# 程序设计2：LR

曹桢 学号：2014211182 班级：2014211302

## 实验题目：

语法分析程序的设计与实现。

## 实验内容：

编写语法分析程序，实现对算术表达式的语法分析。产生式如下：

E->E+T | E-T | T

T->T\*F | T/F | F

F-> id | (E) | num

实验要求：在对输入的算数表达式进行分析的过程中，依次输出所采用的产生式。编写LR语法分析程序，要求如下：

1. 构造识别改文法的所有活前缀DFA
2. 构造该文法的LR分析表。
3. 编程实现算法4.3，构造LR分析程序。并进行错误处理。

## 实现要求：

采用C/C++作为实现语言，手工编写语法分析程序。

## 程序设计说明

1. 输入文件：
2. 待分析字符串
3. 拓广文法
4. S-> E
5. E-> E+T
6. E-> E-T
7. E-> T
8. T->T\*F
9. T-> T/F
10. T-> F
11. F-> (E)
12. F-> num
13. First集，Follow集

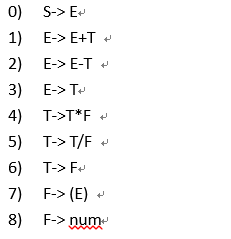
|  |  |  |
| --- | --- | --- |
|  | First | Follow |
| S | (,n | $ |
| E | (,n | $,+,-,) |
| T | (,n | $,+,-,),\*,/ |
| F | (,n | $,+,-,),\*,/ |

1. 分析表

包括错误处理：

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 状态 | action | | | | | | | | goto | | |
| N | + | - | \* | / | ( | ) | $ | E | T | F |
| 0 | S5 | E1 | E1 | E1 | E1 | S4 | E2 | E1 | 1 | 2 | 3 |
| 1 | E3 | S6 | S7 | E2 | E2 | E3 | E2 | ACC |  |  |  |
| 2 | R3 | R3 | R3 | S8 | S9 | R3 | R3 | R3 |  |  |  |
| 3 | R6 | R6 | R6 | R6 | R6 | R6 | R6 | R6 |  |  |  |
| 4 | S5 | E1 | E1 | E1 | E1 | S4 | E2 | E1 | 10 | 2 | 3 |
| 5 | R8 | R8 | R8 | R8 | R8 | R8 | R8 | R8 |  |  |  |
| 6 | S5 | E1 | E1 | E1 | E1 | S4 | E2 | E1 |  | 11 | 3 |
| 7 | S5 | E1 | E1 | E1 | E1 | S4 | E2 | E1 |  | 12 | 3 |
| 8 | S5 | E1 | E1 | E1 | E1 | S4 | E2 | E1 |  |  | 13 |
| 9 | S5 | E1 | E1 | E1 | E1 | S4 | E2 | E1 |  |  | 14 |
| 10 | E3 | S6 | S7 | E2 | E2 | E3 | S15 | E4 |  |  |  |
| 11 | R1 | R1 | R1 | S8 | S9 | R1 | R1 | R1 |  |  |  |
| 12 | R2 | R2 | R2 | S8 | S9 | R2 | R2 | R2 |  |  |  |
| 13 | R4 | R4 | R4 | R4 | R4 | R4 | R4 | R4 |  |  |  |
| 14 | R5 | R5 | R5 | R5 | R5 | R5 | R5 | R5 |  |  |  |
| 15 | R7 | R7 | R7 | R7 | R7 | R7 | R7 | R7 |  |  |  |

1. 输出：对符号串的分析过程
2. 算法思想：
3. 拓广文法



1. First集和Follow集

|  |  |  |
| --- | --- | --- |
|  | First | Follow |
| S | (,n | $ |
| E | (,n | $,+,-,) |
| T | (,n | $,+,-,),\*,/ |
| F | (,n | $,+,-,),\*,/ |

1. 构造拓广文法的项目集规范族，并构造出识别所有前缀的DFA

\*

F

I12:

E-> E-T˙

T-> T˙\*F

T-> T˙/F

num

/

(

I7:

E-> E˙-T

T->˙T\*F

T->˙T/F

T->˙F

F->˙(E)

F->˙num

\*

T

T

I11:

E-> E+T˙

T-> T˙\*F

T-> T˙/F

F

I6:

E-> E˙+T

T->˙T\*F

T->˙T/F

T->˙F

F->˙(E)

F->˙num

/

(

num

-

+

I1:

S-> E˙

E-> E˙+T

E-> E˙-T

F

I13:

T-> T \*F˙

I8:

T-> T\*˙F

F->˙(E)

F->˙num

\*

(

num

I2:

E-> T˙

T-> T˙\*F

T-> T˙/F

/

E

F

T

I9:

T-> T/˙F

F->˙(E)

F->˙num

I14:

T-> T /F˙

I0:

S->˙E

E->˙E+T

E->˙E-T

E->˙T

T->˙T\*F

T->˙T/F

T->˙F

F->˙(E)

F->˙num

num

(

I4:

F-> (˙E)

E->˙E+T

E->˙E-T

E->˙T

T->˙T\*F

T->˙T/F

T->˙F

F->˙(E)

F->˙num

(

I10:

F-> (E˙)

E-> E˙+T

E-> E˙-T

)

E

I15:

T-> (E)˙

+

(

-

T

num

F

I3:

T-> F˙

I5:

F-> num˙

1. 构造文法的LR分析表

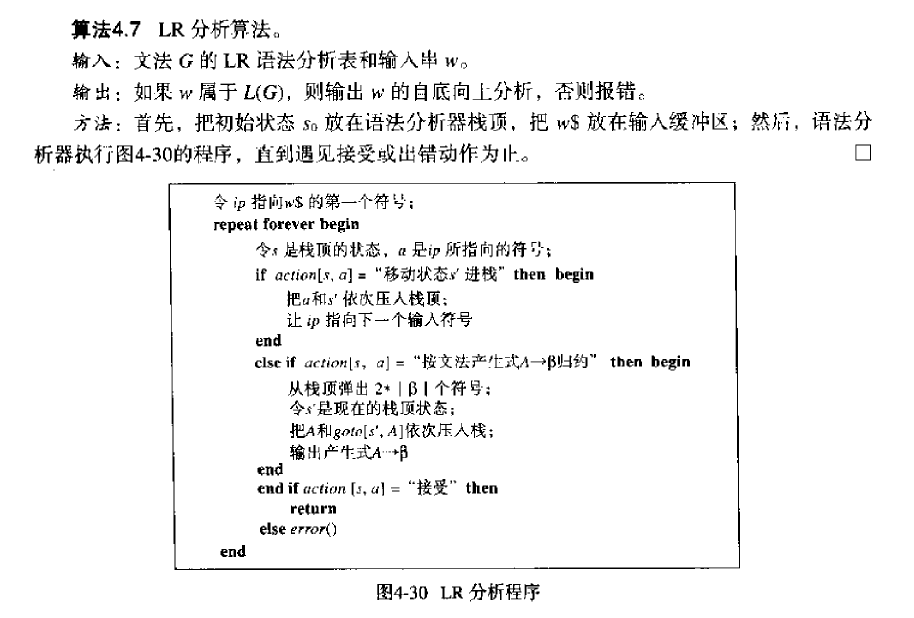
不包括错误处理：

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 状态 | action | | | | | | | | goto | | |
| N | + | - | \* | / | ( | ) | $ | E | T | F |
| 0 | S5 |  |  |  |  | S4 |  |  | 1 | 2 | 3 |
| 1 |  | S6 | S7 |  |  |  |  | ACC |  |  |  |
| 2 |  | R3 | R3 | S8 | S9 |  | R3 | R3 |  |  |  |
| 3 |  | R6 | R6 | R6 | R6 |  | R6 | R6 |  |  |  |
| 4 | S5 |  |  |  |  | S4 |  |  | 10 | 2 | 3 |
| 5 |  | R8 | R8 | R8 | R8 |  | R8 | R8 |  |  |  |
| 6 | S5 |  |  |  |  | S4 |  |  |  | 11 | 3 |
| 7 | S5 |  |  |  |  | S4 |  |  |  | 12 | 3 |
| 8 | S5 |  |  |  |  | S4 |  |  |  |  | 13 |
| 9 | S5 |  |  |  |  | S4 |  |  |  |  | 14 |
| 10 |  | S6 | S7 |  |  |  | S15 |  |  |  |  |
| 11 |  | R1 | R1 | S8 | S9 |  | R1 | R1 |  |  |  |
| 12 |  | R2 | R2 | S8 | S9 |  | R2 | R2 |  |  |  |
| 13 |  | R4 | R4 | R4 | R4 |  | R4 | R4 |  |  |  |
| 14 |  | R5 | R5 | R5 | R5 |  | R5 | R5 |  |  |  |
| 15 |  | R7 | R7 | R7 | R7 |  | R7 | R7 |  |  |  |

