table value if it finds a matching row *and* column; otherwise it returns zero.

```
      FUNCTION SPARSE(I,J)
      COMMON /SP/ N,NROW(500),NCOL(500),VALUE(500)
      DO 10 K = 1,N
      IF (NROW(K).NE.I .AND. NCOL(K).NE.J) GO TO 10
      SPARSE = VALUE(K)
      GO TO 999
   10 CONTINUE
      SPARSE = 0.0
  999 RETURN
      END
```

By definition the sparse array has a value stored at `VALUE(K)` for `(I,J)` if

```
NROW(K).EQ.I .AND. NCOL(K).EQ.J
```

Negating this for the `IF` statement gives

```
.NOT.(NROW(K).EQ.I) .OR. .NOT.(NCOL(K).EQ.J)
```

which in turn is

```
NROW(K).NE.I .OR. NCOL(K).NE.J
```

Compare this line with the `IF` statement in the function. The function **SPARSE** is wrong; it will return the first value where *either* `I` or `J` is matched. The **.AND.** must be changed to `.OR.`. (This error has been corrected in later printings of the text from which it was taken.) Actually, the code would be more direct if it were written with the test stated the way a human reader would say it:

```
      FUNCTION SPARSE(I, J)
      COMMON /SP/ N, NROW(500), NCOL(500), VALUE(500)
      DO 10 K = 1,N
         IF (NROW(K).EQ.I .AND. NCOL(K).EQ.J) GOTO 20
   10 CONTINUE
      SPARSE = 0.0
      RETURN
C
   20 SPARSE = VALUE(K)
      RETURN
      END
```

We have discussed a number of small examples where expressions were either hard to read, misleading, or downright incorrect. Let us conclude this chapter with a larger example, to show how quickly a program can get out of hand when you fail to look after the little things. (This is the first big PL/I program we have looked at — don't let it frighten you.) The program finds the area under the parabola $y=x^2$ between $x=0$ and $x=1$, using a trapezoidal rule, for several different step sizes.