编译器专题实验报告

实验四:语法分析

实验内容:

目的：根据某文法写出SLR(1)分析表。

构造SLR(1)分析表的步骤:

1. 写出拓广文法
2. 画出项目集规范族
3. 求该非终结符的FOLLOW集
4. 判断是否是SLR(1)文法
5. 构造SLR(1)分析表

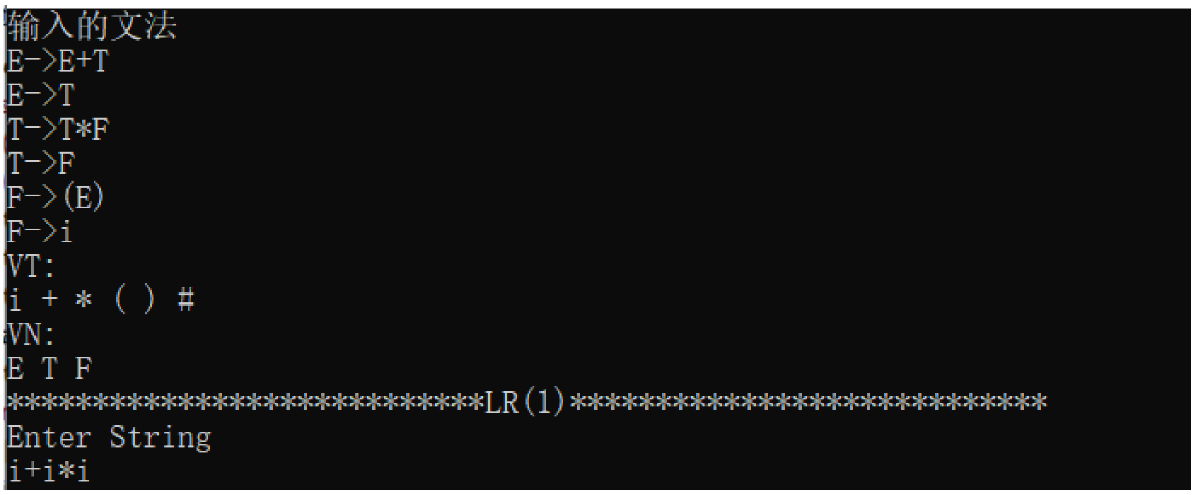
根据提供的slrs.cpp，可输出2种以上的文法SLR(1)分析表，实现分析表的打印输出

Shift、Goto和Reduction函数是解析过程的关键。它们分别处理转换状态、基于非终结符进入新状态以及基于生成规则减少语法。

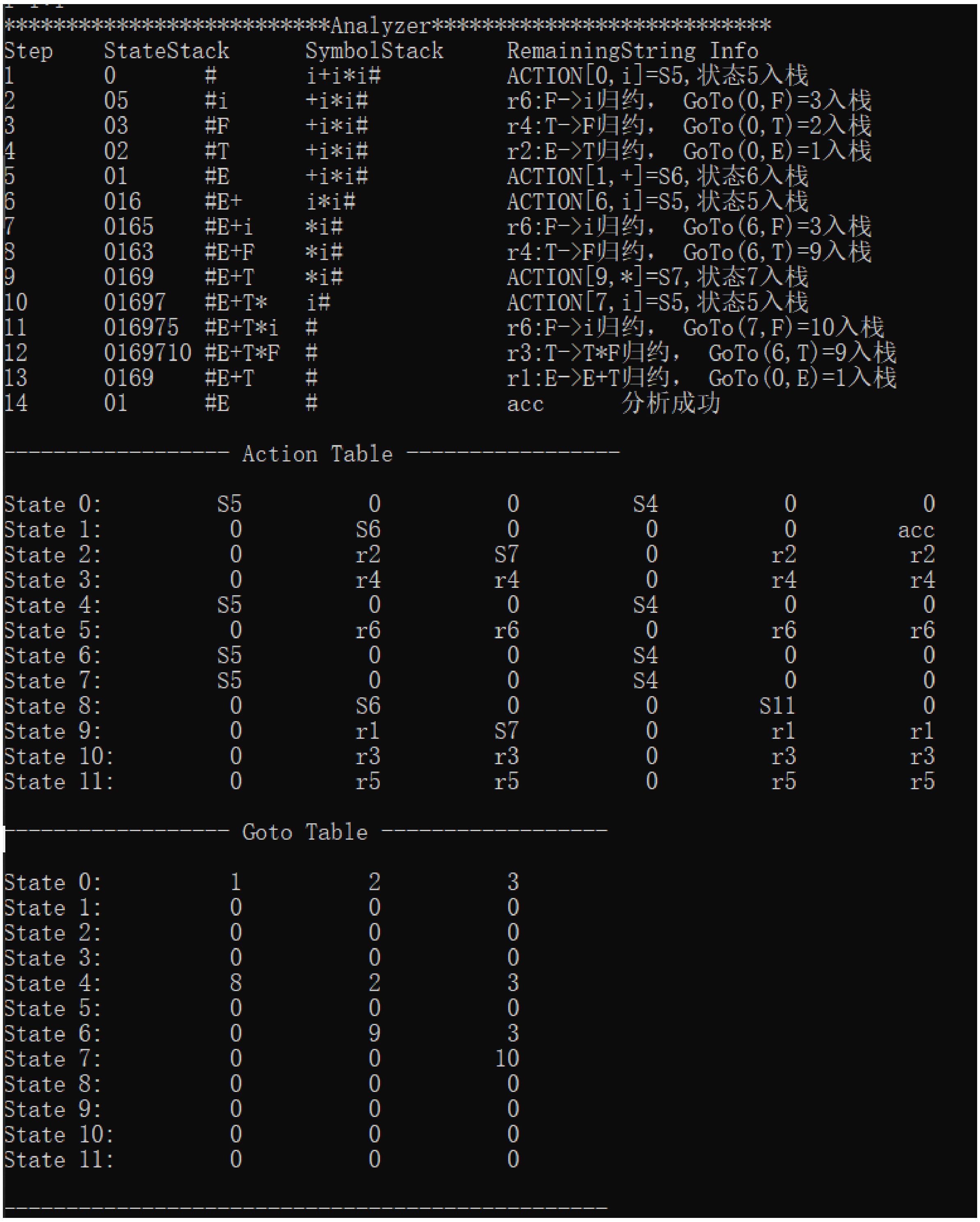
实验结果:

文法1产生式集合： **文法1支持加法和乘法**

`{"E->E+T", "E->T", "T->T*F", "T->F", "F->(E)", "F->i"}`



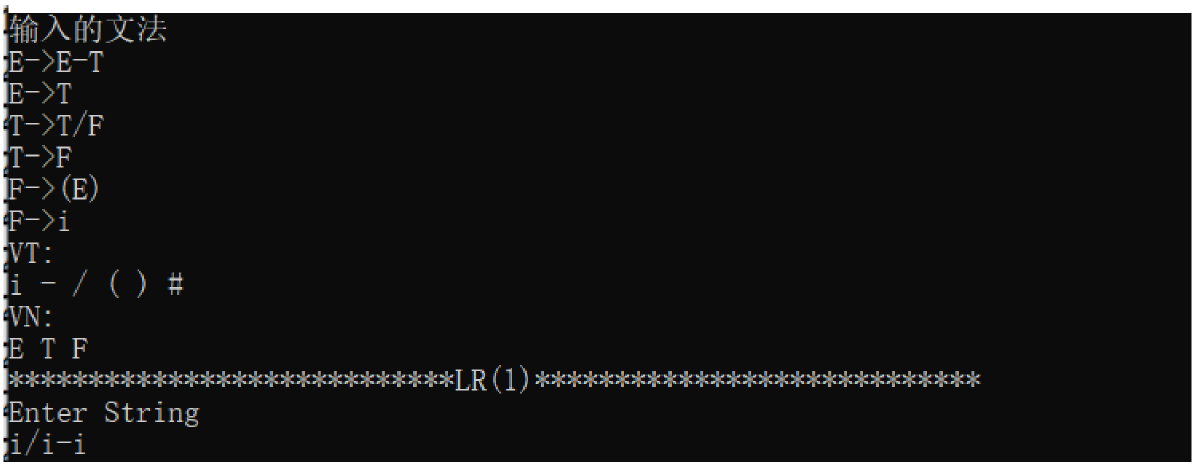
```
输入的文法
E->E+T
E->T
T->T*F
T->F
F->(E)
F->i
VT:
i + * ( ) #
VN:
E T F
****************************LR(1)****************************
Enter String
i+i*i
```



