Лабораторная работа №1

Выполнил: студент группы 4309

Курбанов Д.И.

Лексический анализатор – программа, принимающая на вход текст (последовательность символов из алфавита) и разбивающая его на подстроки (лексемы) в соответствии с некоторым набором регулярных выражений.

Текст задания:

Спроектировать и отладить экранную форму для ввода исходных данных, вывода сообщений программы и управления программой.

Разработать и отладить транслитератор void GetSymbol(), пример имеется в модуле uLexicalAnalizer из папки «Программы».

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.

Код программы:

Class1.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WindowsFormsApp2

{

public enum TState { Start, Continue, Finish }; //тип состояния

public enum TCharType { Letter, Digit, EndRow, EndText, Space, ReservedSymbol, Undefined, Opendoor,Closedoor, point, talepoint,talepointPoint }; // тип символа

public enum TToken { lxmIdentifier, lxmNumber, lxmUnknown, lxmEmpty, lxmLeftParenth, lxmRightParenth, lxmIs, lxmDot, lxmComma };

public class CLex //класс лексический анализатор

{

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

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

public TCharType enumFSelectionCharType;

public char chrFSelection;

private TState enumFState;

private int intFSourceRowSelection;

private int intFSourceColSelection = -1;

private String strFLexicalUnit;

private TToken enumFToken;

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

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

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

public String strPLexicalUnit { set { strFLexicalUnit = value; } get { return strFLexicalUnit; } }

public TToken enumPToken { set { enumFToken = value; } get { return enumFToken; } }

public int intPSourceRowSelection { get { return intFSourceRowSelection; } set { intFSourceRowSelection = value; } }

public int intPSourceColSelection { get { return intFSourceColSelection; } set { intFSourceColSelection = value; } }

public CLex()

{

}

public void GetSymbol() //метод класса лексический анализатор

{

intFSourceColSelection++; // продвигаем номер колонки

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

{

intFSourceRowSelection++;

if (intFSourceRowSelection <= strFSource.Length - 1)

{

intFSourceColSelection = -1;

chrFSelection = '\0';

enumFSelectionCharType = TCharType.EndRow;

enumFState = TState.Continue;

}

else

{

chrFSelection = '\0';

enumFSelectionCharType = TCharType.EndText;

enumFState = TState.Finish;

}

}

else

{

chrFSelection = strFSource[intFSourceRowSelection][intFSourceColSelection]; //классификация прочитанной литеры

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

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

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

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

else if (chrFSelection == '\*') enumFSelectionCharType = TCharType.ReservedSymbol;

else if (chrFSelection == '(') enumFSelectionCharType = TCharType.Opendoor;

else if (chrFSelection == ')') enumFSelectionCharType = TCharType.Closedoor;

else if (chrFSelection == '.') enumFSelectionCharType = TCharType.point;

else if (chrFSelection == ';') enumFSelectionCharType = TCharType.talepointPoint;

else if (chrFSelection == ',') enumFSelectionCharType = TCharType.talepoint;

else if (chrFSelection == '!' || chrFSelection == '+' || chrFSelection == ';' || chrFSelection == '$' || chrFSelection == '!' || chrFSelection == ':' || chrFSelection == '-' || chrFSelection == ',' ) enumFSelectionCharType = TCharType.ReservedSymbol;

else

{

enumFSelectionCharType = TCharType.Undefined;

//throw new System.Exception("Cимвол вне алфавита");

}

enumFState = TState.Continue;

}

}

private void TakeSymbol()

{

char[] c = { chrFSelection };

String s = new string(c);

strFLexicalUnit += s;

GetSymbol();

}

public void NextToken()

{

strFLexicalUnit = "";

if (enumFState == TState.Start)

{

intFSourceRowSelection = 0;

intFSourceColSelection = -1;

GetSymbol();

}

if (chrFSelection == '/')

{

GetSymbol();

if (chrFSelection == '/')

while (enumFSelectionCharType != TCharType.EndRow)

{

GetSymbol();

}

GetSymbol();

}

}

}

}

Form.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 WindowsFormsApp2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

tbFSource.AppendText("");

int n = tbFSource.Lines.Length;

}

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

{

}

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

{

CLex Lex = new CLex();

Lex.strPSource = tbFSource.Lines;

Lex.strPMessage = tbFMessage.Lines;

Lex.intPSourceColSelection = -1;

Lex.intPSourceRowSelection = 0;

int x = tbFSource.TextLength;

int y = tbFSource.Lines.Length;

tbFMessage.Text = "";

try

{

Lex.GetSymbol(); // Выводятся литеры и классификация

while (Lex.enumPState != TState.Finish)

{

Lex.NextToken();

String s = "";

String s1 = "";

switch (Lex.enumFSelectionCharType)

{

case TCharType.Letter: { s1 = "Letter"; break; }

case TCharType.Digit: { s1 = "Digit"; break; }

case TCharType.Space: { s1 = "Space"; break; }

case TCharType.ReservedSymbol: { s1 = "ReservedSymbol"; break; }

case TCharType.EndRow: { s = "KC"; s1 = "EndRow"; break; }

case TCharType.EndText: { s = "KT"; s1 = "EndText"; break; }

case TCharType.Undefined: { s1 = "Undefined"; break; }

case TCharType.point: { s1 = "point"; break; }

case TCharType.Opendoor: { s1 = "Opendoor"; break; }

case TCharType.Closedoor: { s1 = "Closedoor"; break; }

case TCharType.talepointPoint: { s1 = "talepointPoint"; break; }

case TCharType.talepoint: { s1 = "talepoint"; break; }

}

switch (Lex.enumPToken)

{

case TToken.lxmNumber: { s = "Номер"; s1 = Lex.strPLexicalUnit; break; }

case TToken.lxmIdentifier: { s = "Слово"; s1 = Lex.strPLexicalUnit; break; }

}

String m = "(" + s + "," + s1 + ")"; //литера и ее тип

tbFMessage.Text += m; //добавляется в строку сообщение

}

}

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 tbFMessage\_TextChanged(object sender, EventArgs e)

{

}

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

{

}

}

}

Результат работы программы:

