C 编程人员容易犯的10个C#错误1 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/134/2021_2022_C___E7_BC_96_E7_A8_8B_E4_c97_134347.htm 我们知道， C#的语法与C 非常相似，实现从C 向C#的转变，其困难不在于语言本身，而在于熟悉.NET的可管理环境和对.NET框架的理解。 尽管C#与C 在语法上的变化是很小的，几乎不会对我们有什么影响，但有些变化却足以使一些粗心的C 编程人员时刻铭记在心。在本篇文章中我们将讨论C 编程人员最容易犯的十个错误。 陷阱1： 没有明确的结束方法 几乎可以完全肯定地说，对于大多数C 编程人员而言，C#与C 最大的不同之处就在于碎片收集。这也意味着编程人员再也无需担心内存泄露和确保删除所有没有用的指针。但我们再也无法精确地控制杀死无用的对象这个过程。事实上，在C#中没有明确的destructor。如果使用非可管理性资源，在不使用这些资源后，必须明确地释放它。对资源的隐性控制是由Finalize方法（也被称为finalizer）提供的，当对象被销毁时，它就会被碎片收集程序调用收回对象所占用的资源。 finalizer应该只释放被销毁对象占用的非可管理性资源，而不应牵涉到其他对象。如果在程序中只使用了可管理性资源，那就无需也不应当执行Finalize方法，只有在非可管理性资源的处理中才会用到Finalize方法。由于finalizer需要占用一定的资源，因此应当只在需要它的方法中执行finalizer。 直接调用一个对象的Finalize方法是绝对不允许的（除非是在子类的Finalize中调用基础类的Finalize。），碎片收集程序会自动地调用Finalize。 从语法上看，C#中的destructor与C 非常相似，但其实它们

是完全不同的。C#中的destructor只是定义Finalize方法的捷径。因此，下面的二段代码是有区别的： ~MyClass() { // 需要完成的任务 } MyClass.Finalize() { // 需要完成的任务 base.Finalize(). } 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