编译原理实验二实验报告

201840009田永上

1.任务阐述

必做内容：17个错误类型

选做内容：2.3修改前面的C--语言假设5，将结构体间的类型等价机制由名等价改为**结构等价（Structural Equivalence）**。例如，虽然名称不同，但两个结构体类型struct a { int x; float y; }和struct b { int y; float z; }仍然是等价的类型。注意，在结构等价时不要将数组展开来判断，例如struct A { int a; struct { float f; int i; } b[10]; }和struct B { struct { int i; float f; } b[10]; int b;}是不等价的。在新的假设5下，完成错误类型1至17的查。

2.功能实现

符号类型

符号类型定义如下：

```
struct Type_
{
    enum
    {
        BASIC,
        ARRAY,
        STRUCTURE,
        STRUCTDEF,
        FUNCTION
    } kind;
    union
    {
        int basic;
        struct
        {
            int size;//数组大小
            Type elem;//元素类型
        } array;
        struct
        {
            int param_num; //参数个数
            FieldList params;
            Type ret; // 返回值类型
        } function;
        FieldList struct_pnt; // 指向结构体定义
        FieldList struct_def;    // 结构体定义
    } u;
};
```

在实验手册的基础之上添加了一个STRUCTDEF类型，期中STRUCTDEF类型对应于结构体定义。

STRUCTDEF主要用于更便利地实现选做内容2.3，可以直接把STRUCTDEF类型地指针指向地u.struct_def内容进行逐项的匹配来实现结构等价的判定

作用域

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

hash节点

```
struct HashNode_
{
    FieldList data;//当前节点
    HashNode link;//下一个点
};
```

选做实现

在type_match函数中

```
bool type_match(Type tp1, Type tp2)
{
    if (tp1 == NULL || tp2 == NULL)
        return false;
    if (tp1 == tp2)
        return true;
    if (tp1->kind != tp2->kind)
        return false;
    FieldList tp1_def = tp1->u.struct_def;
    FieldList tp2_def = tp2->u.struct_def;
    if (tp1->kind == 0)
    {
        return tp1->u.basic == tp2->u.basic;
    }
    else if (tp1->kind == 1)
    {
        return type_match(tp1->u.array.elem, tp2->u.array.elem);
    }
    else if (tp1->kind == 2)
        return type_match(tp1->u.struct_pnt->type, tp2->u.struct_pnt->type);
    else if (tp1->kind == 3)//对应于STRUCTDEF
    {
        while (tp1_def != NULL || tp2_def != NULL)
        {
            if (tp1_def == NULL || tp2_def == NULL)
                return false;
            if (type_match(tp1_def->type, tp2_def->type) == 0)//逐项比较
                return false;
            tp1_def = tp1_def->tail;
            tp2_def = tp2_def->tail;
        }
    }
    return true;
}
```

可以看到是进行逐项比较类型是否等价，从而实现结构等价

3.编译和执行

```
进行编译
oslab@oslab-virtual-machine:~/Lab2/Code$ make
./parser [*.cmm]执行
oslab@oslab-virtual-machine:~/Lab2/Code$ ./parser Test/m9.cmm
```

4.实验感悟

在debug的过程中我遇到了无数次“段错误（核心已转储）”这个错误，主要原因是访问了非法地址，在debug的过程中我主要通过打印法确定出错的大概位置，然后再检查什么地方可能会访问非法地址。而后我总结发现，大部分错误是因为在if条件句中要比较结点名称未先保证结点非空。此外，一个比较合适的符号表设计会让实验事半功倍。在设计前我参考了github上多份学长学姐的设计方式，最终生成了我认为最为简单的一种设计。