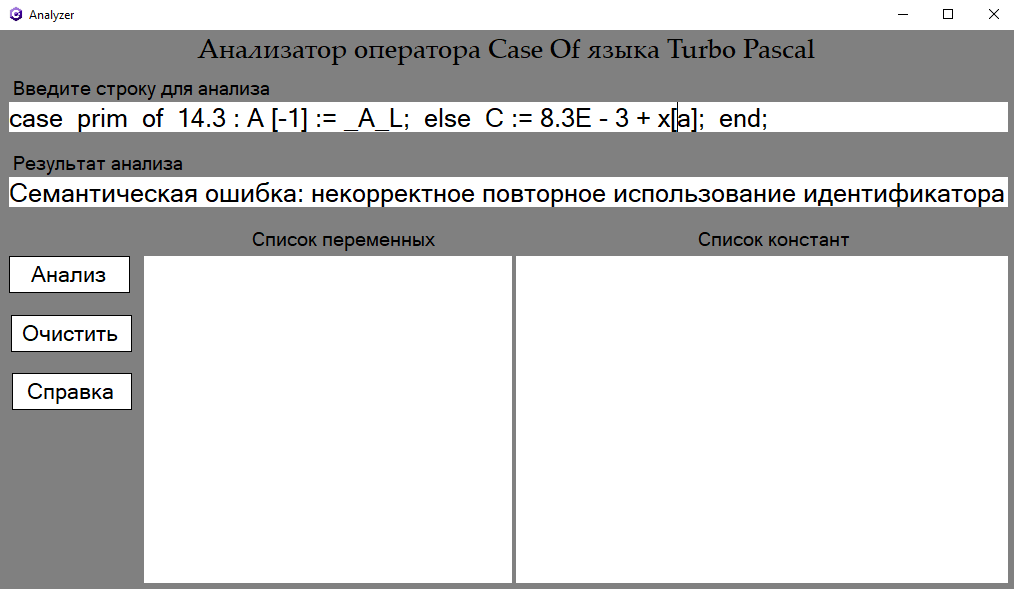


Рисунок 3.4 – Пример неправильного выражения

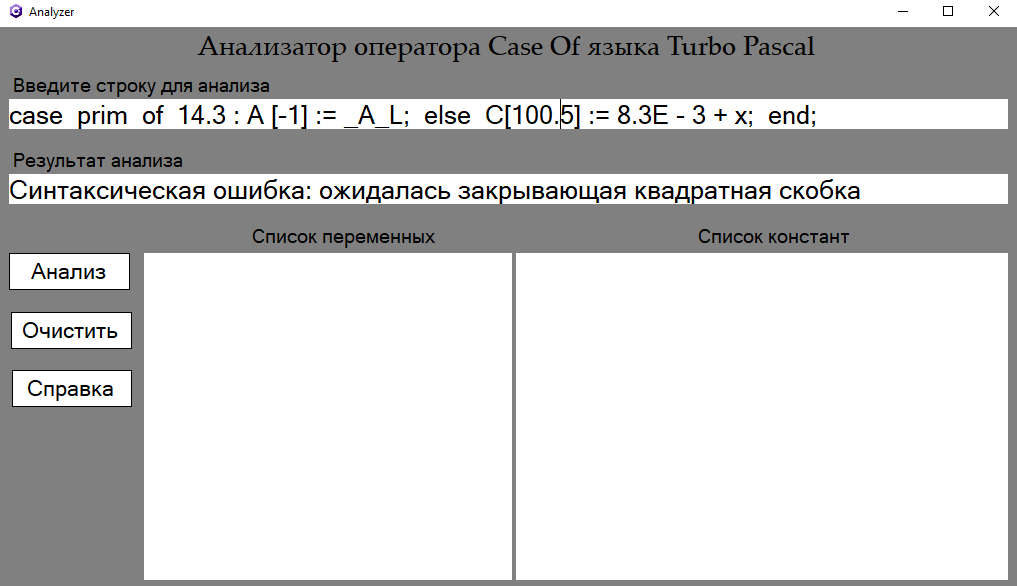


Рисунок 3.5 – Пример неправильного выражения

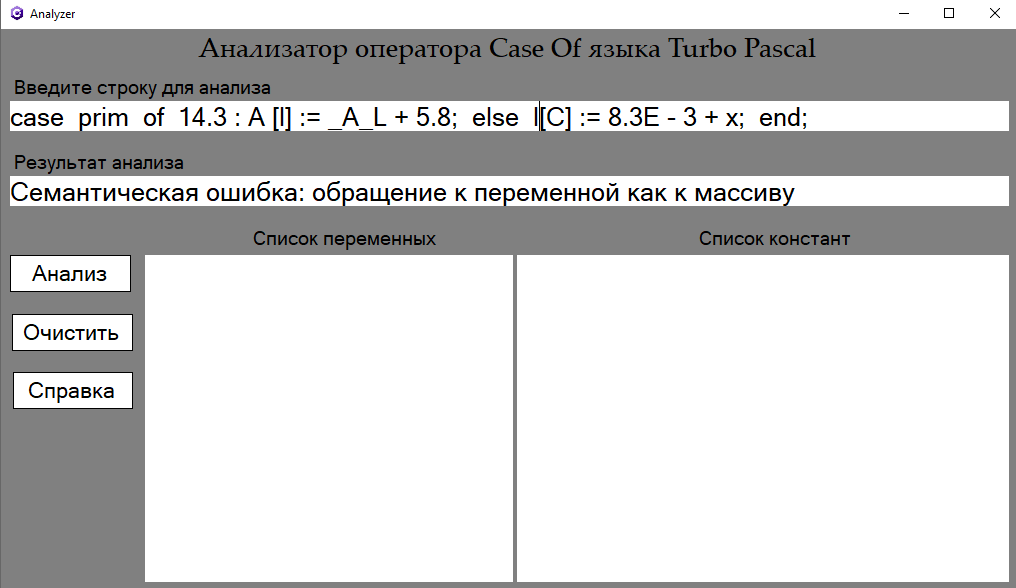


Рисунок 3.6 – Пример неправильного выражения

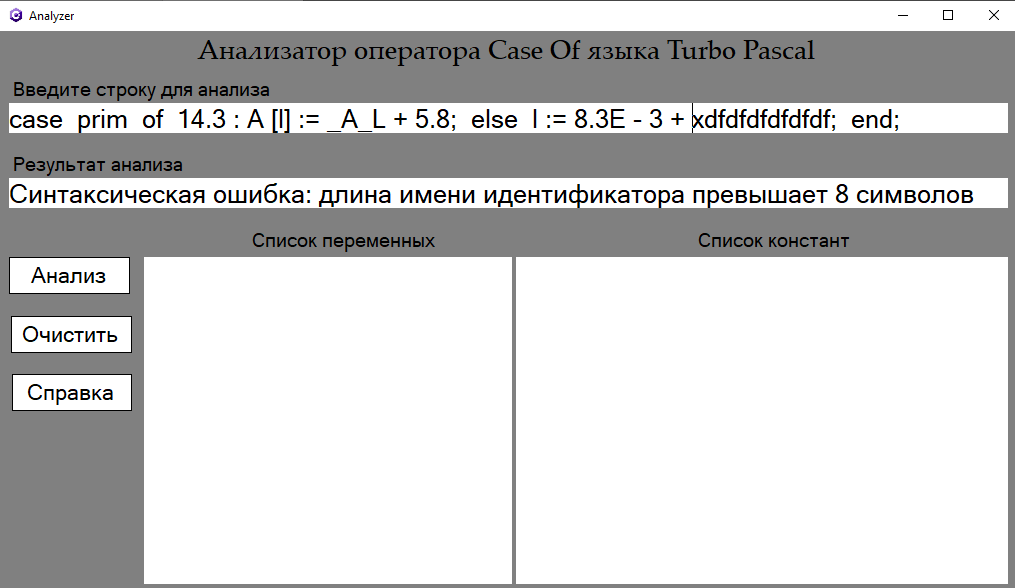


Рисунок 3.7 – Пример неправильного выражения

4 КОД ПРОГРАММЫ

Program.cs

using System;

using System.Windows.Forms;

namespace Analyzer

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

MainForm.Designer.cs

namespace Analyzer

{

partial class MainForm

{

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

private void InitializeComponent()

