实验三 中间代码生成

姓名： 张子谦
学号： 151220166
邮箱： zhangziqian1011@126.com

一、 完成功能

1. 完成所有必做内容，即满足 7 条假设的情况下，将 c—代码翻译成中间代码；
2. 实现要求 3.1，使结构体变量可以出现，且可以作为函数参数；
3. 实现要求 3.2，使多维数组变量可以出现，且可以作为函数参数；
4. 总体来说完成了所有必做部分和额外要求，但对于中间代码几乎没有优化；

二、实现方法

1. 在每一个函数体语义分析完毕之后，按照讲义提供以及自行完成的翻译规则，对于每个 CompSt 进行翻译生成中间代码后打印这些中间代码。
2. 维护的中间代码用的是便捷高效的线性中间代码，结构单元如下:

```
typedef struct InterCodes {
    InterCode *code;
    struct InterCodes *prev, *next;
} InterCodes;
```

以上是链表的单元，下面的是中间代码的内容，其中 num 表示操作数的数量。

```
typedef struct InterCode {
    interCodeKind kind;
    union {
        struct { Operand *single; } value;
        struct { Operand *right, *left; } assign;
        struct { Operand *result, *op1, *op2; } binop;
    };
    union {
        char *relop;
        int size;
    };
    int num;
} InterCode;
```

而具体的操作数则用下面这个结构来存，可以存各种形式的操作数：

```
typedef struct Operand {
    operandKind kind;
    union {
        char *name;
        int id;
        int num;
    };
} Operand;
```

操作数和中间代码的类型都存在 kind 域里面，这些取值定义在下面这些 enum 里面:

```
typedef enum { VARIABLE, CONSTANT, LABEL, FUNCTION, TEMP, } operandKind;
typedef enum { DEFINE_LABEL, DEFINE_FUNCTION,
          ASSIGN, ADD, SUB, MUL, DIV_,
          ASSIGN_ADDR, ASSIGN_VAL, VAL_ASSIGN,
          GOTO, GOTO_COND,
          RETURN_, DEC, ARG, CALL, PARAM,
          READ, WRITE, } interCodeKind;
```

3. 为了做数组和结构体传参，以及对于各种变量的计数，在符号表的结构里增加了 isRef 和 id 域来记录是否是引用型参数和变量编号。

三、运行方式

1. 在 Code 文件夹运行 make 可以在项目目录中产生可执行文件 compiler，./complier testfile 1>outfile 可以对 testfile 进行词法语法语义分析并将中间代码输出到 outfile 里面(注意这个为了方便和

实验手册上描述的不同)。
2. make test 可以测试 testcase/文件夹下面的所有.cmm 文件并输出到对应的.ir 文件中。
3. make clean 清除所有 make 生成的文件。