编译原理与技术

语义分析实验

目录

2017-12-26

[裴子祥 计科七班 学号 2015211921]

[指导老师：刘辰]

一．实验内容与要求

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。要求所分析算数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 | (E) | num

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. 要求写出满足要求的语法制导定义或翻译方案。
2. 编写语法分析和翻译程序，实现对表达式的类型进行检查和求值，并输出：
   i. 分析过程中所有产生式
   ii. 识别出的表达式的类型
   iii. 识别出表达式的值

实验环境：

MICSOFT WINDOWS 10

Visual Studio 2015

C/C++作为实现语言

二． 程序设计与实现

1． 主要步骤

(1) 拓广文法
(2) 构造文法的 FIRST 集
(3) 构造文法的 FOLLOW 集
(4) 语法制导定义
(5) 设计求值代码
(6) 识别所有活前缀的 DFA
(7) 构造 SLR 分析表
(8) 构造 SLR 分析程序扩展至语义分析程序

2． 程序实现

（1） 拓广文法

0) S-> E
1) E-> E+T
2) E-> E-T
3) E-> T
4) T->T*F
5) T-> T/F

6) T-> F
7) F-> (E)
8) F-> num
9) F-> num.num

(2) FIRST 集与 FOLLOW 集

	FIRST	FOLLOW
S	(n	$
E	(n	$ + -)
T	(n	$ * / + -)
F	(n	$ * /) + -

(3) 语法制导定义

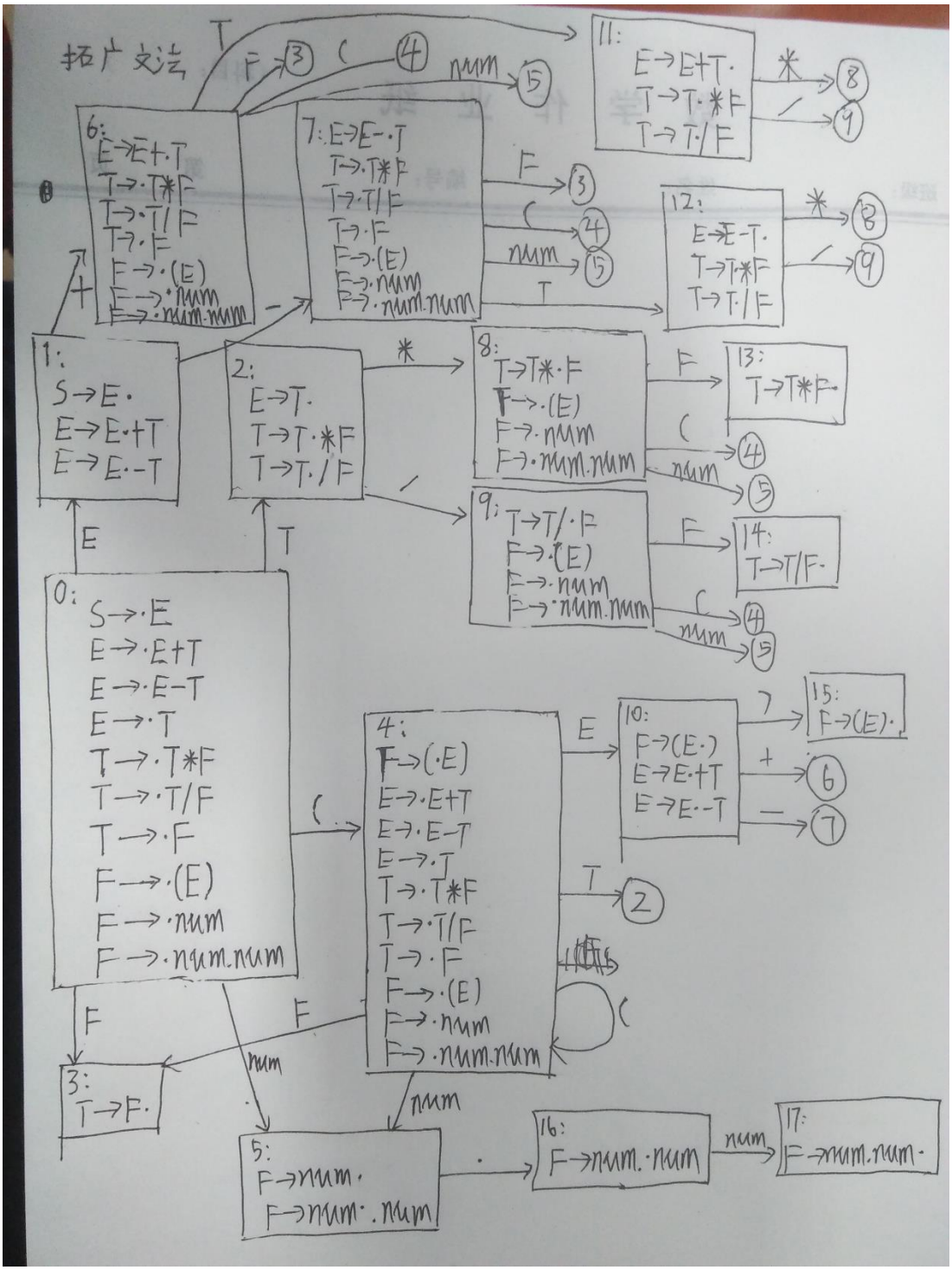
0	S->E	Print(E.val, E.type)
1	E->E1+T	E.val = E1.val+T.val if(T.type ==integer && E1.type == integer) E.type = integer else E.type = real
2	E->E1-T	E.val = E1.val-T.val if(T.type ==integer && E1.type == integer) E.type = integer else E.type = real
3	E->T	E.val = T.val E.type = T.type
4	T->T1*F	T.val=T1.val*F.val if(T1.type ==integer && F.type == integer) T.type = integer else T.type = real
5	T->T1/F	T.val=T1.val/F.val if(T1.type ==integer && F.type == integer && T.val%F.val == 0) T.type = integer else T.type = real
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.lexval F.type = integer
9	F->num.num	F.val=num.num.lexval F.type = real