包括错误处理：

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 状态 | action | | | | | | | | goto | | |
| N | + | - | \* | / | ( | ) | $ | E | T | F |
| 0 | S5 | E1 | E1 | E1 | E1 | S4 | E2 | E1 | 1 | 2 | 3 |
| 1 | E3 | S6 | S7 | E2 | E2 | E3 | E2 | ACC |  |  |  |
| 2 | R3 | R3 | R3 | S8 | S9 | R3 | R3 | R3 |  |  |  |
| 3 | R6 | R6 | R6 | R6 | R6 | R6 | R6 | R6 |  |  |  |
| 4 | S5 | E1 | E1 | E1 | E1 | S4 | E2 | E1 | 10 | 2 | 3 |
| 5 | R8 | R8 | R8 | R8 | R8 | R8 | R8 | R8 |  |  |  |
| 6 | S5 | E1 | E1 | E1 | E1 | S4 | E2 | E1 |  | 11 | 3 |
| 7 | S5 | E1 | E1 | E1 | E1 | S4 | E2 | E1 |  | 12 | 3 |
| 8 | S5 | E1 | E1 | E1 | E1 | S4 | E2 | E1 |  |  | 13 |
| 9 | S5 | E1 | E1 | E1 | E1 | S4 | E2 | E1 |  |  | 14 |
| 10 | E3 | S6 | S7 | E2 | E2 | E3 | S15 | E4 |  |  |  |
| 11 | R1 | R1 | R1 | S8 | S9 | R1 | R1 | R1 |  |  |  |
| 12 | R2 | R2 | R2 | S8 | S9 | R2 | R2 | R2 |  |  |  |
| 13 | R4 | R4 | R4 | R4 | R4 | R4 | R4 | R4 |  |  |  |
| 14 | R5 | R5 | R5 | R5 | R5 | R5 | R5 | R5 |  |  |  |
| 15 | R7 | R7 | R7 | R7 | R7 | R7 | R7 | R7 |  |  |  |

1. 算法4.3，LR分析



1. 高层数据结构设计
2. 常量定义

const int columnNum = 11; //分析表列数

const int Nsize = 4; //终结符个数

const int Gsize = 9; //产生式个数

const int stateNum = 16; //状态个数

1. 关键变量定义

char N[Nsize]={'S','E','T','F'}; //非终结符

char item[columnNum]={'n','+','-','\*','/','(',')','$', 'E','T','F'}; //分析表的列

vector<char> First[Nsize]; //First集

vector<char> Follow[Nsize]; //Follow集

stack<int> stateStack; //状态栈

string AnalyseTable[stateNum][columnNum]; //预测分析表

string R[Gsize]; //拓广文法后的产生式

string inputBuffer; //输入缓冲区

int ip; //输入缓冲区指针

1. 函数说明

/\*将字符映射到列\*/

int map(char x)

/\*建立分析表\*/

void createTable()

/\*打印栈\*/

void printstack()

/\*把输入字符串中的数变成n\*/

string numberToN(string w)

/\*错误处理\*/

void error(char kind = 'x')

/\*LR分析\*/

void LR(string w)

/\*输出\*/

void printout()

## 源代码

/\*coder：曹桢 最后修改时间：2016-10-30 18:19

实验：程序设计2 方法3 LR语法分析程序

\*/

#include <iostream>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <stack>

#include <vector>

#include <iomanip>

using namespace std;

const int columnNum = 11; //分析表列数

const int Nsize = 4; //终结符个数

const int Gsize = 9; //产生式个数

const int stateNum = 16; //状态个数

char N[Nsize]={'S','E','T','F'}; //非终结符

char item[columnNum]={'n','+','-','\*','/','(',')','$', 'E','T','F'}; //分析表的列

vector<char> First[Nsize]; //First集

vector<char> Follow[Nsize]; //Follow集

stack<int> stateStack; //状态栈

string AnalyseTable[stateNum][columnNum]; //预测分析表

string R[Gsize]; //拓广文法后的产生式

string inputBuffer; //输入缓冲区

int ip; //输入缓冲区指针

void Init()

{

/\*First集\*/

First[0].push\_back('(');

First[0].push\_back('n');

First[1].push\_back('(');

First[1].push\_back('n');

First[2].push\_back('(');

First[2].push\_back('n');

First[3].push\_back('(');

First[3].push\_back('n');

/\*Follow集\*/

Follow[0].push\_back('$');

