本次实验题目要求是实现一个PL0语言的编译器，实现词法分析，语法分析和执行代码三个功能。

具体要求如下：

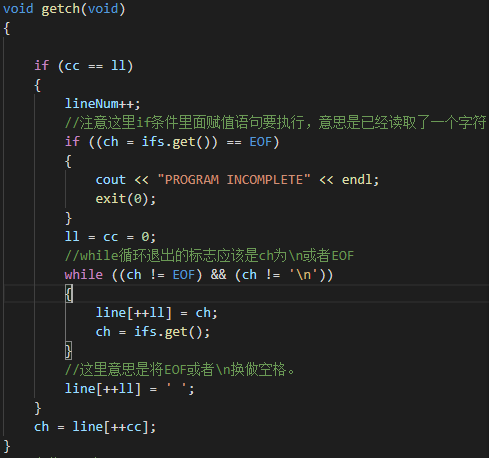
1. 要实现两个函数，getsym()用于得到一个symbol，getch() 用于读取代码中一个一个的字符。
2. 语法分析程序block()函数要实现两件事：1）将说明部分的相关变量放入到table表中，2）生成PL0语言的目标指令，存入code数组中。
3. 建立一个解释执行目标程序的函数。

我采用的是递归处理程序的形式实现的，没有采用建立分析表的方式实现。下面依次讲

解各个方法的实现。

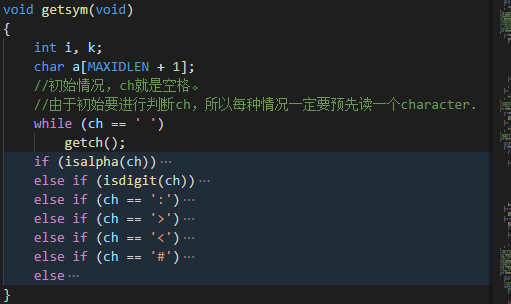
**词法分析程序**

首先是getch()程序，想法是一次读入一行的代码，存储到缓存数组中，这样就不用每次运行词法分析程序都得进行读的操作，并且还只读一个字符！详细代码如下：

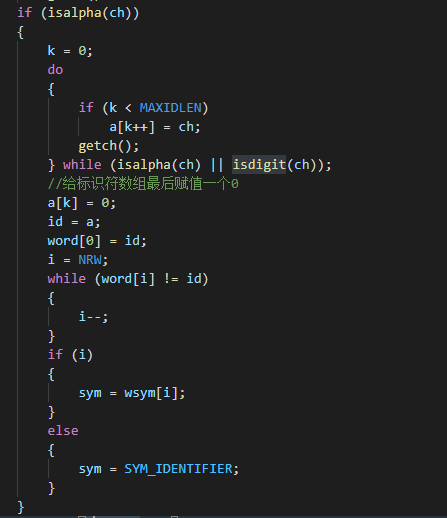


接下来就是getsym这个函数了，symbol也就是标识符，数字，关系运算符，还有标点符号。因此在头文件里面我预先定义了所有的symbol，组成了一个枚举类型。这就是全局变量sym的值域了。

下面是getsym()函数的一个简易的结构：

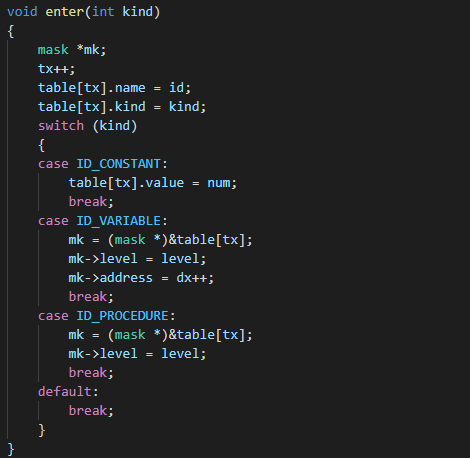


下面以标识符为例进行说明，根据题目的表述，标识符应该是字母打头的字母数字串，这里判断出来标识符以后还要进行保留字的判断，只有符合规则且不是保留字的情况才是标识符。

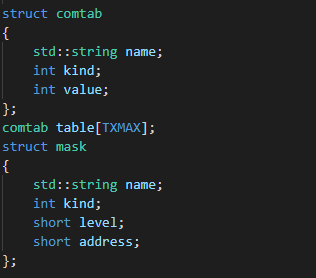


**语法分析程序**

这里需要实现两个任务，一是将所有说明部分的变量放到table表中，这里补充一点PL0语言是先声明再有执行语句的，所以不用担心在执行语句的时候突然蹦出一个变量声明来，这两个区域是严格分开的。处理变量声明的核心方法是enter()方法，

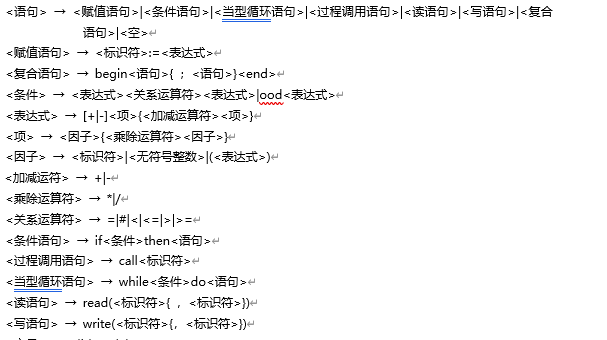


对于不同类别的声明会有不同的处理措施，本质上都是填table这个数组，tx作为table的一个指针是一个全局变量。值得一提的是mask这个类型，这个类型是为了处理变量和过程需要填写的信息有四个，常量需要填写的信息有三个的问题。

