**北京邮电大学计算机学院软件工程专业**

**2020-2021 学年第 1 学期项目总结报告**

**课程名称： 编译原理与技术**

**项目名称： 语法分析程序的设计与实现**

**项目完成人：**

**负责人姓名： 张康 学号： 2018211930**

**姓名： 徐龙 学号： 2018522108**

**姓名： 乔中沛 学号： 2018211871**

**姓名：** **田豪鹏** **学号： 2018211872**

**姓名： 赵君逸 学号： 2018211874**

**指导教师： 修佳鹏**

**日 期： 2020 年 12月 24日**

# 项目目的和要求

1. **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，要求所分析的算术表达式由如下文法产生：

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

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

F->(E)|num

**方法一**：编写递归调用程序实现自顶向下的分析。

**方法二**：编写LL（1）语法分析程序，要求如下：

1. 编程实现算法4.2，为给定的文法自动构造预测分析表。
2. 编程实现算法4.1，构造LL（1）预测分析程序。

**方法三**：编写语法分析程序实现自底向上的分析，要求如下：

1. 构造识别该文法的所有活前缀的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。

**方法四**：利用YACC自动生成语法分析程序，调用LEX自动生成的词法分析程序。

1. **实现要求**

至少采用两种方法实现该语法分析器。

要求在输入的算术表达式进行分析过程中，依次输出所采用的产生式。

# 项目开发环境

1. **高级编程语言部分**

Python 3.7, PyCharm 2020.1

JDK 1.8、 IntelliJ IDEA 2020

1. **Yacc部分**

操作系统：Windows10 1909

开发环境：msys2套件、mingw64、g++、flex、bison

# 项目内容

对本实验的四种方式都进行了实现。小组大致分工如下

|  |  |
| --- | --- |
| **内容** | **负责** |
| 方法一：递归调用下降 | 赵君逸 |
| 方法二：LL(1)文法 | 田豪鹏 |
| 方法三：LR文法 | 张康、乔中沛 |
| 方法四：YACC实现 | 徐龙 |

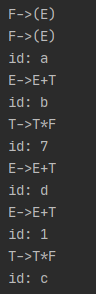
# 项目结果及分析

1. **递归调用程序实现结果**

输入表达式为：



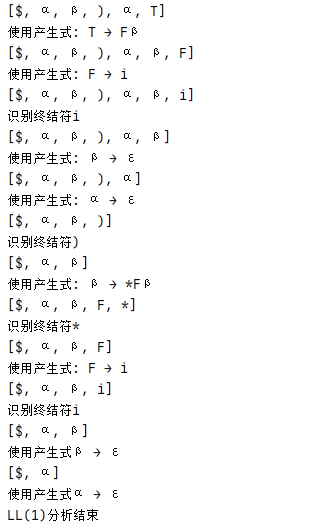
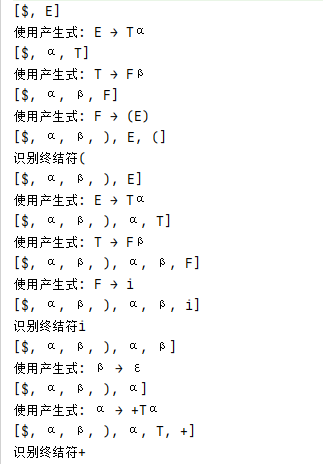
语法分析结果如下图：



