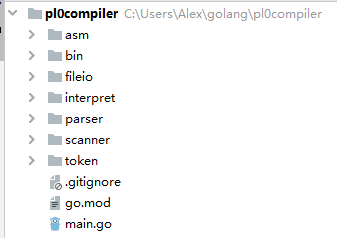
# 编译原理实验报告

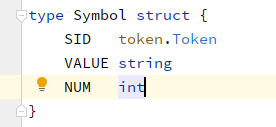
**一、整体概括**

目录结构如下图所示，其中fileio包负责底层文件读入，scanner是语法分析器，parser是词法分析器，interpret是解释执行器。

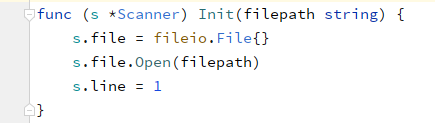


**二、词法分析器**

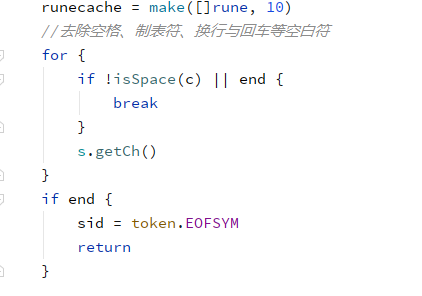
Symbol符号结构体有三个域，分别记录Symbol的类型，文字，如果是整数就记录数的大小



初始化的时候传入文件路径，并置行号为1，行号为了后面报错能够精确到某一行



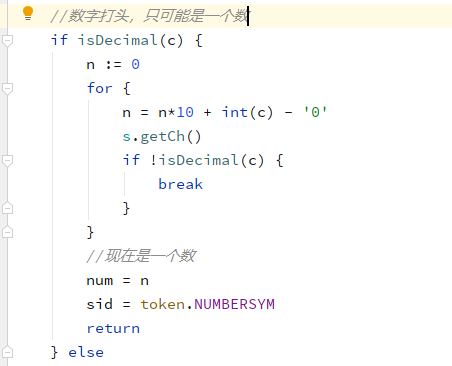
词法分析器主函数一开始先去除遗留的空格，知道遇到第一个非空白符，如果读到结束符就结束



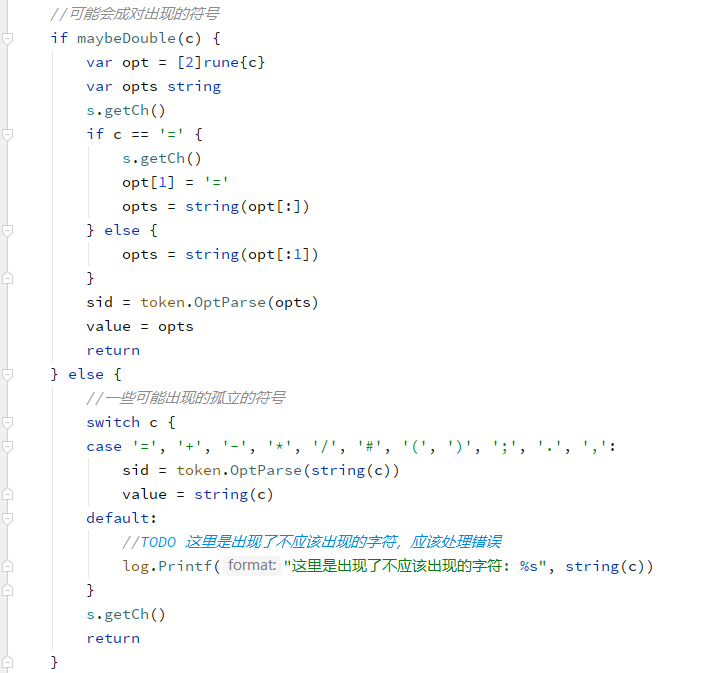
如果是字符，就有可能是一个单词，不断向后扫描，知道遇到第一个空白符为止，当长度过长时就报错。



