a second subroutine, to modularize the code. If we do not fret over making a few "unnecessary" tests, we can unsnarl the tangle of branches with a general "test and store" subroutine, which decides whether one specified direction represents a legal move, and saves it. All that is left then is the constraint on what men may make certain moves, which can be simply encoded:

```
      SUBROUTINE SEARCH(BOARD, I, J, ROW, COL, L)
      INTEGER BOARD(8,8), ROW(4), COL(4)
C               BOARD(I,J)=0 => EMPTY SQUARE
C                          1 => WHITE MAN, 2 => WHITE KING
C                          3 => BLACK MAN, 4 => BLACK KING
C               I INCREASES FORWARD, J INCREASES RIGHT
C               L COUNTS THE MOVES STORED IN ROW,COL
      IF (I.LT.1 .OR. I.GT.8 .OR. J.LT.1 .OR. J.GT.8)
     $   CALL ERROR(9, 'OFF BOARD')
      K = BOARD(I,J)
      IF (K.LT.1 .OR. K.GT.4) CALL ERROR(20, 'ILLEGAL MAN ON BOARD')
      L = 0
      IF (K .NE. 3) CALL STORE(BOARD, I+1, J+1, ROW, COL, L)
      IF (K .NE. 3) CALL STORE(BOARD, I+1, J-1, ROW, COL, L)
      IF (K .NE. 1) CALL STORE(BOARD, I-1, J+1, ROW, COL, L)
      IF (K .NE. 1) CALL STORE(BOARD, I-1, J-1, ROW, COL, L)
      RETURN
      END

      SUBROUTINE STORE(BOARD, IC, JC, ROW, COL, L)
      INTEGER BOARD(8,8), ROW(4), COL(4)
      IF (IC.LT.1 .OR. IC.GT.8 .OR. JC.LT.1 .OR. JC.GT.8) RETURN
      IF (BOARD(IC,JC) .NE. 0) RETURN
      L = L+1
      ROW(L) = IC
      COL(L) = JC
      RETURN
      END
```

Separating code into appropriate modules is an important aspect of writing a program. As we can see here, the subroutine call permits us to summarize the *irregularities* in the argument list, where we can see quickly what is going on. The subroutine itself summarizes the *regularities* of the code, so repeated patterns need not be used. An added advantage of this version is that including complexities like jumps later on will be rather easy. "Optimizing" too early in the life of a program can kill its chances for growth.

We have included calls to an unspecified error-printing routine, to simplify the handling of illegal inputs. Such a subroutine can overcome the inertia normally felt when it comes time to check for possible errors. Some people feel that the calls to ERROR should be inserted "until the program is debugged," then removed. Leave them in indefinitely — the insurance is cheap.

Modularize. Use subroutines.