1. **LL（1）语法分析程序实现结果**

输入表达式为：(i+i)\*i

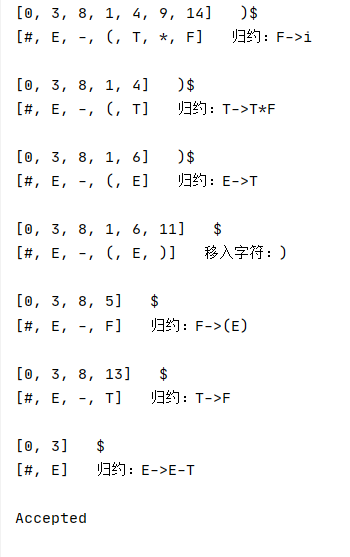
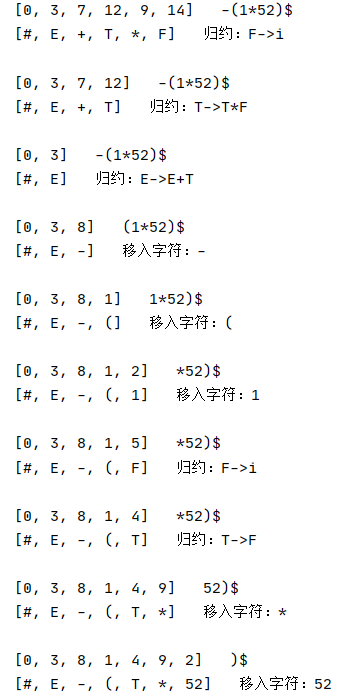
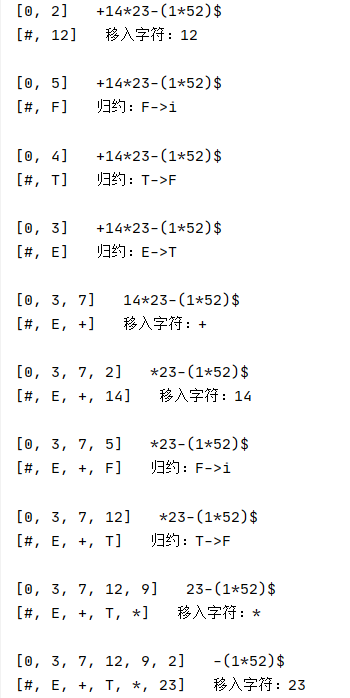
识别过程为：



这里的i代表整数，同样可以用任意正整数代替。

1. **SLR语法分析程序实现结果**

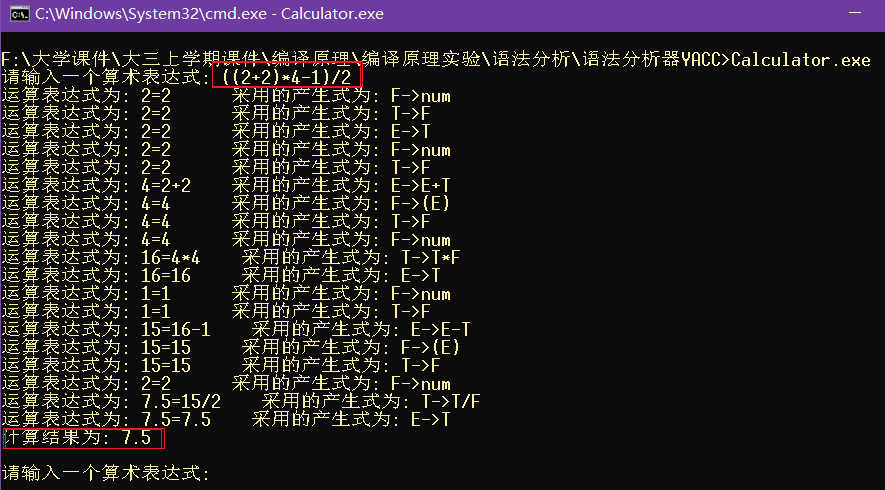
输入为12+14\*23-(1\*25)



每一块都是一条记录，左上角是状态栈，左下角为符号栈，右上角为待读入字符，右下角为当前操作。

1. **Yacc实现结果**

使用flex和bison将编写的Calculator.l及Calculator.y文件编译成C源文件，之后是由g++将两个C源文件共同编译，运行.exe文件后输入一个算术表达式，语法分析器运算结果如下图：



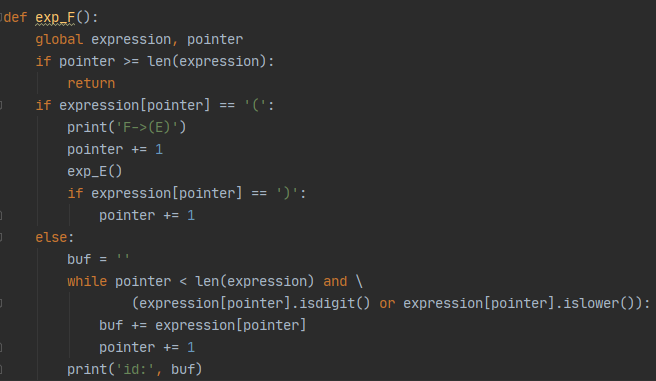
由输出的产生式和计算结果可以判断出，该计算结果正确。本语法分析器在分析算术表达式时会使用的产生式，并把计算结果输出，用户输入的算术表达式中的数可以为整数，也可以为浮点数。