Follow[1].push\_back('$');

Follow[1].push\_back(')');

Follow[1].push\_back('+');

Follow[1].push\_back('-');

Follow[2].push\_back('$');

Follow[2].push\_back(')');

Follow[2].push\_back('+');

Follow[2].push\_back('-');

Follow[2].push\_back('\*');

Follow[2].push\_back('/');

Follow[3].push\_back('$');

Follow[3].push\_back(')');

Follow[3].push\_back('+');

Follow[3].push\_back('-');

Follow[3].push\_back('\*');

Follow[3].push\_back('/');

/\*拓广文法后的产生式\*/

R[0] = "S->E";

R[1] = "E->E+T";

R[2] = "E->E-T";

R[3] = "E->T";

R[4] = "T->T\*F";

R[5] = "T->T/F";

R[6] = "T->F";

R[7] = "F->(E)";

R[8] = "F->n" ;

/\*预测分析表\*/

for(int i = 0; i < stateNum; i++){

for(int j = 0; j < columnNum; j++){

AnalyseTable[i][j] = "";

}

}

}

/\*将字符映射到列\*/

int map(char x)

{

switch(x){

case 'n':

return 0;

case '+':

return 1;

case '-':

return 2;

case '\*':

return 3;

case '/':

return 4;

case '(':

return 5;

case ')':

return 6;

case '$':

return 7;

case 'E':

return 8;

case 'T':

return 9;

case 'F':

return 10;

default:

return -1;

}

}

/\*建立分析表\*/

void createTable()

{

AnalyseTable[0][0] = "S5";

AnalyseTable[0][1] = "E1";

AnalyseTable[0][2] = "E1";

AnalyseTable[0][3] = "E1";

AnalyseTable[0][4] = "E1";

AnalyseTable[0][5] = "S4";

AnalyseTable[0][6] = "E2";

AnalyseTable[0][7] = "E1";

AnalyseTable[0][8] = "1";

AnalyseTable[0][9] = "2";

AnalyseTable[0][10] = "3";

AnalyseTable[1][0] = "E3";

AnalyseTable[1][1] = "S6";

AnalyseTable[1][2] = "S7";

AnalyseTable[1][3] = "E2";

AnalyseTable[1][4] = "E2";

AnalyseTable[1][5] = "E3";

AnalyseTable[1][6] = "E2";

AnalyseTable[1][7] = "ACC";

AnalyseTable[2][0] = "R3";

AnalyseTable[2][1] = "R3";

AnalyseTable[2][2] = "R3";

AnalyseTable[2][3] = "S8";

AnalyseTable[2][4] = "S9";

AnalyseTable[2][5] = "R3";

AnalyseTable[2][6] = "R3";

AnalyseTable[2][7] = "R3";

AnalyseTable[3][0] = "R6";

AnalyseTable[3][1] = "R6";

AnalyseTable[3][2] = "R6";

AnalyseTable[3][3] = "R6";

AnalyseTable[3][4] = "R6";

AnalyseTable[3][5] = "R6";

AnalyseTable[3][6] = "R6";

AnalyseTable[3][7] = "R6";

AnalyseTable[4][0] = "S5";

AnalyseTable[4][1] = "E1";

AnalyseTable[4][2] = "E1";

AnalyseTable[4][3] = "E1";

AnalyseTable[4][4] = "E1";

AnalyseTable[4][5] = "S4";

AnalyseTable[4][6] = "E2";

AnalyseTable[4][7] = "E1";

AnalyseTable[4][8] = "10";

AnalyseTable[4][9] = "2";

AnalyseTable[4][10] = "3";

AnalyseTable[5][0] = "R8";

AnalyseTable[5][1] = "R8";

AnalyseTable[5][2] = "R8";

AnalyseTable[5][3] = "R8";

AnalyseTable[5][4] = "R8";

AnalyseTable[5][5] = "R8";

AnalyseTable[5][6] = "R8";

AnalyseTable[5][7] = "R8";

AnalyseTable[6][0] = "S5";

AnalyseTable[6][1] = "E1";

AnalyseTable[6][2] = "E1";

AnalyseTable[6][3] = "E1";

AnalyseTable[6][4] = "E1";

AnalyseTable[6][5] = "S4";

AnalyseTable[6][6] = "E2";

AnalyseTable[6][7] = "E1";

AnalyseTable[6][9] = "11";

AnalyseTable[6][10] = "3";

AnalyseTable[7][0] = "S5";

