

程序员也都假设机器会使用这种右移。另一方面，对于无符号数，右移必须是逻辑的。

与 C 相比，Java 对于如何进行右移有明确的定义。表达是 $x \gg k$ 会将 x 算术右移 k 个位置，而 $x \ggg k$ 会对 x 做逻辑右移。

旁注 移动 k 位，这里 k 很大

对于一个由 w 位组成的数据类型，如果要移动 $k \geq w$ 位会得到什么结果呢？例如，计算下面的表达式会得到什么结果，假设数据类型 `int` 为 $w=32$ ：

```
int    lval = 0xFEDCBA98 << 32;
int    aval = 0xFEDCBA98 >> 36;
unsigned uval = 0xFEDCBA98u >> 40;
```

C 语言标准很小心地规避了说明在这种情况下该如何做。在许多机器上，当移动一个 w 位的值时，移位指令只考虑位移量的低 $\log_2 w$ 位，因此实际上位移量就是通过计算 $k \bmod w$ 得到的。例如，当 $w=32$ 时，上面三个移位运算分别是移动 0、4 和 8 位，得到结果：

```
lval    0xFEDCBA98
aval    0xFFEDCBA9
uval    0x00FEDCBA
```


不过这种行为对于 C 程序来说是没有保证的，所以应该保持位移量小于待移位值的位数。

另一方面，Java 特别要求位移数量应该按照我们前面所讲的求模的方法来计算。

旁注 与移位运算有关的操作符优先级问题

常常有人会写这样的表达式 $1 \ll 2 + 3 \ll 4$ ，本意是 $(1 \ll 2) + (3 \ll 4)$ 。但是在 C 语言中，前面的表达式等价于 $1 \ll (2 + 3) \ll 4$ ，这是由于加法(和减法)的优先级比移位运算要高。然后，按照从左至右结合性规则，括号应该是这样打的 $(1 \ll (2 + 3)) \ll 4$ ，得到的结果是 512，而不是期望的 52。

在 C 表达式中搞错优先级是一种常见的程序错误原因，而且常常很难检查出来。所以当你拿不准的时候，请加上括号！

 **练习 2.16** 填写下表，展示不同移位运算对单字节数的影响。思考移位运算的最好方式是使用二进制表示。将最初的值转换为二进制，执行移位运算，然后再转换回十六进制。每个答案都应该是 8 个二进制数字或者 2 个十六进制数字。

x		$x \ll 3$		$x \gg 2$ (逻辑的)		$x \gg 2$ (算术的)	
十六进制	二进制	二进制	十六进制	二进制	十六进制	二进制	十六进制
0xC3							
0x75							
0x87							
0x66							

2.2 整数表示

在本节中，我们描述用位来编码整数的两种不同的方式：一种只能表示非负数，而另一种能够表示负数、零和正数。后面我们将会看到它们在数学属性和机器级实现方面密切相关。我们还会研究扩展或者收缩一个已编码整数以适应不同长度表示的效果。

图 2-8 列出了我们引入的数学术语，用于精确定义和描述计算机如何编码和操作整数。这些术语将在描述的过程中介绍，图在此处列出作为参考。