{

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(MainForm));

this.inputTextBox = new System.Windows.Forms.TextBox();

this.analyzeButton = new System.Windows.Forms.Button();

this.title = new System.Windows.Forms.Label();

this.inputTitle = new System.Windows.Forms.Label();

this.resultTextBox = new System.Windows.Forms.TextBox();

this.infoButton = new System.Windows.Forms.Button();

this.resultTitle = new System.Windows.Forms.Label();

this.clearButton = new System.Windows.Forms.Button();

this.variableListView = new System.Windows.Forms.ListView();

this.constantListView = new System.Windows.Forms.ListView();

this.variableTitle = new System.Windows.Forms.Label();

this.constantTitle = new System.Windows.Forms.Label();

this.SuspendLayout();

//

// inputTextBox

//

this.inputTextBox.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right)));

this.inputTextBox.BackColor = System.Drawing.Color.White;

this.inputTextBox.BorderStyle = System.Windows.Forms.BorderStyle.None;

this.inputTextBox.Font = new System.Drawing.Font("Microsoft Sans Serif", 19.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.inputTextBox.Location = new System.Drawing.Point(9, 72);

this.inputTextBox.Margin = new System.Windows.Forms.Padding(2);

this.inputTextBox.Name = "inputTextBox";

this.inputTextBox.Size = new System.Drawing.Size(999, 30);

this.inputTextBox.TabIndex = 0;

//

// analyzeButton

//

this.analyzeButton.Anchor = System.Windows.Forms.AnchorStyles.Top;

this.analyzeButton.BackColor = System.Drawing.Color.White;

this.analyzeButton.Cursor = System.Windows.Forms.Cursors.Hand;

this.analyzeButton.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.analyzeButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 16.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.analyzeButton.Location = new System.Drawing.Point(9, 226);

this.analyzeButton.Margin = new System.Windows.Forms.Padding(2);

this.analyzeButton.Name = "analyzeButton";

this.analyzeButton.Size = new System.Drawing.Size(121, 37);

this.analyzeButton.TabIndex = 2;

this.analyzeButton.Text = "Анализ";

this.analyzeButton.UseVisualStyleBackColor = false;

this.analyzeButton.Click += new System.EventHandler(this.analyzeButton\_Click);

//

// title

//

this.title.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right)));

this.title.BackColor = System.Drawing.Color.Gray;

this.title.Font = new System.Drawing.Font("Palatino Linotype", 19.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.title.Location = new System.Drawing.Point(0, 0);

this.title.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.title.Name = "title";

this.title.Size = new System.Drawing.Size(1017, 37);

this.title.TabIndex = 3;

this.title.Text = "Анализатор оператора Case Of языка Turbo Pascal";

this.title.TextAlign = System.Drawing.ContentAlignment.TopCenter;

//

// inputTitle

//

this.inputTitle.AutoSize = true;

this.inputTitle.Font = new System.Drawing.Font("Microsoft Sans Serif", 13.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.inputTitle.Location = new System.Drawing.Point(9, 46);

this.inputTitle.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.inputTitle.Name = "inputTitle";

this.inputTitle.Size = new System.Drawing.Size(267, 24);

this.inputTitle.TabIndex = 4;

this.inputTitle.Text = "Введите строку для анализа";

//

// resultTextBox

//

this.resultTextBox.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right)));

this.resultTextBox.BackColor = System.Drawing.Color.White;

this.resultTextBox.BorderStyle = System.Windows.Forms.BorderStyle.None;

this.resultTextBox.Font = new System.Drawing.Font("Microsoft Sans Serif", 19.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.resultTextBox.Location = new System.Drawing.Point(9, 147);

this.resultTextBox.Margin = new System.Windows.Forms.Padding(2);

this.resultTextBox.Name = "resultTextBox";

this.resultTextBox.ReadOnly = true;

this.resultTextBox.Size = new System.Drawing.Size(999, 30);

this.resultTextBox.TabIndex = 5;

//

// infoButton

//

this.infoButton.Anchor = System.Windows.Forms.AnchorStyles.Top;

this.infoButton.BackColor = System.Drawing.Color.White;

this.infoButton.Cursor = System.Windows.Forms.Cursors.Hand;

this.infoButton.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.infoButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 16.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.infoButton.Location = new System.Drawing.Point(12, 343);

this.infoButton.Margin = new System.Windows.Forms.Padding(2);

this.infoButton.Name = "infoButton";

this.infoButton.Size = new System.Drawing.Size(120, 37);

this.infoButton.TabIndex = 6;

this.infoButton.Text = "Справка";

this.infoButton.UseVisualStyleBackColor = false;

this.infoButton.Click += new System.EventHandler(this.infoButton\_Click);

