**LR语法分析器实验报告**

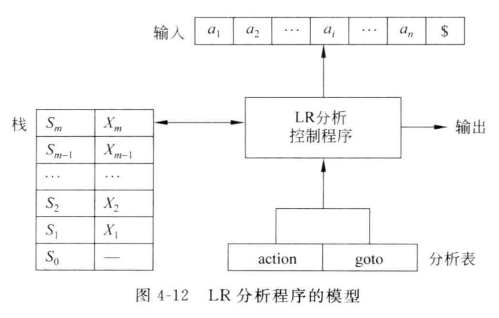
**付彬 2017211307班 2017211357**

1. **运行/开发环境及版本号**

OS：Windows 10

IDE：Visual Studio 2019

1. **LR语法分析器设计**
2. **语法分析器模型**

****

1. **语法分析器模型具体实现**
2. **输入缓冲区**

****

1. **状态栈和符号栈**

****

1. **LR分析控制程序**

****

1. **分析表**

****

1. **文法相关信息**
2. **原始文法**

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

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

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

1. **拓广文法**

**S -> E**

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

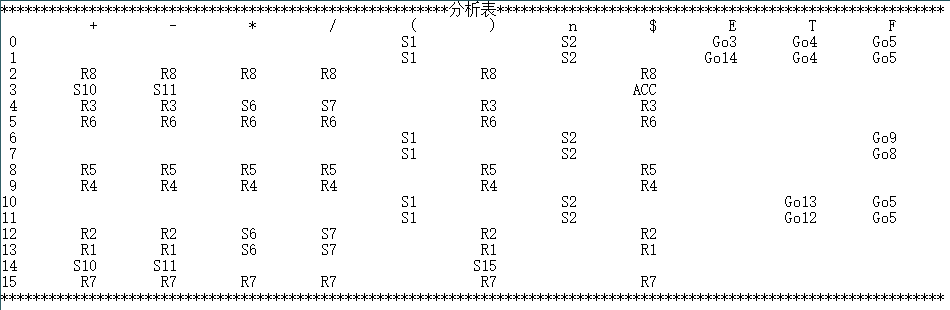
**T -> T\*F | T/F | F**

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

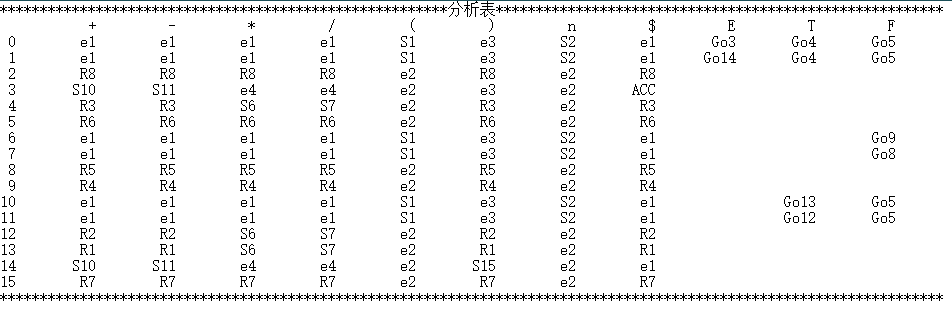
1. **First集和Follow集**

|  |  |  |
| --- | --- | --- |
|  | first | follow |
| S | ( , num | $ |
| E | ( , num | $ , ) , + , - |
| T | ( , num | $ , ) , + , - , \* , / |
| F | ( , num | $ , ) , + , - , \* , / |

1. **分析表（由程序打印得）**

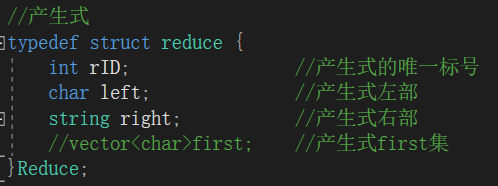


**填充了错误处理信息的分析表：**



1. **实现细节**
2. **数据结构**
3. **产生式**

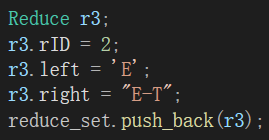
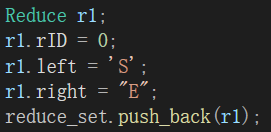
**产生式的结构体定义：**

****

**用vector作为产生式集合的容器：**

****

**将产生式初始化置入vector：**

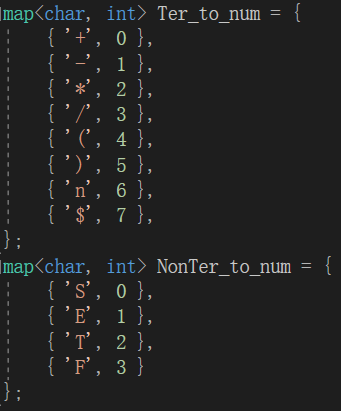
****

1. **终结符和非终结符**

**文法符号的存储：**

****

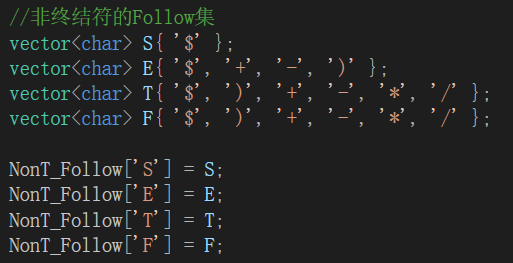
**实现了文法符号到数字的映射，在编程时，利于按文法符号的索引访问其他数据结构：**

****

1. **非终结符Follow集**

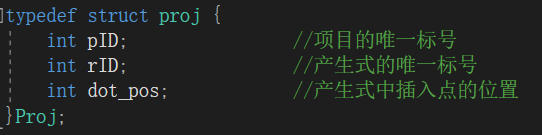
**使用map存储：**

****

**初始化：  
**

1. **项目**

**结构体定义：**

****

**使用了int值存储该项目的产生式的标号。**

**使用了int值存储圆点在产生式中的位置。**

**并定义了圆点在结尾时规约项目的表示方式：**

****

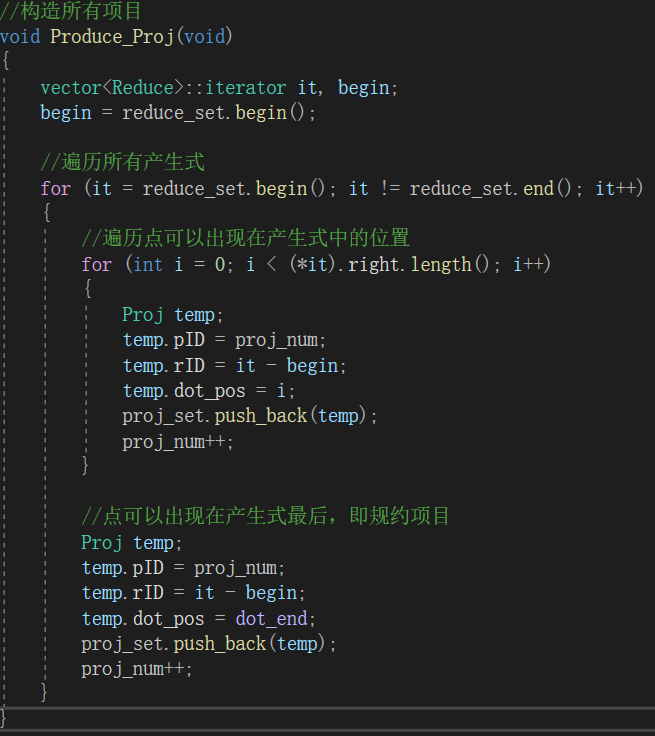
**例如：**

**S -> .E的圆点位置值为0**

**S -> E.的圆点位置值为-1**

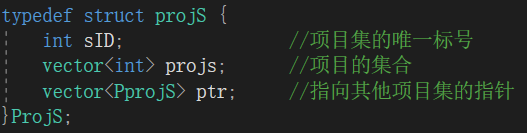
**使用下列函数，构造出所有的项目，放入项目的集合，项目的集合用vector来存储：**

****

****

1. **项目集**

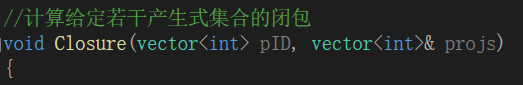
**结构体定义：**

****

**项目的集合中包含了该项目集中的所有项目。**

**项目集中的跳转信息由ptr完成，见下面的解释。**

**项目集的自动构造方式是：给定若干初始项目pID，计算出他们的闭包，得到一个完整项目，然后写入参数projs：**

****

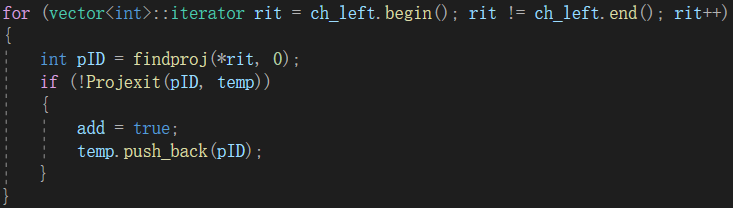
**具体思路是，用一个bool数组代表，以所有非终结符开头的且圆点在开头位置的产生式是否全部加入到了项目集中，**

****

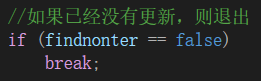
**然后不断的循环，遍历已有的产生式，如果有产生式中圆点的后面是非终结符，并且在上述数组中存的值为false，则需要将以这个**

****

**非终结符开头的所有圆点在最开头的产生式全部加入项目集。**

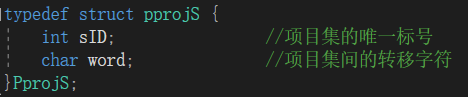
****

**如果循环中不再更新项目集，则退出：**

****

1. **DFA状态跳转**

**结构体定义：**

****

**一个sID代表跳转到哪个项目集，对应的word代表跳转所对应的终结符或非终结符。**

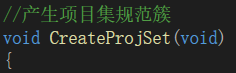
**一个项目集有多路跳转的话，会对应多个上述的跳转，以vector形式存储。**

1. **项目集规范簇**

**以vector形式存储：**

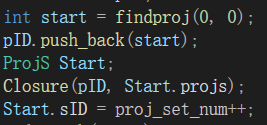
****

**由函数自动构造：**



**具体的思路为：**

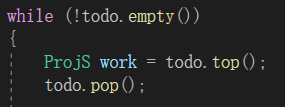
1. **用S -> .E作为初始产生式，产生它的闭包，将这个项目集加入项目集规范簇以及工作栈**

****

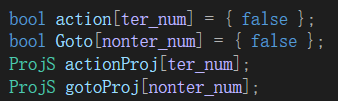
1. **工作栈用来存储，所有已经产生的项目集，但是他们跳转到其他项目集的连接还没有处理**

****

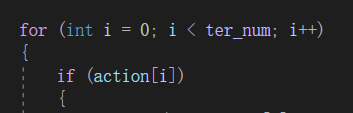
1. **进入循环，每次从todo栈中取出一个项目集，为其建立跳转**

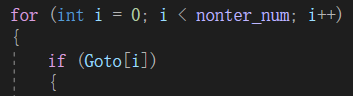
****

1. **记录该项目集中原点后面所跟的所有符号和产生式序号**

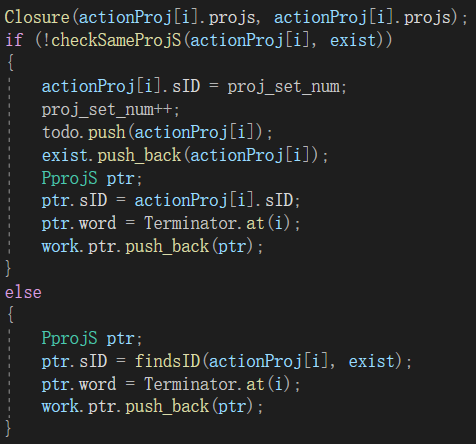
****

1. **分别遍历该产生式中原点后面可能跟的终结符号和非终结符号**

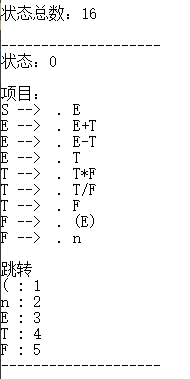
****

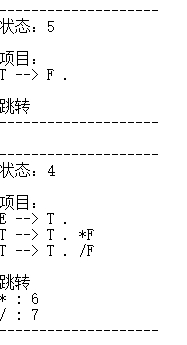
****

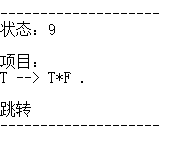
1. **遍历中，给对应的产生式计算闭包，得到新项目集，检查项目集是否已经存在，若不存在，则加入待处理列表，并且建立DFA跳转**

