lab2

211275009 陈铭浩
211275009@smail.nju.edu.cn

实现功能

- 对C--源程序进行词法分析和语法分析，并生成语法分析树（lab1内容），在语法分析树的基础上，进行语义分析，可以检查出包括且不限于要求的17种语义错误（逻辑错误）
- 完成选做2.3

实现思路

1. 首先确定了符号表的数据结构，我选取了讲义推荐的散列表（哈希表），并使用闭合地址法（表项里挂链表）解决冲突问题，表项信息包括name、type以及当前格内的下一表项的地址。type是一个结构体，其中详细记录了符号的类型信息。这里需要注意type中区分了structure（结构体）和struct_def（结构体类型定义），但他们的具体信息都是用union中的structure存的

```c
struct FieldList_{
    char* name; // 域的名字
    Type type; // 域的类型
    FieldList tail; // 下一个域
};

//define symbol table entry
typedef struct SymbolTableEntry {
    char* name;
    Type type;
    struct SymbolTableEntry* next;
} SymbolTableEntry;
```

2. 定义好符号表后，自上而下的对语法分析树进行遍历，过程中正确的insert符号，find符号，并分析语义错误即可。这里注意因为允许函数和变量重名，所以find函数会多传入一个参数，表示要查询的是一个函数名还是变量名

```c
//function:type=1   others:type=2
//函数可以与变量重名
SymbolTableEntry* find(char* name, int type) {
    if(name==NULL) return NULL;
    unsigned int index = hash_pjw(name) % HASH_SIZE;
    SymbolTableEntry* entry = hashTable[index];
    while (entry != NULL) {
        if (strcmp(entry->name, name) == 0) {
            if((type==1&&entry->type->kind==FUNC) || (type==2&&entry->type->kind!=FUNC)){
                return entry;
            }
        }
        entry = entry->next;
    }
    return NULL;
}
```

3. 对于类型匹配，单独定义一个函数typeMatch，其中：只要数组的基类型和维数相同我们即认为两个array类型是匹配的；结构体类型等价需要针对结构体中的每个域逐个进行类型比较（选做2.3：结构等价)；函数等价需要函数返回值类型，参数个数和参数类型均相同

4. 本实验感觉相比lab1简单，但debug的过程（主要通过printf）十分痛苦，我最常遇到的bug：Segmentation fault (core dumped)，这主要是因为访问了空指针，这也警醒我在使用指针时需要格外注意

编译方式

```
在Lab2/Code文件夹下
$ make parser
然后用Code/parser替换Lab2下的parser（当然，在提交的版本中我已经替换过了）
在Lab1文件夹下执行
$ ./parser <测试文件路径>
```