МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний Технічний Університет України
“Київський політехнічний інститут”

Звіт з виконання лабораторної роботи №4
з предмету “Основи розробки трансляторів”
Тема: Генерація таблиці для висхідного розбору за заданою граматикою

Виконав студент 3 курсу
групи ТВ-11
Бодня Олексій

Київ 2013

Граматика

<app>::=	@interface <appName> ⌋ <list of definitions> ⌋ @implementation ⌋ <list of operators> ⌋ @end
<appName>::=	ID;
<list of definitions>::=	⌋ <definition2> | <list of definitions> ⌋ <definition2>
<definition2>::=	<definition>
<definition>::=	int <list of var>
<list of var>::=	, ID | , ID <list of var>
<list of operators>::=	⌋ <operator2> | <list of operators> ⌋ <operator2>
<operator2>::=	<operator>
<operator>::=	<setter> | <input> | <output> | <condition> | <cycle>
<setter>::=	ID = <expression>
<input>::=	input(<list of var>)
<output>::=	output(<list of var>)
<cycle>::=	for ID = <expression> to <expression> step <expression2> ⌋ <list of operators> ⌋ next
<expression2>::=	<expression>
<condition>::=	if <logical expression> ⌋ <list of operators> ⌋ else ⌋ <list of operators> ⌋ endif
<logical expression>::=	<log.exp.lev1> | <logical expression> or <log.exp.lev1>
<log.exp.lev1>::=	<log.exp.lev2> | <log.exp.lev1> and <log.exp.lev2>
<log.exp.lev2>::=	<relation> | !<log.exp.lev2> | [<logical expression>]
<relation>::=	<expression> (> | >= | < | <= | equ | !=) <expression>
<expression>::=	(- | ∅) <term> | <expression> + <term> | <expression> - <term>
<term>::=	<multiplier> | <term> * <multiplier> | <term> / <multiplier>
<multiplier>::=	<expr.response> | <multiplier> ^ <expr.response>
<expr.response>::=	ID | CONST | (<expression>)

Код алгоритму розбору:

```
// prevLevelPrevTerm – Попередній елемент батьківського терміналу
// prevLevelNextTerm – Наступний елемент батьківського терміналу
// grammarPair – поточний термінал для розбору

public void RecursiveSetup(string prevLevelPrevTerm, GrammarPair grammarPair, string prevLevelNextTerm)
{
    List<string> grammarPairs = new List<string>();
    grammarPairs.Add(prevLevelPrevTerm);
    foreach (string pair in grammarPair.PartLexems)
    {
        grammarPairs.Add(pair);
    }
    grammarPairs.Add(prevLevelNextTerm);

    for (int i=1;i<grammarPairs.Count-1; i++)
    {
        string currentTerm = grammarPairs[i-1];
        string nextTerm = grammarPairs[i];

        // Чи ще не було встановлено зв'язок між попереднім та наступним терміналами ?
        if (ConnotialBetweenTerminals(currentTerm,nextTerm) == Connotial.NoConnotial)
        {
            // Зв'язок “менше” між попереднім батьківським терміналом та першим дочірнім
            if (i==1)
            {
                SetConnotialBetweenTerminals(Connotial.LessConnotial,currentTerm,nextTerm);
            }
            // Зв'язок “дорівнює” між двома сусідніми терміналами
            else
            {
                SetConnotialBetweenTerminals(Connotial.EqualConnotial,currentTerm,nextTerm);
            }
            // Перевірка наступного терміналу на наявність в нього дочірніх терміналів
            if (this.grammar.GrammarPairWithRootLexem(nextTerm) != null)
            {
                // ЧерезНаступний термінал (якщо він останній, то встановиться наступний термінал від батька)
                string nextNextTerm = grammarPairs[i+1];
                // Список дочірніх терміналів в наступного термінала
                List<GrammarPair> pairs = this.grammar.GrammarPairWithRootLexem(nextTerm);
                foreach (GrammarPair pair in pairs)
                {
                    // Розглядання підтерміналів наступного термінала
                    RecursiveSetup(currentTerm,pair,nextNextTerm);
                    // Встановлення відношення “більше” для останнього термінала
                    string pairLastTerm = pair.PartLexems[pair.PartLexems.Count-1];
                    SetConnotialBetweenTerminals(Connotial.GreaterConnotial,pairLastTerm,nextNextTerm);
                }
            }
        }
    }
}

// RootLexem – батьківський термінал
// PartLexems – масив дочірніх терміналів
public class GrammarPair
{
    public string RootLexem;
    public List<string> PartLexems;
}
```