****

**自动生成的项目集规范簇：**

****

****

****

****

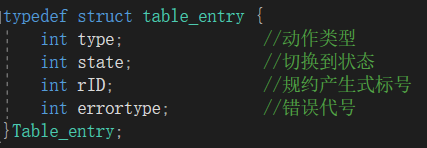
1. **预测分析表**

**分析表存储：**

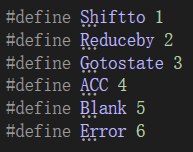
****

**访问预测分析表时，利用提到的文法符号到数字的映射填入矩阵的行下标和列下标，分析表在构建时也是按照下图终结符号和非终结符号的顺序填入产生式的。**

**表项定义：**

****

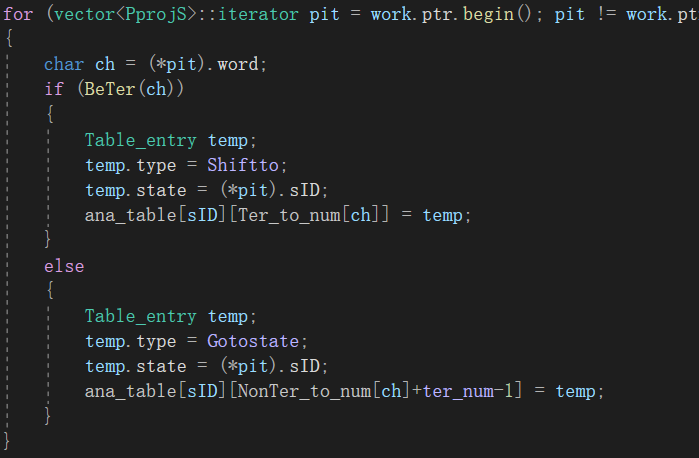
**动作类型：**

****

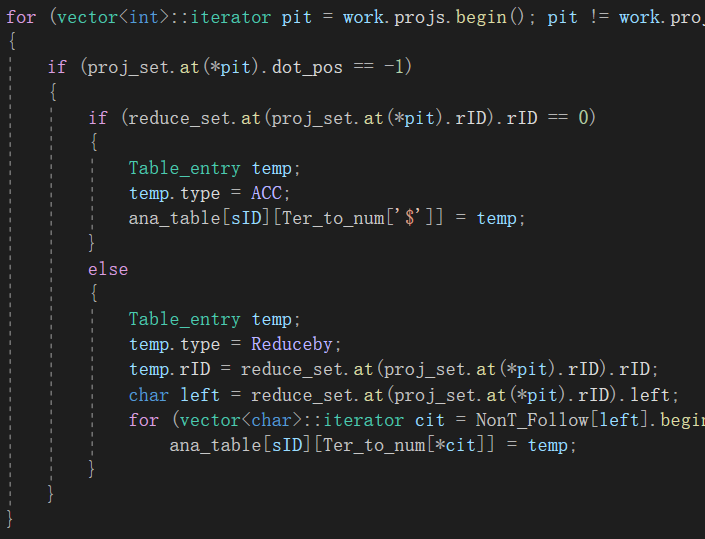
**分析表自动构建：**

****

**填入Shift和Goto部分（遍历跳转信息）：**

****

**填入Reduce和ACC部分（遍历产生式）：**

****

1. **分析栈**

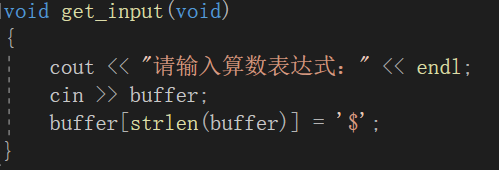
**分析栈存储：**

****

1. **输入缓存区**

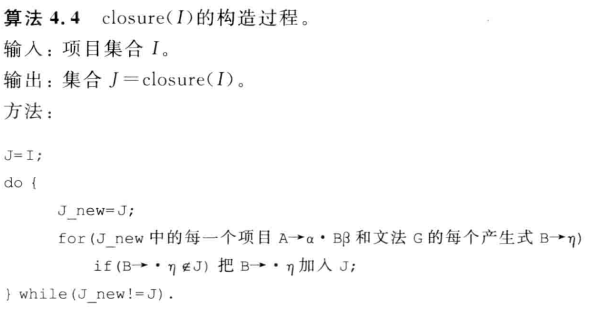
****

**在输入字符串的末尾添加$符号：**

****

1. **主要功能**
2. **自动构造若干项目的闭包**

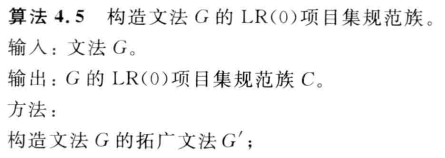
**书中算法：**

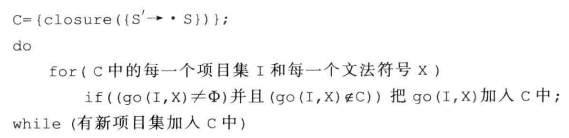
****

**本程序实现了上述算法，Closure函数在ProjectSet.cpp文件中。**

1. **自动构造SLR（1）项目集规范簇**

**书中算法：**

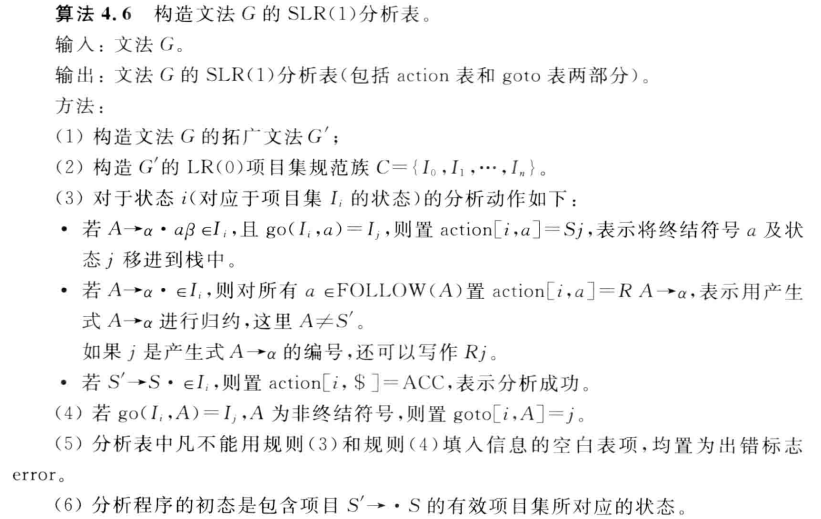
****

****

**本程序实现了上述算法，CreateProjSet函数在ProjectSet.cpp文件中。**

1. **自动构造SLR（1）语法分析表**

**书中算法：**

****

**create\_ana\_table函数在ana\_table.cpp文件中.**

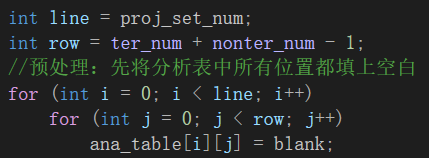
**本程序在实现书中算法的基础上，还增添了附加功能：**

1. **预先将分析表填满空白**

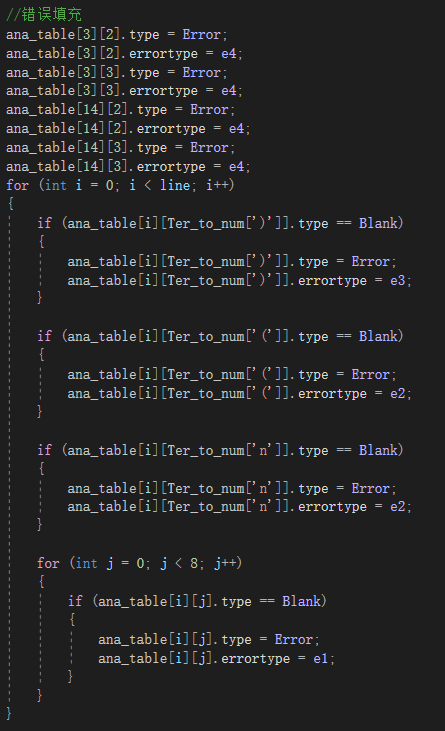
****

****

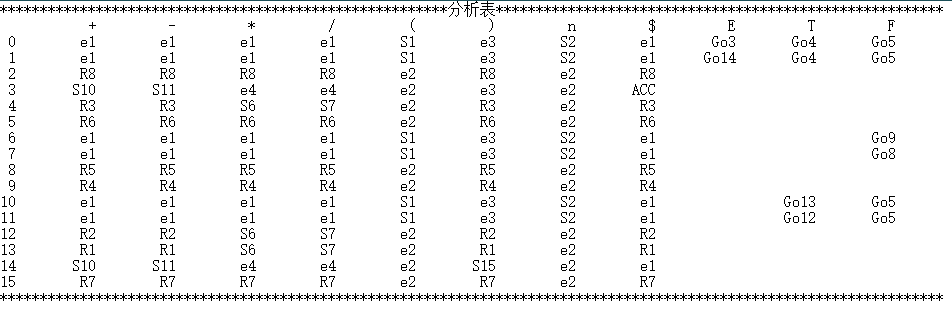
****

****

1. **填入错误处理信息，方便语法分析**

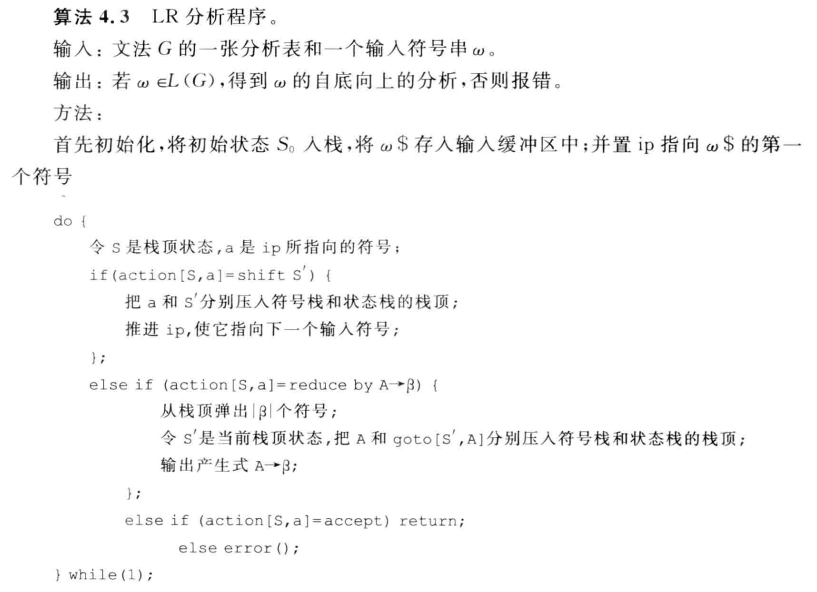
****

1. **最终结果**



1. **对给定文法进行语法分析**

**书中算法：**

****

**本程序在实现书中算法的基础上，还增添了附加功能：主要是丰富上图中error()函数中的内容。增添了更多错误处理功能，见下面。**

1. **检测语法错误，且能够报告出一些语法错误，用于提示用户更正**

**错误处理方式：打印错误原因，停止分析。**

**错误类别：**

**设置了下面4类错误：**

缺少运算对象

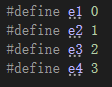
缺少运算符号

括号不匹配

其他错误

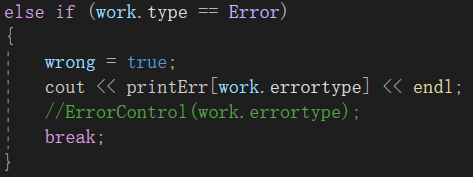


**分析表中的错误标号：**



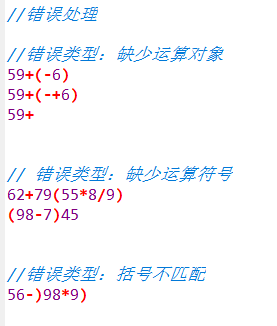
**如下图，错误处理程序会在不同的情况下输出这4类错误信息，用于提示用户，经过测试，在大部分情况下，错误提示能够正确的提示出用户输入错在了哪里，可以作为一定的参考依据。**

