编译原理 Lab 3 实验报告

学号：201250185

姓名：王福森

我实现的功能

本次实验我通过 antlr 工具，在 SysYLexer.g4 和 SysYParser.g4 文件基础上实现了符号表的构建、实验手册中 11 个错误类型检查、对指定符号重命名这三个功能。

我是这么实现的

本次实验直接沿用 Lab 1 和 Lab 2 所写的 SysYLexer.g4 和 SysYParser.g4 文件。

我先定义的一些数据结构，包括三个接口：Scope、Symbol 和 Type，然后为每个接口实现若干类。

对于作用域 Scope 这个接口，我实现了 BaseScope、FunctionScope、GlobalScope、LocalScope，分别对应作用域的公共属性和方法的基本实现类、函数作用域类、全局作用域类、局部作用域类。上述类用于记录输入程序对应的各个类型的作用域。

对于符号 Symbol 这个接口，我实现了 BaseSymbol、BasicTypeSymbol、FunctionSymbol、VariableSymbol，分别对应于符号的公共属性和方法的基本实现类、基本数据类型符号、函数符号、变量符号。上述类用于记录输入程序各个 Label 对应的符号。

对于 Type 这个接口，我实现了 ArrayType、BaseType、FunctionType，分别对应于整数和整型数组类型、基本数据类型、函数类型。上述类用于记录输入程序变量和常量对应的类型。

对于符号表构建和错误类型检查，我通过构建一个 TypeCheckListener 用于遍历语法树的过程中对不同类型的节点进行符号定义、符号解析和类型检查。这一部分的代码量比较大，需要分很多情况讨论，好在每个情况的处理比较简单。具体而言，需要在进入 program、funcDef、block、constDecl、varDecl 、各个 stmt 的产生式、各个 exp的产生式 和各个 cond 的产生式进行处理，在退出 program、funcDef、block 时进行处理，对此及构建好了符号表并且完成错误类型检查。

如果在上述过程中没有出现类型错误，则会通过 FindTargetSymbolListener，根据输入的行号和列号，定位对应的符号，并且解析出该符号的类型，再将该符号传给 RenameListener，通过遍历语法树得到该符号所有的出现地方，并且将其更名，按照 Lab 2 的方式进行输出。到此完成了符号重命名的工作。

我的精巧设计

我的报错是通过表驱动实现的。

我实现 “最本质错误” 是通过 null 完成的。如果子表达式返回的类型为 null，则表明该子表达式有类型错误，则该层表达式不再报错。

过程中有趣现象和印象很深的 bug

这次 Lab 的代码量是真的大，我 TypeCheckListener 就写了快 600 行，Scope 等三个数据结构也大概快 600 行，功能点很多，这一 de 起 bug 来就没完没了，导致我到最后 ac 都不知道我到底 de 了多少个 bug，只知道大概提交了 30+ 次 /(ㄒoㄒ)/~~。到最后 ac 了往回看，感觉更多是实验要求的很多细节没有注意，代码就漏洞百出，只能说**写作业不能心急呀……实验要求一定得搞清楚了再开始写代码**。我出现的 bug 包括但不限于：

1. 函数定义出了问题整个函数体应该跳过解析；
2. 后面写的解析 if / while 语句，如果该语句处于需要跳过的函数体中则需要跳过，但当时忘记加了；
3. 变量/常量定义初始化语句也得解析；
4. 应该解析所有子表达式再判断要不要返回 null 表示该层出错；
5. 不能解析了左子表达式出错就不解析右子表达式；
6. 没有解析 cond 语句；
7. 操作符（包括 exp 中的和 cond 中的）要求操作数必须为 int；
8. 对于变量重复定义和函数重复定义的要求没有搞清楚，在定义变量的时候应该在冲突域解析，而不是逐层向上解析；在解析变量和函数的时候不能按照类型解析，应该按照名字解析；
9. 函数调用输入的实参如果出错则抛弃，即使实参有错也有可能能成功函数调用；
10. 解析 lVal 的时候如果解析出来是函数也要返回类型，虽然实际上 lVal 应该不能作为 lVal 出现，但报错却应该出现在上层，而不是解析 lVal 层。