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 №5

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 метод рекурсивного спуска, а также способы построения синтаксических анализаторов.

Задание:

Разработать и отладить программу нисходящего синтаксического анализатора методом LL(1).

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 LL методом необходимо выполнить некоторые преобразования заданной по варианту грамматики.

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

Обозначим блок B, оператор O, тогда исходное правило имеет вид:

B → O | O;B

Далее проведём необходимые преобразования.

B → OX

X → ;B

X → lambda

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

Обозначим выражение E, фактор F, тогда исходное правило имеет вид:

E → F | E#F

Далее проведём необходимые преобразования.

E → FY

Y → #E

Y → lambda

Те же манипуляции проводим с фактором.

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

Обозначим первичное за P, тогда исходное правило имеет вид:

F → P | F&P

Далее проведём необходимые преобразования.

F → PZ

Z → &F

Z → lambda

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

Таблица 1 – Управляющая таблица

	I	C	T	A	?	:	#	&	(	)	;	$
S	1		1									
B	2		2									
O	5		6									
T			сдвиг									
I	сдвиг											
E	7	7							7			
F	10	10							10			
P	13	14							15			
X											3	4
Y						9	8			9	9	9
Z						12	12	11		12	12	12
A				сдвиг								
C		сдвиг										
?					сдвиг							
:						сдвиг						
#							сдвиг					
&								сдвиг				
(									сдвиг			
)										сдвиг		
;											сдвиг	
$												сдвиг