AnalyseTable[7][1] = "E1";

AnalyseTable[7][2] = "E1";

AnalyseTable[7][3] = "E1";

AnalyseTable[7][4] = "E1";

AnalyseTable[7][5] = "S4";

AnalyseTable[7][6] = "E2";

AnalyseTable[7][7] = "E1";

AnalyseTable[7][9] = "12";

AnalyseTable[7][10] = "3";

AnalyseTable[8][0] = "S5";

AnalyseTable[8][1] = "E1";

AnalyseTable[8][2] = "E1";

AnalyseTable[8][3] = "E1";

AnalyseTable[8][4] = "E1";

AnalyseTable[8][5] = "S4";

AnalyseTable[8][6] = "E2";

AnalyseTable[8][7] = "E1";

AnalyseTable[8][10] = "13";

AnalyseTable[9][0] = "S5";

AnalyseTable[9][1] = "E1";

AnalyseTable[9][2] = "E1";

AnalyseTable[9][3] = "E1";

AnalyseTable[9][4] = "E1";

AnalyseTable[9][5] = "S4";

AnalyseTable[9][6] = "E2";

AnalyseTable[9][7] = "E1";

AnalyseTable[9][10] = "14";

AnalyseTable[10][0] = "E3";

AnalyseTable[10][1] = "S6";

AnalyseTable[10][2] = "S7";

AnalyseTable[10][3] = "E2";

AnalyseTable[10][4] = "E2";

AnalyseTable[10][5] = "E3";

AnalyseTable[10][6] = "S15";

AnalyseTable[10][7] = "E4";

AnalyseTable[11][0] = "R1";

AnalyseTable[11][1] = "R1";

AnalyseTable[11][2] = "R1";

AnalyseTable[11][3] = "S8";

AnalyseTable[11][4] = "S9";

AnalyseTable[11][5] = "R1";

AnalyseTable[11][6] = "R1";

AnalyseTable[11][7] = "R1";

AnalyseTable[12][0] = "R2";

AnalyseTable[12][1] = "R2";

AnalyseTable[12][2] = "R2";

AnalyseTable[12][3] = "S8";

AnalyseTable[12][4] = "S9";

AnalyseTable[12][5] = "R2";

AnalyseTable[12][6] = "R2";

AnalyseTable[12][7] = "R2";

AnalyseTable[13][0] = "R4";

AnalyseTable[13][1] = "R4";

AnalyseTable[13][2] = "R4";

AnalyseTable[13][3] = "R4";

AnalyseTable[13][4] = "R4";

AnalyseTable[13][5] = "R4";

AnalyseTable[13][6] = "R4";

AnalyseTable[13][7] = "R4";

AnalyseTable[14][0] = "R5";

AnalyseTable[14][1] = "R5";

AnalyseTable[14][2] = "R5";

AnalyseTable[14][3] = "R5";

AnalyseTable[14][4] = "R5";

AnalyseTable[14][5] = "R5";

AnalyseTable[14][6] = "R5";

AnalyseTable[14][7] = "R5";

AnalyseTable[15][0] = "R7";

AnalyseTable[15][1] = "R7";

AnalyseTable[15][2] = "R7";

AnalyseTable[15][3] = "R7";

AnalyseTable[15][5] = "R7";

AnalyseTable[15][4] = "R7";

AnalyseTable[15][6] = "R7";

AnalyseTable[15][7] = "R7";

}

/\*打印栈\*/

void printstack()

{

string output = "";

stack<int> temp;

while(!stateStack.empty()){

int t = stateStack.top();

temp.push(t);

stateStack.pop();

}

while(!temp.empty()){

int t = temp.top();

char xoxo[10];

itoa(t, xoxo, 10);

output = output +"|"+ xoxo;

stateStack.push(t);

temp.pop();

}

cout<<output;

}

/\*把输入字符串中的数变成n\*/

string numberToN(string w)

{

bool flag = false;

string result = "";

for(int i = 0; i < w.length(); i++){

while(w[i] >= '0' && w[i] <= '9' && i < w.length()){

flag = true;

i++;

}

if(flag == true){

result = result + 'n';

}

if(i != w.length()){

result = result + w[i];

flag = false;

}

}

return result;

}

/\*错误处理\*/

void error(char kind = 'x')

