编译原理实验 2 实验报告

161220143 吴御洲

一、概述

在实验 1 对 c—代码进行词法分析和语法分析的基础上，进行了语义分析。可以找出程序中不符合实验要求给定假设的语句，并打印出错误类型和错误所在行号。

此外选作部分实现了选作 2.3，将结构体的类型等价机制由名等价改为结构等价。

二、文件结构

lexical.l syntax.y syntaxtree.h syntaxtree.c 文件功能与实验 1 一致；

semantic.h 文件中定义了 Type 和 FieldList 两个结构体来表示 c—语言中的类型和作用域，定义了对符号表进行处理的函数，定义了语义分析用的函数以及一些用于调试的函数，如打印符号表的函数 printTable;

semantic.c 文件中具体实现了.h 文件中的函数。

三、具体实现

Type 和 FieldList 的定义与书上 60 页的例子基本一致，将 kind 的枚举拿到 Type 的结构体之外；在 Type 内的 union 中加入一个 function 结构体，来表示函数，包括函数返回类型、参数列表和参数个数。

FieldList 中的 offset 原意是为了记录本符号在符号表中的位置与 hash 初始位置的差，但后面发现实际没法使用。

符号表为一个大小为 65536 的 FieldList 的数组，根据 hash 值插入；如果当前插入的为一个函数的话，计算 hash 值时的参数+1，以示区别，如果当前位置已经被占，则顺延到下一个位置。

根据名字去符号表中查找的时候额外加一个参数用于判断是否为函数，以便区分同名的函数与变量/数组/结构体；若找到，返回该符号的 FieldList，反之返回 NULL。

isSameType 函数判断两个 Type 是否同一类型时分情况，如果一个为 NULL 或者 kind 不一样，说明不是同一类型，返回 0；如果都是 BASIC，看结构体中的 basic 是否一致；如果都是 ARRAY，判断 array 中的 elem，即数组类型是否一样；如果都是 STRUCTURE，调用 isSameFieldList 函数判断两个的联合体中的 union 是否等价；如果都是 FUNCTION，先后判断返回值类型和参数个数是否一样，然后也调用 isSameFieldList 函数判断参数列表是否等价。

isSameFieldList 函数判断两个 FieldList 是否同一类型时分情况，首先如果有一个为 NULL 的话，则不等价，返回 0；然后以两个同时不为 NULL 为 while 循环条件，里面比较 type 成员是否一致，若不一致返回 0，然后分别指向 tail；退出 while 后，如果有一个不为 NULL，说明两个 FieldList 长度不一样，不等价，返回 0，反之返回 1.

之后的函数基本都是对于语法树的处理。除了 OptTag 和 Tag 外，每个产生式设一个函数，函数名与语法单元同名。重点处理的有：

VarList，涉及到函数的参数；

Args，涉及到函数调用的实参；

DefList，涉及到结构体内部的成员；

这几个的构造过程中基本是顺序进行，新得到的加在 tail 后面。

对 Exp 的处理中重点有：

Exp ASSIGNOP Exp 需要对左边进行判断，对 ID、Exp LB Exp RB、Exp DOT ID 以外，即变量、数组成员、结构体成员以外的报错误类型 6；

对于操作符两端表达式是否等价的判断，为了防止出现如一个未定义变量没有类型导致的一连串报错，设定如果左右有一个类型为 NULL，即未定义，则不报错误类型 7，此时返回类型为不为 NULL 的那个；

对于 Exp RELOP Exp，返回类型必定为 INT。

四、如何运行

所有文件放在/Code 下，在/Code 输入 make parse 后生成可执行文件，之后将 xxx.cmm 测试文件也放在/Code 下，输入./parse xxx.cmm，即可显示结果。

五、效果展示（书上例子）

例 2.1

```
wyz@ubuntu:~/Lab/Code$ ./parser test.cmm
Error type 1 at Line 4: Undefined variable 'j'
wyz@ubuntu:~/Lab/Code$
```

例 2.2

```
wyz@ubuntu:~/Lab/Code$ ./parser test.cmm
Error type 2 at Line 4: Undefined function 'inc'
wyz@ubuntu:~/Lab/Code$
```

例 2.3

```
wyz@ubuntu:~/Lab/Code$ ./parser test.cmm
Error type 3 at Line 4: Redefined variable ‘i’
wyz@ubuntu:~/Lab/Code$
```

例 2.4

```
wyz@ubuntu:~/Lab/Code$ ./parser test.cmm
Error type 4 at Line 6: Redefined function 'func'
wyz@ubuntu:~/Lab/Code$
```

例 2.16

```
wyz@ubuntu:~/Lab/Code$ ./parser test.cmm
Error type 16 at Line 6: Duplicated name ‘Position’
wyz@ubuntu:~/Lab/Code$
```

例 2.17

```
wyz@ubuntu:~/Lab/Code$ ./parser test.cmm
Error type 17 at Line 3: Undefined structure ‘Position’
```

例 2.19

```
wyz@ubuntu:~/Lab/Code$ ./parser test.cmm
Error type B at Line 6: Syntax error.
Error type B at Line 8: Syntax error.
wyz@ubuntu:~/Lab/Code$
```

例 2.20

```
wyz@ubuntu:~/Lab/Code$ ./parser test.cmm
Error type 3 at Line 9: Redefined variable ‘i’
wyz@ubuntu:~/Lab/Code$
```

例 2.22

```
wyz@ubuntu:~/Lab/Code$ ./parser test.cmm
wyz@ubuntu:~/Lab/Code$
```

例 2.23

```
wyz@ubuntu:~/Lab/Code$ ./parser test.cmm
Error type 5 at Line 16: Type mismatched for assignment
wyz@ubuntu:~/Lab/Code$
```

其余例子也经过测试，和书上结果一样。