//

// resultTitle

//

this.resultTitle.AutoSize = true;

this.resultTitle.Font = new System.Drawing.Font("Microsoft Sans Serif", 13.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.resultTitle.Location = new System.Drawing.Point(9, 121);

this.resultTitle.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.resultTitle.Name = "resultTitle";

this.resultTitle.Size = new System.Drawing.Size(180, 24);

this.resultTitle.TabIndex = 7;

this.resultTitle.Text = "Результат анализа";

//

// clearButton

//

this.clearButton.Anchor = System.Windows.Forms.AnchorStyles.Top;

this.clearButton.BackColor = System.Drawing.Color.White;

this.clearButton.Cursor = System.Windows.Forms.Cursors.Hand;

this.clearButton.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.clearButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 16.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.clearButton.Location = new System.Drawing.Point(11, 285);

this.clearButton.Margin = new System.Windows.Forms.Padding(2);

this.clearButton.Name = "clearButton";

this.clearButton.Size = new System.Drawing.Size(121, 37);

this.clearButton.TabIndex = 8;

this.clearButton.Text = "Очистить строки";

this.clearButton.UseVisualStyleBackColor = false;

this.clearButton.Click += new System.EventHandler(this.clearButton\_Click);

//

// variableListView

//

this.variableListView.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

| System.Windows.Forms.AnchorStyles.Left)));

this.variableListView.BackColor = System.Drawing.Color.White;

this.variableListView.BorderStyle = System.Windows.Forms.BorderStyle.None;

this.variableListView.Font = new System.Drawing.Font("Microsoft Sans Serif", 19.8F);

this.variableListView.HideSelection = false;

this.variableListView.Location = new System.Drawing.Point(144, 226);

this.variableListView.Margin = new System.Windows.Forms.Padding(2);

this.variableListView.Name = "variableListView";

this.variableListView.Size = new System.Drawing.Size(368, 327);

this.variableListView.TabIndex = 9;

this.variableListView.UseCompatibleStateImageBehavior = false;

this.variableListView.View = System.Windows.Forms.View.List;

//

// constantListView

//

this.constantListView.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

| System.Windows.Forms.AnchorStyles.Right)));

this.constantListView.BackColor = System.Drawing.Color.White;

this.constantListView.BorderStyle = System.Windows.Forms.BorderStyle.None;

this.constantListView.Font = new System.Drawing.Font("Microsoft Sans Serif", 19.8F);

this.constantListView.HideSelection = false;

this.constantListView.Location = new System.Drawing.Point(516, 226);

this.constantListView.Margin = new System.Windows.Forms.Padding(2);

this.constantListView.Name = "constantListView";

this.constantListView.Size = new System.Drawing.Size(492, 327);

this.constantListView.TabIndex = 10;

this.constantListView.UseCompatibleStateImageBehavior = false;

this.constantListView.View = System.Windows.Forms.View.List;

//

// variableTitle

//

this.variableTitle.AutoSize = true;

this.variableTitle.Font = new System.Drawing.Font("Microsoft Sans Serif", 13.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.variableTitle.Location = new System.Drawing.Point(248, 197);

this.variableTitle.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.variableTitle.Name = "variableTitle";

this.variableTitle.Size = new System.Drawing.Size(193, 24);

this.variableTitle.TabIndex = 11;

this.variableTitle.Text = "Список переменных";

//

// constantTitle

//

this.constantTitle.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Right)));

this.constantTitle.AutoSize = true;

this.constantTitle.Font = new System.Drawing.Font("Microsoft Sans Serif", 13.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.constantTitle.Location = new System.Drawing.Point(694, 197);

this.constantTitle.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.constantTitle.Name = "constantTitle";

this.constantTitle.Size = new System.Drawing.Size(162, 24);

this.constantTitle.TabIndex = 12;

this.constantTitle.Text = "Список констант";

