

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 №6
по дисциплине «Системное программное обеспечение»
Вариант 4

Выполнил:

ст. гр. ИТ/б-22-б-о Донец Н.О.

Принял:

ассистент Ткаченко К.С.

Севастополь

2024 г.

Цель работы:

Изучить восходящий синтаксический анализ методом предшествования, 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 выполнить некоторые преобразования заданной по варианту грамматики.

$\langle \text{блок} \rangle ::= \langle \text{оператор} \rangle | \langle \text{оператор} \rangle ; \langle \text{блок} \rangle$

Обозначим блок В, оператор О, тогда исходное правило имеет вид:

$V \rightarrow O \mid O;V$

Далее проведём необходимые преобразования.

$V \rightarrow X \mid X;V$

$X \rightarrow O$

Также заменяем правую рекурсию в выражении и факторе на левую.

Была разработана матрица отношений предшествования для LR анализатора (таблица 1).

Таблица 1 – Матрица отношений предшествования

	S	B	X	O	T	I	E	F	P	A	C	?	:	#	&	()	;	\$
S																			>
B																			>
X																		=	>
O													=					>	>
T						=													
I										=		=	>	>	>		>	>	>
E													>				=	>	>
F													>	=			>	>	>
P													>	>	=		>	>	>
A						<	=	<	<		<					<			
C													>	>	>		>	>	>
?				=	<	<													
:				=	<	<													
#						<	=	<	<		<					<			
&						<		=	<		<					<			
(<	=	<	<		<					<			
)													>	>	>		>	>	>
;		=	<	<	<	<													
\$	<	<	<	<	<	<													

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

Листинг 1 – LR анализатор

```
#ifndef SYSANALYZER
#define SYSANALYZER

#include <iostream>
#include <map>
#include <stack>
#include <windows.h>
#include <vector>

using namespace std;

class SysAnalyzer {
    stack<char> inputStack;
    stack<char> outputStack;
    stack<pair<char, char>> workingStack;
    map<pair<char, char>, char> relationMatrix;
    map<string, char> rules;

    void FillInputStack(string);
    void SetRelationMatrix();
    void ClearStacks();
    void SetRules();
    bool CheckRelation();
    void UseRule();
    void PrintOutputStack();

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

#endif
```

```

#include "SysAnalyzer.h"

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

    workingStack.push(make_pair('$', '0'));

    while ((inputStack.top() != '$') || (workingStack.top().first != '$')) {
        if(!CheckRelation()) return false;
    }

    return true;
}

bool SysAnalyzer::CheckRelation() {
    cout<< "Check Relation ";
    //working stack top
    char wst = workingStack.top().first;
    //input stack top
    char ist = inputStack.top();

    cout << wst << ' ' << ist;

    if (relationMatrix.find(make_pair(wst, ist)) != relationMatrix.end() ) {
        char relevance = relationMatrix[make_pair(wst, ist)];
        cout << ' ' << relevance << endl;
        if (relevance == '<') {
            outputStack.push(ist);
            workingStack.push(make_pair(ist, relevance));
            inputStack.pop();
        }
        else if (relevance == '=') {
            outputStack.push(ist);
            workingStack.push(make_pair(ist, relevance));
            inputStack.pop();
        }
    }
}

```

```

        else if (relevance == '>') {
            UseRule();
        }
    }
    else {
        return false;
    }

    return true;
}

void SysAnalyzer::UseRule() {
    string str = "";
    while (workingStack.top().second != '<') {
        str = workingStack.top().first + str;
        workingStack.pop();
    }
    str = workingStack.top().first + str;
    workingStack.pop();

    cout<< "STR: "<<str<< " ";

    if (rules.find(str) != rules.end()) {
        inputStack.push(rules[str]);
        cout << " To " << rules[str] << endl;
    }
}

void SysAnalyzer::SetRules() {
    rules.emplace("B", 'S');
    rules.emplace("X", 'B');
    rules.emplace("X;B", 'B');
    rules.emplace("O", 'X');
    rules.emplace("IAE", 'O');
    rules.emplace("TI?O:O", 'O');
    rules.emplace("F", 'E');
    rules.emplace("F#E", 'E');
}

```

```

        rules.emplace("P", 'F');
        rules.emplace("P&F", 'F');
        rules.emplace("I", 'P');
        rules.emplace("C", 'P');
        rules.emplace("(E)", 'P');
    }

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

SysAnalyzer::SysAnalyzer() {
    SetRelationMatrix();
    SetRules();
}

void SysAnalyzer::SetRelationMatrix() {
    relationMatrix.emplace(make_pair('S', '$'), '>');

    relationMatrix.emplace(make_pair('B', '$'), '>');

    relationMatrix.emplace(make_pair('X', ';'), '=');
    relationMatrix.emplace(make_pair('X', '$'), '>');

    relationMatrix.emplace(make_pair('O', ':'), '=');
    relationMatrix.emplace(make_pair('O', ';'), '>');
    relationMatrix.emplace(make_pair('O', '$'), '>');

    relationMatrix.emplace(make_pair('T', 'I'), '=');

    relationMatrix.emplace(make_pair('I', '?'), '=');
    relationMatrix.emplace(make_pair('I', 'A'), '=');
    relationMatrix.emplace(make_pair('I', ':'), '>');
    relationMatrix.emplace(make_pair('I', '#'), '>');
}

