Expanding Files

Having established an overall structure, we now continue our programming project with a look at file handling

The address book program that we have been developing in recent instalments of the Basic Programming course is actually a type of simple database, and as such involves the concept of 'files'. The word is used in a number of related, but slightly different, ways. We shall start by discussing these in a little more detail so that we can subsequently use the word with more precision.

In computer programming, a 'file' can be thought of in much the same way as a file in a filing cabinet. It is a collection of related pieces of information stored together. Computers store files on magnetic tapes or disks. Each 'file' of information is given a unique name so that the computer can gain access to it whenever necessary. The information on a cassette tape or a floppy disk may be a program, or it may be 'data' used by a program. Taking the computerised address book as an example, the information needed consists of two separate parts: the program itself, and the data that the program works on. The program is the set of instructions that allows the computer (and the user) to manipulate and work on the data.

The data used by the program is the set of records containing the information you would expect to find in an address book — names, addresses and so on. It also includes certain types of data not normally available to the user. This is the 'housekeeping' data used by the program to help it work. Examples of this type of data might include: 'flags'; information relating to the current size of the database (i.e. the number of records in it); whether or not a sort has been conducted since a new record was last inserted; or possibly an indication of how many times a particular record has been accessed or printed out. The reason why data such as this — and the data comprising the records — needs to be treated separately from the program will become apparent as soon as we try to implement the program.

In earlier parts of the Basic Programming course we have used the READ and DATA statements as ways of putting data to work within a program. This is suitable only if the data is not subject to change, such as the number of days in a month. If the data is liable to change, the program can prompt for it on the screen, and INPUT, INKEY$ or other methods can be used to convey the data to the program. An example of the appropriate use of this form of data input might be a numbers guessing game, in which part of the program might take the form of:

```
PRINT "GUESS THE NUMBER"
INPUT N
IF N <> COMPNUM THEN ...
```

The data in the address book program, however, is subject to considerable modification. In theory, all the records could be stored within the program and read into appropriate arrays using READ and DATA statements. But then all the data comprising the records would have to be entered as part of the program. Whenever changes were made — names and addresses added or removed, for example — considerable alterations would need to be made to the program itself. At the very least, this would involve printing out the program, checking to see where the changes were needed, writing new segments of the program and then typing them in. The biggest problem, however, would be that the new program segments would not be complete program modules that could be independently tested — the changes would be scattered haphazardly throughout the program. The only way of knowing that the modified program worked properly would be to run it and see.

Fortunately, none of this is necessary because data can be stored independently of the program. This is done by creating data files on the cassette or disk. These files are collections of records treated in much the same way as the data in a DATA statement. The program is able to 'open' one or more of these files, read the data from it (usually into an array) and then 'close' the file. If an alteration to the data is needed, the program opens the appropriate file, reads in the data, modifies it, and then writes the modified data back to the file.

With disk-based computer systems, locating a particular file and reading from it or writing to it is quite fast — the location of the file takes only a fraction of a second and the read or write operations usually take a few seconds at most. A cassette-based computer system, on the other hand, may be quite a lot slower and may involve the user in rewinding the tape and waiting for the tape to play through until the right file has been found. Another advantage of using disks is that it is possible to have more than one file 'open' at a time, whereas this is not practical with cassette-based systems.

Files, then, are collections of data stored on a bulk storage medium that are available to be used by one or more programs. A word processing program, for example, might want access to the same set of names and addresses for 'personalised' automatic letter writing.