```
Y = 0;
L = 1;
DO FRM = 1 TO 10;
   IF X(L) = 10 THEN DO;              /* STRIKE */
      Y = Y + 10 + X(L+1) + X(L+2);
      L = L + 1;
   END;
   ELSE IF X(L) + X(L+1) = 10 THEN DO;    /* SPARE */
      Y = Y + 10 + X(L+2);
      L = L + 2;
   END;
   ELSE DO;                    /* REGULAR */
      Y = Y + X(L) + X(L+1);
      L = L + 2;
   END;
END;
RETURN(Y);
```

(There is another version of this program in Chapter 8.)

As a fringe benefit, the RETURN statement now occurs at the end, where one would normally expect it, instead of being buried inside. It is a good rule of thumb that a program should read from top to bottom in the order that it will be executed; if this is not true, watch out for the bugs that often accompany poor structure.

Make your programs read from top to bottom.

The code we used to express the three-way decision in the bowling program is an example of an important control construction, the multi-way decision, sometimes called a CASE statement. Some languages provide a separate statement for writing such branches; in PL/I, multi-way decisions are usually best expressed as a chain of IF ... ELSE IF ... ELSE, like this:

```
IF condition-1 THEN
  statement-1
ELSE IF condition-2 THEN
  statement-2
...
ELSE IF condition-n THEN
  statement-n
ELSE
  default-statement
```

The *condition*'s are read from top to bottom; at the first *condition* that is satisfied, the *statement* that follows is executed, and then the entire construct is exited. The *statement* parts may be single statements, or (as above) a group of statements enclosed in DO-END. The last ELSE handles the "default" situation, i.e., where none of the other alternatives was chosen. This trailing ELSE part may be omitted if the program logic requires no action for the default, although leaving it in with an error message may help to catch "impossible" conditions.

We will return in a moment to how to handle the CASE statement in Fortran.