实验二 C--语义分析

姓名：肖丹妮　　学号：201220199　　专业：计算机科学与技术系

一、实验环境与编译方法

实验环境与手册一致；利用提供的Makefile文件进行编译。

二、实验实现的功能

- 完成了必做内容，能够对输入文件进行语义分析并能够检查包括1至17所有类型的错误。
- 同时完成了所有选做要求，即要求2.1、2.2、2.3，能在新的要求下检查包括1至19所有类型的错误。

三、实验内容与核心代码

3.1 数据结构

3.1.1 类型表示

Type结构与手册给出的基本一致，在此基础上增加了类型NOTHING，用于初始化；FUNCTIONDEF-函数定义；FUNCTIONDEC-函数声明，还有函数的类型信息如下：

```
struct
{
    Typep return_type;//函数的返回类型
    int argn;//函数的形参个数
    FieldListp args;//函数的形参用链表存储
} function;
```

3.1.2 符号表

采用的是指导手册中基于十字链表 和open hashing散列表的Imperative Style的符号表设计：

a. 符号表元素SymbolNode

```
typedef struct SymbolNode
{
    char name[32];
    Kind_ kind_;//类型
    Typep type;//类型信息
    int deepth;//深度信息，便于查找
    SymbolNodep next;//十字链表中的横向下一个
    SymbolNodep prev;//十字链表中的横向上一个
    SymbolNodep snext;//十字链表中的竖向下一个，是位于同一作用域中的节点
} SymbolNode;
```

b. Hash Table

```
typedef struct SymbolNode **SymbolTable;
```

c. stack

```
typedef struct Stack *Stackp;
typedef struct Stack Stack;
struct Stack
{
    SymbolNodep* snps;
    int top;//栈顶
    int size;
};
```

3.1.3 语法单元的属性

在实验一的基础上进一步对得到的语法树进行语义分析，因此在结构体TreeNode中添加语法单元的继承属性与综合属性以及其他标志信息，具体如下：

```
typedef struct TreeNode
{
   ……
    int childtype;//在构建语法分析树时完成，记录子结点的类型，也就是产生式的类型，例如
    bool isstrdef;//用于标记变量是否是结构体的域，在发现一个变量定义时，先判断结点的
isstrdef是否为0：若为0，则表示不是结构体的域，将变量插入符号表；否则只需设置该结点的属性，将该
变量连接到结构体域链表
    bool isfundec;//用于标记结点属于函数声明还是函数定义，
    FieldListp syn;//结点类型信息的综合属性
    FieldListp inh;//结点类型信息的继承属性
    ……
} TreeNode;
```

3.2 语义分析的整体实现

3.2.1 语法树的语义分析

根据3.1的数据定义需要的全局变量：符号表、结构体定义栈、函数声明记录数组。

```
//符号表
SymbolNodep st[SYMBOL_TABLE_SIZE];//大小为SYMBOL_TABLE_SIZE=0x3ff
Stackp stack_sn;//stack_sn->top标志当前作用域的深度，stack_sn->snps[stack_sn->top]指
向记录在同一层作用域的所有链表
//结构体定义栈
Stackp strdef;
//strdef->snps[strdef->top]指向存储这一层结构体域的链表，结构体定义可能会嵌套，当出现另一层
嵌套时，将strdef->top加一，开始记录内层结构体的定义信息，记录完成，strdef->top再减一，退出内
层，继续记录外层下一个域定义
//函数声明记录数组
FieldListp fundec_st[100];
int fundecn = -1;
int fundec_co[100]; // 函数声明所在位置，便于打印错误信息
int fundec_con = -1;
//记录每一个函数声明（只记录重复声明中的第一个）的信息，在对整个程序分析完后判断是否每个函数声明
都有定义，如果在符号表中没有找到相关的函数定义，就输出错误类型18的信息
```

在main.c中调用begin_semanalysis(root)开始对语法树进行语义分析。

begin_semanalysis()中包含三部分：
对3.2.1全局变量的初始化、调用Analyse_Program(root);开始对语法树的Program进行语义分析、最后逐个检查函数声明记录数组中的函数声明是否都有定义。

3.2.2 语法单元的语义分析

- 各个语法单元的分析函数基本都是由以下几部分构成：
  设置一个错误标志isright。若调用子结点的分析函数，可以由返回值得到语法树子结点是否正确的信息；
- 利用Childi()得到父节点的各子结点；
- 根据结点的childtype分情况讨论对于子结点的分析；
- 若子结点是词法单元，直接设置子结点的综合属性；若子结点是语法单元，通常需要先传递父节点的继承属性，调用子结点的分析函数后得到子结点的综合属性，最后根据子结点的综合属性设置父节点的综合属性；
- 若中间发生错误或者子结点本身含有错误，依据具体情况设置isright，最后函数返回此标志。

3.2.3 选做要求的实现

a. 要求2.1

- 在syntax.y中增加产生式ExtDef→Specifier FunDec SEMI
- 函数声明与函数定义区分开，作为不同类型的符号表结点。
  函数声明出现时，先查表中是否有同函数的函数声明，若符号表中已有函数声明，判断两者返回类型、形参个数、形参类型是否相同，不相同则报错；若表中没有同函数的函数声明，则将函数声明插入符号表。在符号表中只记录某函数第一个出现的函数声明即可。

  函数定义出现时，先查表中是否有同函数的函数声明，若符号表中已有函数声明，判断两者返回类型、形参个数、形参类型是否相同，不相同则报错；再查表中是否有同函数的函数定义，若符号表中已有函数定义，则报错。
- 分析完语法树后，检查每个函数声明是否都能在符号表中找到定义，若没有定义，输出错误类型18。

b. 要求2.2

- 在Analyse_CompSt()函数中，
  在分析子结点前，要进入语句块，故使作用域栈深度stack_sn->top加一，并且若此语句块是函数的定义，就将函数的形参依次加入符号表，并且连接到stack_sn->snps[stack_sn->top] ，建立起符号表元素之间的十字关系。
  在分析完子结点后，要退出语句块，故将stack_sn->snps[stack_sn->top]下挂的链表的每一个元素都从符号表中删去，最后使作用域栈深度stack_sn->top减一。
- 出现变量定义时，先在stack_sn->snps[stack_sn->top]下挂的链表中查找符合条件的变量，若找到了，则变量定义出现了冲突，报错；否则将该变量定义插入符号表。
- 使用变量时，在散列表中查找第一个遇到的符合条件的变量定义，此定义就是同名变量中最靠近当前作用域的变量定义。

c. 要求2.3

- 实现结构体的结构等价只需在遇到两个结构体时，依次递归比较它们类型信息链表每个结点的类型。