# 程序实现方法讲解

1. **递归调用程序实现**

该程序使用Python进行编写。Exp\_E()负责+-的识别，Exp\_T()负责\*/的识别，Exp\_F()负责()的识别。在算术表达式中运算符优先级为()>\*/>+-，所以由Exp\_E调用Exp\_T，Exp\_T调用Exp\_F，Exp\_F在语法分析中最先执行。

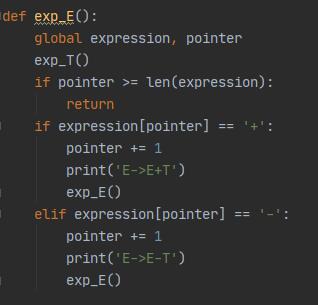
1. **Exp\_F部分**：



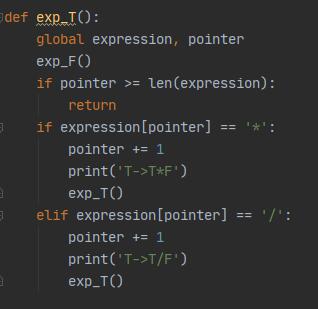
当pointer超过表达式长度时说明表达式已分析完成，直接返回；

当pointer指向’(’时，打印出产生式，调用Exp\_E继续分析；Exp\_E递归调用回到Exp\_F，进入else分支分析非运算符部分，当待分析的字符是数字或小写字母时，将其作为id打印出来

1. **Exp\_E部分:**



1. **Exp\_T部分:**



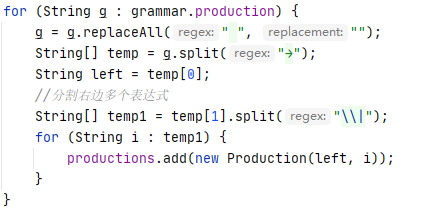
Exp\_E与Exp\_T的逻辑与功能均类似，Exp\_E是用于分析+-符号，Exp\_T用于分析\*/符号，因为+-符的优先级要低于\*/符，为Exp\_E调用Exp\_T。

1. **LL（1）语法分析程序实现**
2. **设定文法：**

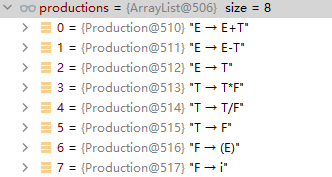
* 非终结符包括"E", "T", "F"，其中E为起始符号；
* 终结符包括"+", "\*", "-", "/", "(", ")", "i", "$"；
* 产生式包括E → E+T | E-T | T，T → T\*F | T/F |F，F → (E) | i。

1. **获取产生式组**

以"|"为记号标识产生式的右部，并进行分割：



得到以下产生式：



1. **消除文法中的左递归：**

消除左递归时，要考虑间接左递归和直接左递归。本程序中，先对间接左递归进行消除转换，然后消除直接左递归。

消除间接左递归：

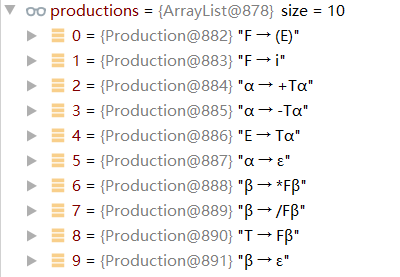


消除直接左递归：

为了避免符号的冲突，本程序中，引入的新非终结符以希腊字母表示。



经过消除左递归后，得到的产生式集如下所示：



1. 消除回溯

为了在分析过程中避免回溯，要通过求First集、Follow集的方法，检查同一非终结符的不同产生式之间是否存在冲突，来避免在遇到同一终结符时，有多个产生式可供匹配的情况。对于冲突的产生式，引入新的产生式组，来避免回溯问题。

求First集的方法参照教材给出的算法，对于给定的串，可以给出其First集，考虑了产生式右部为空串、非空串的情况：



求Follow集的方法参照教材，对于文法符号在末尾的情况，将产生式左部文法符号的Follow集添加到Follow集中；如果不在末尾，则将其后的文法符号串的First集非空元素添加到Follow集中。



