编译原理

实验报告书

实验名称： 语法分析

学 号： 2016210939

姓 名： 于田

实验时间： 年 月 日

**1.实验目的：**

编制一个递归下降分析算法，实现对词法分析程序所提供的单词序列的语法检查和结构分析；

**2.完成的功能模块和模块流程**

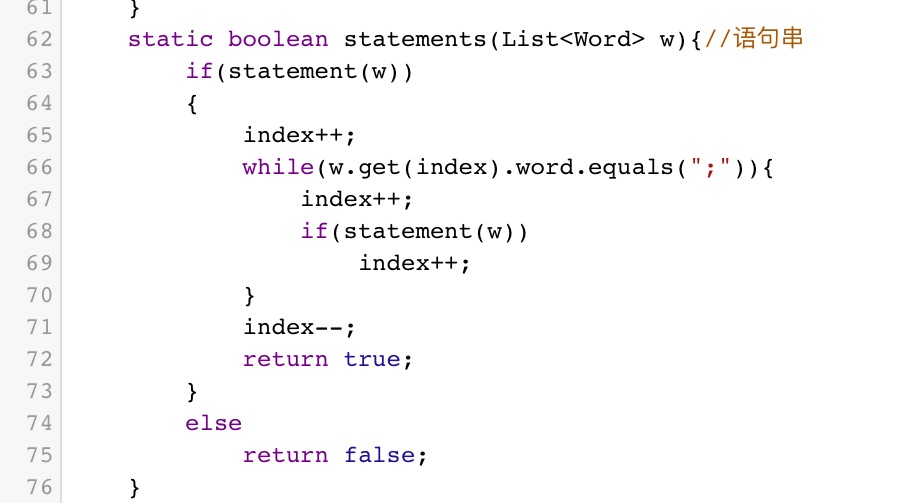
**《1》完成的模块**

（1）program 函数，是整个递归下降分析的开始函数和结束函数，用于最后对于正确与否的判断。规定了整个函数的总体框架是” “begin” 语句块 “end” ”；



（2）statementss 语句块函数，用于确定语句块是由“ ‘{‘ 语句串 ‘}’ ”形式构成。

（3）statements 语句串函数, 规定了语句串是由 “ 语句 { ；语句 } ；”形式构成，其中的“{” “}”是用来表示“， 语句”可以出现零次或多次。



处理方法是，先判断第一部分是不是语句；再看第二部分，采用一个while循环去判断出现一次以及一次以上的语句部分，而对于出现零次，就可以认为这个语句串是符合规范的，对于前面index的移动也要复位，故在返回之前先将index--。

（4）statement 语句函数，判断语句有三种形式， 赋值语句，循环语句，条件语句。

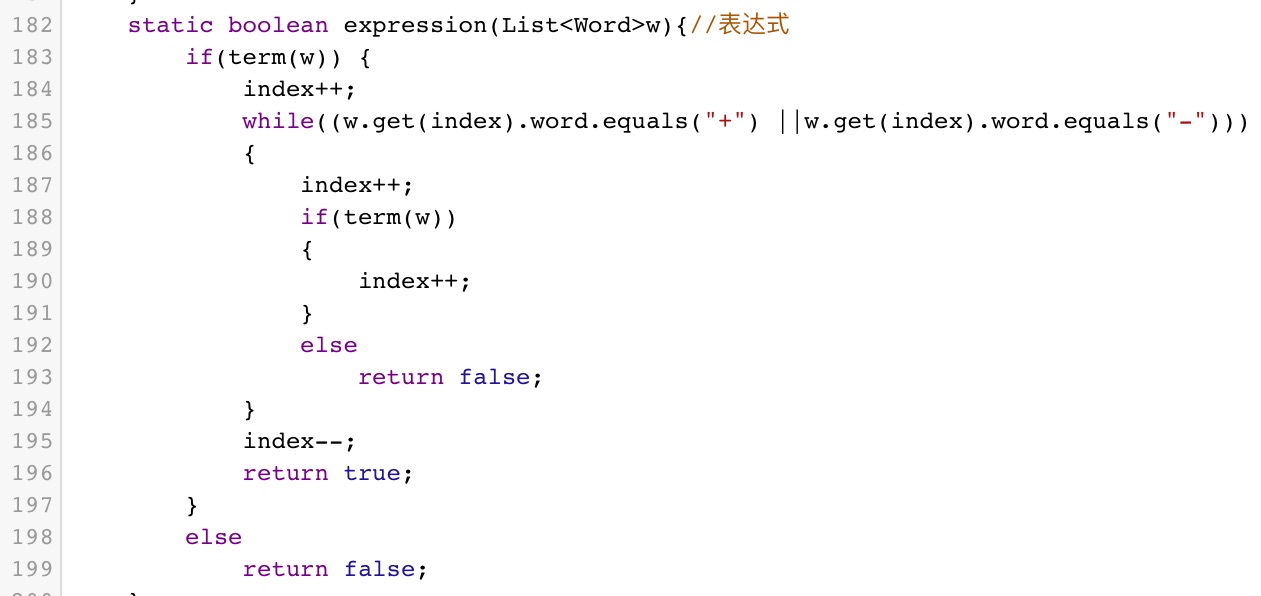
（5）assginment 赋值语句函数，他的形式是“ ID = 表达式” ,需要进一步递归下降分析。

