第二章

2.1 信息存储

2.1.2 字数据大小

有符号	无符号	32位	64位
`char`	`unsigned char`	1	1
`short`	`unsigned short`	2	2
`int`	`unsigned`	4	4
`long`	`unsigned long`	4	8
`int32_t`	`uint32_t`	4	4
`int64_t`	`uin64_t`	8	8
`char *`		4	8
`float`		4	4
`double`		8	8

2.1.3 寻址和字节顺序

大端法

最高有效字节在低地址

小端法

最低有效字节在低地址

2.1.9 C语言中的移位运算

操作	值	
参数 `x`	`0110 0011`	`1001 0101`
`x << 4`	`0011 0000`	`0101 0000`
`x >> 4`（逻辑右移）	`0000 0110`	`0000 1001`
`x >> 4`（算术右移）	`0000 0110`	`1111 1001`

有符号数使用算数右移，无符号数使用逻辑右移

2.2 整数表示

2.2.1 整型数据类型

C数据类型	最小值	最大值
`char`	-128	127
`unsigned char`	0	255
`short`	-32 768	32 767
`unsigned short`	0	65 535
`int`	-2 147 483 648	2 147 483 647
`unsigned int`	0	4 294 967 295
`long`	-9 223 372 036 854 775 808	-9 223 372 036 854 775 807
`unsigned long`	0	18 446 744 073 709 551 165
`int64_t`	-9 223 372 036 854 775 808	-9 223 372 036 854 775 807
`uint64_t`	0	18 446 744 073 709 551 165

2.2.2 无符号数的编码

$$B2U_w(x) = \sum_{i=0}^{w-1} x_i 2^i$$

2.2.3 补码编码

$$B2T_w(x) = -x_{w-1}2^{w-1} + \sum_{i=0}^{w-2} x_i 2^i$$

2.2.4 有符号数和无符号数之间的转换

$$T2U_w(x) = \begin{cases} x + 2^w, & x < 0 \\ x, & x \geqslant 0 \end{cases}$$

$$U2T_w(u) = \begin{cases} u, & u \leqslant TMax_w \\ u - 2^w, & u > TMax_w \end{cases}$$

2.2.5 C语言中的有符号数与无符号数

将有符号数转换成无符号数

2.2.6 扩展一个数字的位表示

无符号数的零扩展

补码数的符号扩展

2.2.7 截断数字

截断无符号数 $x' = x \mod 2^k$

截断补码 $x' = U2T_k(x \mod 2^k)$

2.4 浮点数

2.4.2 IEEE浮点表示

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

$E = exp - (2^{w-1} - 1)$

单精度浮点数 1 + 8 + 23

双精度浮点数 1 + 11 + 52

描述	位表示	指数			小数		值		
		e	E	2^E	f	M	$2^E \times M$	V	十进制
0	0 0000 000	0	−6	$\frac{1}{64}$	$\frac{0}{8}$	$\frac{0}{8}$	$\frac{0}{512}$	0	0.0
最小的非规格化数	0 0000 001	0	−6	$\frac{1}{64}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{512}$	$\frac{1}{512}$	0.001953
	0 0000 010	0	−6	$\frac{1}{64}$	$\frac{2}{8}$	$\frac{2}{8}$	$\frac{2}{512}$	$\frac{1}{256}$	0.003906
	0 0000 011	0	−6	$\frac{1}{64}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{3}{512}$	$\frac{3}{512}$	0.005859
	⋮								
最大的非规格化数	0 0000 111	0	−6	$\frac{1}{64}$	$\frac{7}{8}$	$\frac{7}{8}$	$\frac{7}{512}$	$\frac{7}{512}$	0.013672
最小的规格化数	0 0001 000	1	−6	$\frac{1}{64}$	$\frac{0}{8}$	$\frac{8}{8}$	$\frac{8}{512}$	$\frac{1}{64}$	0.015625
	0 0001 001	1	−6	$\frac{1}{64}$	$\frac{1}{8}$	$\frac{9}{8}$	$\frac{9}{512}$	$\frac{9}{512}$	0.017578
	⋮								
	0 0110 110	6	−1	$\frac{1}{2}$	$\frac{6}{8}$	$\frac{14}{8}$	$\frac{14}{16}$	$\frac{7}{8}$	0.875
	0 0110 111	6	−1	$\frac{1}{2}$	$\frac{7}{8}$	$\frac{15}{8}$	$\frac{15}{16}$	$\frac{15}{16}$	0.9375
1	0 0111 000	7	0	1	$\frac{0}{8}$	$\frac{8}{8}$	$\frac{8}{8}$	1	1.0
	0 0111 001	7	0	1	$\frac{1}{8}$	$\frac{9}{8}$	$\frac{9}{8}$	$\frac{9}{8}$	1.125
	0 0111 010	7	0	1	$\frac{2}{8}$	$\frac{10}{8}$	$\frac{10}{8}$	$\frac{5}{4}$	1.25
	⋮								
	0 1110 110	14	7	128	$\frac{6}{8}$	$\frac{14}{8}$	$\frac{1792}{8}$	224	224.0
最大的规格化数	0 1110 111	14	7	128	$\frac{7}{8}$	$\frac{15}{8}$	$\frac{1920}{8}$	240	240.0
无穷大	0 1111 000	—	—	—	—	—	—	∞	—

图 2-35　8 位浮点格式的非负值示例($k=4$ 的阶码位的和 $n=3$ 的小数位。偏置量是 7)

2.4.6 C语言中的浮点数

当在 int、float 和 double 格式之间进行强制类型转换时，程序改变数值和位模式的原则如下(假设 int 是 32 位的)：

- 从 int 转换成 float，数字不会溢出，但是可能被舍入。
- 从 int 或 float 转换成 double，因为 double 有更大的范围(也就是可表示值的范围)，也有更高的精度(也就是有效位数)，所以能够保留精确的数值。
- 从 double 转换成 float，因为范围要小一些，所以值可能溢出成$+\infty$或$-\infty$。另外，由于精确度较小，它还可能被舍入。
- 从 float 或者 double 转换成 int，值将会向零舍入。例如，1.999 将被转换成 1，而−1.999 将被转换成−1。进一步来说，值可能会溢出。C 语言标准没有对这种情况指定固定的结果。与 Intel 兼容的微处理器指定位模式[10…00](字长为 w 时的 $TMin_w$)为整数不确定(integer indefinite)值。一个从浮点数到整数的转换，如果不能为该浮点数找到一个合理的整数近似值，就会产生这样一个值。因此，表达式 (int)+1e10 会得到-21483648，即从一个正值变成了一个负值。