通过增加新的非终结符和产生式的方法，消除回溯，使得每一步分析都是确定的：



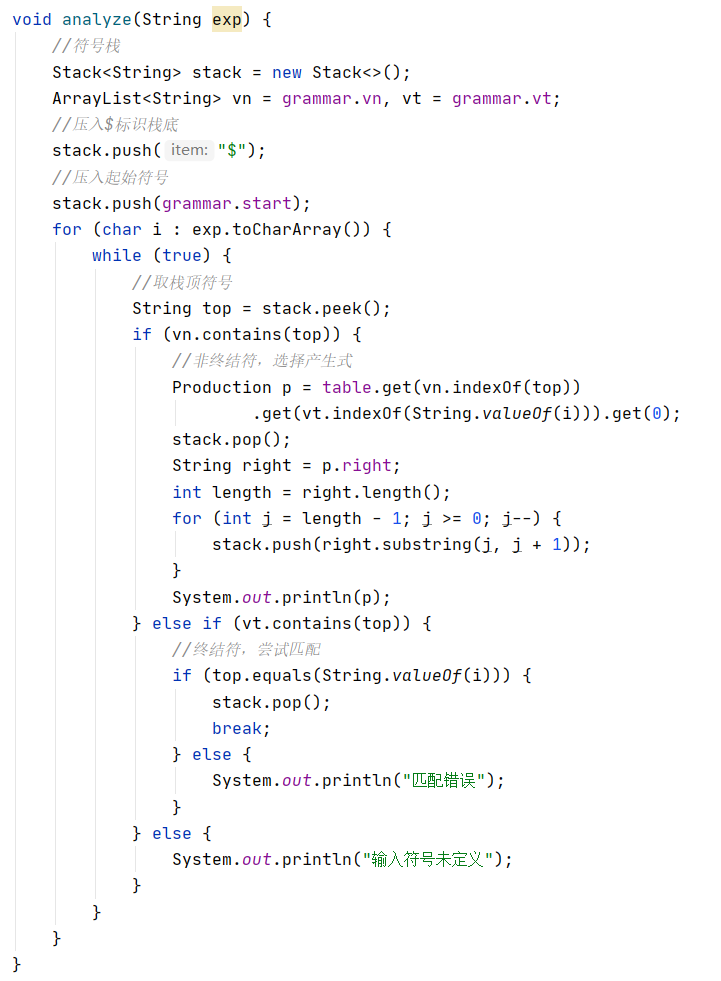
1. 构造LL(1)分析表

使用不包含左递归、无回溯的产生式集，构造预测分析表。



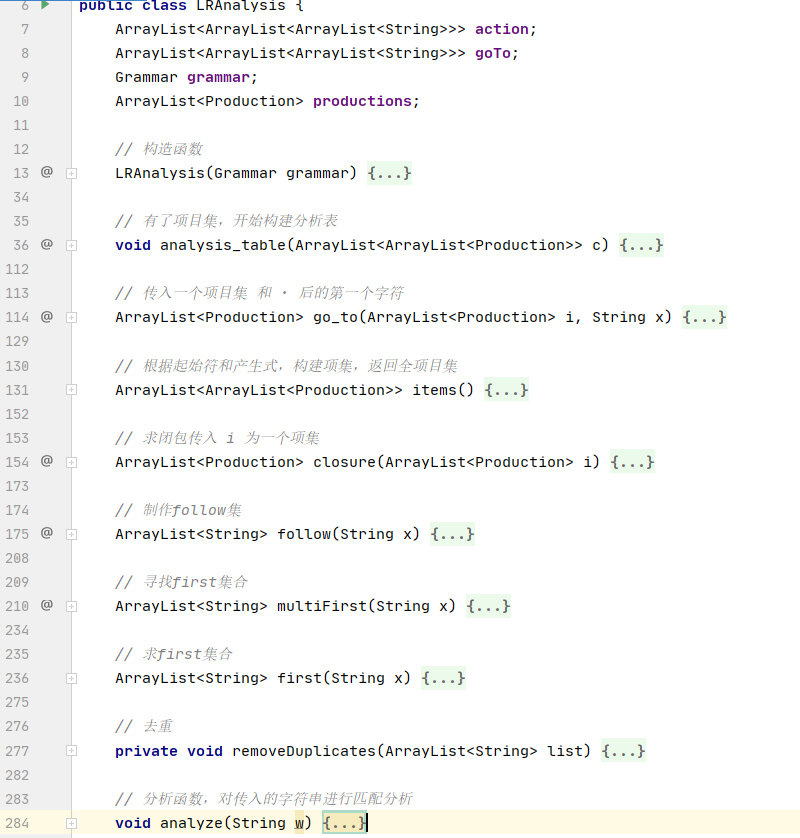
1. 预测分析程序实现

根据栈顶符号和当前读入的符号，进行分析操作，栈顶为非终结符则选取对应的产生式，栈顶为终结符则进行匹配操作：



1. **SLR语法分析程序实现**

程序的所有结构如下

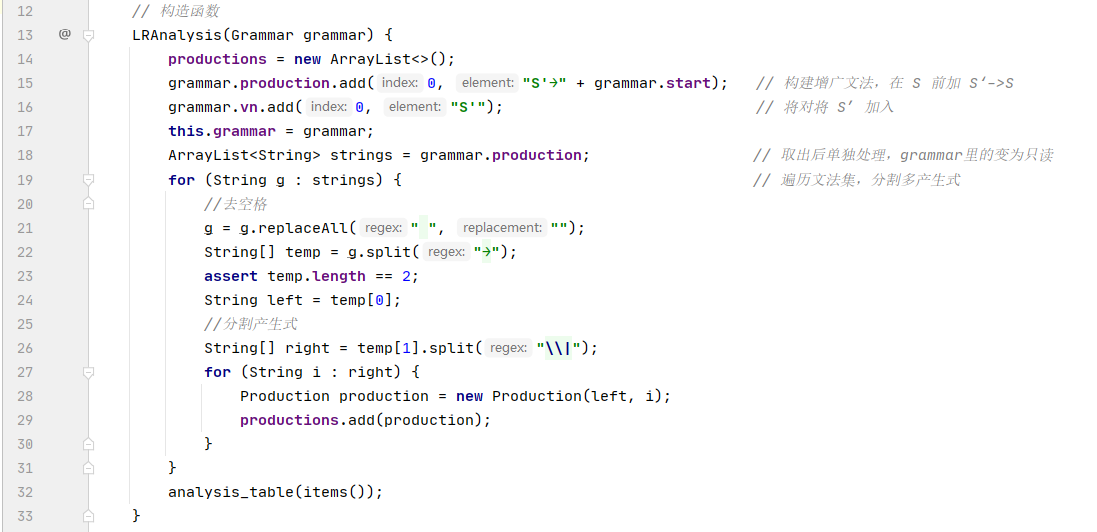


其中grammer存储传入的原始文法，productions存储产生式规则。而属性 action和goto是根据传入文法构建的状态转移表和动作表。