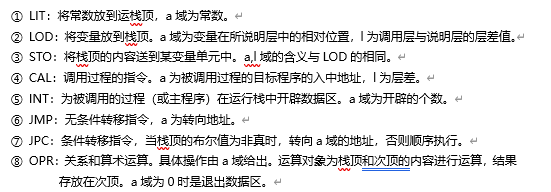


上图就是两个类型，但占用空间是一样的。

完成了填表的操作之后，就是生成PL0目标代码了，具体就是针对于每一个语句，设计一套等价的执行代码：



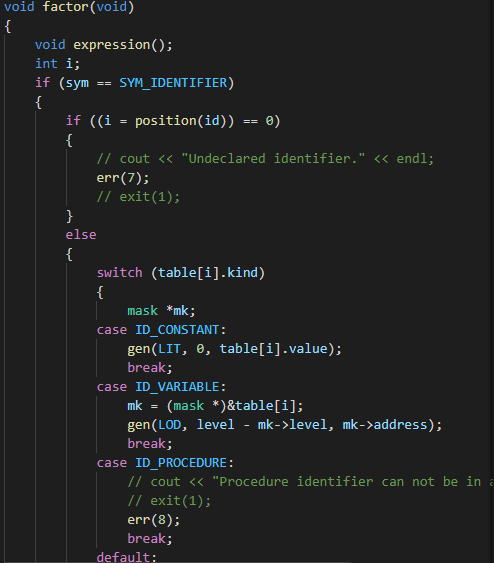
上图是关于PL0语言的一个文法表示。可见需要设计赋值语句，条件，读语句，写语句和复合语句的等价目标代码。



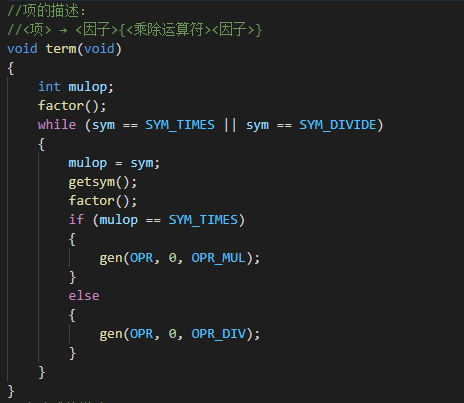
上图是PL0允许的目标指令，

对于赋值语句（a = 5）将常数5放到栈顶，执行STO指令。对于条件语句（a + b > c +d）先处理左边的表达式，将a变量放到栈顶（变量在栈空间内被分配了一块地方），将b变量放到栈顶，执行OPR 0 ADD命令，此命令是将次顶和栈顶的值进行ADD运算，而后将结果存在次顶中（因为运行完之后，次顶就是栈顶！），从两个例子可以看出来栈顶是相当重要的，操作命令都是和栈顶有关的。下面以条件语句为例子进行说明。

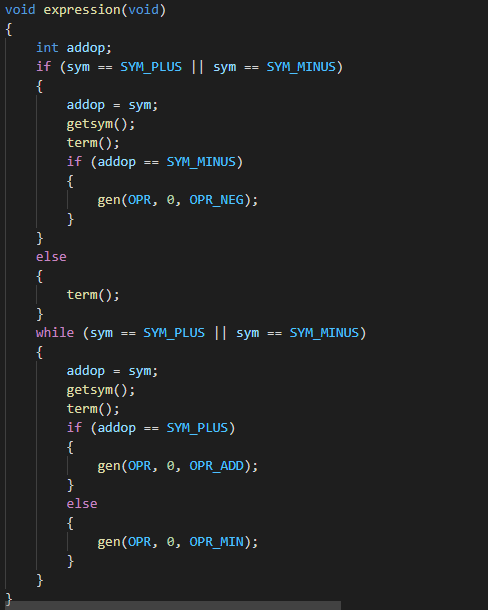
条件->表达式->加减项->因子



上图是在处理因子的情况，



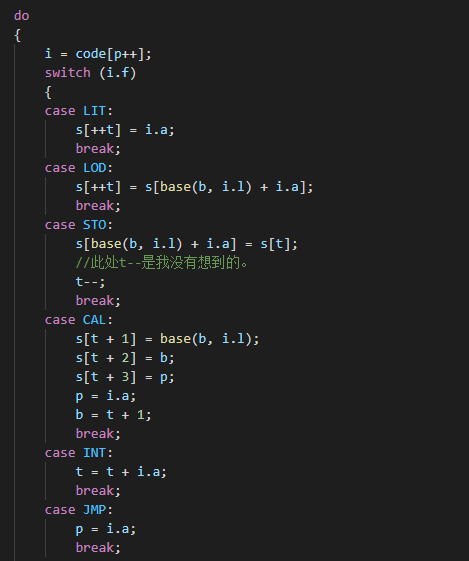
上图是处理项的相关代码。



上图是关于对于整个表达式读取并处理（生成目标代码）的函数。

**建立一个解释执行目标程序的函数**

执行的时候，内存中只有存放目标程序的code数组和运行时的栈式数据空间S，实行这个函数的时候，要重点了解每个过程被调用（包括主过程，要不然dx初始值怎么是3呢）时，栈顶分配三个单元，一个单元是调用链最开始的过程的数据段的基地址，一个单元是该过程运行之前正在运行的过程的数据区基地址，一个单元是当时程序寄存器P的值，用于过程执行完的时候继续执行之前的代码。



上图是执行函数的一个片段，其中s是栈式数据区，执行的逻辑大致就是根据每条命令要做的事情，需要的先把操作数移到栈顶，再进行运算。