{

switch(kind){

case '1':

cout<<"缺少运算对象"<<endl;

stateStack.push(3);

break;

case '2':

cout<<"括号不匹配"<<endl;

ip++;

break;

case '3':

cout<<"缺少运算符号"<<endl;

stateStack.push(4);

break;

case '4':

cout<<"缺少右括号"<<endl;

stateStack.push(9);

break;

default:

cout<<"未知错误"<<endl;

}

}

/\*LR分析\*/

void LR(string w)

{

int X;

char a;

ip = 0;

int step = 0;

stateStack.push(0);

inputBuffer = w+"$";

cout<<left<<setw(10)<<"步骤"<<left<<setw(20)<<"栈"<<left<<setw(20)<<"输入"<<left<<setw(20)<<"输出"<<endl;

do{

X = stateStack.top();

a = inputBuffer[ip];

step++;

int inta = map(a);

cout<<left<<setw(10)<<step;

cout<<left<<setw(20);

printstack();

cout<<left<<setw(20)<<inputBuffer.substr(ip);

if(inta == -1){

cout<<"非法输入"<<endl;

return;

}

if(AnalyseTable[X][inta][0] == 'S'){

string t = "";

for(int i = 1; i < AnalyseTable[X][inta].length(); i++){

t = t + AnalyseTable[X][inta][i];

}

int num = atoi(t.c\_str());

stateStack.push(num);

ip++;

cout<<"Shift "<<num<<endl;

} else if(AnalyseTable[X][inta][0] == 'R'){

int num = int(AnalyseTable[X][inta][1]) - '0';

int beta = R[num].length()-3;

for(int i = 0; i < beta; i++){

stateStack.pop();

}

X = stateStack.top();

a = R[num][0];

inta = map(a);

cout<<"Reduce by "<<R[num]<<endl;

string t = "";

for(int i = 0; i < AnalyseTable[X][inta].length(); i++){

t = t + AnalyseTable[X][inta][i];

}

int state = atoi(t.c\_str());

stateStack.push(state);

}else if(AnalyseTable[X][inta] == "ACC"){

cout<<"ACC"<<endl;

return;

}

else if(AnalyseTable[X][inta][0] == 'E'){

error(AnalyseTable[X][inta][1]);

}

else{

error();

}

}while(1);

}

/\*输出\*/

void printout()

{

cout<<"由给定产生式进行语法分析，产生式如下："<<endl;

cout<<"E->E+T | E-T | T\nT->T\*F | T/F | F\nF-> (E) | num"<<endl;

cout<<"------------------------------------------------------------------"<<endl;

cout<<"拓展文法后："<<endl;

for(int i = 0; i < Gsize; i++){

cout<<'('<<i<<')'<<R[i]<<endl;

}

cout<<"------------------------------------------------------------------"<<endl;

cout<<"\tFirst\tFollow"<<endl;

for(int i = 0; i < Nsize; i++){

cout<<N[i]<<"\t";

for(int j = 0; j < First[i].size(); j++){

cout<<First[i][j]<<" ";

}

cout<<"\t";

for(int j = 0; j < Follow[i].size(); j++){

cout<<Follow[i][j]<<" ";

}

cout<<"\n";

}

cout<<"------------------------------------------------------------------"<<endl;

string emp = "";

cout<<left<<setw(6)<<emp;

for(int j = 0; j < columnNum; j++){

cout<<left<<setw(6)<<item[j];

}

cout<<endl;

for(int i = 0; i < stateNum; i++){

cout<<left<<setw(6)<<i;

for(int j = 0; j < columnNum; j++){

cout<<left<<setw(6)<<AnalyseTable[i][j];

}

cout<<endl;

}

cout<<endl;

cout<<"------------------------------------------------------------------"<<endl;

}

int main()

{

string w;

string y;

cout<<"请输入要分析的符号串：";

cin>>w;

y = numberToN(w);

Init();

createTable();

printout();

LR(y);

return 0;

