C 箴言：通过composition模拟“ has-a ”PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/245/2021_2022_C___E7_AE_B4_E8_A8_80_EF_c97_245960.htm composition（复合）是在 objects of one type（一个类型的对象）包含 objects of another type（另一个类型的对象）时，types（类型）之间的关系。例如： class Address { ... }. // where someone lives class PhoneNumber { ... }. class Person { public: ... private: std::string name. // composed object Address address. // ditto PhoneNumber voiceNumber. // ditto PhoneNumber faxNumber. // ditto }. 此例之中，Person objects（对象）由 string，Address，和 PhoneNumber objects（对象）组成。在程序员中，术语 composition（复合）有很多同义词。它也可以称为 layering，containment，aggregation，和 embedding。《C 箴言：确保公开继承模拟“ is-a ”》解释了 public inheritance（公有继承）意味着 "is-a"。composition（复合）也有一个含意。实际上，他有两个含意。composition（复合）既意味着 "has-a"（有一个），又意味着 "is-implemented-in-terms-of"（是根据……实现的）。这是因为你要在你的软件中处理两个不同的 domains（领域）。你程序中的一些 objects（对象）对应你所模拟的世界里的东西，例如，people（人），vehicles（交通工具），video frames（视频画面）等等。这样的 objects（对象）是 application domain（应用领域）的部分。另外的 objects（对象）纯粹是 implementation artifacts（实现的产物），例如，buffers（缓冲区），mutexes（互斥体），search trees（搜索树）等等。这些各类 objects（对象）定义应你的软件的

implementation domain（实现领域）。当 composition（复合）发生在 application domain（应用领域）的 objects（对象）之间，它表达一个 has-a（有一个）的关系，当它发生在 implementation domain（实现领域），它表达一个 is-implemented-in-terms-of（是根据……实现的）的关系 上面的 Person class（类）示范了 has-a（有一个）的关系。一个 Person object（对象）has a（有一个）名字，一个地址，以及语音和传真电话号码。你不能说一个人 is a（是一个）名字或一个人 is an（是一个）地址。你可以说一个人 has a（有一个）名字和 has an（有一个）地址。大多数人对此区别不难理解，所以混淆 is-a和 has-a（有一个）之间的角色的情况非常少见。 is-a和 is-implemented-in-terms-of（是根据……实现的）之间的区别稍微有些棘手。例如，假设你需要一个类的模板来表现相当小的 objects（对象）的 sets，也就是说，排除重复的集合。因为 reuse（复用）是一件受人欢迎的事情，你的第一个直觉就是使用标准库中的 set template（模板）。当你能使用已经被写好的东西时，为什么还要写一个新的 template（模板）呢？ 不幸的是，set 的典型实现导致每个元素三个指针的开销。这是因为 sets 通常被作为 balanced search trees（平衡搜索树）来实现，这允许它们保证 logarithmic-time（对数时间）的 lookups（查找），insertions（插入）和 erasures（删除）。当速度比空间更重要的时候，这是一个合理的设计，但是当空间比速度更重要时，对你的程序来说就有问题了。因而，对你来说，标准库的 set 为你提供了不合理的交易。看起来你终究还是要写你自己的 template（模板）。 reuse（复用）依然是一件受人欢迎的事情。作为 data structure（数据结构

）的专家，你知道实现 sets 的诸多选择，其中一种是使用 linked lists（线性链表）。你也知道标准的 C 库中有一个 list template（模板），所以你决定（复）用它。 具体地说，你决定让你的新的 Set template（模板）从 list 继承。也就是说，Set 将从 list 继承。毕竟，在你的实现中，一个 Set object（对象）实际上就是一个 list object（对象）。于是，你就像这样声明你的 Set template（模板）： template // the wrong way to use list for Set class Set: public std::list { ... }. 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com