C 入门C 代码优化计算机等级考试 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_85_A5_E9_97_A8C_c97_645727.htm 谈到优化，很多人都会直接想到汇编。难道优化只能在汇编层次吗？当然不是，C 层次一样可以作代码优化，其中有些常常是意想不到的。在C 层次进行优化，比在汇编层次优化具有更好的移植性，应该是优化中的首选做法。 确定浮点型变量和表达式是float型 为了让编译器产生更好的代码(比如说产生3DNow! 或SSE指令的代码)，必须确定浮点型变量和表达式是 float 型的。要特别注意的是，以"；F"；或"；f"；为后缀（比如：3.14f）的浮点常量才是 float 型，否则默认是 double 型。为了避免 float 型参数自动转化为 double，请在函数声明时使用 float。 使用32位的数据类型 编译器有很多种，但它们都包含的典型的32位类型是：int，signed，signed int，unsigned，unsigned int，long，signed long，long int，signed long int，unsigned long，unsigned long int。尽量使用32位的数据类型，因为它们比16位的数据甚至8位的数据更有效率。 明智使用有符号整型变量 在很多情况下，你需要考虑整型变量是有符号还是无符号类型的。比如，保存一个人的体重数据时不可能出现负数，所以不需要使用有符号类型。但是，如果是要保存温度数据，就必须使用到有符号的变量。 在许多地方，考虑是否使用有符号的变量是必要的。在一些情况下，有符号的运算比较快；但在一些情况下却相反。 比如：整型到浮点转化时，使用大于16位的有符号整型比较快。因为x86构架中提供了从有符号整型转化到浮点型的指令，但没有提供从无符号整型转

化到浮点的指令。看看编译器产生的汇编代码：不好的代码：编译前编译后 double x； mov [foo 4], 0 unsigned int i； mov eax, i x = i； mov [foo], eax flid qword ptr [foo] fstp qword ptr [x] 上面的代码比较慢。不仅因为指令数目比较多，而且由于指令不能配对造成的FLID指令被延迟执行。最好用以下代码代替：推荐的代码：编译前编译后 double x；fild dword ptr [i] int i；fstp qword ptr [x] x = i； 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