}

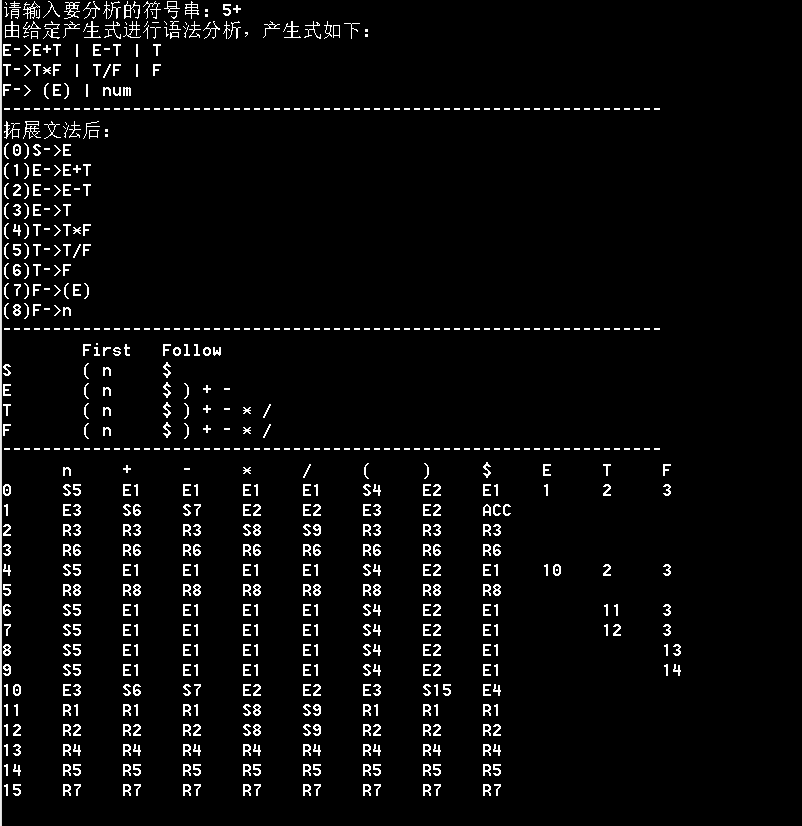
## 测试报告

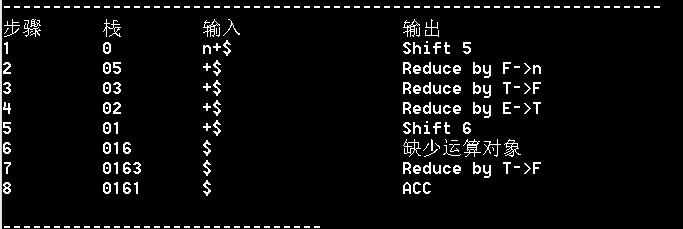
1、

输入：



输出：



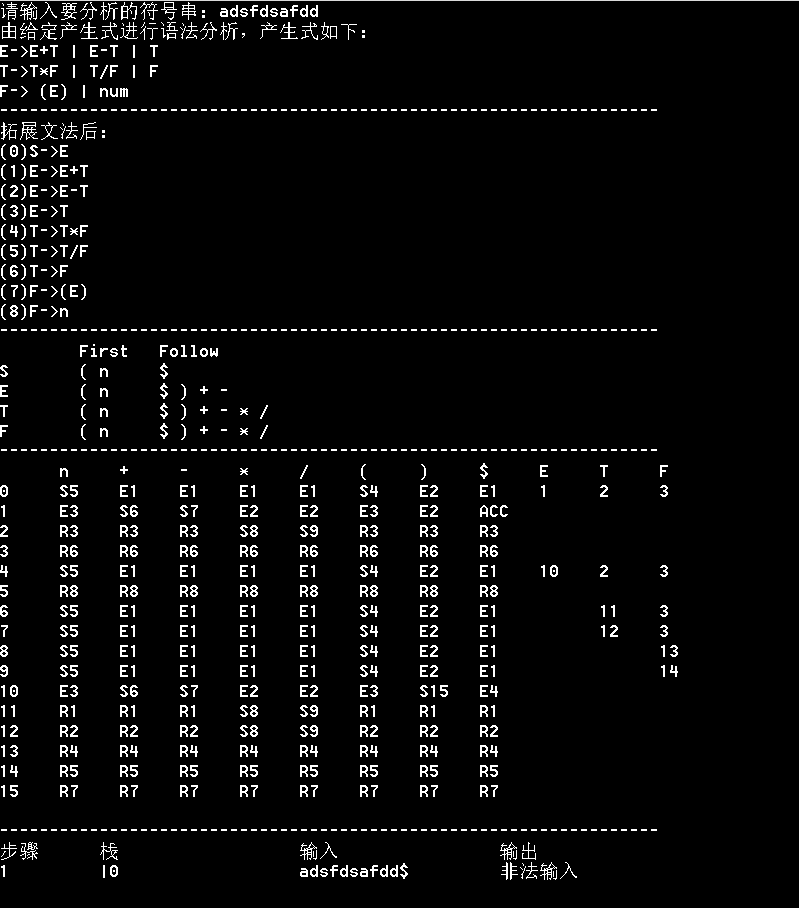


分析说明：能识别错误的符号串，判断错误类型，并做出相应恢复。

