编译原理与技术

SLR 语法分析实验

目录

2017-11-23

[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。编写自底向上 LR 语法分析程序：

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) 识别所有活前缀的 DFA
(5) 构造 SLR 分析表
(6) 构造 SLR 分析控制程序

2．　主要算法

```
算法 4.3-----------------------------> LR 分析控制程序
输入：文法 G 的一张分析表和一个输入符号串
输出：若 w∈L(G)，得到　的自底向上的分析，否则报错
方法：开始时，初始状态 S0 在栈顶，　$在输入缓冲区中。
          置 ip 指向 w$的第一个符号；
          do {
                令 S 是栈顶状态，a 是 ip 所指向的符号
                 if (action[S, a]==shift S') {
                      把 a 和 S　分别压入符号栈和状态栈;
                      推进 ip，使它指向下一个输入符号;
                 };
                 else if (action[S, a]==reduce by A->β) {
```

```
                从栈顶弹出|    |个符号; （令 S    是现在的栈顶状态）
                把 A 和 goto[S    ,   A]分别压入符号栈和状态栈;
                输出产生式 A        ;
            };
            else if (action[S,  a]==accept) return;
                    else    error();
    } while(1).
```

3． 程序实现

(1) 输入

待分析表达式字符串

如*180*62*+3

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

(2) 输出文件形式部分举例

对符号串的分析过程

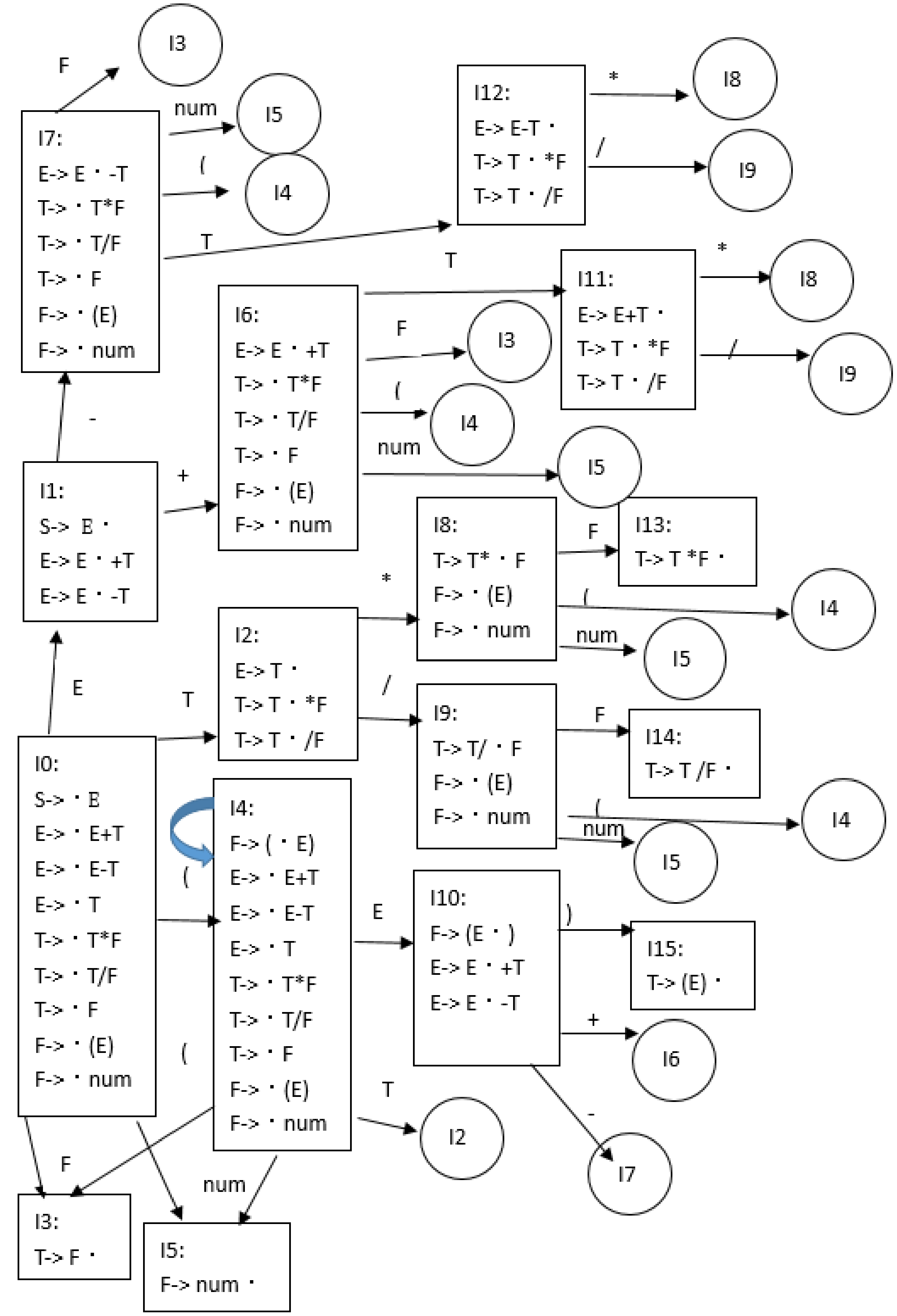
(3) 拓广文法

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

(4) FIRST 集与 FOLLOW 集

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

(5) 构造拓广文法的项目集规范族，识别所有活前缀的 DFA


I0:
S-> · E
E-> · E+T
E-> · E-T
E-> · T
T-> · T*F
T-> · T/F
T-> · F
F-> · (E)
F-> · num
I1:
S-> E ·
E-> E · +T
E-> E · -T
I2:
E-> T ·
T-> T · *F
T-> T · /F
I3:
T-> F ·
I4:
F-> (· E)
E-> · E+T
E-> · E-T
E-> · T
T-> · T*F
T-> · T/F
T-> · F
F-> · (E)
F-> · num
I5:
F-> num ·
I6:
E-> E · +T
T-> · T*F
T-> · T/F
T-> · F
F-> · (E)
F-> · num
I7:
E-> E · -T
T-> · T*F
T-> · T/F
T-> · F
F-> · (E)
F-> · num
I8:
T-> T* · F
F-> · (E)
F-> · num
I9:
T-> T/ · F
F-> · (E)
F-> · num
I10:
F-> (E ·)
E-> E · +T
E-> E · -T
I11:
E-> E+T ·
T-> T · *F
T-> T · /F
I12:
E-> E-T ·
T-> T · *F
T-> T · /F
I13:
T-> T *F ·
I14:
T-> T /F ·
I15:
T-> (E) ·
E
T
(
F
num
+
-
*
/
)
I2
I3
I4
I5
I6
I7
I8
I9

（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			
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			

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	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			

