我们已经看到了许多无符号运算的细微特性，尤其是有符号数到无符号数的隐式转换，会导致错误或者漏洞的方式。避免这类错误的一种方法就是绝不使用无符号数。实际上，除了C以外很少有语言支持无符号整数。很明显，这些语言的设计者认为它们带来的麻烦要比益处多得多。比如，Java只支持有符号整数，并且要求以补码运算来实现。正常的右移运算符>>被定义为执行算术右移。特殊的运算符>>>被指定为执行逻辑右移。

当我们想要把字仅仅看做是位的集合而没有任何数字意义时，无符号数值是非常有用的。例如，往一个字中放入描述各种布尔条件的标记(flag)时，就是这样。地址自然地就是无符号的，所以系统程序员发现无符号类型是很有帮助的。当实现模运算和多精度运算的数学包时，数字是由字的数组来表示的，无符号值也会非常有用。

2.3　整数运算

许多刚入门的程序员非常惊奇地发现，两个正数相加会得出一个负数，而比较表达式x<y和比较表达式x-y<0会产生不同的结果。这些属性是由于计算机运算的有限性造成的。理解计算机运算的细微之处能够帮助程序员编写更可靠的代码。

2.3.1　无符号加法

考虑两个非负整数x和y，满足$0 \leqslant x$，$y < 2^w$。每个数都能表示为w位无符号数字。然而，如果计算它们的和，我们就有一个可能的范围$0 \leqslant x+y \leqslant 2^{w+1}-2$。表示这个和可能需要$w+1$位。例如，图2-21展示了当$x$和$y$有4位表示时，函数$x+y$的坐标图。参数(显示在水平轴上)取值范围为0～15，但是和的取值范围为0～30。函数的形状是一个有坡度的平面(在两个维度上，函数都是线性的)。如果保持和为一个$w+1$位的数字，并且把它加上另外一个数值，我们可能需要$w+2$个位，以此类推。这种持续的“字长膨胀”意味着，要想完整地表示算术运算的结果，我们不能对字长做任何限制。一些编程语言，例如Lisp，实际上就支持无限精度的运算，允许任意的(当然，要在机器的内存限制之内)整数运算。更常见的是，编程语言支持固定精度的运算，因此像“加法”和“乘法”这样的运算不同于它们在整数上的相应运算。

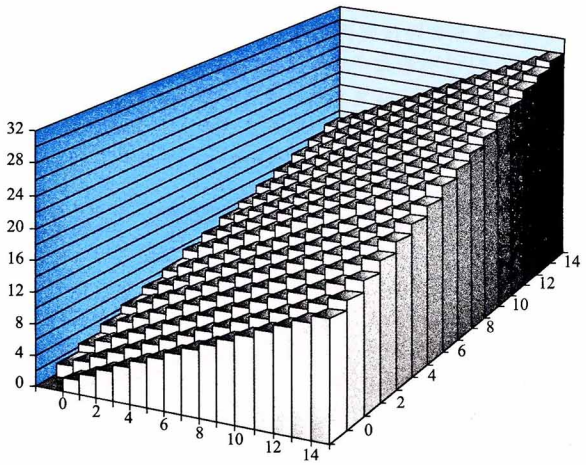

图2-21　整数加法。对于一个4位的字长，其和可能需要5位