CMM解释器个人文档

2015302580239 冯清铎

本次实验，我主要负责词法分析语法分析的完善，语义分析生成中间代码以及符号表的构造。

1. 词法语法分析完善

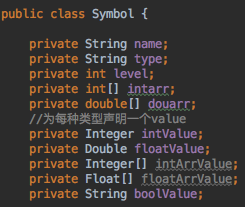
因为在原文法基础上添加了for循环，所以需要对词法文法进行完善，在词法的保留字中添加for；

在语法规则中加入forstmt -> for(expression; expression; expression){expression;}

并且对相应的函数中做修改，为forstatement添加一个递归下降函数。

1. 符号表的构造

符号表的定义：

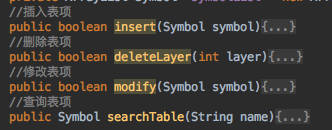


首先是符号的申明，每个符号都对应一个类型，一个符号名，一个值以及对应的层次，因为在进行构造的时候，传入的都是String类型，不方便进行之后的操作运算，所以为每种类型都申明了一个value变量，构造时对类型进行判断，然后赋值到对应的value中。

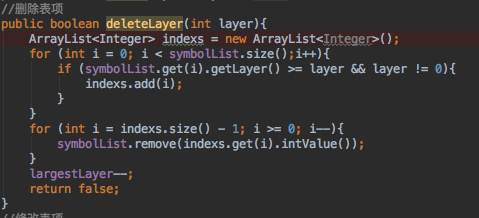


然后是符号表，使用一个ArrayList来存放所有符号，一个符号对应一个Symbol类型变量。

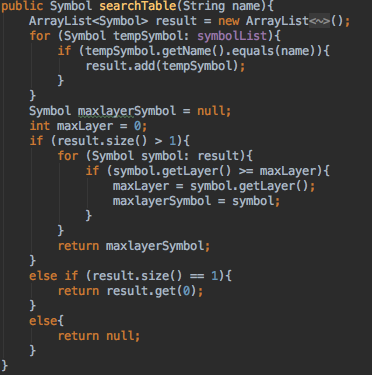
定义了如下几个方法对符号表进行操作：



在这里主要阐述一下删除操作。删除操作传入的是int型变量，因为变量在申明之后不会释放，除非变量层发生了变化，比如说进入了while循环，其中申明的变量全部为临时变量，当跳出while循环时，刚刚申明的变量需要全部释放，这时候才需要调用删除函数，对所有该层以及该层以上的函数进行删除 ；删除时因为是在遍历符号表循环中进行，所以不能直接对符号表进行修改，要先记录下符号下标，然后从后往前进行逐个删除。



查找函数如下，因为允许在嵌套层中对层外申明过的变量重新申明，例如int i ; while(true){int i;}所以要对搜索和更改进行修改，防止对层外的变量进行修改，因为在嵌套中总是对最里层的变量进行修改，所以，需要找到最里层的变量，也就是相同符号中layer最大的。



1. 语义分析
2. 中间代码生成

使用四元式来表示中间代码，为四元式申明一个类，如下：



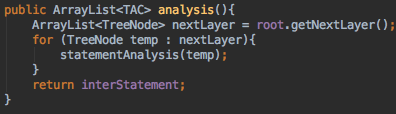
用一个类申明了中间代码的形式，内部有四个变量，都为String类型，在构造时就要进行填充；backfill函数用来对四元式进行回填操作；toString函数用来输出四元式为(a,b,c,d)的形式

1. 语义分析

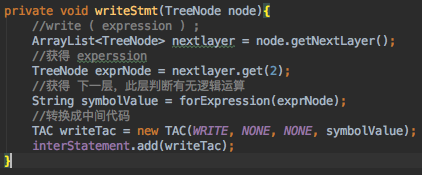
为每一个语法分析产生的非终结符节点申明一个方法，用来进行语义分析，得到四元式。对需要获取非终结符值的节点，需要返回String变量，用来表示遍历得到的终结符或者计算得到的临时变量。所有函数如下：



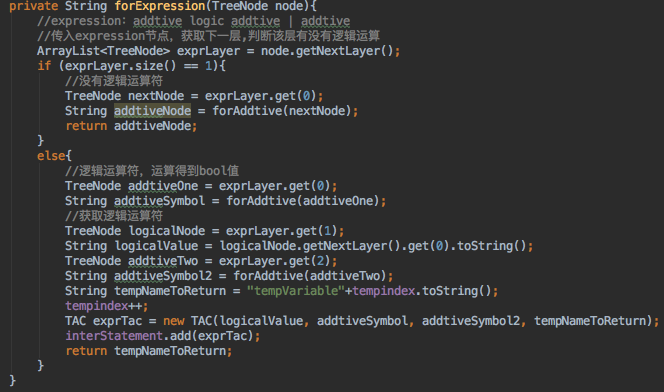
函数入口为analysis，循环调用statementAnalysis来解析statement节点。



这里用write(1 + 2)；作为例子进行阐述，首先语法分析树中该句为一个Statement节点，进一步需要解析writeStatement节点，进入该函数：



我们通过获得下一层的第3个节点，也就是expression节点，可以得到我们要进行打印的值，这是要调用forExpression方法，获取要打印的变量或者数字。

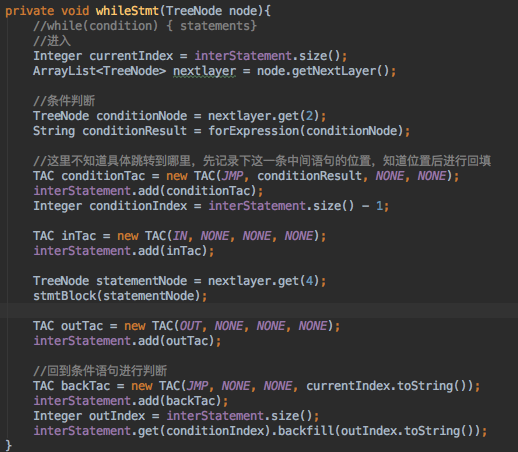


在该方法中，我们需要解析逻辑运算，然后跳转到foraddtive方法解析加减运算得到term + term，然后跳转到forTerm方法中解析乘除运算，最后跳转到forFactor方法中得到我们所要的1 和2，然后回到forAddtive方法中计算1+2的值并赋值给临时变量，添加四元式为(+,1,2, tempvariable0)，然后将tempvariable0返回给forexpression进而返回给writeStatement，添加四元式(write,\_,\_,tempvariable0)。

1. 重点难点分析

在这里解释一下临时变量的赋值操作，定义了一个int型私有变量，用来控制临时变量的不重复，每次申明之后变量加一，通过字符串拼接得到一个临时变量值。

另外需要说明一下回填过程，在while和if语句解析的过程当中，需要需要加入跳转，如果不满足条件，就要跳转到while循环之外，但是我们在语义分析过程中暂时不知道该位置在哪，所以先将address是申明为NONE，在while循环语义分析结束时再对address进行回填得到完整的四元式，代码如下：



对于运算也需进行一些说明，在运算操作时，例如3+5\*4/5-9，得到的四元式顺序是/,\*,-,+计算结果与实际不符，是因为语义分析中按照计算的优先级顺序进行解析（逻辑，加减，乘除），且同等级是从右向左进行解析（因为语法规则中消除了左递归）所以造成了上述错误，所以我们需要对forTerm，forAddtive函数进行改造，直接在函数中进行循环，而不是递归调用，将运算顺序改为从左到右，函数如下：



1. 测试结果（输出产生中间代码）：

