just assigned to it. But in PL/I this is not necessarily so. In the assignment any fractional part is discarded, but in the comparison it is retained. The code thus tests whether IYEAR is divisible by n, and branches if it is. Try to explain that to a novice programmer! Such unobvious code should surely be replaced by

```
IF MOD(IYEAR,n) = 0 THEN GOTO label;
```

This still may not be obvious to a rank beginner, but it is unambiguous and easily learned.

Calendar computations are notoriously complex, but the approach shown above makes them seem even worse. The rococo structure of the calendar is intimately intertwined with the control flow in an attempt to arrive at the proper answer with a minimum number of tests.

Clarity is certainly not worth sacrificing just to save three tests per access (on the average) — the irregularities must be brought under control. In the next chapter we will discuss using subroutines to achieve regularity: the procedure body shows what is common to each invocation, and the differences are concisely summarized in the parameter list for each call. In a similar way, the irregularities in a computation can often be captured in well-chosen data structures. All of the information about how many days precede the first of each month can be put in a table instead of being strung throughout the code. Then a more organized approach is possible. Accordingly, in the first edition of this book, we wrote

```
DATES: PROCEDURE OPTIONS (MAIN);
   DECLARE NDAYS(0:1,0:12) INITIAL(
      0,31,59,90,120,151,181,212,243,273,304,334,365,  /* NON-LEAP */
      0,31,60,91,121,152,182,213,244,274,305,335,366); /* LEAP */

   DO WHILE('1'B);
      GET LIST (IYEAR, IDATE) COPY;

      IF MOD(IYEAR,400)=0 |
         (MOD(IYEAR,100)¬=0 & MOD(IYEAR,4)=0)
            THEN LEAP = 1;
            ELSE LEAP = 0;

      IF IYEAR<1753 | IYEAR>3999 | IDATE<=0 | IDATE>NDAYS(LEAP,12)
         THEN PUT SKIP LIST('BAD YEAR, DATE -', IYEAR, IDATE);

      ELSE DO;
         DO MO = 1 TO 12 WHILE (IDATE>NDAYS(LEAP,MO));
         END;

         PUT SKIP LIST(MO, IDATE-NDAYS(LEAP,MO-1), IYEAR);
      END;
   END;
END DATES;
```

Even though this code is a single procedure, internally it decomposes into several almost independent pieces. A date is input, LEAP is computed, the date is validated (with excessive zeal), the conversion is made and the result is printed. Each of these pieces could be picked up as a unit and planted as needed in some other environment with a good chance of working unaltered, because there are no unnecessary labels or other cross references between pieces. The control flow