（6）condition 条件语句函数，他的形式是“ if( 条件 ) { 语句块 }”；对于条件和语句块，同样还需要递归下降分析。

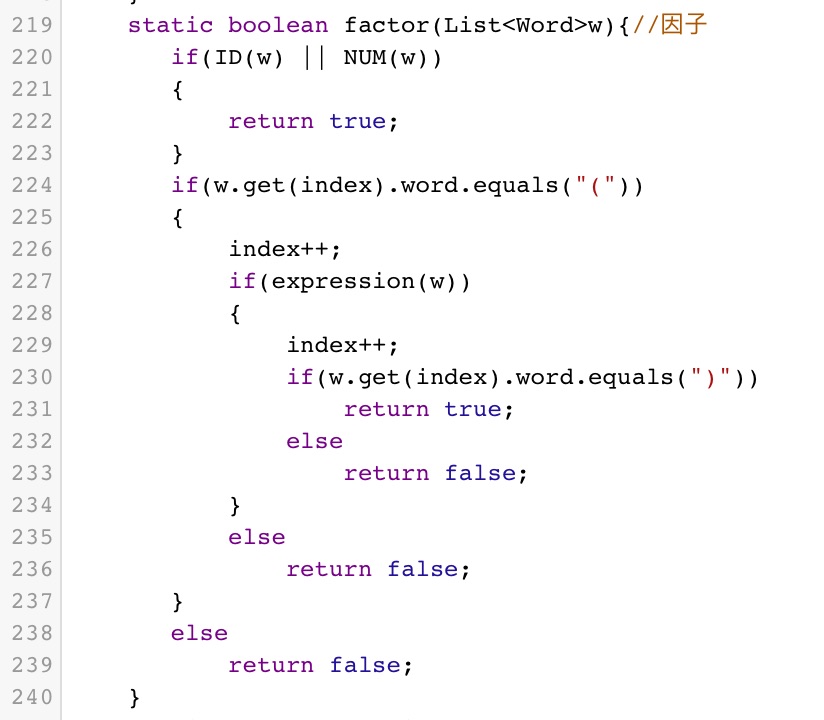
（7）loop 循环语句函数 ，他的形式是“while（条件）{ 语句块 }”；

（8）conditions 条件语句函数，他的形式是“表达式 关系运算符 表达式”;

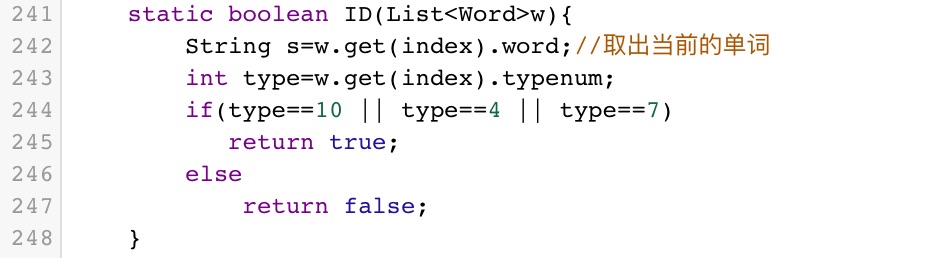
（9）expression 表达式语句函数,他的形式是“项 { +项 | - 项}”；对于第二部分大括号的内容，同样是可以出现零次或多次。



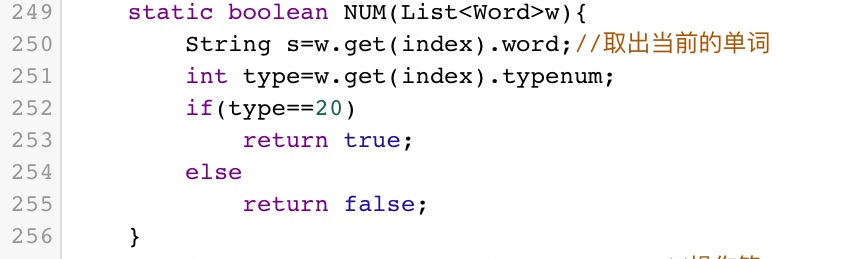
（10）term 项函数，项的形式为“因子 {\*因子 | /因子}”，大括号的内容可以出现零次或多次。



