# 实验二 语义分析

201220176 刘兴

## 一、实现功能

1．完成附录中**最基本的语法分析**。

2．完成要求 2.1—**函数除了在定义之外还可以进行声明**。

3．完成要求 2.2—**多层作用域**。

4．完成要求 2.3—**结构体间的类型等价机制由名等价改为结构等价**。

## 二、如何编译

1．通过 makefile 进行编译。

## 三、设计思路以及数据结构

### 1．数据结构

（1）修改了 Type 数据结构，增加了类型 FUNCTION,记录了函数的返回值以及参数。且增加 is_var 变量来判断是否是变量，还是说是结构体的名字。

```
typedef struct Type{
    enum { BASIC, ARRAY, STRUCTURE,FUNCTION} kind;
    union {
        //基本类型
        int  basic;
        //数组类型信息包括元素类型与数组大小构成
        struct {struct Type* elem; int size; } array;
        //结构体类型信息是一个链表
        struct FieldList* structure;
        struct { struct Type* ret; struct FieldList* para; } function;
    } u;
    //判断是否是变量，还是说是结构体的名字。
    int is_var;
}Type;
```

（2）hashNode 数据结构如下，主要是增加了为了实现多层作用域而使其成为了十字链表。

```
typedef struct hashNode{
    char* name; //节点名字
    Type* type;
    int line;   //节点的行数
    int defined;//是否为定义，或者是声明
    int size;   //符号的 size
    int depth;  //深度，用来实现作用域
    //opening hashing
    struct hashNode * next;
    //下一个同层的 Hashnode，用来实现作用域
    struct hashNode * under;
}hashNode;
```

2．设计思路：

（1）最基础的要求：只需要根据文法，为各个非终结符号构造对应函数，然后按照文法的结构进行相应的调用。在运行过程中，需要维护好一个符号表，并在适当时候检查是否出现相应的错误类型即可。①增加符号只会出现在 “OptTag -> ID(我的设计将结构体名字也记录在符号表中)”、“FunDec -> ID LP (VarList) RP”、“VarDec -> ID”，由于函数的参数名不会重复，所以我将函数的参数也会加入到符号表中。

（2）函数声明的实现：首先需要修改文法，增加相应的函数声明文法即“ExtDef->Specifier FunDec SEMI”；其次主要是增加了一个“defined”变量来告知是声明还是定义，过程中根据 defined 的值进行分别的处理，并且会在程序结束时检查函数表是否全都进行了定义来排查错误类型 18。

（3）多层定义域的实现：首先是修改散列表以及 hashNode 为十字交叉链表，横向为正常的 opening hashing，纵向为串联所有深度相同的节点。每进入一个语句块即 CompSt 便增加深度，为这一层语句变量创建链表；每离开一个语句块就要删除。

（4）结构等价的实现：相对较为简单。进行一个递归的判断即可。

```
int BeSameType(Type* Ltype,Type*Rtype){
    if(Ltype->kind==STRUCTURE)
    {
        FieldList *p1=Ltype->u.structure,*p2=Rtype->u.structure;
        while(p1&&p2)
        {
            if(!BeSameType(p1->type,p2->type))
                return 0;
            p1=p1->tail;
            p2=p2->tail;
        }
        if(p1||p2)
            return 0;
        return 1;}
```

## 四、遇到的问题

在这第二次编译原理的实验中，还是遇到了诸多的问题。

首先还是刚上手时的不知所措，在看了很长一段时间才逐渐知道该怎么做。

主要的问题就是空指针的引用问题，在设计时出现了太多的漏洞导致经常会出现空指针的引用而引起段错误，导致实验一直卡壳。

第二个问题就是本次实验错误类型太多，需要检查的东西太多，所以需要耐心的进行设计，这个过程确实让人头疼