格式A		格式B		注
位	值	位	值	
011 0000	1	0111 000	1	
101 1110	$\frac{15}{2}$	1001 111	$\frac{15}{2}$	
010 1001	$\frac{25}{32}$	0110 100	$\frac{3}{4}$	向下舍入
110 1111	$\frac{31}{2}$	1011 000	16	向上舍入
000 0001	$\frac{1}{64}$	0001 000	$\frac{1}{64}$	非规格化→规格化

2.53　一般来说，使用库宏(library macro)会比你自己写的代码更好一些。不过，这段代码似乎可以在多种机器上工作。

假设值 1e400 溢出为无穷。

```
#define POS_INFINITY 1e400
#define NEG_INFINITY (-POS_INFINITY)
#define NEG_ZERO (-1.0/POS_INFINITY)
```

2.54　这个练习可以帮助你从程序员的角度来提高研究浮点运算的能力。确信自己理解下面每一个答案。

A. `x == (int)(double) x`

真，因为 double 类型比 int 类型具有更大的精度和范围。

B. `x == (int)(double) x`

假，例如当 x 为 $TMax$ 时。

C. `d == (double)(float) d`

假，例如当 d 为 1e40 时，我们在右边得到 $+\infty$。

D. `f == (float)(double) f`

真，因为 double 类型比 float 类型具有更大的精度和范围。

E. `f == -(-f)`

真，因为浮点数取非就是简单地对它的符号位取反。

F. `1.0/2 == 1/2.0`

真，在执行除法之前，分子和分母都会被转换成浮点表示。

G. `d*d>=0.0`

真，虽然它可能会溢出到 $+\infty$。

H. `(f+d)-f == d`

假，例如当 f 是 1.0e20 而 d 是 1.0 时，表达式 f+d 会舍入到 1.0e20，因此左边的表达式求值得到 0.0，而右边是 1.0。