如果是数字开头，就有可能是一个数字，不断向后扫描，知道遇到第一个非数字，在扫描过程中计算数字的大小



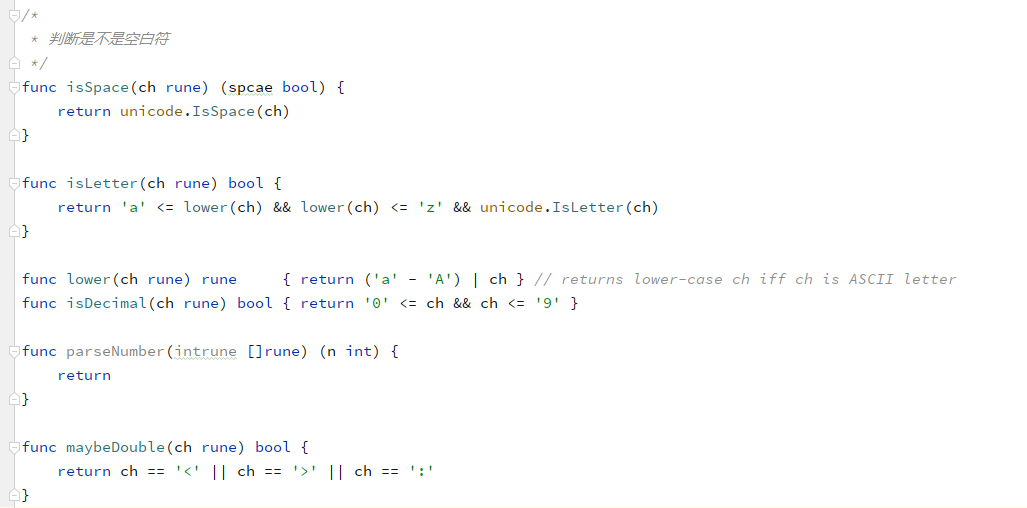
如果遇到 ：、>、<这三个符号，就有可能是:=\>=\<=，需要再看一位才能确定，如果是其他符号就返回相应的Symbol



