1. 静态成员
   1. 静态成员变量
      1. 所有对象都共享同一份数据
      2. 编译阶段就分配内存
      3. 类内声明、类外初始化
      4. 访问方式有两种：通过对象访问、通过类名访问
      5. 静态成员变量也是有访问权限
   2. 静态成员函数
      1. 所有对象都共享同一份函数
      2. 静态成员函数 只可以访问 静态成员变量，不可以访问非静态成员变量
      3. 静态成员函数 也是有访问权限的
      4. 静态成员函数 有两种访问方式：通过对象 、通过类名
2. 单例模式 – 主席类案例
   1. 通过一个类 只能实例化唯一的一个对象
   2. 私有化
      1. 默认构造
      2. 拷贝构造
      3. 唯一实例指针
   3. 对外提供 getInstance 接口，将指针返回
3. 单例模式 – 打印机案例
   1. 和主席类案例一样设计单例模式
   2. 提供打印功能并且统计打印次数
4. C++对象模型初探
   1. 类中的成员变量 和 成员函数 是分开存储的
   2. 只有非静态成员变量 属于类对象上
   3. 空类的sizeof结果 1
5. this指针
   1. this指针 指向 被调用的成员函数 所属的对象
   2. this指针可以解决名称冲突
   3. this指针 隐式加在每个成员函数中
   4. \*this 就是本体
   5. p1.personAddPerson(p2).personAddPerson(p2).personAddPerson(p2); //链式编程
6. 空指针访问成员函数
   1. 如果成员函数中没有用到this指针，可以用空指针调用成员函数
   2. 如果成员函数中用到了this，那么这个this需要加判断，防止代码down掉
7. 常对象和常函数
   1. 常函数
      1. 成员函数 声明后面加const
      2. void showPerson() **const**
      3. const目的是为了修饰成员函数中的this指针，让指针指向的值不可以修改
      4. 有些属性比较特殊，依然在常函数或者常对象中可以修改，需要加入关键字 mutable
   2. 常对象
      1. const Person p
      2. 常对象也不许修改成员属性
      3. 常对象只能调用常函数
   3. 对于成员函数 ，可不可以 用static 和 const同时修饰 ，不可以
8. 友元
   1. 全局函数作为友元函数
      1. 利用friend关键字让全局函数 goodGay作为本类好朋友，可以访问私有成员
      2. friend void goodGay(Building \* buliding);
   2. 类作为友元类
      1. 让goodGay类作为 Building的好朋友，可以访问私有成员
      2. friend class GoodGay;
   3. 类中的成员函数作为友元函数
      1. //让GoodGay类中的 visit成员函数作为友元
      2. friend void GoodGay::visit();