1. 数据的共享
2. 静态数据类型
3. 用处：静态数据成员可以被一个类的所有对象共同拥有，共同访问，从而可以达成计数等目的。专门存储在静态存储区，因此它属于整个类。
4. 定义： public: static int GetNum();

Private: static int counter;

1. 使用： 静态数据成员必须在类外初始化，而不能在构造函数中初始化

且更推荐将初始化放在测试代码中

如（在类的定义之外）int People:: counter =0 （√）

//Attention: 只有在定义时才加上static标识，初始化等操作不加static

但是累加操作放在构造函数中就非常合理

4、访问形式：可以使用类名调用或对象名调用，更推荐使用类名调用

//Attention: 静态数据没有this指针，因为它不需要依赖于某个对象来调用

也因为这样，静态数据的函数中也就不能对普通成员函数进行操作

//Special Thinking: 析构函数中可以写入counter--等，在对象消失析构时便能使计数器自减

1. 友元关系
2. 友元函数：

在类内部声明：friend float total(boat , car)

在类外实现：float total(&boat , &car){ return boat.weight+car.weight; }

使用时就像是一个普通函数，作用于全局

1. 友元类：

在A类的声明内声明：friend class B;

**当B是A的友元类时，B的成员函数都是A的友元函数**

//Attention: ①友元类的使用会损害数据的保护原则

②不能传递，不能继承，非双向

（即A把B当成friend，而B不把A当成friend，则B能访问A的函数而 A不能访问B的函数）

1. 数据的保护
2. 常对象：const 类型的对象为常对象，无法被变易函数修改其数据
3. 常成员函数：const类型的函数可以被常对象调用，比如print( );
4. 常数据成员： 常数据成员的初始化必须由预定参数来进行，且初始化之后也不能被 成员函数修改它的值
5. 常参数：Const函数即使是引用了参数的地址，也不会对参数本身造成改变

//Attention: 如果要使用已有对象来创建新对象，则不可以使用原来的构造函数

而是需要构造一个新的构造函数

同时为了保护已有对象的值，新的构造函数中可采用const常 参数

1. 左值运算：**当函数的返回值是地址时**，可以将函数放在左侧而数据值放在右侧

即将右侧数据赋值给左侧地址，会导致原地址变量的修改。

//指针数据类型必须先分配空间（new）才能让其储存数据