STYLE COUNSEL

Documenting a program involves much more than adding comments to it or writing user instructions. A well-documented program has sufficient information to indicate what it is meant to be doing, and how it is meant to be doing it. We show how a simple program, in BASIC and PASCAL, can be given suitable documentation.

Consider the first version of our program (Listing 1). It is clearly a great mystery; what it does is anyone's guess. Apart from saying that it 'inputs two numbers, multiplies them with two other numbers, adds the two results together and prints the answer', there is very little evidence of the precise task that the code performs. Now look at the second version of the program (Listing 2). All is revealed. Yet no comments have been added, no program titles or REMark lines inserted and no external documents have been produced.

It is worth taking a detailed look at the differences between these two versions. First of all, the meaningless numbers of the first listing have been replaced by names (AYEAR and AMONTH). Numbers whose values do not change while the program is running are called *constants*. Some languages, such as PASCAL, have a special notation for constants (in Listing 2, the two constants are defined separately from the variables), while other languages, like BASIC, do not. (Lines 10 and 20 of the BASIC program use variables to define the constants.) Giving names to constants is really only worthwhile if they are going to be used frequently, otherwise comments in the program would serve the purpose just as well.

The second crucial difference is that all the confusing variable names have been given longer, intelligible names. The ones that we chose here (NYEARS to replace A, ageinsecs instead of e, and so on) were picked because they are each less than 10 characters long, and the first two characters distinguish them from each other. The reason for this last requirement is explained shortly.

Generally, it is good practice to give your variables names that are related to the role they play in the program. For example, you could call a loop counter LOOP (instead of the usual J or I), and the first and last values of the counter could be put into constants or variables with appropriate names. Thus, a loop reading like this:

```
FOR J = 1 TO 10 ... NEXT J
```

could look like this:

```
FOR LOOP = FIRST TO TENTH ... NEXT LOOP
```

Long variable names do, of course, take longer to type in and use up more memory, but they do have the advantage of making programs easier to understand and speed up the debugging process. If your language uses only the first two characters of a name to distinguish between them, make sure that the names you choose differ in the first two characters. Thus, two long variable names (say CODENO and COMP) may look different to the programmer, but be indistinguishable to the computer.

Another major difference between the listings is that the second uses long and meaningful prompts for its input and adds a sensible explanation to its output (the PRINT lines in BASIC, the write lines in PASCAL). This achieves two very important things. The first is that it makes the program more readable. Even if the variables were single letters, the program would still make a lot more sense than previously. The second, and more important, benefit is that it makes the program accessible, even to someone who has never seen it before.

PROGRAM LAYOUT

PASCAL users will already be aware of the advantages of laying out a program neatly on the screen. Very simple features — like indenting lines, leaving blank lines and using a mixture of upper and lower case — can turn an impenetrable mass of symbols into a tidy and legible piece of logic. Formatting a program for the screen or the printer really comes into its own when your programs use loop constructs (FOR...NEXT, WHILE...WEND, REPEAT ...UNTIL) and especially when loops are nested inside other loops.

Having said this, it is a lamentable fact that most BASICS give very little option about how you lay out the program. In this respect, the compiled languages like PASCAL are far more flexible in that they are usually written with a text editor (or word processor). On the other hand, editing a BASIC program is generally a rather crude affair (unless, like Microsoft's MBASIC, your interpreter will take an ASCII version of the program and 'tokenise' it to turn it into a runnable program). Worse still, many BASICS will take the programs you write and reformat them to remove indentation! Some, on the other hand, will add indentation for you. The BBC Micro is quite good at this, but you have to remember to give it the LISTO command. Most PASCAL systems will include a formatter and they are generally very useful. However, for the sake of your own clarity of thought, it is a good idea to devise some formatting conventions, within the limits of your language.

Comments are, of course, the main way of documenting your programs within the programs themselves. Again conventions vary from