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

Листинг 1 – LL(1) анализатор

```
#ifndef LLANALYZER
#define LLANALYZER
#include <iostream>
#include <vector>
#include <map>
#include <stack>
#include <windows.h>
using namespace std;
class LLAnalyzer {
    map<pair<char, char>, int> table;
    map<int, pair<string, string>> rules;
    int head;
    stack<char> workingStack;
    stack<char> outputStack;
    void SetTable();
    void SetRules();
    void DropStacks();
    bool ProcessInputSymbol(char s);
    void UseRule(int ruleNumber);
    void Shift();
```

```
    void PrintOutputStack();
    public:
    LLAnalyzer();
    bool Analyze(string _input);
};
#endif
#include "LLAnalyzer.h"
void LLAnalyzer::DropStacks() {
    while (!workingStack.empty()) workingStack.pop();
    while (!outputStack.empty()) outputStack.pop();
}
void LLAnalyzer::SetTable() {
    table.emplace(make_pair('S', 'I'), 1);
    table.emplace(make_pair('S', 'T'), 1);
    table.emplace(make_pair('B', 'T'), 2);
    table.emplace(make_pair('B', 'I'), 2);
    table.emplace(make_pair('O', 'I'), 5);
    table.emplace(make_pair('O', 'T'), 6);
    table.emplace(make_pair('T', 'T'), 16);
    table.emplace(make_pair('I', 'I'), 17);
    table.emplace(make_pair('?', '?'), 18);
    table.emplace(make_pair(':', ':'), 19);
    table.emplace(make_pair('A', 'A'), 20);
    table.emplace(make_pair('X', ';'), 3);
    table.emplace(make_pair('X', '$'), 4);
    table.emplace(make_pair(';', ';'), 21);
    table.emplace(make_pair('$', '$'), 22);
    table.emplace(make_pair('E', 'I'), 7);
    table.emplace(make_pair('E', 'C'), 7);
    table.emplace(make_pair('E', '('), 7);
    table.emplace(make_pair('F', 'I'), 10);
    table.emplace(make_pair('F', 'C'), 10);
    table.emplace(make_pair('F', '('), 10);
    table.emplace(make_pair('Y', '#'), 8);
    table.emplace(make_pair('Y', '$'), 9);
    table.emplace(make_pair('Y', ':'), 9);
    table.emplace(make_pair('Y', ';'), 9);
    table.emplace(make_pair('Y', ')'), 9);
    table.emplace(make_pair('#', '#'), 23);
    table.emplace(make_pair('P', 'I'), 13);
    table.emplace(make_pair('P', 'C'), 14);
    table.emplace(make_pair('P', '('), 15);
    table.emplace(make_pair('(', '('), 24);
    table.emplace(make_pair(')', ')'), 25);
    table.emplace(make_pair('Z', '&'), 11);
    table.emplace(make_pair('Z', '$'), 12);
    table.emplace(make_pair('Z', ':'), 12);
    table.emplace(make_pair('Z', ';'), 12);
    table.emplace(make_pair('Z', ')'), 12);
    table.emplace(make_pair('Z', '#'), 12);
    table.emplace(make_pair('C', 'C'), 26);
    table.emplace(make_pair('&', '&'), 27);
}
void LLAnalyzer::SetRules() {
    rules.emplace(1, make_pair("S", "B"));
    rules.emplace(2, make_pair("B", "OX"));
    rules.emplace(3, make_pair("X", ";B"));
    rules.emplace(4, make_pair("X", "lambda"));
    rules.emplace(5, make_pair("O", "IAE"));
    rules.emplace(6, make_pair("O", "TI?O:O"));
    rules.emplace(7, make_pair("E", "FY"));
    rules.emplace(8, make_pair("Y", "#E"));
    rules.emplace(9, make_pair("Y", "lambda"));
```

```
    rules.emplace(10, make_pair("F", "PZ"));
    rules.emplace(11, make_pair("Z", "&F"));
    rules.emplace(12, make_pair("Z", "lambda"));
    rules.emplace(13, make_pair("P", "I"));
    rules.emplace(14, make_pair("P", "C"));
    rules.emplace(15, make_pair("P", "(E)"));
    rules.emplace(16, make_pair("T", "0"));
    rules.emplace(17, make_pair("I", "0"));
    rules.emplace(18, make_pair("?", "0"));
    rules.emplace(19, make_pair(":", "0"));
    rules.emplace(20, make_pair("A", "0"));
    rules.emplace(21, make_pair(";", "0"));
    rules.emplace(22, make_pair("$", "0"));
    rules.emplace(23, make_pair("#", "0"));
    rules.emplace(24, make_pair("(", "0"));
    rules.emplace(25, make_pair(")", "0"));
    rules.emplace(26, make_pair("C", "0"));
    rules.emplace(27, make_pair("&", "0"));
}
LLAnalyzer::LLAnalyzer() {
    SetTable();
    SetRules();
}
bool LLAnalyzer::Analyze(string _input) {
    DropStacks();
    workingStack.push('$');
    workingStack.push('S');
    head = 0;
    _input += '$';
    while (head < _input.size()) {
        if (!ProcessInputSymbol(_input[head])) {
            cout<<"Symbol: "<<_input[head]<<endl;
            PrintOutputStack();
            return false;
        }
    }
    PrintOutputStack();
    return true;
}
bool LLAnalyzer::ProcessInputSymbol(char c) {
    int ruleNumber;
    if (table.find(make_pair(workingStack.top(), c)) != table.end()) {
        ruleNumber = table[make_pair(workingStack.top(), c)];
        if (rules[ruleNumber].second == "0") {
            cout<<"SHIFT WS TOP: "<< workingStack.top() << " C:" << c <<endl;
            Shift();
        }
        else if (rules[ruleNumber].second == "lambda") {
            cout<<"Lambda: "<<workingStack.top()<<endl;
            workingStack.pop();
        }
        else {
            cout<<"Use rule: "<<ruleNumber <<" WS TOP: "<<workingStack.top()
<< " C:" << c <<endl;
            UseRule(ruleNumber);
        }
        return true;
    }
    else {
        return false;
    }
}
void LLAnalyzer::UseRule(int ruleNumber) {
```

```
    char c = workingStack.top();
    outputStack.push(c);
    workingStack.pop();
    for (int j = rules[ruleNumber].second.size()-1; j >= 0 ; j--) {
        //cout<<"Add: "<<rules[ruleNumber].second[j]<<endl;
        workingStack.push(rules[ruleNumber].second[j]);
    }
}
void LLAnalyzer::Shift() {
    char c = workingStack.top();
    workingStack.pop();
    outputStack.push(c);
    head++;
}
void LLAnalyzer::PrintOutputStack() {
    cout<<"\nOutput stack: ";
    while (!outputStack.empty()) {
        cout<<outputStack.top();
        outputStack.pop();
    }
    cout<<'\n';
}
```