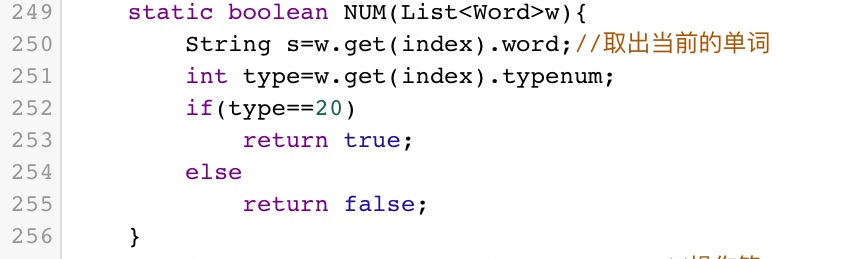
（11）ID 函数，ID指的是标识符，我们可以用词法分析里的type去判断。Type等于10为以字母开头字母或数字组成的标识符，type等于4为关键字“if”，type等于7为关键字“while”.



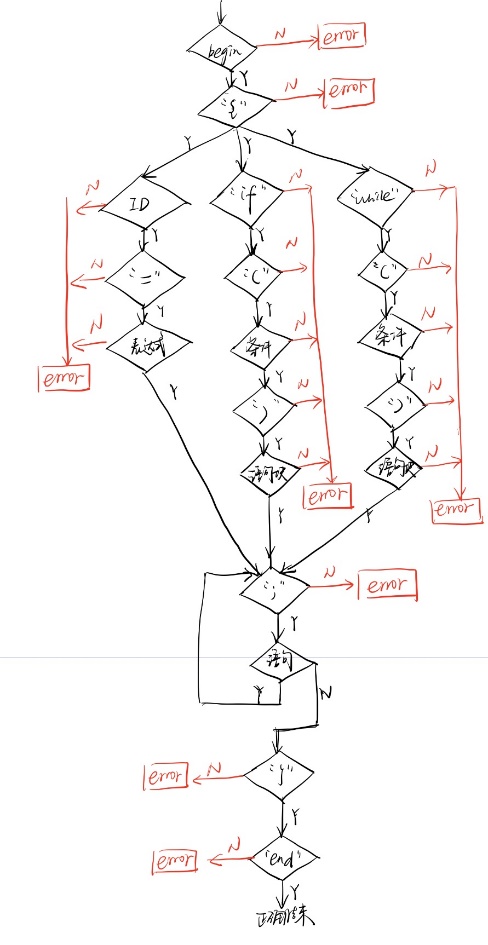
（12）NUM 函数，是用来判断是否为数字。



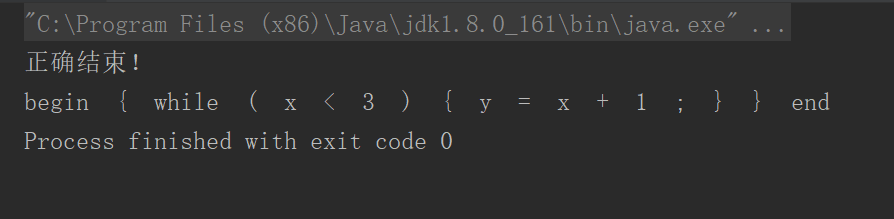
（13）operator 操作符函数，用来判断是否为操作符。



**《2》模块流程**



**《3》运行结果**



**3．关键技术和难点**

（1）对于递归下降分析的理解，是确定的自上而下的分析法，对于文法中的每个非终结符编写一个函数，每个函数都是识别由该非终结符所表示的语法成分，这些函数之间递归调用。

（2）因为本程序没有写scan 函数，所以设置了一个全局变量index，用来记录扫描的位置，而且在出错时，可以反馈出错的索引位置。

（3）本程序调用了上次实验的词法分析程序，使得程序的切分以及对于词的判断变得容易实现。因为词法分析程序最后的结果就是按照顺序的切分好的词以及词对应的属性名称。

（4）对于语句串中语句出现零次或多次，表达式中“+项| -项”出现零次或多次，项中“\*因子|因子”出现零次或多次的处理，采用while循环的方法进行递归循环判断。

**4.编程调试心得**

（1）编程之前要先理解项目的目的，因为我开始仓促的实现程序，没有读清楚语法分析的规则，使得后期重新修改浪费了很多时间。

（2）开始对于{}中的内容出现零次或多次没有很好地理解，导致程序出现持久的很难修改的bug，令人崩溃。所以要认真的理解完要求在进行实现，对于基本的符号含义要熟记。

（3）debug真的是很好用的操作，可以让我发现很多没考虑到的问题，所以在编程的时候，我会编写多个样例进行试验，如果出现问题，那也是一件好事，经过debug发现一些细节上没有考虑的东西，使得程序更加完善。