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

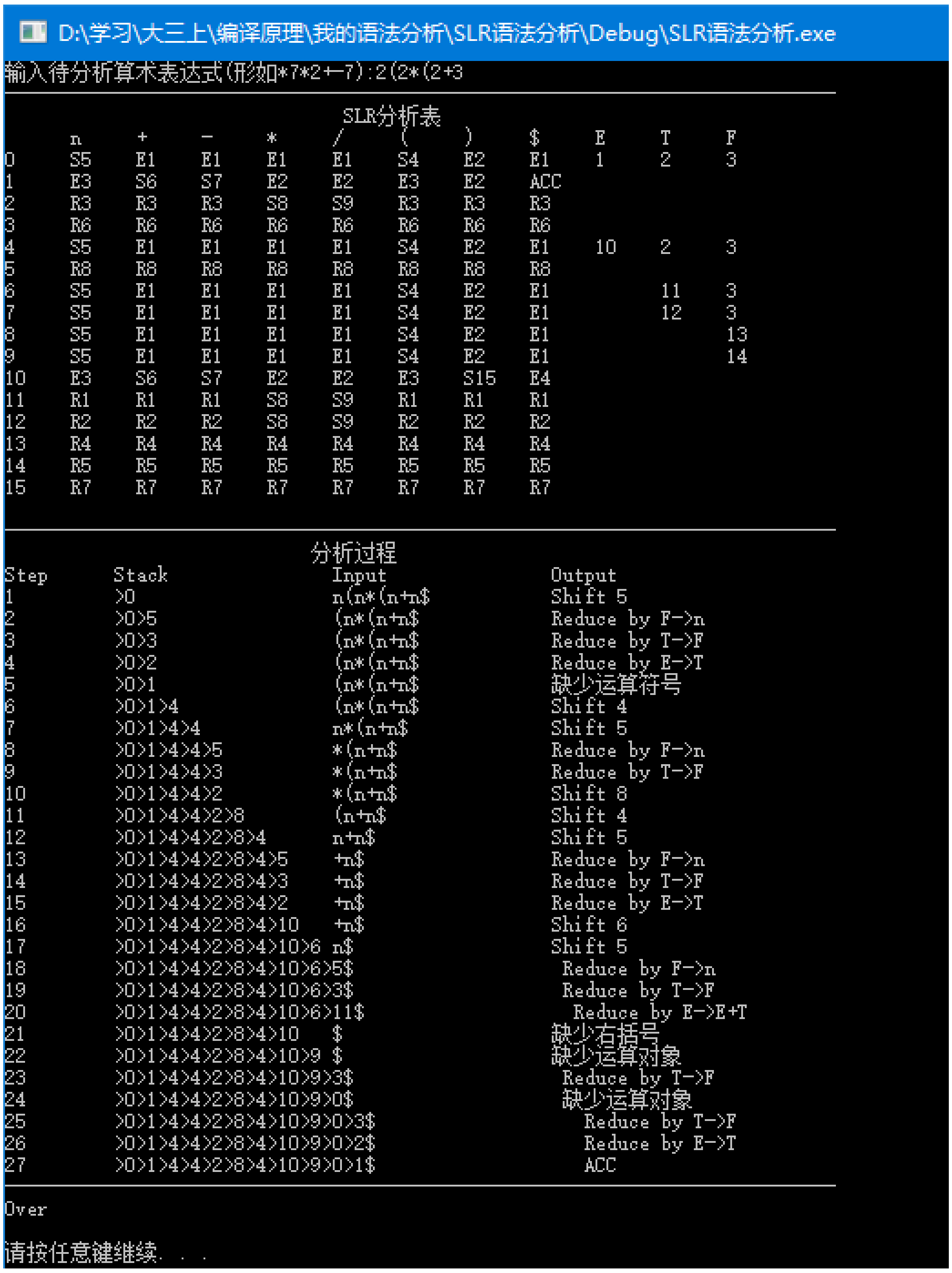
```
const int VnSize = 4;                          //非终结符个数
const int ProductionSize = 9;                  //产生式个数
const int StateNum = 16;                       //状态数
const int colNum = 11;                         //分析表列数

char Vn[VnSize] = { 'S', 'E', 'T','F' };       //非终结符
char GsItem[colNum] = { 'n','+','-','*','/','(',')','$', 'E','T','F' }; //分析表的列，文法符号
vector<char> FIRST[VnSize];                    //FIRST集
vector<char> FOLLOW[VnSize];                   //FOLLOW集

stack<int> AnalyseStack;                       //分析栈
string AnalyseTable[StateNum][colNum];         //预测分析表
string Production[ProductionSize];             //文法产生式
string inputBuffer;                            //输入缓冲区
int ip;                                        //输入缓冲区指针
```

(8)　函数与过程

```
int mapGs(char c);                             //非终结符与数字对应

void Initial();                                //产生式、FIRST、FOLLOW初始化

void createAnalyseTable();                     //创建预测分析表，算法4.2

void outStack();                               //输出栈内状态内容

void error(int kind);                          //错误处理

void SLR_analyse(string w);                    //算法4.3,SLR分析方法

string inStringChange(string w);               //输入数字转化为非终结符'n'

void TablePrint();                             //输出预测分析表
```

