北京邮电大学

编译原理与技术 实验三

语义分析程序实验报告

**学院: 计算机学院**

|  |  |  |
| --- | --- | --- |
| **姓名** | **学号** | **班级** |
| **刘科孟** | **2014211169** | **2014211302** |

**2016年 12 月 17 日**

1. **实验题目及要求：**
2. 实验题目： 语义分析程序的设计与实现。
3. 实验内容和要求：
4. 实验内容：编写语义分析和翻译程序，实现对算术表达式的类型检查和求值。要求所分析算术表达式由如下的文法产生：

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

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

F -> num|num.num|( E )

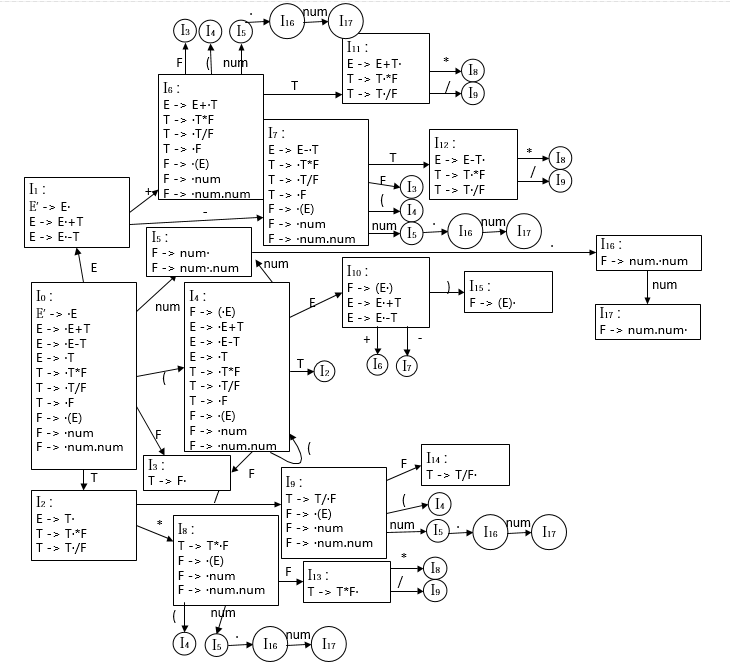
实验要求：用自底向上的语法制导翻译技术实现对表达式的分析和翻译。

1. 写出满足要求的语法制导定义或翻译方案。
2. 编写语义分析和翻译程序，实现对表达式的类型进行检查和求值，并输出：
3. 分析过程中所用产生式。
4. 识别出子表达式的类型。
5. 识别出子表达式的值。
6. 实验方法：手工编写分析程序。
7. **程序设计说明：**
8. 构造LR预测分析表：
9. 构造识别该文法所有活前缀的DFA：

构造所得的DFA如下图所示：

相关说明：

该DFA的状态1,2,5,11,12中均存在移进-归约冲突，根据SLR(1)方法均可以得到相应的解决。



1. 构造该文法的LR分析表：

根据该文法所有活前缀的DFA，以及SLR(1)方法解决状态1,2,5,11,12中的移进-归约冲突，分析过程中所可能出现的错误情况，可以得到如下的SLR(1)预测分析表：

注释说明：

S：代表Shift，将状态压入栈中；

R：代表Reduce，将产生式根据编号的文法产生式进行归约；

ACC：代表Accept，该表达式符合文法；

E1：缺少运算对象；

E2：括号不匹配；

E3：缺少运算符号；

E4：缺少右括号；

E5：词法错误。

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 状态 | action | | | | | | | | | goto | | |
| + | - | \* | / | ( | ) | num | $ | . | E | T | F |
| 0 | E1 | E1 | E1 | E1 | S4 | E1 | S5 | E1 | E1 | 1 | 2 | 3 |
| 1 | S6 | S7 | E3 | E3 | E3 | E2 | E3 | ACC | E3 |  |  |  |
| 2 | R3 | R3 | S8 | S9 | E3 | R3 | E3 | R3 | E3 |  |  |  |
| 3 | R6 | R6 | R6 | R6 | E3 | R6 | E3 | R6 | E3 |  |  |  |
| 4 | E1 | E1 | E1 | E1 | S4 | E1 | S5 | E1 | E1 | 10 | 2 | 3 |
| 5 | R8 | R8 | R8 | R8 | E3 | R8 | E3 | R8 | S16 |  |  |  |
| 6 | E1 | E1 | E1 | E1 | S4 | E1 | S5 | E1 | E1 |  | 11 | 3 |
| 7 | E1 | E1 | E1 | E1 | S4 | E1 | S5 | E1 | E1 |  | 12 | 3 |
| 8 | E1 | E1 | E1 | E1 | S4 | E1 | S5 | E1 | E1 |  |  | 13 |
| 9 | E1 | E1 | E1 | E1 | S4 | E1 | S5 | E1 | E1 |  |  | 14 |
| 10 | S6 | S7 | E3 | E3 | E3 | S15 | E3 | E4 | E3 |  |  |  |
| 11 | R1 | R1 | S8 | S9 | E3 | R1 | E3 | R1 | E3 |  |  |  |
| 12 | R2 | R2 | S8 | S9 | E3 | R2 | E3 | R2 | E3 |  |  |  |
| 13 | R4 | R4 | R4 | R4 | E3 | R4 | E3 | R4 | E3 |  |  |  |
| 14 | R5 | R5 | R5 | R5 | E3 | R5 | E3 | R5 | E3 |  |  |  |
| 15 | R7 | R7 | R7 | R7 | E3 | R7 | E3 | R7 | E3 |  |  |  |
| 16 | E5 | E5 | E5 | E5 | E5 | E5 | S17 | E5 | E5 |  |  |  |
| 17 | R9 | R9 | R9 | R9 | E3 | R9 | E3 | R9 | E3 |  |  |  |

1. 语法制导定义的翻译方案：

E -> E1 + T{if(E1.type == T.type) E.type = E1.type, E.num = E1.num + T.num;

else if(E1.type == double)

E.type = E1.type, E.num = E1.num + (double)T.num;

else if(T.type = double)

E.type = T.type, E.num = (double)E1.num + T.num;

printf(E.type, E.num);}

E -> E1 - T{if(E1.type == T.type) E.type = E1.type, E.num = E1.num - T.num;

else if(E1.type == double)

E.type = E1.type, E.num = E1.num - (double)T.num;

else if(T.type = double)

E.type = T.type, E.num = (double)E1.num - T.num;

printf(E.type, E.num);}

E -> T{E.type = T.type, E.num = T.num, printf(E.type, E.num);}

T -> T1 \* F{if(T1.type == T.type) T.type = T1.type, T.num = T1.num \* F.num;

else if(T1.type == double)

T.type = T1.type, T.num = T1.num \* (double)F.num;

else if(T.type = double)

T.type = F.type, T.num = (double)T1.num \* F.num;

printf(T.type, T.num);}

T -> T1 / F{if(T1.type == T.type) T.type = T1.type, T.num = T1.num / F.num;

else if(T1.type == double)

