编译原理实验2报告

组长：王齐剑 221900059

组员：晁宇豪 221240013 高歌 221240075

实验内容：

完成所有必做内容，实现了简单的语义分析。

完成选做3.1：函数的声明在相互一致的情况下可以重复出现。

完成选做3.2：完成可嵌套作用域的语义分析，设计了一个栈的结构体来存储作用域信息。

完成选做3.3：完成了允许结构体不同名但是内容相同的等价性

实验亮点：

1. 多层嵌套作用域的高效实现

设计了栈结构管理变量的作用域，完美支持C语言中的嵌套块作用域特性：通过`incStackDepth`和`decStackDepth`函数精确维护当前作用域深度，实现`delete_stack_curdepth`函数在离开作用域时自动清理变量，变量查找遵循"就近原则"，优先查找最内层作用域的变量，支持任意深度的嵌套（最大深度可配置为`max_st_depth`）。（代码略）

2. 函数声明与定义的一致性处理

实现了C语言中函数可多次声明但仅能单次定义的特性：使用`Funstate`枚举记录函数是声明状态还是定义状态，通过`equalType`函数递归比较函数签名的一致性，设计`DetectFunc_Undefined`函数在程序结束时检查未定义函数，当函数有多个一致的声明时，允许在程序的不同位置声明。（代码略）

3. 结构体的内容等价机制

支持结构体的结构等价而非仅名称等价：实现 `equalFieldList`函数递归比较结构体字段的类型和顺序，当两个结构体内部字段完全一致时，即使结构体名称不同，也视为等价类型，支持匿名结构体的定义和使用，自动生成唯一标识符。

```
// 结构体字段列表的等价性比较
int equalFieldList(FieldList field1, FieldList field2) {
    if (field1 == NULL && field2 == NULL)
        return 1;
    if (field1 == NULL || field2 == NULL)
        return 0;
    return equalType(field1->type, field2->type) &&
           equalFieldList(field1->tail, field2->tail);
}

// 类型等价性判断中处理结构体
case STRUCTURE:
    return equalFieldList(type1->u.structure, type2->u.structure);

// 无名结构体处理
if (tagnode->firstchild == NULL) {
    // 无名结构体
    table->unnamed_struct_num++;
    tagname = (char *)malloc(sizeof(char) * 20);
    snprintf(tagname, 20, "%d", table->unnamed_struct_num);
}
```

4. 高效的符号表设计

采用哈希表与作用域栈的复合结构：哈希表提供O(1)的符号查找效率（通过 `hashfn`函数实现高效哈希），栈结构实现作用域的精确管理（按深度组织符号），符号冲突检测(`objConflict`)支持不同作用域同名变量，针对函数符号的特殊查找优化(`searchtab_func`)。

```
// 高效的哈希函数实现
unsigned hashfn(char *name) {
    unsigned val = 0, i;
    for (; *name; ++name) {
        val = (val << 2) + *name;
        if (i = val & ~hashsz)
            val = (val ^ (i >> 12)) & hashsz;
    }
    return val;
}

// 符号冲突检测，支持不同作用域同名变量
int objConflict(ptab table, pobj obj) {
    pobj curobj = searchtab(table, obj->name);
    if (curobj == NULL)
        return 0;
```

```
    while (curobj != NULL) {
        if (!STRcmp(curobj->name, obj->name)) {
            if (curobj->type->kind == STRUCTURE || obj->type->kind == STRUCTURE) {
                return 1;
            }
            else if (curobj->stack_depth == obj->stack_depth) {
                if ((curobj->type->kind == FUNCTION && obj->type->kind != FUNCTION)
||
                    (curobj->type->kind != FUNCTION && obj->type->kind ==
FUNCTION)) {
                    continue;
                }
                else
                    return 1;
            }
        }
        curobj = curobj->hash_next;
    }
    return 0;
}
```