(4) 求值代码设计

0	S->E	Print(val[top])
1	E->E_1+T	Val[newtop]=val[top-2]+val[top]
2	E->E_1-T	Val[newtop]=val[top-2]-val[top]
3	E->T	
4	T->T_1*F	Val[newtop]=val[top-2]*val[top]

5	T->T_1/F	Val[newtop]=val[top-2]/val[top]
6	T->F	
7	F->(E)	Val[newtop]=val[top-1]
8	F->num	
9	F->num.num	

（5） 构造拓广文法的项目集规范族，识别所有活前缀的 DFA



（6） 构造 SLR 分析表

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	S16			
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				
16	S17											
17		R9	R9	R9	R9		R9	R9				

LR分析的错误恢复策略:

短语级恢复：

1. 对剩余输入作局部纠正，用可以使分析器继续分析的串来代替剩余输入的前缀
2. 尽量避免从分析栈中弹出与非终结符有关的状态，因为归约出的非终结符都是分析成功的

添加错误恢复后SLR分析表：

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

14	R5	R5	R5	R5	R5	R5	R5	R5	E3			
15	R7	R7	R7	R7	R7	R7	R7	R7	E3			
16	S17	E1	E1	E1	E1	E3	E2	E1	E3			
17	R9	R9	R9	R9	R9	R9	R9	R9	R9			

错误种类：

E1:缺少运算对象，状态3入栈，局部纠错

E2:括号不匹配，跳过该输入符号

E3:缺少运算符号，状态4入栈，局部纠错

E4:缺少右括号，状态9入栈，补充右括号，局部纠错

(7) 全局变量

```
const int columnNum = 12;                                          //分析表列数
const int Nsize = 4;                                               //非终结符个数
const int Gsize = 10;                                              //产生式个数
const int stateNum = 18;                                           //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
stack<VAL> valStack;
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
```

