Bitwise Shift Operators

The bitwise shift operators can transform the binary representation of an integer by shifting its bits to the left or right. C provides two shift operators, which are shown in Table 20.1.

Table 20.1
Bitwise Shift Operators

Symbol	*Meaning*
<<	left shift
>>	right shift

The operands for << and >> may be of any integer type (including `char`). The integer promotions are performed on both operands: the result has the type of the left operand after promotion.

The value of `i << j` is the result when the bits in `i` are shifted left by `j` places. For each bit that is "shifted off" the left end of `i`, a zero bit enters at the right. The value of `i >> j` is the result when `i` is shifted right by `j` places. If `i` is of an unsigned type or if the value of `i` is nonnegative, zeros are added at the left as needed. If `i` is a negative number, the result is implementation-defined: some implementations add zeros at the left end, while others preserve the sign bit by adding ones.

portability tip *For portability, it's best to perform shifts only on unsigned numbers.*

The following examples illustrate the effect of applying the shift operators to the number 13. (For simplicity, these examples—and others in this section—use short integers, which are typically 16 bits.)

```
unsigned short i, j;

i = 13;         /* i is now 13 (binary 0000000000001101) */
j = i << 2;     /* j is now 52 (binary 0000000000110100) */
j = i >> 2;     /* j is now  3 (binary 0000000000000011) */
```

As these examples show, neither operator modifies its operands. To modify a variable by shifting its bits, we'd use the compound assignment operators <<= and >>=:

```
i = 13;         /* i is now 13 (binary 0000000000001101) */
i <<= 2;        /* i is now 52 (binary 0000000000110100) */
i >>= 2;        /* i is now 13 (binary 0000000000001101) */
```

⚠ The bitwise shift operators have lower precedence than the arithmetic operators, which can cause surprises. For example, `i << 2 + 1` means `i << (2 + 1)`, not `(i << 2) + 1`.