Также были проведены тесты работы анализатора (Рисунок 1).

```
if e?d=d#d#(d&c):a=5
TI?IAI#I#(I&I):IAC
Use rule: 1 WS TOP: S C:T
Use rule: 2 WS TOP: B C:T
Use rule: 6 WS TOP: O C:T
SHIFT WS TOP: T C:T
SHIFT WS TOP: I C:I
SHIFT WS TOP: ? C:?
Use rule: 5 WS TOP: O C:I
SHIFT WS TOP: I C:I
SHIFT WS TOP: A C:A
Use rule: 7 WS TOP: E C:I
Use rule: 10 WS TOP: F C:I
Use rule: 13 WS TOP: P C:I
SHIFT WS TOP: I C:I
Lambda: Z
Use rule: 8 WS TOP: Y C:#
SHIFT WS TOP: # C:#
Use rule: 7 WS TOP: E C:I
Use rule: 10 WS TOP: F C:I
Use rule: 13 WS TOP: P C:I
SHIFT WS TOP: I C:I
Lambda: Z
Use rule: 8 WS TOP: Y C:#
SHIFT WS TOP: # C:#
Use rule: 7 WS TOP: E C:(
Use rule: 10 WS TOP: F C:(
Use rule: 15 WS TOP: P C:(
SHIFT WS TOP: ( C:(
Use rule: 7 WS TOP: E C:I
Use rule: 10 WS TOP: F C:I
Use rule: 13 WS TOP: P C:I
SHIFT WS TOP: I C:I
Use rule: 11 WS TOP: Z C:&
SHIFT WS TOP: & C:&
Use rule: 10 WS TOP: F C:I
Use rule: 13 WS TOP: P C:I
SHIFT WS TOP: I C:I
Lambda: Z
Lambda: Y
SHIFT WS TOP: ) C:)
Lambda: Z
Lambda: Y
SHIFT WS TOP: ; C:;
Use rule: 5 WS TOP: O C:I
SHIFT WS TOP: I C:I
SHIFT WS TOP: A C:A
Use rule: 7 WS TOP: E C:C
Use rule: 10 WS TOP: F C:C
Use rule: 14 WS TOP: P C:C
SHIFT WS TOP: C C:C
Lambda: Z
Lambda: Y
Lambda: X
SHIFT WS TOP: $ C:$

Output stack: $CPFEAIO:)IPF&ZIPFE(PFE#YIPFE#YIPFEAIO?ITOBS
The string belongs to the language

if abs?c-d
No operator matches: -
TI?I
Use rule: 1 WS TOP: S C:T
Use rule: 2 WS TOP: B C:T
Use rule: 6 WS TOP: O C:T
SHIFT WS TOP: T C:T
SHIFT WS TOP: I C:I
SHIFT WS TOP: ? C:?
Use rule: 5 WS TOP: O C:I
SHIFT WS TOP: I C:I
Symbol: $

Output stack: IO?ITOBS
The string does not belong to the language
```

```
$ ./testAnalyzer.exe
if a?a:=c:a=(b); a:=g
TI?IAI:IA(I);IAI
Use rule: 1 WS TOP: S C:T
Use rule: 2 WS TOP: B C:T
Use rule: 6 WS TOP: O C:T
SHIFT WS TOP: T C:T
SHIFT WS TOP: I C:I
SHIFT WS TOP: ? C:?
Use rule: 5 WS TOP: O C:I
SHIFT WS TOP: I C:I
SHIFT WS TOP: A C:A
Use rule: 7 WS TOP: E C:I
Use rule: 10 WS TOP: F C:I
Use rule: 13 WS TOP: P C:I
SHIFT WS TOP: I C:I
Lambda: Z
Lambda: Y
SHIFT WS TOP: : C::
Use rule: 5 WS TOP: O C:I
SHIFT WS TOP: I C:I
SHIFT WS TOP: A C:A
Use rule: 7 WS TOP: E C:(
Use rule: 10 WS TOP: F C:(
Use rule: 15 WS TOP: P C:(
SHIFT WS TOP: ( C:(
Use rule: 7 WS TOP: E C:I
Use rule: 10 WS TOP: F C:I
Use rule: 13 WS TOP: P C:I
SHIFT WS TOP: I C:I
Lambda: Z
Lambda: Y
SHIFT WS TOP: ) C:)
Lambda: Z
Lambda: Y
Use rule: 3 WS TOP: X C:;
SHIFT WS TOP: ; C:;
Use rule: 2 WS TOP: B C:I
Use rule: 5 WS TOP: O C:I
SHIFT WS TOP: I C:I
SHIFT WS TOP: A C:A
Use rule: 7 WS TOP: E C:I
Use rule: 10 WS TOP: F C:I
Use rule: 13 WS TOP: P C:I
SHIFT WS TOP: I C:I
Lambda: Z
Lambda: Y
Lambda: X
SHIFT WS TOP: $ C:$

Output stack: $IPFEAIOB;X)IPFE(PFEAIO:IPFEAIO?ITOBS
The string belongs to the language

a=12
IAC
Use rule: 1 WS TOP: S C:I
Use rule: 2 WS TOP: B C:I
Use rule: 5 WS TOP: O C:I
SHIFT WS TOP: I C:I
SHIFT WS TOP: A C:A
Use rule: 7 WS TOP: E C:C
Use rule: 10 WS TOP: F C:C
Use rule: 14 WS TOP: P C:C
SHIFT WS TOP: C C:C
Lambda: Z
Lambda: Y
Lambda: X
SHIFT WS TOP: $ C:$

Output stack: $CPFEAIOBS
The string belongs to the language
```

Рисунок 1 – Тесты

Выводы

В ходе лабораторной работы был изучен LL(1) метод, а также были изучены способы построения синтаксических анализаторов.