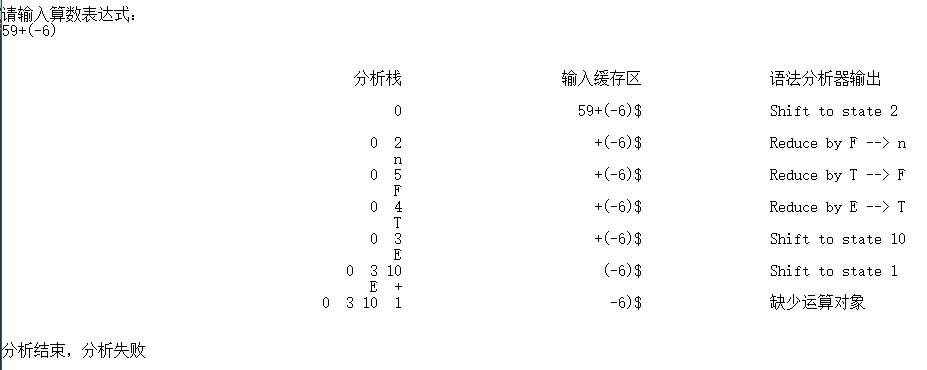
**打印错误信息的代码：**

****

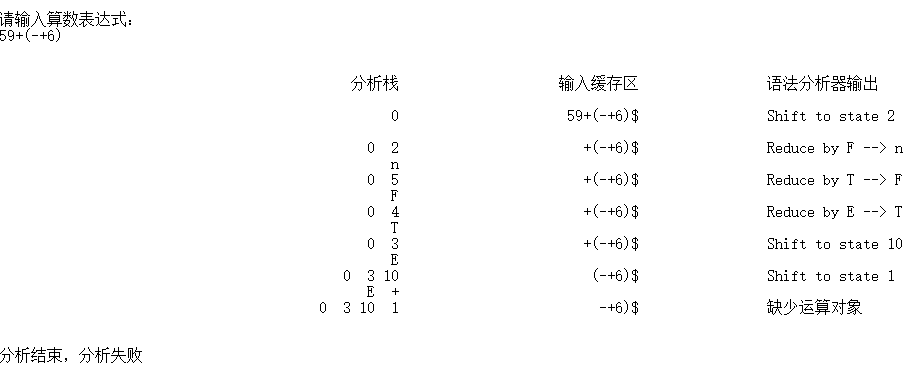
**下面是错误处理的截图：**

**样例：**

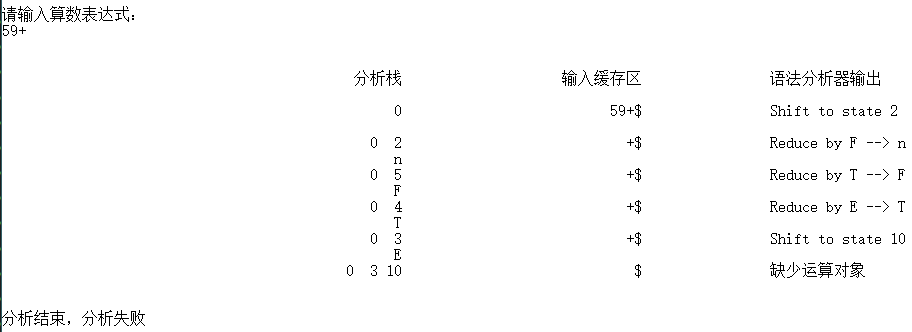
****

**59+(-6)：缺少运算对象**

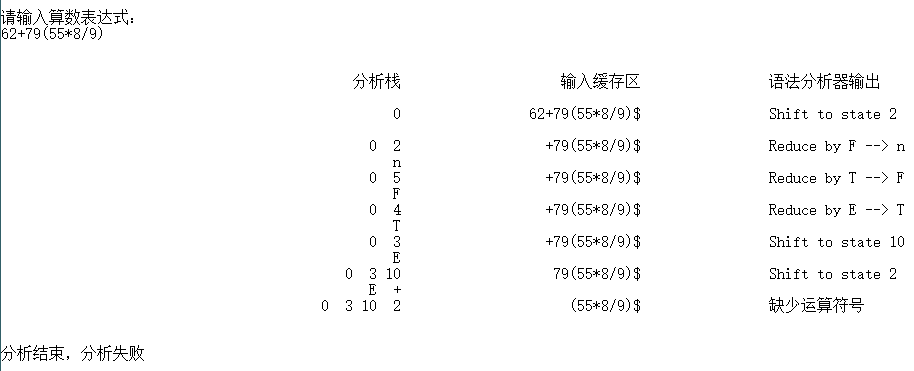
**59+(-+6)：缺少运算对象**

****

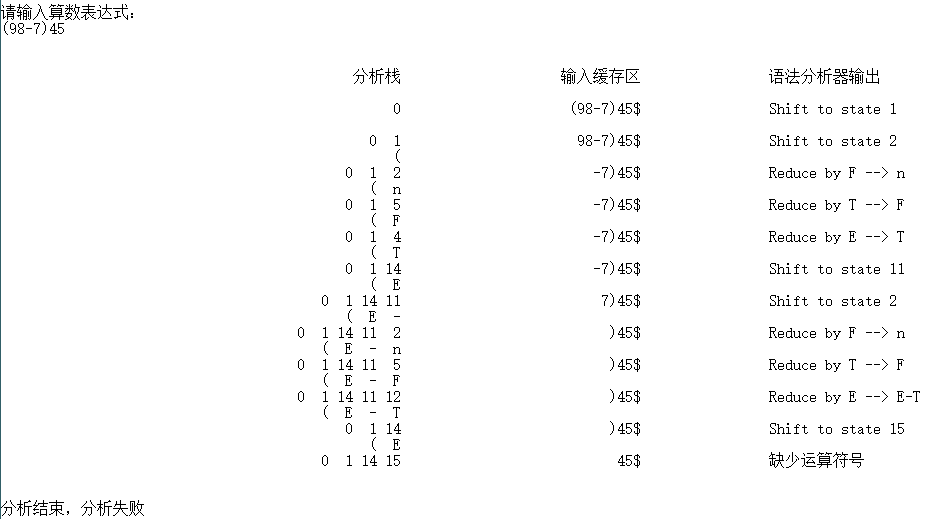