(8) 函数与过程

```
void Init();              //Fisrt、Follow、拓广文法初始化

int map(char x);          //将字符映射到列

void createTable();           //建立SLR分析表

void printstack();            //打印状态栈

void printval();          //打印数据栈

double stringToDouble(string str); //把字符串转换为doulbe
```

```
string doubleToString(double d);    //把double转化成string

void error(char kind = 'x');        //错误处理

void SemanticAnalyse(string w);     //SLR分析改进延伸，语义分析程序

void printout();                    //打印输出
```

（9） 输入

待分析表达式字符串

如*180*62*+3

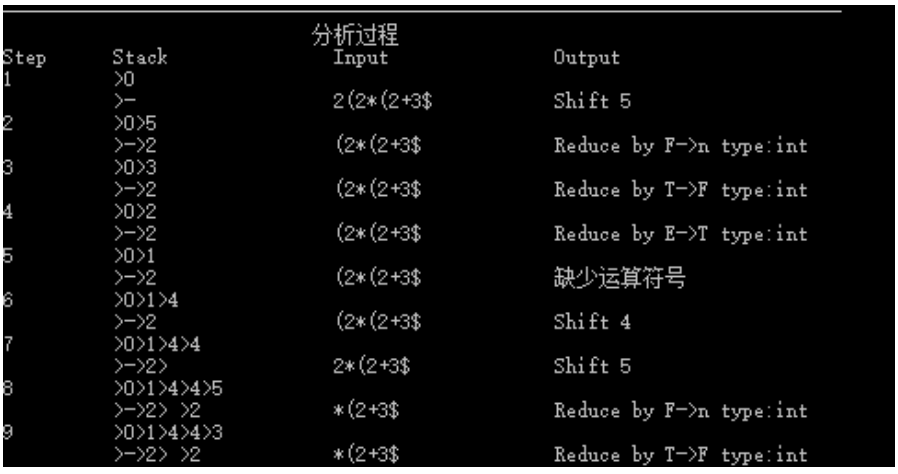
10+(1+2)*3+(22/8)

