Bit fields typically are used in low-level programming. Often, using an integral type and the bitwise operators of Appendix E, "Other Operators," provides an alternative approach.

**Unions**

A *union* is a data format that can hold different data types but only one type at a time. That is, whereas a structure can hold, say, an *int* and a *long* and a *double*, a union can hold an *int* or a *long* or a *double*. The syntax is like that for a structure, but the meaning is different. For example, consider the following declaration:

```c
union one4all
{
    int int_val;
    long long_val;
    double double_val;
};
```

You can use a *one4all* variable to hold an *int*, a *long*, or a *double*, just as long as you do so at different times:

```c
one4all pail;
pail.int_val = 15;        // store an int
cout << pail.int_val;
pail.double_val = 1.38;   // store a double, int value is lost
cout << pail.double_val;
```

Thus, *pail* can serve as an *int* variable on one occasion and as a *double* variable at another time. The member name identifies the capacity in which the variable is acting. Because a union only holds one value at a time, it has to have space enough to hold its largest member. Hence, the size of the union is the size of its largest member.

One use for the union is to save space when a data item can use two or more formats but