**59+：缺少运算对象**

****

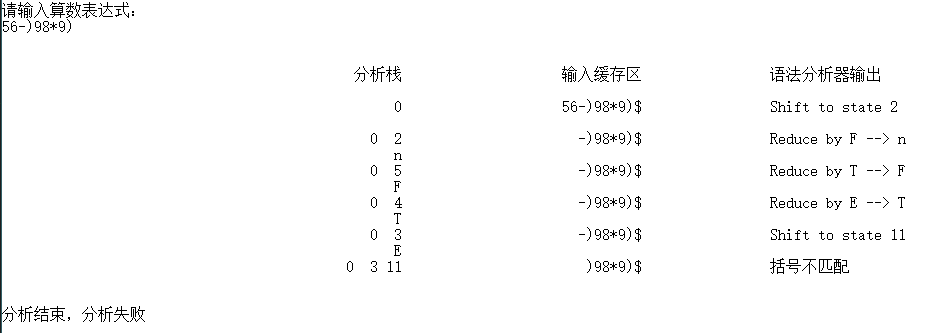
**62+79(55\*8/9)：缺少运算符号**

****

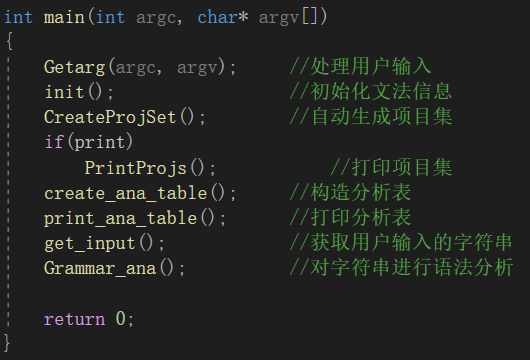
**(98-7)45：缺少运算符号**

****

**56-)98\*9)：括号不匹配**

****

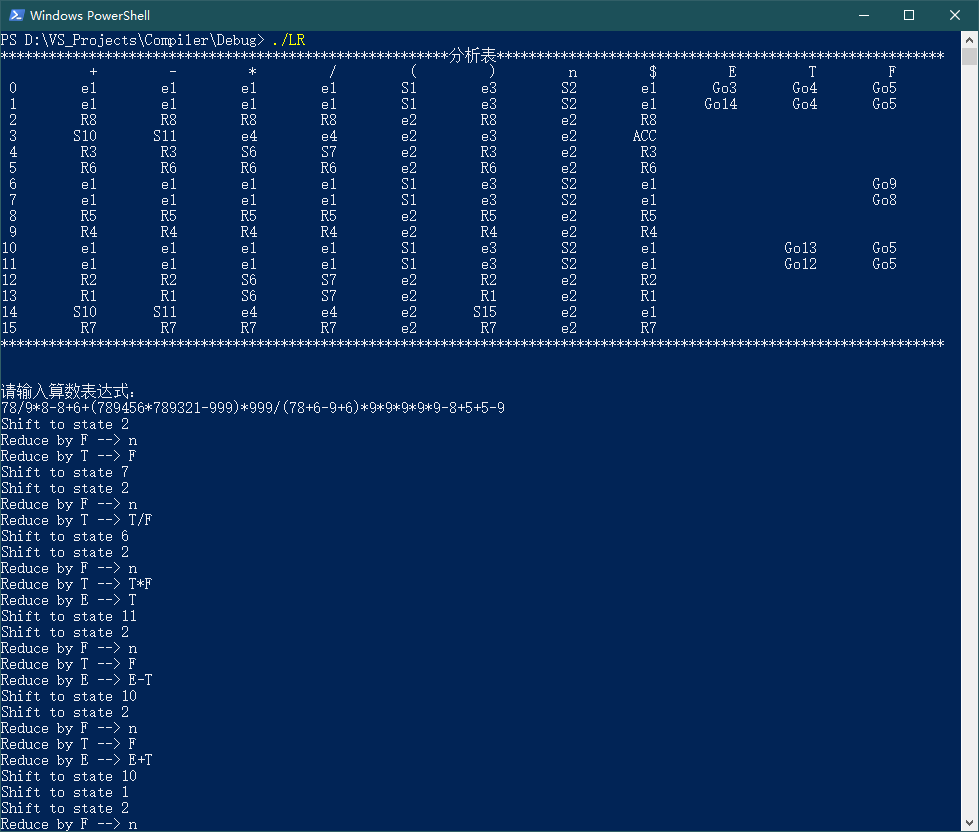
1. **程序结构**

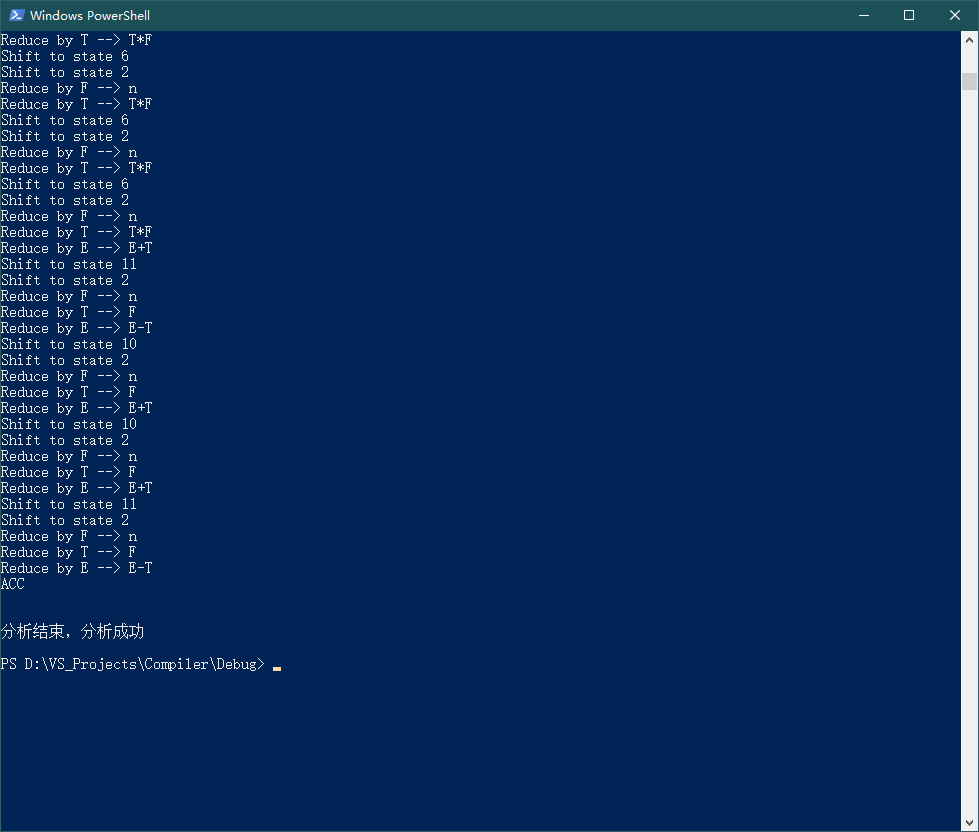
