Bitwise Complement, *And*, Exclusive *Or*, and Inclusive *Or*

Table 20.2 lists the remaining bitwise operators.

Table 20.2
Other Bitwise Operators

Symbol	*Meaning*
~	bitwise complement
&	bitwise *and*
^	bitwise exclusive *or*
|	bitwise inclusive *or*

The ~ operator is unary; the integer promotions are performed on its operand. The other operators are binary; the usual arithmetic conversions are performed on their operands.

The ~, &, ^, and | operators perform Boolean operations on all bits in their operands. The ~ operator produces the complement of its operand, with zeros replaced by ones and ones replaced by zeros. The & operator performs a Boolean *and* operation on all corresponding bits in its two operands. The ^ and | operators are similar (both perform a Boolean *or* operation on the bits in their operands); however, ^ produces 0 whenever both operands have a 1 bit, whereas | produces 1.

Don't confuse the *bitwise* operators & and | with the *logical* operators && and ||. The bitwise operators sometimes produce the same results as the logical operators, but they're not equivalent.

The following examples illustrate the effect of the ~, &, ^, and | operators:

```
unsigned short i, j, k;

i = 21;       /* i is now    21 (binary 0000000000010101) */
j = 56;       /* j is now    56 (binary 0000000000111000) */
k = ~i;       /* k is now 65514 (binary 1111111111101010) */
k = i & j;    /* k is now    16 (binary 0000000000010000) */
k = i ^ j;    /* k is now    45 (binary 0000000000101101) */
k = i | j;    /* k is now    61 (binary 0000000000111101) */
```

The value shown for `~i` is based on the assumption that an `unsigned short` value occupies 16 bits.

The ~ operator deserves special mention, since we can use it to help make even low-level programs more portable. Suppose that we need an integer whose bits are all 1. The preferred technique is to write `~0`, which doesn't depend on the number of bits in an integer. Similarly, if we need an integer whose bits are all 1 except for the last five, we could write `~0x1f`.