Lab2 实验报告

学生信息

姓名：李子沐；学号：221180029；邮箱：221180029@smail.nju.edu.cn

实验环境和编译方法

实验环境与OJ要求相同，使用makefile编译即可。

实现功能和细节

完成实验要求的必做和选做部分。数据结构方面，我设计了哈希表和堆栈维护符号表和作用域，包括：

1. 基于十字链表SymbolTableNode，在“纵轴”上用HashTable维护，在“横轴”上用ScopeHeadStack维护。“纵轴”符号表，为哈希表，“横轴”作用域为栈
2. 基于普通链表的StructureTable和FunctionTable以完成选做要求的。

```
struct SymbolTableNode {
    Type *type;
    char *name;
    int kind;                           // 0 for variable, 1 for struct
    SymbolTableNode *nextBucketNode;    // next node in the symbol table, linked list insered at th
    SymbolTableNode *nextScopeNode;     // next node in the same scope
};
struct ScopeHead {
    SymbolTableNode *ScopeHeadNode;     // hash table for this scope
    ScopeHead *nextScope;               // next scope in the stack
};
struct ScopeHeadStack {
    ScopeHead *top;                     // top of the stack
};
/* Hash Table is an array of bucket(pointer) */
struct HashTable {
    SymbolTableNode *buckets[TABLESIZE];
};
```

基于这些数据结构，表类型里实现了必要的`create`、`insert`、`get`，栈类型里实现了`create`、`push`、`pop`。由于删除节点本质上就是一个作用域的终结，因此弹栈也代表着删除符号表的一系列节点。

在实现Compst的块作用域时，本来打算在Compst分析内部进行push和pop，但是考虑到函数体的定义，从形参表开始就属于新的作用域，因此对于Compst的作用域全部在调用前进行push和pop，例如

```
scope_begin();// a new scope of FUNCTION begin at Formal Param
FunDec(node1, TRUE, type);
CompSt(node2, type);
scope_end();
```

其余的语义分析，并没有特别的内容，重点是仔细考虑每个细节即可。
实际调试时发现一些问题：

1. 我在lab1中用数组来实现多叉树的多个孩子，因此产生的空串仍然会占用一个，例如A->BCD，如果C产生了空串，那么我仍然要通过访问第三个孩子，才可以获得C，而不是访问第二个孩子
2. 写数据结构的时候留下了很多错误，导致很多segmentation fault，以后还是得先仔细检查数据结构写没写错……
3. 在更多的测试用例上发现，我的lab1存在问题，无法识别if ()else if () ... else的语句，由于测试这些更复杂的用例时已经是周五十点，因此没有修改，后续会对这些问题进行更正。
4. 我使用了全局的structTable来存储定义的结构体标识符，且没有考虑无名结构体的存储，未来有待完善。

总结

闭关四天勉勉强强能通过样例测试，最后两小时发现lab1有隐患的时候有点崩溃。代码量非常大，写完真的是收获满满（尽管仍然存在一些bug.....

后记

1. if else是可以识别的，之前在分析IF LP Exp RP Stmt ELSE Stmt时，分析第二个Stmt传参数传成了第一个Stmt导致这个问题……
2. 重名的变量与结构体仍然应该插入，例如测试用例A18中，如果不将main的Node加入符号表，会额外导致未定义Node的错误，因此我认为应该优先插入最近声明的变量。例如考虑

下面的代码

```
int main(){
    float a, f2;
    int a, b;
    b = 0;
    a = b;
    return a;
}
```

如果不将最新声明的a作为int类型插入符号表，那么会导致连锁的a = b赋值不匹配，return返回不匹配。
另外，我认为错误类型16是不合理的，变量名完全可以与结构名重合。
3. 在错误恢复方面还有很多值得斟酌的细节，例如在A13-A15的测试用例中，我还会识别出错误返回类型不是整型不能作为条件判断的错误。
访问未定义的域名之类的问题，因为无法确认这个不存在的域名的Type，因此我返回了NULL，但是会导致连锁的错误（尽管确实错了，但是根源只有Field Undefined），考虑类似error产生式的恢复方法，直接跳过这条语法单元，或许是一个更好的选择……
4. 考虑了表达式的运算错误，若两个操作数类型不同，返回前一个操作数类型，若是逻辑运算则错误时返回整型
5. 在DefList_in_struct函数中，每次插入新的域，直接采用了`field->nextFieldList = 解析返回的域`，否则对于域中声明了`int a, b; int c`，第二个Def会插入到第一个Def的第一个域中，即a的下一个域是c，而不是b，导致连续声明的后面的域名丢失。