```
TRY: PROCEDURE(I1, J1, I2, J2) RECURSIVE RETURNS (BIT(1));
   DECLARE (I1, J1, I2, J2) FIXED BINARY;

   IF MAZE(I1, J1) = WALL THEN
      RETURN(NO);
   IF MAZE(I2, J2) = WALL | STATE(I2, J2) = USED THEN
      RETURN(NO);
   STATE(I1, J1) = USED;
   PATHPTR = PATHPTR + 1;
   IPATH(PATHPTR) = I1;
   JPATH(PATHPTR) = J1;
   IF I2 = 1 | I2 = M | J2 = 1 | J2 = N THEN DO;
      STATE(I2, J2) = USED;
      PATHPTR = PATHPTR + 1;
      IPATH(PATHPTR) = I2;
      JPATH(PATHPTR) = J2;
      RETURN(YES);
   END;
   IF TRY(I2, J2, I2, J2-1) THEN
      RETURN(YES);
   IF TRY(I2, J2, I2+1, J2) THEN
      RETURN(YES);
   IF TRY(I2, J2, I2, J2+1) THEN
      RETURN(YES);
   IF TRY(I2, J2, I2-1, J2) THEN
      RETURN(YES);
   PATHPTR = PATHPTR - 1;
   RETURN(NO);
END TRY;

PRINTPATH: PROCEDURE;
   STATE(*, *) = 'X';
   DO I = 1 TO PATHPTR;
      STATE(IPATH(I), JPATH(I)) = ' ';
   END;
   PUT SKIP(2) EDIT (((STATE(I,J) DO J = 1 TO N) DO I = 1 TO M))
      (COLUMN(1), (N)A(1));
END PRINTPATH;

END MOUSE;
```

There is no claim that we went directly from original conception to final working PL/I without a single slip. But it is true that most of our mistakes were made with pseudo-code, and corrected long before the program made it into PL/I, let alone onto a machine. At each stage of the process, it was easy to analyze, test, and revise, because the program structure was clearly expressed as a handful of routines and a few lines of code, not a hundred lines of PL/I.

The maze program is pretty big; it takes effort to analyze. Yet our version has fewer statements than the original, and is far easier to understand. This is not because we have better comments, nor is it because our identifiers are more meaningful. The main difference is structural.

We chose our control structures on the basis of legibility; people tend to understand th:m with little effort. Our version of the maze program has only one label besides procedure names. That in itself is no great accomplishment, but it indicates that the flow of control must be essentially from top to bottom.