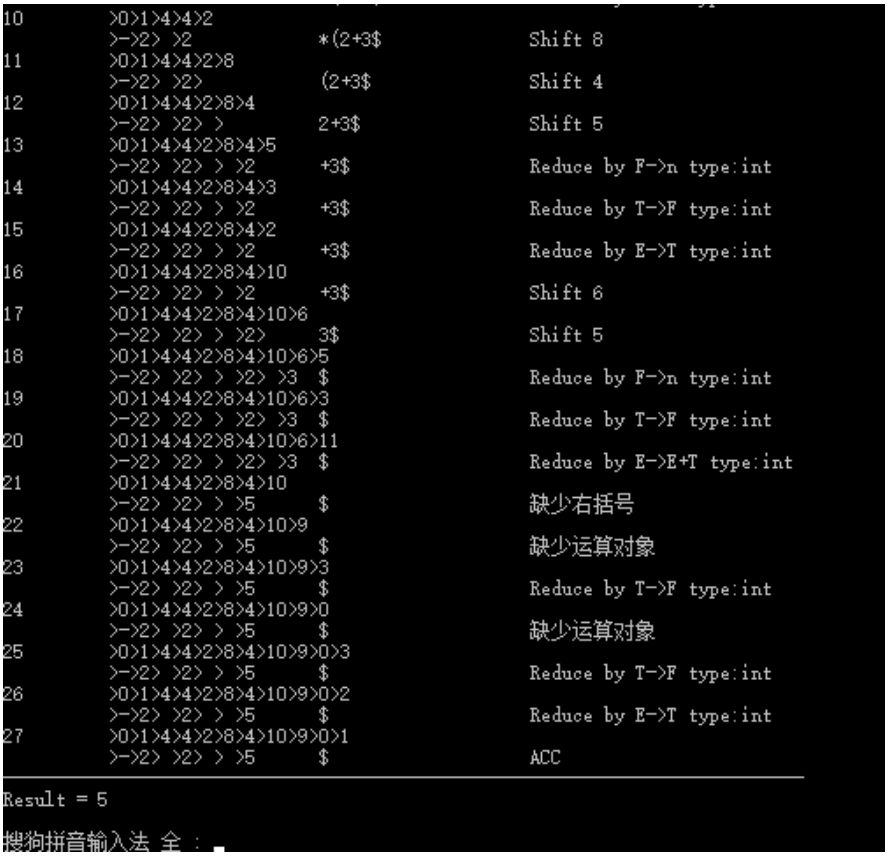
（10） 输出文件形式部分举例

对符号串的语义分析过程

3． 样例结果

（1） 示例输入 1，特意错误出入 2(2*(2+3





结果分析

通过示例 1，能够看出程序能够正确地做出错误处理判断，并通过局部纠错

错误种类：

E1:缺少运算对象，状态3入栈，局部纠错

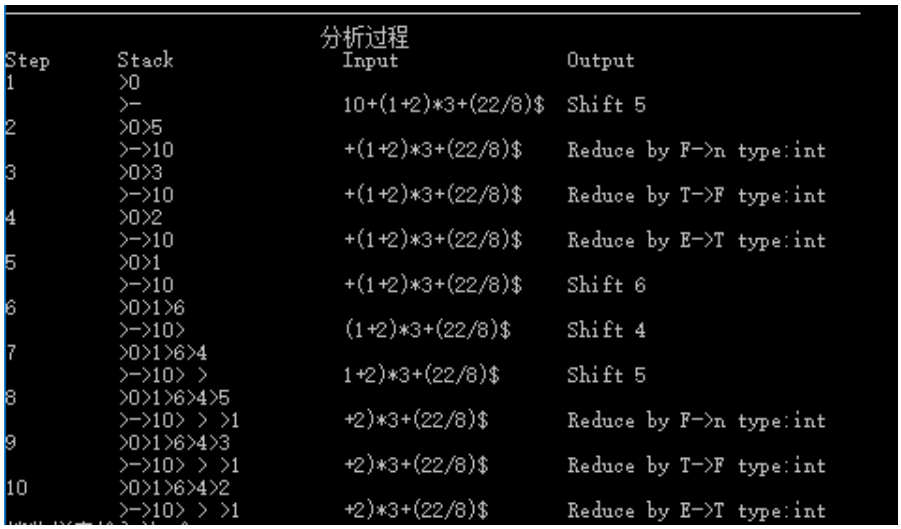
E2:括号不匹配，跳过该输入符号

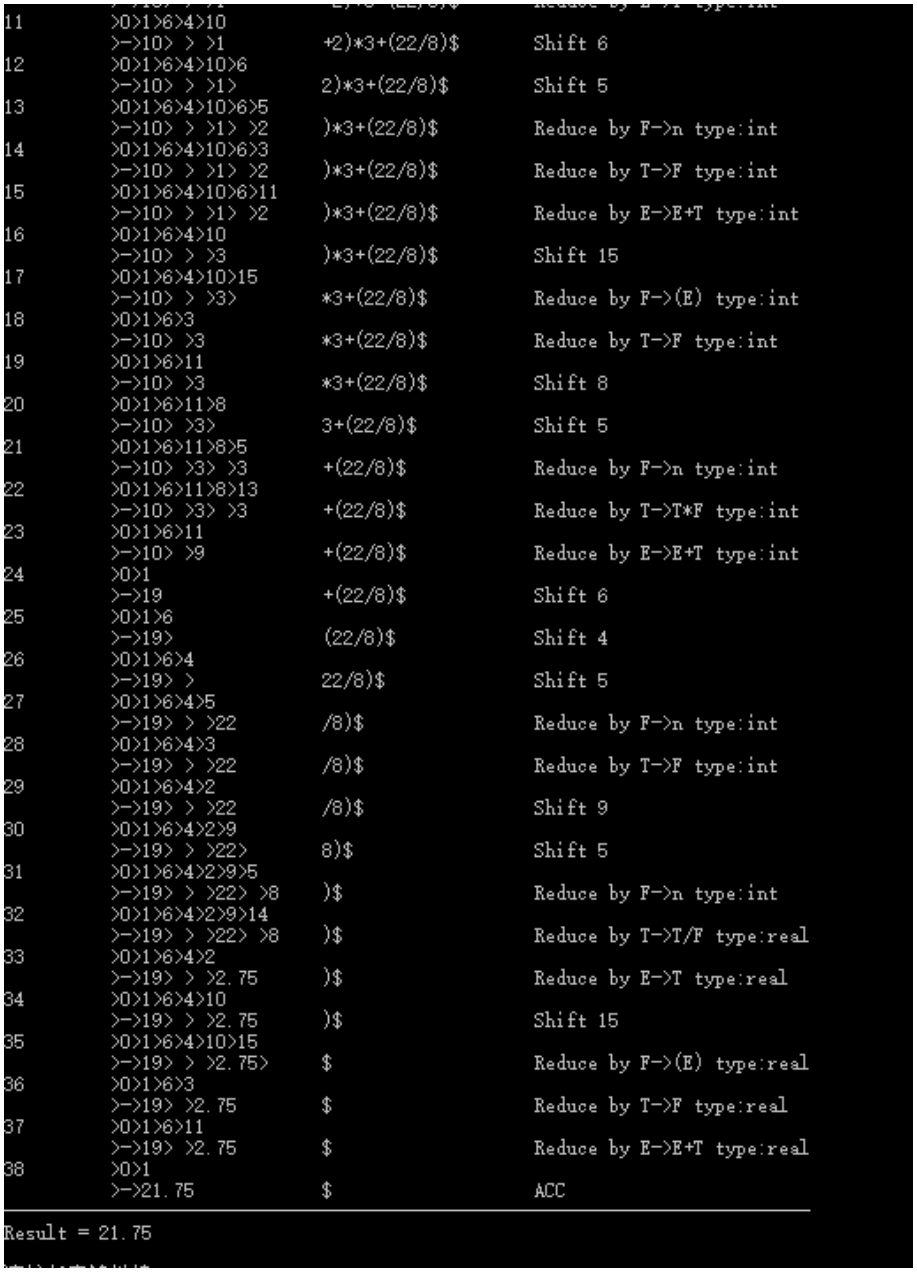
E3:缺少运算符号，状态4入栈，局部纠错

E4:缺少右括号，状态9入栈，补充右括号，局部纠错

最终结果计算为2+3

（2） 示例输入 2，10+(1+2)*3+(22/8)





```
11      >O>1>6>4>10
        >->10> > >1         +2)*3+(22/8)$       Shift 6
12      >O>1>6>4>10>6
        >->10> > >1>        2)*3+(22/8)$        Shift 5
13      >O>1>6>4>10>6>5
        >->10> > >1> >2     )*3+(22/8)$         Reduce by F->n type:int
14      >O>1>6>4>10>6>3
        >->10> > >1> >2     )*3+(22/8)$         Reduce by T->F type:int
15      >O>1>6>4>10>6>11
        >->10> > >1> >2     )*3+(22/8)$         Reduce by E->E+T type:int
16      >O>1>6>4>10
        >->10> > >3         )*3+(22/8)$         Shift 15
17      >O>1>6>4>10>15
