

浮点数在计算机中是如何表示的？

```
#include <stdio.h>
void main(void){
    int num=9; /* num是整型变量，设为9 */
    float* pFloat=# /* pFloat表示num的内存地址，但是设为浮点数 */
    printf("num的值为: %d\n",num); /* 显示num的整型值 */
    printf("*pFloat的值为: %f\n",*pFloat); /* 显示num的浮点值 */
    *pFloat=9.0; /* 将num的值改为浮点数 */
    printf("num的值为: %d\n",num); /* 显示num的整型值 */
    printf("*pFloat的值为: %f\n",*pFloat); /* 显示num的浮点值 */
}
```

运行结果如下：

num的值为： 9

*pFloat的值为： 0.000000

num的值为： 1091567616

*pFloat的值为： 9.000000

我很惊讶，num和*pFloat在内存中明明是同一个数，为什么浮点数和整数的解读结果会差别这么大？

要理解这个结果，一定要搞懂浮点数在计算机内部的表示方法。我读了一些资料，下面就是我的笔记。

在讨论浮点数之前，先看一下整数在计算机内部是怎样表示的。

```
int num=9;
```

上面这条命令，声明了一个整数变量，类型为int，值为9（二进制写法为1001）。普通的32位计算机，用4个字节表示int变量，所以9就被保存为00000000 00000000 00000000 00001001，写成16进制就是0x00000009。

那么，我们的问题就简化成：为什么0x00000009还原成浮点数，就成了0.000000？

根据国际标准IEEE 754，任意一个二进制浮点数V可以表示成下面的形式：

$$V = (-1)^s \times M \times 2^E$$

（1） $(-1)^s$ 表示符号位，当s=0，V为正数；当s=1，V为负数。

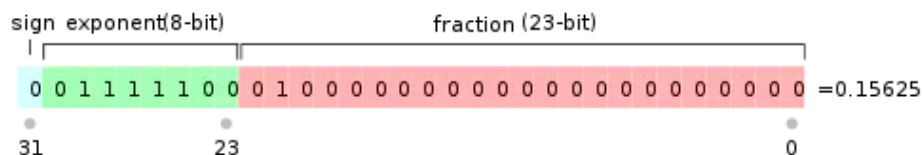
（2）M表示有效数字，大于等于1，小于2。

（3） 2^E 表示指数位。

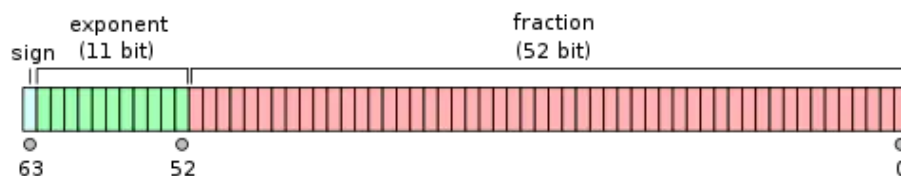
举例来说，十进制的5.0，写成二进制是101.0，相当于 1.01×2^2 。那么，按照上面V的格式，可以得出s=0，M=1.01，E=2。

十进制的-5.0，写成二进制是-101.0，相当于 -1.01×2^2 。那么，s=1，M=1.01，E=2。

IEEE 754规定，对于32位的浮点数，最高的1位是符号位s，接着的8位是指数E，剩下的23位为有效数字M。



对于64位的浮点数，最高的1位是符号位S，接着的11位是指数E，剩下的52位为有效数字M。



IEEE 754对有效数字M和指数E，还有一些特别规定。

前面说过， $1 \leq M < 2$ ，也就是说，M可以写成1.xxxxxx的形式，其中xxxxxx表示小数部分。IEEE 754规定，在计算机内部保存M时，默认这个数的第一位总是1，因此可以被舍去，只保存后面的xxxxxx部分。比如保存1.01的时候，只保存01，等到读取的时候，再把第一位的1加上。这样做的目的，是节省1位有效数字。以32位浮点数为例，留给M只有23位，将第一位的1舍去以后，等于可以保存24位有效数字。

至于指数E，情况就比较复杂。

首先，E为一个无符号整数（unsigned int）。这意味着，如果E为8位，它的取值范围为0~255；如果E为11位，它的取值范围为0~2047。但是，我们知道，科学计数法中的E是可以出现负数的，所以IEEE 754规定，E的真实值必须再减去一个中间数，对于8位的E，这个中间数是127；对于11位的E，这个中间数是1023。

比如， 2^{10} 的E是10，所以保存成32位浮点数时，必须保存成 $10+127=137$ ，即10001001。

然后，指数E还可以再分成三种情况：

(1) E不全为0或不全为1。这时，浮点数就采用上面的规则表示，即指数E的计算值减去127（或1023），得到真实值，再将有效数字M前加上第一位的1。

(2) E全为0。这时，浮点数的指数E等于1-127=126（或者1-1023），有效数字M不再加上第一位的1，而是还原为0. xxxxxx的小数。这样做是为了表示±0，以及接近于0的很小的数字。

(3) E全为1。这时，如果有效数字M全为0，表示±无穷大（正负取决于符号位s）；如果有效数字M不全为0，表示这个数不是一个数（NaN）。

好了，关于浮点数的表示规则，就说到这里。

下面，让我们回到一开始的问题：为什么0x00000009还原成浮点数，就成了0.000000？

首先，将0x00000009拆分，得到第一位符号位s=0，后面8位的指数E=00000000，最后23位的有效数字M=000 0000 0000 0000 0000 1001。

由于指数E全为0，所以符合上一节的第二种情况。因此，浮点数V就写成：

$$V = (-1)^0 \times 0.00000000000000000000001001 \times 2^{(-126)} = 1.001 \times 2^{(-146)}$$

显然，V是一个很小的接近于0的正数，所以用十进制小数表示就是0.000000。

再看例题的第二部分：请问浮点数9.0，如何用二进制表示？还原成十进制又是多少？

首先，浮点数9.0等于二进制的1001.0，即 1.001×2^3 。那么，第一位的符号位s=0，有效数字M等于001后面再加20个0，凑满23位，指数E等于3+127=130，即10000010。所以，写成二进制形式，应该是s+E+M，即0 10000010 001 0000 0000 0000 0000 0000。这个32位的二进制数，还原成十进制，正是1091567616。

