C#与C 资源管理方式对阵GC对比RAII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_23_E4_B8_8EC___E8_B5_c97_645678.htm 资源管理 在C语言中，资源管理是一个极为繁琐易错的工作，大多复杂的C系统都面临着内存泄露、悬挂指针等问题。这是一方面是由底层语言的特点决定.另一方面也是由于C语言特性相对较少，严重依赖程序员进行正确的资源管理，缺乏有效的支持手段。 C#和C 两门语言的定位不同，它们在资源管理方面采取了两种截然不同的方式：一为GC，一为RAII。GC让程序建立在更高的抽象层次上，使资源管理变得更方便，更安全.而RAII则保留了C的底层能力，同时在C 特性的支持下提供了简单有效的资源管理方式。我们知道C 最激烈的批评往往来自于C社区，而在我看来C程序员可以不接受虚函数，不接受模板，但有什么理由不接受RAII呢?可以说RAII是C 相对C来说几乎无副作用的明显进步。 下面就从GC开始： 引用代替指针 C#通过CLR管理托管内存，用引用抽象代替指针间接操作托管内存，让程序员在更高的层次上安全地使用资源。这使得C#失去了直接管理内存的能力，但换来了以下好处： 1.类型安全：在C/C 中可以通过类型转换把整数或其他类型的指针转换为特定类型的指针，这意味着指针是非类型安全的，必须由程序员来保证指针代表的内存空间的合法性。而C#引用可以看作是类型安全的指针，as运算符可以保证转换的类型安全性。 2.内存整理：创建对象需要从堆中动态分配连续的内存空间，由于不同对象的内存大小是不同的，常见的首次匹配和最优匹配堆分配算法都会造成堆中的内存碎片问题。碎片

的存在使实际可用内存小于物理内存，所以应尽量减少碎片的产生。一个方向是设计更好的内存分配算法.另一个方向是通过周期性地进行内存整理调整优化。在C/C 中，由于指针代表了绝对地址，因此不存在通用的内存整理算法.而C#屏蔽了指针，通过引用操作对象，就使得内存整理成为可能。PS：这并不意味着C/C 内存分配就弱于C#，C/C 程序可以为某种类型的对象设计专用的内存分配方式，甚至把对象指定分配到某一物理地址空间，这些都是C#不具备的。 托管和非托管资源 在C#中，资源分为托管资源和非托管资源两种。GC在回收无用对象资源时，可以自动回收托管资源(比如托管内存)，但对于非托管资源(比如Socket、文件、数据库连接)必须在程序中显式释放。 托管资源的回收首先需要GC识别无用对象，然后回收其资源。一般无用对象是指通过当前的系统根对象和调用堆栈对象不可达的对象。对象有一个重要的特点导致无用对象判断的复杂性：对象间的相互引用!如果没有相互引用，就可以通过“ 引用计数 ”这种简单高效的方式实现无用对象的判断，并实现实时回收。正是由于相互引用的存在导致GC需要设计更为复杂的算法，这样带来的最大问题在于丧失了资源回收的实时性，而变成一种不确定的方式。对于非托管资源的释放，C#提供了两种方式： 1.Finalizer：写法貌似C 的析构函数，本质上却相差甚远。Finalizer是对象被GC回收之前调用的终结器，初衷是在这里释放非托管资源，但由于GC运行时机的不确定性，通常会导致非托管资源释放不及时。另外，Finalizer可能还会有意想不到的副作用，比如：被回收的对象已经没有被其他可用对象所引用，但Finalizer内部却把它重新变成可用，这就破坏了GC垃圾收

集过程的原子性，增大了GC开销。 2.Dispose Pattern：C#提供using关键字支持Dispose Pattern进行资源释放。这样能通过确定的方式释放非托管资源，而且using结构提供了异常安全性。所以，一般建议采用Dispose Pattern，并在Finalizer中辅以检查，如果忘记显式Dispose对象则在Finalizer中释放资源。可以说，GC为程序带来安全方便的同时也付出了不小的代价：一则丧失了托管资源回收的实时性，这在实时系统和资源受限系统中是致命的.二则没有把托管资源和非托管资源的管理统一起来，造成概念割裂。C 的定位之一是底层开发能力，所以不难理解GC并没有成为C 的语言特性。虽然我们在 C 0x和各种第三方库都能看到GC的身影，但GC对于C 来讲并不是那么重要，至多是一个有益的补充。C 足以傲视C，并和C# GC一较高下的是它的RAII。 栈语义 在介绍RAII之前，让我们先来看一道C 面试题：“ 重构下面的代码，在保证正确释放资源的情况下，去掉多余的try catch ”//C void f(){ try{ int *ptr = new int(123). …//do something with ptr 0delete ptr. } catch { 0delete ptr. } } 代码中new int在堆上分配内存，并通过0delete小心翼翼地释放内存。这是典型的C风格的C 代码，虽然用了try、catch等高级语法，但资源管理方式依旧是C。按C 特有的方式可以重构成这样： //C 资源管理方式 //定义资源代理类模板 template class Resource{ public: Resource(T *ptr) { this- 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