**Лабораторная работа №5**

Выполнила: студент группы 4309

Курбанов Д.И.

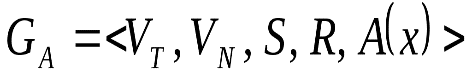
*Атрибутная грамматика* – это КС-грамматика, с узлами дерева вывода которой связаны атрибуты (семантические правила). КС-правилам сопоставляются правила вычисления атрибутов. Правило вычисления значений атрибутов, соответствующее данному КС-правилу, применяется для всех вхождений этого правила в дерево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

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp4

{

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();

TToken first;

bool init;

TreeView tree;

public uSyntAnalyzer(TreeView treeView)

{

tree = treeView;

}

public void S(TreeNode highParent)

{

TreeNode parent = new TreeNode("S");

highParent.Nodes.Add(parent);

Lex.NextToken();

if (Lex.enumPToken == TToken.lxmNumber)

{

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

Lex.NextToken();

A(parent);

S(parent);

}

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

{

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

Lex.NextToken();

}

else throw new Exception("Ожидался число или идентификатор");

// throw new Exception("Конец слова, текст верный.");

}

public void A(TreeNode highParent)

{

TreeNode parent = new TreeNode("A");

highParent.Nodes.Add(parent);

Lex.NextToken();

if (Lex.enumPToken == TToken.lxmIdentifier)

{

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

Lex.NextToken();

S(parent);

A(parent);

}

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

{

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

Lex.NextToken();

}

else throw new Exception("Ожидался число или идентификатор");

}

}

}

Form1.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 WindowsFormsApp4

{

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 Form1()

{

InitializeComponent();

//textBox2.AppendText("010000001 010000001 caaaa");

textBox2.AppendText("010000001 caaa cab 010000001 caaaa");

int n = textBox2.Lines.Length;

}

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

{

}

private void Form1\_Load\_1(object sender, EventArgs e)

{

}

private void label5\_Click(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

textBox4.Clear();

uSyntAnalyzer Synt = new uSyntAnalyzer(SyntTree);

Synt.Lex.strPSource = textBox2.Lines;

Synt.Lex.strPMessage = textBox4.Lines;

Synt.Lex.enumPState = TState.Start;

Synt.Lex.hashTableRezerv = hashTableRezerv;

try

{

TreeNode parent = new TreeNode("Tree");

SyntTree.Nodes.Add(parent);

Synt.S(parent);

Synt.Lex.NextToken();

throw new Exception("Текст верный");

}

catch (Exception exc)

{

textBox4.Text += exc.Message;

textBox2.Select();

textBox2.SelectionStart = 0;

int n = 0;

for (int i = 0; i < Synt.Lex.intPSourceRowSelection; i++) n += textBox2.Lines[i].Length + 2;

n += Synt.Lex.intPSourceColSelection;

textBox2.SelectionLength = n;

}

}

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

{

CLex Lex = new CLex();

Lex.strPSource = textBox2.Lines;

Lex.strPMessage = textBox4.Lines;

Lex.intPSourceColSelection = 0;

Lex.intPSourceRowSelection = 0;

int x = textBox2.TextLength;

int y = textBox2.Lines.Length;

textBox4.Text = "";

try

{

while (Lex.enumPState != TState.Finish)

{

Lex.NextToken();

string s1 = "", s = "";

switch (Lex.enumPToken)

{

case TToken.lxmIdentifier:

{

hashFunction.AddWord(hashTableIdentifier, Lex.strPLexicalUnit);

listBox4.Items.Add(Lex.strPLexicalUnit);

s1 = "id " + Lex.strPLexicalUnit;

break;

}

case TToken.lxmNumber:

{

hashFunction.AddWord(hashTableDigital, Lex.strPLexicalUnit);

listBox5.Items.Add(Lex.strPLexicalUnit);

s1 = "num " + Lex.strPLexicalUnit;

break;

}

case (TToken.lxmtz):

{

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

listBox6.Items.Add(Lex.strPLexicalUnit);

s1 = "rez \_" + Lex.strPLexicalUnit;

break;

}

}

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

textBox4.Text += m;

}

}

catch (Exception exc)

{

textBox4.Text += exc.Message;

textBox2.Select();

textBox2.SelectionStart = 0;

int n = 0;

for (int i = 0; i < Lex.intPSourceRowSelection; i++) n += textBox2.Lines[i].Length + 2;

n += Lex.intPSourceColSelection;

textBox2.SelectionLength = n;

}

}

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

{

if (hashFunction.SearchWord(hashTableIdentifier, listBox4.SelectedItem.ToString()) == 1)

{

button3.BackColor = Color.Green;

}

else

{

button3.BackColor = Color.Red;

}

}

private void button4\_Click(object sender, EventArgs e)

{

hashFunction.AddWord(hashTableIdentifier, textBox3.Text.ToString());

}

private void button5\_Click(object sender, EventArgs e)

{

if (hashFunction.RemoveWord(hashTableIdentifier, listBox4.SelectedItem.ToString()))

{

button5.BackColor = Color.Green;

}

else

{

button5.BackColor = Color.Red;

}

}

private void button6\_Click(object sender, EventArgs e)

{

if (hashFunction.RemoveWord(hashTableIdentifier, listBox4.SelectedItem.ToString()))

{

hashFunction.AddWord(hashTableIdentifier, textBox3.Text.ToString());

button6.BackColor = Color.Green;

}

else

{

button6.BackColor = Color.Red;

}

}

private void button7\_Click(object sender, EventArgs e)

{

listBox4.Items.Clear();

listBox5.Items.Clear();

listBox6.Items.Clear();

foreach (var entry in hashTableIdentifier)

{

listBox4.Items.Add(string.Join(", ", entry.Value));

}

foreach (var entry in hashTableDigital)

{

listBox5.Items.Add(string.Join(", ", entry.Value));

}

foreach (var entry in hashTableRezerv)

{

listBox6.Items.Add(string.Join(", ", entry.Value));

}

}

private void label7\_Click(object sender, EventArgs e)

{

}

private void listBox6\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

Результат программы:  
