程序设计 3

题目：语义分析程序的设计与实现。

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)

实现要求：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：

① 分析过程中所用产生式。

② 识别出的子表达式的类型。

③ 识别出子表达式的值。

（3） 实验方法：手工编写分析程序。。

输入：输入需要分析的符号串

输出：验证输入串是不是符合该语言语法规则的一个程序，若是，则输出其分析树，并输出符号串表达式的类型以及值，若不是，则表明输入串中存在语法错误，需要报告错误的性质和位置。

设计过程：

1. 通过 LR 分析方法进行语义分析，其主要由以下几个部分构成：

（1） 输入缓冲区：存放待分析的输入符号串，并以$作为符号串的结束标志。

（2） 输出：是 LR 分析控制程序分析输入符号串的过程中所采用的动作序列。

（3） 栈：由状态栈和数值栈构成（State 和 Value）

（4） LR(0)项目集规范族及识别其所有活前缀的 DFA

（5） 分析表：分析表是一个确定有限自动机的状态转移表，该自动机的字母表即由全部文法符号构成的集合。其中终结符号及$对应的列构成动作（Action）表，非终结符好对应的列构成状态转移（goto）表

Goto[Sm,A]保存了当前状态 Sm 相对于非终结符号 A 的后继状态。

Action[Sm,ai]规定了当前状态 Sm 面临输入符号 ai 时应采取的分析动作，主要有移进，归约，接收，出错几个动作。

（6） 根据文法编写对应的翻译方案。

① E->E’+T{E.val = E’.val + T.val}
{if (E’.type == real || T.type == real) E.type = real; else E.type = integer;}

② E->E’-T{E.val = E’.val - T.val}
{if (E’.type == real || T.type == real) E.type = real; else E.type = integer;}

③ E ->T{E.val = T.val}{E.type = T.type}

④ T->T’*F{T.val = T’.val * F.val}
{if (T’.type == real || F.type == real) T.type = real; else T.type = integer;}

⑤ T->T’/F{T.val = T’.val / F.val}
{if (T’.type == real || F.type == real) T.type = real; else T.type = integer;}

⑥ T ->F{T.val = F.val}{T.type = F.type}

⑦ F ->(E){F.val = E.val}{F.type = E.type}

⑧ F ->num{F.val = num.val} {F.type = integer}

⑨ F ->num.num{F.val = num.num.val} {F.type = real}

总结：整体的过程为人工计算出表达式文法的所有活前缀的DFA，再由DFA构造出LR分析表，构造Action[][]和Goto[][]两个表，然后初始化VT（终结符集）、VN（非终结符集）、P（产生式）表，以字符单位依次读入输入串并根据分析表进行分析，分析过程中进行出错恢复处理，从而完成语法分析。在进行归约时，调用Translate()方法对表达式的类型进行判断，对表达式的值进行计算，最后更新两个栈中的内容。

2. 构造拓广文法，并求出文法中各非终结符的FIRST、FOLLOW集

拓广文法：(0)E'->E (5)T->T/F

(1)E->E+T (6)T->F

(2)E->E-T (7)F->(E)

(3)E->T (8)F->num

(4)T->T*F (9)F->num.num

各非终结符的FIRST、FOLLOW集：

	E'	E	T	F
FIRST	(, num	(, num	(, num	(, num
FOLLOW	$	$,), +, -	$,), +, -, *, /	$,), +, -, *, /

3. 构造LR(0)项目集规范族及识别其所有活前缀的DFA

根据书本算法4.6中所描述，当出现I={X->α·bβ，A->α·,B->β·}类似的项目集时会出现移进-归约和归约-归约冲突，为解决冲突，需要分析所有含有A和B的句型，考查这些句型中跟在A和B后面出现的终结符号分别有哪些,，即FOLLOW(A)和FOLLOW(B)，如果这两个集合不相交，也不含有b，则当前状态I面临输入符号为a时，采用的分析动作：

- 当a=b时，采取移进动作，把b移进栈里。
- 当a∈FOLLOW(A)时，采取归约动作，用产生式A->α进行归约。
- 当a∈FOLLOW(B)时，采取归约动作，用产生时B->β进行归约。

