软件工程理念

高内聚，低耦合

- 内聚：每个模块尽可能独立完成自己的功能，不依赖于模块外部的代码
- 耦合：模块与模块之间接口的复杂程度，模块之间联系越复杂耦合度越高，牵一发而动全身

变量类别和编译预处理

作用域与生命期

	作用域	生命期	存放区域
局部	从定义处开始到函数结束	函数开始到函数结束	栈空间
块	从定义处开始到块结束	函数开始到函数结束	栈空间
全局	工程的任何地方	程序模块载入到卸载	数据区
静态全局	从定义处开始到文件结束	同全局	同全局

注

- 局部优先于全局
- 内层可用访问外层，外层无法访问内层（内层属于受编译器约束的局部变量，在编译期间不允许越界访问）

```
void foo()
{
    int m = 7;

    // 块开始
    {
        int n = 8;      // 受编译器约束
    }
    // 块结束

    printf("%d\n", n);      // 编译失败
    printf("%d\n", (&m)[-1]);   // 编译通过
}
```

- 在多文件访问全局变量时需要加`extern`关键字进行声明

```
/****** file: main.c ******/
int global_var = 8;
...

/****** file: foo.c ******/
extern int global_var;      // 不能加动作：比如赋值、运算等
...
```

- 编译器会对变量、函数等进行名称粉碎（对其改名字）
  `extern "C" int global_var`，可以在`cpp`与`c`联合编译时使用，导出C式命名