T.type = T1.type, T.num = T1.num / (double)F.num;

else if(T.type = double)

T.type = F.type, T.num = (double)T1.num / F.num;

printf(T.type, T.num);}

T -> F{T.type = F.type, T.num = F.num, printf(T.type, T.num);}

F -> num{F.type = lexval(num.type), F.num = lexval(num), printf(F.type,F.num);}

F -> num.num{F.type = lexval(num.num.type), F.num = lexval(num.num),

printf(F.type,F.num);}

F -> ( E ){F.type = E.type, F.num = E.num, printf(F.type, F.num);}

说明：以上语法制导定义翻译方案在程序中转换为在栈中进行计算，具体在栈中的位置指针均可以简单地求得。

1. 源程序的主要模块和数据结构：
2. 相关数据结构：

map<string, int> m\_id; //将所有文法符号映射为整数

map<int, string> rm\_id; //整数值对应的文法符号

map<string, int> p\_id; //将所有的文法产生式映射为整数

map<int, string> rp\_id; //整数值对应的产生式

vector< pair<string, string> > production;//存储文法产生式

string Table[20][20]; //存储所构造的LR预测分析表

string ex[50]; //存储输入需要分析的表达式

string ans\_ex; //语法分析过程所需的表达式

int i\_val[50]; //存放符号综合属性

struct Res{ //存放每一步子表达式的值

int resa = -99999;

double resb = -99999.0;

}res0;

1. 相关的函数方法：

void init() //建立文法符号，产生式等相关的映射关系

void analyze\_table() //构造LR预测分析表

void output() //输出LR预测分析表

int spl(string str, int mode)//用于对输入表达式进行预处理

int toTen(string str) //把预测分析表中字符数值转换为整数

bool Analyze(string exp\_str)