//

// MainForm

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.Gray;

this.ClientSize = new System.Drawing.Size(1017, 563);

this.Controls.Add(this.constantTitle);

this.Controls.Add(this.variableTitle);

this.Controls.Add(this.constantListView);

this.Controls.Add(this.variableListView);

this.Controls.Add(this.clearButton);

this.Controls.Add(this.resultTitle);

this.Controls.Add(this.infoButton);

this.Controls.Add(this.resultTextBox);

this.Controls.Add(this.inputTitle);

this.Controls.Add(this.title);

this.Controls.Add(this.analyzeButton);

this.Controls.Add(this.inputTextBox);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));

this.Margin = new System.Windows.Forms.Padding(2);

this.Name = "MainForm";

this.Text = "Analyzer";

this.Load += new System.EventHandler(this.MainForm\_Load);

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.TextBox inputTextBox;

private System.Windows.Forms.Button analyzeButton;

private System.Windows.Forms.Label title;

private System.Windows.Forms.Label inputTitle;

private System.Windows.Forms.TextBox resultTextBox;

private System.Windows.Forms.Button infoButton;

private System.Windows.Forms.Label resultTitle;

private System.Windows.Forms.Button clearButton;

private System.Windows.Forms.ListView variableListView;

private System.Windows.Forms.ListView constantListView;

private System.Windows.Forms.Label variableTitle;

private System.Windows.Forms.Label constantTitle;

}

}

MainForm.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace Analyzer

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

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

{

variableListView.Clear();

constantListView.Clear();

try

{

StringParser sp = new StringParser(inputTextBox.Text);

sp.ParseString();

variableListView.Items.AddRange(Checker.variables.Select(x => new ListViewItem(x.ToString())).ToArray());

constantListView.Items.AddRange(Checker.constants.Select(x => new ListViewItem(x.ToString())).ToArray());

resultTextBox.Text = "Строка принадлежит языку";

}

catch (KeyNotFoundException ex)

{

resultTextBox.Text = ex.Message;

}

catch (ArgumentException ex)

{

HighlightErrorSymbol(StringParser.pos, 1);

resultTextBox.Text = ex.Message;

}

catch (FormatException ex)

{

HighlightErrorSymbol(StringParser.pos, 1);

resultTextBox.Text = ex.Message;

}

}

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

{

MessageBox.Show("Идентификатор языка Turbo Pascal, начинается с буквы или знака подчеркивания, далее " +

"могут следовать буквы, цифры и знаки подчеркивания.\n Ограничения:\n" +

"– имеет длину не более 8 символов;\n" +

"– не является зарезервированным словом: case, of, else, end, div, mod.\n Целые числа в диапазоне - 32768 – 32767.");

}

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

{

inputTextBox.Text = String.Empty;

resultTextBox.Text = String.Empty;

variableListView.Clear();

constantListView.Clear();

}

private void HighlightErrorSymbol(int i, int errorLength)

{

errorLength = 0;

inputTextBox.SelectionStart = i - errorLength;

inputTextBox.SelectionLength = errorLength;// - (errorLength == 1 ? 0 : 1);

inputTextBox.Focus();

}

private void textBox1\_KeyDown(object sender, KeyEventArgs e)

{

}

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

{

}

}

}

State.cs

namespace Analyzer

{

enum State

{

A1,

A2,

B1,

B2,

B3,

F

}

}

StringParser.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Analyzer

{

class StringParser

{

public static int pos = 0;

private String source;

private int choice\_type = 0;

// 0 - undefined

// 1 - char

// 2 - int

// 3 - double

// variable, its description

public SortedDictionary<String, String> variables;

public SortedDictionary<String, (String, String)> constants;

public static bool IsDigit(char c)

{

return ('0' <= c && c <= '9');

}

public static bool IsLetter(char c)

{

return ('a' <= c && c <= 'z');

}

public StringParser(String source)

{

this.source = source.ToLower();

if (string.IsNullOrEmpty(source))

{

throw new KeyNotFoundException("Строка не может быть пустой");

}

variables = new SortedDictionary<string, string>();

constants = new SortedDictionary<string, (string, string)>();

pos = 0;

}

public void ParseString()

{

SkipWS();

if (source.Substring(pos).StartsWith("case "))

{

pos += 5;

}

else

{

throw new FormatException("Синтаксическая ошибка: не обнаружено ключевое слово case");

}

SkipWS();

nextIdentifier("идентификатор выбора");

if (pos < source.Length && source[pos] == ' ')

{

++pos;

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидался пробел");

}

SkipWS();

if (source.Substring(pos).StartsWith("of "))