在初始时新建语法分析器时，传入原始文法，将会在构造函数中进行分析表的构建。

然后可以使用analyze函数判断传入的字符串是否为合法符号串。下面对重点的函数进行讲解。

1. **构造方法-分析表构造**

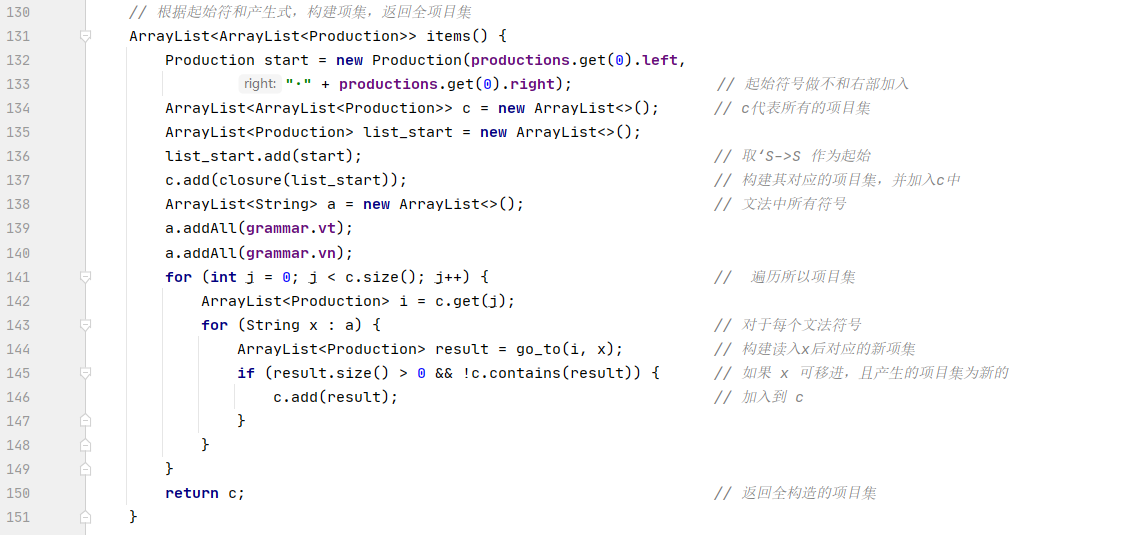


第15-17行构建增广文法并存入成员变量grammer中。

19行的循环去除产生式中的空格，并将所有候选产生式拆开分别加入到成员变量production中

在32行调用构建表方法annalysis构建分析表，出入的item()的返回值，item()方法将使用制定文法中的起始符号和上面存储的产生式构建LR文法的全项目集。

