在设计C++软件时，经常会出现一些问题.如何让构造函数和非成员函数像虚函数一样工作？

如何限制类的实例数量？

如何防止在堆上创建对象？

你怎么能保证它们会在那里被创造出来呢？

如何创建对象，使其在调用其他类的成员函数时自动执行某些操作？

如何让不同的对象共享数据结构，同时让客户端产生每个对象都有自己的副本的错觉？如何区分operator[]的读和写用法？

如何创建一个行为依赖于多个对象的动态类型的虚函数？

第25项：虚拟化构造函数和非成员函数。

工厂模式，策略模式。

第26条：限制类的对象数量。

构建C++的哲学支柱之一是你不应该为你不使用的东西付费的想法，将像我们的打印机这样的对象定义为函数中的静态对象是坚持这一哲学的一种方式。这是一种你应该随时坚持的哲学。

单例模式。

文本, 信件

描述已自动生成

第27条：要求或禁止基于堆的对象。

**“对象是否在堆上”这一概念并不总是明确的**，因为 C++ 本身并没有直接的语言机制来强制对象在堆上或栈上分配。因此，需要通过一些技巧来实现这些需求。

手机屏幕截图

描述已自动生成

虽然“堆”和“栈”的概念在编程中非常常见，但 **C++ 标准并没有严格定义它们**。标准只规定了 new 和 delete 的行为，而具体的实现细节（如内存布局、分配策略等）由编译器和运行时环境决定。

第28项：智能指针