中间代码生成

# 一、实验目的

通过上一步语法分析得出的语法树，将其转化为xml格式的四元式。

# 二、实验环境

系统：WINDOWS7

软件：IntelliJ IDEA

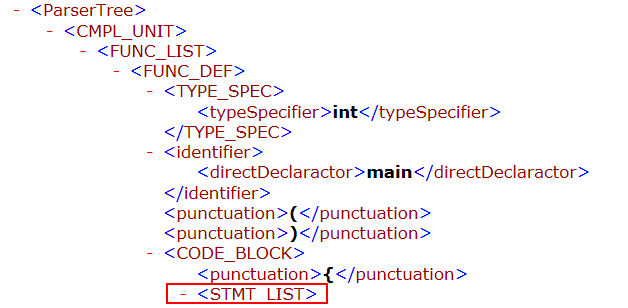
# 三、实验步骤

## 3.1 算法介绍

首先，整体的算法是先对语法树进行读写，将每一个STMT（statement）节点放入一个LIST中，然后不断地从LIST中取出节点，进行四元式的转换（当然其中涉及到循环操作的话需要从队列的头放入节点），直到队列为空，生成结束。

## 3.2 递归读取语法树

首先，假定只有main函数，因此可以直接将根节点锁定到语法树中的STMT\_LIST节点。



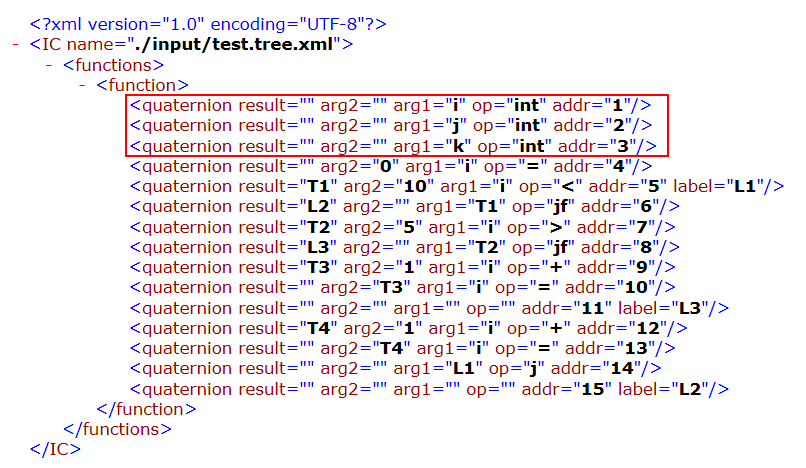
由于STMT\_LIST在语法树中是递归展开的，即STMT\_LIST->STMT STMT\_LIST | ε ，因此在读取的时候也应该递归地进行读取，具体实现代码为pushQueue方法。在读取后，直接将这个节点放入LIST中。至于为什么要采用LIST的数据结构，是因为我在最开始考虑的时候认为当递归读取结束后，不会有新的节点加入队列，因此觉得采用先入先出的队列方便处理。但是在后面当我遇到循环节点的时候（比如for循环），发现还需要将for循环节点下的BLOCK中的STMT加入队列处理，而且处理的顺序是马上处理，因此将队列的数据结构改为了两端开口的ArrayList。

## 3.3 LIST中节点分析

每次从LIST的头取出一个元素（remove方法），然后判断子节点的个数。由于在语法树中，定义变量statement有3个子节点，for、if、其它语句有1个子节点，因此可以通过节点个数的特点将他们分开，进行四元式的生成。

### 3.3.1 变量定义四元式转换

如果从LIST中取出的节点的子节点有3个，那么他一定是变量定义。这3个子节点分别是TYPE\_SPEC、identifier、punctuation。再遍历前2个子节点就可以得到变量类型、变量名，然后可以生成四元式如下图红色框中：



在老师给的例子中，我没有看到变量定义的四元式，但是我认为由于最后的目标代码生成完全是依赖于这个四元式的，因此变量定义是必须的。

### 3.3.2 for循环四元式转换

如果LIST中取出的节点的子节点只有一个，那么对它的名字进行判断，如果是FOR\_STMT，那么就转入for循环的转换。

对于一个for(e1;e2;e3) {block}循环，如果所有部分都不为空的话，至少需要生成5个四元式。首先是e1，也就是变量的赋值，然后是e2，判断关系，这里面需要生成2个四元式，一个是用来储存判断的结果，另一个是根据结果进行跳转(jf, \_, \_, res)，然后是生成block的四元式，这里面要注意，block中可能继续嵌套任意语句，因此需要将block的STMT子节点放入LIST的头。但是要注意最后还有e3，但是block的STMT需要在e3之前生成，因此在将STMT放入LIST前还需要将e3放入LIST的头。