错误处理（共7种）：

e1：状态 0 4 6 7 8 9期待(或者num，却输入为+ - * /或$缺少运算对象，把假想的num压入栈，并转移到状态5；

e2：状态0 1 2 4 6 7 8 9期待运算对象首字符或运算符号，却输入为)括号不匹配，直接略过；

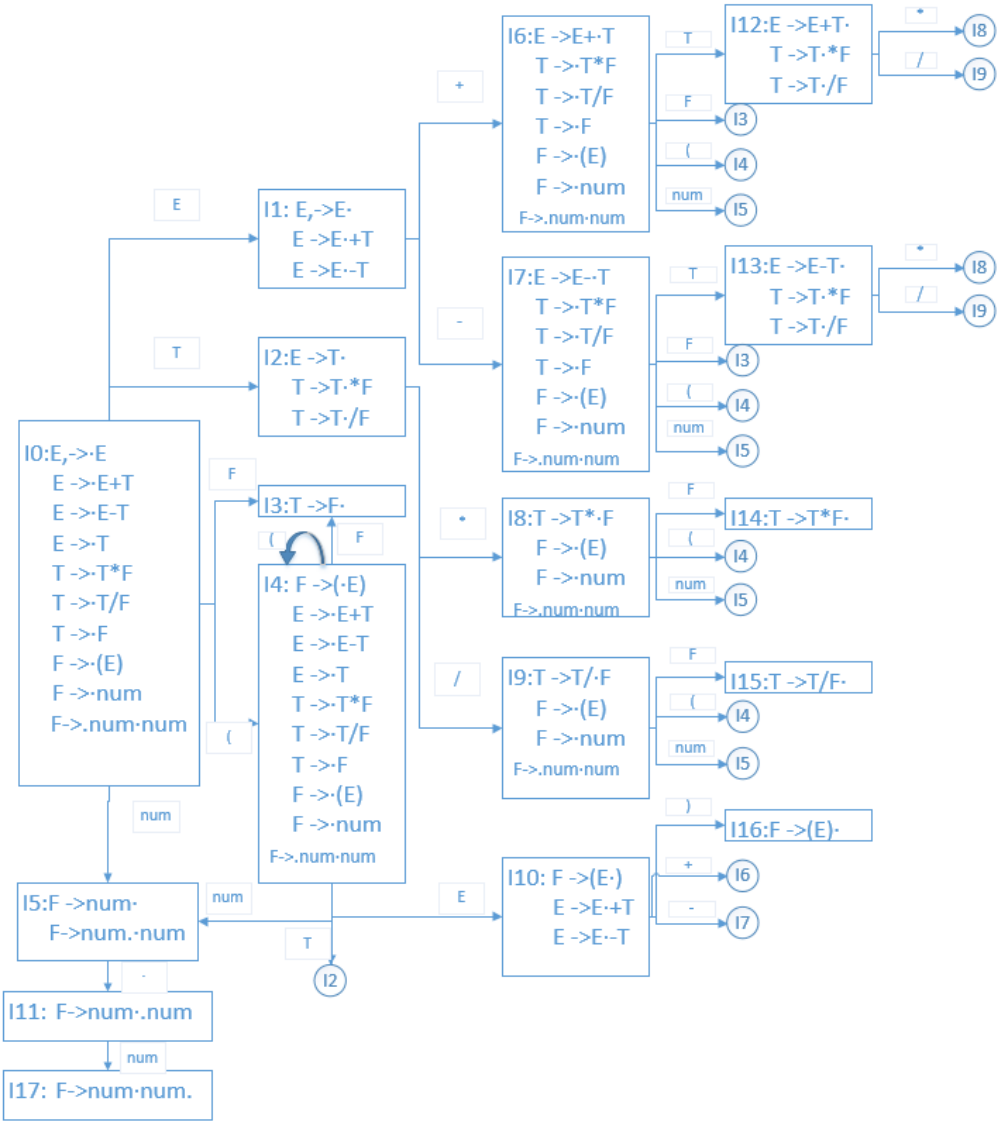
e3：状态 1 2 10 11 12期待输入符号为运算符号或)却输入（或者num缺少运算符，将假想的"+"压入栈，并转移到状态6；

e4：状态10期待输入符号为运算符号或)却输入$，缺少右括号，将假想的")"压入栈，并转移到状态16；

e5：状态1 10期待输入+或-却输入*或/，运算符号错误将假想的"+"压入栈，并转移到状态6；

e6：归约时输入为(或者num，直接跳过；

e7：小数点后缺数字，将0放置到小数点后，并转移到状态17；



4. 构造 LR 分析表：

状态	Action									Goto		
	+	-	*	/	(	)	.	num	$	E	T	F
0	e1	e1	e1	e1	S4	e2	e1	S5	e1	1	2	3
1	S6	S7	e5	e5	e3	e2	e1	e3	ACC			
2	R3	R3	S8	S9	e3	R3	e1	e3	R3			
3	R6	R6	R6	R6	e6	R6	e1	e6	R6			
4	e1	e1	e1	e1	S4	e2	e1	S5	e1	10	2	3
5	R8	R8	R8	R8	e6	R8	S11	e6	R8			
6	e1	e1	e1	e1	S4	e2	e1	S5	e1		12	3
7	e1	e1	e1	e1	S4	e2	e1	S5	e1		13	3
8	e1	e1	e1	e1	S4	e2	e1	S5	e1			14
9	e1	e1	e1	e1	S4	e2	e1	S5	e1			15
10	S6	S7	e5	e5	e3	S16	e1	e3	e4			
11	e7	e7	e7	e7	e7	e7	e7	S17	e7			
12	R1	R1	S8	S9	e3	R1	e1	e3	R1			
13	R2	R2	S8	S9	e3	R2	e1	e3	R2			
14	R4	R4	R4	R4	e6	R4	e1	e6	R4			
15	R5	R5	R5	R5	e6	R5	e1	e6	R5			
16	R7	R7	R7	R7	e6	R7	e1	e6	R7			
17	R9	R9	R9	R9	e6	R9	e1	e6	R9			

