北京邮电大学　计算机学院

《编译原理与技术》实验报告

姓名　　王睿嘉

学号 2015211906

班级 2015211307

语义分析程序的设计与实现

一、 实验内容和环境描述

1. 实验内容

编写语义分析和翻译程序，实现对算数表达式的类型检查和求值，所分析的表达式由如下文法产生：

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

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

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

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识别出的子表达式类型；

③识别出的子表达式值。

2. 实验环境

本次实验在 Win10 下，使用 VS2015 作为 IDE 编写代码，并完成相关调试工作。

二、 程序设计说明

1. 程序功能

1）功能说明

由已知分析表及对应语法制导翻译方案，不仅实现了对目标文法的语义分析过程，还实现了对输入字符串的词法分析过程。

用户输入待分析符号序列，程序输出分析过程中所调用的产生式，并输出所识别出的子表达式的值和类型，最终于 Accept 时，输出整个表达式的值和类型。若输入的符号序列不符合文法规则，则输出已完成的分析结果，并打印错误信息。

分析表构造过程及语义制导翻译方案如下所示。

2）拓广文法

○ E’->E

① E->E+T

② E->E-T

③ E->T

④ T->T*F

⑤ T->T/F

⑥ T->F

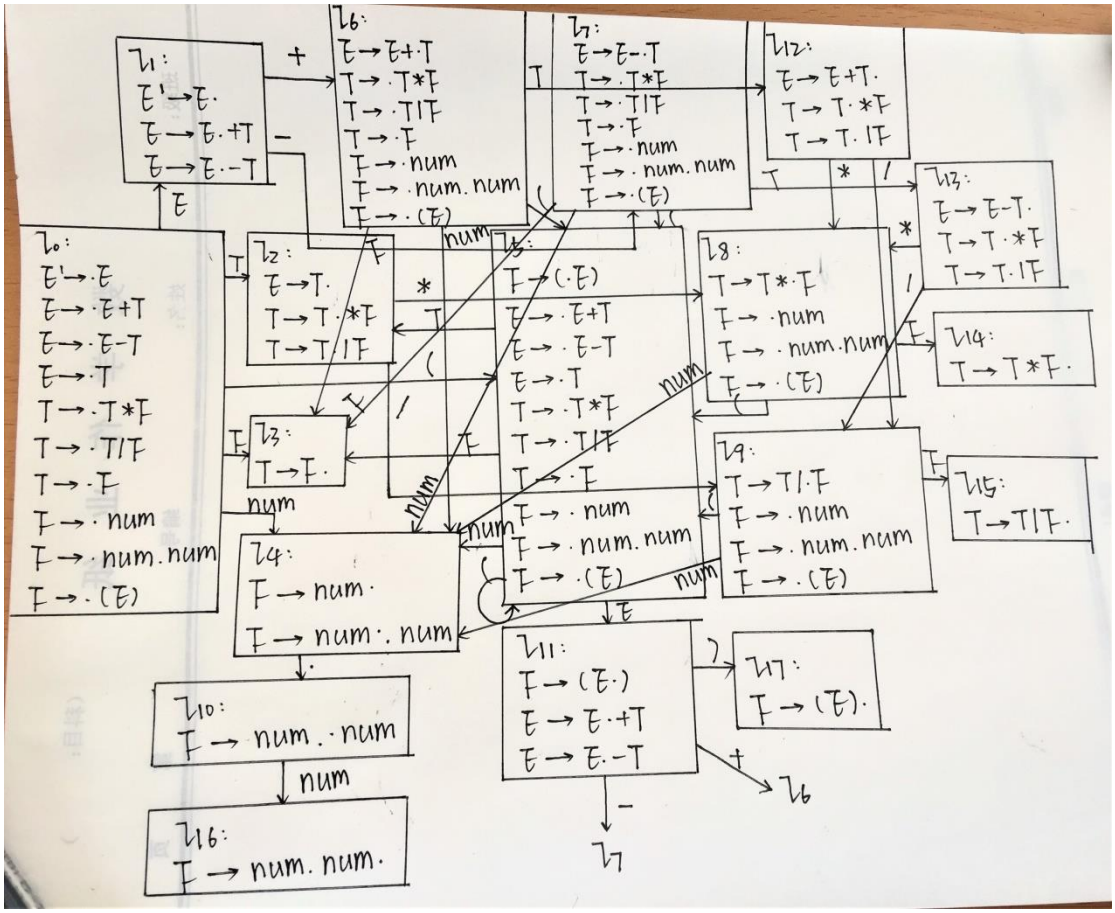
⑦ F->num

⑧ F->num.num

⑨ F->(E)

