**Лабораторная работа №5**

Выполнила: студент группы 4309

Канькина Ольга Александровна

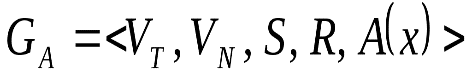
*Атрибутная грамматика* – это КС-грамматика, с узлами дерева вывода которой связаны атрибуты (семантические правила). КС-правилам сопоставляются правила вычисления атрибутов. Правило вычисления значений атрибутов, соответствующее данному КС-правилу, применяется для всех вхождений этого правила в дерево вывода.

Атрибуты могут быть двух видов - *синтезированные и унаследованные*.

*Синтезированные атрибуты* вычисляются с учетом значений атрибутов узлов потомков.

*Унаследованные атрибуты* - это атрибуты, значение которых вычисляется с учетом значений атрибутов его предков.

Формально **атрибутная грамматика** – это пятёрка объектов:

, где

- множество терминальных символов;

- множество нетерминальных символов;

- начальный выделенный символ;

- это правила вывода.

TreeView представляет визуальный элемент в виде дерева. Дерево содержит узлы, которые представляют объекты TreeNode. Узлы могут содержать другие подузлы и могут находиться как скрытом, так и в раскрытом состоянии. Все узлы содержатся в свойстве Nodes.

Внутреннее свойство Nodes, доступное в элементе TreeView возвращает коллекцию узлов дерева, и по умолчанию она пуста. Как видите, чтобы добавить новый элемент, достаточно использовать метод Add(string text) для данного свойства.

Метод Add(string text) возвращает добавленный узел в виде объекта класса TreeNode и к нему точно так же можно обратиться для добавления его дочерних узлов - через свойство Nodes и тот же метод Add(string text).

Синтаксическое дерево - это древовидное представление синтаксической структуры фрагмента исходного кода. Обычно он используется в процессе разработки компилятора, чтобы представить структуру кода таким образом, чтобы его было легче анализировать и манипулировать.

**Задание**. Включить в синтаксический анализатор из лабораторной работы №.3 построение синтаксического дерева. Использовать атрибутный метод Кнута, т.е. преобразовать КС – грамматику из лабораторной работы № 3 в атрибутную грамматику добавлением атрибутов и правил построения синтаксического дерева. Расширить программу синтаксического анализатора из лабораторной работы № 3 введением действий по построению синтаксического дерева.

Код программы:

Analyzer.cs

using System;

using System.Windows.Forms;

using nsLex;

namespace nsSynt

{

internal class uSyntAnalyzer

{

private String[] strFSource;

private String[] strFMessage;

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

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

public CLex Lex = new CLex();

private TreeView tree; // Declare TreeView for function calls

public uSyntAnalyzer(TreeView treeView)

{

tree = treeView;

}

public int i = -1;

public void S()

{

TreeNode parent = new TreeNode("S");

tree.Nodes.Add(parent);

Console.WriteLine("S");

A(parent);

Console.WriteLine(Lex.enumPToken);

while (Lex.enumPToken == TToken.lxmmul || Lex.enumPToken != TToken.lxmEmpty)

{

parent.Nodes.Add(new TreeNode("\_"));

TreeNode highParent = new TreeNode("S");

parent.Nodes.Add(highParent);

Lex.NextToken();

A(highParent);

}

throw new Exception("Текст верный.");

}

public void C(TreeNode highParent)

{

highParent.Nodes.Add(new TreeNode("\_"));

Console.WriteLine("C");

if (Lex.enumPToken == TToken.lxmmul)

{

Lex.NextToken();

B(highParent);

if (Lex.enumPToken == TToken.lxmmul)

{

C(highParent);

}

}

}

public void A(TreeNode highParent)

{

Console.WriteLine("A");

TreeNode parent = new TreeNode("A");

highParent.Nodes.Add(parent);

B(parent);

if (Lex.enumPToken == TToken.lxmmul)

{

//parent.Nodes.Add (new TreeNode(","));

