编译原理 Lab2 报告

171860659 吴紫航 联系邮箱：401986905@qq.com

实验目的

在词法分析和语法分析程序的基础上编写一个程序，对 CMM 源代码进行语义分析和类型检查，并打印结果。实验的代码将手写完成。

实验要求

1.检查出语义错误类型 1-17：即不符合假设和语义的错误
2. (任务号为 6) 选作要求 2.2：即变量的定义受到嵌套作用域影响。

整体思路

1）在词法和语法分析后，如果没有错误，则继续调用语义分析的模块 semantic
2）语义分析的过程是对语法分析树进行语义检查。本质是一个 DFS 遍历的过程，遍历的过程中，通过传递参数，并对参数进行识别、对比、确认等操作，以发现语义错误。
3）在检查过程中，除了参数传递，还需要实现符号表的插入、查询等管理，以及类型的结构体表示等。
4）核心的实现脉络就是 17 种错误类型，过程比较繁琐。需要逐一的在适当位置检查这 17 种错误类型。完成后再实现选作 2.2，即作用域相关处理。

数据结构

1）HashType

枚举结构，用来区分 hash 节点的类型。分别为：变量、结构体、函数三种 hash 节点

2）TypeKind

枚举结构，用来区分变量的类型。分别为：基础变量、数组变量、结构体变量

3）Type

用来表示变量的类型，除了 TypeKind 标识以外还有具体的类型信息

4）FieldList

用来存储结构体的各个域，通过链表的方式

5）Structure

用来表示结构体，包含结构体名、结构体 id（匿名结构体专用）、域链表 FieldList

6) Variable

用来表示变量，包含变量名和类型

7）Function

用来表示函数。包括函数名、返回类型、参数列表、参数个数

8）HashNode

用来表示 hash 表的一个桶中的一个节点。需要有一个成员 HashType 作为标识、具体的节点内容（是变量或结构体定义或函数定义）、行号、深度（用来处理作用域信息）、以及 next 指针，指向本桶的下一个节点。

9）HashList

本结构内部成员有一个 HashNode，本结构作为一个链表节点，目的是把同一层的 HashNode 给串联到一起。

10）HashListHead

作为 HashList 的头，内部成员既有本层 hashList 的第一个节点指针、又有下一个 HashListHead 的指针。是一个连接纵横的结构。

关键步骤

本实验的实现比较繁琐，但是思想比较单一：围绕着规定错误类型进行全面的检查并报错。定义时，写符号表；使用时，读符号表。这些显然的步骤就不再赘述。以下介绍本次实验过程中，遇到的比较关键的难点和解决方案。

1）符号表

```
struct HashNode* HashTable[HASH_LENGTH];
```

采用了闭地址哈希表的形式。兼顾有高效和易实现的特点。哈希函数采用了手册上的写法。**在插入时，采用头插法，保证读符号表的时候先发现的是最新被定义的节点。**同时检查变量重定义的时候 ，只寻找同一层的节点，保证作用域内外层区分的目的。

2）匿名结构体

因为匿名结构体的名字是独一无二的。因此，单独在结构体中设置一个成员 id。改成员只有匿名结构体才需要使用。id 表示该结构体是已发现的第几个被定义的匿名结构体。如果是非匿名结构体，则 id 为 0。这样就可以达到匿名结构体的名字是独一无二的目的。比较结构体相同时，如果发现结构体是匿名的（name 是 null）则比较 id 即可。
当然此时需要一个全局变量 anonymousStrtNum 记录已经定义了几个匿名函数

3）作用域嵌套

作用域嵌套的实现，采用了手册上描述的，十字链表和 open Hashing 的方法。维护风格是 Imperative Style。在发现块{}以后，在检查块前。把全局变量 currentDepth 增 1。并且在写符号表时，把同一层的节点同时连接一个链表上。也是头插法，保证将来先删除的节点是最新被定义的。当块结束以后，调用 removeCurrentDepth，删去当前深度的哈希节点的定义记录。并且把 currentDepth 减 1。

4）函数定义和参数定义的难点

在实现完作用域嵌套后发现有个小 bug。那就是函数参数看作是作用域在函数体内的变量。函数参数的深度应该是 1，而函数的深度是 0。这就意味着函数应该先于函数参数被定义。但是只有在函数参数确认以后，才能确认函数的结构，因此产生一个矛盾。也就是函数在函数参数之后完成定义，而此时深度已经设置为 1 了。采取的解决方案是，对于深度为 0 的 hash 节点，不进行 hashList 纵向连接，因为 0 层的定义是不用在结束语义分析前删除的。此外，当发现插入的是函数节点时，可以特殊设置其深度为 0，并且断言 assert 当前深度肯定是 1（因为定义完参数后马上定义了函数）

5）数组变量的生成

数组类型的变量，在生成过程中也算是一个难点。**值得庆幸的是，数组变量的文法是左递归的**：VarDec->VarDec' LB INT RB。

这样在生成数组类型变量的时候，可以从最右边向左边传递参数。最开始的类型是继承属性，已经被 VarDec 在调用时接收为函数参数。然后类型就变成一个数组类型，其元素类型是刚刚的继承属性参数；其元素个数是文法的 INT。然后该数组类型生成后作为新的继承属性，传给 VarDec'。这样在 VarDec'也类似的进行处理。直到 VarDec->ID。这样 ID 是一个变量名，其类型作为继承属性传入。生成一个数组变量实体后，该变量通过综合属性返回即可。

结果测试

测试的方式是直接用助教提供的 Makefile(没有任何修改)

输入 make clean

输入 make parser

输入 make test

得到测试结果。

注意：因为助教提供的原 makefile 里只对 test1.cmm 进行了测试，所以测试文件的名字是 test1.cmm。若想对其他命名的 cmm 进行测试就需要更改 makefile

补充说明

小组任务编号为 6、选作部分为 2.2。吴紫航 171860659 联系邮箱：401986905@qq.com