CHAPTER 6: COMMON BLUNDERS

A major concern of programming is making sure that a program can defend against bad data. But even with correct data, there is no guarantee that a program will work. In this chapter we will discuss other aspects of making software reliable.

We begin, appropriately enough, with initialization, for failing to set a variable to some value before using it is a fruitful source of error. For example:

```
      DOUBLE PRECISION FUNCTION SIN(X,E)
C     THIS DECLARATION COMPUTES SIN(X)TO ACCURACY E
      DOUBLE PRECISION E,TERM,SUM
      REAL X
      TERM=X
      DO 20 I=3,100,2
      TERM=TERM*X**2/(I*(I-1))
      IF(TERM.LT.E)GO TO 30
      SUM=SUM+(-1**(I/2))*TERM
   20 CONTINUE
   30 SIN=SUM
      RETURN
      END
```

This program is a straightforward implementation of the Maclaurin series

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

Although large values of x will cause truncation errors long before convergence, it will work for small values of x.

At least it should, if properly programmed. But what is the value of SUM when it is first referenced inside the loop? A search shows that SUM has never been set to anything, so it begins as garbage and in most systems accumulates more garbage with each successive call. This oversight is readily corrected, *if it is detected.* How do we find it? We might run some sample cases and compare them with a table or with another sine routine. (The latter is better because it is faster and less prone to error.) The important thing, however, is to check, for a casual look at the output may not always reveal that something is amiss.

Make sure all variables are initialized before use.