C(parent);

}

if (Lex.enumPToken == TToken.lxmNumber || Lex.enumPToken == TToken.lxmIdentifier)

{

throw new Exception("ERROR");

}

}

public void B(TreeNode highParent)

{

TreeNode parent = new TreeNode("B");

highParent.Nodes.Add(parent);

Console.WriteLine("B");

if (Lex.enumPToken == TToken.lxmNumber)

{

parent.Nodes.Add(new TreeNode(Lex.strPLexicalUnit));

while (Lex.enumPToken == TToken.lxmNumber) Lex.NextToken();

}

else if (Lex.enumPToken == TToken.lxmIdentifier)

{

parent.Nodes.Add(new TreeNode(Lex.strPLexicalUnit));

while (Lex.enumPToken == TToken.lxmIdentifier)

{

Lex.NextToken();

}

if (Lex.enumPToken == TToken.lxmmul)

{

parent.Nodes.Add(new TreeNode("\_"));

Lex.NextToken();

if (Lex.enumPToken == TToken.lxmNumber)

{

parent.Nodes.Add(new TreeNode(Lex.strPLexicalUnit));

while (Lex.enumPToken == TToken.lxmNumber) Lex.NextToken();

}

else throw new Exception("Текст верный");

}

else throw new Exception("Ожидалось \_");

}

else throw new Exception("Ожидался идентификатор");

}

}

}

Form1.cs

using nsLex;

using nsLexMainForm;

using nsSynt;

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 WindowsFormsApp26;

namespace CHashTableList

{

public partial class Form1 : Form

{

Dictionary<int, List<string>> hashTableIdentifier = new Dictionary<int, List<string>>();

Dictionary<int, List<string>> hashTableDigital = new Dictionary<int, List<string>>();

Dictionary<int, List<string>> hashTableRezerv = new Dictionary<int, List<string>>();

public MyHashFunction hashFunction = new MyHashFunction();

public CHashTableList htl = new CHashTableList(2);

public Form1()

{

InitializeComponent();

tbFSource.AppendText("100\_a\_100\_a\_" + "\r\n");

int n = tbFSource.Lines.Length;

}

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

{

}

public void TablesToMemo(object sender, System.EventArgs e)

{

List<string> listTable = new List<string>();

listBox1.Items.Clear();

listBox2.Items.Clear();

listBox3.Items.Clear();

htl.TableToStringList(0, listTable);

//for (int i = 0; i < listTable.Count; i++)

// listBox1.Items.Add(listTable[i]);

foreach (var entry in hashTableIdentifier)

{

listBox1.Items.Add(string.Join(", ", entry.Value));

}

listTable.Clear();

htl.TableToStringList(1, listTable);

//for (int i = 0; i < listTable.Count; i++)

// listBox2.Items.Add(listTable[i]);

foreach (var entry in hashTableDigital)

{

listBox2.Items.Add(string.Join(", ", entry.Value));

}

listTable.Clear();

foreach (var entry in hashTableRezerv)

{

listBox3.Items.Add(string.Join(", ", entry.Value));

}

listTable.Clear();

}

private void btnFStart\_Click\_1(object sender, EventArgs e)

{

tbFMessage.Clear();

uSyntAnalyzer Synt = new uSyntAnalyzer(SyntTree); // Используем существующий экземпляр treeView2

Synt.Lex.strPSource = tbFSource.Lines;

Synt.Lex.strPMessage = tbFMessage.Lines;

Synt.Lex.enumPState = TState.Start;

try

{

Synt.Lex.NextToken();

Synt.S();

throw new Exception("Текст верный");

}

catch (Exception exc)

{

tbFMessage.Text += exc.Message;

tbFSource.Select();

tbFSource.SelectionStart = 0;

int n = 0;

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

n += Synt.Lex.intPSourceColSelection;

tbFSource.SelectionLength = n;

}

}

private void btnFRecord\_Click\_1(object sender, EventArgs e)

{

CLex Lex = new CLex();

