depending on whether input succeeded or not. In this loop, for example, \texttt{cin} expects the user to enter two numbers. If, instead, you enter \texttt{q}, as we did, \texttt{cin \textgreater>q} recognizes that \texttt{q} is not a number. It leaves the \texttt{q} in the input queue and returns a value that's converted to \texttt{false}, terminating the loop.

Compare this approach for reading numbers to this simpler one:

\begin{verbatim}
for (int i = 0; i < limit; i++)
{
    cout << "Enter value #" << (i + 1) << ": ";
    cin >> temp;
    if (temp < 0)
        break;
    ar[i] = temp;
}
\end{verbatim}

To terminate that loop early, you enter a negative number. That restricts input to non-negative values. That restriction fits the needs of some programs, but more typically you would want a means of terminating a loop that didn't exclude certain numeric values. Using \texttt{cin \textgreater>q} as the test condition eliminates such restrictions, for it accepts all valid numeric input. Keep this trick in mind when you need an input loop for numbers. Also, keep in mind that non-numeric input sets an error condition that prevents the reading of any more input. If your program needs input subsequent to the input loop, you must use \texttt{cin.clear()} to reset input, and you might then need to get rid of the offending input by reading it. \texttt{Listing 7.7} illustrated those techniques.

**Passing Structure Addresses**

Suppose you want to save time and space by passing the address of a structure instead of passing the entire structure. This requires rewriting the functions so that they use pointers to structures. First, let's see how to rewrite the \texttt{show_polar()} function. You need to make three changes:

- When calling the function, pass it the address of the structure (\texttt{&pplace}) rather than the structure itself (\texttt{pplace}).
- Declare the formal parameter to be a pointer-to-\texttt{polar}, that is, type \texttt{polar *}. 