//对输入的表达式根据预测分析表进行分析

double cal\_float(int num) //计算浮点数的值

int cal\_num(string num) //计算字符串转为整数的值

void word\_analyze(string str)//输出语法分析的产生式

1. **源程序代码：**

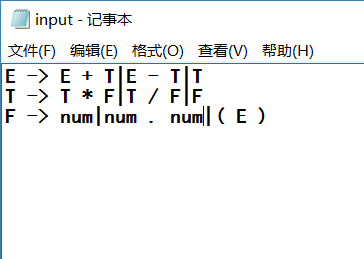
见附录。

1. **可执行程序：**

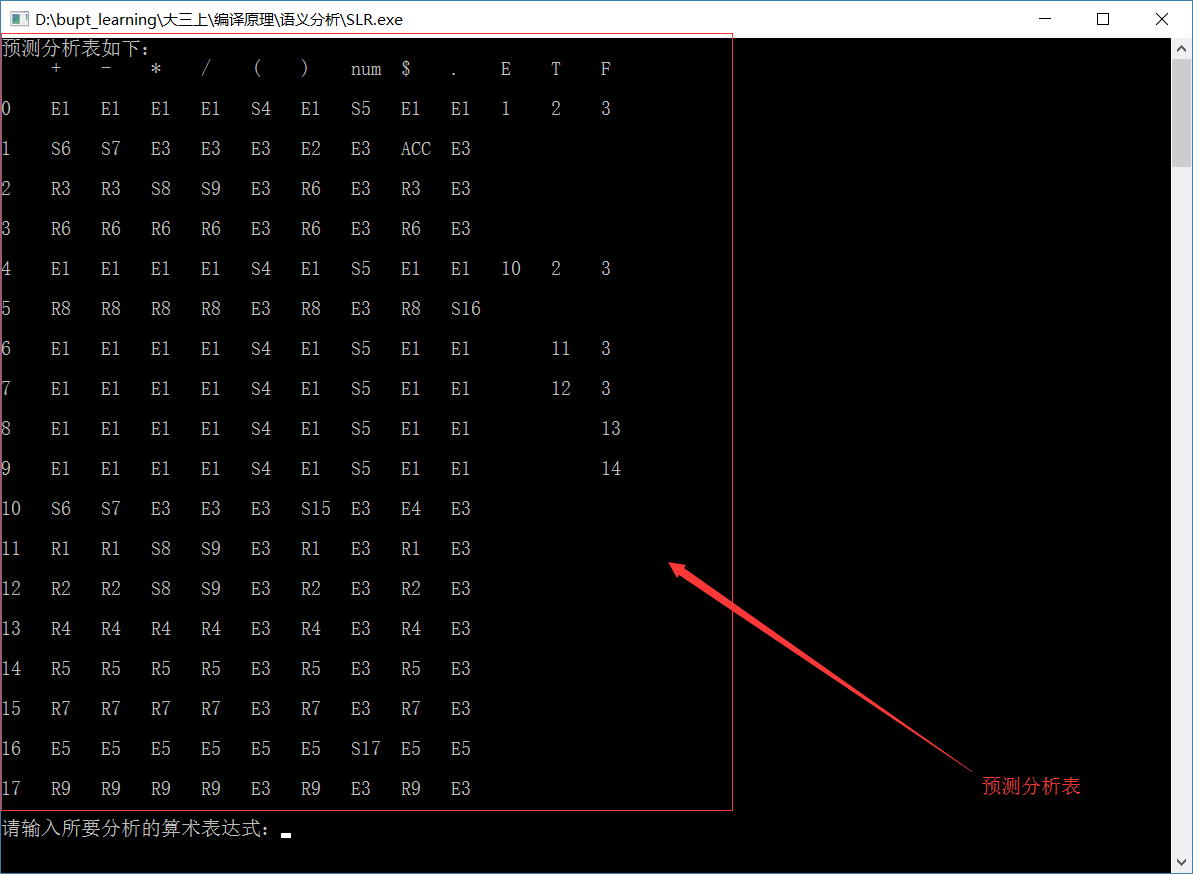
见附件.exe文件。

1. **测试报告：**
2. 文法输入：

（如下图）



1. 由语法分析程序输出的LR预测分析表：



1. 对输入的文法表达式进行分析及得到的结果：

样例1：

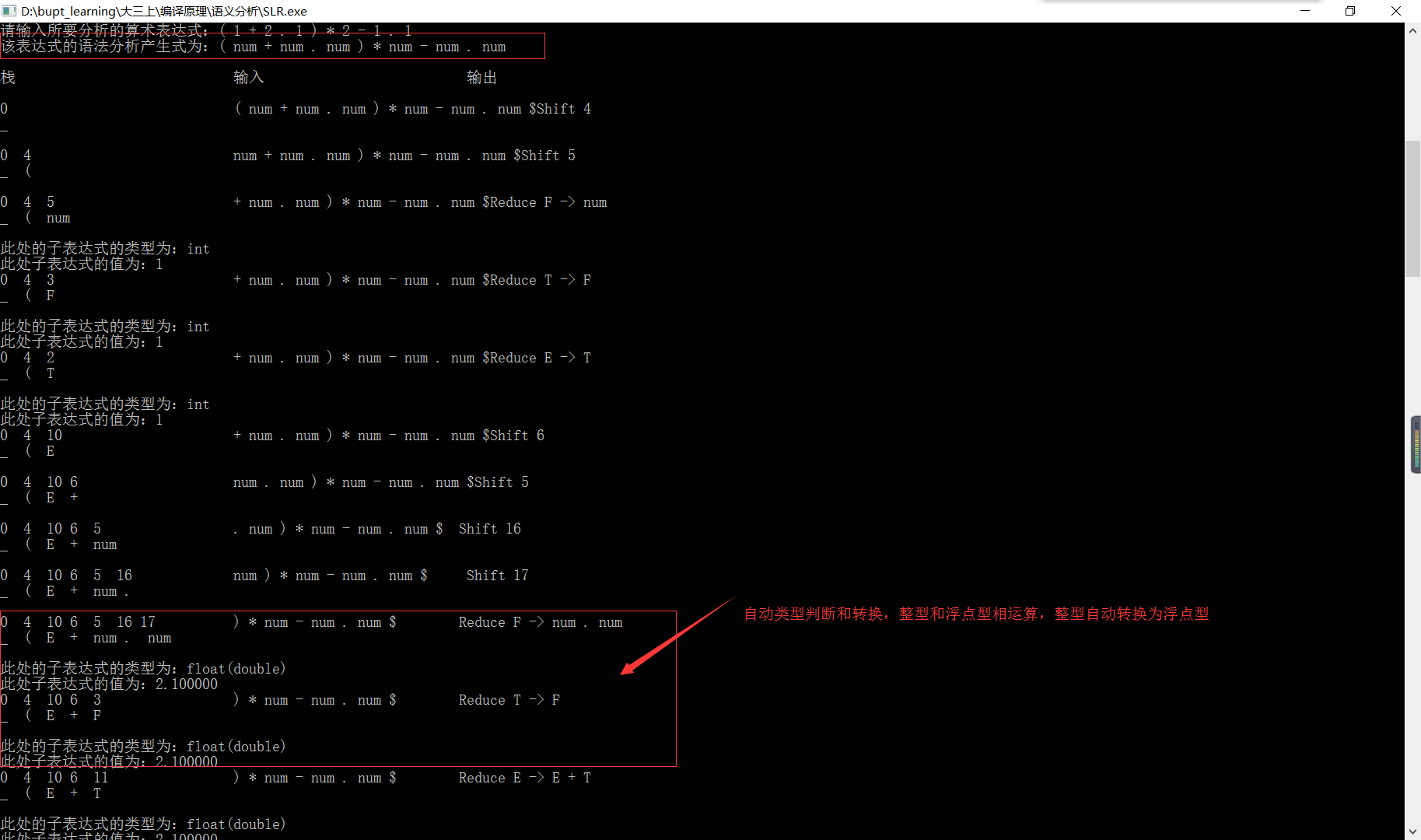
输入为 1 + 2 \* 3，得到结果正确。

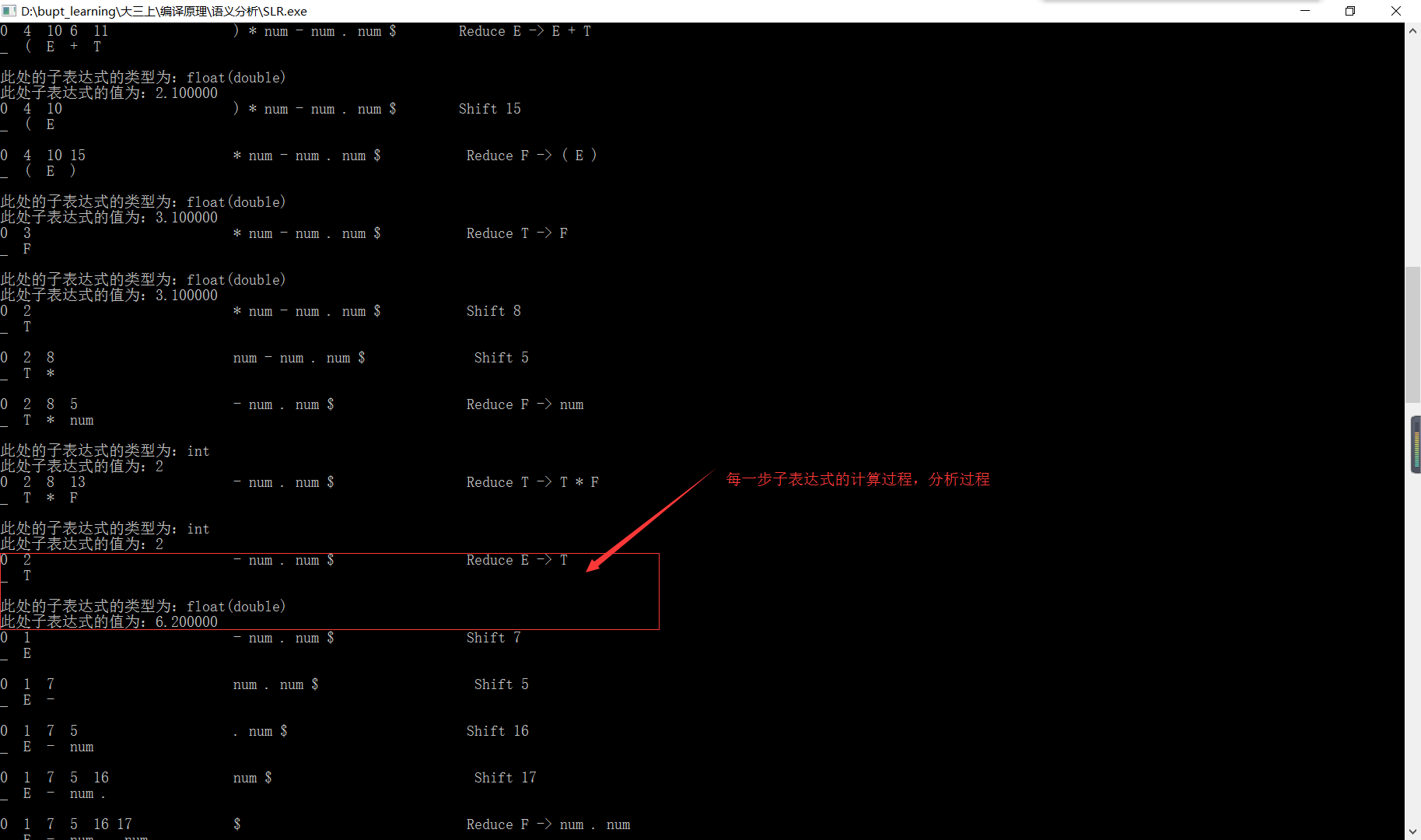




样例2：

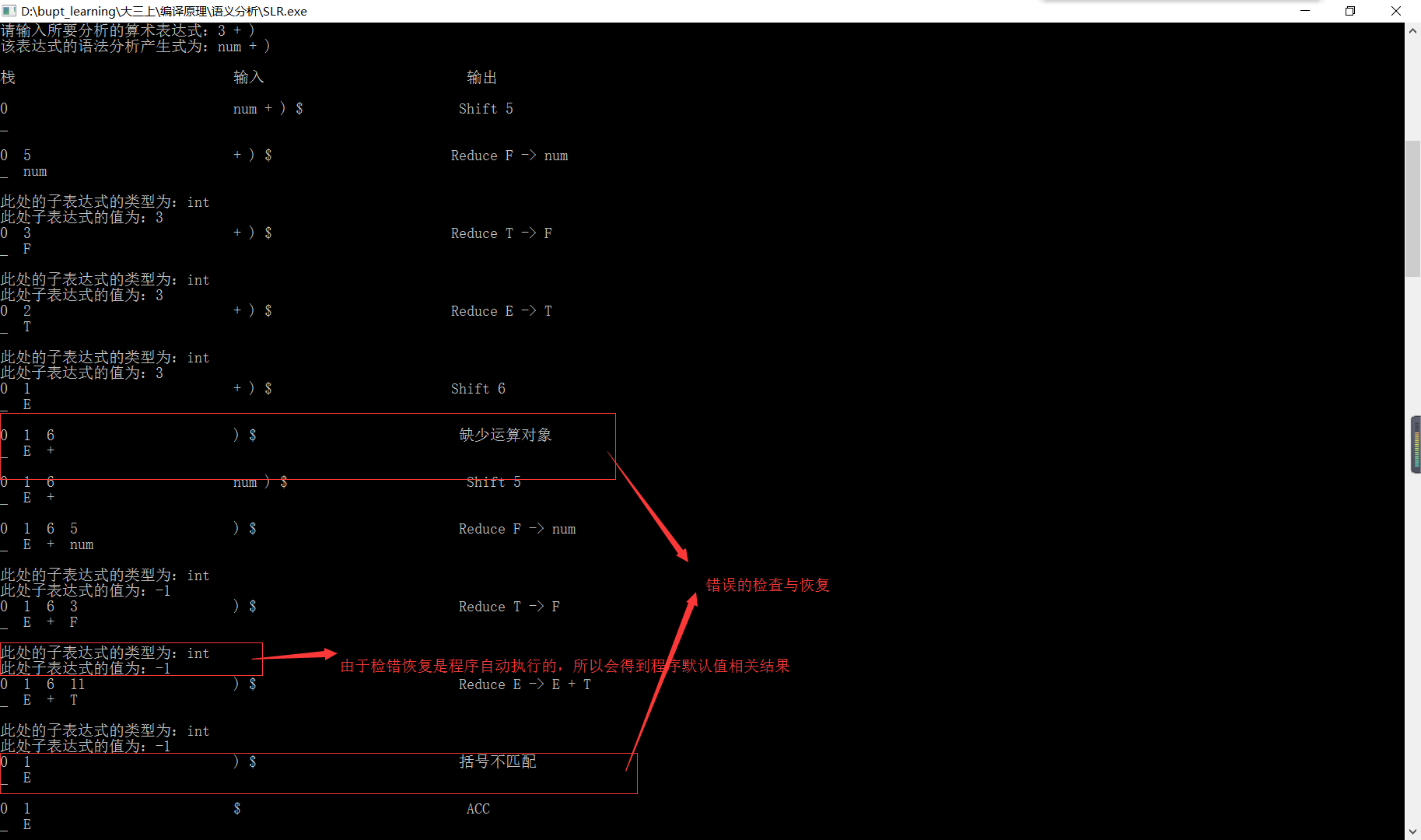
输入为 ( 1 + 2 . 1 ) \* 2 - 1 . 1，得到结果正确。

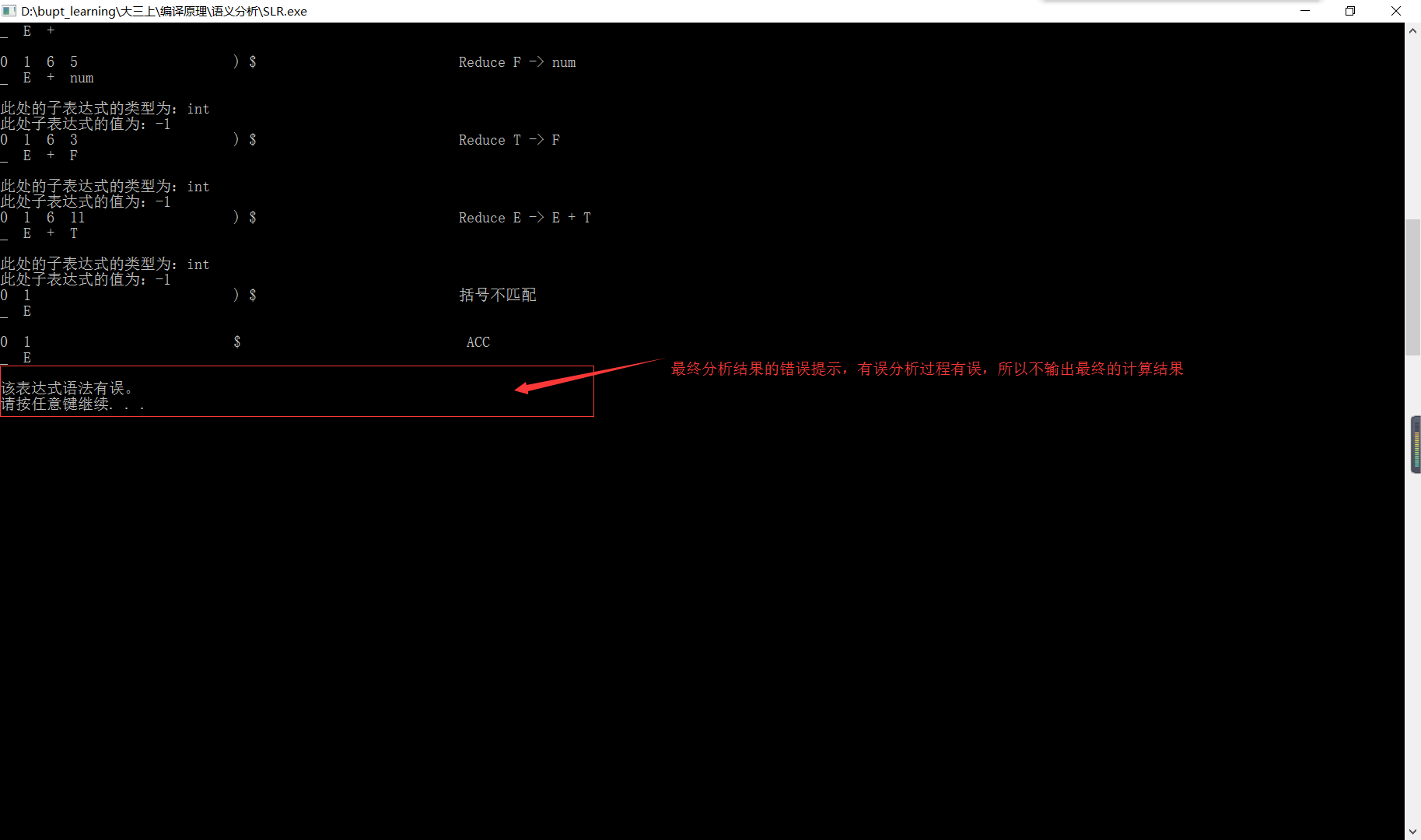




样例3：

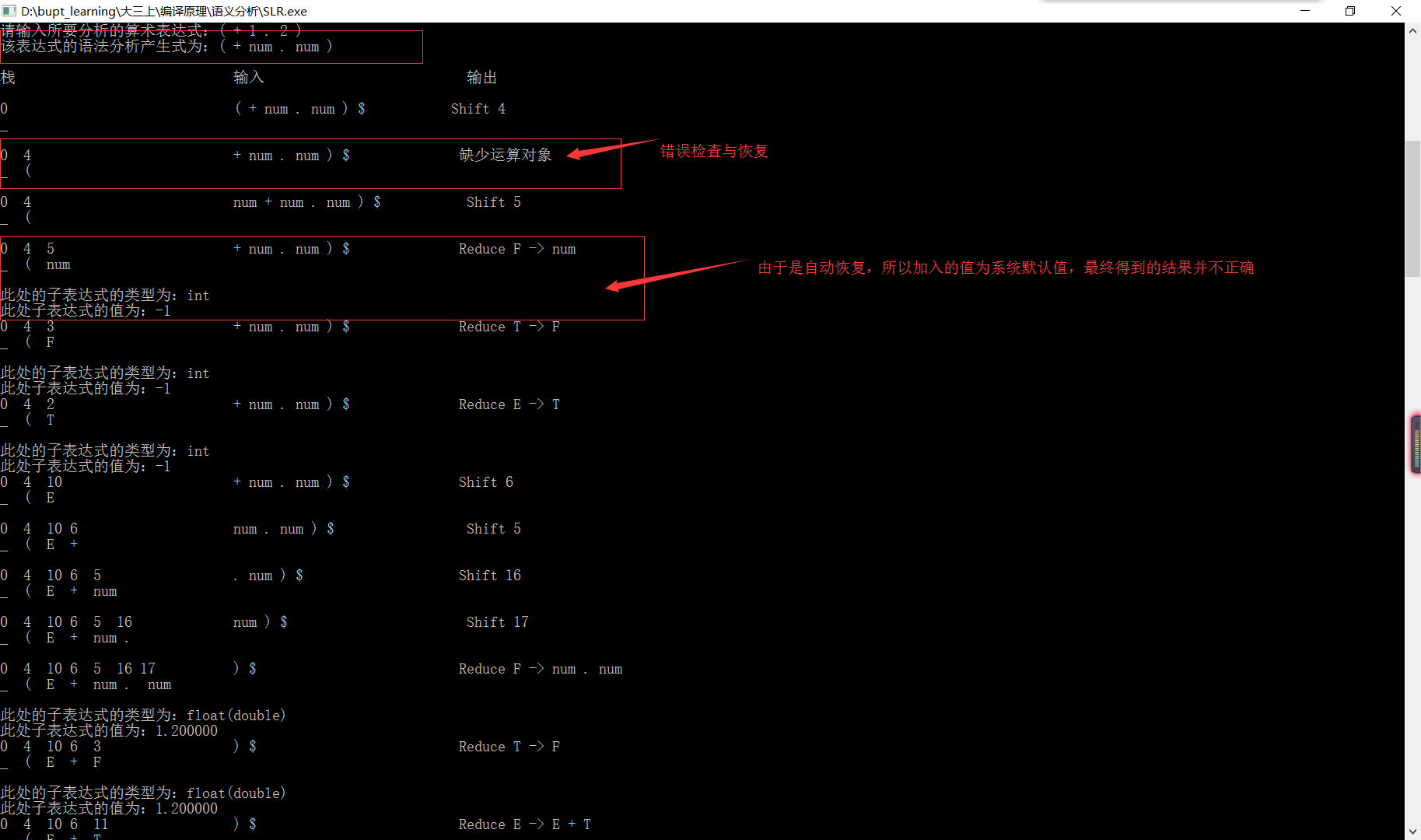
输入为 3 + )，得到结果有误。

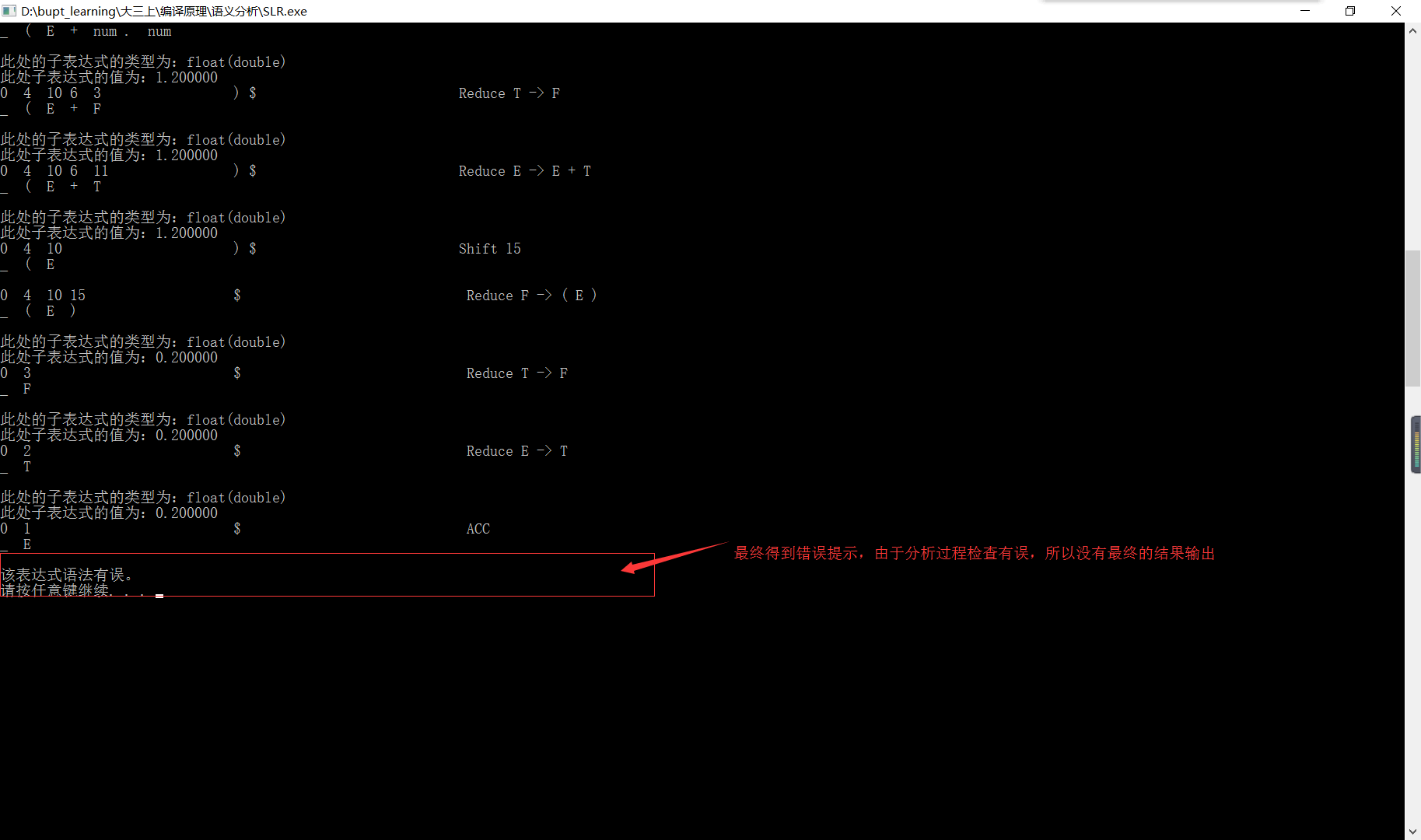




样例4：

输入为( + 1 . 2 )，得到结果有误。





分析说明：

该语法制导定义翻译方案结合SLR(1)文法的预测分析表所得到语义分析程序可以根据SLR(1)预测分析表以及翻译方案准确的对输入的算术表达式进行语义分析，并得出正确与否的结果。对于有错误的表达式，该语义分析程序还会对其进行相应的应急处理和反馈输出。

该语义分析程序默认词法分析没有任何词法上的错误，只存在语法错误，针对语法错误可以进行恢复，但没有结果输出。

对于每一步的子表达式的类型和值都有相应的输出，体现语义分析程序根据语法制导定义翻译方案的翻译过程。

1. **实验总结：**
2. 通过本次语义分析程序的实验，重新熟悉了关于语法分析预测分析表的构造方法以及SLR(1)方法解决移进-规约冲突的过程；
3. 通过本次语义分析程序的实验，熟悉了构造语法制导定义翻译方案的过程，所构造的方案能准确的对算术表达式进行类型判断和求值；
4. 通过对语义分析程序的编码，熟悉了语法制导定义翻译方案的动作在程序中的执行时机和过程，在每个需要规约的生活根据预测分析表对输入的算术表达式进行分析及翻译；
5. 实验中对于词法分析过程忽略，默认词法上没有错误；
6. 对于浮点数的处理，由于所给的文法的限制，小数点必须是一个终结符而存在，所以输入的算术表达式中，将浮点数拆成了三个部分进行分析，在需要翻译再进行计算相应的数值结果；
7. 通过本次实验提高了自身解决问题，思考问题的能力。
8. **附录（源程序）：**

#include <iostream>

#include <iomanip>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <algorithm>