5. 语义分析程序源代码：

```
#include<iostream>
#include<cmath>
#include<vector>
#include<sstream>
#include<string>
using namespace std;

vector<char> VN; //非终结符集合
vector<string> VT;  //终结符集合
vector<string>P; //产生式集合
int State[50];       //状态栈
double Value[50];   //数值栈
int Type[50];       //数值类型，为 1 则为 integer,为 2 则为 real
int top = 0;        //栈顶指针

//初始化 Action 表
string Action[20][20] = {
 { "e1","e1","e1","e1","S4","e2","e1","S5","e1" },
 { "S6","S7","e5","e5","e3","e2","e1","e3","ACC" },
 { "R3","R3","S8","S9","e3","R3","e1","e3","R3" },
 { "R6","R6","R6","R6","e6","R6","e1","e6","R6" },
 { "e1","e1","e1","e1","S4","e2","e1","S5","e1" },
 { "R8","R8","R8","R8","e6","R8","S11","e6","R8" },
 { "e1","e1","e1","e1","S4","e2","e1","S5","e1" },
 { "e1","e1","e1","e1","S4","e2","e1","S5","e1" },
 { "e1","e1","e1","e1","S4","e2","e1","S5","e1" },
 { "e1","e1","e1","e1","S4","e2","e1","S5","e1" },
 { "S6","S7","e5","e5","e3","S16","e1","e3","e4" },
 { "e7","e7","e7","e7","e7","e7","e7","S17","e7" },
 { "R1","R1","S8","S9","e3","R1","e1","e3","R1" },
 { "R2","R2","S8","S9","e3","R2","e1","e3","R2" },
 { "R4","R4","R4","R4","e6","R4","e1","e6","R4" },
 { "R5","R5","R5","R5","e6","R5","e1","e6","R5" },
 { "R7","R7","R7","R7","e6","R7","e1","e6","R7" },
 { "R9","R9","R9","R9","e6","R9","e1","e6","R9" }
};

//初始化 Goto 表
int Goto[20][4] = {
 { 1,2,3 },
 { -1,-1,-1 },
 { -1,-1,-1 },
 { -1,-1,-1 },
 { 10,2,3 },
 { -1,-1,-1 },
 { -1,12,3 },
 { -1,13,3 },
 { -1,-1,14 },
 { -1,-1,15 },
 { -1,-1,-1 },
```

```
 { -1,-1,-1 },
 { -1,-1,-1 },
 { -1,-1,-1 },
 { -1,-1,-1 },
 { -1,-1,-1 },
};

//输入产生式
void Input_P() {
 VN.push_back('E');
 VN.push_back('T');
 VN.push_back('F');
 VT.push_back("+");
 VT.push_back("-");
 VT.push_back("*");
 VT.push_back("/");
 VT.push_back("(");
 VT.push_back(")");
 VT.push_back(".");
 VT.push_back("num");
 P.push_back("E->E+T");
 P.push_back("E->E-T");
 P.push_back("E->T");
 P.push_back("T->T*F");
 P.push_back("T->T/F");
 P.push_back("T->F");
 P.push_back("F->(E)");
 P.push_back("F->num");
 P.push_back("F->num.num");
}

//将 string 型转换为 int 型
int SToI(string str) {
 int num;
 stringstream stream;
 stream << str;
 stream >> num;
 return num;
}

//定位 VN 集中的非终结符, 返回其下标
int findVN(vector<char> VN, char c) {
 vector<char>::iterator it;
 for (it = VN.begin(); it != VN.end(); it++)
 {
     if ((*it)==c)
     {
         return it - VN.begin();
     }
 }
 return -1;
```

```
}

//定位VT集中的终结符，返回其下标
int findVT(vector<string>VT, string c) {
 vector<string>::iterator it;
 for ( it = VT.begin(); it !=VT.end(); it++)
 {
     if ((*it) == c) {
         return it - VT.begin();
     }
 }
 return -1;
}

//翻译方案
void Translate(int number) {