4． 样例结果

（1） 示例输入 1，特意错误出入 2(2*(2+3



D:\学习\大三上\编译原理\我的语法分析\SLR语法分析\Debug\SLR语法分析.exe

```
输入待分析算术表达式(形如*7*2+-7):2(2*(2+3
----------------------------------------------------------------------------
                               SLR分析表
      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

----------------------------------------------------------------------------
                            分析过程
Step      Stack               Input               Output
1         >0                  n(n*(n+n$           Shift 5
2         >0>5                (n*(n+n$            Reduce by F->n
3         >0>3                (n*(n+n$            Reduce by T->F
4         >0>2                (n*(n+n$            Reduce by E->T
5         >0>1                (n*(n+n$            缺少运算符号
6         >0>1>4              (n*(n+n$            Shift 4
7         >0>1>4>4            n*(n+n$             Shift 5
8         >0>1>4>4>5          *(n+n$              Reduce by F->n
9         >0>1>4>4>3          *(n+n$              Reduce by T->F
10        >0>1>4>4>2          *(n+n$              Shift 8
11        >0>1>4>4>2>8        (n+n$               Shift 4
12        >0>1>4>4>2>8>4      n+n$                Shift 5
13        >0>1>4>4>2>8>4>5    +n$                 Reduce by F->n
14        >0>1>4>4>2>8>4>3    +n$                 Reduce by T->F
15        >0>1>4>4>2>8>4>2    +n$                 Reduce by E->T
16        >0>1>4>4>2>8>4>10   +n$                 Shift 6
17        >0>1>4>4>2>8>4>10>6 n$                  Shift 5
18        >0>1>4>4>2>8>4>10>6>5$                   Reduce by F->n
19        >0>1>4>4>2>8>4>10>6>3$                   Reduce by T->F
20        >0>1>4>4>2>8>4>10>6>11$                   Reduce by E->E+T
21        >0>1>4>4>2>8>4>10   $                   缺少右括号
22        >0>1>4>4>2>8>4>10>9 $                   缺少运算对象
23        >0>1>4>4>2>8>4>10>9>3$                   Reduce by T->F
24        >0>1>4>4>2>8>4>10>9>0$                   缺少运算对象
25        >0>1>4>4>2>8>4>10>9>0>3$                   Reduce by T->F
26        >0>1>4>4>2>8>4>10>9>0>2$                   Reduce by E->T
27        >0>1>4>4>2>8>4>10>9>0>1$                   ACC
----------------------------------------------------------------------------
Over

请按任意键继续. . .
```

