Directory Enquiry

Using all the techniques of Basic programming that we have learnt so far, we now take the first steps towards developing a database program

Now that many of the fundamentals of BASIC have been covered, it is time to put what we have learned to good use by developing a real program. Of course, all the programs we have encountered so far have been 'real' programs in the sense that they performed a specific task, but they were illustrations of how various parts of the BASIC language work rather than the kind of program you might want to use every day. They illustrated how the 'cogs' of BASIC could be put together to make a simple mechanism. Now let's build a whole clock!

One of the most common questions put to computer users by non-users is: 'What can the computer actually be used for?' The question is not as simplistic as it seems. The usual responses tend to go along the lines of: 'Well, you can computerise your recipes' or 'You can create a computerised telephone or address book.' This tactic seldom works because the questioner usually responds with remarks like: 'I can look at my recipe book when I want to cook something and I can look through my address book when I want to find somebody's address, without the trouble of spending hours writing a program to do it.' At what point does a problem become worthy of a computer solution rather than a conventional solution? We'll answer this by working through the specific example of a computerised address book.

An ordinary address book commonly features an alphabetical finger index designed to enable the user to locate, very approximately, the location of any particular name. Names are usually added as and when needed, and not in strict alphabetical order. Your first entry under P might be David Peterson. Later, you might add Brian Peters or Shashi Patel. Although these are not in alphabetical order, they are all grouped together under P so the task of finding any particular surname beginning with P will not be too great. On the other hand, if you did not use any kind of index finding a name would be a nightmare.

The other entries usual in an address book are the person's address and telephone number together, perhaps, with more personal information. A conventional address book, however, couldn't give you a separate list of all the people who live in Birmingham, or who corresponds to the telephone number 258 1194.

Now this may not represent a serious shortcoming, but if you were the owner of a small mail order company it could be a valuable asset if you could obtain specific information about the people on your mailing list. For example, if you had a new line of children's nightwear you could anticipate new orders by informing your clients, but to save on postage it would probably not be worth sending 'mail shots' to customers without children. This is the sort of consideration that must be evaluated before deciding if a problem is worthy of a computer solution or a conventional solution.

If the computer solution is suitable, then the next consideration must be whether to buy commercially available software or not. A glance through the advertisements in the computer magazines would suggest that every possible eventuality has already been thought of by computer programmers. Closer examination, however, may show that a commercially available program may not do exactly what you want, or it may not be available for your model of computer, or it may be too expensive. The cost of a program generally reflects the development costs. A word processor package might cost £350, but writing your own could cost you far more if it takes you six months of dedicated work.

On the positive side, software you write yourself can be made to do exactly what you want it to. The other factor is the sheer satisfaction of successfully writing a major program for yourself and by yourself.

Designing a program involves several stages. The first is a thorough understanding of the problem. This involves a Clear Statement of the Problem — the CSP stage.

The second is to find an approach to the solution of the problem. This involves a description of the expected form of the input and output as a 'first level description' of the problem. The problems and the solutions should be stated in the broadest terms and these should be gradually refined until we are at the stage where we can code into a particular language.

The third stage is the coding itself. We will use BASIC as our high level language, but it could just as well be any other language. Up to the final stage of coding into BASIC, we will use a pseudo-language intermediate between the freedom and flexibility of English and the rigid structures of an actual computer language such as BASIC.

The approach to programming just described is usually called 'top-down' programming. It works from the topmost level — a general statement of the overall objectives, through various levels of