 int i; double j;//临时变量
 switch (number)
 {
 case 1://E->E+T
     Value[top - 2] = Value[top - 2] + Value[top];
     if (Type[top]==2||Type[top-2]==2)
     {
         Type[top - 2] = 2;
     }
     else {
         Type[top - 2] = 1;
     }
     break;
 case 2://E->E-T
     Value[top - 2] = Value[top - 2] - Value[top];
     if (Type[top] == 2 || Type[top - 2] == 2)
     {
         Type[top - 2] = 2;
     }
     else {
         Type[top - 2] = 1;
     }
     break;
 case 3://E->T
     Value[top] = Value[top];
     Type[top] = Type[top];
     break;
 case 4://T->T*F
     Value[top - 2] = Value[top - 2] * Value[top];
     if (Type[top] == 2 || Type[top - 2] == 2)
     {
         Type[top - 2] = 2;
     }
     else {
         Type[top - 2] = 1;
```

```
    }
    break;
case 5://T->T/F
    if (Type[top] == 2 || Type[top - 2] == 2 || (int)Value[top - 2] %
(int)Value[top] != 0) {
        //不能够整除则为 real
        Type[top - 2] = 2;
    }
    else
    {
        Type[top - 2] = 1;
    }
    Value[top - 2] = Value[top - 2] / Value[top];
    break;
case 6://T->F
    Value[top] = Value[top];
    Type[top] = Type[top];
    break;
case 7://F->(E)
    Value[top - 2] = Value[top - 1];
    Type[top - 2] = Type[top - 1];
    break;
case 8://F->num
    Value[top] = Value[top];
    Type[top] = 1;
    break;
case 9://F->num.num
    j = Value[top];
    //计算小数有几位
    for ( i = 0;j>=1 ; i++)
    {
        j = j / 10;
    }
    Value[top - 2] = Value[top - 2] + Value[top] * pow(0.1, i);
    Type[top - 2] = 2;
    break;
default:
    break;
}
}

//输出状态栈 0 和数值栈 1
void output(int number) {
int i;
if (number==0)
{
    for ( i = 0; i <= top; i++)
    {
        cout << State[i] << " ";
    }
}
```

```
else
{
    for ( i = 0; i <= top; i++)
    {
        cout << Value[i] << " ";
    }
}
}

void LR() {
int i, j, k;    //临时变量
int p = 0;      //输入串下标
int length = 0; //输入字符长度
int S;          //状态栈栈顶内容
char a;         //依次分析输入串各个字符
string temp;
VT.push_back("$");
string input;
cout << "请输入要分析的符号串：" << endl;
cin >> input;
cout << endl;
input += "$";
State[top] = 0;//置状态栈栈顶为 0
cout << "开始分析：" << endl;
while (1)
{
    S = State[top];
    a = input[p];
    temp.clear();
    cout << "State: ";
    output(0);
    cout << endl;
    cout << "  Val: ";
    output(1);
    cout << endl << "输入：" << input.substr(p) <<"\t"<< "分析动作：";
    if (a == '+'|| a == '-' || a == '*' || a == '/' ||
        a == '('|| a == ')' || a == '.' || a == '$')
    {
        length = 1;
    }
    else
    {
        j = p;
        i = 0;
        length = 0;
        while (isdigit(a))
        {
            i = i * 10 + a - '0';
            length++;
            j++;
            a = input[j];
```

