南京大学本科生实验报告

课程名称： **编译原理**

学号 191220029 **姓名** 傅小龙

1.实验名称

C--语义分析

2.实验内容

本次实验中我实现了所有基础功能（检查错误类型1-17）和所有的拓展功能（要求2.1，要求2.2，要求2.3）

2.1基础功能

本次实验中对C--语言规定的类型采用和实验手册（P66）中相似的数据结构，这里不再赘述。

符号表采用哈希表的方式实现。其地址映射方式采用实验手册（P61）中P.J. Weinberger提出的方法。符号表结点的数据结构及其解释如下：

```
#define HASHSIZE 0x3FFF
struct _HashTable
{
    char* name;             //结点对应变量/函数的名称
    sType* type;            //C--类型
    HashTable* next;        //open hashing方式解决冲突
    unsigned size;          //结点是基础类型/数组时的大小
} *hashTable[HASHSIZE+1];
```

本实验的语义分析采用自顶向下的L属性文法SDT方法实现。利用上一次实验中词法/语法分析生成的语法树，对每一种类别的语法结点及其产生式，根据自顶向下SDT思想设计对应的函数实现语义分析的功能。具体实现见`semanteme.h`,`semanteme.c`文件。

错误类型1-17的检测思路如下：

错误类型1:变量在使用时未定义。和变量相关的产生式只有：`Exp -> ID`,故在对该产生式进行语义分析时，检查`ID`结点对应的变量是否符号表中有对应项。若无，则报错。

错误类型2:函数在使用时未定义。调用函数时相关的产生式有：`Exp -> ID LP Args RP|ID LP RP`,检查`ID`结点对应的变量是否符号表中有对应项。若无，则报错。

错误类型3:变量出现重复定义，或变量与前面定义过的结构体名字重复。定义变量相关的产生式只有：`VarDec -> ID`.检查`ID`结点对应的变量是否符号表中有对应项，若有，且该变量不是一个结构体的域，那么报该类型错误。

错误类型4:函数出现重复定义。函数定义相关的产生式有：`FunDec -> ID LP VarList RPFunDec|ID LP RP`.检查`ID`结点对应的变量是否符号表中有对应项，若有，则报错。

错误类型5：赋值号两边的表达式类型不匹配。检查产生式中`ASSIGNOP`两侧语法单元在完成语义分析后得到的类型是否一致。若不一致，则报错。

错误类型6:赋值号左边出现一个只有右值的表达式。赋值号左边可能出现只有右值的产生式只有：`Exp -> Exp ASSIGNOP Exp`.具有左值的`Exp`可导出的产生式为：`Exp -> Exp LB Exp RB|Exp DOT ID|ID`.故若赋值号左侧的`Exp`不是又前面的产生式导出，那么报该类型错误。

错误类型7：操作数类型不匹配或操作数类型与操作符不匹配。根据对C--语言作出的假设2，产生式`Stmt -> IF LP Exp RP Stmt|IF LP Exp RP Stmt ELSE Stmt| WHILE LP Exp RP Stmt`和`Exp -> Exp AND Exp|Exp OR Exp|Exp RELOP Exp|NOT Exp`中的`Exp`应该是`int`类型的。若不是，则报错。

根据对C--语言作出的假设1和假设2，对于产生式`Exp -> Exp PLUS Exp|Exp MINUS Exp|Exp STAR Exp|Exp DIV Exp`,运算符两侧的`Exp`必须具有相同的类型且只能是`int`或`float`类型；产生式`Exp -> Minus Exp`中运算符右侧的`Exp`只能是`int`或`float`类型。

错误类型8：return语句的返回类型与函数定义的返回类型不匹配。相关的产生式只有`Stmt -> RETURN Exp SEMI`.根据属性L文法，函数定义时的返回值类型将由`Stmt`结点的继承属性给出，故将`Exp`的类型和`Stmt`结点的继承属性给出的类型比较，若不相同，那么报错。

错误类型9：函数调用时实参与形参的数目或类型不匹配。相关的产生式为`Exp -> ID LP Args RP|ID LP RP`.对于有参数列表的函数调用，根据属性L文法，的分析参数列表的分析结果将由`Args`的综合属性给出，函数的参数列表通过`ID`结点给出的函数名查阅散列表获取逐个，比较参数列表的中各参数和函数定义时给出的参数列表中的各参数，若有不同，则报错；对于没有参数的函数调用，判断被调用函数的参数数量是否为0，若不是，则报错。

错误类型10：对非数组型变量使用“[...]”（数组访问）操作符。相关的产生式只有`Exp -> Exp LB Exp RB`，判断操作符`[]`左侧的`Exp`在完成语义分析后其综合属性的类型是否是数组类型，若不是，则报错。

