
Avoid the Fortran arithmetic IF.

The influence of the arithmetic IF often extends into misuse of the logical IF. For example,

```
IF((X(I) - X(N)) .LE. 0.) GO TO 300
```

is a literal translation of an arithmetic into a logical IF, which should be written

```
IF (X(I) .LE. X(N)) GOTO 300
```

("Say what you mean.") And

```
      IF (MOD(K,N1).NE.0) GO TO 9
      WRITE (6,4) K,X
    9 ...
```

is better rendered as

```
IF (MOD(K,N1) .EQ. 0) WRITE (6,4) K, X
```

The same observation holds for PL/I:

```
GROSSPAY = BASERATE *
  TOTALHRS;
IF TOTALHRS <= 40 THEN GO TO
  NOVT;
GROSSPAY = GROSSPAY + 0.5 *
  BASERATE * (TOTALHRS - 40);
NOVT: ...
```

Since the GOTO branches around only a single statement, it is clearly unnecessary. Rewriting gives

```
GROSSPAY = BASERATE * TOTALHRS;
IF TOTALHRS > 40 THEN
   GROSSPAY = GROSSPAY + 0.5 * BASERATE * (TOTALHRS-40);
```

A conditional expression can also be disguised by using a Fortran computed GOTO:

```
      GOTO(65,70),PRNT
   65 WRITE(6,105) X
   70 ...
```

The computed GOTO has a definite place, but this is not it. Since labels 65 and 70 appear nowhere else in the program, this code is certainly better written as

```
IF (PRNT .NE. 2) WRITE (6,105) X
```

to eliminate the two statement numbers. Now we can tell at a glance that there is only one way to reach the WRITE statement.

These last three examples show a tendency to follow all IF's with branches, even when they do not have to be. Such usage eventually leads to circumlocutions like