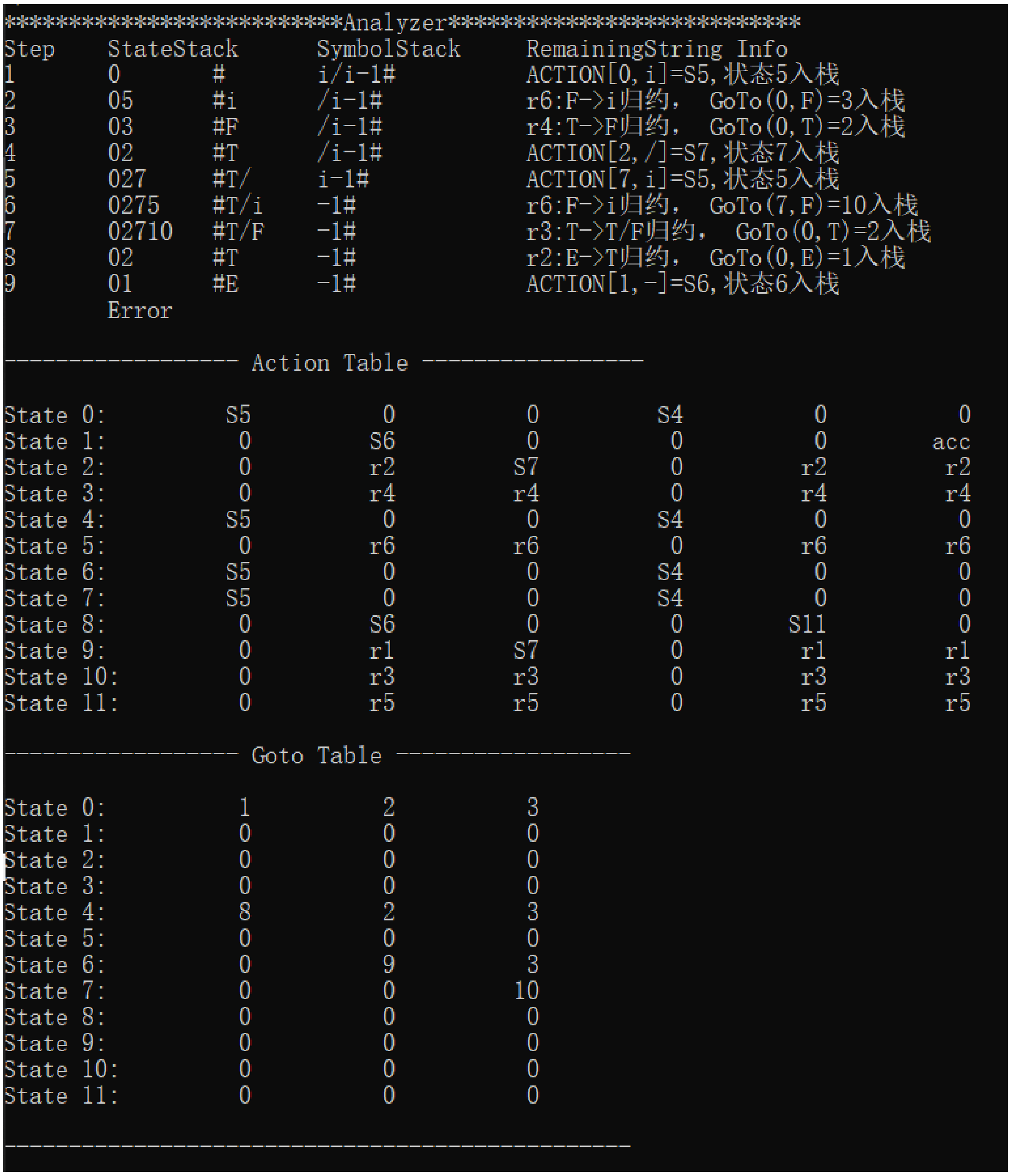
```
************************Analyzer*************************
Step    StateStack      SymbolStack     RemainingString Info
1       0       #       i+i*i#          ACTION[0,i]=S5,状态5入栈
2       05      #i      +i*i#           r6:F->i归约,  GoTo(0,F)=3入栈
3       03      #F      +i*i#           r4:T->F归约,  GoTo(0,T)=2入栈
4       02      #T      +i*i#           r2:E->T归约,  GoTo(0,E)=1入栈
5       01      #E      +i*i#           ACTION[1,+]=S6,状态6入栈
6       016     #E+     i*i#            ACTION[6,i]=S5,状态5入栈
7       0165    #E+i    *i#             r6:F->i归约,  GoTo(6,F)=3入栈
8       0163    #E+F    *i#             r4:T->F归约,  GoTo(6,T)=9入栈
9       0169    #E+T    *i#             ACTION[9,*]=S7,状态7入栈
10      01697   #E+T*   i#              ACTION[7,i]=S5,状态5入栈
11      016975  #E+T*i  #               r6:F->i归约,  GoTo(7,F)=10入栈
12      0169710 #E+T*F  #               r3:T->T*F归约,  GoTo(6,T)=9入栈
13      0169    #E+T    #               r1:E->E+T归约,  GoTo(0,E)=1入栈
14      01      #E      #               acc      分析成功

------------------ Action Table ------------------

State 0:          S5         0         0        S4         0         0
State 1:           0        S6         0         0         0       acc
State 2:           0        r2        S7         0        r2        r2
State 3:           0        r4        r4         0        r4        r4
State 4:          S5         0         0        S4         0         0
State 5:           0        r6        r6         0        r6        r6
State 6:          S5         0         0        S4         0         0
State 7:          S5         0         0        S4         0         0
State 8:           0        S6         0         0       S11         0
State 9:           0        r1        S7         0        r1        r1
State 10:          0        r3        r3         0        r3        r3
State 11:          0        r5        r5         0        r5        r5

------------------ Goto Table ------------------

State 0:           1         2         3
State 1:           0         0         0
State 2:           0         0         0
State 3:           0         0         0
State 4:           8         2         3
State 5:           0         0         0
State 6:           0         9         3
State 7:           0         0        10
State 8:           0         0         0
State 9:           0         0         0
State 10:          0         0         0
State 11:          0         0         0

------------------------------------------------
```

文法2产生式集合：**文法2支持减法和除法**

`{"E->E-T", "E->T", "T->T/F", "T->F", "F->(E)", "F->i"}`



```
输入的文法
E->E-T
E->T
T->T/F
T->F
F->(E)
F->i
VT:
i - / ( ) #
VN:
E T F
*****************************LR(1)****************************
Enter String
i/i-i
```