        >->10> > >3>        *3+(22/8)$          Reduce by F->(E) type:int
18      >O>1>6>3
        >->10> >3           *3+(22/8)$          Reduce by T->F type:int
19      >O>1>6>11
        >->10> >3           *3+(22/8)$          Shift 8
20      >O>1>6>11>8
        >->10> >3>          3+(22/8)$           Shift 5
21      >O>1>6>11>8>5
        >->10> >3> >3       +(22/8)$            Reduce by F->n type:int
22      >O>1>6>11>8>13
        >->10> >3> >3       +(22/8)$            Reduce by T->T*F type:int
23      >O>1>6>11
        >->10> >9           +(22/8)$            Reduce by E->E+T type:int
24      >O>1
        >->19               +(22/8)$            Shift 6
25      >O>1>6
        >->19>              (22/8)$             Shift 4
26      >O>1>6>4
        >->19> >            22/8)$              Shift 5
27      >O>1>6>4>5
        >->19> > >22        /8)$                Reduce by F->n type:int
28      >O>1>6>4>3
        >->19> > >22        /8)$                Reduce by T->F type:int
29      >O>1>6>4>2
        >->19> > >22        /8)$                Shift 9
30      >O>1>6>4>2>9
        >->19> > >22>       8)$                 Shift 5
31      >O>1>6>4>2>9>5
        >->19> > >22> >8    )$                  Reduce by F->n type:int
32      >O>1>6>4>2>9>14
        >->19> > >22> >8    )$                  Reduce by T->T/F type:real
33      >O>1>6>4>2
        >->19> > >2.75      )$                  Reduce by E->T type:real
34      >O>1>6>4>10
        >->19> > >2.75      )$                  Shift 15
35      >O>1>6>4>10>15
        >->19> > >2.75>     $                   Reduce by F->(E) type:real
36      >O>1>6>3
        >->19> >2.75        $                   Reduce by T->F type:real
37      >O>1>6>11
        >->19> >2.75        $                   Reduce by E->E+T type:real
38      >O>1
        >->21.75            $                   ACC
-------------------------------------------------------------------------------
Result = 21.75
```

