Note that we align all of the ELSE's in a CASE, rather than lining up each ELSE with the corresponding IF. This emphasizes that all the legs of the CASE have equal status and keeps them from marching off the right side of the page.

Use IF ... ELSE IF ... ELSE IF ... ELSE ...
to implement multi-way branches.

The CASE statement is often recognizable in a sequence of related decisions, where only minor rearrangement is needed to bring things into the proper form:

```
IF AMT_OF_SALES <= 50.00 THEN COMM = 00.00;
IF AMT_OF_SALES > 50.00 THEN IF AMT_OF_SALES <= 100.00
   THEN COMM = .02 * AMT_OF_SALES;
IF AMT_OF_SALES > 100.00 THEN
   COMM = .03 * AMT_OF_SALES;
```

The two IF's in the second line can certainly be compressed into a single test with the logical operator &. So an improved version might read

```
IF AMT_OF_SALES <= 50.00 THEN
   COMM = 00.00;
IF AMT_OF_SALES > 50.00 & AMT_OF_SALES <= 100.00 THEN
   COMM = 0.02 * AMT_OF_SALES;
IF AMT_OF_SALES > 100.00 THEN
   COMM = 0.03 * AMT_OF_SALES;
```

But the tests in an ELSE-IF chain are done in the prescribed order, and this fact may be used to advantage. If it fails the first test, AMT_OF_SALES is greater than 50; if it fails the second, it is greater than 100. Neither test need be repeated if an ELSE-IF is used:

```
IF AMT_OF_SALES <= 50.00 THEN
   COMM = 00.00;
ELSE IF AMT_OF_SALES <= 100.00 THEN
   COMM = 0.02 * AMT_OF_SALES;
ELSE
   COMM = 0.03 * AMT_OF_SALES;
```

Peeling off cases in numerical order in this way is also highly readable and easy to change.

We have now mentioned several control flow constructions:

statement grouping with, for example, DO-END or BEGIN-END;

decision making with IF-ELSE;

looping with DO and DO-WHILE;

subroutines, functions, or procedures.

The DO loop comes in at least two flavors in PL/I, indexed and DO-WHILE, and the IF-ELSE can be extended into the CASE or multi-way decision.