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 → 1 B 2

2) B → 1,4 O 3 ; 4 B 5

3) B → 1,4 O 3

4) O → 1,4,11,13 I 6 A 7 E 8

5) O → 1,4,11,13 T 9 I 10 ? 11 O 12 : 13 O 14

6) E → 7,16,23 F 15 # 16 E 17

7) E → 7,16,23 F 15

8) F → 7,16,19,23 P 18 & 19 F 20

9) F → 7,16,19,23 P 18

10) P → 7,16,19,23 I 21

11) P → 7,16,19,23 C 22

12) P → 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 | допуск | 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).

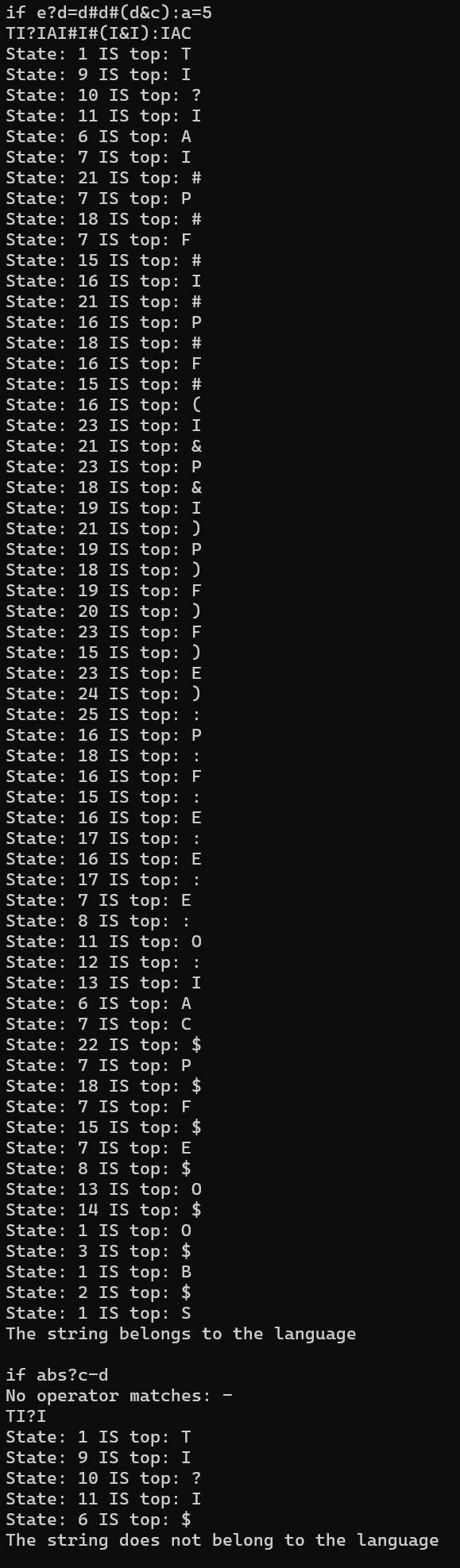
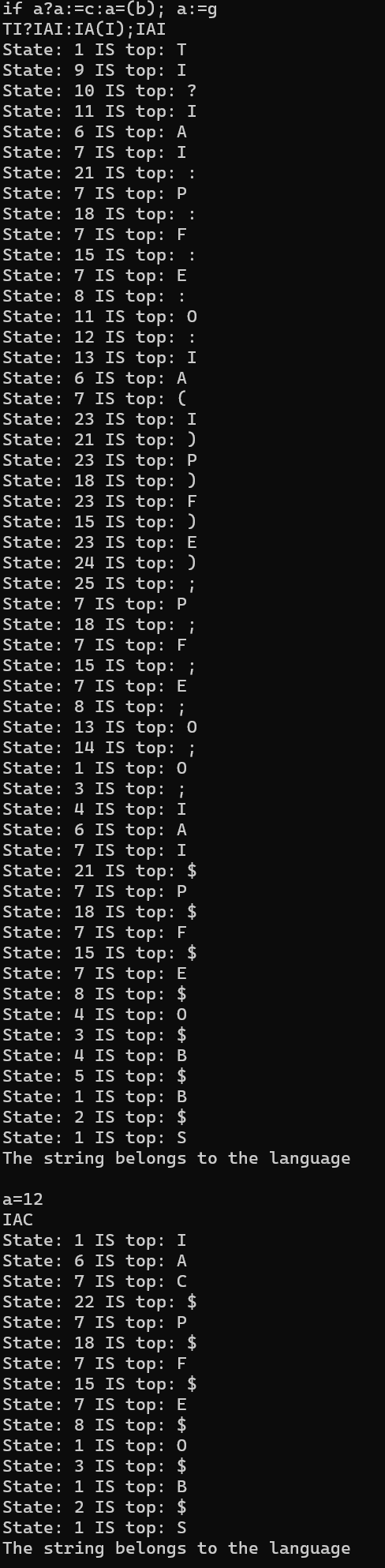


Рисунок 1 – Тесты

**Выводы**

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