****

1. **使用说明**
2. **执行方式：**

**建议使用命令行模式执行该程序：**

****

****

****

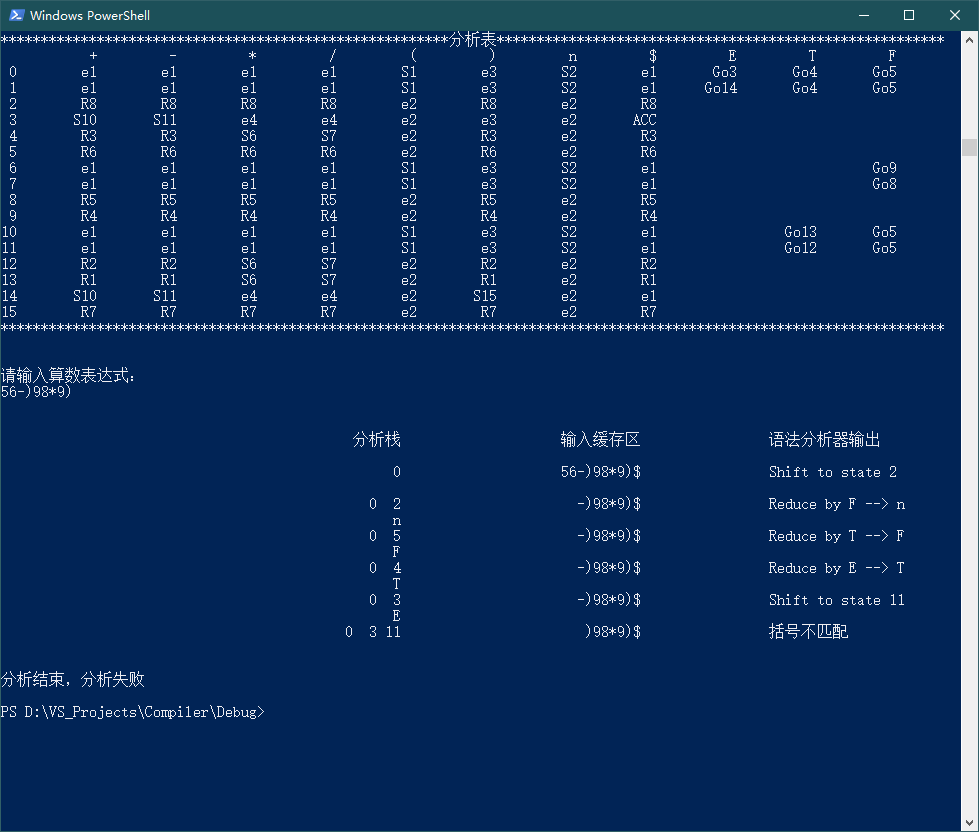
1. **打印详细分析过程**

**本程序设置了打印详细分析过程的模式，用户在命令行执行该程序可执行文件时，可以输入 –d，以打印详细信息，如图：**

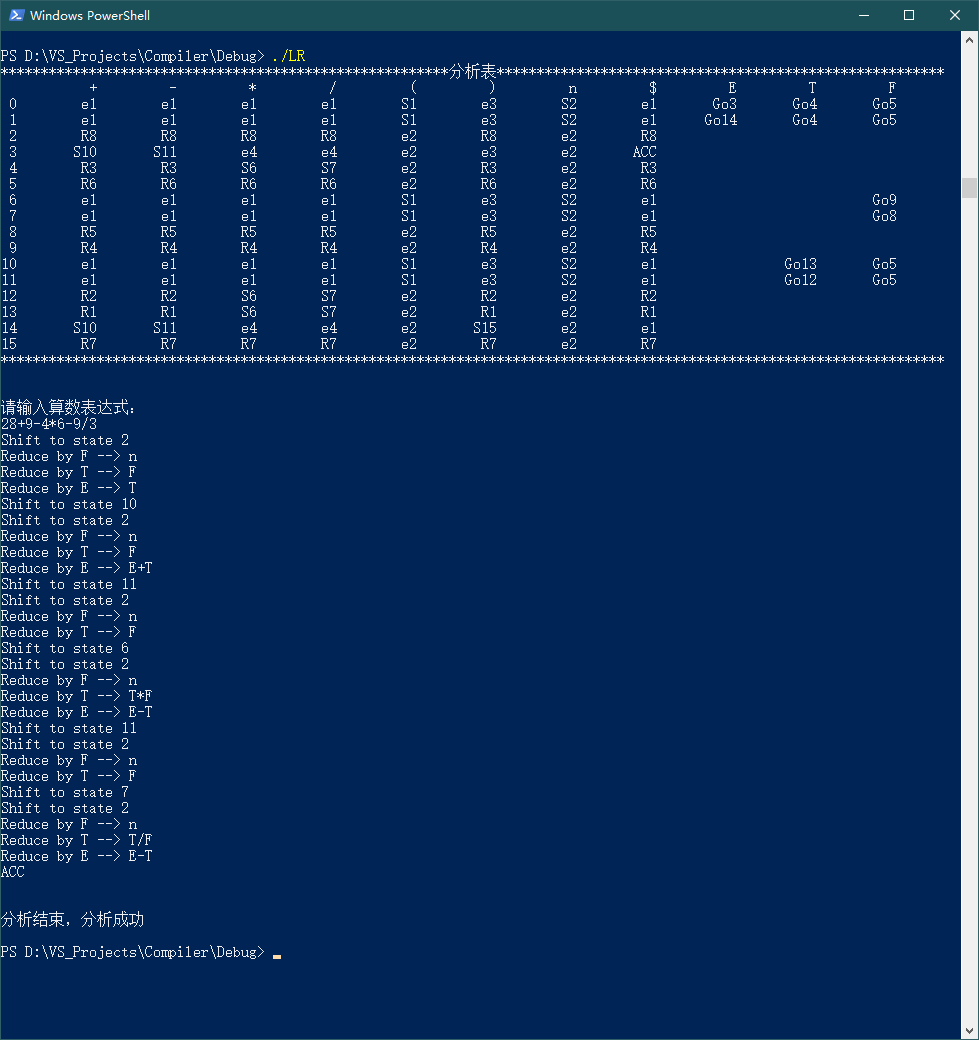
****

**注意：下图这种方式能够详细的输出分析栈、符号栈和输入缓冲区的实时信息，但是由于打印格式的原因，仅支持输入运算表达式长度为30个字符以内，过长的字符串会导致输出异常（格式上的问题）。