错误类型11：对普通变量使用“(...)”或“()”（函数调用）操作符。相关的产生式为`Exp -> ID LP Args RP|ID LP RP`.根据`ID`给出的函数名查阅散列表，若得到的符号其类型不是函数，则报错。

错误类型12：数组访问操作符“[...]”中出现非整数。相关的产生式只有`Exp -> Exp LB Exp RB`.操作符`[]`内的`Exp`在完成语义分析后，若其综合属性的类型不是`int`类型，则报错。

错误类型13：对非结构体型变量使用“.”操作符。相关的产生式只有`Exp -> Exp DOT ID`.若`.`操作符前的`Exp`在完成语义分析后其综合属性的类型不是结构体变量，那么报错。

错误类型14：访问结构体中未定义过的域。相关的产生式只有`Exp -> Exp DOT ID`.遍历`Exp`对应的结构体变量所有的域，若没有找到和`ID`匹配的域，那么报错。

错误类型15：结构体中域名重复定义（指同一结构体中），或在定义时对域进行初始化。结构体中域名重复定义相关的产生式只有`VarDec -> ID`.在向符号表插入`ID`对应的符号前，检查符号表中是否有和`ID`同名的符号，若有，那么报错。

在定义时对域进行初始化相关的产生式只有`Dec -> VarDec ASSIGNOP Exp`.若`VarDec`结点在完成语义分析后发现该节点对应一个结构体中的一个域，那么报错。

错误类型16：结构体的名字与前面定义过的结构体或变量的名字重复。相关的产生式只有`OptTag -> ID`.在向符号表插入`ID`对应的符号前，检查符号表中是否有和`ID`同名的符号，若有，那么报错。

错误类型17：直接使用未定义过的结构体来定义变量。相关的产生式有`ExtDecList -> VarDec|VarDec COMMA ExtDecList`,`DecList -> Dec|Dec COMMA DecList`.对于`ExtDecList`和`DecList`，其定义的变量由其父节点`ExtDef`和`Def`对应产生式中的`Specifier`结点给出，在完成`Specifier`结点的语义分析后，其给出的类型由继承属性传递给`ExtDecList`或`DecList`。若其属性类型为结构体变量并且其域链表为空（分析结构体变量的`Opt`结点时，若符号表中无对应项，将其域链表留空），那么报错。

2.2 拓展功能

2.2.1 要求2.1：函数声明

添加文法：`ExtDef -> Specifier FunDec SEMI`以识别函数声明。

向`FieldList`结构体定义中添加域`(bool) funDecFlag`以区分函数定义和声明。

添加待定义函数链表`(HashTable) *funDecList`. 语义分析时，若遇到一个函数声明且尚未有该函数的定义或声明，则将其加入到该表中。若遇到函数定义，那么遍历该链表，若发现有相同函数且不冲突的声明，那么从链表中删去之。

错误类型18：函数进行了声明，但没有被定义。在完成语义分析后，若待定义函数链表中仍有结点，那么这些结点对应的函数都没有定义，遍历该链表并报错。

错误类型19：函数的多次声明互相冲突或者声明与定义之间互相冲突。函数多次声明冲突：在遇到函数声明时，若待定义函数链表中有同名的函数声明，且两个声明的参数列表并不相同，那么报错。

声明与定义之间冲突：在遇到函数定义时，若待定义函数链表中有同名的函数声明，且两个声明的参数列表并不相同，那么报错。

2.2.2 要求2.2 嵌套作用域

根据实验手册（63页），将符号表改为open hashing散列表的Imperative Style的符号表设计。

添加活动作用域链表`(HashTable)* hashStack`，修改`2.1 基础功能`中提出的`_HashTable`数据结构，添加域：`(HashTable*) down`作位十字链表中的垂直链表指针；`(HashTable*) stackRoot`指向十字链表中的作用域结点。

在遇到`LC`({) 结点时，向作用域链表中添加新的结点，在该作用域内生成的符号全部挂靠在该结点下。

在遇到`RC`(}) 结点时，删除作用域链表表尾的结点及其挂靠的所有符号结点。

2.2.3 要求2.3 结构等价

根据实验手册对要求2.3的描述，在判断两个结构体是否相同时，只是依次比较其中的域是否具有相同的类型，并且不展开对数组的判断。具体实现详见`semanteme.c`文件中的`bool structFieldCmp(FieldList* a, FieldList* b);`.

3. 编译本程序

在提交文件的根目录下，`/extensive_file`文件夹内为实现实验手册中所有基础和选做要求的词法语法分析器。请使用这个文件夹下的`Makefile`编译生成可执行对象。

在`/extensive_file`目录下的终端输入`make`指令即可生成可执行目标`parser`，使用形如`./parser <test file>`的指令即可对参数`<test file>`进行语义分析.