Министерство науки и высшего образования Российской Федерации

Муромский институт (филиал)

Федерального государственного бюджетного образовательного учреждения высшего образования

«Владимирский государственный университет   
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет ИТР

Кафедра ПИн

*ЛАБОРАТОРНАЯ*

*РАБОТА №2*

По Теория автоматов и формальных языков

Тема Лексический анализ (классификация лексем)

Руководитель

Кульков Я.Ю.

(фамилия, инициалы)

(подпись) (дата)

Студент ПИН - 121

(группа)

Софронов В.А.

(фамилия, инициалы)

(подпись) (дата)

Муром 2022

**Лабораторная работа №2**

Тема: Классификация и сохранение информации об обнаруженной лексеме.

Цель работы: Ознакомление с назначением и принципами работы

лексических анализаторов, получение практических навыков построения

сканера на примере заданного входного языка.

**Ход работы:**

**Задание**

Дополнить лексический анализатор, подготовленный на первой лабораторной

работе, выполнив классификацию выделенных лексем.

1. Продумать архитектуру системы.

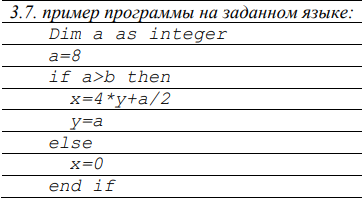
2. Продумать организацию структур данных.

3. Дополнить программу лексического анализа классификатором лексем в

виде токенов.

4. Выводить на форму лексемы с указанием типа каждой из них.

5. Составить тестовые наборы данных и проверить на них работу программы.



Код класса Form1 (для 1-й и 2-й лабораторной работы)

using System.Data;

using System.Windows.Forms;

namespace WinFormsApp123

{

public partial class Form1 : Form

{

Analysis analysis = new Analysis();

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e) //get path to file

{

Analysis.PathToFile(textBox1, richTextBox2);

}

private void button2\_Click(object sender, EventArgs e) //get analys

{

analysis.Gate(richTextBox1, richTextBox2);

}

private void button3\_Click(object sender, EventArgs e) //get token

{

workToken.ReWork(richTextBox2, richTextBox3);

}

private void richTextBox1\_TextChanged(object sender, EventArgs e)

{

}

private void richTextBox1\_MouseClick(object sender, MouseEventArgs e)

{

try

{

int firstcharindex = richTextBox1.GetFirstCharIndexOfCurrentLine();

int currentline = richTextBox1.GetLineFromCharIndex(firstcharindex);

string currentlinetext = richTextBox1.Lines[currentline];

richTextBox1.Select(firstcharindex, currentlinetext.Length + 1);

}

catch { }

}

}

}

Код класса Lexems (для 1-й лабораторной работы)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp123

{

public class Lexems

{

public static bool IsIDperator(string text)

{

return text == "integer" || text == "if" || text == "else" || text == "c" || text == "as" || text == "then" || text == "end";

}

public static bool IsSeparator(string text)

{

return text == "\n" || text == "||" || text == "&&" || text == "()" || text == "<" || text == ">" || text == ":" || text == "=" || text == ":=" || text == "{" || text == "}" || text == "+" || text == "-" || text == "\*" || text == "/" || text == ";" || text == "(" || text == ")" || text == ",";

}

public static bool IsLiteral(string text)

{

int x;

return int.TryParse(text, out x);

}

public static bool IsErrorLetter(string text)

{

return text == "@" || text == "$" || text == "[" || text == "]" || text == "`" || text == "~" ;

}

public static bool IsIDVariable(string text)

{

bool Flag = true;

if (text.Length == 0 || !char.IsLetter(text[0]))

Flag = false;

else

foreach (char s in text)

{

if (s != '\_' && !char.IsDigit(s) && !char.IsLetter(s))

Flag = false;

}

return Flag;

}

}

}

Код класса Analysis (для 1-й лабораторной работы)

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WinFormsApp123

{

class Analysis

{

Dictionary<string, string> parts = new Dictionary<string, string>();

public static void PathToFile(System.Windows.Forms.TextBox box1, RichTextBox box2)