结果分析

如运行结果所示，较完整的正确输入下，能够实现正确的语义过程 !!

最终结果 10+(1+2)*3+(22/8)=21.75

4． 实验总结

这是编译原理与技术第三次程序设计实验，程序设计极大地在第二次 SLR 语法分析实验的基础上扩展，熟悉掌握了文法的语法制导定义。实现了语义程序的要求，并能根据输入得到正确的结果，实现对算术表达式的类型检查和求值，也能实现错误处理，局部纠错，是一次很好的编译程序编写体验，在不断的 debug 中，对语法制导、语义分析的基本内容也有就好的掌握。

5． 源码附件

```
/**
*SemanticAnalyse.cpp 语义分析程序
*作者:裴子祥
*时间:2017.12.26
*/

#include <iostream>
#include <string>
#include <cstdio>
#include <cstdlib>
#include <stack>
#include <vector>
#include <iomanip>
#include <sstream>

using namespace std;

typedef struct val {
    string value;
    int type;
}VAL;

const int columnNum = 12;                                          //分析表列数
const int Nsize = 4;                                               //非终结符个数
const int Gsize = 10;                                              //产生式个数
const int stateNum = 18;                                           //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
stack<VAL> valStack;
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();              //Fisrt、Follow、拓广文法初始化
```

```
int map(char x);           //将字符映射到列

void createTable();        //建立 SLR 分析表

void printstack();         //打印状态栈

void printval();           //打印数据栈

double stringToDouble(string str);  //把字符串转换为 doulbe

string doubleToString(double d);    //把 double 转化成 string

void error(char kind = 'x');        //错误处理

void SemanticAnalyse(string w);     //SLR 分析改进延伸，语义分析程序

void printout();                   //打印输出

int main()
{

    string w;
    cout << "输入待分析算术表达式(形如 7*2+1):";
    cin >> w;

    Init();
    createTable();
    printout();
    SemanticAnalyse(w);

    system("pause");
    return 0;
}

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";
    R[9] = "F->n.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 '+':
```

```
        return 1;
    case '-':
        return 2;
    case '*':
        return 3;
    case '/':
        return 4;
    case '(':
        return 5;
    case ')':
        return 6;
    case '$':
        return 7;
    case '.':
        return 8;
    case 'E':
        return 9;
    case 'T':
        return 10;
    case 'F':
        return 11;
    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] = "E3";
    AnalyseTable[0][9] = "1";
    AnalyseTable[0][10] = "2";
    AnalyseTable[0][11] = "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[1][8] = "E3";

    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[2][8] = "E3";

    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[3][8] = "E3";

    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] = "E3";
    AnalyseTable[4][9] = "10";
    AnalyseTable[4][10] = "2";
    AnalyseTable[4][11] = "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[5][8] = "S16";

    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][8] = "E3";
    AnalyseTable[6][10] = "11";
    AnalyseTable[6][11] = "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][8] = "E3";
    AnalyseTable[7][10] = "12";
    AnalyseTable[7][11] = "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][8] = "E3";
    AnalyseTable[8][11] = "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][8] = "E3";
    AnalyseTable[9][11] = "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[10][8] = "E3";

    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[11][8] = "E3";

    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[12][8] = "E3";


    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[13][8] = "E3";

    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[14][8] = "E3";

    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";
    AnalyseTable[15][8] = "E3";

    AnalyseTable[16][0] = "S17";
    AnalyseTable[16][1] = "E1";
    AnalyseTable[16][2] = "E1";
    AnalyseTable[16][3] = "E1";
    AnalyseTable[16][4] = "E1";
    AnalyseTable[16][5] = "E3";
    AnalyseTable[16][6] = "E2";
    AnalyseTable[16][7] = "E1";
    AnalyseTable[16][8] = "E3";

    AnalyseTable[17][0] = "R9";
    AnalyseTable[17][1] = "R9";
    AnalyseTable[17][2] = "R9";
    AnalyseTable[17][3] = "R9";
    AnalyseTable[17][4] = "R9";
    AnalyseTable[17][5] = "R9";
    AnalyseTable[17][6] = "R9";
    AnalyseTable[17][7] = "R9";
    AnalyseTable[17][8] = "R9";
}

//打印状态栈
```

```
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_s(t, xoxo, 10);
        output = output + ">" + xoxo;
        stateStack.push(t);
        temp.pop();
    }
    cout << output;
}

//打印数据栈
void printval()
{
    string output = "";
    stack<VAL> temp;
    while (!valStack.empty()) {
        VAL t = valStack.top();
        temp.push(t);
        valStack.pop();
    }
    while (!temp.empty()) {
        VAL t = temp.top();
        output = output + ">" + t.value;
        valStack.push(t);
        temp.pop();
    }
    cout << output;
}

//把字符串转换为 doulbe
double stringToDouble(string str)
{
    int len = str.length();
    double outcome = 0;
    double nn = 0;
```

```
    int i = 0;
    for (; i < len && str[i] >= '0' && str[i] <= '9'; i++) {
        outcome = outcome * 10 + str[i] - '0';
    }
    if (str[i] == '.') {
        for (int j = len - 1; j > i; j--) {
            nn = nn*0.1 + str[j] - '0';
        }
    }
    nn = nn*0.1;
    outcome = outcome + nn;
    return outcome;
}

//把 double 转化成 string
string doubleToString(double d)
{
    ostringstream oss;
    oss << d;
    string str = oss.str();
    return str;
}

//错误处理
void error(char kind)
{
    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;
```

```
    }
}

//SLR 分析改进延伸，语义分析程序
void SemanticAnalyse(string w)
{
    int X;
    char a;
    int inta;
    double n = 0;
    int thetype;
    string ss = "";
    ip = 0;
    int step = 0;
    stateStack.push(0);
    VAL vl;
    vl.type = 0;
    vl.value = "-";
    valStack.push(vl);
    inputBuffer = w + "$";

