C 箴言：只要可能就用const 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_133677.htm 关键字 const 非常多才多艺。在类的外部，你可以将它用于全局常量或命名空间常量，就像那些在文件、函数或模块范围内被声明为 static 的对象。在类的内部，你可以将它用于 static 和 non-static 数据成员上。对于指针，你可以指定这个指针本身是 const，或者它所指向的数据是 const，或者两者都是，或者都不是。 char greeting[] = "Hello". char *p = greeting. // non-const pointer, // non-const data const char *p = greeting. // non-const pointer, // const data char * const p = greeting. // const pointer, // non-const data const char * const p = greeting. // const pointer, // const data 这样的语法本身其实并不像表面上那样反复无常。如果 const 出现在 * 左边，则指针指向的内容为常量；如果 const 出现在 * 右边，则指针自身为常量；如果 const 出现在 * 两边，则两者都为常量。 当指针指向的内容为常量时，一些人将 const 放在类型之前，另一些人将 const 放在类型之后 * 之前。两者在意义上并没有区别，所以，如下两个函数具有相同的参数类型： void f1(const Widget *pw). // f1 takes a pointer to a // constant Widget object void f2(Widget const *pw). // so does f2 因为它们都存在于实际的代码中，你应该习惯于这两种形式。 STL iterators 以指针为原型，所以一个 iterator 在行为上非常类似于一个 T* 指针。声明一个 iterator 为 const 就类似于声明一个指针为 const（也就是说声明一个 T* const 指针）：不能将 iterator 指向另外一件不同的东西，但是它所指向的东西本身

可以变化。如果你要一个 iterator 指向一个不能变化的东西（也就是 const T* 的 STL 对等物），你应该用 const_iterator：std::vector vec. ... const std::vector::iterator iter = // iter acts like a T* const vec.begin(). *iter = 10. // OK, changes what iter points to iter. // error! iter is const std::vector::const_iterator cIter = //cIter acts like a const T* vec.begin(). *cIter = 10. // error! *cIter is const cIter. // fine, changes cIter 