```

```

relationMatrix.emplace(make_pair('I', '&'), '>');
relationMatrix.emplace(make_pair('I', ')'), '>');
relationMatrix.emplace(make_pair('I', ';'), '>');
relationMatrix.emplace(make_pair('I', '$'), '>');

relationMatrix.emplace(make_pair('E', ':'), '>');
relationMatrix.emplace(make_pair('E', ')'), '=');
relationMatrix.emplace(make_pair('E', ':'), '>');
relationMatrix.emplace(make_pair('E', ';'), '>');
relationMatrix.emplace(make_pair('E', '$'), '>');

relationMatrix.emplace(make_pair('F', ':'), '>');
relationMatrix.emplace(make_pair('F', '#'), '=');
relationMatrix.emplace(make_pair('F', ')'), '>');
relationMatrix.emplace(make_pair('F', ';'), '>');
relationMatrix.emplace(make_pair('F', '$'), '>');

relationMatrix.emplace(make_pair('P', ':'), '>');
relationMatrix.emplace(make_pair('P', '#'), '>');
relationMatrix.emplace(make_pair('P', ')'), '>');
relationMatrix.emplace(make_pair('P', ';'), '>');
relationMatrix.emplace(make_pair('P', '$'), '>');
relationMatrix.emplace(make_pair('P', '&'), '=');

relationMatrix.emplace(make_pair('A', 'I'), '<');
relationMatrix.emplace(make_pair('A', 'E'), '=');
relationMatrix.emplace(make_pair('A', 'F'), '<');
relationMatrix.emplace(make_pair('A', 'P'), '<');
relationMatrix.emplace(make_pair('A', 'C'), '<');
relationMatrix.emplace(make_pair('A', '('), '<');

relationMatrix.emplace(make_pair('C', ':'), '>');
relationMatrix.emplace(make_pair('C', '#'), '>');
relationMatrix.emplace(make_pair('C', ')'), '>');
relationMatrix.emplace(make_pair('C', ';'), '>');
relationMatrix.emplace(make_pair('C', '$'), '>');
relationMatrix.emplace(make_pair('C', '&'), '>');

```



```

relationMatrix.emplace(make_pair('?', 'O'), '=');
relationMatrix.emplace(make_pair('?', 'T'), '<');
relationMatrix.emplace(make_pair('?', 'I'), '<');

relationMatrix.emplace(make_pair(':', 'O'), '=');
relationMatrix.emplace(make_pair(':', 'T'), '<');
relationMatrix.emplace(make_pair(':', 'I'), '<');

relationMatrix.emplace(make_pair('#', 'E'), '=');
relationMatrix.emplace(make_pair('#', 'I'), '<');
relationMatrix.emplace(make_pair('#', 'F'), '<');
relationMatrix.emplace(make_pair('#', 'P'), '<');
relationMatrix.emplace(make_pair('#', 'C'), '<');
relationMatrix.emplace(make_pair('#', '('), '<');

relationMatrix.emplace(make_pair('&', 'I'), '<');
relationMatrix.emplace(make_pair('&', 'F'), '=');
relationMatrix.emplace(make_pair('&', 'P'), '<');
relationMatrix.emplace(make_pair('&', 'C'), '<');
relationMatrix.emplace(make_pair('&', '('), '<');

relationMatrix.emplace(make_pair('(', 'E'), '=');
relationMatrix.emplace(make_pair('(', 'I'), '<');
relationMatrix.emplace(make_pair('(', 'F'), '<');
relationMatrix.emplace(make_pair('(', 'P'), '<');
relationMatrix.emplace(make_pair('(', 'C'), '<');
relationMatrix.emplace(make_pair('(', '('), '<');

relationMatrix.emplace(make_pair(')', ':'), '>');
relationMatrix.emplace(make_pair(')', '#'), '>');
relationMatrix.emplace(make_pair(')', ')'), '>');
relationMatrix.emplace(make_pair(')', ';'), '>');
relationMatrix.emplace(make_pair(')', '$'), '>');
relationMatrix.emplace(make_pair(')', '&'), '>');

relationMatrix.emplace(make_pair(';', 'B'), '=');

```

```

relationMatrix.emplace(make_pair(':', 'O'), '<');
relationMatrix.emplace(make_pair(':', 'T'), '<');
relationMatrix.emplace(make_pair(':', 'I'), '<');
relationMatrix.emplace(make_pair(':', 'X'), '<');

relationMatrix.emplace(make_pair('$', 'S'), '<');
relationMatrix.emplace(make_pair('$', 'B'), '<');
relationMatrix.emplace(make_pair('$', 'O'), '<');
relationMatrix.emplace(make_pair('$', 'T'), '<');
relationMatrix.emplace(make_pair('$', 'I'), '<');
relationMatrix.emplace(make_pair('$', 'X'), '<');
}

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

void SysAnalyzer::PrintOutputStack() {
    cout<<"\nOutput stack: ";
    while (!outputStack.empty()) {
        cout<<outputStack.top();
        outputStack.pop();
    }
    cout<<"\n";
}

