实验二——C–语义分析器实现

实验二的任务是在词法分析和语法分析程序的基础上编写一个语法分析器，对C–源码进行语义分析和类型检查，并打印结果。与实验一不同，实验二的所有任务需手工完成 17 中错误类型检测及完成选做内容

实验特色：

我是选做内容 1
首先为了能够匹配上实验一所写的 C—文法，我写了一个外置函数来判断是否使用了该条规则

```
// 判断是否使用了指定的产生式
bool Use_This_Rule(struct Node* node, int index, ...){
        if(DEBUG_FLAG){
                printf("Go in Function:Use_This_Rule\n");
        }
        va_list stringlist;
        va_start(stringlist, index);
        bool if_this = true;
        struct Node* curNode = node->firstChild;
        for(int i = 0; i < index; i++, curNode = curNode->bro){
                if(curNode == NULL){
                        if_this = false;
                        break;
                }
                char* nowstr = va_arg(stringlist, char*);
                if(DEBUG_FLAG){
                        printf("nowNode is %s and curNode is %s\n",nowstr, curNode->nodeName);
                }
                if(!equal_string(nowstr, curNode->nodeName)){
                        if_this = false;
                        break;
                }
        }
        va_end(stringlist);
        if(curNode != NULL)
                if_this = false;
        return if_this;

}
```

然后对于函数入口：

```
}
// 语义分析入口函数
void semantic_check_start(struct Node* Root){
        initHashTable();
        if(DEBUG_FLAG){
                printf("Analyse start!\n");
                Print_Tree(Root, 0);
        }
        // 遍历语法树中的ExtDefList节点
        ExtDefList(Root->firstChild);
        checkFunc();
}
```

最后的 checkFunc 是用于解决选做内容中对于声明但是未定义的函数问题。

```
struct Function_{
        char*name;      //名称
        int linenum;    //行号
        Type type;      //返回值类型
        int state;      //是否定义，0为声明，1为定义
        FieldList next; // 下一个域
};
```

为了解决这个问题我在函数的 struct 内新增了一个 int 类型，用于说明该函数是否处于未定义的情况。

```
void checkFunc(){
        if(DEBUG_FLAG)  printf("Go in checkFunc!\n");
        for(int i = 0; i < 1024; i++){
                if(gTable[i] != NULL && gTable[i]->type->kind == FUNCTION){
                        if(gTable[i]->type->inform.function->state == 0){
                                printError("18", gTable[i]->type->inform.function->linenum, "函数进行了声明，但没有被定义");
                        }
                }
        }
}
```

如图所示。
最后为了实现声明可以重复多次的目标，我在 Function 的函数多增加了一个参数 state，用于辨别当前为声明还是定义，对于声明的函数我们不对其的 VarList 进行检查即可。

```
Function FunDec(struct Node*node, Type ntype, int state){
```

实验感想：

在实验一的基础上开发实验二比单独做实验一要容易少许，根据 C—文法逐步向下分析语法树即可完成实验，再基于选做内容增添部分文法进行修改就好了。