实验二报告

一、 实现的功能

对以下错误的检查：

- 错误类型2：函数在调用时未经定义

  查询符号表中*u.func.defined*是否为*true*

```
struct symbol_
{
    enum {VARIABLE, FUNCTION, STRUCT_TAG} kind;
    union {
        Type variable;
        Type struct_tag;
        struct {
            int defined;
            Type ret;
            FieldList parameter;
        } func;
    }u;
    char name[NAME_SIZE];
    int first_lineno;
    void *belong;
    symbol hash_nxt;
    symbol list_nxt;
};
```

  函数调用时声明了但没定义的情况，归结到错误类型18

- 错误类型18：函数进行了声明，但没有被定义

  最后额外遍历一边符号表，检查所有函数是否声明并定义

```c
void CheckFun(){
    Assert(list_head != NULL);
    symbol cur = list_head->sym;
    while(cur){
        if(cur->kind == FUNCTION){
            if(!cur->u.func.defined) {
                semantic_error(18, cur->first_lineno,
"declared but not defined");
            }
        }cur = cur->list_nxt;
    }
}
```

要求实现：

- 要求2.2：变量的定义受可嵌套作用域的影响， 外层语句块中定义的变量可在内层语句块中重复定义，内层语句块中定义的变量到了外层语句块中就会消亡，不同函数体内定义的局部变量可以相互重名

  *struct symbol*_中引入变量*list_nxt*支持从另一维度引入链表将符号表中属于同一层作用域的所有变量都串起来

- 要求2.3：将结构体间的类型等价机制由名等价改为结构等价

  类型比较时展开比较

```c
int type_com(Type dst, Type src){
    Assert( dst != NULL && src != NULL);
    if(dst == Error_Type || src == Error_Type) return
true;
    if(dst->kind != src->kind) return false;
    switch (dst->kind) {
        case BASIC:
            return dst->u.basic == src->u.basic;
        case ARRAY:
            return array_com(dst, src);
        case STRUCTURE:
            return field_com(dst->u.structure, src-
>u.structure);
        default:
            Assert(0);
    }
}

int field_com(FieldList dst, FieldList src){
```

```
    if(dst == NULL || src == NULL) return dst == src;
    else{
        if(type_com(dst->type, src->type))
            return field_com(dst->tail, src->tail);
        return false;
    }
}
```

二、编译

```
Code文件夹下  make
```