МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Институт информационных технологий

Кафедра «Информационные технологии и компьютерные системы»

ОТЧЕТ

по лабораторной работе №7

по дисциплине “Системное программное обеспечение”

Вариант 4

Выполнил:
ст. гр. ИТ/б-22-6-о Донец Н.О.
Принял:
ассистент Ткаченко К.С.

Севастополь
2024 г.

Цель работы:

Изучить восходящий синтаксический анализ методом LR, а также способы построения синтаксических анализаторов.

Задание:

Разработать и отладить программу восходящего синтаксического анализатора методом LR.

Грамматика языка Logic4:

<программа>::=<блок>

<блок>::=<оператор>|<оператор>;< блок >

<оператор>:=<переменная>:=<выражение>

<оператор>:= if <переменная> ? <оператор> : <оператор>

<выражение>::=<фактор>|<выражение>#<фактор>

<фактор>::=<первичное>|<фактор>&<первичное>

<первичное>::=<идент.>|<константа>|(<выражение>)

<константа>::=<целая константа>

<целая константа>::=<число>

<число>::=<цифра>|<число><цифра>

<цифра>::=0|1|2|3|4|5|6|7|8|9

<идент.>::=<буква>|<идент.><буква>

<буква>::=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

Ход работы:

Для того чтобы можно было воспользоваться LR методом необходимо разметить порождающие правила грамматики номерами состояний анализатора, а также заменить правую рекурсию в выражении и факторе на левую. Для простоты и удобства будут использованы некоторые сокращения.

1) $S \rightarrow {}_{1} B_{2}$
2) $B \rightarrow {}_{1,4} O_{3} ; {}_{4} B_{5}$
3) $B \rightarrow {}_{1,4} O_{3}$
4) $O \rightarrow {}_{1,4,11,13} I_{6} A_{7} E_{8}$
5) $O \rightarrow {}_{1,4,11,13} T_{9} I_{10} ?_{11} O_{12} :_{13} O_{14}$
6) $E \rightarrow {}_{7,16,23} F_{15} \#_{16} E_{17}$
7) $E \rightarrow {}_{7,16,23} F_{15}$
8) $F \rightarrow {}_{7,16,19,23} P_{18} \&_{19} F_{20}$
9) $F \rightarrow {}_{7,16,19,23} P_{18}$
10) $P \rightarrow {}_{7,16,19,23} I_{21}$
11) $P \rightarrow {}_{7,16,19,23} C_{22}$
12) $P \rightarrow {}_{7,16,19,23} (_{23} E_{24})_{25}$

Была разработана управляющая таблица для LR анализатора (таблица 1).

Таблица 1 – Управляющая таблица

	S	B	O	E	F	P	T	I	C	A	;	?	:	(	)	#	&	$
1	доп уск	сдв, 2	сдв, 3				сдв, 9	сдв, 6										
2																		1
3											сдв, 4							3
4		сдв, 5	сдв, 3				сдв, 9	сдв, 6										
5																		2
6										сдв, 7								

7				сдв, 8	сдв, 15	сдв. 18		сдв, 21	сдв, 22					сдв, 23				
8											4		4					4
9								сдв, 10										
10												сдв, 11						
11			сдв, 12				сдв, 9	сдв, 6										
12													сдв, 13					
13			сдв, 14				сдв, 9	сдв, 6										
14											5		5					5
15											7		7		7	сдв, 16		7
16				сдв, 17	сдв, 15	сдв, 18		сдв, 21	сдв, 22					сдв, 23				
17											6		6		6			6
18											9		9		9	9	сдв, 19	9
19					сдв, 20	сдв, 18		сдв, 21	сдв, 22					сдв, 23				
20											8		8		8	8		8
21											10		10		10	10	10	10
22											11		11		11	11	11	11
23				сдв, 24	сдв, 15	сдв, 18		сдв, 21	сдв, 22					сдв, 23				
24															сдв, 25			
25											12		12		12	12	12	12

Был разработан класс LR анализатора, решающий поставленную задачу (Листинг 1).

