浮点数取整

考虑 C 语言中对双精度浮点数（double d）取整的不同方法，尤其是魔数取整的原理。

强制类型转换

直接使用 int(d) 进行强制类型转换会向零截断（int(-3.7)=-3），但是超出 int 范围的转换是未定义行为，可能得到错误结果。

标准库函数

math 库提供了 round（四舍五入）、rint（四舍六入五取偶）、trunc（向零截断）等不同语义的取整函数。

在 x87（早期 x86 CPU 的浮点指令集）中，浮点取整的硬件指令 FRNDINT 默认使用“取偶”语义，但是存在以下问题：

1. 非单周期指令，延迟大；
2. 通过一个不精确标志处理浮点数中的特殊值等情况；
3. 使用浮点栈，需要在寄存器和内存间来回搬运。

导致基于 FRNDINT 实现 rint 需要生成其他指令，执行效率低。

在支持 SSE/AVX（SIMD 指令集）的现代 CPU 上，rint 会被直接编译成一条硬件指令，且速度极快，不再需要魔数取整。

魔数取整

针对 x87 上存在的问题，编译器开始用魔数取整法实现 rint，以替代缓慢的 FRNDINT 实现。即 rint(d)$= d + A - A$，其中魔数 $A = 2^{52}$。

设 double d = 1.75，在 IEEE-754 标准下，其机器数如下：

s（1位）	e（11位）	m（52位）
0	011 1111 1111	110...0

$d = (-1)^s \times 1.m \times 2^{(e-1023)}$

1.75 D = 1.11×2^0 B = $(-1)^0 \times 1.11 \times 2^{(1023-1023)}$

$$
\begin{aligned}
&1.75 + 2^{52} \\
=&1.11 \times 2^0 + 1.0 \times 2^{52} \\
=&0.0\cdots0111(\text{小数点后51个0}) \times 2^{52} + 1.0 \times 2^{52} \\
=&1.0\cdots0111(\text{小数点后51个0}) \times 2^{52} \\
=&1.0\cdots010(\text{舍入，小数点后50个0} + 10) \times 2^{52}
\end{aligned}
$$

$$
\begin{aligned}
&1.75 + 2^{52} - 2^{52} \\
=&1.0\cdots010(\text{小数点后50个}0+10)\times 2^{52} - 1.0\times 2^{52} \\
=&0.0\cdots010(\text{小数点后50个}0+10)\times 2^{52} \\
=&1.0\times 2^{1}(\text{规格化}) \\
=&2.0D
\end{aligned}
$$

-1.75、1.5 等类似，可以得到正确的结果。

不太一样地，$-1.5 + 2^{52} - 2^{52} = -1.5$，而不是预期的 -2.0。

$$
\begin{aligned}
&-1.5 + 2^{52} \\
=&-1.1\times 2^{0} + 1.0\times 2^{52} \\
=&-0.0\cdots011(\text{小数点后51个}0)\times 2^{52} + 1.0\times 2^{52} \\
=&0.1\cdots101(\text{小数点后51个}1+01)\times 2^{52} \\
=&1.1\cdots101(\text{规格化，小数点后50个}1+01)\times 2^{51}
\end{aligned}
$$

$$
\begin{aligned}
&-1.5 + 2^{52} - 2^{52} \\
=&1.1\cdots101(\text{小数点后50个}1+01)\times 2^{51} - 1.0\times 2^{52} \\
=&0.1\cdots101(\text{小数点后51个}1+01)\times 2^{52} - 1.0\times 2^{52} \\
=&-0.0\cdots011(\text{小数点后51个}0)\times 2^{52} \\
=&-1.1\times 2^{0}(\text{规格化}) \\
=&-1.5D
\end{aligned}
$$

由于双精度浮点数的尾数部分只有 52 位，当 $|d+A| > 2^{52}$ 时，浮点数 $|d+A|$ 的最低有效位单位（ULP）为 1.0，即小数部分会被舍入到临近整数，再减 A 就实现了对 d 取整。

当 $d=-1.5$ 时，$|d+A| < 2^{52}$ 其小数部分能够保留，取整失败，所以对于负数 d，正确的顺序是 $d-A+A$。为了避免判断正负的开销，可以设置魔数 $A = 2^{52} + 2^{51}$，这样对于任意 $|d| \le 2^{51}$，`rint(d)`$= d + A - A$。当 $|d| > 2^{51}$ 时，$|d+A| > 2^{53}$，ULP=2.0，此时对于所有奇数 d 取整的结果是错误的。

参考

1. https://gcc.gnu.org/legacy-ml/gcc/2004-04/msg00872.html
2. https://cgit.freebsd.org/src/tree/lib/msun/src/s_rint.c