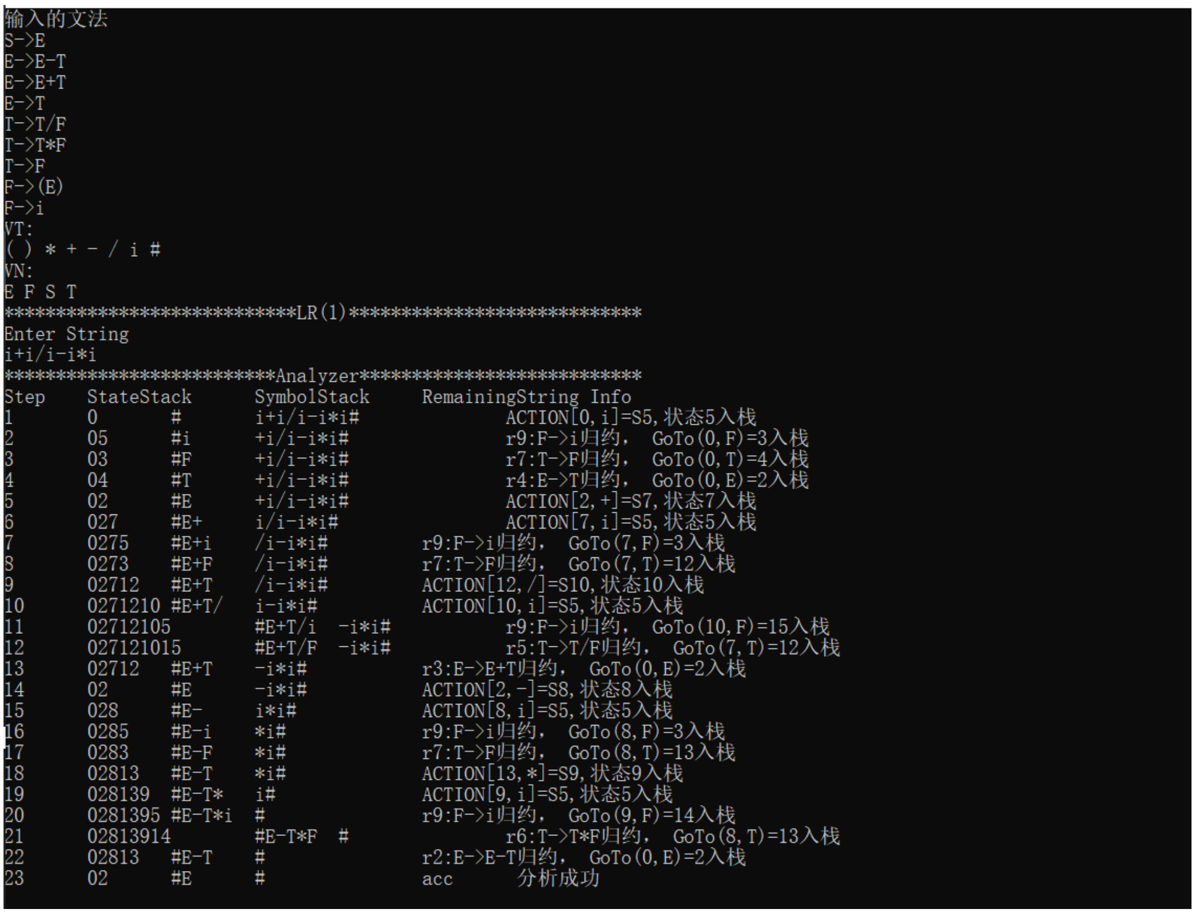


```
*************************Analyzer*************************
Step    StateStack      SymbolStack     RemainingString Info
1       0       #       i/i-1#          ACTION[0,i]=S5,状态5入栈
2       05      #i      /i-1#           r6:F->i归约,  GoTo(0,F)=3入栈
3       03      #F      /i-1#           r4:T->F归约,  GoTo(0,T)=2入栈
4       02      #T      /i-1#           ACTION[2,/]=S7,状态7入栈
5       027     #T/     i-1#            ACTION[7,i]=S5,状态5入栈
6       0275    #T/i    -1#             r6:F->i归约,  GoTo(7,F)=10入栈
7       02710   #T/F    -1#             r3:T->T/F归约,  GoTo(0,T)=2入栈
8       02      #T      -1#             r2:E->T归约,  GoTo(0,E)=1入栈
9       01      #E      -1#             ACTION[1,-]=S6,状态6入栈
        Error

------------------ Action Table ------------------

State 0:         S5         0          0          S4         0          0
State 1:         0          S6         0          0          0          acc
State 2:         0          r2         S7         0          r2         r2
State 3:         0          r4         r4         0          r4         r4
State 4:         S5         0          0          S4         0          0
State 5:         0          r6         r6         0          r6         r6
State 6:         S5         0          0          S4         0          0
State 7:         S5         0          0          S4         0          0
State 8:         0          S6         0          0          S11        0
State 9:         0          r1         S7         0          r1         r1
State 10:        0          r3         r3         0          r3         r3
State 11:        0          r5         r5         0          r5         r5

------------------ Goto Table ------------------

State 0:         1          2          3
State 1:         0          0          0
State 2:         0          0          0
State 3:         0          0          0
State 4:         8          2          3
State 5:         0          0          0
State 6:         0          9          3
State 7:         0          0          10
State 8:         0          0          0
State 9:         0          0          0
State 10:        0          0          0
State 11:        0          0          0

------------------------------------------------
```