Lex.strPSource = tbFSource.Lines;

Lex.strPMessage = tbFMessage.Lines;

//Lex.intPSourceColSelection = 0;

//Lex.intPSourceRowSelection = 0;

int x = tbFSource.TextLength;

int y = tbFSource.Lines.Length;

tbFMessage.Text = "";

try

{

while (Lex.enumPState != TState.Finish)

{

Lex.NextToken();

string s1 = "", s = "";

switch (Lex.enumPToken)

{

case TToken.lxmIdentifier:

{

hashFunction.AddWord(hashTableIdentifier, Lex.strPLexicalUnit);

s1 = "id " + Lex.strPLexicalUnit; int b = 0;

if (htl.AddLexicalUnit(Lex.strPLexicalUnit, 0, ref b))

{

TablesToMemo(this, e);

}

break;

}

case TToken.lxmNumber:

{

hashFunction.AddWord(hashTableDigital, Lex.strPLexicalUnit);

s1 = "num " + Lex.strPLexicalUnit; int b = 0;

if (htl.AddLexicalUnit(Lex.strPLexicalUnit, 1, ref b))

{

TablesToMemo(this, e);

}

break;

}

case (TToken.lxmdt):

{

hashFunction.AddWord(hashTableRezerv, ":");

s1 = "rez " + Lex.strPLexicalUnit; int b = 0;

if (htl.AddLexicalUnit(Lex.strPLexicalUnit, 1, ref b))

{

TablesToMemo(this, e);

}

break;

}

case (TToken.lxmComma):

{

hashFunction.AddWord(hashTableRezerv, ",");

s1 = "rez " + Lex.strPLexicalUnit; int b = 0;

if (htl.AddLexicalUnit(Lex.strPLexicalUnit, 1, ref b))

{

TablesToMemo(this, e);

}

break;

}

case (TToken.lxmmul):

{

hashFunction.AddWord(hashTableRezerv, "\_");

s1 = "rez " + Lex.strPLexicalUnit; int b = 0;

if (htl.AddLexicalUnit(Lex.strPLexicalUnit, 1, ref b))

{

TablesToMemo(this, e);

}

break;

}

}

String m = "(" + s + "" + s1 + ")";

tbFMessage.Text += m;

}

}

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 searchBtn\_Click\_1(object sender, EventArgs e)

{

if (hashFunction.SearchWord(hashTableIdentifier, listBox1.SelectedItem.ToString()) == 1)

{

searchBtn.BackColor = Color.Green;

}

else

{

searchBtn.BackColor = Color.Red;

}

}

private void addBtn\_Click\_1(object sender, EventArgs e)

{

hashFunction.AddWord(hashTableIdentifier, textBox1.Text.ToString());

}

private void deleteBtn\_Click\_1(object sender, EventArgs e)

{

if (hashFunction.RemoveWord(hashTableIdentifier, listBox1.SelectedItem.ToString()))

{

deleteBtn.BackColor = Color.Green;

}

else

{

deleteBtn.BackColor = Color.Red;

}

}

private void changeBtn\_Click\_1(object sender, EventArgs e)

{

if (hashFunction.RemoveWord(hashTableIdentifier, listBox1.SelectedItem.ToString()))

{

hashFunction.AddWord(hashTableIdentifier, textBox1.Text.ToString());

changeBtn.BackColor = Color.Green;

}

else

{

changeBtn.BackColor = Color.Red;

}

}

private void reloadBtn\_Click\_1(object sender, EventArgs e)

{

listBox1.Items.Clear();

listBox2.Items.Clear();

listBox3.Items.Clear();

foreach (var entry in hashTableIdentifier)

{

listBox1.Items.Add(string.Join(", ", entry.Value));

}

foreach (var entry in hashTableDigital)

{

listBox2.Items.Add(string.Join(", ", entry.Value));

}

foreach (var entry in hashTableRezerv)

{

listBox3.Items.Add(string.Join(", ", entry.Value));

}

}

}

}