#include <vector>

#include <stack>

#include <map>

using namespace std;

map<string, int> m\_id;

map<int, string> rm\_id;

map<string, int> p\_id;

map<int, string> rp\_id;

vector< pair<string, string> > production;

string Table[20][20];

string ex[50];

string ans\_ex;

int i\_val[50];

struct Res{

int resa = -99999;

double resb = -99999.0;

}res0;

void init()

{

production.push\_back(make\_pair("E'", "E"));

production.push\_back(make\_pair("E", "E + T"));

production.push\_back(make\_pair("E", "E - T"));

production.push\_back(make\_pair("E", "T"));

production.push\_back(make\_pair("T", "T \* F"));

production.push\_back(make\_pair("T", "T / F"));

production.push\_back(make\_pair("T", "F"));

production.push\_back(make\_pair("F", "( E )"));

production.push\_back(make\_pair("F", "num"));

production.push\_back(make\_pair("F", "num . num"));

m\_id["+"] = 0;

m\_id["-"] = 1;

m\_id["\*"] = 2;

m\_id["/"] = 3;

m\_id["("] = 4;

m\_id[")"] = 5;

m\_id["num"] = 6;

m\_id["$"] = 7;

m\_id["."] = 8;

m\_id["E"] = 9;

m\_id["T"] = 10;

m\_id["F"] = 11;

rm\_id[0] = "+";

rm\_id[1] = "-";

rm\_id[2] = "\*";

rm\_id[3] = "/";

rm\_id[4] = "(";

rm\_id[5] = ")";

rm\_id[6] = "num";

rm\_id[7] = "$";

rm\_id[8] = ".";

rm\_id[9] = "E";

rm\_id[10] = "T";

rm\_id[11] = "F";

p\_id["E' -> E"] = 0;

p\_id["E -> E + T"] = 1;

p\_id["E -> E - T"] = 2;

p\_id["E -> T"] = 3;

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

p\_id["T -> T / F"] = 5;

