Lab3实验报告

201250141 刘屿

实验思路

1. 使用 Listener 进行类型检查，使用 Visitor 进行变量重命名
2. 首先设计**作用域、符号、类型**，这三部分的类结构与继承关系
3. 通过 Listener 中 `program, funcDef, block` 的 `enter, exit` 方法实现作用域的切换
4. 在剩下的文法中进行类型检查
5. 为每个符号定义了一个 `ArrayList<Position>` 用于记录所有出现的位置，并在退出每个作用域时检查有没有符合要求的符号，获取到就把 Symbol 记录下来，作为参数传递给 Visitor 的构造函数
6. 在 Visitor 中对每个终结符判断出现位置是否在需要更换的 Symbol 的位置列表中，是的话就打印新的变量名

精巧的设计

- 将打印错误抽象为单独的函数 `reportError(int type, int line, String msg)`
- 有些函数重复定义过，不需要扫描内容，我为此定义了一个全局boolean变量 `valid`，当进入需要跳过的函数声明时，就将其设为 false，退出再变回 true。只有当 valid 为真时才调用 `reportError`
- 增加 `ParseTreeProperty<Type> typeProperty` 全局变量，用于添加每个节点的类型信息。可以处理 "最本质的错误"，只要获取子节点的类型信息为 null ，就说明子节点内部有错误并在内部处理好了，上层节点就不需要重复处理了
  - 处理到最低层就是 `lVal, functionCall, number` 这三种情况，这三种处理好了上层就很好解决了
- 为类型增加 `int level` 属性，基本类型的 level 为 0，数组的 level 是其 `subType` 的 level + 1，通过 level 来判断维度是否一致，能不能进行赋值操作
  - 通过这个方法可以解决变量和数组所有赋值的情况
- 对于Type类型，添加 `isArray, isFunction` 属性，在类创建时就已经确定了，所以只有接口类型Type的情况下也可以确定具体是哪种类型
- 定义了 `Position` 类型，以及配套的方法，保存出现位置以及检查变得更加方便

遇到的困难及解决办法

- 对于数组类型的变量，不能不限制地进行[]操作，在循环操作中要判断当前维度，以便报错
- 在遇到 `return` 语句时如何获取到当前函数的返回类型
  - 我定义的 `FunctionSymbol` 本身继承自 `BaseScope`，只需要将 `currentScope` 强转即可
  - 但是 `return` 语句可能是在嵌套的代码块中，因此需要通过 `while` 循环以及 `instanseof` 方法得到函数本身的作用域
- 在什么时候得到需要替换的符号
  - 退出作用域时，对当前作用域的符号表进行扫描，检查是否有匹配的符号