C 处理for循环作用域 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/468/2021_2022_C___E5_A4_84_E7_90_86f_c67_468683.htm 在 C 标准制定之前，在 for 循环中声明的变量在循环外也可以访问。例如： for (int n=0. n〈MAX. n) { //..do something } n. //OK in pre-standard C . illegal in ISO C 然而，在 ISO C 中，for 循环变量的作用域被限制为循环本身。虽然这一改变不可否认地具有其意义，但是它却影响到了老代码以及新代码。下面我将示范一些迁移技术帮助你处理这一改动。 遗留代码 对于那些使用标准制定之前的作用域规则的遗留代码，如果使用与标准兼容的编译器编译它们，那么就有可能出现错误。解决这一问题的最好方法是修改代码。但是，代码修正需要彻底测试，而且有时候还会招致一连串的缺陷和编译错误。 如果不想处理这一难于解决的问题，同时又希望升级编译器，那么不妨检查一下能否有办法恢复标准制定之前 for 循环变量的行为。如果确实使用了这种变量，那么打开这个选项，然后（使用注释）明确地在代码记下这一事实，以使将来的程序员知道如何正确地编译这一代码。 If you ’ re wary of relying on compilers ’ favors, there ’ s an alternative patch: move the variable ’ s definition outside the for-loop: 如果不愿依赖编译器的帮助，那么还有一种替代方案：将变量的定义移至 for 循环之外： int n=0. //originally was inside the for loop for (/*n was here*/. n〈MAX. n) { //..do something } int x=n. //OK 要确保在有改动的地方添加一句描述性注释。 作用域规则可能也会影响新代码。比如说，假设我们需要使用一个依然保持标准制定之前行为的编译器来编译

新代码。 为了确保 for 循环变量不在循环之外被访问，我们可以将整个循环之外包围一对大括号。 {//restrict for-loop variables ’ scope for (int n=0. n〈MAX. n) { //..do something } }//restrict for-loop variables ’ scope With the help of conditional compilation, you can use a macro that inserts braces only when they are needed: 在条件编译的帮助下，我们可以使用一个宏来控制只在需要的地方插入大括号： #if defined(OLD_FOR_SCOPING) #define OPEN_FOR_GUARD { #define CLOSE_FOR_GUARD } #else #define OPEN_FOR_GUARD #define CLOSE_FOR_GUARD #endif OPEN_FOR_GUARD for (int n=0. n〈MAX. n) { //..do something } CLOSE_FOR_GUARD 如果你正在使用新代码以及与标准兼容的编译器，那么新作用域规则将不会带来任何问题。然而，当涉及到遗留代码或老编译器时，你可以应用我在这个小技巧中描述的技术绕开这一问题 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