    cout << "                          分析过程" << endl;
    cout << left << setw(10) << "Step" << left << setw(20) << "Stack" << left <<
setw(20) << "Input" << left << setw(20) << "Output" << endl;

    do {
        X = stateStack.top();
        a = inputBuffer[ip];
        ss = "";
        step++;

        cout << left << setw(10) << step;
        cout << left << setw(20);
        printstack();
        cout << endl;
        cout << left << setw(10) << " ";
        cout << left << setw(20);
        printval();
        cout << left << setw(20) << inputBuffer.substr(ip);

        if (a >= '0' && a <= '9')
        {
            inta = 0;
            while (a >= '0' && a <= '9')
```

```
            {
                ss = ss + a;
                ip++;
                a = inputBuffer[ip];
            }
            if (a == '.')
            {
                thetype = 2;
                ss = ss + a;
                ip++;
                a = inputBuffer[ip];
                while (a >= '0' && a <= '9')
                {
                    ss = ss + a;
                    ip++;
                    a = inputBuffer[ip];
                }
            }
            else
            {
                thetype = 1;
            }

            ip--;
        }
        else
        {
            inta = map(a);
        }

        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);
            if (inta == 0)
```

```
            {
                vl.type = thetype;
                vl.value = ss;
                valStack.push(vl);
            }
            else
            {
                vl.type = 0;
                vl.value = " ";
                valStack.push(vl);
            }

            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);

            //输出 A->β
            cout << "Reduce by " << R[num];

            string t = "";                  //转移状态号
            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);         //状态入栈
            VAL temp1, temp2, temp3;
            double d1, d2;
            switch (num) {
            case 1:
                temp1 = valStack.top();
                valStack.pop();
                valStack.pop();
                temp2 = valStack.top();
```

```
            valStack.pop();
            d1 = stringToDouble(temp1.value);
            d2 = stringToDouble(temp2.value);
            temp3.value = doubleToString(d2 + d1);
            if (temp1.type == 1 && temp2.type == 1) {
                temp3.type = 1;
                valStack.push(temp3);
            }
            else {
                temp3.type = 2;
                valStack.push(temp3);
            }

            break;
        case 2:
            temp1 = valStack.top();
            valStack.pop();
            valStack.pop();
            temp2 = valStack.top();
            valStack.pop();
            d1 = stringToDouble(temp1.value);
            d2 = stringToDouble(temp2.value);
            temp3.value = doubleToString(d2 - d1);
            if (temp1.type == 1 && temp2.type == 1) {
                temp3.type = 1;
                valStack.push(temp3);
            }
            else {
                temp3.type = 2;
                valStack.push(temp3);
            }
            break;
        case 4:
            temp1 = valStack.top();
            valStack.pop();
            valStack.pop();
            temp2 = valStack.top();
            valStack.pop();
            d1 = stringToDouble(temp1.value);
            d2 = stringToDouble(temp2.value);
            temp3.value = doubleToString(d2*d1);
            if (temp1.type == 1 && temp2.type == 1) {
                temp3.type = 1;
                valStack.push(temp3);
            }
```

```
            else {
                temp3.type = 2;
                valStack.push(temp3);
            }
            break;
        case 5:
            temp1 = valStack.top();
            valStack.pop();
            valStack.pop();
            temp2 = valStack.top();
            valStack.pop();
            d1 = stringToDouble(temp1.value);
            d2 = stringToDouble(temp2.value);
            temp3.value = doubleToString(d2 / d1);
            if (temp1.type == 1 && temp2.type == 1) {
                if ((int)d2 % (int)d1 == 0) {
                    temp3.type = 1;
                    valStack.push(temp3);
                }
                else {
                    temp3.type = 2;
                    valStack.push(temp3);
                }

            }
            else {
                temp3.type = 2;
                valStack.push(temp3);
            }
            break;
        case 7:
            valStack.pop();
            temp1 = valStack.top();
            valStack.pop();
            valStack.pop();
            valStack.push(temp1);
            break;
        default:
            break;
        }
        temp1 = valStack.top();
        if (temp1.type == 1) {
            cout << " type:int" << endl;
        }
        else if (temp1.type == 2) {
```

```
            cout << " type:real" << endl;
        }

    }
    else if (AnalyseTable[X][inta] == "ACC")
    {//成功接收
        cout << "ACC" << endl;
        cout << "--------------------------------------------------------------------------------" << endl;
        string sss = valStack.top().value;
        cout << "Result = " << sss << endl << endl;
        return;
    }
    else if (AnalyseTable[X][inta][0] == 'E') {
        error(AnalyseTable[X][inta][1]);
    }
    else {
        error();
    }
  } while (1);
}

//打印输出
void printout()
{
    cout << "--------------------------------------------------------------------------------" << endl;
    cout << "                              SLR 分析表" << endl;
    cout << left << setw(6) << " ";

    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;
```

```
}
```