2019 年 11 月 1 日 星期五

编译原理实验二　实验报告

171860689 蔡跃龙 857570220@qq.com
171860673 林昊 1184264181@qq.com

实验内容

本次实验任务是在词法分析和语法分析程序的基础上编写一个 C 语言程序，对 C—源代码进行语义分析和 17 种错误类型检查，并打印分析结果。同时要求完成选做 2.1——任何函数除了在定义之外还可以进行声明，还需检查新的错误类型 18 和 19。

程序实现功能以及过程

- 符号表

我们采用了散列表形式的符号表，散列表的冲突解决方案为开散列。另外，散列表提供 hash_pjw、find_item、add_item 函数分别供程序计算散列值、查找表项和插入表项。散列表的具体实现参见 hashTable.c 文件。

- 语义分析

语义分析部分考虑保持实验一当中编写的 Bison 代码用于构造语法树，而把和语义分析相关的代码都放到单独的文件 semantic.c 中。在 semantic.c 文件中，我们将语法分析使用的文法中的每一个非终结符号具体化为一个函数，而不同的函数有不同的形式参数用以传递继承属性（如果有的话），也有不同的返回值用以传递综合属性。当然，为了能够完成对语法分析所生成的语法分析树的遍历，每一个函数内部都安排有适当的位置对语法分析树的子节点所代表非终结符号函数进行调用。

对每一个函数的编写，我们都遵循了同一的标准：先检测当前结点的子节点个数，然后对子节点的名称进行比较。在以上所有条件都符合的情况下方可参照对应的产生式进行语义分析和类型检查，否则输出错误并 assert(0)方便错误检查。

函数当中除了需要完成语义分析的语法制导翻译功能，还需要将变量、数组、结构体和函数的 ID 加入符号表。由前述提供的符号表接口，我们可以在适当的地方调用之来完成语义分析和符号表的交互动作。

另外需要考虑完成选做 2.1：在 syntax.y 中编写新的产生式 ExtDef->Specifier FunDec SEMI 即可允许程序进行函数声明。需要注意的是，我们为函数类型新定义了 Function 结构体，在其中添加关键的 isDef 域以辅助我们判断当前函数是否定义。

- 类型检查

程序的类型检查被嵌入各个产生式函数当中。在程序的执行过程中，我们增加的条件分支可以在适当的时机对可能存在的语义错误进行检查并报错。另外需要注意我们新增了 equalJudge.c 文件，其中编写了用于判断类型相等和结构相等的函数。

程序编译方法

- 程序编译

命令行进入 Lab/Code 目录，依次键入 make clean、make、make test 命令并回车即可生成 parser 处理程序并测试相应测试文件。另外，如果进入 Makefile 文件进行编辑，可以测试不同的测试文件。

程序设计创新

- 工程设计

新编写的程序文件按照模块分成三个文件 hashTable.c、semantic.c、equalJudge.c 并引用同一个头文件 semantic.h，这样能够在保持相似功能的高聚集性的同时，维护相异功能的互斥性。

- 程序测试

在程序的开发过程中，曾使用多达 45 个测试样例对程序的鲁棒性进行检测。