**

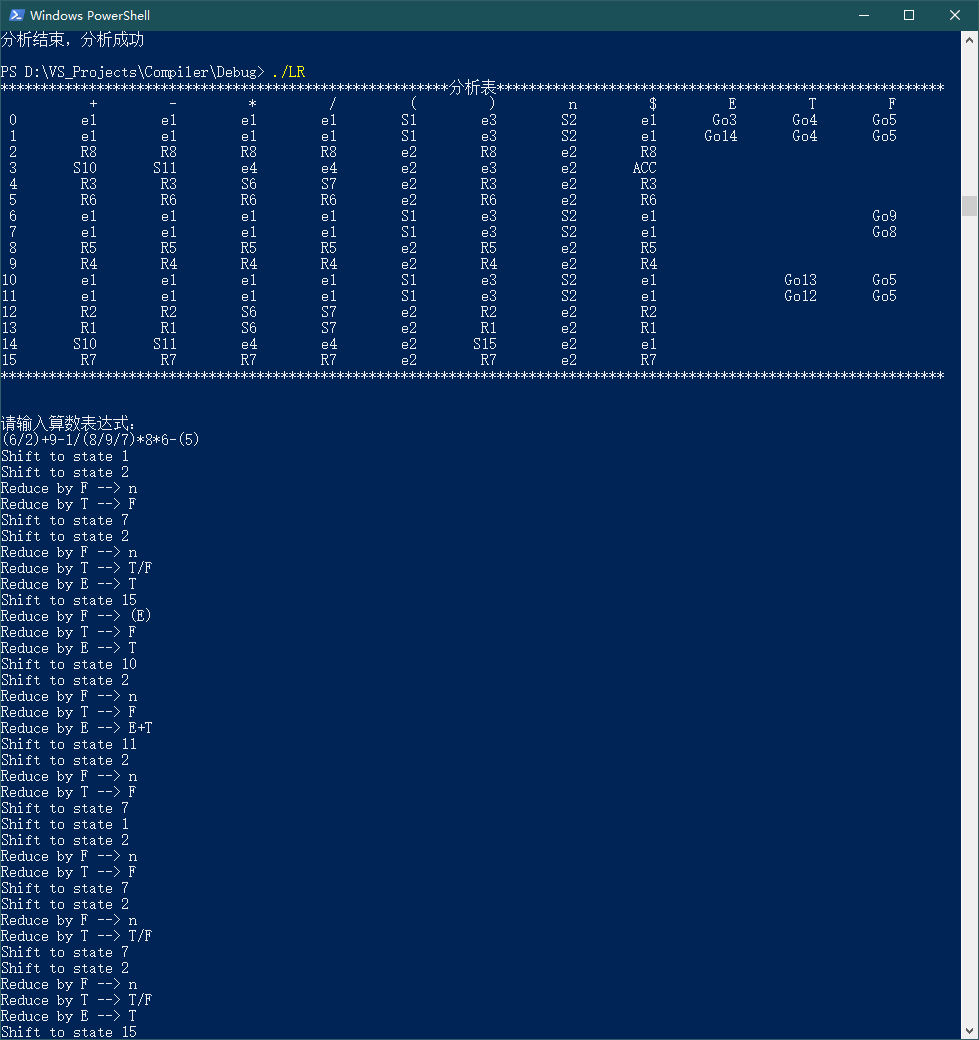
**如果要测试超长的运算表达式请勿选择这一模式！**

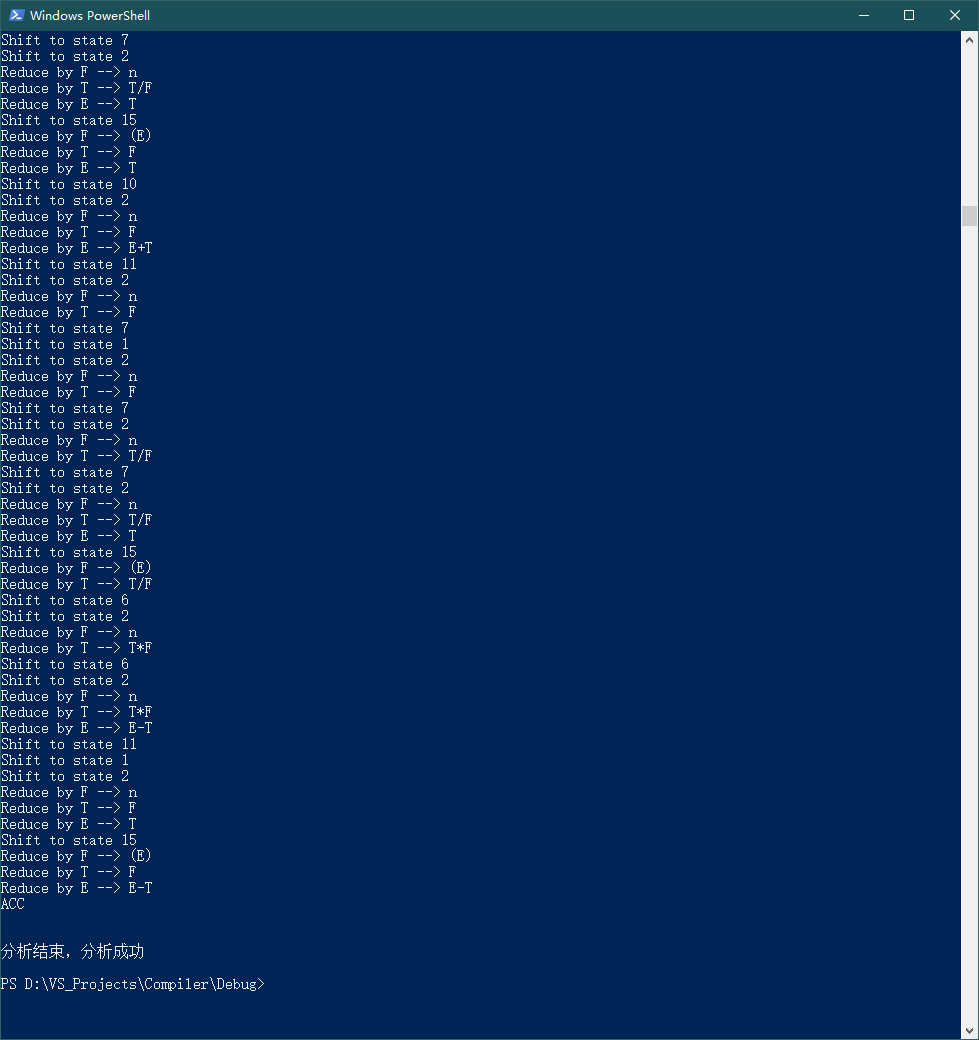
****

1. **运行截图**
2. **普通模式：28+9-4\*6-9/3**

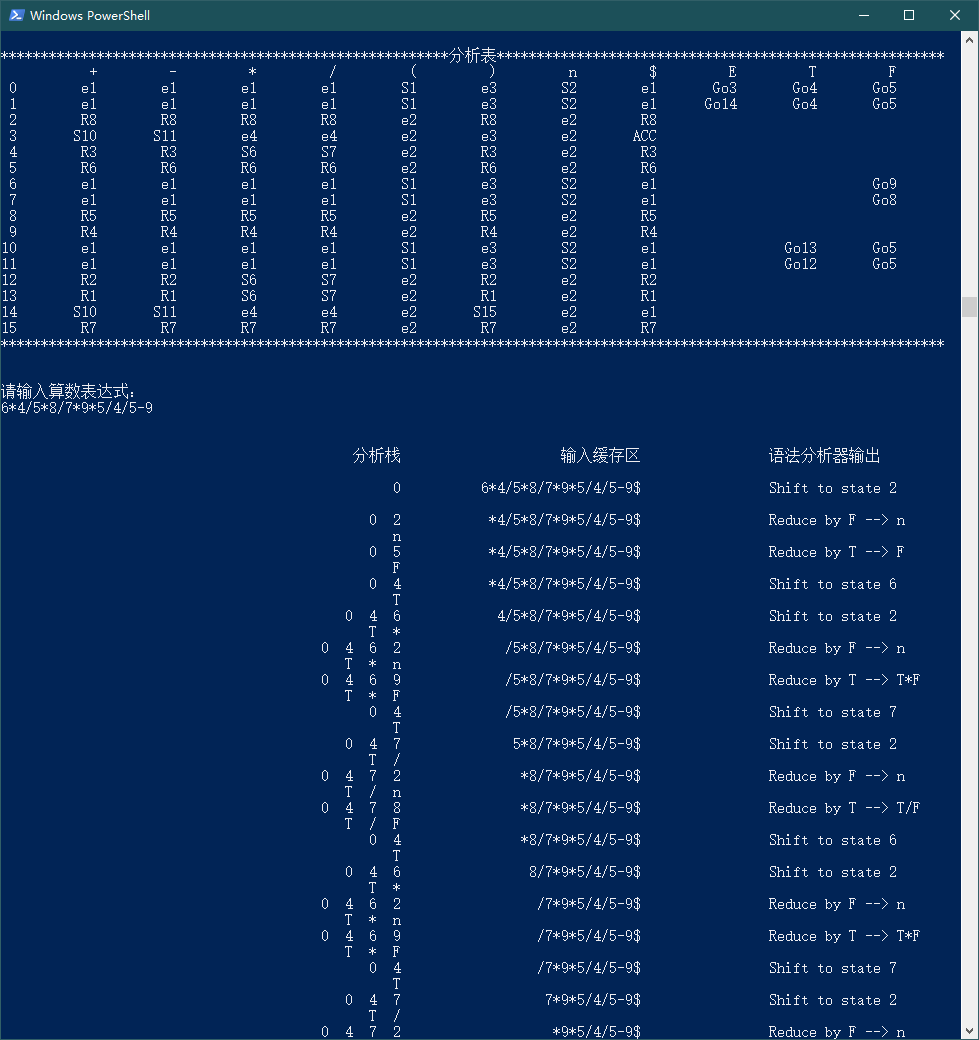


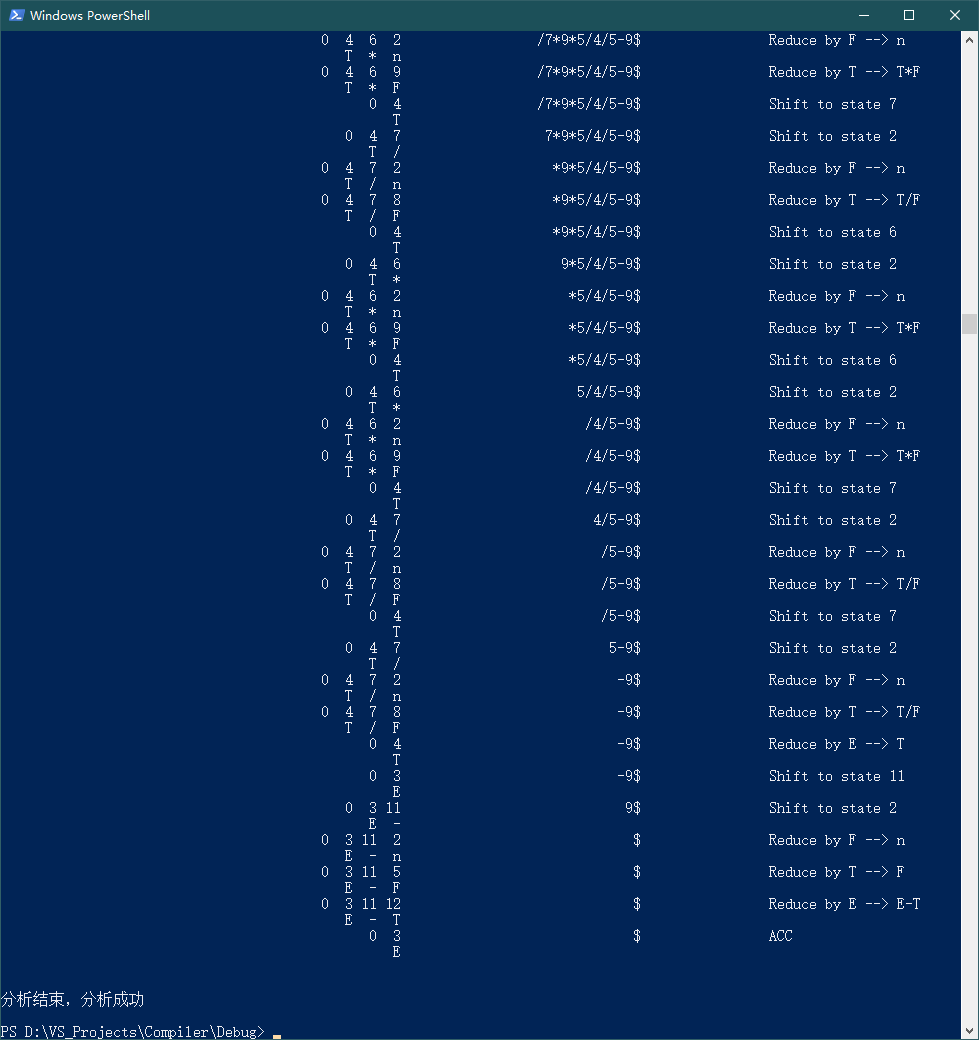
1. **普通模式：(6/2)+9-1/(8/9/7)\*8\*6-(5)**