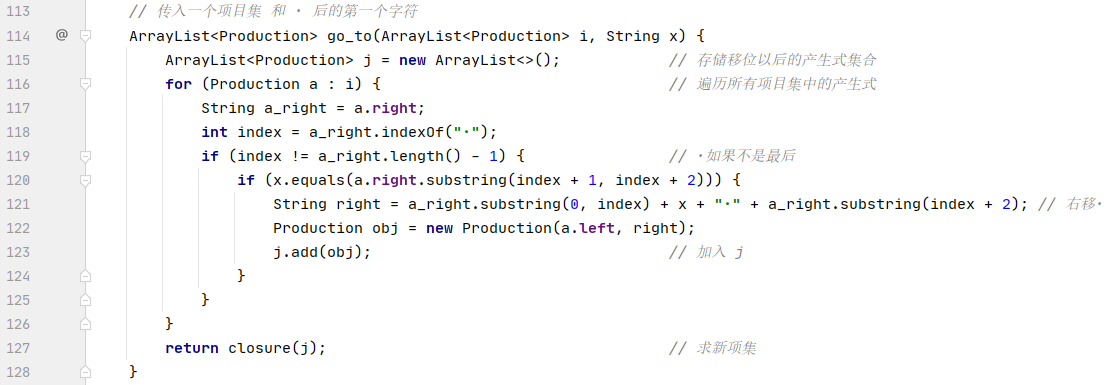
1. **item() 方法-全项目集构建**



构建全项目集。核心在142行循环内，从S’->S开始，对于已有的每个项目集合遍历所有可能的向前看符号，构建项目集加入总项目集c中（对于重复的则不再加入）。

最终返回构建的全项目集合。

1. **go\_to()方法-构建新的项目集**

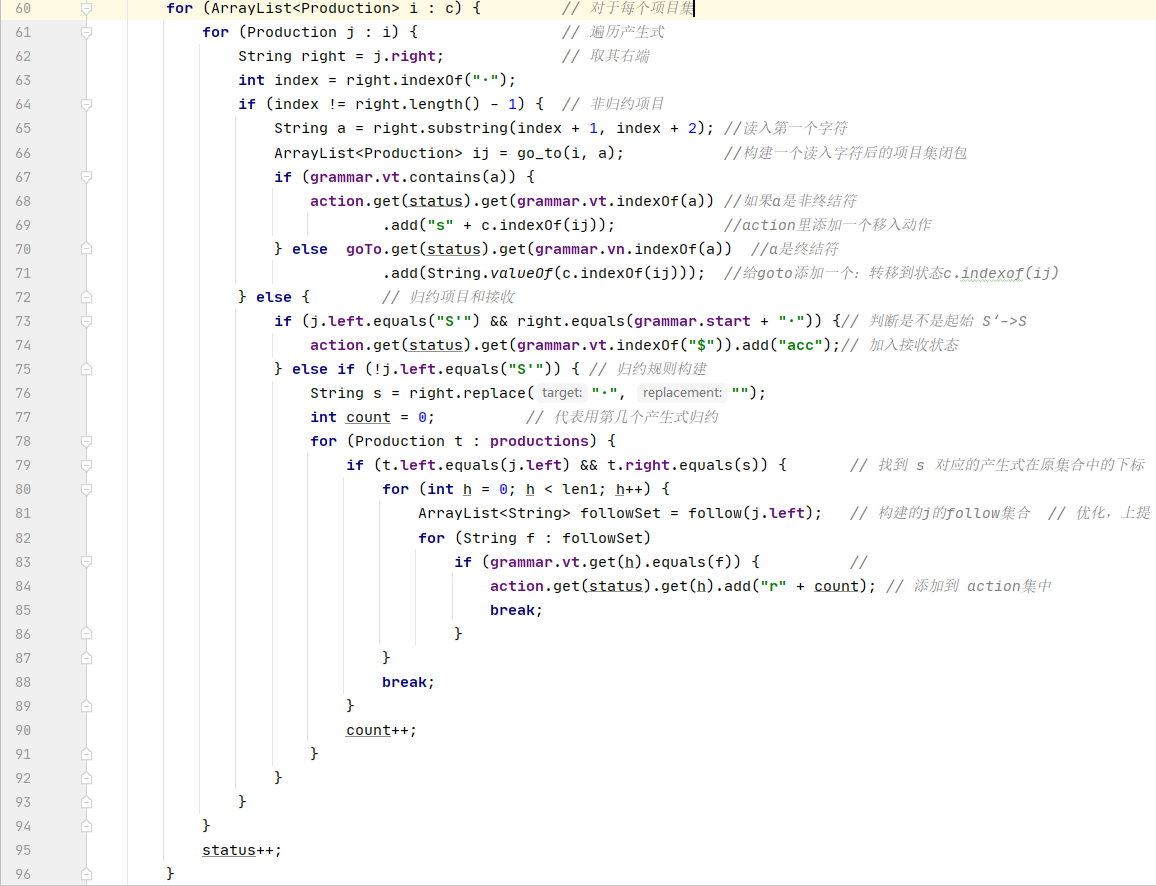


go\_to()方法被item()方法所调用，第一个参数为原项目集，第二个参数是该项目集对应的向前看符号。