3）识别所有活前缀的 DFA

如下图所示：



4）FIRST 集和 FOLLOW 集

如下表所示：

	E	T	F
FIRST	num,(	num,(	num,(
FOLLOW	$,+,-,)	$,+,-,),*,/	$,+,-,),*,/

5）LR 分析表

如下表所示：

	Action									Goto		
状态	+	-	*	/	num	.	(	)	$	E	T	F
0					S4		S5			1	2	3
1	S6	S7							ACC			
2	R3	R3	S8	S9				R3	R3			
3	R6	R6	R6	R6				R6	R6			
4	R7	R7	R7	R7		S10		R7	R7			
5					S4		S5			11	2	3
6					S4		S5				12	3
7					S4		S5				13	3
8					S4		S5					14
9					S4		S5					15
10					S16							
11	S6	S7						S17				
12	R1	R1	S8	S9				R1	R1			
13	R2	R2	S8	S9				R2	R2			
14	R4	R4	R4	R4				R4	R4			
15	R5	R5	R5	R5				R5	R5			
16	R8	R8	R8	R8				R8	R8			
17	R9	R9	R9	R9				R9	R9			

6）语法制导翻译方案

引入：

① 综合属性 value 和 type，分别用于记录非终结符的值和类型；

② 虚拟综合属性 print，用于打印结果。

翻译方案如下：

○ E’->E{print(E.value);print(E.type);}

① E->E_1+T{E.value=E_1.value+T.value;if(E_1.type==real||T.type==real)E.type=real;else E.type=integer;}

② E->E_1-T{E.value=E_1.value-T.value;if(E_1.type==real||T.type==real)E.type=real;else E.type=integer;}

③ E->T{E.value=T.value;E.type=T.type;}

④ T->T_1*F{T.value=T_1.value*F.value;if(T_1.type==real||F.type==real)T.type=real;else T.type=integer;}

⑤ T->T_1/F{T.value=T_1.value/F.value;if(T_1.type==real||F.type==real)T.type=real;else T.type=integer;}

⑥ T->F{T.value=F.value;T.type=F.type;}

⑦ F->num{F.value=num.lexval;F.type=integer;}

⑧ F->num.num_1 {F.value=num.lexval+num_1.lexval/10;F.type=real;}

⑨ F->(E){F,value=E.value;F.type=E.type;}

2. 输入与输出

1) 输入的定义

用户只需要输入待分析符号序列。

2) 输出的定义

程序给出两部分输出：

该文法的分析表（action&goto）；

对应于输入符号序列的分析结果（各步骤中所调用的产生式，及所识别出的子表达式的值和类型）。

3. 程序实现说明

1) 函数说明

```
char isDigit(char);                    //判断是否为数字
int map(char);                         //将文法符号映射为下标
void initial();                        //初始化，构建分析表
```

在初始化时，将初始状态S0入栈，从而完成语义分析的预备工作。

```
void printTable();                     //输出分析表
void analyze(string);                  //语义分析及翻译程序
```

语义分析及翻译程序的关键在于如何保存文法符号的综合属性值及保存属性值的数据结构如何与分析栈相联系。

前者的解决方案为：修改分析栈，增加两个域分别用来存储 value 和 type 两个综合属性值。

后者的解决方案为：为每个语义规则编写一段代码，用于计算属性值。每当进行归约时，由栈中正在归约的产生式右部符号的属性值计算其左部符号的综合属性值，即：

（1）对于终结符号，其综合属性值由词法分析程序产生，当分析程序执行移进操作时，其属性值跟随状态符号一起入栈；

（2）对于每一个产生式 A->XYZ，把属性值的计算与归约动作联系起来。归约前，执行与产生式相关的代码段。归约时，右部符号的相应状态及其属性出栈，左部符号的相应状态及其属性入栈。