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：

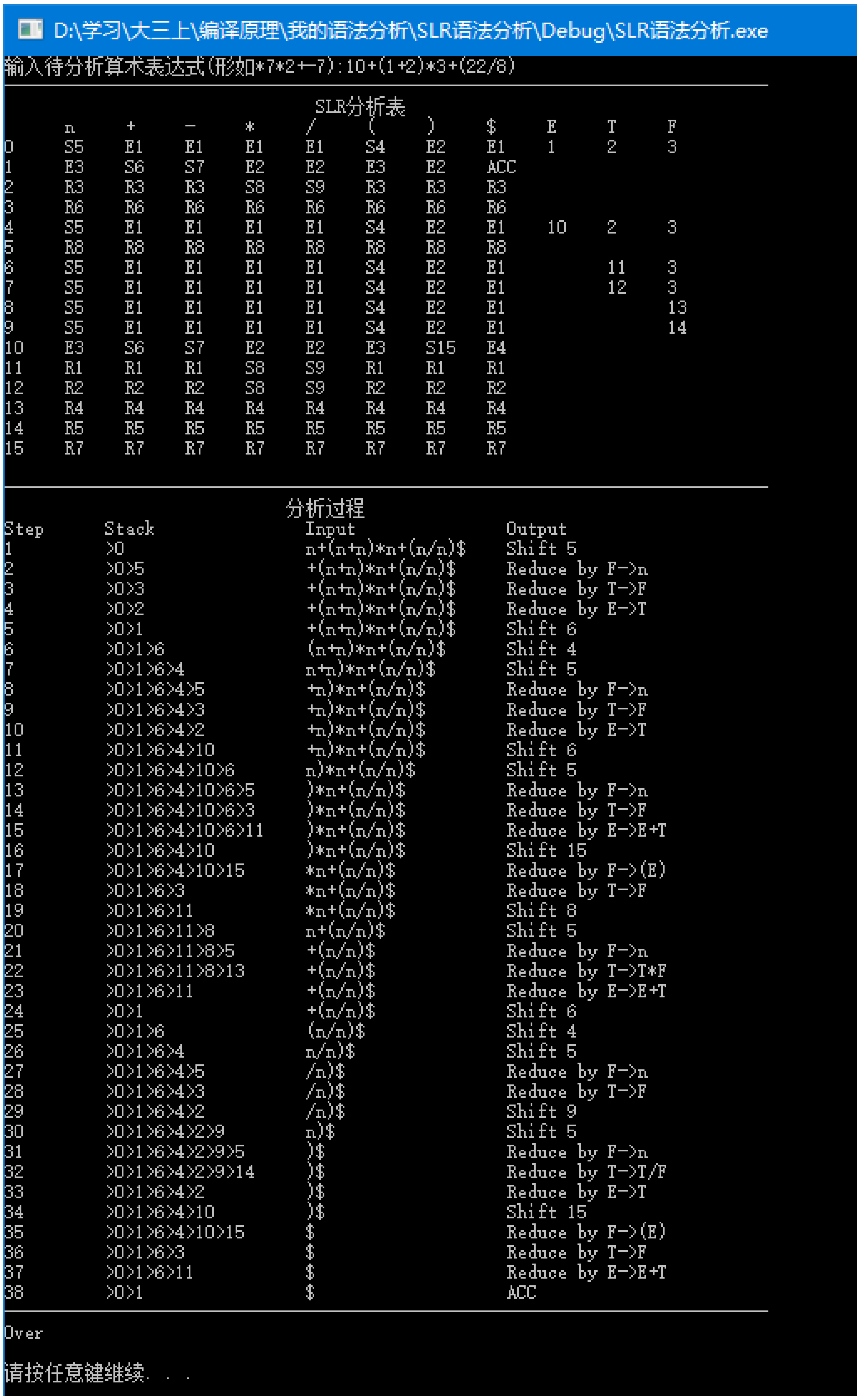
E1:缺少运算对象，状态3入栈，局部纠错

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

（2）　示例输入 2，10+(1+2)*3+(22/8)



D:\学习\大三上\编译原理\我的语法分析\SLR语法分析\Debug\SLR语法分析.exe

```
输入待分析算术表达式(形如*7*2+-7):10+(1+2)*3+(22/8)
--------------------------------------------------------------------------
                                       SLR分析表
      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

--------------------------------------------------------------------------
                                    分析过程
Step      Stack               Input                   Output
1         >0                  n+(n+n)*n+(n/n)$        Shift 5
2         >0>5                +(n+n)*n+(n/n)$         Reduce by F->n
3         >0>3                +(n+n)*n+(n/n)$         Reduce by T->F
4         >0>2                +(n+n)*n+(n/n)$         Reduce by E->T
5         >0>1                +(n+n)*n+(n/n)$         Shift 6
6         >0>1>6              (n+n)*n+(n/n)$          Shift 4
7         >0>1>6>4            n+n)*n+(n/n)$           Shift 5
8         >0>1>6>4>5          +n)*n+(n/n)$            Reduce by F->n
9         >0>1>6>4>3          +n)*n+(n/n)$            Reduce by T->F
10        >0>1>6>4>2          +n)*n+(n/n)$            Reduce by E->T
11        >0>1>6>4>10         +n)*n+(n/n)$            Shift 6
12        >0>1>6>4>10>6       n)*n+(n/n)$             Shift 5
13        >0>1>6>4>10>6>5     )*n+(n/n)$              Reduce by F->n
14        >0>1>6>4>10>6>3     )*n+(n/n)$              Reduce by T->F
15        >0>1>6>4>10>6>11    )*n+(n/n)$              Reduce by E->E+T
