C 箴言：防止异常离开析构函数 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/133/2021_2022_C___E7_AE_B4_E8_A8_80_EF_c97_133683.htm C 并不禁止从析构函数中引发异常，但是这确实妨碍了实践。至于有什么好的理由，考虑： class Widget { public: ... ~Widget() { ... } // assume this might emit an exception }. void doSomething() { std::vector v. ... } // v is automatically destroyed here 当 vector v 被析构时，它有责任销毁它包含的所有 Widgets。假设 v 中有十个 Widgets，在销毁第一个的时候，抛出一个异常。其他 9个 Widgets 仍然必须被销毁（否则他们持有的任何资源将被泄漏），所以 v 应该调用它们的析构函数。但是假设在这个调用期间，第二个 Widgets 的析构函数又抛出一个异常。现在有两个异常同时在活动中，对于 C 来说这太多了。在非常巧合的条件下发生这样两个同时活动的异常，程序的执行会终止或者引发未定义行为。在本例中，将引发未定义行为。与此相同，使用任何标准库容器（比如，list，set），任何 TR1中的容器，甚至是一个数组，都可能会引发未定义问题。并非必须是容器或数组才会陷入麻烦。程序夭折或未定义行为是析构函数引发异常的结果，即使没有使用容器或数组也会如此。C 不喜欢引发异常的析构函数。 这比较容易理解，但是如果你的析构函数需要执行一个可能失败而抛出异常的操作，该怎么办呢？例如，假设你与一个数据库连接类一起工作： class DBConnection { public: ... static DBConnection create(). // function to return // DBConnection objects. params // omitted for simplicity void close(). // close connection. throw an }. // exception if closing