p\_id["T -> F"] = 6;

p\_id["F -> ( E )"] = 7;

p\_id["F -> num"] = 8;

p\_id["F -> num . num"] = 9;

rp\_id[1] = "E -> E + T";

rp\_id[2] = "E -> E - T";

rp\_id[3] = "E -> T";

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

rp\_id[5] = "T -> T / F";

rp\_id[6] = "T -> F";

rp\_id[7] = "F -> ( E )";

rp\_id[8] = "F -> num";

rp\_id[9] = "F -> num . num";

}

void analyze\_table()

{

Table[0][4] = "S4";

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

Table[0][9] = "1";

Table[0][10] = "2";

Table[0][11] = "3";

Table[1][0] = "S6";

Table[1][1] = "S7";

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

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

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

Table[2][2] = "S8";

Table[2][3] = "S9";

Table[2][5] = "R6";

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

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

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

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

Table[3][3] = "R6";

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

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

Table[4][4] = "S4";

Table[4][6] = "S5";

Table[4][9] = "10";

Table[4][10] = "2";

Table[4][11] = "3";

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

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

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

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

Table[5][5] = "R8";

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

Table[5][8] = "S16";

Table[6][4] = "S4";

Table[6][6] = "S5";

Table[6][10] = "11";

Table[6][11] = "3";

Table[7][4] = "S4";

Table[7][6] = "S5";

Table[7][10] = "12";

Table[7][11] = "3";

Table[8][4] = "S4";

Table[8][6] = "S5";

Table[8][11] = "13";

Table[9][4] = "S4";

Table[9][6] = "S5";

Table[9][11] = "14";

Table[10][0] = "S6";

Table[10][1] = "S7";

Table[10][5] = "S15";

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

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

Table[11][2] = "S8";

Table[11][3] = "S9";

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

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

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

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

Table[12][2] = "S8";

Table[12][3] = "S9";

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Table[16][6] = "S17";

Table[17][0] = "R9";

Table[17][1] = "R9";

Table[17][2] = "R9";

Table[17][3] = "R9";

Table[17][5] = "R9";

Table[17][7] = "R9";

Table[0][0] = "E1", Table[0][1] = "E1", Table[0][2] = "E1";

Table[0][3] = "E1", Table[0][5] = "E1", Table[0][7] = "E1";

Table[0][8] = "E1";

Table[1][2] = "E3", Table[1][3] = "E3", Table[1][4] = "E3";

Table[1][5] = "E2", Table[1][6] = "E3", Table[1][8] = "E3";

Table[2][4] = "E3", Table[2][6] = "E3", Table[2][8] = "E3";

Table[3][4] = "E3", Table[3][6] = "E3", Table[3][8] = "E3";