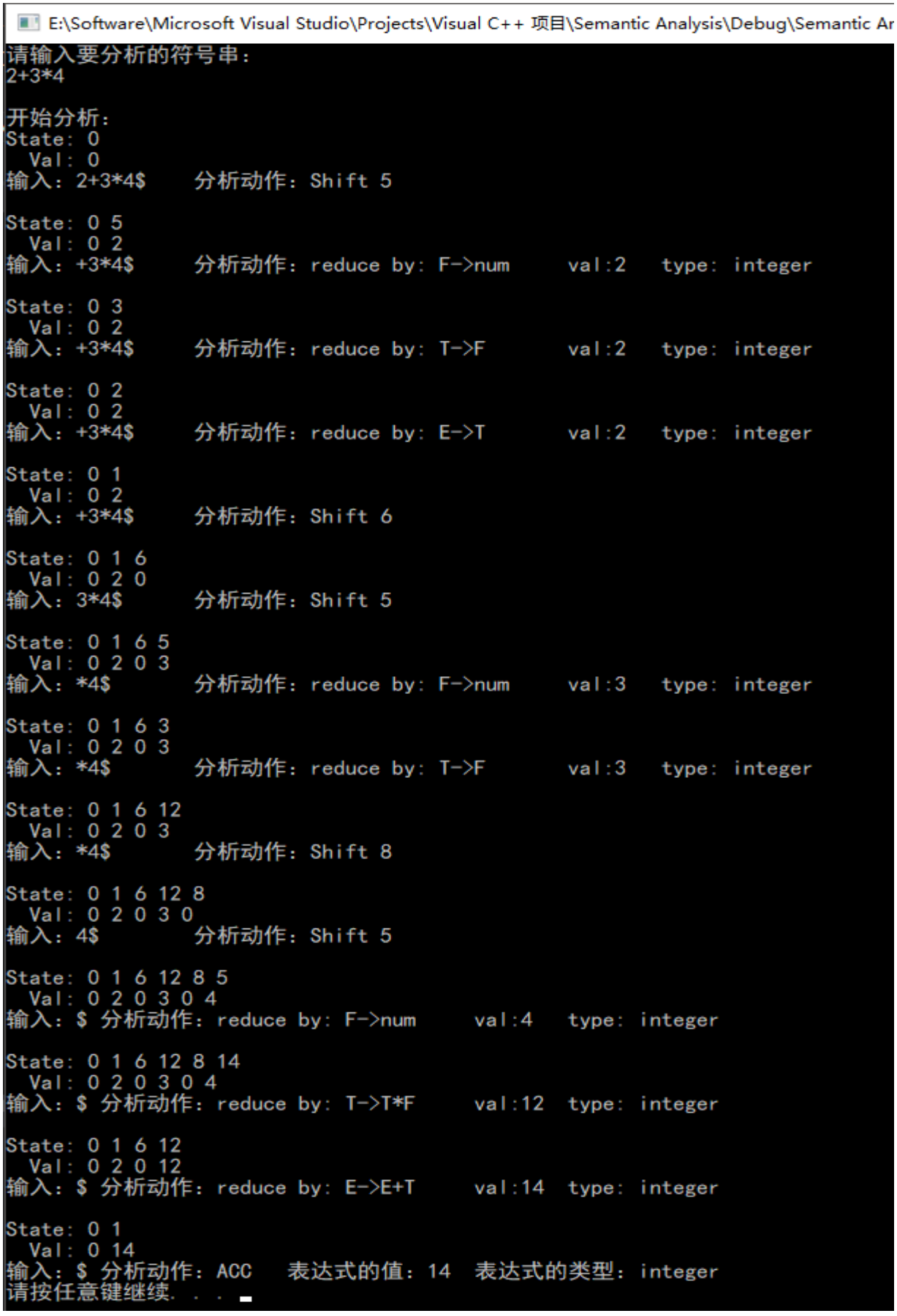
```
    }
    a = 'n';
}
if (a == 'n')
{
    temp = "num";
}
else
{
    temp += a;
}
k = findVT(VT, temp);
if (Action[S][k][0]=='S')//移进
{
    top++;
    if (temp=="num")
    {
        Value[top] = i;
    }
    else
    {
        Value[top] = 0;
    }
    State[top] = SToI(Action[S][k].substr(1));
    p += length;
    cout << "Shift " << Action[S][k].substr(1) << endl << endl;
    temp.clear();
}
else if (Action[S][k][0]=='R')//归约
{
    int r = SToI(Action[S][k].substr(1));//获得需要归约的产生式序号
    int count = 0;//用于计算A->|b|中|b|的长度
    //归约前进行翻译动作
    Translate(r);
    string str;
    str = P[r - 1].substr(3);//提取产生式箭头右部部分
    int sp=0;
    string temp1;
    while (sp<str.length())
    {
        temp1 += str[sp];
        if (findVN(VN,str[sp])>=0||findVT(VT,temp1)>=0)
        {
            count++;
            temp1.clear();
        }
        sp++;
    }
    temp1.clear();
    //栈顶弹出|b|个元素
    for (int m = 0; m < count; m++)
```

```
    {
        top--;
    }
    //_S 为当前栈顶内容
    int _S = State[top];
    string A = P[r - 1].substr(0, 1);
    //将 Goto[_S,A]压入 State 栈中
    int vn = findVN(VN, A[0]);
    top++;
    State[top] = Goto[_S][vn];
    //输出 A->b
    cout << "reduce by: " << P[r - 1] << "\tval:" << Value[top];
    cout << "\ttype: ";
    Type[top] == 1? cout << "integer" : cout << "real";
    cout << endl << endl;
}
else if (Action[S][k][0]=='A')
{
    cout << "ACC" << "\t 表达式的值: " << Value[top];
    cout << "\t 表达式的类型: ";
    Type[top] == 1 ? cout << "integer" : cout << "real";
    cout << endl;
    break;
}
else if (Action[S][k][0]=='e')
{
    cout << "error :" << endl;
    if (Action[S][k][1]=='1')
    {//状态 0 4 6 7 8 9 期待(或者 num, 却输入为+ - * /或$
     //缺少运算对象, 把假想的 num 压入栈, 并转移到状态 5
        top++;
        Value[top] = 0;
        State[top] = 5;
        cout << "缺少运算对象, 将 num 压入栈" << endl;
    }
    else if (Action[S][k][1] == '2')
    {//状态 0 1 2 4 6 7 8 9 期待运算对象首字符或运算符号, 却输入为)
     //括号不匹配, 直接略过
        p++;
        cout << "括号不匹配, 指向下一个符号" << endl;
    }
    else if (Action[S][k][1] == '3')
    {//状态 1 2 10 11 12 期待输入符号为运算符号或)却输入 (或者 num
     //缺少运算符, 将假想的"+"压入栈, 并转移到状态 6
        Value[top] = '+';
        State[top] = 6;
        cout << "缺少运算符号, 将+压入栈" << endl;
    }
    else if (Action[S][k][1] == '4')
    {//状态 10 期待输入符号为运算符号或)却输入$
     //缺少右括号, 将假想的")"压入栈, 并转移到状态 16
```

