编译系统实验二

姓名：姚舜宇　学号：1190202107　班级：1903602

实验目的

在词法分析和语法分析程序的基础上对 C—源代码进行语义分析和类型检查。

实验环境

GNU Linux Release: Ubuntu
GCC
GNU Flex
GNU Bison

实验内容

加入函数声明的文法：ExtDef->Specifier FunDec SEMI
符号表与函数表：

```
struct TABLE* table[16384];
struct FUNCTION* function[16384];

struct TABLE{
    int is_def_struct;
    FieldList field;
    struct TABLE* next_for_openhash;
    int linenumber;
};
struct FUNCTION{
    char* name;
    FieldList field;
    Type return_type;
    int declaration;
    int definition;
    int linenumber;
    struct FUNCTION* next_for_openhash;
};
```

对每个语法单元即产生式中的非终结符设置相应的函数，根据产生式进行调用。

Program：
根据子节点是否为空，递归调用函数 ExtDef。

```
struct Node* extdeflist=now->child;
while(extdeflist->child!=NULL){
   struct Node* extdef=extdeflist->child;
   ExtDef(extdef);
   extdeflist=extdef->brother;
}
```

Specifier：
若类型为 TYPE，则根据情况设置为 int 或 float，若类型为 StructSpecifier，则调用函数 StructSpecifier()。

```
if(strcmp(child->name,"TYPE\0")==0){
    Type return_type=(Type)malloc(sizeof(struct Type_));
    return_type->kind=BASIC;
    if(strcmp(child->char_name,"int")==0) return_type->u.basic=0;
    else return_type->u.basic=1;
    return return_type;
}
else{
    Type return_type=StructSpecifier(child);
    return return_type;
}
```

StructSpecifier：

1. STRUCT OptTag LC DefList RC
   读取OptTag中的ID信息,生成结构体名称,查找该命名是否已经被定义过。调用DefList生成域，若无错误，则将该结构体加入符号表。
2. STRUCT Tag
   读取Tag中的ID信息，在符号表中查找，找到则返回对应类型，否则报错。

DefList, DecList, VarList, ExtDecList, StmtList
根据产生式进行递归调用。根据情况把内容保存到FieldList中。

Dec：

1. VarDec
   调用函数VarDec()。
2. VarDec ASSIGNOP Exp
   根据传递的judge值，若为0，则结构体定义，对应错误类型15。否则调用Exp()，若返回类型与type不同，则对应错误类型5。

VarDec：

1. VarDec LB INT RB
   说明类型为数组ARRAY，向上传递调用VarDec()。

```
struct Node* int_num=fir_bro->brother;
Type vardec_type=(Type)malloc(sizeof(struct Type_));
vardec_type->kind=ARRAY;
vardec_type->u.array.size=int_num->int_number;
vardec_type->u.array.elem=type;
FieldList find_upper=VarDec(child,vardec_type,judge);
return find_upper;
```

2. ID
   创建一个FieldList，读取ID值。函数声明的参数：返回type，不加入符号表；函数定义的参数或变量定义：符号表中查询，若重复则报出错误类型3，否则加入符号表；结构体的定义：符号表中查询，若重复则报出错误类型15，否则加入符号表。

FunDec：
创建一个函数FUNCTION，设置对应的类型等值。
声明：判断是否已经加入FUNCTION表。
定义：判读是否已经定义过，若已定义则报出错误类型4.

Stmt：

1. Exp SEMI
   调用Exp()。
2. CompSt
   调用CompSt()。
3. RETURN Exp SEMI
   调用Exp()，比较返回值和type的值，若不同则报出错误类型8，return的返回类型和函数定义类型不匹配。
4. IF或WHILE
   调用Exp(),若不是int类型,则报出错误类型7,即操作数类型不匹配。否则调用Stmt()。

Exp：

1. INT或FLOAT
   直接生成对应type。
2. ID

在符号表中查找是否存在，不存在则错误类型 1，变量未经定义就使用。存在则返回 FieldList 中的 type。

3. MINUS Exp 或 LP Exp RP
   调用 Exp()。
4. NOT Exp 或 AND 或 OR
   调用 Exp()，若不为 int 类型，则对应错误类型 7，操作数类型不匹配。
5. ASSIGNOP
   判断左部是否满足左值的产生式，不满足则对应错误类型 6，赋值号左边出现一个只有右值的表达式。若左右 Exp 类型不同，则对应错误类型 5，赋值号两边表达式类型不匹配。
6. RELOP 或 PLUS 或 MINUS 或 STAR 或 DIV
   若左右表达式类型不同，则对应错误类型 7。
7. Exp DOT ID
   判断左边表达式是否为 STURCTURE 类型，不是则对应错误类型 13，即对非结构体类型变量使用“.”操作符。判断右边 ID 是否定义过，未经定义则对应错误类型 14，访问结构体中未定义的域。
8. Exp LB Exp RB
   判断左边表达式是否为数组 ARRAY 类型，不是则对应错误类型 10，对非数组型变量使用“[…]”。判断右边表达式是否为 int 类型，不满足则对应错误类型 12，即数组访问操作符“[…]”中出现非整数。
9. ID LP Args RP 或 ID LP RP
   判断左边 ID 是否在符号表，若在，则对应错误类型 11，对普通变量使用“(…)”或“()”操作符。判断左边 ID 是否在 FUNCTION 表，不在则对应错误类型 2，函数在调用时未经定义。调用 Args()判断函数参数是否符合规范，不满足则对应错误类型 9，函数调用时实参与形参的数目或类型不匹配。

Args：
调用 FieldList->type=Exp()，并将相连的 Args 产生的 FieldList 通过链表进行连接。

编译过程：

```
bison -d syntax.y
flex lexical.l
gcc syntax.tab.c semantic.c main.c -lfl -o parser
./parser test.cmm
```