```

Также были проведены тесты работы анализатора (рисунки 1-2).

if a?a:=c:=a(b); a:=g	a=12
TI?IAI:IA(I);IAI	IAC
Check Relation \$ T ^	Check Relation \$ I <
Check Relation T I ==	Check Relation I A =
Check Relation I ? ==	Check Relation A C <
Check Relation ? I ^	Check Relation C \$ >
Check Relation I A ==	STR: C To P
Check Relation A I ^	Check Relation A P <
Check Relation I : v ^	Check Relation P \$ >
STR: I To P	STR: P To F
Check Relation A P ^	Check Relation A F <
Check Relation P : v ^	Check Relation F \$ >
STR: P To F	STR: F To E
Check Relation A F ^	Check Relation A E =
Check Relation F : v ^	Check Relation E \$ >
STR: F To E	STR: IAE To O
Check Relation A E ==	Check Relation : O ==
Check Relation E : v ^	STR: TI?O:O To O
STR: IAE To O	Check Relation \$ O <
Check Relation ? O ==	Check Relation O \$ >
Check Relation : I ^	STR: O To X
Check Relation I A ==	Check Relation \$ X <
Check Relation A C ^	Check Relation X \$ >
Check Relation (I ^	STR: X To B
Check Relation I) v ^	Check Relation \$ B <
STR: I To P	Check Relation B \$ >
Check Relation (P ^	STR: B To S
Check Relation P) v ^	Check Relation \$ S <
STR: P To F	Check Relation S \$ >
Check Relation (F ^	STR: S The string belongs to the language
Check Relation F : v ^	
STR: F To E	
Check Relation (E ==	
Check Relation E) ==	
Check Relation) ; v ^	
STR: (E) To P	
Check Relation A P ^	
Check Relation P : v ^	
STR: P To F	
Check Relation A F ^	
Check Relation F : v ^	
STR: F To E	
Check Relation A E ==	
Check Relation E : v ^	
STR: IAE To O	
Check Relation : O ==	
STR: TI?O:O To O	
Check Relation \$ O <	
Check Relation O \$ >	
STR: O To X	
Check Relation \$ X <	
Check Relation X \$ >	
STR: X To B	
Check Relation \$ B <	
Check Relation B \$ >	
STR: B To S	
Check Relation \$ S <	
Check Relation S \$ >	
STR: S The string belongs to the language	

Рисунок 1 – Первые тесты

```

if e?d=d#d#(d&c):a=5
TI?IAI#I#(I&I):IAC
Check Relation $ T <
Check Relation T I =
Check Relation I ? =
Check Relation ? I <
Check Relation I A =
Check Relation A I <
Check Relation I # >
STR: I To P
Check Relation A P <
Check Relation P # >
STR: P To F
Check Relation A F <
Check Relation F # =
Check Relation # I <
Check Relation I # >
STR: I To P
Check Relation # P <
Check Relation P # >
STR: P To F
Check Relation # F <
Check Relation F # =
Check Relation # ( <
Check Relation ( I <
Check Relation I & >
STR: I To P
Check Relation ( P <
Check Relation P & =
Check Relation & I <
Check Relation I ) >
STR: I To P
Check Relation & P <
Check Relation P ) >
STR: P To F
Check Relation & F =
Check Relation F ) >
STR: P&F To F
Check Relation ( F <
Check Relation F ) >
STR: F To E
Check Relation ( E =
Check Relation E ) =
Check Relation ) : >
STR: (E) To P
Check Relation # P <
Check Relation P : >
STR: P To F
Check Relation # F <
Check Relation F : >
STR: F To E
Check Relation # E =
Check Relation E : >
STR: F#E To E
Check Relation # E =
Check Relation E : >
STR: F#E To E
Check Relation A E =
Check Relation E : >
STR: IAE To 0
Check Relation ? 0 =
Check Relation 0 : =
Check Relation : I <
Check Relation I A =
Check Relation A C <
Check Relation C $ >
STR: C To P
Check Relation A P <
Check Relation P $ >
STR: P To F
Check Relation A F <
Check Relation F $ >
STR: F To E
Check Relation A E =
Check Relation E $ >
STR: IAE To 0
Check Relation : 0 =
Check Relation 0 $ >
STR: TI?0:0 To 0
Check Relation $ 0 <
Check Relation 0 $ >
STR: 0 To X
Check Relation $ X <
Check Relation X $ >
STR: X To B
Check Relation $ B <
Check Relation B $ >
STR: B To S
Check Relation $ S <
Check Relation S $ >
STR: S The string belongs to the language

```

```

if abs?c-d

```

```

No operator matches: -

```

```

TI?I

```

```

Check Relation $ T <

```

```

Check Relation T I =

```

```

Check Relation I ? =

```

```

Check Relation ? I <

```

```

Check Relation I $ >

```

```

STR: I To P

```

```

Check Relation ? PThe string does not belong to the language

```

Рисунок 2 – Последующие тесты

Выводы

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