南京大学

计算机科学与技术系

编译原理实验报告

实验名称：L2-语义分析

组长学号：211830115

组长姓名：郑九铭

组员学号：211180213

组员姓名：胡德谐

实验时间：3.20-4.19

报告撰写：胡德谐

一、 程序功能介绍

1. **必做部分**

本实验完成了对于给定程序，分析其语义结构，从而检查出程序中的语义错误并输出对应的错误类型及行号。若程序无错误，则不需要输出任何内容。

其中，检测语义错误通过在semantic.c中实现语义分析，并从中识别出语义错误。同时我在semantic.h定义了Type_ 、FieldList_ 、Structure_ 、Function_ 、Node_等数据结构，从而帮助构建了哈希表、链表来实现语义分析。

2. **选做部分**

本实验完成了全部选做部分的要求。为：

（1）实现函数声明。

（2）实现变量在可嵌套作用域下的定义。

（3）实现结构体的结构等价。

实现方式均为在语义分析过程中对上述选做部分的部分进行特殊识别与处理。

3. **实验亮点**

本人认为该实验的设计亮点有二：

- 安全便捷的内存复制函数：

本实验中定义了下列函数Type _ *copy_type(Type _ *type)、FieldList_ *copy_field_list(FieldList_ *type) 、char *copy_name(char *name)其通过malloc()函数分配新的内存，再将对应的值与指针一一复制 ，与memcpy ()函数相比更加安全，便于进行内存管理。（具体代码见semantic.c 文件中对以上函数的定义与使用）

- 利用git.nju.edu.cn代码托管平台进行代码管理：

  本实验由于是多人组队，因此采用了南京大学提供的代码托管平台进行代码管理，这样做便于代码版本维护和代码传输，提高了合作效率，同时降低了代码出现不可逆转的问题而无法恢复的风险。

二、 程序编译方式

1. **编译**

   在命令行中执行以下指令：

   ```
   cd CODE_PATH
   ```

   这里 CODE_PATH 是指项目文件夹 Code 所在的路径，确保执行后所处的位置在 Code 文件夹中；

   ```
   make
   ```

   此时即完成了对代码的编译，可以看到可执行文件 parser 已经生成。

2. **运行**

   在命令行中执行以下指令

   ```
   ./parser ../Test/test1.cmm
   ```

   即可分析test文件夹中的test1.cmm程序。分析其余程序同理，只需将./parser后的路径修改为待分析程序的路径即可。

三、 个人感想与问题

1. **个人感想**

   实验难度适中，但是工作量较大，过程较为繁琐，需要花费大量时间完成。

2. **问题**

个人认为实验指导手册中对于某些可能遇到的输入情况下程序应该表现的行为是未定义的，这大大增加了不必要的实验难度，如：

（1）函数名和变量名相同是否算语义错误?

（2）变量、函数、结构体在可嵌套定义域中的定义有那些不同?

这些问题虽然已经解决，还是在前期增加了不必要的工作量。