读取一个字符的函数，如果遇到换行符需要给行号加一



一些辅助方法，判断字符的类型

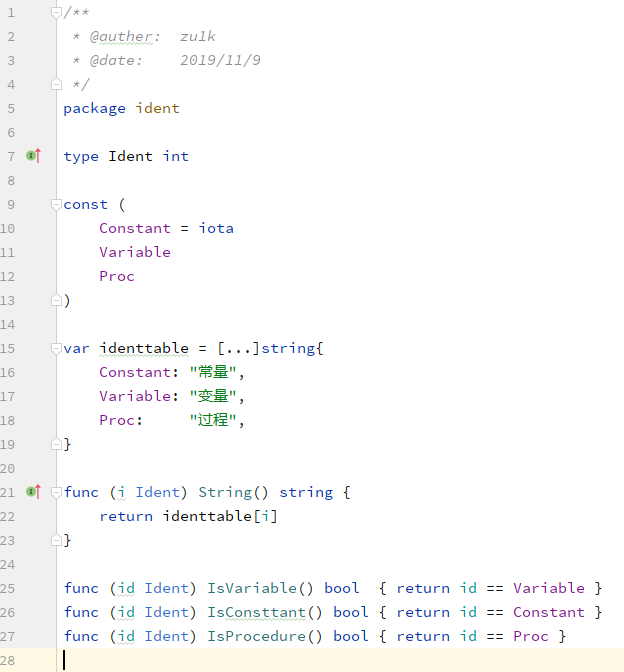


**三、语法分析器**

在语法分析过程中需要维护两个表格，一个是符号表，另一个是中间代码表。

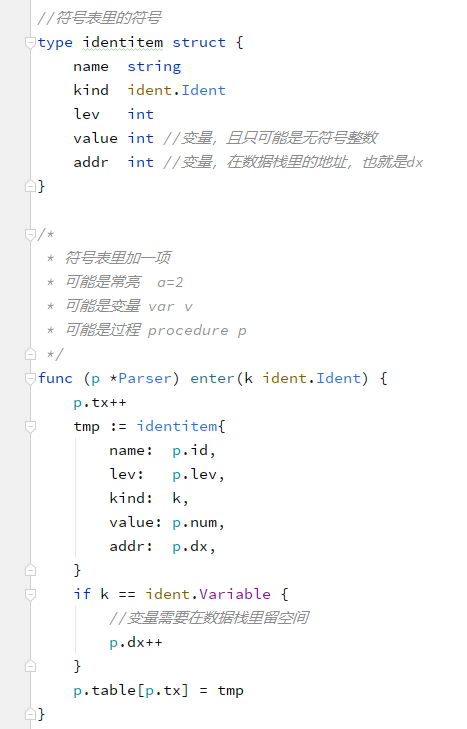


对于每一个标识符都有三种可能，常量、变量、过程，这里准备一种类型进行记录并准备好判断的方法供后面使用。

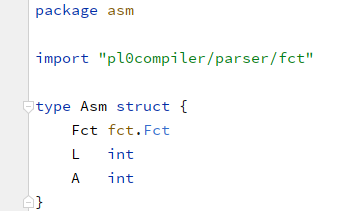


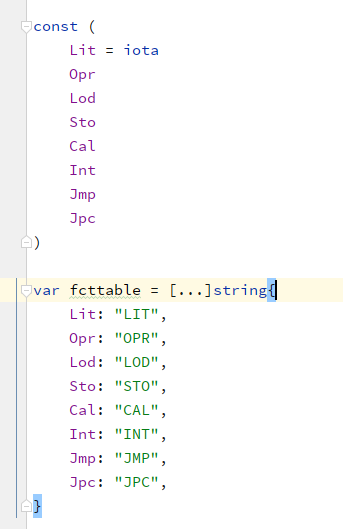
对于符号表，需要准备好存储在符号表中的每个符号的结构体，每一个符号都需要保存它的文字、类型、层级、地址，如果是数字需要保存它是值。

还需要准备好向符号表中添加一项的方法enter.