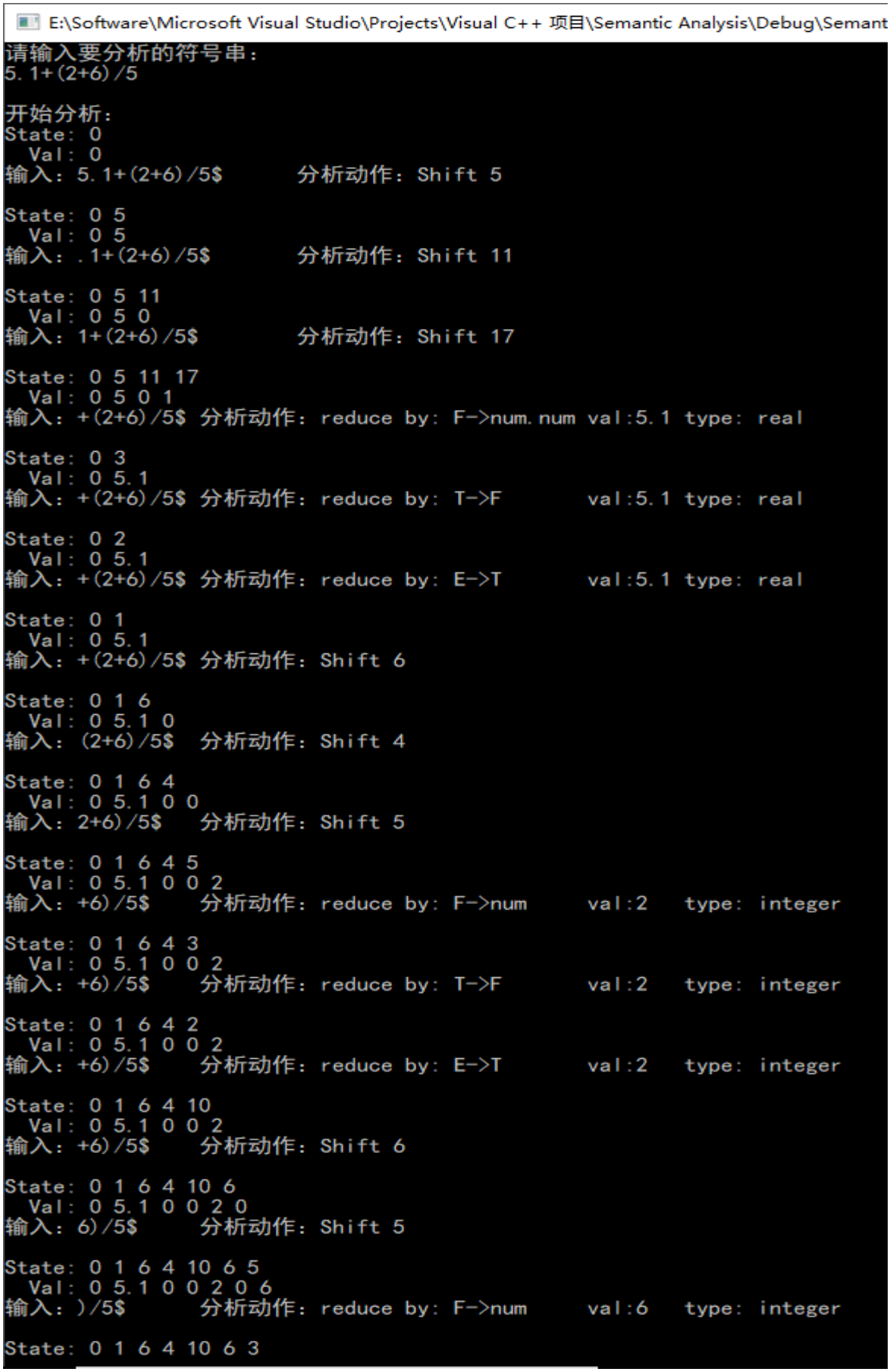
```
                Value[top] = ')';
                State[top] = 16;
                cout << "缺少右括号，将)压入栈" << endl;
            }
            else if (Action[S][k][1] == '5')
            {//状态 1 10 期待输入+或-却输入*或/
             //运算符号错误，将假想的"+"压入栈，并转移到状态 6
                Value[top] = '+';
                State[top] = 6;
                cout << "缺少+/-，将+压入栈" << endl;
            }
            else if (Action[S][k][1] == '6')
            {//归约时输入为(或者 num，直接跳过
                p++;
                cout << "归约时产生多余符号，直接跳过" << endl;
            }
            else
            {//小数点后缺数字，将 0 放置到小数点后，并转移到状态 17
                top++;
                Value[top] = 0;
                State[top] = 17;
                cout << "小数点后缺数字" << endl;
            }
            cout << endl;
        }
    }
    return;
}
void main() {
 Input_P();
 LR();
 system("pause");
}
```