16        >0>1>6>4>10         )*n+(n/n)$              Shift 15
17        >0>1>6>4>10>15      *n+(n/n)$               Reduce by F->(E)
18        >0>1>6>3            *n+(n/n)$               Reduce by T->F
19        >0>1>6>11           *n+(n/n)$               Shift 8
20        >0>1>6>11>8         n+(n/n)$                Shift 5
21        >0>1>6>11>8>5       +(n/n)$                 Reduce by F->n
22        >0>1>6>11>8>13      +(n/n)$                 Reduce by T->T*F
23        >0>1>6>11           +(n/n)$                 Reduce by E->E+T
24        >0>1                +(n/n)$                 Shift 6
25        >0>1>6              (n/n)$                  Shift 4
26        >0>1>6>4            n/n)$                   Shift 5
27        >0>1>6>4>5          /n)$                    Reduce by F->n
28        >0>1>6>4>3          /n)$                    Reduce by T->F
29        >0>1>6>4>2          /n)$                    Shift 9
30        >0>1>6>4>2>9        n)$                     Shift 5
31        >0>1>6>4>2>9>5      )$                      Reduce by F->n
32        >0>1>6>4>2>9>14     )$                      Reduce by T->T/F
33        >0>1>6>4>2          )$                      Reduce by E->T
34        >0>1>6>4>10         )$                      Shift 15
35        >0>1>6>4>10>15      $                       Reduce by F->(E)
36        >0>1>6>3            $                       Reduce by T->F
37        >0>1>6>11           $                       Reduce by E->E+T
38        >0>1                $                       ACC
--------------------------------------------------------------------------

Over

请按任意键继续. . .
```