Листинг 1 – LR анализатор

```
#ifndef LRANALYZER
#define LRANALYZER

#include "../lr6/SysAnalyzer.h"

#include <map>
#include <iostream>
#include <stack>

using namespace std;

class LRAnalyzer {
    map<pair<unsigned short, char>, pair<char, size_t>> rules;
    map<pair<unsigned short, char>, unsigned short> shifts;
    stack<char> inputStack;
    stack<char> outputStack;
```

```
    stack<pair<char, unsigned short>> workingStack;

    void SetRules();
    void SetShifts();
    void ClearStacks();
    void FillInputStack(string);

    public:
    LRAnalyzer();
    bool Analyze(string);
};

#endif
#include "LRAnalyzer.h"

bool LRAnalyzer::Analyze(string str) {
    ClearStacks();
    FillInputStack(str);

    workingStack.push(make_pair('$', 1));

    while (workingStack.top().second != 0) {
        cout<<"State: "<< workingStack.top().second << " IS top: " <<
inputStack.top() << endl;
        if (rules.find(make_pair(workingStack.top().second,
inputStack.top())) != rules.end()) {
            auto rule = rules[make_pair(workingStack.top().second,
inputStack.top())];
            for (int i = 0; i < rule.second; i++) {
                workingStack.pop();
            }
            inputStack.push(rule.first);
        } else if (shifts.find(make_pair(workingStack.top().second,
inputStack.top())) != shifts.end()) {
            unsigned short state =
shifts[make_pair(workingStack.top().second, inputStack.top())];
            outputStack.push(inputStack.top());
            workingStack.push(make_pair(inputStack.top(), state));
            inputStack.pop();
        } else {
            return false;
        }
    }
    return true;
}

void LRAnalyzer::FillInputStack(string str) {
    inputStack.push('$');
    for (int i = str.size() - 1; i >= 0; i--) {
        inputStack.push(str[i]);
    }
}

void LRAnalyzer::SetRules() {
    //rule 1
    rules.emplace(make_pair(2, '$'), make_pair('S', 1));
    //rule 2
    rules.emplace(make_pair(5, '$'), make_pair('B', 3));
    //rule 3
    rules.emplace(make_pair(3, '$'), make_pair('B', 1));
    //rule 4
    rules.emplace(make_pair(8, ';'), make_pair('O', 3));
    rules.emplace(make_pair(8, ':'), make_pair('O', 3));
```

```
    rules.emplace(make_pair(8, '$'), make_pair('O', 3));
    //rule 5
    rules.emplace(make_pair(14, ';'), make_pair('O', 6));
    rules.emplace(make_pair(14, ':'), make_pair('O', 6));
    rules.emplace(make_pair(14, '$'), make_pair('O', 6));
    //rule 6
    rules.emplace(make_pair(17, ';'), make_pair('E', 3));
    rules.emplace(make_pair(17, ':'), make_pair('E', 3));
    rules.emplace(make_pair(17, '$'), make_pair('E', 3));
    rules.emplace(make_pair(17, ')'), make_pair('E', 3));
    //rule 7
    rules.emplace(make_pair(15, ';'), make_pair('E', 1));
    rules.emplace(make_pair(15, ':'), make_pair('E', 1));
    rules.emplace(make_pair(15, '$'), make_pair('E', 1));
    rules.emplace(make_pair(15, ')'), make_pair('E', 1));
    //rule 8
    rules.emplace(make_pair(20, ';'), make_pair('F', 3));
    rules.emplace(make_pair(20, ':'), make_pair('F', 3));
    rules.emplace(make_pair(20, '$'), make_pair('F', 3));
    rules.emplace(make_pair(20, ')'), make_pair('F', 3));
    rules.emplace(make_pair(20, '#'), make_pair('F', 3));
    //rule 9
    rules.emplace(make_pair(18, ';'), make_pair('F', 1));
    rules.emplace(make_pair(18, ':'), make_pair('F', 1));
    rules.emplace(make_pair(18, '$'), make_pair('F', 1));
    rules.emplace(make_pair(18, ')'), make_pair('F', 1));