Table[4][0] = "E1", Table[4][1] = "E1", Table[4][2] = "E1";

Table[4][3] = "E1", Table[4][5] = "E1", Table[4][7] = "E1";

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

Table[5][4] = "E3", Table[5][6] = "E3";

Table[6][0] = "E1", Table[6][1] = "E1", Table[6][2] = "E1";

Table[6][3] = "E1", Table[6][5] = "E1", Table[6][7] = "E1";

Table[6][8] = "E1";

Table[7][0] = "E1", Table[7][1] = "E1", Table[7][2] = "E1";

Table[7][3] = "E1", Table[7][5] = "E1", Table[7][7] = "E1";

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

Table[8][0] = "E1", Table[8][1] = "E1", Table[8][2] = "E1";

Table[8][3] = "E1", Table[8][5] = "E1", Table[8][7] = "E1";

Table[8][8] = "E1";

Table[9][0] = "E1", Table[9][1] = "E1", Table[9][2] = "E1";

Table[9][3] = "E1", Table[9][5] = "E1", Table[9][7] = "E1";

Table[9][8] = "E1";

Table[10][2] = "E3", Table[10][3] = "E3", Table[10][4] = "E3";

Table[10][6] = "E3", Table[10][7] = "E4", Table[10][8] = "E3";

Table[11][4] = "E3", Table[11][6] = "E3", Table[11][8] = "E3";

Table[12][4] = "E3", Table[12][6] = "E3", Table[12][8] = "E3";

Table[13][4] = "E3", Table[13][6] = "E3", Table[13][8] = "E3";

Table[14][4] = "E3", Table[14][6] = "E3", Table[14][8] = "E3";

Table[15][4] = "E3", Table[15][6] = "E3", Table[15][8] = "E3";

Table[16][0] = "E5", Table[16][1] = "E5", Table[16][2] = "E5";

Table[16][3] = "E5", Table[16][4] = "E5", Table[16][5] = "E5";

Table[16][7] = "E5", Table[16][8] = "E5";

Table[17][4] = "E3", Table[17][6] = "E3", Table[17][8] = "E3";

}

void output()

{

cout << "预测分析表如下：" << endl;

for(int i = -1; i < 18; i++)

{

for(int j = -1; j < 12; j++)

{

if(i == -1 && j == -1)

cout << std::left << setw(5) << ' ';

else if(i == -1)

cout << std::left << setw(5) << rm\_id[j];

else if(j == -1)

cout << std::left << setw(5) << i;

else

cout << std::left << setw(5) << Table[i][j];

}

puts("");

puts("");

}

}

int cal\_num(string num)

{

int res = 0;

int ex = 1;

for(int i = num.length() - 1; i >=0; i--)

{

res += (num[i] - '0') \* ex;

ex \*= 10;

}

return res;

}

double cal\_float(int num)

{

double res = (double)num;

while(res >= 1)

res /= 10;

return res;

}

void word\_analyze(string str)

{

string tmp = "";

ans\_ex = "";

int cnt = 1;

memset(i\_val, -1, sizeof(i\_val));

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

{

if(str[i] != ' ')

tmp += str[i];

else

{

if(tmp[0] >= '0' && tmp[0] <= '9')

{

i\_val[cnt] = cal\_num(tmp);

ans\_ex += "num";

}

else

ans\_ex += tmp;

cnt++;

ans\_ex += " ";

tmp = "";

}

}

if(tmp[0] >= '0' && tmp[0] <= '9')

{

i\_val[cnt] = cal\_num(tmp);

ans\_ex += "num";

}

else

ans\_ex += tmp;

}

int spl(string str, int mode)

{

if(mode == 1)

{

for(int i = 0; i < 50; i++)

ex[i] = "";

int j = 0;

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

{

if(str[i] != ' ')

ex[j] += str[i];

else

j++;

}

j++;

return j;

}

else

{

int j = 0;

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

{

if(str[i] == ' ')

j++;

}

j++;

return j;

}

}

int toTen(string str)

{

int x = 1, res = 0;

for(int i = str.length() - 1; i > -1; i--)

{

res = res + (str[i] - '0') \* x;

x \*= 10;

}

return res;

}

bool Analyze(string exp\_str)

{

cout << std::left << setw(30) << "栈";

cout << std::left << setw(30) << "输入";

cout << std::left << "输出" << endl;

puts("");

//stack<int> state;

//stack<string> symble;

bool mark = true;

int state[100];

string symble[100];

struct value{

int a;

double b;

}val[100];

bool type[100];

exp\_str = exp\_str + ' ' + '$';

int tmp = spl(exp\_str, 1);

state[0] = 0;

symble[0] = "\_";

int s = state[0];

int top = 0;

int ip = 0;

bool ans = true;

do{

s = state[top];

string a = ex[ip];

if(Table[s][m\_id[a]].substr(0,1) == "S")

{

for(int i = 0; i < top; i++)

cout << std::left << setw(2) << state[i] << ' ';

cout << std::left << setw(30 - top \* 3) << state[top];

for(int i = ip; i < tmp - 1; i++)

cout << ex[i] << ' ';

cout << std::left << setw(30 - (tmp - ip - 1) \* 3) << ex[tmp - 1];

int l = Table[s][m\_id[a]].length();

int new\_state = toTen(Table[s][m\_id[a]].substr(1, l - 1));

cout << "Shift" << ' ' << new\_state << endl;

for(int i = 0; i < top; i++)

cout << std::left << setw(2) << symble[i] << ' ';

cout << std::left << setw(30 - top \* 3) << symble[top] << endl;

puts("");

state[++top] = new\_state;

symble[top] = a;

ip++;

}

else if(Table[s][m\_id[a]].substr(0,1) == "R")

{

for(int i = 0; i < top; i++)

cout << std::left << setw(2) << state[i] << ' ';

cout << std::left << setw(30 - top \* 3) << state[top];

for(int i = ip; i < tmp - 1; i++)

cout << ex[i] << ' ';

cout << std::left << setw(30 - (tmp - ip - 1) \* 3) << ex[tmp - 1];

int l = Table[s][m\_id[a]].length();

int id = toTen(Table[s][m\_id[a]].substr(1, l - 1));

l = rp\_id[id].length();

l = spl(rp\_id[id].substr(5, l - 5), 0);

cout << "Reduce" << ' ' << rp\_id[id] << endl;

for(int i = 0; i < top; i++)

cout << std::left << setw(2) << symble[i] << ' ';

cout << std::left << setw(30 - top \* 3) << symble[top] << endl;

puts("");

if(id == 1)

{

int newtop = top - 2;

if(type[top] == type[top - 2])

type[newtop] = type[top];

else

{

if(!type[top - 2])

val[top - 2].b = (double)val[top - 2].a;

else if(!type[top])

val[top].b = (double)val[top].a;

type[newtop] = true;

}

if(type[newtop])

val[newtop].b = (type[top]?val[top].b:val[top].a) + (type[top - 2]?val[top - 2].b:val[top - 2].a), res0.resb = val[newtop].b;

else

val[newtop].a = val[top].a + val[top - 2].a, res0.resa = val[newtop].a;

if(type[top])

printf("此处的子表达式的类型为：float(double)\n此处子表达式的值为：%f\n", val[top].b);

else

printf("此处的子表达式的类型为：int\n此处子表达式的值为：%d\n", val[top].a);

