1. 静态成员
   1. 静态成员变量
      1. 在成员变量前 + static
      2. 共享一份数据
      3. 在编译阶段 分配内存
      4. 类内声明、类外初始化
      5. 访问模式 有两种
         1. 通过对象
         2. 通过类名
      6. 静态成员变量也是有访问权限
   2. 静态成员函数
      1. 在成员函数前加 + static
      2. 这个函数 也只有一份
      3. 访问模式 有两种
         1. 通过对象
         2. 通过类名
      4. 静态成员函数 只能访问静态成员变量、不可以访问非静态成员变量
      5. 静态成员函数 也是有访问权限的
2. 单例模式 --- 主席类
   1. 一个类中 只有唯一的一个实例对象
   2. 私有化
      1. 默认构造函数
      2. 拷贝构造函数
      3. 唯一主席指针 变成只读状态
   3. 对外提供公共接口
      1. static Chairman \* getInstance()
   4. 测试
3. 单例模式 --- 打印机案例
   1. 实现单例模式
   2. 对外提供打印功能
   3. 并且统计打印次数
4. C++对象模型初探
   1. 成员变量和成员函数分开存储
   2. 一个类中 只有非静态成员变量属于类大小，其他都不计算在类内部
   3. 空类的大小 1
5. this指针
   1. this指针指向的是被调用的成员函数所属的对象
   2. \*this 对象本体
   3. this可以解决名称冲突
   4. 链式编程
      1. p1.AddAge(p2).AddAge(p2).AddAge(p2); //链式编程
6. 空指针访问成员函数
   1. 如果是一个空指针
   2. 可以访问 没有this的一些成员函数
   3. 如果函数中用到了this指针，程序会down掉，如果想让代码严谨，加判断
7. 常函数和常对象
   1. this指针本质 指针常量 type \* const this
      1. 指针的指向是不可以修改的，指针指向的值可以改
      2. 如果想让指针指向的值也不能修改 const type \* const this
      3. 需要将成员函数改成常函数
      4. 成员函数的后面 加 const
   2. 有些特例的属性，即使是常函数 或者 常对象 也是可以修改的，需要在这个属性前加关键字 mutable
   3. 常对象 const Person p;
   4. 常对象 只能调用常函数，不可以调用普通的成员函数
8. 友元
   1. 全局函数作为友元函数
      1. 有一个全局函数 作为本类的友元函数，可以访问到私有内容
      2. 类的内部写入： friend + 函数声明
   2. 类作为友元类
      1. 告诉编译器 goodGay类是 本类的好朋友，可以访问到里面私有内容
      2. 在类内部 写入 ： friend class goodGay;
   3. 成员函数 作为友元函数
      1. //告诉编译器 goodGay类中的visit成员函数 可以访问本类中的私有内容
      2. friend void goodGay::visit();