    rules.emplace(make_pair(18, '#'), make_pair('F', 1));
    //rule 10
    rules.emplace(make_pair(21, ';'), make_pair('P', 1));
    rules.emplace(make_pair(21, ':'), make_pair('P', 1));
    rules.emplace(make_pair(21, '$'), make_pair('P', 1));
    rules.emplace(make_pair(21, ')'), make_pair('P', 1));
    rules.emplace(make_pair(21, '#'), make_pair('P', 1));
    rules.emplace(make_pair(21, '&'), make_pair('P', 1));
    //rule 11
    rules.emplace(make_pair(22, ';'), make_pair('P', 1));
    rules.emplace(make_pair(22, ':'), make_pair('P', 1));
    rules.emplace(make_pair(22, '$'), make_pair('P', 1));
    rules.emplace(make_pair(22, ')'), make_pair('P', 1));
    rules.emplace(make_pair(22, '#'), make_pair('P', 1));
    rules.emplace(make_pair(22, '&'), make_pair('P', 1));
    //rule 12
    rules.emplace(make_pair(25, ';'), make_pair('P', 3));
    rules.emplace(make_pair(25, ':'), make_pair('P', 3));
    rules.emplace(make_pair(25, '$'), make_pair('P', 3));
    rules.emplace(make_pair(25, ')'), make_pair('P', 3));
    rules.emplace(make_pair(25, '#'), make_pair('P', 3));
    rules.emplace(make_pair(25, '&'), make_pair('P', 3));
}

void LRAnalyzer::SetShifts() {
    shifts.emplace(make_pair(1, 'S'), 0);
    shifts.emplace(make_pair(1, 'B'), 2);
    shifts.emplace(make_pair(1, 'O'), 3);
    shifts.emplace(make_pair(1, 'T'), 9);
    shifts.emplace(make_pair(1, 'I'), 6);

    shifts.emplace(make_pair(3, ';'), 4);

    shifts.emplace(make_pair(4, 'B'), 5);
    shifts.emplace(make_pair(4, 'O'), 3);
    shifts.emplace(make_pair(4, 'T'), 9);
    shifts.emplace(make_pair(4, 'I'), 6);
```

```
    shifts.emplace(make_pair(6, 'A'), 7);

    shifts.emplace(make_pair(7, 'E'), 8);
    shifts.emplace(make_pair(7, 'F'), 15);
    shifts.emplace(make_pair(7, 'P'), 18);
    shifts.emplace(make_pair(7, 'I'), 21);
    shifts.emplace(make_pair(7, 'C'), 22);
    shifts.emplace(make_pair(7, '('), 23);

    shifts.emplace(make_pair(9, 'I'), 10);

    shifts.emplace(make_pair(10, '?'), 11);

    shifts.emplace(make_pair(11, 'O'), 12);
    shifts.emplace(make_pair(11, 'T'), 9);
    shifts.emplace(make_pair(11, 'I'), 6);

    shifts.emplace(make_pair(12, ':'), 13);

    shifts.emplace(make_pair(13, 'O'), 14);
    shifts.emplace(make_pair(13, 'T'), 9);
    shifts.emplace(make_pair(13, 'I'), 6);

    shifts.emplace(make_pair(15, '#'), 16);

    shifts.emplace(make_pair(16, 'F'), 15);
    shifts.emplace(make_pair(16, 'P'), 18);
    shifts.emplace(make_pair(16, 'I'), 21);
    shifts.emplace(make_pair(16, 'C'), 22);
    shifts.emplace(make_pair(16, '('), 23);
    shifts.emplace(make_pair(16, 'E'), 17);

    shifts.emplace(make_pair(18, '&'), 19);

    shifts.emplace(make_pair(19, 'F'), 20);
    shifts.emplace(make_pair(19, 'P'), 18);
    shifts.emplace(make_pair(19, 'I'), 21);
    shifts.emplace(make_pair(19, 'C'), 22);
    shifts.emplace(make_pair(19, '('), 23);

    shifts.emplace(make_pair(23, 'F'), 15);
    shifts.emplace(make_pair(23, 'P'), 18);
    shifts.emplace(make_pair(23, 'I'), 21);
    shifts.emplace(make_pair(23, 'C'), 22);
    shifts.emplace(make_pair(23, '('), 23);
    shifts.emplace(make_pair(23, 'E'), 24);

    shifts.emplace(make_pair(24, ')'), 25);
}

LRAnalyzer::LRAnalyzer() {
    SetRules();
    SetShifts();
}

void LRAnalyzer::ClearStacks() {
    while (!inputStack.empty()) inputStack.pop();
    while (!outputStack.empty()) outputStack.pop();
    while (!workingStack.empty()) workingStack.pop();
}
```

