1. 静态成员变量和静态成员函数
   1. 静态成员变量
      1. 编译阶段分配内存
      2. 所有对象共享数据
      3. 通过对象访问、通过类名访问
      4. 有权限控制
      5. 类内声明 类外初始化
   2. 静态成员函数
      1. 可以访问静态成员变量，不可以方法普通成员变量
      2. 普通成员函数 都可以访问
      3. 静态成员函数也有权限
      4. 可以通过对象访问，也可以通过类名进行访问
2. 单例模式案例 – 主席
   1. 目的 为了让类中只有一个实例，实例不需要自己释放
   2. 将 默认构造 和 拷贝构造 私有化
   3. 内部维护一个 对象指针
   4. 私有化唯一指针
   5. 对外提供 getInstance方法来访问这个指针
   6. 保证类中只能实例化唯一一个对象
3. 单例模式案例 – 打印机案例
   1. 类似主席案例
   2. 提供打印功能
   3. 提供统计打印次数功能
4. C++对象模型初探
   1. 成员变量和成员函数是分开存储的
   2. 空类的大小 1
   3. 只有非静态成员属性才属于对象身上
5. this指针使用
   1. 指针永远指向当前对象
   2. 解决命名冲突
   3. \*this 指向对象本体
   4. 非静态的成员函数才有this指针
6. 空指针访问成员函数
   1. 如果成员函数没有用到this，那么空指针可以直接访问
   2. 如果成员函数用的this指针，就要注意，可以加if判断，如果this为NULL就return
7. 常函数 常对象
   1. 常函数 void func() const {} 常函数
   2. 常函数 修饰是this指针 const Type \* const this
   3. 常函数 不能修改this指针执行的值
   4. 常对象 在对象前 加入 const修饰 const Person p1
   5. 常对象 不可以调用普通的成员函数
   6. 常对象 可以调用常函数
   7. 用mutable修饰的关键字是在常函数可以修改的
8. 友元
   1. 全局函数做友元函数
      1. 全局函数写到 类中做声明 并且最前面写关键字 friend
   2. 让整个类 做友元类
      1. friend class 类名
      2. 友元类 是单向，不可传递的
   3. 让成员函数做友元函数
      1. friend void goodGay::visit();