在方法中遍历项目集中所有产生式，对于非代约项目，右移·并把它加入项目集j。

返回j的闭包，j的闭包（closure方法与go\_to方法类似，这里就不细讲了）就是要求的新的项目集

1. **analysis\_table()方法**



仿照书本上的做法，进行构建分析表。上述for循环始对所有的项目集的所有产生式进行遍历，通过判断·的位置派分别构造分析表中的规约项、转移项和移近项。

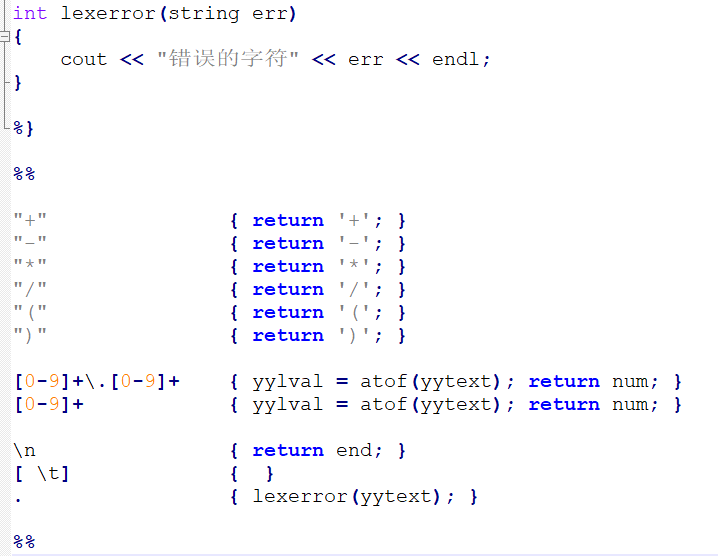
其中，在64行if中构建的是移入项目和转移项目。而72行始对归约项目和接受项目的构建。