实验结果：

输入 2+3*4，无报错

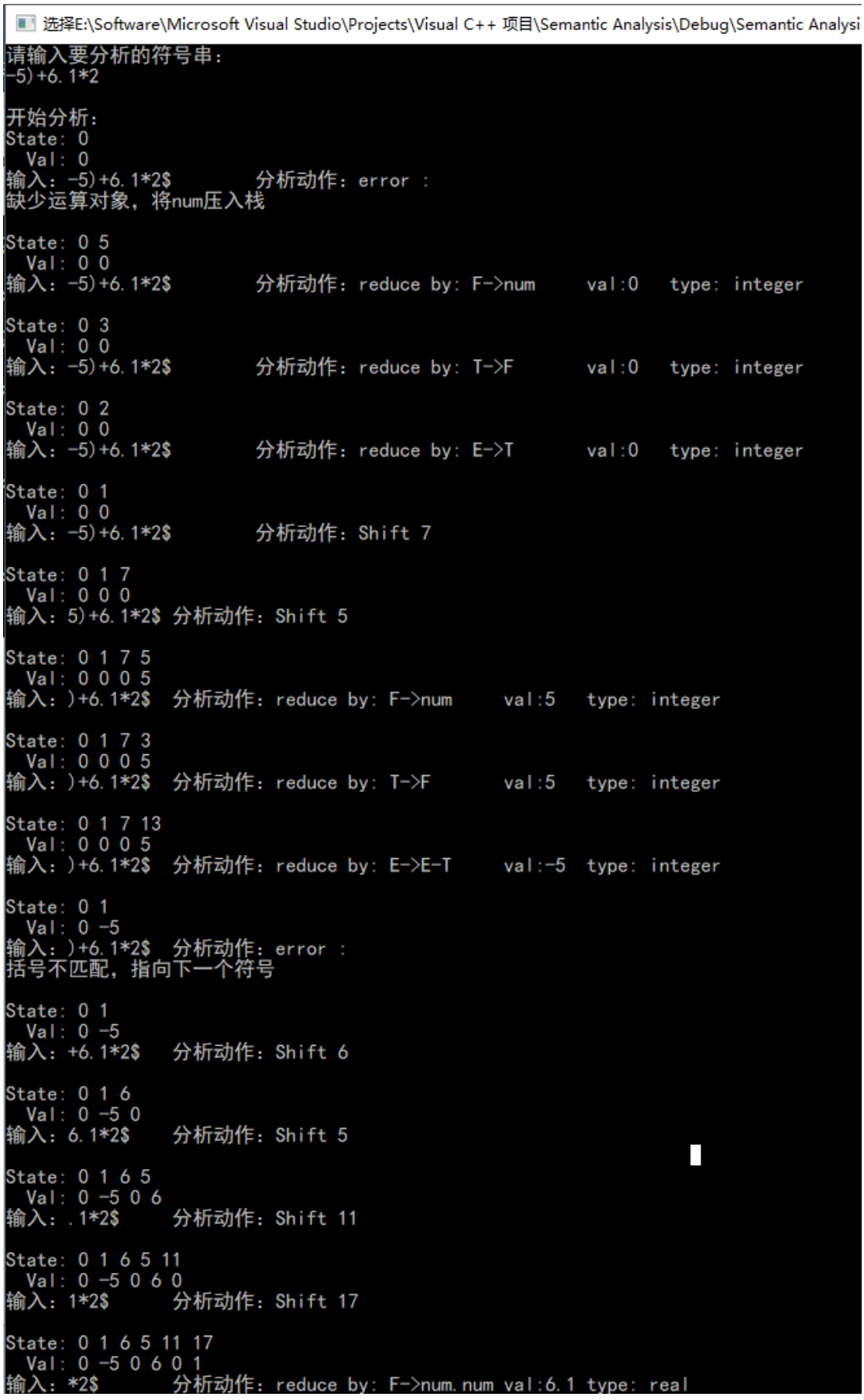

E:\Software\Microsoft Visual Studio\Projects\Visual C++ 项目\Semantic Analysis\Debug\Semantic An
请输入要分析的符号串：
2+3*4
开始分析：
State: 0
Val: 0
输入：2+3*4$ 分析动作：Shift 5
State: 0 5
Val: 0 2
输入：+3*4$ 分析动作：reduce by: F->num val:2 type: integer
State: 0 3
Val: 0 2
输入：+3*4$ 分析动作：reduce by: T->F val:2 type: integer
State: 0 2
Val: 0 2
输入：+3*4$ 分析动作：reduce by: E->T val:2 type: integer
State: 0 1
Val: 0 2
输入：+3*4$ 分析动作：Shift 6
State: 0 1 6
Val: 0 2 0
输入：3*4$ 分析动作：Shift 5
State: 0 1 6 5
Val: 0 2 0 3
输入：*4$ 分析动作：reduce by: F->num val:3 type: integer
State: 0 1 6 3
Val: 0 2 0 3
输入：*4$ 分析动作：reduce by: T->F val:3 type: integer
State: 0 1 6 12
Val: 0 2 0 3
输入：*4$ 分析动作：Shift 8
State: 0 1 6 12 8
Val: 0 2 0 3 0
输入：4$ 分析动作：Shift 5
State: 0 1 6 12 8 5
Val: 0 2 0 3 0 4
输入：$ 分析动作：reduce by: F->num val:4 type: integer
State: 0 1 6 12 8 14
Val: 0 2 0 3 0 4
输入：$ 分析动作：reduce by: T->T*F val:12 type: integer
State: 0 1 6 12
Val: 0 2 0 12
输入：$ 分析动作：reduce by: E->E+T val:14 type: integer
State: 0 1
Val: 0 14
输入：$ 分析动作：ACC 表达式的值：14 表达式的类型：integer
请按任意键继续. . .