{

pos += 3;

}

else

{

throw new FormatException("Синтаксическая ошибка: не обнаружено ключевое слово of");

}

var state = State.A1;

while (pos < source.Length)

{

switch (state)

{

case State.A1:

SkipWS();

if (pos >= source.Length)

{

throw new FormatException("Синтаксическая ошибка: ожидалась константа выбора");

}

if (source[pos] == '\'')

{

if (choice\_type > 1)

{

throw new FormatException("Семантическая ошибка: несоответствие типов констант выбора");

}

choice\_type = 1;

++pos;

String text\_name = "";

if (pos < source.Length && (IsDigit(source[pos]) || IsLetter(source[pos])))

{

text\_name += source[pos++];

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидался символ");

}

while (pos < source.Length && (IsDigit(source[pos]) || IsLetter(source[pos])))

{

text\_name += source[pos++];

}

if (pos < source.Length && source[pos] == '\'')

{

++pos;

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидалась кавычка");

}

constants.Add("\'" + text\_name + "\'", ("конст. выбора", "строковая"));

}

else

{

if (choice\_type == 1)

{

throw new FormatException("Семантическая ошибка: несоответствие типов констант выбора");

}

bool its\_type = nextConstant("конст. выбора");

choice\_type = Math.Max(choice\_type, (its\_type ? 2 : 3));

}

state = State.A2;

break;

case State.A2:

SkipWS();

if (pos < source.Length)

{

if (source[pos] == ',')

{

state = State.A1;

}

else

{

if (source[pos] == ':')

{

state = State.B1;

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидалась запятая или двоеточие");

}

}

++pos;

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидалась запятая или двоеточие");

}

break;

case State.B1:

SkipWS();

if (pos < source.Length)

{

nextAssignmentOperator();

SkipWS();

if (pos < source.Length && source[pos] == ';')

{

++pos;

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидалась точка с запятой");

}

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидался оператор присваивания");

}

state = State.B2;

break;

case State.B2:

SkipWS();

if (source.Substring(pos).StartsWith("end;"))

{

pos += 4;

if (pos != source.Length)

{

throw new FormatException("Синтаксическая ошибка: встречен код после END;");

}

state = State.F;

break;

}

if (pos >= source.Length)

{

throw new FormatException("Синтаксическая ошибка: ожидалась точка с запятой, список констант или ELSE");

}

if (source.Substring(pos).StartsWith("else "))

{

pos += 5;

SkipWS();

nextAssignmentOperator();

SkipWS();

if (pos < source.Length && source[pos] == ';')

{

++pos;

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидалась точка с запятой");

}

SkipWS();

if (source.Substring(pos).StartsWith("end;"))

{

pos += 4;

state = State.F;

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидался end;");

}

if (pos != source.Length)

{

throw new FormatException("Синтаксическая ошибка: встречен код после END;");

}

}

else

{

if (source[pos] == '\'')

{

if (choice\_type > 1)

{

throw new FormatException("Семантическая ошибка: несоответствие типов констант выбора");

}

choice\_type = 1;

++pos;

String text\_name = "";

if (pos < source.Length && (IsDigit(source[pos]) || IsLetter(source[pos])))

{

text\_name += source[pos++];

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидался символ");

}

while (pos < source.Length && (IsDigit(source[pos]) || IsLetter(source[pos])))

{

text\_name += source[pos++];

}

if (pos < source.Length && source[pos] == '\'')

{

++pos;

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидалась кавычка");

}

constants.Add("\'" + text\_name + "\'", ("конст. выбора", "строковая"));

}

else

{

if (choice\_type == 1)

{

throw new FormatException("Семантическая ошибка: несоответствие типов констант выбора");

}

bool its\_type = nextConstant("конст. выбора");

choice\_type = Math.Max(choice\_type, (its\_type ? 2 : 3));

}

state = State.B3;

}

break;

case State.B3:

SkipWS();

if (pos >= source.Length)

{

throw new FormatException("Синтаксическая ошибка: ожидась точка с запятой, двоеточие или список констант");

}

if (source[pos] == ':')

{

++pos;

state = State.B1;

}

else

{

if (source[pos] != ',')

{

throw new FormatException("Синтаксическая ошибка: ожидилась запятая или двоеточие");

}

++pos;

