实验二：语义分析

成员	学号	院系	邮箱
张路远	221220144	计算机学院	craly199@foxmail.com
姜帅	221220115	计算机学院	jsissosix1221@gmail.com

在实验二中，我们需要在实验一词法分析和语法分析的基础上编写一个程序，以实验一输出的语法树作为输入，对C--源代码进行语义分析和类型检查，并打印分析结果.

实现方法

概览

从语法树的根节点开始遍历。访问到某个非终结符节点时，由于语法树已经完整构建，我们可以提前访问子节点，结合子节点数目和个别关键词就能决定按照哪一条产生式展开，以及实行对应动作

以`start_semantic_analysis`为语义分析的入口：初始化符号表后调用`SemanticAnalysis`开始语义分析

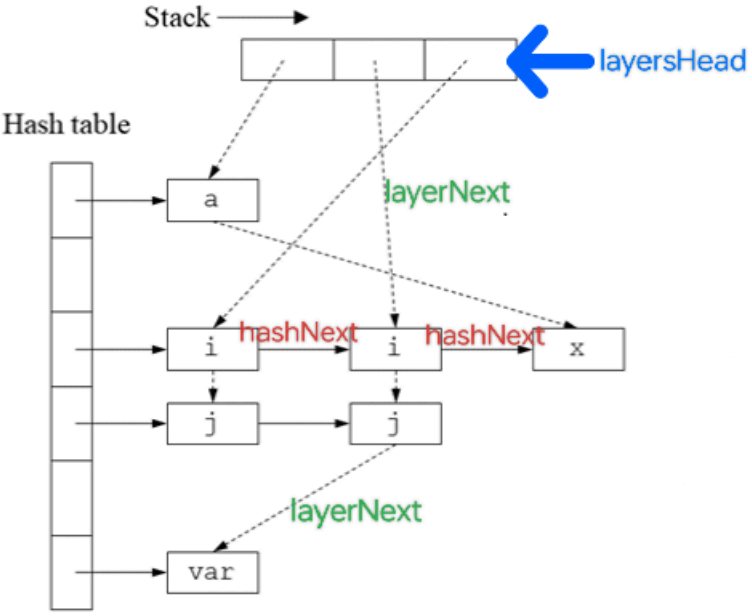
`SemanticAnalysis`验证输入的根节点`node`有效后，调用`ExtDefList`；

`ExtDefList`验证子节点非空后，调用`ExtDef`和`ExtDefList`...

如此递归下去，完成整个语法树的遍历和语义分析

符号表

为了实现语义分析诸如检查变量类型以及重复定义等功能，我们需要定义一个符号表进行填表和查表工作，我们的选做为3.2，需要处理变量的定义域，我们采用**open hashing 散列表**和**Imperative Style**的支持多层作用域的符号表设计，符号表设计图，散列表的节点以及符号表相关定义如下, type参考了手册示例的实现。



```c
struct Symbol_
{
    char name[32];       ///< Symbol name (variable, function, etc.)
    Type_t *type;        ///< Associated type
    int lineno;          ///< Line number of declaration
    Symbol_t *hashNext;  ///< Next symbol in hash table bucket
    Symbol_t *layerNext; ///< Next symbol in scope layer
};

// Symbol table and layer stack definitions
Symbol symbol_table[TABLE_SIZE];
Symbol layersHead;
```

符号插入、查找、删除： 插入符号时，需要将符号插入到对用槽位的链表头部, 以及当前作用域（`layersHead->hashNext`）的链表头部。符号的查找分为在当前作用域查找（`search_symbol_layer`）和全局查找(`search_symbol`）。

作用域管理：包括插入新作用域（`enter_layer()`）和离开作用域（`exit_layer()`）

当进入函数或块时就创建一个新作用域节点，插入到 `layersHead->hashNext`,使得新创建的作用域位于栈顶，此后在此作用域下插入的符号会通过`layerNext`属性连接

要离开作用域时从 `layersHead->hashNext` 取当前作用域。遍历 `layerNext` 链表，删除当前作用域的每个符号。

填表和查表部分

对于变量：插入变量到符号表之前首先检查冲突，首先在当前作用域下检查冲突，然后再全局检查，仅当当前作用域下存在冲突或全局存在冲突且符号类型为结构体定义时报`name confilct`错误。

对于结构体：遇到结构体的定义时，尝试插入前首先使用 `search_symbol_all` 检查全局符号表中是否已有同名符号，若有则报错。如果没有结构体命名冲突，继续检查这个结构体内部是否有命名冲突，该操作通过函数`check_dup_in_struct`完成

对于函数：遇到函数定义时，首先试用 `search_symbol_all` 检查全局符号表中是否已有同名，若有则报错，否则正常插入函数名，接着处理函数体`CompSt`，在进入`CompSt`前需要创建新的作用域，以确保函数体内单独进行处理。对于函数的参数，我们会传递给`CompSt`一个`char*`类型的名为`funcName`参数，该参数用于通过函数名确定函数本身的`Symbol`，确定函数后，将函数的参数依次插入到当前作用域（遇到函数的“{}”后新创建的作用域）中，确保不同函数参数之间不会发生命名冲突。

前序改动

假设3：不允许函数声明：需要修改实验一中的`syntax.y`，在`ExtDef`块中额外添加：

```
ExtDef  : ...
        | Specifier FunDec SEMI         {
                yyerror("function declaration not allowed");
                Berrors++;
                $$=createNode("ExtDef", NODE_TYPE_NORMAL, @$.first_line,
3, $1, $2, $3);
             }
        ...
        ;
```

在语法分析时直接拦截函数声明，并报告为语法错误(错误类型B)