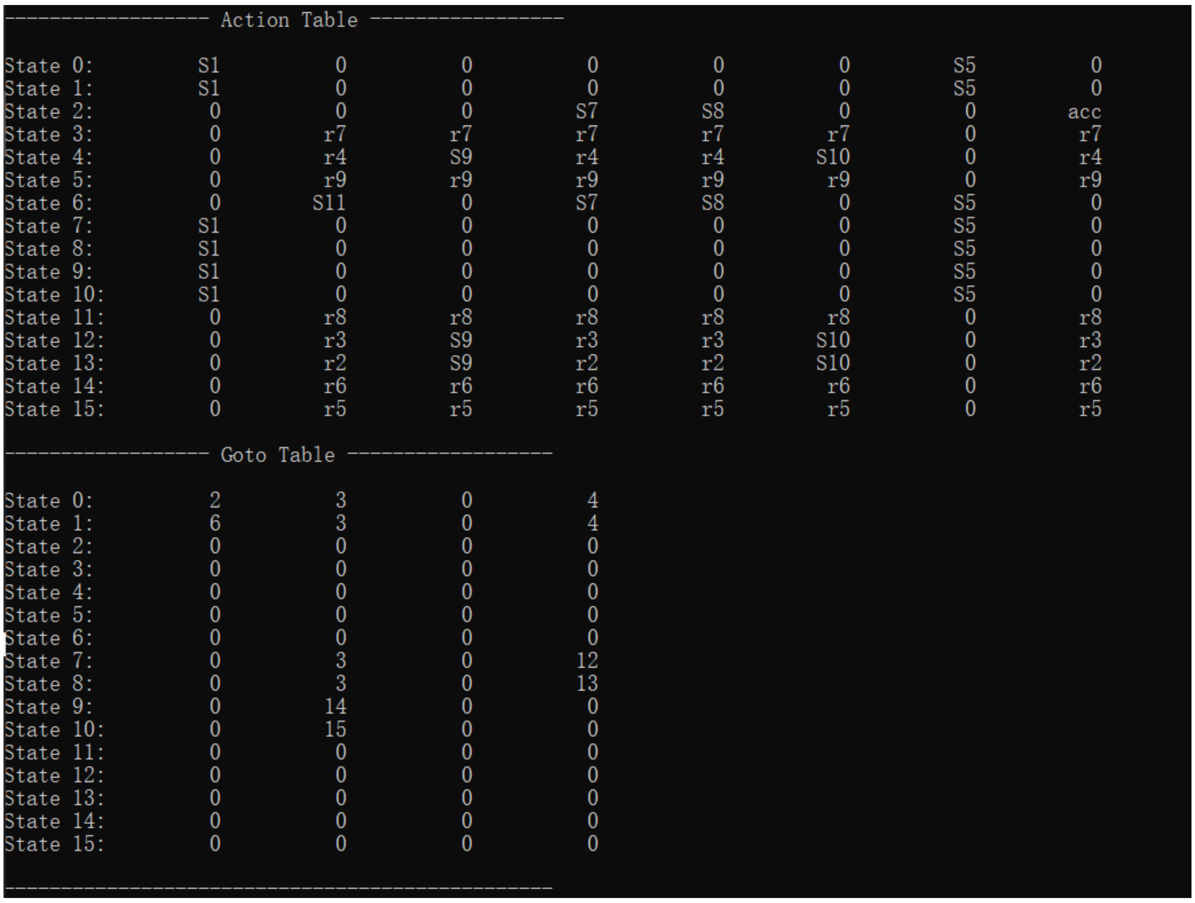
文法3产生式集合: **文法3支持加减乘除**

```
//第三种文法:
string action[16][8] = {"S1", "0", "0", "0", "0", "0","S5","0",
                        "S1", "0", "0", "0", "0", "0","S5","0",
                        "0", "0", "0", "S7", "S8", "0","0","acc",
                        "0", "r7", "r7", "r7", "r7", "r7","0","r7",
                        "0", "r4", "S9", "r4", "r4", "S10","0","r4",
                        "0", "r9", "r9", "r9", "r9", "r9","0","r9",
                        "0", "S11", "0", "S7", "S8", "0","S5","0",
                        "S1", "0", "0", "0", "0", "0","S5","0",
                        "S1", "0", "0", "0", "0", "0","S5","0",
                        "S1", "0", "0", "0", "0", "0","S5","0",
                        "S1", "0", "0", "0", "0", "0","S5","0",
                        "0", "r8", "r8", "r8", "r8", "r8","0","r8",
                        "0", "r3", "S9", "r3", "r3", "S10","0","r3",
                        "0", "r2", "S9", "r2", "r2", "S10","0","r2",
                        "0", "r6", "r6", "r6", "r6", "r6","0","r6",
                                "0", "r5", "r5", "r5", "r5", "r5","0","r5"
                                };
int gotoarr[16][4] = {2,3,0,4,
                      6,3,0,4,
                      0,0,0,0,
                      0,0,0,0,
                      0,0,0,0,
                      0,0,0,0,
                      0,0,0,0,
                      0,3,0,12,
                      0,3,0,13,
                      0,14,0,0,
                      0,15,0,0,
                      0,0,0,0,
                      0,0,0,0,
                      0,0,0,0,
                      0,0,0,0,
                      0,0,0,0
                  };
char vt[8] = {'(', ')', '*', '+','-', '/', 'i', '#'}; //存放终结符
char vn[4] = {'E','F', 'S', 'T'}; //存放非终结符
string Production[9] = {"S->E","E->E-T", "E->E+T","E->T", "T->T/F","T->T*F", "T->F", "F->(E)", "F->i"};//产生式集合
```

运行结果:



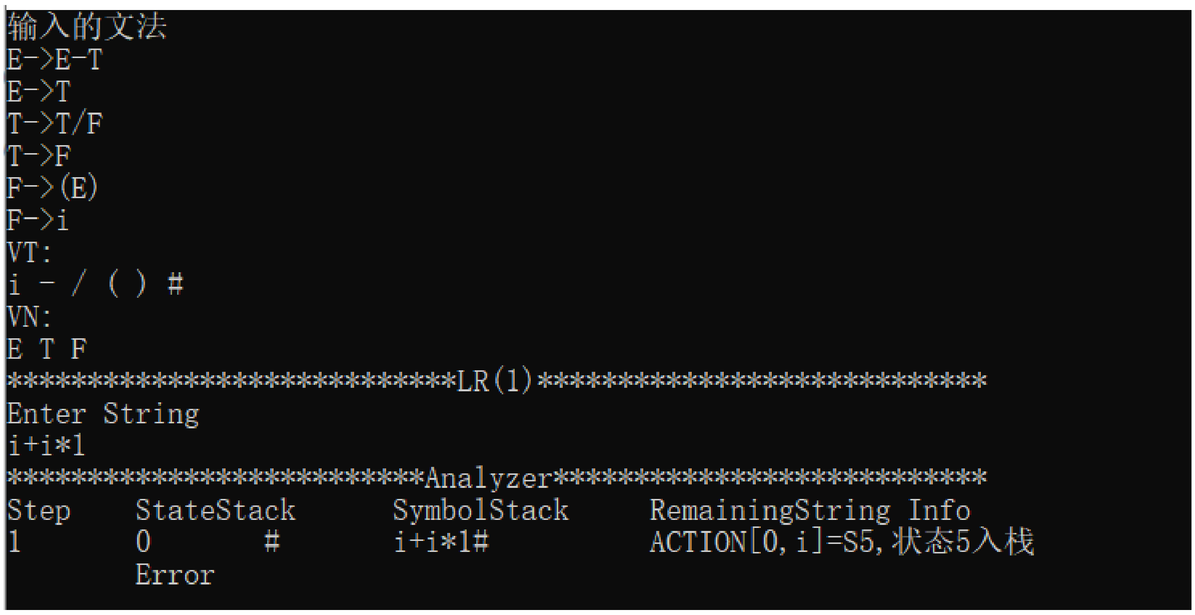
```
输入的文法
S->E
E->E-T
E->E+T
E->T
T->T/F
T->T*F
T->F
F->(E)
F->i
VT:
( ) * + - / i #
VN:
E F S T
****************************LR(1)****************************
Enter String
i+i/i-i*i
**************************Analyzer***************************
Step    StateStack      SymbolStack     RemainingString Info
1       0       #       i+i/i-i*i#              ACTION[0,i]=S5,状态5入栈
2       05      #i      +i/i-i*i#               r9:F->i归约,  GoTo(0,F)=3入栈
3       03      #F      +i/i-i*i#               r7:T->F归约,  GoTo(0,T)=4入栈
4       04      #T      +i/i-i*i#               r4:E->T归约,  GoTo(0,E)=2入栈
5       02      #E      +i/i-i*i#               ACTION[2,+]=S7,状态7入栈
6       027     #E+     i/i-i*i#                ACTION[7,i]=S5,状态5入栈
7       0275    #E+i    /i-i*i#         r9:F->i归约,  GoTo(7,F)=3入栈
8       0273    #E+F    /i-i*i#         r7:T->F归约,  GoTo(7,T)=12入栈
9       02712   #E+T    /i-i*i#         ACTION[12,/]=S10,状态10入栈
10      0271210 #E+T/   i-i*i#          ACTION[10,i]=S5,状态5入栈
11      02712105        #E+T/i  -i*i#           r9:F->i归约,  GoTo(10,F)=15入栈
12      027121015       #E+T/F  -i*i#           r5:T->T/F归约,  GoTo(7,T)=12入栈
13      02712   #E+T    -i*i#           r3:E->E+T归约,  GoTo(0,E)=2入栈
14      02      #E      -i*i#           ACTION[2,-]=S8,状态8入栈
15      028     #E-     i*i#            ACTION[8,i]=S5,状态5入栈
16      0285    #E-i    *i#             r9:F->i归约,  GoTo(8,F)=3入栈
17      0283    #E-F    *i#             r7:T->F归约,  GoTo(8,T)=13入栈
18      02813   #E-T    *i#             ACTION[13,*]=S9,状态9入栈
19      028139  #E-T*   i#              ACTION[9,i]=S5,状态5入栈
20      0281395 #E-T*i  #               r9:F->i归约,  GoTo(9,F)=14入栈
21      02813914        #E-T*F  #               r6:T->T*F归约,  GoTo(8,T)=13入栈
22      02813   #E-T    #               r2:E->E-T归约,  GoTo(0,E)=2入栈
23      02      #E      #               acc     分析成功
```



```
----------------- Action Table -----------------

State 0:        S1      0       0       0       0       0       S5      0
State 1:        S1      0       0       0       0       0       S5      0
State 2:        0       0       0       S7      S8      0       0       acc
State 3:        0       r7      r7      r7      r7      r7      0       r7
State 4:        0       r4      S9      r4      r4      S10     0       r4
State 5:        0       r9      r9      r9      r9      r9      0       r9
State 6:        0       S11     0       S7      S8      0       S5      0
State 7:        S1      0       0       0       0       0       S5      0
State 8:        S1      0       0       0       0       0       S5      0
State 9:        S1      0       0       0       0       0       S5      0
State 10:       S1      0       0       0       0       0       S5      0
State 11:       0       r8      r8      r8      r8      r8      0       r8
State 12:       0       r3      S9      r3      r3      S10     0       r3
State 13:       0       r2      S9      r2      r2      S10     0       r2
State 14:       0       r6      r6      r6      r6      r6      0       r6
State 15:       0       r5      r5      r5      r5      r5      0       r5

----------------- Goto Table ------------------

State 0:        2       3       0       4
State 1:        6       3       0       4
State 2:        0       0       0       0
State 3:        0       0       0       0
State 4:        0       0       0       0
State 5:        0       0       0       0
State 6:        0       0       0       0
State 7:        0       3       0       12
State 8:        0       3       0       13
State 9:        0       14      0       0
State 10:       0       15      0       0
State 11:       0       0       0       0
State 12:       0       0       0       0
State 13:       0       0       0       0
State 14:       0       0       0       0
State 15:       0       0       0       0

-----------------------------------------------
```

