链滴

转：计算机程序的思维逻辑 (5) - 小数计算为什么会出错?

作者：Angonger

原文链接：https://ld246.com/article/1487581097098

来源网站：链滴

违反直觉的事实
计算机之所以叫"计算"机就是因为发明它主要是用来计算的，"计算"当然是它的特长，在大家的象中，计算一定是非常准确的。但实际上，即使在一些非常基本的小数运算中，计算的结果也是不精的。
比如：
<blockquote>
float f = 0.1f*0.1f;
System.out.println(f);
</blockquote>
这个结果看上去，不言而喻，应该是 0.01，但实际上，屏幕输出却是 0.010000001，后面多了个1。
看上去这么简单的运算，计算机怎么会出错了呢？
简要答案
实际上，不是运算本身会出错，而是计算机根本就不能精确的表示很多数，比如 0.1 这个数。</>
计算机是用一种二进制格式存储小数的，这个二进制格式不能精确表示 0.1，它只能表示一个非接近 0.1 但又不等于 0.1 的一个数。
数字都不能精确表示，在不精确数字上的运算结果不精确也就不足为奇了。
0.1 怎么会不能精确表示呢？在十进制的世界里是可以的，但在二进制的世界里不行。在说二进之前，我们先来看下熟悉的十进制。
实际上，十进制也只能表示那些可以表述为 10 的多少次方和的数，比如 12.345，实际上表示的110+21+30.1+40.01+5*0.001，与整数的表示类似，小数点后面的每位置也都有一个位权，从左到右，依次为 0.1,0.01,0.001,...即 10^(-1), 10^(-2), 10^(-3)。
很多数，十进制也是不能精确表示的，比如 1/3, 保留三位小数的话，十进制表示是 0.333，但论后面保留多少位小数，都是不精确的，用 0.333 进行运算，比如乘以 3，期望结果是 1，但实际上是 0.999。
二进制是类似的，但二进制只能表示哪些可以表述为 2 的多少次方和的数，来看下 2 的次方的些例子：
| 2 的次方
| 十进制
|
| 2^(-1)
| 0.5
|
| 2^(-2)
| 0.25
|
| 2^(-3)
| 0.125
|
| 2^(-4)
| 0.0625
|
可以精确表示为 2 的某次方之和的数可以精确表示，其他数则不能精确表示。
为什么一定要用二进制呢？
为什么就不能用我们熟悉的十进制呢？在最最底层，计算机使用的电子元器件只能表示两个状态通常是低压和高压，对应 0 和 1，使用二进制容易基于这些电子器件构建硬件设备和进行运算。如果要使用十进制，则这些硬件就会复杂很多，并且效率低下。
有什么有的小数计算是准确的
如果你编写程序进行试验，你会发现有的计算结果是准确的。比如，我用 Java 写：
<blockquote>
System.out.println(0.1f+0.1f);
System.out.println(0.1f*0.1f);
</blockquote>

第一行输出 0.2，第二行输出 0.010000001。按照上面的说法，第一行的结果应该也不对啊？</
>
其实，这只是 Java 语言给我们造成的假象，计算结果其实也是不精确的，但是由于结果和 0.2
够接近，在输出的时候，Java 选择了输出 0.2 这个看上去非常精简的数字，而不是一个中间有很多 0
的小数。
在误差足够小的时候，结果看上去是精确的，但不精确其实才是常态。
怎么处理计算不精确
计算不精确，怎么办呢？大部分情况下，我们不需要那么高的精度，可以四舍五入，或者在输出
时候只保留固定个数的小数位。
如果真的需要比较高的精度，一种方法是将小数转化为整数进行运算，运算结束后再转化为小数
另外的方法一般是使用十进制的数据类型，这个没有统一的规范，在 Java 中是 BigDecimal，运算更
确，但效率比较低，本节就不详细说了。
二进制表示
我们之前一直在用"小数"这个词表示 float 和 double 类型，其实，这是不严谨的，"小数"是在
学中用的词，在计算机中，我们一般说的是"浮点数"。float 和 double 被称为浮点数据类型，小数运
被称为浮点运算。
为什么要叫浮点数呢？这是由于小数的二进制表示中，表示那个小数点的时候，点不是固定的，
是浮动的。
我们还是用 10 进制类比，10 进制有科学表示法，比如 123.45 这个数，直接这么写，就是固定
示法，如果用科学表示法，在小数点前只保留一位数字，可以写为 1.2345E2 即 1.2345*(10^2)，即
科学表示法中，小数点向左浮动了两位。
二进制中为表示小数，也采用类似的科学表示法，形如 m*(2^e)。m 称为尾数，e 称为指数。
数可以为真，也可以为负，负的指数表示哪些接近 0 的比较小的数。在二进制中，单独表示尾数部分
指数部分，另外还有一个符号位表示正负。
几乎所有的硬件和编程语言表示小数的二进制格式都是一样的，这种格式是一个标准，叫做 IEEE
54 标准，它定义了两种格式，一种是 32 位的，对应于 Java 的 float，另一种是 64 位的，对应于 Ja
a 的 double。
32 位格式中，1 位表示符号，23 位表示尾数，8 位表示指数。64 位格式中，1 位表示符号，52
位表示尾数，11 位表示指数。
在两种格式中，除了表示正常的数，标准还规定了一些特殊的二进制形式表示一些特殊的值，比
负无穷，正无穷，0，NaN (非数值，比如 0 乘以无穷大)。
IEEE 754 标准有一些复杂的细节，初次看上去难以理解，对于日常应用也不常用，本文就不介
了。
如果你想查看浮点数的具体二进制形式，在 Java 中，可以使用如下代码：
<blockquote>
Integer.toBinaryString(Float.floatToIntBits(value))
Long.toBinaryString(Double.doubleToLongBits(value));
</blockquote>
小结
小数计算为什么会出错呢？理由就是：很多小数计算机中不能精确表示。
计算机的基本思维是二进制的，所以，意料之外，情理之中!
文章转自<a href="https://ld246.com/forward?goto=http%3A%2F%2Fmp.weixin.qq.com%
Fs%3F__biz%3DMzIxOTI1NTk5Nw%3D%3D%26mid%3D402536430%26idx%3D1%26sn%3D7
3338665945d96337054a62eda86458%26scene%3D19%23wechat_redirect" target="_blank" re
="nofollow ugc">老马说编程