}

else if(id == 2)

{

int newtop = top - 2;

if(type[top] == type[top - 2])

type[newtop] = type[top];

else

{

if(!type[top - 2])

val[top - 2].b = (double)val[top - 2].a;

else if(!type[top])

val[top].b = (double)val[top].a;

type[newtop] = true;

}

if(type[newtop])

val[newtop].b = (type[top - 2]?val[top - 2].b:val[top - 2].a) - (type[top]?val[top].b:val[top].a), res0.resb = val[newtop].b;

else

val[newtop].a = val[top].a - val[top - 2].a, res0.resa = val[newtop].a;

if(type[top])

printf("此处的子表达式的类型为：float(double)\n此处子表达式的值为：%f\n", val[top].b);

else

printf("此处的子表达式的类型为：int\n此处子表达式的值为：%d\n", val[top].a);

}

else if(id == 3)

{

if(type[top])

printf("此处的子表达式的类型为：float(double)\n此处子表达式的值为：%f\n", val[top].b);

else

printf("此处的子表达式的类型为：int\n此处子表达式的值为：%d\n", val[top].a);

}

else if(id == 4)

{

int newtop = top - 2;

if(type[top] == type[top - 2])

type[newtop] = type[top];

else

{

if(!type[top - 2])

val[top - 2].b = (double)val[top - 2].a;

else if(!type[top])

val[top].b = (double)val[top].a;

type[newtop] = true;

}

if(type[newtop])

val[newtop].b = (type[top]?val[top].b:val[top].a) \* (type[top - 2]?val[top - 2].b:val[top - 2].a);

else

val[newtop].a = val[top].a \* val[top - 2].a;

if(type[top])

printf("此处的子表达式的类型为：float(double)\n此处子表达式的值为：%f\n", val[top].b);

else

printf("此处的子表达式的类型为：int\n此处子表达式的值为：%d\n", val[top].a);

}

else if(id == 5)

{

int newtop = top - 2;

if(type[top] == type[top - 2])

type[newtop] = type[top];

else

{

if(!type[top - 2])

val[top - 2].b = (double)val[top - 2].a;

else if(!type[top])

val[top].b = (double)val[top].a;

type[newtop] = true;

}

if(type[newtop])

val[newtop].b = (type[top]?val[top].b:val[top].a) / (type[top - 2]?val[top - 2].b:val[top - 2].a);

else

val[newtop].a = val[top].a / val[top - 2].a;

if(type[top])

printf("此处的子表达式的类型为：float(double)\n此处子表达式的值为：%f\n", val[top].b);

else

printf("此处的子表达式的类型为：int\n此处子表达式的值为：%d\n", val[top].a);

}

else if(id == 6)

{

if(type[top])

printf("此处的子表达式的类型为：float(double)\n此处子表达式的值为：%f\n", val[top].b);

else

printf("此处的子表达式的类型为：int\n此处子表达式的值为：%d\n", val[top].a);

}

else if(id == 8)

{

type[top] = false;

val[top].a = i\_val[ip];

if(type[top])

printf("此处的子表达式的类型为：float(double)\n此处子表达式的值为：%f\n", val[top].b);

else

printf("此处的子表达式的类型为：int\n此处子表达式的值为：%d\n", val[top].a);

}

else if(id == 9)

{

type[top - 2] = true;

val[top - 2].b = cal\_float(i\_val[ip]) + i\_val[ip - 2];

if(type[top - 2])

printf("此处的子表达式的类型为：float(double)\n此处子表达式的值为：%f\n", val[top - 2].b);

else

printf("此处的子表达式的类型为：int\n此处子表达式的值为：%d\n", val[top - 2].a);

}

else if(id == 7)

{

type[top - 2] = type[top - 1];

if(type[top - 2])

val[top - 2].b = val[top - 1].b;

else

val[top - 2].b = val[top - 1].b;

if(type[top - 2])

printf("此处的子表达式的类型为：float(double)\n此处子表达式的值为：%f\n", val[top - 2].b);

else

printf("此处的子表达式的类型为：int\n此处子表达式的值为：%d\n", val[top - 2].a);

}

while(l > 0)

{

top--;

l--;

}

s = state[top];

int new\_state = toTen(Table[s][m\_id[rp\_id[id].substr(0, 1)]]);

state[++top] = new\_state;

symble[top] = rp\_id[id].substr(0, 1);

}

else if(Table[s][m\_id[a]] == "ACC")

{

for(int i = 0; i < top; i++)

cout << std::left << setw(2) << state[i] << ' ';

cout << std::left << setw(30 - top \* 3) << state[top];

for(int i = ip; i < tmp - 1; i++)

cout << ex[i] << ' ';

cout << std::left << setw(30 - (tmp - ip - 1) \* 3) << ex[tmp - 1];

cout << "ACC" << endl;

for(int i = 0; i < top; i++)

cout << std::left << setw(2) << symble[i] << ' ';

cout << std::left << setw(30 - top \* 3) << symble[top] << endl;

puts("");

return ans;

}

else

{

for(int i = 0; i < top; i++)

cout << std::left << setw(2) << state[i] << ' ';

cout << std::left << setw(30 - top \* 3) << state[top];

for(int i = ip; i < tmp - 1; i++)

cout << ex[i] << ' ';

cout << std::left << setw(30 - (tmp - ip - 1) \* 3) << ex[tmp - 1];

if(Table[s][m\_id[a]] == "E1")

{

ex[--ip] = "num";

cout << "缺少运算对象" << endl;

}

else if(Table[s][m\_id[a]] == "E2")

{

ip++;

cout << "括号不匹配" << endl;

}

else if(Table[s][m\_id[a]] == "E3")

{

ex[--ip] = "+";

cout << "缺少运算符号" << endl;

}

else if(Table[s][m\_id[a]] == "E4")

{

ex[--ip] = ")";

cout << "缺少右括号" << endl;

}

else if(Table[s][m\_id[a]] == "E5")

{

ex[--ip] = "num";

mark = false;

cout << "词法错误" << endl;

}

ans = false;

for(int i = 0; i < top; i++)

cout << std::left << setw(2) << symble[i] << ' ';

cout << std::left << setw(30 - top \* 3) << symble[top] << endl;

puts("");

}

}while(1);

}

int main()

{

init();

analyze\_table();

output();

cout << "请输入所要分析的算术表达式：";

string exp\_str;

getline(cin, exp\_str);

word\_analyze(exp\_str);

cout << "该表达式的语法分析产生式为：" << ans\_ex << endl;

puts("");

if(Analyze(ans\_ex))

{

cout << "该表达式语法正确。" << endl;

puts("");

if(res0.resa != -99999)

printf("计算结果为：%d\n", res0.resa);

else

printf("计算结果为：%f\n", res0.resb);

}

else

cout << "该表达式语法有误。" << endl;

system("pause");

return 0;

}