{

string text = "";

using (StreamReader fs = new StreamReader(box1.Text))

{

while (true)

{

string temp = fs.ReadLine();

if (temp == null) break;

text += temp;

text += " \n ";

}

box2.Text = text;

}

}

public void Gate(RichTextBox box1, RichTextBox box2)

{

parts = new Dictionary<string, string>();

int i = 0;

string subText = "";

bool IsErrorFlag = false;

foreach (char s in box2.Text)

{

try

{

if (Lexems.IsIDperator(subText) && (s == ' ' || s == '<' || s == '>' || s == ';'))

{

i++;

parts.Add(i.ToString() + " ", subText + " - Идентификатор - оператор;");

subText = "";

}

else if (Lexems.IsSeparator(subText) && (s == ' ' || s == '(' || char.IsDigit(s) || char.IsLetter(s)))

{

i++;

if (subText != "\n")

parts.Add(i.ToString() + " ", subText + " - Разделитель;");

else

{

subText = "'/n'";

parts.Add(i.ToString() + " ", subText + " - Разделитель (символ новой строки);");

}

subText = "";

}

else if (Lexems.IsLiteral(subText) && (s == ' ' || s == ';' || s == ')'))

{

i++;

parts.Add(i.ToString() + " ", subText + " - Литератор;");

subText = "";

}

else if (Lexems.IsIDVariable(subText) && !(Lexems.IsIDperator(subText) && (s == ' ' || s == '<' || s == '>' || s == ';' || s == '+' || s == '-' || s == '\*' || s == '/' || s == ',' || s == '(' || s == ')')))

{

i++;

parts.Add(i.ToString() + " ", subText + " - Идентификатор - переменная;");

subText = "";

}

else if (Lexems.IsErrorLetter(subText))

{

MessageBox.Show($"Ошибка в программе. Неизвестный символ '{subText}'.");

subText = "";

break;

}

else if (subText == Environment.NewLine || subText == " ")

{

subText = "";

}

subText += s;

}

catch (Exception)

{

MessageBox.Show("Не удалось закончить лексический анализ.");

}

}

box1.Clear();

foreach (KeyValuePair<string, string> pair in parts)

{

box1.Text += pair.Key.PadRight(10) + " " + pair.Value + Environment.NewLine;

}

MessageBox.Show("Лексический анализ закончен");

}

}

}

Код класса Token (для 2-й лабораторной работы)

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace laba1\_v45\_6

{

public class Token

{

public TokenType Type;

public string Value;

public Token(TokenType type)

{

Type = type;

}

public override string ToString()

{

return string.Format("{0}, {1}", Type, Value);

}

public enum TokenType

{

AS,DIM,INTEGER, MAIN, THEN, INT, BOOL, LITERAL, IDENTIFIER, CURLYBRACECLOSE,

IF, ELSE, TRUE, FALSE, PLUS, MORE, LESS, SEMICOLON, CURLYBRACEOPEN,

MINUS, EQUAL, MULTIPLY, RPAR, LPAR, ENTER, DIVISION, COMMA, VARIABLE }

public static Dictionary<string, TokenType> SpecialWords = new Dictionary<string, TokenType>()

{

{ "Dim", TokenType.DIM },

{ "if", TokenType.IF },

{ "integer", TokenType.INTEGER },

{ "else", TokenType.ELSE },

{ "as", TokenType.AS },

{ "then", TokenType.THEN }, };

public static bool IsSpecialWord(string word)

{

if (string.IsNullOrEmpty(word))

{

return false;

}

return SpecialWords.ContainsKey(word);

}

public static Dictionary<char, TokenType> SpecialSymbols = new Dictionary<char, TokenType>()