SkipWS();

if (source[pos] == '\'')

{

++pos;

String text\_name = "";

if (pos < source.Length && (IsDigit(source[pos]) || IsLetter(source[pos])))

{

text\_name += source[pos++];

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидался символ");

}

while (pos < source.Length && (IsDigit(source[pos]) || IsLetter(source[pos])))

{

text\_name += source[pos++];

}

if (pos < source.Length && source[pos] == '\'')

{

++pos;

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидалась кавычка");

}

constants.Add("\'" + text\_name + "\'", ("конст. выбора", "строковая"));

}

else

{

if (choice\_type == 1)

{

throw new FormatException("Семантическая ошибка: несоответствие типов констант выбора");

}

bool its\_type = nextConstant("конст. выбора");

choice\_type = Math.Max(choice\_type, (its\_type ? 2 : 3));

}

state = State.B3;

}

break;

default:

throw new Exception("Something went wrong...");

}

}

}

public String nextIdentifier(String base\_type)

{

String id\_name = "";

if (pos < source.Length && (IsLetter(source[pos]) || source[pos] == '\_'))

{

id\_name += source[pos++];

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидался идентификатор");

}

while (pos < source.Length && (IsLetter(source[pos]) || IsDigit(source[pos]) || source[pos] == '\_'))

{

id\_name += source[pos++];

}

if (id\_name.Length > 8)

{

pos -= id\_name.Length;

throw new FormatException("Синтаксическая ошибка: длина имени идентификатора превышает 8 символов");

}

List<String> reserved\_names = new List<String>() { "case", "of", "else", "end", "div", "mod" };

if (reserved\_names.Contains(id\_name))

{

pos -= id\_name.Length;

throw new FormatException("Синтаксическая ошибка: имя идентификатора совпадает с зарезервированным словом " + id\_name);

}

if (!variables.ContainsKey(id\_name))

{

variables.Add(id\_name, base\_type);

}

return id\_name;

}

public void nextIntegralConstant(String base\_type)

{

String res = "";

if (pos >= source.Length)

{

throw new FormatException("Синтаксическая ошибка: ожидалась целая константа");

}

if (source[pos] == '+' || source[pos] == '-')

{

res += source[pos++];

}

if (IsDigit(source[pos]))

{

if (source[pos] == '0')

{

if (pos + 1 < source.Length || IsDigit(source[pos + 1]))

{

throw new FormatException("Синтаксическая ошибка: ведущие нули в константе");

}

res += "0";

++pos;

}

else

{

while (pos < source.Length && IsDigit(source[pos]))

{

res += source[pos++];

}

}

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидалась целая константа");

}

int val1;

if (int.TryParse(res, out val1))

{

if (val1 < -32768 || 32767 < val1)

{

throw new FormatException("Значение константы не является допустимым");

}

}

if (!constants.ContainsKey(res))

constants.Add(res, (base\_type, "целая"));

}

public bool nextConstant(string base\_type) // if constant is integer

{

String res = "";

if (pos >= source.Length)

{

throw new FormatException("Синтаксическая ошибка: ожидалась константа");

}

if (source[pos] == '+' || source[pos] == '-')

{

res += source[pos++];

}

if (IsDigit(source[pos]))

{

if (source[pos] == '0')

{

++pos;

if (pos < source.Length && IsDigit(source[pos]) && source[pos] != '.')

{

throw new FormatException("Синтаксическая ошибка: ведущие нули в константе");

}

res += '0';

}

else

{

while (pos < source.Length && IsDigit(source[pos]))

{

res += source[pos++];

}

}

if (pos < source.Length && source[pos] == '.')

{

res += '.';

++pos;

while (pos < source.Length && IsDigit(source[pos]))

{

res += source[pos++];

}

if (pos < source.Length && source[pos] == 'e')

{

res += 'e';

++pos;

SkipWS();

if (pos >= source.Length || !IsDigit(source[pos]) && source[pos] != '+' && source[pos] != '-')

{

throw new FormatException("Синтаксическая ошибка: ожидалась экспонента");

}

if (source[pos] == '+' || source[pos] == '-')

{

res += source[pos++];

}

SkipWS();

while (pos < source.Length && IsDigit(source[pos]))

{

res += source[pos++];

}

if (!constants.ContainsKey(res))

constants.Add(res, (base\_type, "вещ. с плав. точкой"));