2) 数据结构说明

本次实验的难点之一在于数据结构的使用：以何种方式实现栈？以何种方式存储输入符号串？以何种方式存储分析表，并能快速访问表项？

```
vector<char> stack;                    //符号栈
```

其作用为：存放一系列文法符号。

```
vector<int> state;                     //状态栈
```

其作用为：存放一系列状态序列。

```
vector<double> value;                  //属性栈
```

其作用为：存放对应文法符号的value综合属性。

```
vector<string> type;
```

其作用为：存放对应文法符号的 type 综合属性。

```
string table[18][12];                  //分析表（action&goto）
```

利用二维数组存储分析表，实现二元映射，分析时利用[][]进行索引，从而快速获得所需要的表项。

```
string input;                          //输入符号序列
```

三、 运行测试

1. 测试用例 1

输入及运行结果如下：

```
************************************************************************************************
>>                                         分析表                                             <<
************************************************************************************************
```

	+	-	*	/	num	.	(	)	$	E	T	F
0	error	error	error	error	S4	error	S5	error	error	1	2	3
1	S6	S7	error	error	error	error	error	error	ACC	error	error	error
2	RE->T	RE->T	S8	S9	error	error	error	RE->T	RE->T	error	error	error
3	RT->F	RT->F	RT->F	RT->F	error	error	error	RT->F	RT->F	error	error	error
4	RF->num	RF->num	RF->num	RF->num	error	S10	error	RF->num	RF->num	error	error	error
5	error	error	error	error	S4	error	S5	error	error	11	2	3
6	error	error	error	error	S4	error	S5	error	error	error	12	3
7	error	error	error	error	S4	error	S5	error	error	error	13	3
8	error	error	error	error	S4	error	S5	error	error	error	error	14
9	error	error	error	error	S4	error	S5	error	error	error	error	15
10	error	error	error	error	S16	error	error	error	error	error	error	error
11	S6	S7	error	error	error	error	error	S17	error	error	error	error
12	RE->E+T	RE->E+T	S8	S9	error	error	error	RE->E+T	RE->E+T	error	error	error
13	RE->E-T	RE->E-T	S8	S9	error	error	error	RE->E-T	RE->E-T	error	error	error
14	RT->T*F	RT->T*F	RT->T*F	RT->T*F	error	error	error	RT->T*F	RT->T*F	error	error	error
15	RT->T/F	RT->T/F	RT->T/F	RT->T/F	error	error	error	RT->T/F	RT->T/F	error	error	error
16	RF->num.num	RF->num.num	RF->num.num	RF->num.num	error	error	error	RF->num.num	RF->num.num	error	error	error
17	RF->(E)	RF->(E)	RF->(E)	RF->(E)	error	error	error	RF->(E)	RF->(E)	error	error	error