1. **Analyze()方法**



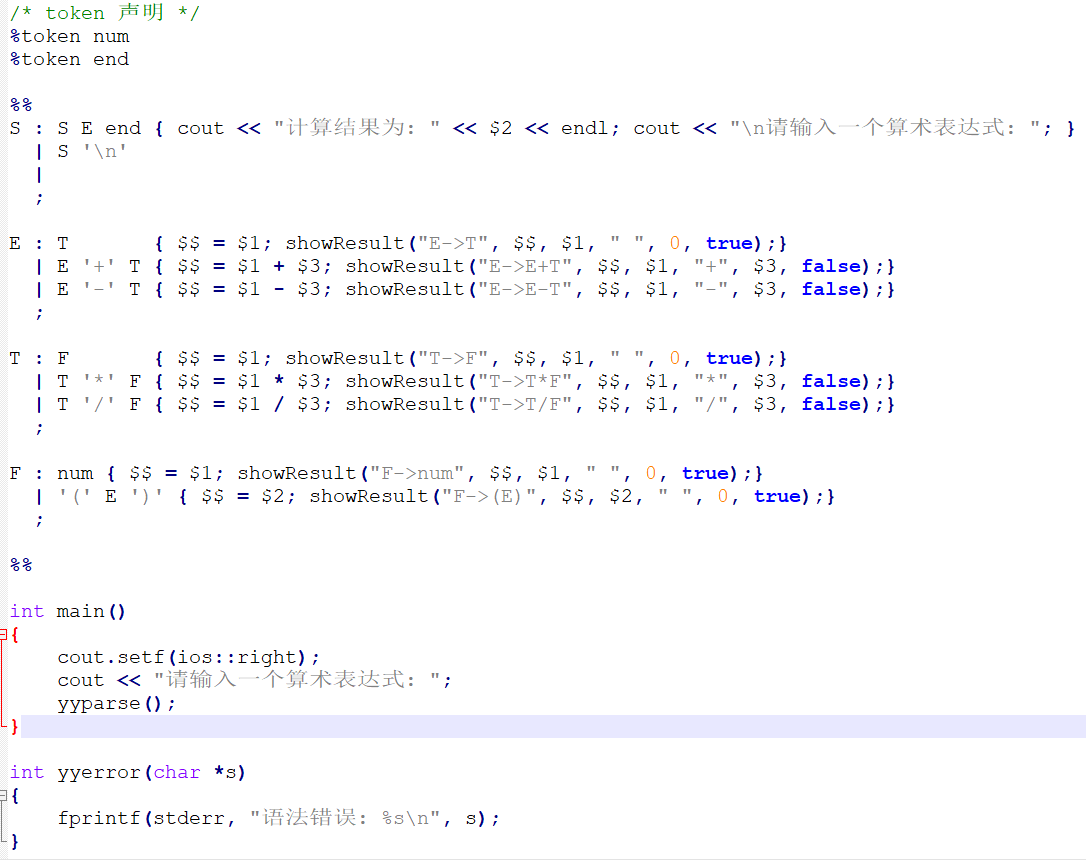
按照书上的查表方法实现，原理较为简单。这里介绍一下对于整数的处理办法，在282行对读入字符是不是数字进行判断，假如是则置数字的标志位为真，反复读到不是数字为止。然后回退一个，将读入字符置为“i”（其就是在标识数字终结符）。上述过程中指针索引count一直在随着变化，所以不影响后续读入其它符号。则可以将任意长度的正整数数字映射到终结符“i”。这样完成了对数字的辨识。

1. **YACC实现**
2. **Calculator.l**



该部分主要为词法识别部分，用户输入算术表达式之后，首先由词法分析器进行词法识别，如果识别的时操作符，那么就返回该操作符记号，如果识别出浮点数或整数那么就返回num记号，通过yylval给该记号赋值，构成各一个二元式记号，如果识别出错误的字符那么就出该字符。

1. **Calculator.y**



该部分主要为语法分析部分，语法分析器通过词法分析器返回的记号，执行对应的产生式，每执行一个产生式就调用showResult()输出执行的产生式和对应数值，其中num为数字记号，保存着读到的数值，end为换行符记号，当语法分析器遇到”\n”后输出计算结果。当语法分析器分析出错误的语法后，输出错误信息，然后跳过错误信息继续执行。