C 程序设计最佳实践 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/261/2021_2022_C___E7_A8_8B_E5_BA_8F_E8_c67_261312.htm 随着计算机语言的发展，我们现在编写一个程序越来越容易了。利用一些软件开发工具，往往只要通过鼠标的拖拖点点，计算机就会自动帮你生成许多代码。但在很多时候，计算机的这种能力被滥用了，我们往往只考虑把这个程序搭起来，而不去考虑程序的性能如何，程序是否足够的健壮。而此节课的目的主要是介绍一些编码的经验，让大家编写的程序更加健壮和高性能。 1、Prefer const and inline to #define 在C 编程中应该尽量使用const和inline来代替#define，尽量做到能不用#define就不用。#define常见的用途有“ 定义常量 ”以及“ 定义宏 ”，但其中存在诸多的弊病。 第一，查错不直观，不利于调试。Define的定义是由预处理程序处理的，作的是完全的文本替换，不做任何的类型检查。在编译器处理阶段，define定义的东西已经被完全替换了，这样在debug的时候就看不到任何的相关信息，即跟踪时不能step into宏。例如，把ASPECT_RATIO用define定义成1.653，编译器就看不到ASPECT_RATIO这个名字了。如果编译器报1.653错，那么就无从知道此1.653来自于何处。在真正编码的时候应该使用如下的语句来定义：static const double ASPECT_RATIO = 1.653. 第二，没有任何类型信息，不是type safe。因为它是文本级别的替换，这样不利于程序的维护。 第三，define的使用很容易造成污染。比如，如果有两个头文件都定义了ASPECT_RATIO, 而一个CPP文件又同时包含了这两个头文

件，那么就会造成冲突。更难查的是另外一种错误，比如有如下的代码： // in header file def.h #define Apple 1 #define Orange 2 #define Pineapple 3 … // in some cpp file that includes the def.h enum Colors {White, Black, Purple, Orange}. 在.h文件中Orange被定义成水果的一种，而在.cpp文件中Orange又成为了一种颜色，那么编译器就会把此处的Orange替换成2，编译可能仍然可以通过，程序也能够运行，但是这就成了一个bug，表现出古怪的错误，且很难查错。再比如定义了一个求a与b哪个数大的宏，#define max(a,b) ((a) > (b) ? (a) : (b)) int a = 5, b = 0. max(a, b). max(a, b 10). 在上面的操作中，max(a, b). 语句中a被 了两次，而max(a, b 10). 语句中a只加了一次，这样在程序处理中就很有可能成为一个bug，且此bug也非常的难找。在实际编码时可以使用如下的语句来做： template inline const Tamp. a, const Tamp. UPInt::operator () { *this = 1. // increment return *this. // fetch } // postfix form: fetch and increment const UPInt UPInt::operator (int) { UPInt oldValue = *this. // fetch (*this). // increment return oldValue. // return what was fetched } 一般情况下不需要区分是先，还是后，但是我们在编写程序的时候最好能习惯性的将其写成 i的形式，如在使用STL中的iterator时，prefix与postfix会有相当大的性能差异。请不要小看这些细节，实际在编写程序的时候，若不注意具体细节，你会发现程序的性能会非常的低。但要注意，虽然在大多数情况下可以用prefix来代替postfix，但有一种情况例外，那就是有[]操作符时，比如gzArray [index] 是不等于gzArray[index]的。