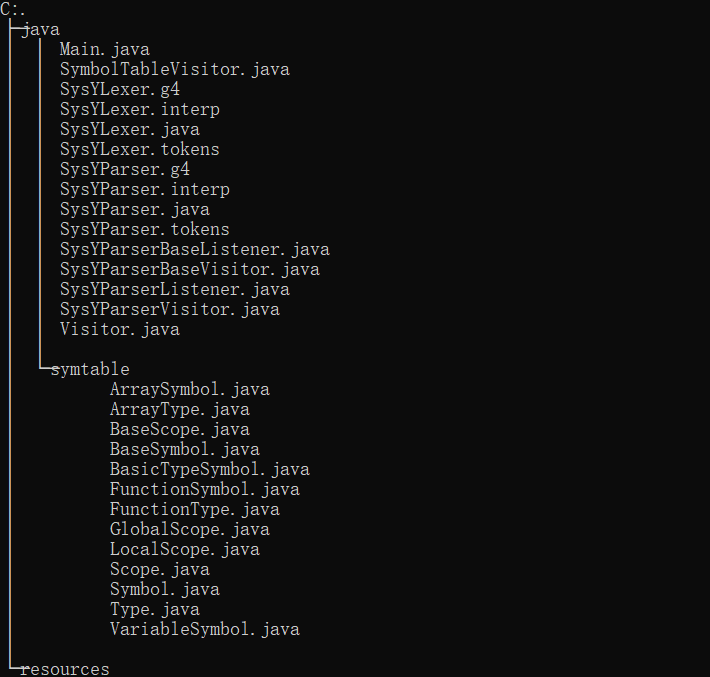
编译原理lab4

1. 作业结构



Symtable包中类的实现参考老师上课所讲代码。

主目录下主要文件有Visitor.java，SymbolTableVisitor.Java，SysYLexer.g4，SysYParser.g4,以及mian.java

Visitor.java 是在lab2中遍历语法树并打印的方法的基础上改进的，新增了重命名的相关方法。

SymbolTableVisitor.Java 是遍历语法树并生成符号表，顺便进行类型检查，获取重命名变量相关信息的类。

Main函数以及两个.g4文件则不过多介绍。

1. 实验主要思想：

首先参考老师上课所讲内容构建符号表以及作用域相关的类放在symtable包中，其中为了方便之后变量重命名，在BaseSymbol类中多加了一个ArrayList<Integer> pos来记录符号所出现的位置（行、列号）。

之后则首先按照老师录播所说使用listener来构建符号表，但在函数定义错误需要跳过其内容时，我发现使用listener不太好处理，而lab2中我已对visitor有些理解。故转而使用visitor进行遍历构建符号表，以及类型检查。

在经历了不断的调整以及debug后（此处省去一万字），终于过了oj，之后则开始写重命名。

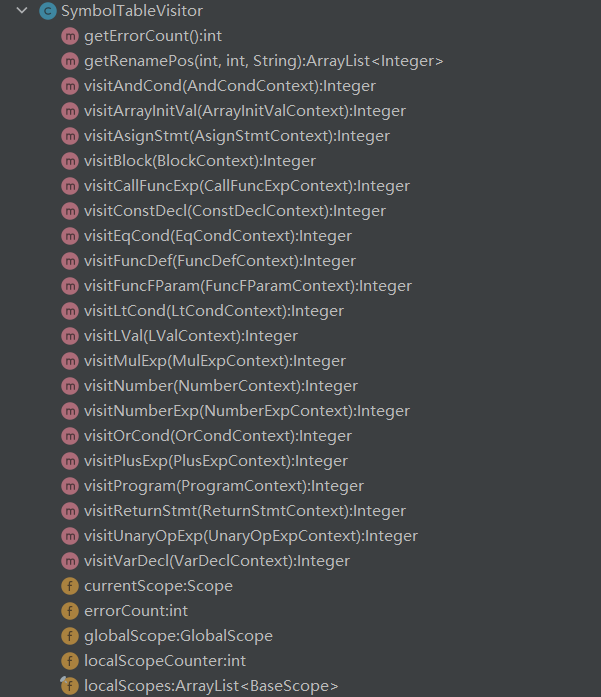
首先我先在main函数中遍历tokens得到所对应应重命名的变量的符号。

之后我便使用SymbolTableVisitor类中新定义的getRenamePos（）函数，遍历作用域得到需要重命名的变量的所有位置，最后再将该位置通过main函数传到Visitor类中，在遍历时对所有Terminal节点都进行判断，检查其位置是否在应重命名的位置上，若是则改变其名字。

1. 具体实现：

SymbolTableVisitor继承自SysYParserBaseVisitor<Integer>,返回值设为Integer，重载了以下方法。绝大多数方法返回值代表其数据结构的维度，比如0为常数，1为1维数组，-1为函数

Null为error。具体细节详见代码。



而Visitor则是在lab2基础上进行改进，其结构相对简单，只在lab2基础上多加了给判断是否需要重命名的判断函数needReplace，以及设定所需重命名变量的位置的setRenamePos函数。

