Министерство образования Российской Федерации

Федеральное агентство по образованию

КАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. А.Н. Туполева - КАИ

Кафедра АСОИУ

Лабораторная работа №1

по дисциплине

«ТЕОРИЯ языков программирования и методов трансляций»

по теме «Разработка транслитератора»

Вариант 3

Выполнил:

Студент группы 4309

Габдуллин С.Ж.

Проверил: Барков И.А.

Казань 2023

**Текст задания:**

1. Спроектировать и отладить экранную форму для ввода исходных данных, вывода сообщений программы и управления программой.
2. Разработать и отладить транслитератор **void GetSymbol()**, пример имеется в модуле **uLexicalAnalizer** из папки «Программы».
3. Для отладки транслитератора временно включить в обработчик нажатия кнопки цикл чтения с помощью функции **GetSymbol()** символов исходного текста и вывода результатов анализа в поле диагностических сообщений.
4. Входной алфавит символов: 0 1 a b c d [ ] “пробел”.

**Исходный текст программы:**

**uMainForm.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;

using nsLex;

namespace nsLexMainForm

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

// tbFSource.Lines.Count = 2;

tbFSource.AppendText("abcd\r\n");

tbFSource.AppendText("[]01\r\n");

tbFSource.AppendText("daa0\r\n");

}

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

{

CLex Lex = new CLex();

// CSourceCode SourceCode = new CSourceCode();

Lex.strPSource = tbFSource.Lines;

Lex.strPMessage = tbFMessage.Lines;

Lex.enumPState = TState.stInitial;

tbFMessage.Text = "";

try

{

while (Lex.enumPState != TState.stExorbitant)

{

CLiteral Literal;

Literal = Lex.NextLiteral();

String s = "";

switch (Literal.enumFSelectionCharType)

{

case TCharType.Letter: { s = "Letter " + Literal.chrFSelection; break; }

case TCharType.Digit: { s = "Digit " + Literal.chrFSelection; break; }

case TCharType.ReservedSymbol: { s = "ReservedSymbol " + Literal.chrFSelection; break; }

case TCharType.EndRow: { s = "EndRow "; break; }

case TCharType.Space: { s = "Space "; break; }

case TCharType.EndText: { s = "EndText "; break; }

}

tbFMessage.Text += "(" + s + ")";

}

tbFMessage.Text+= "Текст верный";

}

catch (Exception exc)

{

tbFMessage.Text += "(" + exc.Message + ")";

tbFSource.Select();

tbFSource.SelectionStart = 0;

int n = 0;

for (int i = 0; i < Lex.intPSourceRowSelection; i++) n += tbFSource.Lines[i].Length + 2;

n += Lex.intPSourceColSelection;

tbFSource.SelectionLength = n;

}

}

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

{

}

}

}

**uLex.cs:**

using System;

namespace nsLex

{

public enum TState { stInitial, stMiddle, stExorbitant };

public enum TCharType { Letter, Digit, EndRow, EndText, Space, ReservedSymbol, Unknown };

public struct CLiteral

{

public char chrFSelection;

public TCharType enumFSelectionCharType;

}

public class CSourceCode

{

private String[] strFSource; // указатель на массив строк

private TState enumFState;

private int intFSourceRowSelection;

public int intFSourceColSelection;

public String[] strPSource { set { strFSource = value; } get { return strFSource; } }

public TState enumPState { set { enumFState = value; } get { return enumFState; } }

public int intPSourceRowSelection { get { return intFSourceRowSelection; } }

public int intPSourceColSelection { get { return intFSourceColSelection; } }

public CSourceCode()

{

intFSourceRowSelection = 0;

intFSourceColSelection = -1;

}

public CLiteral GetSymbol()

{

CLiteral Literal;

intFSourceColSelection++;

if (intFSourceColSelection > strPSource[intFSourceRowSelection].Length - 1)

{

intFSourceRowSelection++;

if (intFSourceRowSelection <= strFSource.Length - 1)

{

intFSourceColSelection = -1;

Literal.chrFSelection = '\0';

Literal.enumFSelectionCharType = TCharType.EndRow;

enumFState = TState.stMiddle;

return Literal;

}

else

{

Literal.chrFSelection = '\0';

Literal.enumFSelectionCharType = TCharType.EndText;

enumFState = TState.stExorbitant;

return Literal;

}

}

else

{

Literal.chrFSelection = strFSource[intFSourceRowSelection][intFSourceColSelection];

Literal.enumFSelectionCharType = TCharType.Unknown;

enumFState = TState.stMiddle;

return Literal;

}

}

}

public class CLex : CSourceCode

{

private String[] strFMessage; // указатель на массив строк

private CLiteral Literal;

public String[] strPMessage { set { strFMessage = value; } get { return strFMessage; } }

public CLex()

{

}

public CLiteral NextLiteral()

{

Literal = GetSymbol();

if (Literal.enumFSelectionCharType == TCharType.Unknown)

if (Literal.chrFSelection == ' ') Literal.enumFSelectionCharType = TCharType.Space;

else if (Literal.chrFSelection >= 'a' && Literal.chrFSelection <= 'd') Literal.enumFSelectionCharType = TCharType.Letter;

else if (Literal.chrFSelection == '0' || Literal.chrFSelection == '1') Literal.enumFSelectionCharType = TCharType.Digit;

else if (Literal.chrFSelection == '[') Literal.enumFSelectionCharType = TCharType.ReservedSymbol;

else if (Literal.chrFSelection == ']') Literal.enumFSelectionCharType = TCharType.ReservedSymbol;

else throw new System.Exception(Literal.chrFSelection.ToString() + " Cимвол вне алфавита");

return Literal;

}

}

}

**Результаты тестирования:**



