编译原理实验二实验报告

171860607

白晋斌

810594956@qq.com

任务号:14

目录

实验要求

实验二要求基于 7 种假设(假设 3.4.5 可能因后面的不同选做要求而有所改变).程序需要对输入文件进行语义分析(输入文件中可能包含函数、结构体、一维和高维数组)并检查共 17 种错误.

此外,我们需要完成要求 2.2,即修改假设 4 为:变量的定义受可嵌套作用域的影响,外层语句块中定义的变量可在内层语句块中重复定义(但此时在内层语句块中就无法访问到外层语句块的同名变量),内层语句块中定义的变量到了外层语句块就会消亡,不同函数题内定义的局部变量可以相互重名.

值得注意的是,1.数组类型的等价,只要数组的基类型和维数相同我们就认为类型等价.2.允许类型等价的结构体变量之间的直接赋值.3.结构体类型的等价,每个匿名结构体类型我们认为均具有一个独有的隐藏名字.

测试环境

测试环境如下,值得注意的是,gcc 版本为 5.5.0,与助教测试机接近,不再是讲义所述的上古 gcc 版本.

~~gcc version 4.7.4 (Ubuntu/Linaro 4.7.4-3ubuntu12)(~~实验一~~)~~

gcc version 5.5.0 20171010 (Ubuntu 5.5.0-12ubuntu1~16.04)

flex 2.6.0

bison (GNU Bison) 3.0.4

运行方法

make clean; make; ./parser xxx.cmm

实验步骤

1.数据结构

数据结构采用讲义所述,并进行适当拓展.更详细的解释在注释中体现.

其中,符号表使用 FieldList 作为基本数据单元,我们选择将所有符号组织成一张表,这里选择了散列表作为符号表的数据结构,hash 函数的算法选择了 P.J. Weinberger 提出的算法.并使用了 open hashing 的方法作为处理散列表冲突的方式.

此外,由于实验需要支持多层作用域,这里我们采用了 Imperative Style 作为符号表维护风格,并设计了一个基于十字链表和 openhashing 的改进后的散列表,对比讲义的区别主要在于我们关于层深的 FieldList 与 Hash table 并无联系,而是单独维护本深度的符号表.简单说,就是维护一个全局变量 curFuncDep,每当遇到 CompSt 结构的语句块,进入前 curFuncDep+=1,并构造一个本深度的 FieldList 头,当在这个语句块中遇到新的变量定义时,把该变量插入到符号表中,并用 FieldList 头串起来.当语句块中出现任何表达式使用某个变量,使用 hash 函数直接查询. 退出后 CompSt 语句块后,将当前深度的 FieldList 头所串连起来的符号全部从表中删除,再令 curFuncDep-=1.

2.语义分析

借助实验一建立的语法树,我们可以以深度优先遍历的方式遍历语法树,在遍历过程中进行语义分析以及错误检查. 更详细的解释在注释中体现.

实验主体代码在 lab2.h 以及 lab2.c,为便于实验二展开,我们将实验一中的数据结构抽离出来单独形成 lab1.h 的文件.

之后,我们按照附录 A 的语法规则,基本将每条语法规则都封装成函数,以 Type 和 FieldList 为主要接口.

2.错误检查

每次在插入符号表前执行查找操作,以及在通过 typeCheck 函数进行类型检查,通过对不同情况的考虑,我们就可以检查出 17 种错误. 更详细的解释在注释中体现.

对于一些未定义错误以及与假设矛盾的输入,我们尽可能地做了兼容处理,基本原则是未定义的错不乱报.

此外考虑到程序要能够同时检查出多种错误,在发现错误的某些无法挽救的情况我们会返回 NULL,之后层层向上直到某个可以接受 NULL 的语法(一般是在 SEMI 之后).

值得注意的是,假设 7:结构体中的域不与变量重名,并且不同结构体中的域互不重名.基于此,我们对违背假设 7 的输入实现了错误检查,但注释掉了报错(不报错),由此基于不同实现,可能与完全不做检查产生输出上的差异.

遇到的问题

一路遇到很多问题,这里挑选一部分经典问题与对应解决方案链接列出:

https://blog.csdn.net/betty13006159467/article/details/78394974

https://blog.csdn.net/qunxuan/article/details/40887643

https://blog.csdn.net/huoyin/article/details/7206800

https://blog.csdn.net/xinyu1026/article/details/78703449

https://blog.csdn.net/jq_ak47/article/details/52865113

https://blog.csdn.net/qq_42206669/article/details/90143961