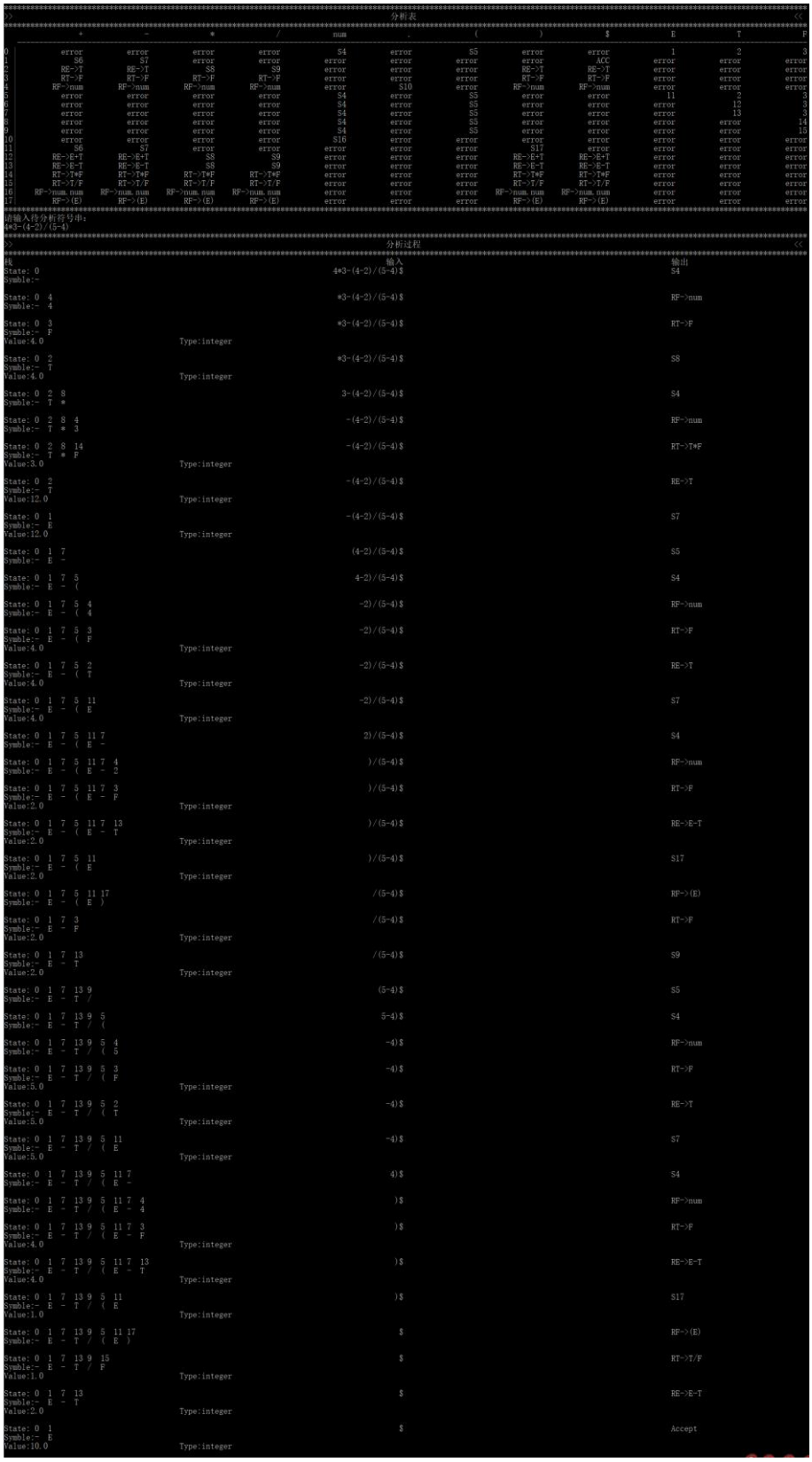
```
************************************************************************************************
请输入待分析符号串:
(6-3)*3-6.0/2
************************************************************************************************
>>                                         分析过程                                           <<
************************************************************************************************
栈                                                  输入                             输出
State: 0                                            (6-3)*3-6.0/2$                   S5
Symble:-

State: 0  5                                         6-3)*3-6.0/2$                    S4
Symble:-  (

State: 0  5  4                                      -3)*3-6.0/2$                     RF->num
Symble:-  (  6

State: 0  5  3                                      -3)*3-6.0/2$                     RT->F
Symble:-  (  F
Value:6.0                   Type:integer

State: 0  5  2                                      -3)*3-6.0/2$                     RE->T
Symble:-  (  T
Value:6.0                   Type:integer

State: 0  5  11                                     -3)*3-6.0/2$                     S7
Symble:-  (  E
Value:6.0                   Type:integer

State: 0  5  11 7                                   3)*3-6.0/2$                      S4
Symble:-  (  E  -

State: 0  5  11 7  4                                )*3-6.0/2$                       RF->num
Symble:-  (  E  -  3

State: 0  5  11 7  3                                )*3-6.0/2$                       RT->F
Symble:-  (  E  -  F
Value:3.0                   Type:integer

State: 0  5  11 7  13                               )*3-6.0/2$                       RE->E-T
Symble:-  (  E  -  T
Value:3.0                   Type:integer

State: 0  5  11                                     )*3-6.0/2$                       S17
Symble:-  (  E
Value:3.0                   Type:integer

State: 0  5  11 17                                  *3-6.0/2$                        RF->(E)
Symble:-  (  E  )

State: 0  3                                         *3-6.0/2$                        RT->F
Symble:-  F
Value:3.0                   Type:integer