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 SLR 分析过程！！

5． 实验总结

这是编译原理与技术第二次程序设计实验，程序设计极大地参考并实现了课本上所给出的算法 4.3，还有输入输出形式，采用了较为规范的编程形式，使程序具有良好的可读性，在不断地改进与 debug 后能够实现所给出的算术表达式的分析，还有就是错误处理的实现，根据书上局部纠错的步骤得出错误信息填入 SLR 分析表，使之能够识别 4 种错误，这是令人欢欣鼓舞的！！

但是程序也有不足之处，程序中的文法改写，得出 FIRST、FOLLOW 集，关键的还有 SLR 分析表都是通过人力在纸上运算得出，然后手动配置进程序的 Initial()函数中，若能实现程序自动求出，将会变得更为智能。这是能够改进的。

这次实验获益匪浅，加深了对语法分析 SLR 文法的理解，也极大地激励了学习编译原理的热情，对课本内容的不断理解与探索，对 debug 的一丝不苟，都是对后面的学习将是极大的鼓励。

6． 源码附件

```
/**
*SLR_Parser.cpp 文件内容
*作者:裴子祥
*时间:2017.11.23
*/
#include<iostream>
#include<vector>
#include<string>
#include<stack>
#include<iomanip>

using namespace std;

const int VnSize = 4;                                //非终结符个数
const int ProductionSize = 9;                         //产生式个数
const int StateNum = 16;                             //状态数
const int colNum = 11;                               //分析表列数

char Vn[VnSize] = { 'S', 'E', 'T','F' };          //非终结符
char GsItem[colNum] = { 'n','+','-','*','/','(',')','$', 'E','T','F' }; //分析表的列，文
法符号
vector<char> FIRST[VnSize];                          //FIRST 集
vector<char> FOLLOW[VnSize];                         //FOLLOW 集

stack<int> AnalyseStack;                             //分析栈
string AnalyseTable[StateNum][colNum];                 //预测分析表
string Production[ProductionSize];                   //文法产生式
string inputBuffer;                                 //输入缓冲区
int ip;                                           //输入缓冲区指针


int mapGs(char c);                                  //非终结符与数字对应

void Initial();                                    //产生式、FIRST、FOLLOW 初始化

void createAnalyseTable();                            //创建预测分析表，算法 4.2

void outStack();                                    //输出栈内状态内容

void error(int kind);                                //错误处理

void SLR_analyse(string w);                           //算法 4.3,SLR 分析方法
```

```
string inStringChange(string w);                //输入数字转化为非终结符'n'

void TablePrint();                            //输出预测分析表


int main()//主函数
{
    string s;
    cout << "输入待分析算术表达式(形如*7*2+-7):";
    cin >> s;
    string w = inStringChange(s);
    Initial();
    createAnalyseTable();
    TablePrint();
    SLR_analyse(w);

    system("pause");
    return 0;
}

int mapGs(char c)//文法符号与数字对应
{
    switch (c) {
    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;
    }
}



void Initial()
{
    //拓广文法
    Production[0] = "S->E";
    Production[1] = "E->E+T";
    Production[2] = "E->E-T";
    Production[3] = "E->T";
    Production[4] = "T->T*F";
    Production[5] = "T->T/F";
    Production[6] = "T->F";
    Production[7] = "F->(E)";
    Production[8] = "F->n";

    //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('/');

    for (int i = 0; i < StateNum; ++i)
    {
        for (int j = 0; j < colNum; ++j)
        {
            AnalyseTable[i][j] = "";//SLR 分析表项置空
        }
    }
}

//创建 SLR 分析表，人工计算后获得
void createAnalyseTable()
{
    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";

}

//输出站内容
void outStack()
{
    string s = "";
    int itemp;
    stack<int> stemp;
    while (!AnalyseStack.empty())
    {
        itemp = AnalyseStack.top();
        AnalyseStack.pop();
        stemp.push(itemp);
    }
    while (!stemp.empty())
    {
        itemp = stemp.top();

        char num[5];
        _itoa_s(itemp, num, 10);
        s = s + '>' + num;

        AnalyseStack.push(itemp);
        stemp.pop();
    }
    cout << s;
}


void error(int kind)
{
    switch (kind)
    {
    case 1:
        cout << "缺少运算对象" << endl;
        AnalyseStack.push(3);
        break;
    case 2:
        cout << "括号不匹配" << endl;
        ip++;
        break;
    case 3:
        cout << "缺少运算符号" << endl;
