编译原理实验二　语义分析

实验报告

组号：2　　组长：郑来栋

一．　实验进度

目前已完成实验二全部要求

二．　一些关键的点

1. 符号表

符号表和类型系统都采用实验指导书的方案，使用哈希表存储，其中做了一些小的改动。针对选做要求，增加函数类型

```
struct {
        enum {DECLARED, DEFINED} state;
        struct Type *retValueType;
        struct FieldList *params;
}function;
```

state 表示分析到目前为止该函数是被声明了还是被定义了，整个程序分析结束后，需要检查看是否有函数处于 state=DECLARED 的状态（即声明而未定义）

retValueType 和 params 分别是函数返回值的类型和形参列表。

2. 语义分析思想

语义分析阶段要做的事情主要有以下几点

(1) 收集类型信息，构建符号表

(2) 进行类型检查，报告类型错误

方法就是遍历语法树，针对不同的语法节点，对该子树进行特定的语义分析动作，一些例子：

(1) 遇到 VarDec，说明有变量定义

(2) 遇到 Specifier，则以该节点为根的子节点表示一个 Type

(3) 遇到 ExtDef->Specifier FunDec CompSt

| Specifier FunDec SEMI

根据第三个 child 是 CompSt 还是 SEMI 判断是函数声明还是函数定义

总之，根据当前节点表示什么，相对应的进行相对的语义分析动作。

3. DecList -> Dec | Dec COMMA DecList 这样的产生式如何处理？

这可以借鉴链表的思想，展开之后像这样 DecList -> Dec COMMA Dec COMMA Dec，解决它的伪码如下

```
while(1){
    Node Dec = DecList->children[0]
    analyse(Dec)
    if(DecList -> children number == 1)
        break
    DecList = DecList->children[2]
}
```

还有很多这样的结构，可以用相似的方式处理

4．类型检查

类型检查是判断类型和期望的类型是否匹配的重要内容（如函数返回值类型是否正确， 函数参数类型是否一致， 操作符的 operands 是否符合规则等），我们可以用递归的方式，伪码如下

```
typeCheck(Type t1, Type t2)
{
    if t1.kind ≠ t2.kind
        return false
    if t1.kind == BASIC
        return t1.basic == t2.basic  //如果是基本类型，直接判断 basic 是否相等
    if t1.kind == ARRAY
        return typeCheck(t1.array.elem, t2.array.elem)
    if t1.kind == STRUCTURE
        return t1.structure.name == t2.structure.name
    //FUNCTION
    if(typeCheck(t1.function.retValueType,t2.function.retValueType)==false|
        |t1.funciton.paramNumber ≠ t2.function.paramNumber)
        return false
    for(i = 1; i < t1.function.paramNumber; i++)
        if typeCheck(t1.function.params[i], t2.function.params[i]) == false
            return false
    return true
}
```

5.比较复杂的就是 Exp 的分析

对于一个 Exp，上层可能需要用到这个 Exp 的类型（实际大多数情况都需要），因此对 Exp 没有出现语义错误，则返回该 Exp 的 Type，否则返回 NULL。另外，我们需要根据不同的组合进行不同的分析，其中比较复杂的就是 Function，和 Assignop 以及数组引用。代码中专门针对这些组合设计了专门的函数来进行 analyse

6. 如何区分变量定义是否在结构体内？

初始化结构体中的变量也是一种语义错误，因此我们要能够分析出当前的 VarDec 是否在结

构体内，如果是 Dec 只能有 Dec->VarDec 这条产生式，而 Dec->VarDec ASSIGNOP Exp 是不允许的。
那么如何区分呢？代码中设计了一个变量 inStruct，每当遇到 StructSpecifier，inStruct 加 1(结构体可以嵌套定义)，分析完一个之后 inStruct 减 1，因此，如果 inStruct 不等于 0，就表明当前是处于结构体中。

三． 编译运行方式

程序没有对 Makefile 进行修改