State: 0  2                                         *3-6.0/2$                        S8
Symble:-  T
Value:3.0                   Type:integer

State: 0  2  8                                      3-6.0/2$                         S4
Symble:-  T  *

State: 0  2  8  4                                   -6.0/2$                          RF->num
Symble:-  T  *  3

State: 0  2  8  14                                  -6.0/2$                          RT->T*F
Symble:-  T  *  F
Value:3.0                   Type:integer

State: 0  2                                         -6.0/2$                          RE->T
Symble:-  T
Value:9.0                   Type:integer

State: 0  1                                         -6.0/2$                          S7
Symble:-  E
Value:9.0                   Type:integer

State: 0  1  7                                      6.0/2$                           S4
Symble:-  E  -

State: 0  1  7  4                                   .0/2$                            S10
Symble:-  E  -  6

State: 0  1  7  4  10                               0/2$                             S16
Symble:-  E  -  6  .

State: 0  1  7  4  10 16                            /2$                              RF->num.num
Symble:-  E  -  6  .  0

State: 0  1  7  3                                   /2$                              RT->F
Symble:-  E  -  F
Value:6.0                   Type:real

State: 0  1  7  13                                  /2$                              S9
Symble:-  E  -  T
Value:6.0                   Type:real

State: 0  1  7  13 9                                2$                               S4
Symble:-  E  -  T  /

State: 0  1  7  13 9  4                             $                                RF->num
Symble:-  E  -  T  /  2

State: 0  1  7  13 9  15                            $                                RT->T/F
Symble:-  E  -  T  /  F
Value:2.0                   Type:integer

State: 0  1  7  13                                  $                                RE->E-T
Symble:-  E  -  T
Value:3.0                   Type:real

State: 0  1                                         $                                Accept
Symble:-  E
Value:6.0                   Type:real
```