### 3.3.3 条件判断四元式转换

在条件判断四元式中，由于在for循环的e2部分我已经写过了条件判断的跳转，因此在这里我直接拿过来进行了一些修改就可以了。在此不做赘述

### 3.3.4 其它语句四元式转换

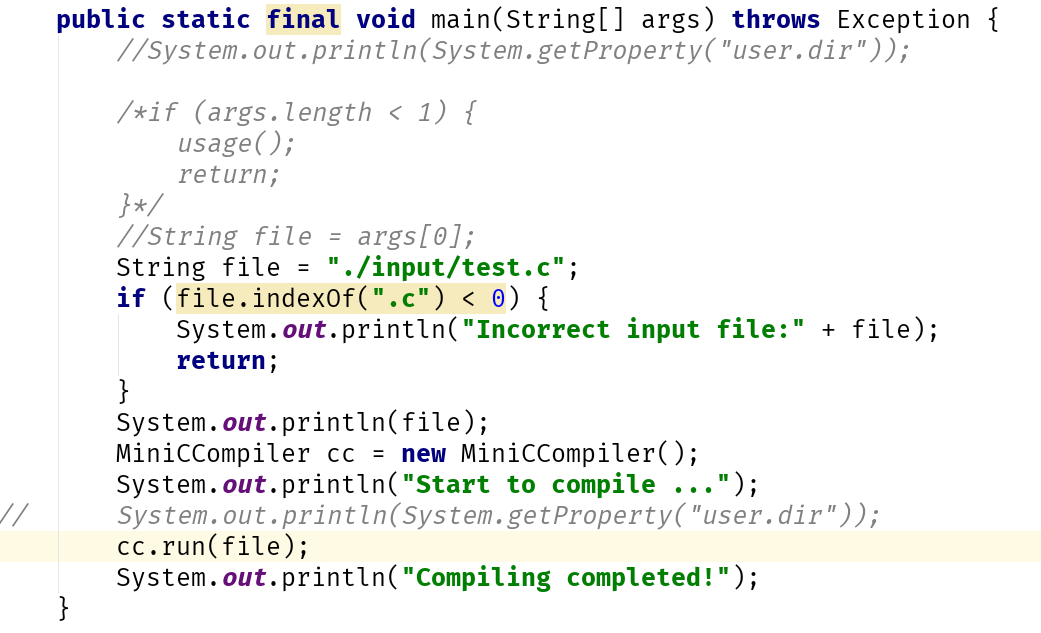
其它语句，包括变量自增/减操作，由于在上一步我做语法分析的时候没有完善加减乘除这些操作，因此在这里先暂时不做考虑。自增/减需要生成2个四元式，第一个用来做+/-1，第二个用来赋值。

# 四、总结

## 4.1 说明

对于代码test.c，生成的语法树为test.tree.xml，生成的中间代码（四元式）为test.ic.xml

我对运行方式进行了修改，不需要像原来一样输入文件路径，默认路径为./input/test.c



并且将依赖的jar包放在了./lib下，因此直接build后run就可以了。

## 4.2 心得体会

虽然这一步中间代码生成是选做的，但是我认为这步是整个环节中必不可少的一步，并且在理论课的时候我对四元式，特别是加入循环后的四元式理解就不是很深刻，因此希望通过完成中间代码生成来加深对它的理解。

在实验中，我采用了dom4j的jar包进行xml文件的读写。之前在做词法分析、语法分析的时候都用的是w3c的jar包，但是由于后面的四元式中需要设置属性（AddAttribute），而w3c的增加属性不允许设置变量值，只能设置属性的值常量，所以在上网查阅资料后，决定采用dom4j。

此外，在编写的时候仍然有很多的bug，其中遇到最大的问题就是NullPointerException，但是由于有了上次语法分析的经验，我在调试解决问题的时候省了不少时间。这个Exception抛出的主要原因是因为读取到了空的元素，而又没有进行判断而直接调用它的方法或者数据，因此只要关注这个空的元素是怎么被读取的就行了。并且我还自学了部分正则表达式的用法，在代码中关于label的匹配的地方有体现。