{

{ '#', TokenType.ENTER },

{ '(', TokenType.LPAR },

{ ')', TokenType.RPAR },

{ '+', TokenType.PLUS },

{ '-', TokenType.MINUS },

{ '=', TokenType.EQUAL },

{ '>', TokenType.MORE },

{ '<', TokenType.LESS },

{ '\*', TokenType.MULTIPLY },

{ '/', TokenType.DIVISION },

{ ',', TokenType.COMMA },

{ ';', TokenType.SEMICOLON },

{ '{', TokenType.CURLYBRACEOPEN },

{ '}', TokenType.CURLYBRACECLOSE }

};

public static bool IsSpecialSymbol(char ch)

{

return SpecialSymbols.ContainsKey(ch);

}

public static void PrintTokens(RichTextBox box3, List<Token> list)

{

int i = 0;

foreach (var t in list)

{

i++;

box3.Text += $"{i} {t} ";

box3.Text += Environment.NewLine;

}

}

}

}

Код класса AnalysisToken (для 2-й лабораторной работы)

using System;

using System.Collections.Generic;

using System.IO;

using System.IO.Ports;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace laba1\_v45\_6

{

internal class AnalysisToken

{

public List<string> forToken = new List<string>();

public List<string> listBuf = new List<string>();

public List<char> forChar = new List<char>();

public List<Token> tokens = new List<Token>();

string str;

string type;

char ch;

Token token;

public static void PathToFile(System.Windows.Forms.TextBox box1, RichTextBox box2)

{

string text = "";

using (StreamReader fs = new StreamReader(box1.Text))

{

while (true)

{

string temp = fs.ReadLine();

if (temp == null) break;

text += temp;

text += " \n ";

}

box2.Text = text;

}

}

public void ReWork(RichTextBox box2, RichTextBox box3)

{

listBuf = new List<string>();

forToken = new List<string>();

forChar = new List<char>();

int i = 0;

string subText = "";

foreach (char s in box2.Text)

{

try

{

if (Lexems.IsIDperator(subText) && (s == ' ' || s == '<' || s == '>' || s == ';'))

{

i++;

listBuf.Add(subText + " "); // - Идентификатор - оператор;

forToken.Add("I");

forChar.Add(' ');

subText = "";

}

else if (Lexems.IsLiteral(subText) && (s == ' ' || s == ';' || s == ')'))

{

i++;

listBuf.Add(subText + " "); // - Литерал;

forToken.Add("D");

forChar.Add(' ');

subText = "";

}

else if (Lexems.IsSeparator(subText) && (s == ' ' || s == '(' || char.IsDigit(s) || char.IsLetter(s)))

{

i++;

if (subText != "\n")

{

listBuf.Add(subText + " "); // - Разделитель;

forToken.Add("R");

forChar.Add(s);

}

else

{

subText = "'/n'";

listBuf.Add(subText + " "); // - разделитель (символ новой строки);

forToken.Add("r");

forChar.Add(s);

}

subText = "";

}

else if (Lexems.IsIDVariable(subText) && !Lexems.IsIDperator(subText) && (s == ' ' || s == '<' || s == '>' || s == ';' || s == '+' || s == '-' || s == '\*' || s == '/' || s == ',' || s == '(' || s == ')'))

{

i++;

listBuf.Add(subText + " "); // - Идентификатор - переменная;

forToken.Add("P");

forChar.Add(' ');

subText = "";

}

else if (subText == Environment.NewLine || subText == " ")

{

subText = "";

}

subText += s;

}

catch (Exception) { }

}

MessageBox.Show("Лексический анализ закончен");

//-------------------------------------------------------------//

for (i = 0; i < listBuf.Count; i++)

{

str = listBuf[i];

type = forToken[i];

if (type == "I")

{

try

{

if (Token.IsSpecialWord(str))

{

token = new Token(Token.SpecialWords[str]);

tokens.Add(token);

continue;

}

else

{

token = new Token(Token.TokenType.IDENTIFIER);

token.Value = str;

tokens.Add(token);

continue;

}

} catch (Exception) { MessageBox.Show("Непредвиденная ошибка в поиске специального слова"); }

}

else if (type == "D")

{

try

{

token = new Token(Token.TokenType.LITERAL);

token.Value = str;

tokens.Add(token);

continue;