说明：该测试样例的输入为正确的符号串序列。语义分析及翻译程序进行分析，给出分析过程中所调

用的产生式，并输出所识别出的子表达式的值和类型。

2. 测试用例 2

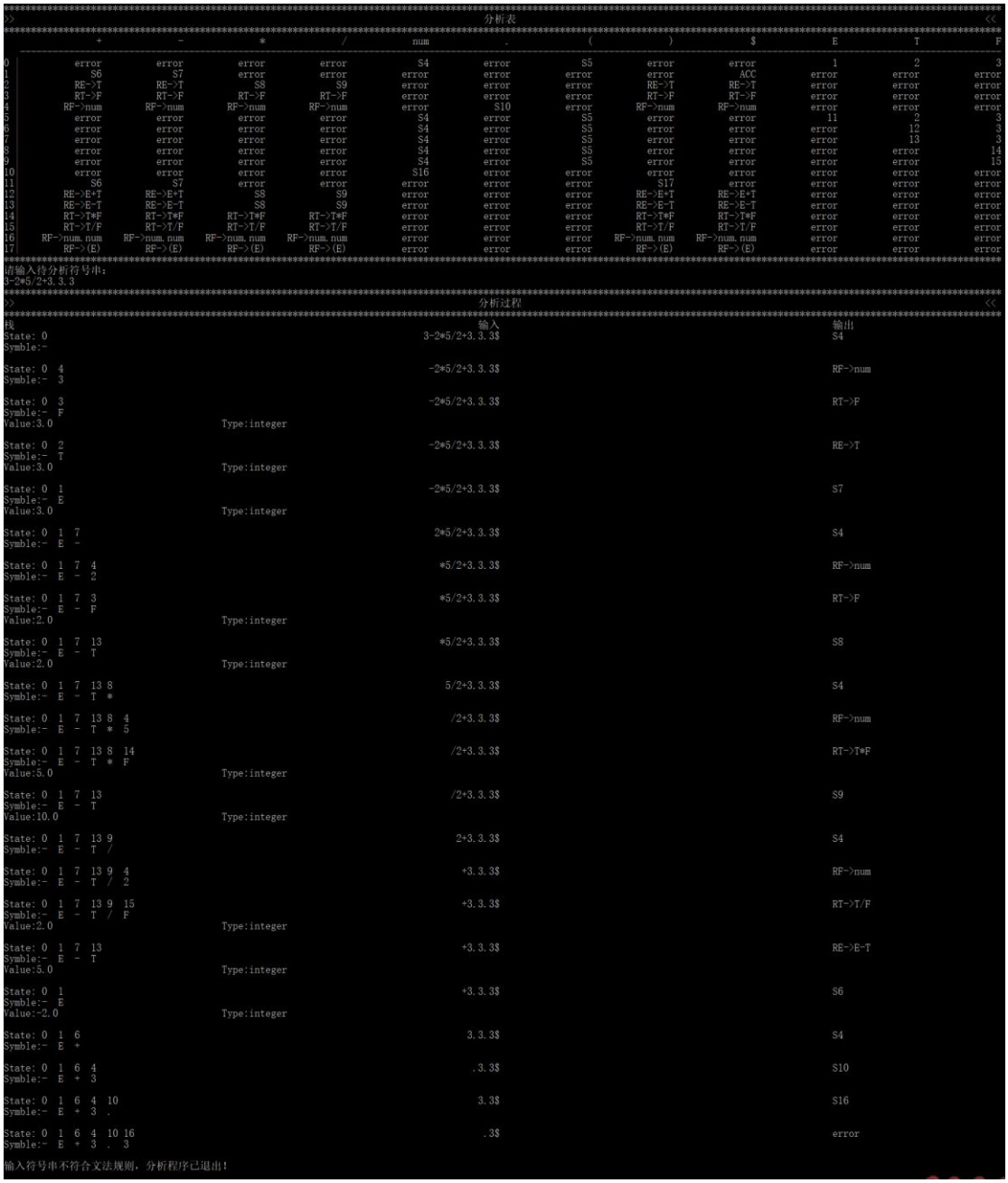
输入及运行结果如下：



说明：该测试样例的输入仍为正确的符号串序列。语义分析及翻译程序进行分析，给出分析过程中所调用的产生式，并输出所识别出的子表达式的值和类型。

3. 测试用例 3

输入及运行结果如下：



分析表

	+	-	*	/	num	.	(	)	$	E	T	F
0	error	error	error	error	S4	error	S5	error	error	1	2	3
1	S6	S7	error	error	error	error	error	error	ACC	error	error	error
2	RE->T	RE->T	S8	S9	error	error	error	RE->T	RE->T	error	error	error
3	RT->F	RT->F	RT->F	RT->F	error	error	error	RT->F	RT->F	error	error	error
4	RF->num	RF->num	RF->num	RF->num	error	S10	error	RF->num	RF->num	error	error	error
5	error	error	error	error	S4	error	S5	error	error	11	2	3
6	error	error	error	error	S4	error	S5	error	error	error	12	3
7	error	error	error	error	S4	error	S5	error	error	error	13	3
8	error	error	error	error	S4	error	S5	error	error	error	error	14
9	error	error	error	error	S4	error	S5	error	error	error	error	15
10	error	error	error	error	S16	error	error	error	error	error	error	error
11	S6	S7	error	error	error	error	error	S17	error	error	error	error
12	RE->E+T	RE->E+T	S8	S9	error	error	error	RE->E+T	RE->E+T	error	error	error
13	RE->E-T	RE->E-T	S8	S9	error	error	error	RE->E-T	RE->E-T	error	error	error
14	RT->T*F	RT->T*F	RT->T*F	RT->T*F	error	error	error	RT->T*F	RT->T*F	error	error	error
15	RT->T/F	RT->T/F	RT->T/F	RT->T/F	error	error	error	RT->T/F	RT->T/F	error	error	error
16	RF->num.num	RF->num.num	RF->num.num	RF->num.num	error	error	error	RF->num.num	RF->num.num	error	error	error
17	RF->(E)	RF->(E)	RF->(E)	RF->(E)	error	error	error	RF->(E)	RF->(E)	error	error	error

```
请输入待分析符号串：
3-2*5/2+3.3.3
```