}

else

{

if (!constants.ContainsKey(res))

constants.Add(res, (base\_type, "вещ. с фикс. точкой"));

}

return false;

}

else

{

int val1;

if (int.TryParse(res, out val1))

{

if (val1 < -32768 || 32767 < val1)

{

throw new FormatException("Значение константы не является допустимым");

}

}

if (!constants.ContainsKey(res))

constants.Add(res, (base\_type, "целая"));

return true;

}

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидалась целая константа");

}

}

public void nextOperand()

{

if (pos >= source.Length)

{

throw new FormatException("Синтаксическая ошибка: ожидался операнд");

}

if (IsDigit(source[pos]) || source[pos] == '+' || source[pos] == '-')

{

nextConstant("конст. выражения");

}

else

{

String id\_name = nextIdentifier("идентификатор");

SkipWS();

if (pos < source.Length && source[pos] == '[')

{

if (variables[id\_name] != "массив" && variables[id\_name] != "идентификатор")

{

throw new FormatException("Семантическая ошибка: обращение к переменной как к массиву");

}

variables[id\_name] = "массив";

++pos;

SkipWS();

if (pos >= source.Length)

{

throw new FormatException("Синтаксическая ошибка: ожидался идентификатор или целая константа");

}

if (IsDigit(source[pos]) || source[pos] == '+' || source[pos] == '-')

{

nextIntegralConstant("индекс");

}

else

{

String index\_id = nextIdentifier("индекс");

if (variables[index\_id] != "индекс" && variables[index\_id] != "переменная")

{

pos -= index\_id.Length;

throw new FormatException("Семантическая ошибка: некорректное повторное использование идентификатора");

}

variables[index\_id] = "индекс";

}

if (pos >= source.Length || source[pos++] != ']')

{

throw new FormatException("Синтаксическая ошибка: ожидалась закрывающая квадратная скобка");

}

}

else

{

if (variables[id\_name] != "переменная" && variables[id\_name] != "идентификатор" && variables[id\_name] != "индекс")

{

throw new FormatException("Семантическая ошибка: некорректное повторное использование идентификатора");

}

variables[id\_name] = "переменная";

}

}

}

public void nextAssignmentOperator()

{

String id\_name = nextIdentifier("идентификатор");

SkipWS();

if (pos >= source.Length)

{

throw new FormatException("Синтаксическая ошибка: ожидалось присваивание");

}

if (source[pos] == '[')

{

if (variables[id\_name] != "массив" && variables[id\_name] != "идентификатор")

{

throw new FormatException("Семантическая ошибка: обращение к переменной как к массиву");

}

variables[id\_name] = "массив";

pos += 1;

SkipWS();

if (pos >= source.Length)

{

throw new FormatException("Синтаксическая ошибка: ожидался идентификатор или целая константа");

}

if (IsDigit(source[pos]) || source[pos] == '+' || source[pos] == '-')

{

nextIntegralConstant("индекс");

}

else

{

String index\_id = nextIdentifier("индекс");

if (variables[index\_id] != "индекс" && variables[index\_id] != "переменная")

{

throw new FormatException("Семантическая ошибка: некорректное повторное использование идентификатора");

}

variables[index\_id] = "индекс";

}

if (pos >= source.Length || source[pos++] != ']')

{

throw new FormatException("Синтаксическая ошибка: ожидалась закрывающая квадратная скобка");

}

}

else

{

if (variables[id\_name] != "переменная" && variables[id\_name] != "идентификатор" && variables[id\_name] != "индекс")

{

throw new FormatException("Семантическая ошибка: некорректное повторное использование идентификатора");

}

variables[id\_name] = "переменная";

}

SkipWS();

if (pos + 1 < source.Length && source[pos] == ':' && source[pos + 1] == '=')

{

pos += 2;

}

else

{

throw new FormatException("Синтаксическая ошибка: ожидалось присваивание");

}

SkipWS();

nextOperand();

SkipWS();

foreach (String oper in new List<string>{ "+", "-", "\*", "/", "div", "mod"})

{

if (source.Substring(pos).StartsWith(oper))

{

pos += oper.Length;

SkipWS();

nextOperand();

break;

}

}

}

public void SkipWS()

{

while (pos < source.Length && source[pos] == ' ')

{

++pos;

}

}

}

}