对于中间代码，我使用一个结构体asm来存储每一项的操作符、操作数、层级





/\* lit 0, a : load constant a

opr 0, a : execute operation a

lod l, a : load variable l, a

sto l, a : store variable l, a

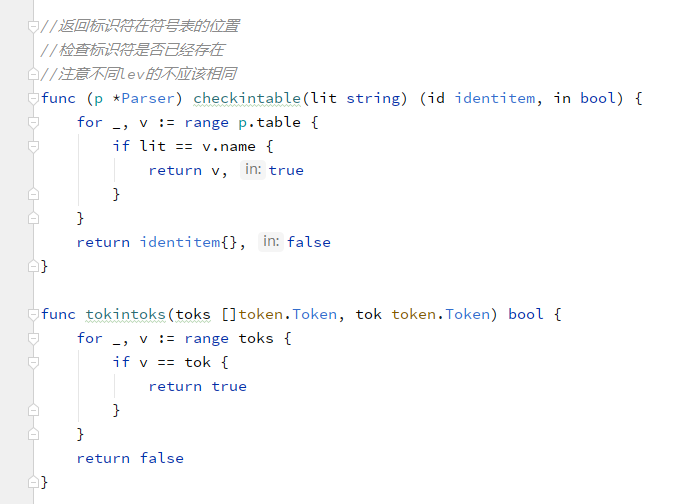
cal l, a : call procedure a at level l

Int 0, a : increment t-register by a

jmp 0, a : jump to a

jpc 0, a : jump conditional to a \*/

在遇到一个符号调用时需要判断这个标识符是否已经在符号表中存在



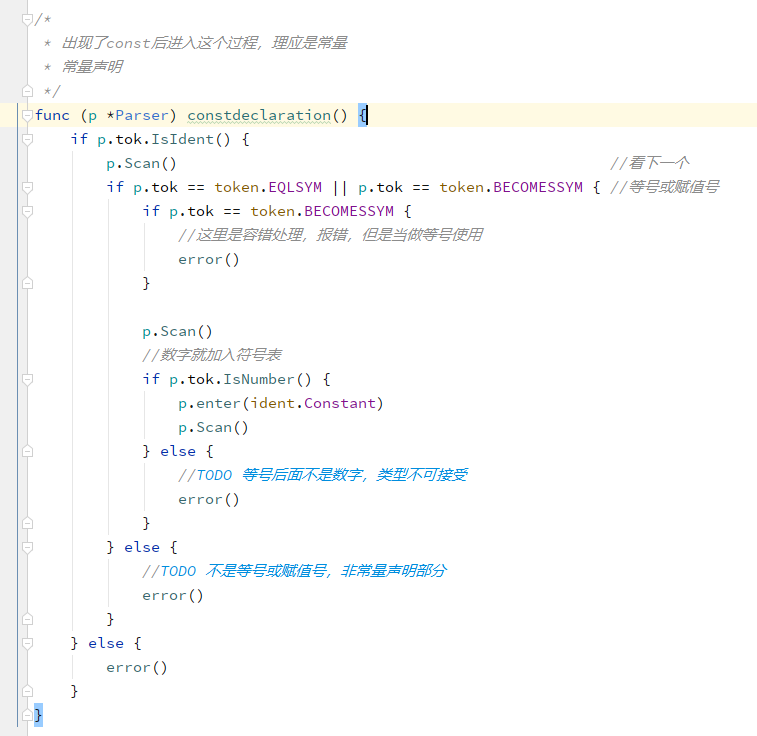
同时我还需要准备一个方法判断当前栈顶单词是否在select集中



做好准备工作，开始正式进入语法分析。

我用的是递归下降的分析方法，需要针对每一个产生式写一个分析方法，让后让语法分析程序从顶层函数block开始自动递归的进入其他分支过程，完成语法分析。

当出现const关键词后需要进入常量声明过程，紧接着是标识符、赋值号、数字，这个过程如果没有问题的话就可以通过enter加入符号表了，如果后面接着逗号，需要再次进入这个过程，知道遇到分号结束。