} catch (Exception) { MessageBox.Show("Непредвиденная ошибка в поиске литерала"); }

}

else if (type == "P")

{

try

{

token = new Token(Token.TokenType.VARIABLE);

token.Value = str;

tokens.Add(token);

continue;

} catch (Exception) { MessageBox.Show("Непредвиденная ошибка в поиске переменной"); }

}

else if (type == "R")

{

try

{

for (int j = 0; j < str.Length; j++)

{

ch = str[j];

if (Token.IsSpecialSymbol(ch))

{

token = new Token(Token.SpecialSymbols[ch]);

token.Value = str;

tokens.Add(token);

continue;

}

}

} catch (Exception) { MessageBox.Show("Непредвиденная ошибка в поиске разделителей"); }

}

}

Token.PrintTokens(box3, tokens);

}

}

}

Рисунок 1 – вид программы

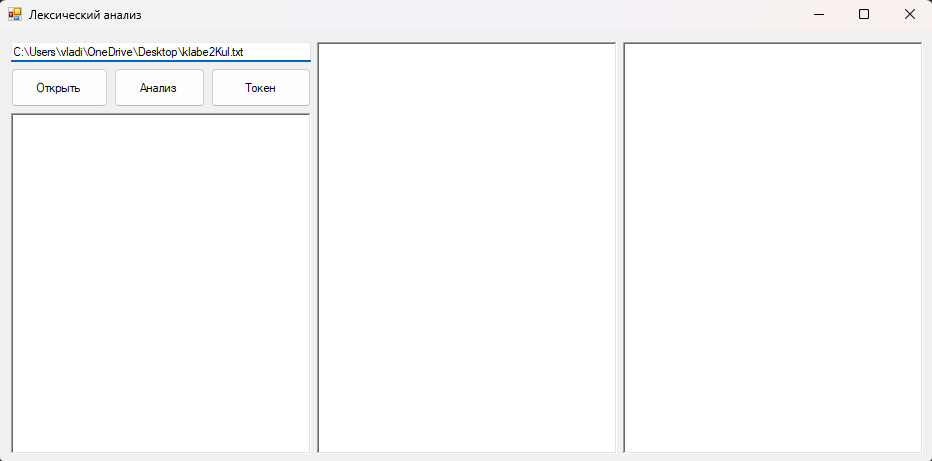
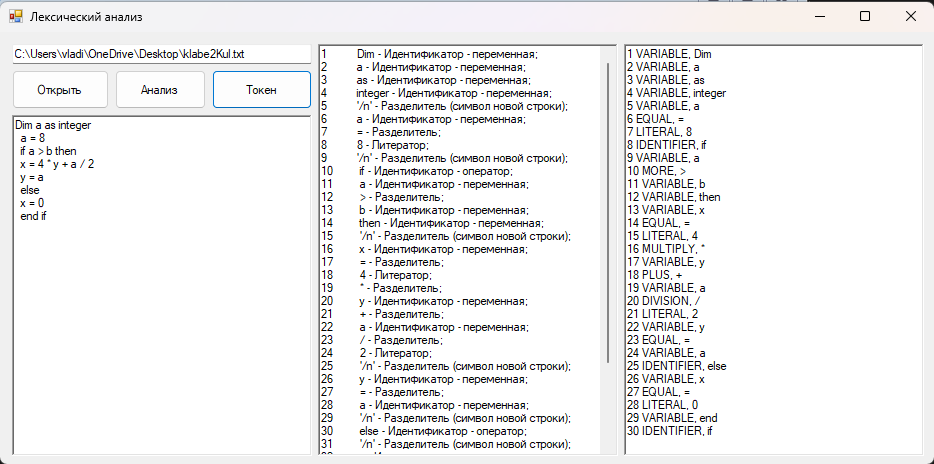


Рисунок 2 – результат работы программы



Вывод: в ходе данной лабораторной работы были получены знания работы лексических анализаторов, получены практические навыки построения сканера на примере заданного входного языка C#.