C 常见的内存错误及其对策计算机等级考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/645/2021_2022_C___E5_B8_B8_E8_A7_81_E7_c97_645553.htm 编辑特别推荐: 全国计算机等级考试（等考）指定教材 全国计算机等级考试学习视频 全国计算机等级考试网上辅导招生 全国计算机等级考试时间及科目预告 百考试题教育全国计算机等级考试在线测试平台 全国计算机等级考试资料下载 全国计算机等级考试论坛 发生内存错误是件非常麻烦的事情。编译器不能自动发现这些错误，通常是在程序运行时才能捕捉到。而这些错误大多没有明显的症状，时隐时现，增加了改错的难度。有时用户怒气冲冲地把你找来，程序却没有发生任何问题，你一走，错误又发作了。常见的内存错误及其对策如下： 内存分配未成功，却使用了它。编程新手常犯这种错误，因为他们没有意识到内存分配会不成功。常用解决办法是，在使用内存之前检查指针是否为NULL。如果指针p 是函数的参数，那么在函数的入口处用assert(p!=NULL)进行检查。如果是用malloc 或new 来申请内存，应该用if(p==NULL)或if(p!=NULL)进行防错处理。 内存分配虽然成功，但是尚未初始化就引用它。犯这种错误主要有两个起因：一是没有初始化的观念；二是误以为内存的缺省初值全为零，导致引用初值错误（例如数组）。 内存的缺省初值究竟是什么并没有统一的标准，尽管有些时候为零值，我们宁可信 其无不可信其有。所以无论用何种方式创建数组，都别忘了赋初值，即便是赋零值也不可省略，不要嫌麻烦。 内存分配成功并且已经初始化，但操作越过了内存的边界。例如在使用数组时经常发生下标“ 多1 ”或者“

少1 ” 的操作。特别是在for 循环语句中，循环次数很容易搞错，导致数组操作越界。 忘记了释放内存，造成内存泄露。含有这种错误的函数每被调用一次就丢失一块内存。刚开始时系统的内存充足，你看不到错误。终有一次程序突然死掉，系统出现提示：内存耗尽。动态内存的申请与释放必须配对，程序中malloc 与free 的使用次数一定要相同，否则肯定有错误（new/0delete 同理）。 释放了内存却继续使用它。 有三种情况： （1）程序中的对象调用关系过于复杂，实在难以搞清楚某个对象究竟是否已经释放了内存，此时应该重新设计数据结构，从根本上解决对象管理的混乱局面。 （2）函数的return 语句写错了，注意不要返回指向 “ 栈内存 ” 的 “ 指针 ” 或者 “ 引用 ”，因为该内存在函数体结束时被自动销毁。 （3）使用free 或0delete 释放了内存后，没有将指针设置为NULL。导致产生 “ 野指针 ”。 【规则1】用malloc 或new 申请内存之后，应该立即检查指针值是否为NULL。防止使用指针值为NULL 的内存。 【规则2】不要忘记为数组和动态内存赋初值。防止将未被初始化的内存作为右值使用。 【规则3】避免数组或指针的下标越界，特别要当心发生 “ 多1 ” 或者 “ 少1 ” 操作。 【规则4】动态内存的申请与释放必须配对，防止内存泄漏。 【规则5】用free 或0delete 释放了内存之后，立即将指针设置为NULL，防止产生 “ 野指针 ”。