Также были проведены тесты работы анализатора (рисунок 1).

```
if a?a:=c:a=(b); a:=g
TI?IAI:IA(I);IAI
State: 1 IS top: T
State: 9 IS top: I
State: 10 IS top: ?
State: 11 IS top: I
State: 6 IS top: A
State: 7 IS top: I
State: 21 IS top: :
State: 7 IS top: P
State: 18 IS top: :
State: 7 IS top: F
State: 15 IS top: :
State: 7 IS top: E
State: 8 IS top: :
State: 11 IS top: O
State: 12 IS top: :
State: 13 IS top: I
State: 6 IS top: A
State: 7 IS top: (
State: 23 IS top: I
State: 21 IS top: )
State: 23 IS top: P
State: 18 IS top: )
State: 23 IS top: F
State: 15 IS top: )
State: 23 IS top: E
State: 24 IS top: )
State: 25 IS top: ;
State: 7 IS top: P
State: 18 IS top: ;
State: 7 IS top: F
State: 15 IS top: ;
State: 7 IS top: E
State: 8 IS top: ;
State: 13 IS top: O
State: 14 IS top: ;
State: 1 IS top: O
State: 3 IS top: ;
State: 4 IS top: I
State: 6 IS top: A
State: 7 IS top: I
State: 21 IS top: $
State: 7 IS top: P
State: 18 IS top: $
State: 7 IS top: F
State: 15 IS top: $
State: 7 IS top: E
State: 8 IS top: $
State: 4 IS top: O
State: 3 IS top: $
State: 4 IS top: B
State: 5 IS top: $
State: 1 IS top: B
State: 2 IS top: $
State: 1 IS top: S
The string belongs to the language

a=12
IAC
State: 1 IS top: I
State: 6 IS top: A
State: 7 IS top: C
State: 22 IS top: $
State: 7 IS top: P
State: 18 IS top: $
State: 7 IS top: F
State: 15 IS top: $
State: 7 IS top: E
State: 8 IS top: $
State: 1 IS top: O
State: 3 IS top: $
State: 1 IS top: B
State: 2 IS top: $
State: 1 IS top: S
The string belongs to the language
```

```
if e?d=d#d#(d&c):a=5
TI?IAI#I#(I&I):IAC
State: 1 IS top: T
State: 9 IS top: I
State: 10 IS top: ?
State: 11 IS top: I
State: 6 IS top: A
State: 7 IS top: I
State: 21 IS top: #
State: 7 IS top: P
State: 18 IS top: #
State: 7 IS top: F
State: 15 IS top: #
State: 16 IS top: I
State: 21 IS top: #
State: 16 IS top: P
State: 18 IS top: #
State: 16 IS top: F
State: 15 IS top: #
State: 16 IS top: (
State: 23 IS top: I
State: 21 IS top: &
State: 23 IS top: P
State: 18 IS top: &
State: 19 IS top: I
State: 21 IS top: )
State: 19 IS top: P
State: 18 IS top: )
State: 19 IS top: F
State: 20 IS top: )
State: 23 IS top: F
State: 15 IS top: )
State: 23 IS top: E
State: 24 IS top: )
State: 25 IS top: :
State: 16 IS top: P
State: 18 IS top: :
State: 16 IS top: F
State: 15 IS top: :
State: 16 IS top: E
State: 17 IS top: :
State: 16 IS top: E
State: 17 IS top: :
State: 7 IS top: E
State: 8 IS top: :
State: 11 IS top: O
State: 12 IS top: :
State: 13 IS top: I
State: 6 IS top: A
State: 7 IS top: C
State: 22 IS top: $
State: 7 IS top: P
State: 18 IS top: $
State: 7 IS top: F
State: 15 IS top: $
State: 7 IS top: E
State: 8 IS top: $
State: 13 IS top: O
State: 14 IS top: $
State: 1 IS top: O
State: 3 IS top: $
State: 1 IS top: B
State: 2 IS top: $
State: 1 IS top: S
The string belongs to the language

if abs?c-d
No operator matches: -
TI?I
State: 1 IS top: T
State: 9 IS top: I
State: 10 IS top: ?
State: 11 IS top: I
State: 6 IS top: $
The string does not belong to the language
```

Рисунок 1 – Тесты

Выводы

В ходе лабораторной работы был изучен LR метод, а также были изучены способы построения синтаксических анализаторов.