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»

Выполнила:

Студентка группы 4309

Хафизов Н.Р.

Проверила: Бикмуллина И.И.

Казань, 2023

**Лабораторная работа № 1. Разработка транслитератора**

**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.

**Теория. Основные правила разработки программ**

Разработка программ имеет множество правил и рекомендаций, которые помогают создать эффективный и надежный код. Вот основные правила разработки программ:

1. Используйте понятные и описательные имена переменных, функций и классов. Хорошие имена помогают другим разработчикам легко понять ваш код и улучшают его читаемость. Например, вместо x, y, z предпочтительнее использовать более осмысленные имена, такие как width, height, radius.

2. Разбивайте код на модули или функции с определенной функциональностью. Это улучшает читаемость кода и позволяет повторно использовать функции в разных частях программы.

3. Следуйте принципу единственной ответственности (Single Responsibility Principle). Каждая функция или класс должны быть ответственными только за одну часть функциональности. Это делает код более модульным и легко поддерживаемым.

4. Используйте комментарии для пояснения сложного или непонятного кода. Хорошо написанные комментарии помогут другим разработчикам понять ваш код и его цель. Однако, избегайте комментирования очевидных вещей и старайтесь писать понятный код без необходимости комментариев.

5. Следуйте стандартам и рекомендациям языка программирования, которым вы пользуетесь. Различные языки программирования имеют свои собственные стандарты и рекомендации, которые помогают создавать качественный код. Например, в Python есть рекомендации PEP 8, которые определяют стиль написания кода в этом языке.

6. Пишите тесты для своего кода. Тестирование помогает обнаружить ошибки и убедиться в правильной работе программы. Существуют различные подходы к тестированию, такие как модульное тестирование, функциональное тестирование и интеграционное тестирование.

7. Используйте систему контроля версий, такую как Git, для отслеживания изменений в коде и совместной работы с другими разработчиками. Система контроля версий позволяет вам сохранять историю изменений, легко возвращаться к предыдущим версиям и объединять изменения от разных разработчиков.

8. Обрабатывайте ошибки и исключения. Ваш код должен быть устойчивым к возможным ошибкам и исключительным ситуациям. Обработка ошибок помогает предотвратить сбои программы и предоставить пользователю информацию о возникшей проблеме.

9. Оптимизируйте код только по необходимости. Оптимизация должна быть основана на реальных проблемах производительности и должна проводиться только после профилирования и выявления узких мест в коде.

**Код программы:**

**Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using ТЯПЛР1;

namespace ТЯПЛР1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

tbFSource.AppendText("(0;1,a!b)" + "\r\n");

tbFSource.AppendText("1 a");

int n = tbFSource.Lines.Length;

}

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

{

CLex Lex = new CLex();

Lex.strPSource = tbFSource.Lines;

Lex.strPMessage = tbFMessage.Lines;

int x = tbFSource.TextLength;

int y = tbFSource.Lines.Length;

tbFMessage.Text = "";

try

{

while (Lex.enumPState != TState.Finish)

{

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

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

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

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

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

case TCharType.EXP: { s1 = "voskl"; break; }

case TCharType.COM: { s1 = "zap"; break; }

case TCharType.TCHZP: { s1 = "tochzap"; 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;

}

}

}

}

**СLex.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace ТЯПЛР1

{

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

public enum TCharType { Letter, Digit, EndRow, EndText, Space, ReservedSymbol,OPbr,CLbr,EXP,COM,TCHZP }; // 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;

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.OPbr;

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

else if (chrFSelection == '!' ) enumFSelectionCharType = TCharType.EXP;

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

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

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

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

enumFState = TState.Continue;

}

intFSourceColSelection++;

}

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();

}

intFSourceColSelection++;

}

}

}

**Результаты тестирования:**