        AnalyseStack.push(4);
        break;
    case 4:
        cout << "缺少右括号" << endl;
        AnalyseStack.push(9);
        break;
    default:
        cout << "未知错误" << endl;
        break;
    }
}
```

```
void SLR_analyse(string w)//算法 4.1,非递归预测分析方法
{
    int X;//指向栈顶状态
    char a;//ip 所指向的输入符号
    int aCol = -1;//a 对应的列号
    ip = 0;//w$指针，初始指向第一个符号
    int step = 0;//记录步骤
    inputBuffer = w;
    AnalyseStack.push(0);
    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 = AnalyseStack.top();
        a = inputBuffer[ip];
        aCol = mapGs(a);
        ++step;
        cout << left << setw(10) << step << left << setw(20);
        outStack();
        cout << left << setw(20) << inputBuffer.substr(ip);

        if (aCol == -1)
        {
            cout << "非法输入" << endl;
            exit(0);
        }

        if (AnalyseTable[X][aCol][0] == 'S')
        {//移进
            string temp = "";//读出转移状态号
            for (int i = 1; i < AnalyseTable[X][aCol].length(); ++i)
            {
                temp += AnalyseTable[X][aCol][i];
            }
            int numCol = atoi(temp.c_str());
            AnalyseStack.push(numCol);
            ++ip;
            cout << "Shift " << numCol << endl;
        }
        else if (AnalyseTable[X][aCol][0] == 'R')
        {//规约
```

```
            int numP = int(AnalyseTable[X][aCol][1] - '0');//产生式号
            int ProdRightLen = Production[numP].length() - 3;
            for (int i = 0; i < ProdRightLen; ++i)
            {
                AnalyseStack.pop();
            }
            X = AnalyseStack.top();
            a = Production[numP][0];
            aCol = mapGs(a);

            string temp = "";//转移状态号
            for (int i = 0; i < AnalyseTable[X][aCol].length(); ++i)
            {
                temp += AnalyseTable[X][aCol][i];
            }
            int state = atoi(temp.c_str());
            AnalyseStack.push(state);//状态入栈

            //输出 A->β
            cout << "Reduce by " << Production[numP] << endl;
        }
        else if (AnalyseTable[X][aCol] == "ACC")
        {//成功接收
            cout << "ACC" << endl;
            cout << "-----------------------------------------------------------------------
---------" << endl;
            cout << "Over" << endl << endl;
            return;
        }
        else if (AnalyseTable[X][aCol][0] == 'E')
        {
            error(int(AnalyseTable[X][aCol][1] - '0'));
        }
        else
        {
            error(-1);
        }

    } while (true);
}


string inStringChange(string w)
{
    string temp = "";
    for (int i = 0; i < w.length(); ++i)
    {
        //cout << w[i] << endl;
        if (w[i] >= '0'&&w[i] <= '9'&&i < w.length())
        {//是数字
            ++i;
            while (w[i] >= '0'&&w[i] <= '9'&&i < w.length())
            {
                ++i;
            }
            temp += 'n';
            --i;
        }
        else
        {
            temp += w[i];
        }
    }
    temp += '$';
    return temp;
}


//输出 SLR 分析表
void TablePrint()
{
    cout << "---------------------------------------------------------------------------
--" << endl;
    cout << "                                    SLR 分析表" << endl;
    cout << left << setw(6) << " ";
    for (int j = 0; j < colNum; j++) {
        cout << left << setw(6) << GsItem[j];
    }
    cout << endl;
    for (int i = 0; i < StateNum; i++) {
        cout << left << setw(6) << i;
        for (int j = 0; j < colNum; j++) {
            cout << left << setw(6) << AnalyseTable[i][j];
        }
        cout << endl;
    }
    cout << endl;
    cout << "---------------------------------------------------------------------------
--" << endl;
}
```