另外遇到的问题和解决思路（可选）：

规约错误的情况：



```
输入的文法
E->E-T
E->T
T->T/F
T->F
F->(E)
F->i
VT:
i - / ( ) #
VN:
E T F
****************************LR(1)****************************
Enter String
i+i*1
***************************Analyzer**************************
Step     StateStack      SymbolStack     RemainingString Info
1        0        #      i+i*1#          ACTION[0,i]=S5,状态5入栈
         Error
```

代码很原创（可选）：

```
#include <iostream>
#include <stack>
#include <string>
#include <cstdlib>
#include <iomanip>
using namespace std;
/*
//第一种文法:
```

```
string action[12][6] = {"S5", "0", "0", "S4", "0", "0",
                        "0", "S6", "0", "0", "0", "acc",
                        "0", "r2", "S7", "0", "r2", "r2",
                        "0", "r4", "r4", "0", "r4", "r4",
                        "S5", "0", "0", "S4", "0", "0",
                        "0", "r6", "r6", "0", "r6", "r6",
                        "S5", "0", "0", "S4", "0", "0",
                        "S5", "0", "0", "S4", "0", "0",
                        "0", "S6", "0", "0", "S11", "0",
                        "0", "r1", "S7", "0", "r1", "r1",
                        "0", "r3", "r3", "0", "r3", "r3",
                        "0", "r5", "r5", "0", "r5", "r5"};
int gotoarr[12][3] = {1, 2, 3, //GOTO 表
                      0, 0, 0,
                      0, 0, 0,
                      0, 0, 0,
                      8, 2, 3,
                      0, 0, 0,
                      0, 9, 3,
                      0, 0, 10,
                      0, 0, 0,
                      0, 0, 0,
                      0, 0, 0,
                      0, 0, 0};
char vt[6] = {'i', '+', '*', '(', ')', '#'}; //存放终结符
char vn[3] = {'E', 'T', 'F'}; //存放非终结符
string Production[6] = {"E->E+T", "E->T", "T->T*F", "T->F", "F->(E)", "F->i"};//
产生式集合
*/

//第二种文法:
string action[16][8] = {"S1", "0", "0", "0", "0", "0","S5","0",
                        "S1", "0", "0", "0", "0", "0","S5","0",
                        "0", "0", "0", "S7", "S8", "0","0","acc",
                        "0", "r7", "r7", "r7", "r7", "r7","0","r7",
                        "0", "r4", "S9", "r4", "r4", "S10","0","r4",
                        "0", "r9", "r9", "r9", "r9", "r9","0","r9",
                        "0", "S11", "0", "S7", "S8", "0","S5","0",
                        "S1", "0", "0", "0", "0", "0","S5","0",
                        "S1", "0", "0", "0", "0", "0","S5","0",
                        "S1", "0", "0", "0", "0", "0","S5","0",
                        "S1", "0", "0", "0", "0", "0","S5","0",
                        "0", "r8", "r8", "r8", "r8", "r8","0","r8",
                        "0", "r3", "S9", "r3", "r3", "S10","0","r3",
                        "0", "r2", "S9", "r2", "r2", "S10","0","r2",
                        "0", "r6", "r6", "r6", "r6", "r6","0","r6",
                        "0", "r5", "r5", "r5", "r5", "r5","0","r5"
                        };
int gotoarr[16][4] = {2,3,0,4,
                      6,3,0,4,
                      0,0,0,0,
                      0,0,0,0,
                      0,0,0,0,
                      0,0,0,0,
                      0,0,0,0,
                      0,3,0,12,
```

```cpp
                        0,3,0,13,
                        0,14,0,0,
                        0,15,0,0,
                        0,0,0,0,
                        0,0,0,0,
                        0,0,0,0,
                        0,0,0,0,
                        0,0,0,0
                      };
char vt[8] = {'(', ')', '*', '+','-', '/', 'i', '#'}; //存放终结符
char vn[4] = {'E','F', 'S', 'T'}; //存放非终结符
string Production[9] = {"S->E","E->E-T", "E->E+T","E->T", "T->T/F","T->T*F", "T->F", "F->(E)", "F->i"};//产生式集合

int com = 0;
int line = 1;//记录处理的步骤数
bool flag = false;
int StatusNumber = 1;//栈中状态数
string StackString = "#";//记录符号栈中内容
int Status[50] = {0};//记录状态栈
stack<char> Stack;//创建一个符号栈
stack<int> status;//创建一个状态栈

void Judge(int &i, int j, const char arr[], char ch, string s) {//判断输入串是否由文法终结符组成
    flag = false;
    for (int l = 0; l < j; l++) {
        if (ch == arr[l]) {
            flag = true;
            i = l;
            break;
        }
    }
    if (!flag) {
        cout << "\tError" << endl;
        com = s.size();
    }
}

void OutputStatus() {//输出状态集
    for (int i = 0; i < StatusNumber; i++)
        cout << Status[i];
}

void OutputString(string s) {//输出未处理的字符串
    for (int i = com; i < s.size(); i++)
        cout << s.at(i);
}

void Output(string s) {//输出步骤、 状态集、 符号集、 输入串
    cout << line << "\t";
    OutputStatus();
    cout << "\t" << StackString << "\t";