2、

输入：

输出：



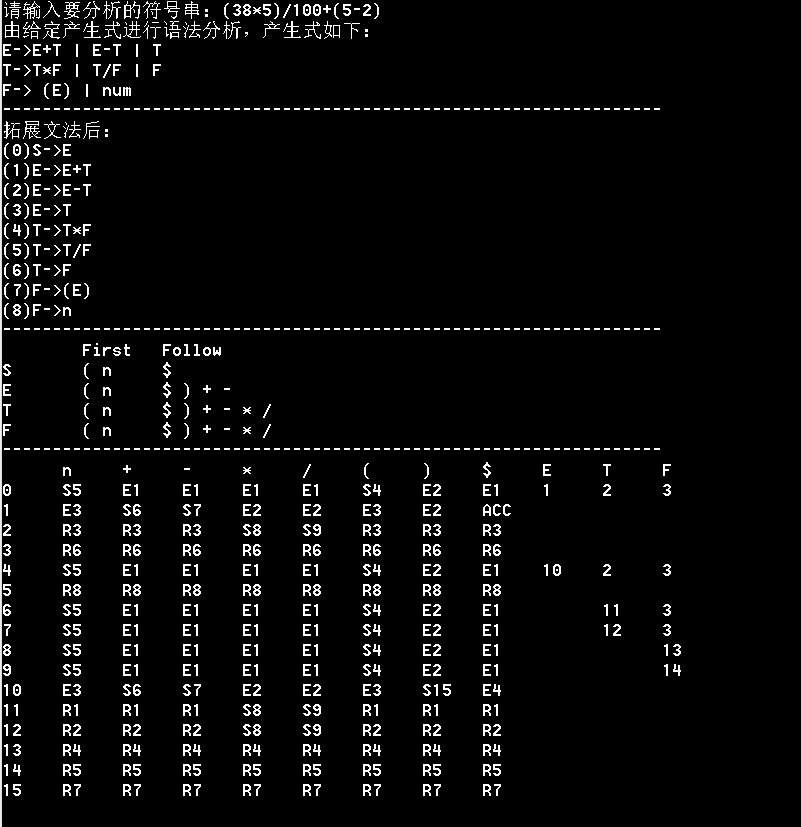
分析说明：在胡乱输入时，程序能够识别并给出提示。

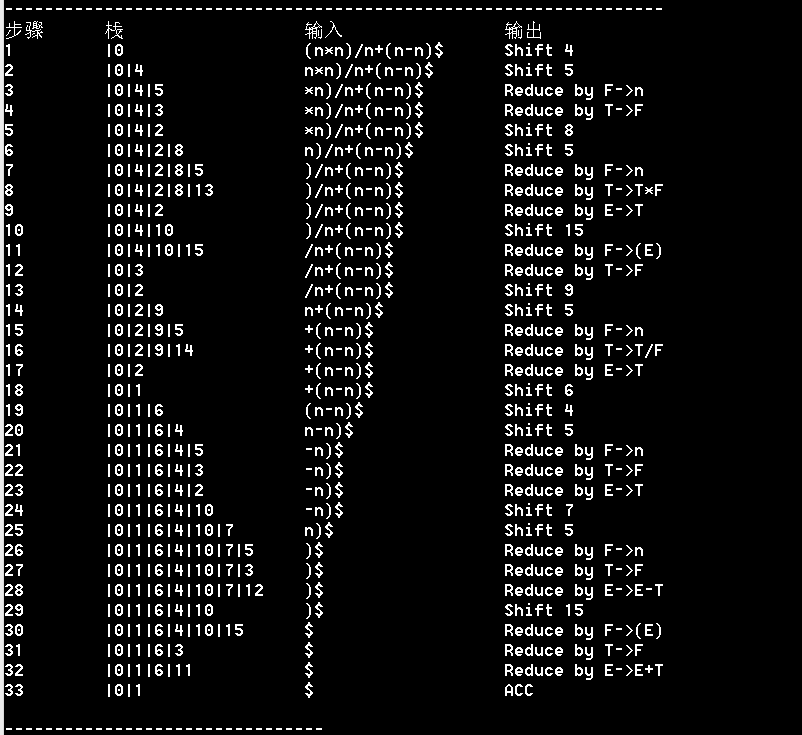
3、

输入：



输出：





分析说明：能识别正确的符号串。