Lab2实验报告

赵铭南 161271033

一、完成的功能

- 语义分析：

  可对输入的*.cmm文件进行语义分析，并检查如下类型的错误：

  1. 错误类型1：变量在使用时未定义。
  2. 错误类型2：函数在调用时未经定义。
  3. 错误类型3：变量出现重复定义，或变量与前面定义过的结构体名字重复。
  4. 错误类型4：函数出现重复定义。
  5. 错误类型5：赋值号两边的表达式类型不匹配。
  6. 错误类型6：赋值号左边出现一个只有右值的表达式。
  7. 错误类型7：操作数类型不匹配或操作数类型与操作符不匹配。
  8. 错误类型8：return语句的返回类型与函数定义的返回类型不匹配。
  9. 错误类型9：函数调用时实参与形参的数目或类型不匹配。
  10. 错误类型10：对非数组型变量使用“[…]”（数组访问）操作符。
  11. 错误类型11：对普通变量使用“(…)”或“()”（函数调用）操作符。
  12. 错误类型12：数组访问操作符“[…]”中出现非整数。
  13. 错误类型13：对非结构体型变量使用“.”操作符。
  14. 错误类型14：访问结构体中未定义过的域。
  15. 错误类型15：结构体中域名重复定义（同一结构体中），或在定义时对域进行初始化。
  16. 错误类型16：结构体的名字与前面定义过的结构体或变量的名字重复。
  17. 错误类型17：直接使用未定义过的结构体来定义变量。

二、实现方法

- 语义分析：

  在实验一所构建的语法树的基础上，通过对语法树进行遍历完成对符号表的构造以及数据类型的构造和检查。采用的做法是Bison代码只用于构造语法树，另建文件`semantic.c`来完成语义分析的相关操作，主要由`void scanning(struct treenode* r)`完成。

  语义分析过程主要有两部分：符号表创建以及类型检查。

  符号表创建：

  - 每当遇到语法单元ExtDef，说明定义了全局变量或结构体或函数：
    - 全局变量：调用`Type gettype(struct treenode* specifier)`解析出变量类型，再调用`void insert(struct treenode* node, Type type)`将相关变量以解析出的类型存放进符号表(`struct VarTable* head`开头，`struct VarTable* end`结尾)；

        - 结构体：调用`FieldList getstructure(struct treenode* node, FieldList fks)`解析出结构体的结构，再调用`void addStru2Table(char* name, Type type)`将结构体类型存放进结构体类型表(`struct StruTable* StruTable_head`开头，`struct StruTable* StruTable_end`结尾)，如果同时定义了结构体变量，那么将定义的变量加入符号表；
        - 函数：调用`Type gettype(struct treenode* specifier)`解析出函数类型，类似结构体的做法将函数参数当作域组织成链表，最后调用`void addFunc2Table(char* name, Type type, int varnum, FieldList structure)`将解析出的函数及其参数结构存入函数表(`struct FuncTable* FuncTable_head`开头，`struct FuncTable* FuncTable_end`结尾)；
    - 每当遇到语法单元Ext：
        - 调用`Type gettype(struct treenode* specifier)`解析出变量类型，再调用`void insert(struct treenode* node, Type type)`将相关变量以解析出的类型存放进符号表(`struct VarTable* head`开头，`struct VarTable* end`结尾)。

类型检查：

- 每当遇到语法单元Exp，说明会对变量或者函数进行调用：
    - 函数`int check(struct treenode* exp)`检查语法单元Exp表达式是否符合语义；
    - 函数`Type returntype(struct treenode* exp)`返回符合语义的表达式Exp的类型；
    - 函数`int typecmp(Type t1, Type t2)`判断两个类型是否相同。

三、编译运行方法

- 环境：

```
GNU Linux Release: Ubuntu 18.04, kernel version 4.15.0-47
GCC version 7.3.0
GNU Flex 2.6.4
GNU Bison 3.0.4
```

- 编译：

  进入目录`./Code`，终端下输入：`make`，或：

```
flex lexical.l
bison -d syntax.y
gcc tree.c syntax.tab.c semantic.c main.c -lfl -ly -o parser
```

- 运行：

  终端下输入：`./parser [filename]`

四、实验总结

实验二完成了编译器的语义分析的基础功能，能够对C--源代码进行语义分析和类型检查，加深了对编译原理中语义分析过程与类型检查的理解。