****

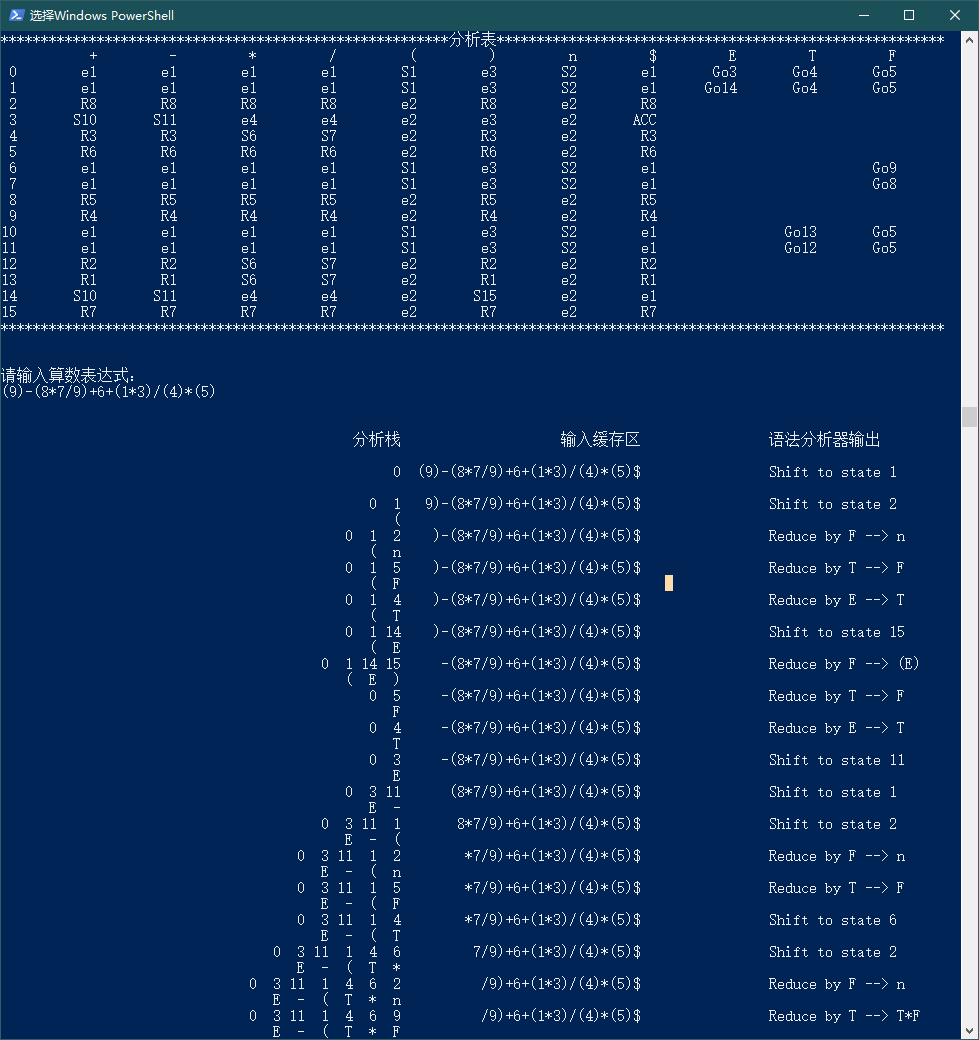
****

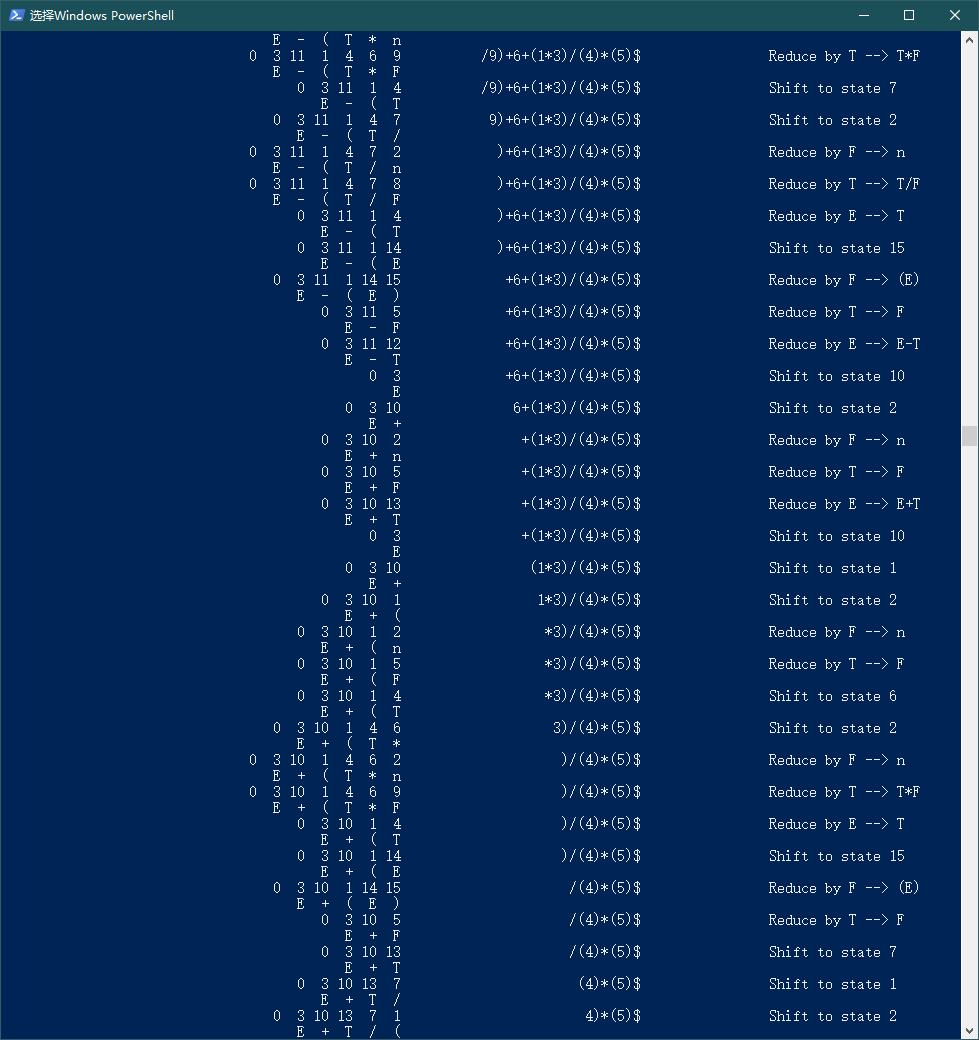
1. **详细分析过程模式：6\*4/5\*8/7\*9\*5/4/5-9**

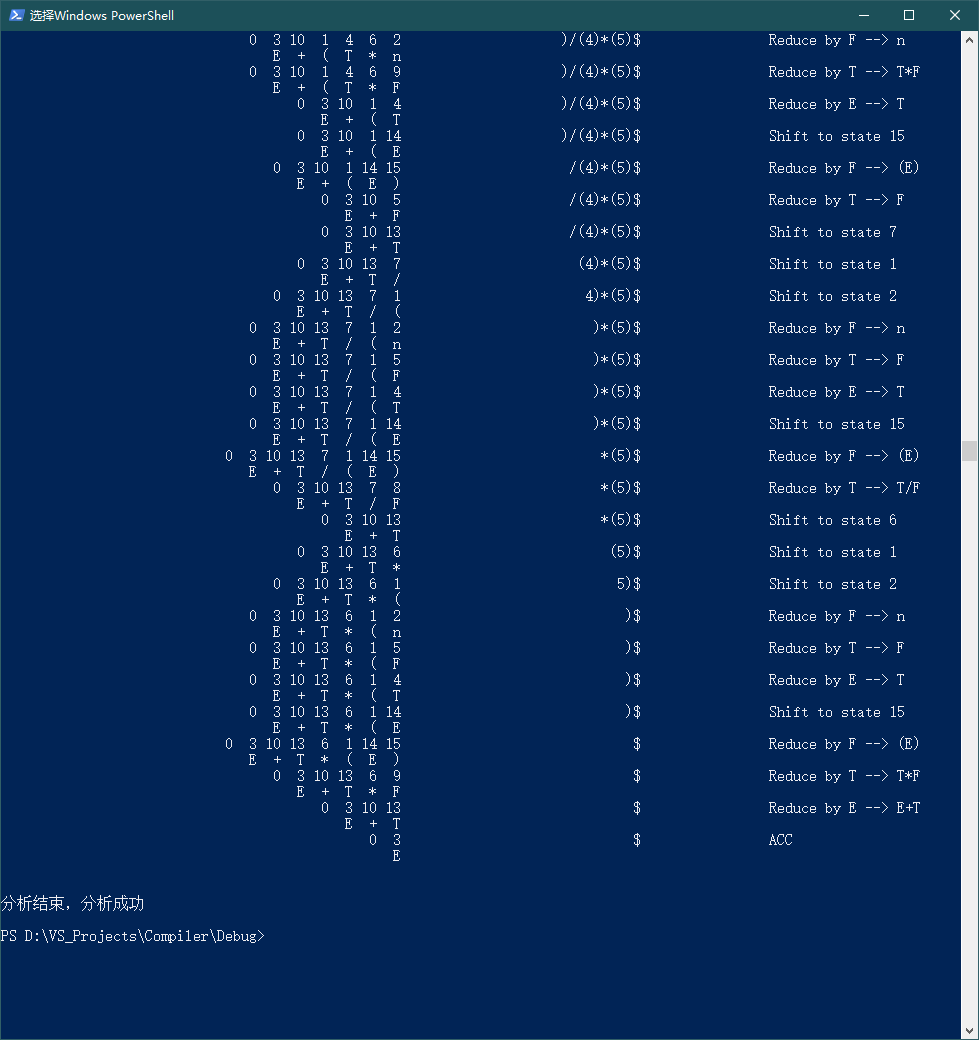
****

****

1. **详细分析过程模式：(9)-(8\*7/9)+6+(1\*3)/(4)\*(5)**

****

****

****

1. **详细分析过程模式：(9\*8/563)(89-5+7)/**

