```
      INTEGER FUNCTION READCH(CHAR)
      INTEGER LINE(81), NEXTCH, CHAR, YES, NO
      DATA NEXTCH /82/, LINE(81) /' '/, YES /1/, NO /0/
C
      IF (NEXTCH .LE. 81) GOTO 20
         READ(5,11,END=90) (LINE(I), I=1,80)
   11       FORMAT(80A1)
         NEXTCH = 1
   20 CHAR = LINE(NEXTCH)
      NEXTCH = NEXTCH + 1
      READCH = YES
      RETURN
C END OF FILE
   90 READCH = NO
      RETURN
      END
```

The construction END=90 in the READ statement causes a branch to statement 90 when end of file occurs. This feature is well on its way to becoming a standard. (It is part of Fortran 77.) We have also assumed that the end of a card should mark the end of a word. To ensure this, READCH returns a blank after the end of each card. (Notice that this is done with a well-chosen data structure: there is a dummy 81st column on the card which is always blank. READCH fetches a new card only after this blank has been returned.)

There are some pragmatic advantages to having a separate function for input, many of which we discussed in Chapter 4. Most important is simply breaking a big job into smaller, non-interacting pieces. Furthermore, I/O is often the most system-dependent part of a program; when a program has to be moved or changed, it's *much* better to have all input and output in one place than scattered randomly throughout a large program. As another benefit, consider how easy it is to implement centralized functions like stripping off trailing blanks or performing character set translations.

Localize input and output in subroutines.

Input/output is the interface between a program and its environment. Two rules govern all I/O programming: NEVER TRUST ANY DATA, and REMEMBER THE USER. This requires that a program be as foolproof as is reasonably possible, so that it behaves intelligently even when used incorrectly, and that it be easy to use correctly. Ask yourself: Will it defend itself against the stupidity and ignorance of its users (including myself)? Would I want to have to use it myself?

To summarize the major principles discussed in this chapter:

(1) Check input data for validity and plausibility.

(2) Make sure that data does not violate limitations of the program.