当出现var关键词，需要进入变量声明过程，与常量声明类似。



如果下一步应该是一个引子，则进入因子的处理函数。



当判定是一个项时就进入项的处理函数。



当判断下面应该是一个表达式的时候，就进入表达式的处理函数。



如果是if while判断，需要是一个条件，就进入条件的处理函数。



语句的产生式比较多，在函数里面需要进行更多的switch和if判断，判断进入哪个产生式的处理流程。



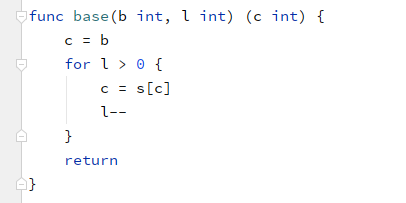
最顶层的block函数处理整个代码



**四、解释执行器**

解释执行器需要解析每一条中间代码，模拟栈并进行相应的运算。

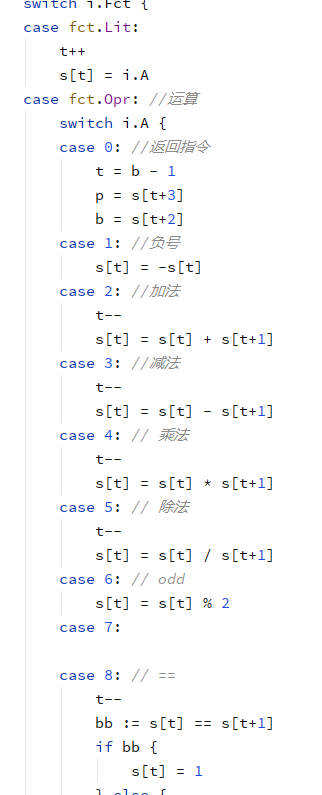
因为有层级的存在，所以需要一个专门的函数计算真正的内存地址

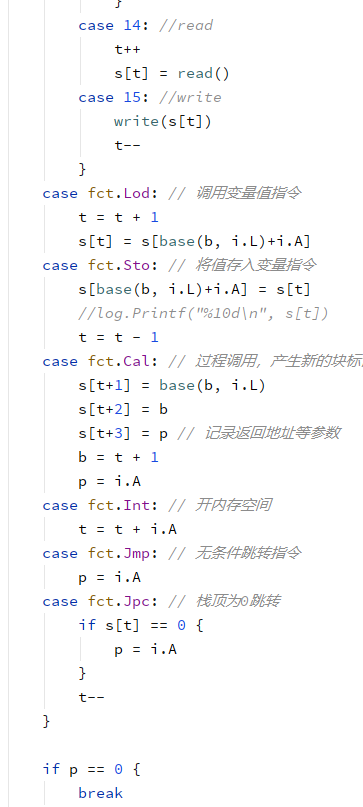


同时需要维护四个寄存器，分别如图所示。



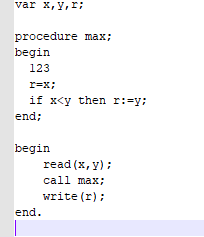
Opr指令是比较特殊的，需要根据后面的操作数来选择操作类型。

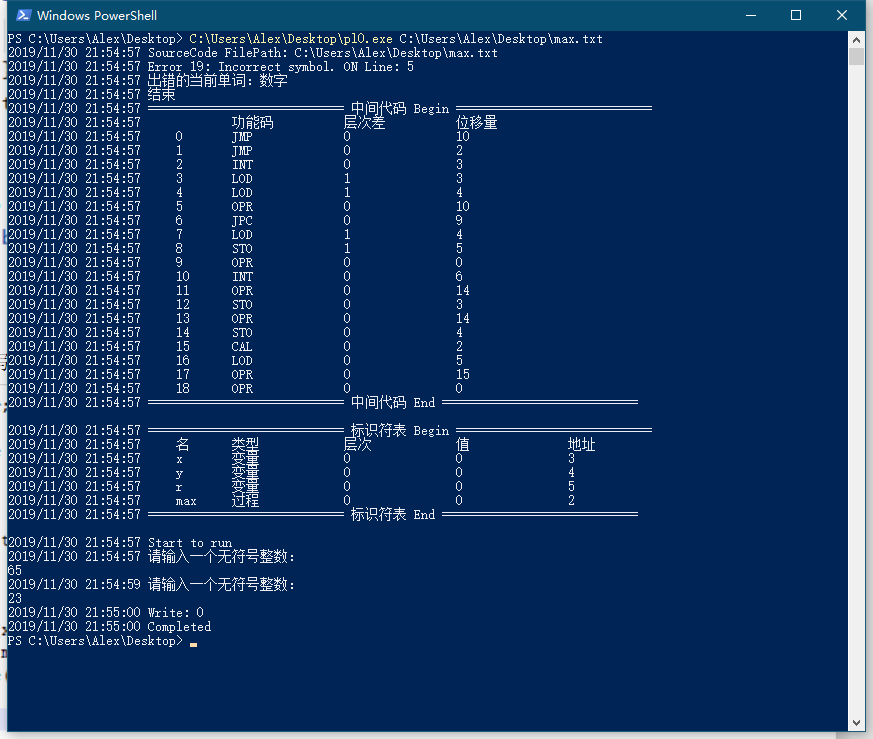


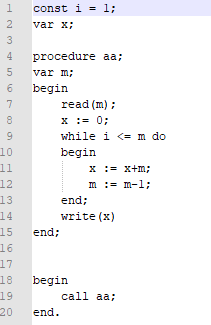


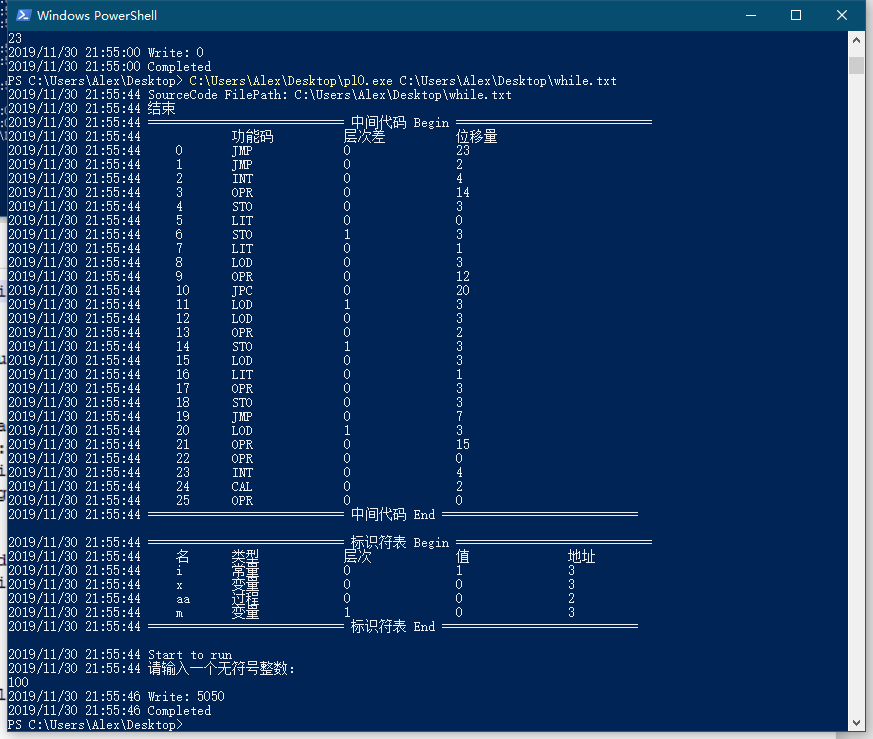
**五、测试**

我用pl0编写了两个程序，分别是求最大值和累加，用来验证程序的正确性。









**六、总结**

通过这次实验，对编译器的词法分析和语法分析过程更加了解了，更加深刻的认识了中间代码的生成过程。