编译原理实二实验报告

马珩峻 林立强

一、程序功能

实现了对 c--语言的语义分析，除去基本要求的 14 中语义错误，完成了选做 2 的内容，对变量的作用域进行了限制。

采用的数据结构为 hash 表存储变量符号加十字链表确定作用域的方法。Hash 函数使用了讲义中提供的 P.J.Weinberger 提出的 hash 函数。

相关数据结构定义如下

```
typedef struct fieldList {
    char* name;       // 域的名字
    pType type;       // 域的类型
    pFieldList tail;  // 下一个域
} FieldList;

typedef struct tableItem {
    int symbolDepth;
    pFieldList field;
    pItem nextSymbol;  // same depth next symbol, linked from stack
    pItem nextHash;    // same hash code next symbol, linked from hash table
} TableItem;

typedef struct hashTable {
    pItem* hashArray;
} HashTable;

typedef struct stack {
    pItem* stackArray;
    int curStackDepth;
} Stack;

typedef struct table {
    pHash hash;
    pStack stack;
    int unNamedStructNum;
    // int enterStructLayer;
} Table;
```

此外为了区分函数符号以及结构体的 dec 和 def，在讲义中 type 的基础上进行了修改，存储了结构体和函数的信息，如果是结构体的 def 的话结构体名会存储于上一层的 FieldList 中，type 中的 structname 为 null，如果为 dec 的话，上一层的 FieldList 中会存储变量的名字，而此处的 type 中会存储变量结构体类型的名字。对于匿名结构体，因为词法分析中规定了 id 不能为纯数字，所以此处我们用数字编号作为匿名结构体的名字，可以避免与其他符号的命名冲突，数字编号通过在 table 结构体中记录匿名结构体的个数来获得编号。

```
typedef struct type {
    Kind kind;
    union {
        // 基本类型
        BasicType basic;
        // 数组类型信息包括元素类型与数组大小构成
        struct {
            pType elem;
            int size;
        } array;
        // 结构体类型信息是一个链表
        struct {
            char* structName;
            pFieldList field;
        } structure;

        struct {
            int argc;           // argument counter
            pFieldList argv;    // argument vector
            pType returnType;   // returnType
        } function;
    } u;
} Type;
```

代码中将结构体看做 oop 语言中的类，对每个结构体都封装了对应的操作方法，包括基本的 new（构造新的结构体） delete（释放结构体内存空间） copy（拷贝结构体（均为深拷贝））以及部分结构体特有的方法，比如 Type 的 checkType（类型检测） 等。通过封装会频繁调用的功能函数来方便后面的语义分析。

语义分析中，当程序碰到符号需要处理时必定是进入了 ExtDef 后，所以我们的程序遍历语法树，遇到 ExtDef 节点便从该节点开始进行语义分析，然后对 ExtDef 展开会得到的所有非终结符节点进行处理对符号表进行操作以及查找语义错误。

二、使用课程网站提供的 Makefile 进行编译运行即可