1 静态成员变量和静态成员函数
- 1.1 静态成员变量
  - 1.1.1 编译阶段分配内存
  - 1.1.2 所有对象共享数据
  - 1.1.3 通过对象访问、通过类名访问
  - 1.1.4 有权限控制
  - 1.1.5 类内声明 类外初始化
- 1.2 静态成员函数
  - 1.2.1 可以访问静态成员变量，不可以方法普通成员变量
  - 1.2.2 普通成员函数 都可以访问
  - 1.2.3 静态成员函数也有权限
  - 1.2.4 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
- 2.1 目的 为了让类中只有一个实例，实例不需要自己释放
- 2.2 将 默认构造 和 拷贝构造 私有化
- 2.3 内部维护一个 对象指针
- 2.4 私有化唯一指针
- 2.5 对外提供 getInstance 方法来访问这个指针
- 2.6 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
- 3.1 类似主席案例
- 3.2 提供打印功能
- 3.3 提供统计打印次数功能

4 C++对象模型初探
- 4.1 成员变量和成员函数是分开存储的
- 4.2 空类的大小 1
- 4.3 只有非静态成员属性才属于对象身上

5 this 指针使用
- 5.1 指针永远指向当前对象
- 5.2 解决命名冲突
- 5.3 *this 指向对象本体
- 5.4 非静态的成员函数才有 this 指针

6 空指针访问成员函数
- 6.1 如果成员函数没有用到 this，那么空指针可以直接访问
- 6.2 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
- 7.1 常函数 void func() const {} 常函数
- 7.2 常函数 修饰是 this 指针 const Type * const this
- 7.3 常函数 不能修改 this 指针执行的值
- 7.4 常对象 在对象前 加入 const 修饰 const Person p1
- 7.5 常对象 不可以调用普通的成员函数
- 7.6 常对象 可以调用常函数

7.7 用 mutable 修饰的关键字是在常函数可以修改的

8 友元

8.1 全局函数做友元函数

8.1.1 全局函数写到 类中做声明 并且最前面写关键字 friend

8.2 让整个类 做友元类

8.2.1 friend class 类名

8.2.2 友元类 是单向，不可传递的

8.3 让成员函数做友元函数

8.3.1 `friend void goodGay::visit();`