Lab3实验报告

1.程序功能

实现基本的类型检查和变量重命名。

其中功能包括：符号表构建、类型检查、利用符号表的变量重命名。

2.实现过程

2.1.项目结构

1.类型类：定义Type类型，用来表达对应的数据类型，包括int、数组、函数类型等。

2.作用域类：用于遍历过程中记录作用域，这是一颗单向的树，它的节点拥有对其父节点的引用。

3.符号类：用来记录符号信息，存放在符号表中。包含变量名、作用域、变量使用的行列信息等。

2.2符号表构建&类型检查

这是本次实验最为复杂的部分。

1.符号表的构建使用visitor对语法树进行遍历。

通过重写各个visit函数来实现。

此处的重点在于在Decl和functionDef中记录声明和定义的变量、函数，存放在符号表中。

2.类型检查。

通过visitor，根据super.visit在函数中的位置来实现对节点的访问和退出。

通过进入函数、block等，生成新的作用域，通过退出这些节点来回到父作用域，通过这一过程来确定符号的作用域。然后在遍历过程中，实现不同类型的类型检查。

2.3重命名

重命名采用的是：

第一次遍历，构建符号表的过程中记录每一次该变量出现的行列信息。

第二次遍历，进行输出时，从符号表中查询待重命名的符号，并且判断当前符号的行列位置是否包含于该变量出现的位置，如果是，那么就输出其新名称。

这个过程的重点在于要正确记录所有变量出现的位置，存放在符号表中。

3.遇到过的bug&处理

类型检查的过程是极其繁琐的。所以过程中遇到了很多的bug：

1.常见的空引用错误，因为在类型检查的过程中，每个语句可能有大量不同的分支，因此这个过程中，某些节点不一定具有某些特定种类的子节点，如果不做null非空判断的话，很容易产生空引用错误

2.对分支的检查。这里使用的是instance of来判断某一基类是否可以转型为它的某种派生类，通过这个方法来具体判断每个expr究竟对应了哪个分支，这一在获取expr的类型时就可以利用它进行递归。

3.函数作用域的问题。函数的Fparam作用域和后面的block是同一作用域，因此应该进行特殊处理。处理方法：增加一个标志位，标志该block是否属于某个函数的函数作用域而非localScope。

4.符号表的结构。采用的是HashMap<String,Symbol>，它的键是作用域+符号名构成的字符串，这样在遍历过程中查询时只需要构建一个简单的字符串拼接函数getSymbol(String id)就可以得到对应的符号。

5.类型的设计。采用了PrimaryType,ArrayType,FunctionType,并且定义了一个空的UndefinedType。由于类型判断非常常用，所有需要抽象出一个判断任意表达式类型的函数getExpType(ExpContext exp)，通过递归，来获取任意表达式的类型。并且这其中需要区分null和undefined，null意味着递归过程中返回了err即不匹配问题，需要报错。undefined意味着出现了未定义的变量，不需要报错，应该直接跳过。

6.变量的问题：变量是否重定义只需要判断同一作用域下是否重名，而不要判断全局作用域。这意味着局部变量会覆盖全局变量。