JAVA循环谜题34:被计数击倒了 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/138/2021_2022_JAVA_E5_BE_AA_E7_8E_AF_c97_138523.htm 谜题26和27中的程序一样，下面的程序有一个单重的循环，它记录迭代的次数，并在循环终止时打印这个数。那么，这个程序会打印出什么呢？ public class Count { public static void main(String[] args) { final int START = 2000000000. int count = 0. for (float f = START. f count . System.out.println(count). } } 表面的分析也许会认为这个程序将打印50，毕竟，循环变量（f）被初始化为2,000,000,000，而终止值比初始值大50，并且这个循环具有传统的“ 半开 ”形式：它使用的是 然而，这种分析遗漏了关键的一点：循环变量是float类型的，而非int类型的。回想一下谜题28，很明显，增量操作（f ）不能正常工作。F的初始值接近于Integer.MAX_VALUE，因此它需要用31位来精确表示，而float类型只能提供24位的精度。对如此巨大的一个float数值进行增量操作将不会改变其值。因此，这个程序看起来应该无限地循环下去，因为f永远也不可能解决其终止值。但是，如果你运行该程序，就会发现它并没有无限循环下去，事实上，它立即就终止了，并打印出0。怎么回事呢？ 问题在于终止条件测试失败了，其方式与增量操作失败的方式非常相似。这个循环只有在循环索引f比(float)(START 50)小的情况下才运行。在将一个int与一个float进行比较时，会自动执行从int到float的提升[JLS 15.20.1]。遗憾的是，这种提升是会导致精度丢失的三种拓宽原始类型转换的一种[JLS 5.1.2]。（另外两个是从long到float和从long到double。） f的初始值太大了

，以至于在对其加上50，然后将结果转型为float时，所产生的数值等于直接将f转换成float的数值。换句话说，(float)2000000000 == 2000000050，因此表达式f 订正这个程序非常简单，只需将循环变量的类型从float修改为int即可。这样就避免了所有与浮点数计算有关的不精确性： for (int f = START. f count . 如果不使用计算机，你如何才能知道2,000,000,050与2,000,000,000有相同的float表示呢？关键是要观察到2,000,000,000有10个因子都是2：它是一个2乘以9个10，而每个10都是5×2。这意味着2,000,000,000的二进制表示是以10个0结尾的。50的二进制表示只需要6位，所以将50加到2,000,000,000上不会对右边6位之外的其他为产生影响。特别是，从右边数过来的第7位和第8位仍旧是0。提升这个31位的int到具有24位精度的float会在第7位和第8位之间四舍五入，从而直接丢弃最右边的7位。而最右边的6位是2,000,000,000与2,000,000,050位以不同之处，因此它们的float表示是相同的。 这个谜题寓意很简单：不要使用浮点数作为循环索引，因为它会导致无法预测的行为。如果你在循环体内需要一个浮点数，那么请使用int或long循环索引，并将其转换为float或double。在将一个int或long转换成一个float或double时，你可能会丢失精度，但是至少它不会影响到循环本身。当你使用浮点数时，要使用double而不是float，除非你肯定float提供了足够的精度，并且存在强制性的性能需求迫使你使用float。适合使用float而不是double的时刻是非常非常少的。 对语言设计者的教训，仍然是悄悄地丢失精度对程序员来说是非常令人迷惑的。请查看谜题31有关这一点的深入讨论。 