分析过程

栈	输入	输出
State: 0 Symble:-	3-2*5/2+3.3.3$	S4
State: 0 4 Symble:- 3	-2*5/2+3.3.3$	RF->num
State: 0 3 Symble:- F Value:3.0 Type:integer	-2*5/2+3.3.3$	RT->F
State: 0 2 Symble:- T Value:3.0 Type:integer	-2*5/2+3.3.3$	RE->T
State: 0 1 Symble:- E Value:3.0 Type:integer	-2*5/2+3.3.3$	S7
State: 0 1 7 Symble:- E -	2*5/2+3.3.3$	S4
State: 0 1 7 4 Symble:- E - 2	*5/2+3.3.3$	RF->num
State: 0 1 7 3 Symble:- E - F Value:2.0 Type:integer	*5/2+3.3.3$	RT->F
State: 0 1 7 13 Symble:- E - T Value:2.0 Type:integer	*5/2+3.3.3$	S8
State: 0 1 7 13 8 Symble:- E - T *	5/2+3.3.3$	S4
State: 0 1 7 13 8 4 Symble:- E - T * 5	/2+3.3.3$	RF->num
State: 0 1 7 13 8 14 Symble:- E - T * F Value:5.0 Type:integer	/2+3.3.3$	RT->T*F
State: 0 1 7 13 Symble:- E - T Value:10.0 Type:integer	/2+3.3.3$	S9
State: 0 1 7 13 9 Symble:- E - T /	2+3.3.3$	S4
State: 0 1 7 13 9 4 Symble:- E - T / 2	+3.3.3$	RF->num
State: 0 1 7 13 9 15 Symble:- E - T / F Value:2.0 Type:integer	+3.3.3$	RT->T/F
State: 0 1 7 13 Symble:- E - T Value:5.0 Type:integer	+3.3.3$	RE->E-T
State: 0 1 Symble:- E Value:-2.0 Type:integer	+3.3.3$	S6
State: 0 1 6 Symble:- E +	3.3.3$	S4
State: 0 1 6 4 Symble:- E + 3	.3.3$	S10
State: 0 1 6 4 10 Symble:- E + 3 .	3.3$	S16
State: 0 1 6 4 10 16 Symble:- E + 3 . 3	.3$	error

```
输入符号串不符合文法规则，分析程序已退出！
```

说明：该测试样例的输入为不符合文法的符号串序列。语义分析及翻译程序进行分析，遇到错误时输出提示信息，并退出。

四、 实验结论和心得

1. 实验结论

通过此次实验，进一步了解了语义分析及翻译程序的工作原理，重点解析了 LR 文法和语义制导翻译，并对其中一些关键点，如分析表的构造、语义制导翻译方案的设计等，有了更深刻的理解和更理性的认识。

2. 实验心得

在本次实验中，遇到的主要问题有以下三点：

a) 分析表的结构。LR 分析程序所利用的分析表有两个，即 action 和 goto。起初，考虑用两个二维数组分别实现。经思考，goto 表只在归约时进行访问，且访问方式与 action 相同，完全可以将二表合一存储，提高了程序的执行效率；

b) 栈的实现。语法分析程序编写时，利用字符数组进行栈的模拟，关于栈的操作不易实现。本实验则利用 vector 实现栈，数据结构有所改进。其对象自带的 push_back()和 pop_back()函数，可以轻松地实现出入栈操作；

c) 对于‘num.num’的处理。‘num.num’无法与其余产生式采用相同的方法入栈，需要进行特殊处理，即将前者的 value 值与后者 value 除以 10 的值相加后，再入栈；

实践出真知，本次语义分析程序的设计与实现实验是对课堂和书本所学知识的补充。语义分析的实际情形与已了解到的原理大体一致，但又复杂许多。通过自己动手，亲力亲为编写代码，加深了对语义分析及翻译程序的理解和记忆，收获颇丰。