输入 5.1+（2+6）/5，无报错


E:\Software\Microsoft Visual Studio\Projects\Visual C++ 项目\Semantic Analysis\Debug\Semant
请输入要分析的符号串：
5.1+(2+6)/5
开始分析：
State: 0
Val: 0
输入：5.1+(2+6)/5$ 分析动作：Shift 5
State: 0 5
Val: 0 5
输入：.1+(2+6)/5$ 分析动作：Shift 11
State: 0 5 11
Val: 0 5 0
输入：1+(2+6)/5$ 分析动作：Shift 17
State: 0 5 11 17
Val: 0 5 0 1
输入：+(2+6)/5$ 分析动作：reduce by: F->num.num val:5.1 type: real
State: 0 3
Val: 0 5.1
输入：+(2+6)/5$ 分析动作：reduce by: T->F val:5.1 type: real
State: 0 2
Val: 0 5.1
输入：+(2+6)/5$ 分析动作：reduce by: E->T val:5.1 type: real
State: 0 1
Val: 0 5.1
输入：+(2+6)/5$ 分析动作：Shift 6
State: 0 1 6
Val: 0 5.1 0
输入：(2+6)/5$ 分析动作：Shift 4
State: 0 1 6 4
Val: 0 5.1 0 0
输入：2+6)/5$ 分析动作：Shift 5
State: 0 1 6 4 5
Val: 0 5.1 0 0 2
输入：+6)/5$ 分析动作：reduce by: F->num val:2 type: integer
State: 0 1 6 4 3
Val: 0 5.1 0 0 2
输入：+6)/5$ 分析动作：reduce by: T->F val:2 type: integer
State: 0 1 6 4 2
Val: 0 5.1 0 0 2
输入：+6)/5$ 分析动作：reduce by: E->T val:2 type: integer
State: 0 1 6 4 10
Val: 0 5.1 0 0 2
输入：+6)/5$ 分析动作：Shift 6
State: 0 1 6 4 10 6
Val: 0 5.1 0 0 2 0
输入：6)/5$ 分析动作：Shift 5
State: 0 1 6 4 10 6 5
Val: 0 5.1 0 0 2 0 6
输入：)/5$ 分析动作：reduce by: F->num val:6 type: integer
State: 0 1 6 4 10 6 3

```
  Val: 0 5.1 0 0 2 0 6
输入：)/5$       分析动作：reduce by: T->F       val:6   type: integer

State: 0 1 6 4 10 6 12
  Val: 0 5.1 0 0 2 0 6
输入：)/5$       分析动作：reduce by: E->E+T     val:8   type: integer

State: 0 1 6 4 10
  Val: 0 5.1 0 0 8
输入：)/5$       分析动作：Shift 16

State: 0 1 6 4 10 16
  Val: 0 5.1 0 0 8 0
输入：/5$        分析动作：reduce by: F->(E)     val:8   type: integer

State: 0 1 6 3
  Val: 0 5.1 0 8
输入：/5$        分析动作：reduce by: T->F       val:8   type: integer

State: 0 1 6 12
  Val: 0 5.1 0 8
输入：/5$        分析动作：Shift 9

State: 0 1 6 12 9
  Val: 0 5.1 0 8 0
输入：5$         分析动作：Shift 5

State: 0 1 6 12 9 5
  Val: 0 5.1 0 8 0 5
输入：$ 分析动作：reduce by: F->num     val:5   type: integer

State: 0 1 6 12 9 15
  Val: 0 5.1 0 8 0 5
输入：$ 分析动作：reduce by: T->T/F     val:1.6 type: real

State: 0 1 6 12
  Val: 0 5.1 0 1.6
输入：$ 分析动作：reduce by: E->E+T     val:6.7 type: real

State: 0 1
  Val: 0 6.7
输入：$ 分析动作：ACC   表达式的值：6.7 表达式的类型：real
请按任意键继续. . .
```

输入-5）+6.1*2，报出错误信息


选择E:\Software\Microsoft Visual Studio\Projects\Visual C++ 项目\Semantic Analysis\Debug\Semantic Analysi
请输入要分析的符号串：
-5)+6.1*2
开始分析：
State: 0
Val: 0
输入：-5)+6.1*2$ 分析动作：error ：
缺少运算对象，将num压入栈
State: 0 5
Val: 0 0
输入：-5)+6.1*2$ 分析动作：reduce by: F->num val:0 type: integer
State: 0 3
Val: 0 0
输入：-5)+6.1*2$ 分析动作：reduce by: T->F val:0 type: integer
State: 0 2
Val: 0 0
输入：-5)+6.1*2$ 分析动作：reduce by: E->T val:0 type: integer
State: 0 1
Val: 0 0
输入：-5)+6.1*2$ 分析动作：Shift 7
State: 0 1 7
Val: 0 0 0
输入：5)+6.1*2$ 分析动作：Shift 5
State: 0 1 7 5
Val: 0 0 0 5
输入：)+6.1*2$ 分析动作：reduce by: F->num val:5 type: integer
State: 0 1 7 3
Val: 0 0 0 5
输入：)+6.1*2$ 分析动作：reduce by: T->F val:5 type: integer
State: 0 1 7 13
Val: 0 0 0 5
输入：)+6.1*2$ 分析动作：reduce by: E->E-T val:-5 type: integer
State: 0 1
Val: 0 -5
输入：)+6.1*2$ 分析动作：error ：
括号不匹配，指向下一个符号
State: 0 1
Val: 0 -5
输入：+6.1*2$ 分析动作：Shift 6
State: 0 1 6
Val: 0 -5 0
输入：6.1*2$ 分析动作：Shift 5
State: 0 1 6 5
Val: 0 -5 0 6
输入：.1*2$ 分析动作：Shift 11
State: 0 1 6 5 11
Val: 0 -5 0 6 0
输入：1*2$ 分析动作：Shift 17
State: 0 1 6 5 11 17
Val: 0 -5 0 6 0 1
输入：*2$ 分析动作：reduce by: F->num.num val:6.1 type: real

```
State: 0 1 6 3
  Val: 0 -5 0 6.1
输入：*2$       分析动作：reduce by: T->F       val:6.1 type: real

State: 0 1 6 12
  Val: 0 -5 0 6.1
输入：*2$       分析动作：Shift 8

State: 0 1 6 12 8
  Val: 0 -5 0 6.1 0
输入：2$        分析动作：Shift 5

State: 0 1 6 12 8 5
  Val: 0 -5 0 6.1 0 2
输入：$ 分析动作：reduce by: F->num     val:2   type: integer

State: 0 1 6 12 8 14
  Val: 0 -5 0 6.1 0 2
输入：$ 分析动作：reduce by: T->T*F     val:12.2        type: real

State: 0 1 6 12
  Val: 0 -5 0 12.2
输入：$ 分析动作：reduce by: E->E+T     val:7.2 type: real

State: 0 1
  Val: 0 7.2
输入：$ 分析动作：ACC   表达式的值：7.2 表达式的类型：real
请按任意键继续. . .
```