    OutputString(s);
    cout << "\t\t";
    line++;
```

```cpp
}

void Shift(int i, string s) {//移进函数 S
    Output(s);
    cout << "ACTION[" << status.top() << "," << s.at(com) << "]=S" << i << ",状态" << i << "入栈" << endl;
    status.push(i);//将状态 i 压进状态
    Status[StatusNumber] = i;//Status 记录状态栈的内容
    Stack.push(s.at(com));//将当前面临的输入串符号压进符号栈
    StackString = StackString + s.at(com);//StackString 记录符号栈的内容
    com++;//当前面临的输入串字符往后移一位
    StatusNumber++;//状态数加一
}

void Goto(stack<int> st1, stack<char> st2, string s) {//GoTo 语句
    int j = -1;
    int ch1 = st1.top();
    char ch2 = st2.top();
    Judge(j, 4, vn, ch2, s);//求得 ch2 在非终结符表中的位置
    if (gotoarr[ch1][j] == 0) {
        cout << "\tError" << endl;
        com = s.size();
    } else {
        status.push(gotoarr[ch1][j]);//新状态进栈
        Status[StatusNumber] = gotoarr[ch1][j];
        StatusNumber++;
    }
}

void Reduction(int i, string s) {//归约函数 R
    Output(s);
    cout << "r" << i << ":" << Production[i - 1] << "归约, GoTo(";
    int N = Production[i - 1].length() - 3;
    for (int j = 0; j < N; j++) {//消除要归约的状态及符号
        status.pop();
        Stack.pop();
        StatusNumber--;
        StackString.erase(StackString.length() - 1);
    }
    cout << status.top() << "," << Production[i - 1].at(0) << ")=";
    Stack.push(Production[i - 1].at(0));//符号进栈
    StackString = StackString + Stack.top();
    Goto(status, Stack, s);
    cout << status.top() << "入栈" << endl;
    Status[StatusNumber] = status.top();
}

void Analyse(string s) {//具体分析函数
    Stack.push('#');//初始化
    status.push(0);
    s = s + "#";
    int t = -1;//记录 ch 在数组 vt 的位置
    while (com < s.size()) {
        int i = status.top();
        char ch = s.at(com);
        Judge(t, 8, vt, ch, s);
        if (flag) {
            if (action[i][t] != "acc" && action[i][t] != "0") {
                if (action[i][t].at(0) == 'S') {
                    action[i][t].erase(0, 1); //删除 action[i][t]的首字母 S
                    Shift(atoi(action[i][t].c_str()), s);//atoi(action[i][t].c_str()), 将action[i][t]转换为整型
                    action[i][t].insert(0, "S");//将 S 添加回 action[i][t]
                } else if (action[i][t].at(0) == 'r') {
                    action[i][t].erase(0, 1);//删除 action[i][t]的首字母 r
                    Reduction(atoi(action[i][t].c_str()), s);//atoi(action[i][t].c_str()), 将action[i][t]转换为整型
                    action[i][t].insert(0, "r");//将 r 添加回 action[i][t]
                }
            } else if (action[i][t] == "0") {
                cout << "\tError" << endl;
                break;
            } else if (action[i][t] == "acc") {
                Output(s);
                cout << "acc" << "\t 分析成功" << endl;
                break;
            }
        } else if (!flag)
            break;
    }
}

void printTable() {
    cout << "\n------------------ Action Table -----------------\n" << endl;
    for(int i = 0; i < 16; i++){
        if(i < 10) {
            cout << "State " << i << ": ";
        } else {
            cout << "State " << i << ":";
        }
        for(int j = 0; j < 8; j++){
            cout << setw(10) << action[i][j] << " ";
        }
        cout << "\n";
    }
    cout << "\n------------------ Goto Table ------------------\n" << endl;
    for(int i = 0; i < 16; i++){
        if(i < 10) {
            cout << "State " << i << ": ";
        } else {
            cout << "State " << i << ":";
        }
        for(int j = 0; j < 4; j++){
            cout << setw(10) << gotoarr[i][j] << " ";
        }
        cout << "\n";
    }

    cout << "\n------------------------------------------------\n" << endl;
}

int main() {
    string s;
    cout << "输入的文法" << endl;
    for (int j = 0; j < 9; j++)
        cout << Production[j] << endl;
    cout << "VT:" << endl;
    for (int i = 0; i < 8; i++)
        cout << vt[i] << " ";
    cout << endl;
    cout << "VN:" << endl;
    for (int i = 0; i < 4; i++)
        cout << vn[i] << " ";
    cout << endl;
    cout << "****************************LR(1)****************************" << endl;
    char T;
    cout << "Enter String" << endl;
    cin >> s;
    cout << "**************************Analyzer***************************" << endl;
    cout << "Step" << "\t" << "StateStack" << "\t" << "SymbolStack" << "\t" << "RemainingString" << "\t" << "Info"
         << endl;
    Analyse(s);
    com = 0;
    line = 1;
    StackString = "#";
    StatusNumber = 1;
    while (!Stack.empty())
        Stack.pop();
    while (!status.empty())
        status.pop();

    printTable();  // 调用打印函数

    return 0;
}

```