generates code, and so uses only those expressions "known" to be "better." For instance

```
      10 F1=X1-X2*X2
         F2=1.0-X2
         FX=F1*F1+F2*F2
   C NOTE THAT IT IS MORE EFFICIENT TO COMPUTE
   C F1*F1 THAN TO COMPUTE F1**2.
```

Whether "efficient" means "takes less time" or "takes fewer machine instructions," the comment is not always true. Many compilers recognize the special case `F1**2` and generate the same code as for `F1*F1`. Some compilers would, in fact, generate shorter and faster code for

```
10 FX = (X1 - X2**2)**2 + (1.0 - X2)**2
```

than for the original version. (Ours produced 15 instructions for the original version, 13 for the revision.)

This rendition also happens to be more readable and eliminates the temporary variables `F1` and `F2`, which have little mnemonic value. The fewer temporary variables in a program, the less chance there is that one will not be properly initialized, or that one will be altered unexpectedly before it is used. "Temporary" is a dirty word in programming — it suggests that a variable can be used with less thought than a "normal" (permanent?) one, and it encourages the use of one variable for several unrelated calculations. Both are dangerous practices.

Avoid temporary variables.

Even if the comment about efficiency were true in a particular environment, there is still little justification for using the more obscure mode of expression. We shall discuss the question of efficiency further in Chapter 7. For now, we observe simply that a program usually has to be read several times in the process of getting it debugged. The harder it is for *people* to grasp the intent of any given section, the longer it will be before the program becomes operational. Trying to outsmart a compiler defeats much of the purpose of using one.

Write clearly — don't sacrifice clarity for "efficiency."

A variation of this is

```
/*   NOTE THAT '110010' IN BINARY IS '50' IN DECIMAL     */
/*        THIS WILL BE USED FOR LINE COUNTING            */
 ...
        IF NO>101111B THEN DO ; PUT PAGE; NO=0B;
                           END;
```

The programmer evidently hopes to avoid a run-time type-conversion by using `FIXED BINARY` constants in expressions involving `FIXED BINARY` variables. The