Files are handled in different ways according to the version of BASIC used. The best way to find out how your computer deals with them is to see what the manual has to say about the OPEN and CLOSE statements and try out a few examples. The description we are presenting here is very generalised and is designed to give an overall impression of using files.

Files may be either sequential or random. In a serial file, the information is stored with the first piece of information first, followed by the next piece, followed by the third and so on. A random file is organised so that the computer can go directly to the piece of data required, without having to start at the beginning and go through the data until the required piece has been located. Watching a film is more like a serial file; you start at the beginning and watch it all the way through to the end. Watching a film on a video recorder at home is a little bit like a random file; you can wind the tape back and forth and watch any part you choose. We shall consider only sequential files, because they're more appropriate to cassette systems.

Suppose you want to keep a record of average daily temperatures for a week. These might be:

MONDAY	13.6
TUESDAY	9.6
WEDNESDAY	11.4
THURSDAY	10.6
FRIDAY	11.5
SATURDAY	11.1
SUNDAY	10.9

To keep things simple, all this data will be treated as numeric data, with Monday being Day 1 and Sunday being Day 7. The data can then be represented like this:

```
1,13.6,2,9.6,3,11.4,4,10.6,5,11.5,6,11.1,7,10.9
```

To store this data in a sequential file, the following steps will be needed in the program:

```
OPEN the file
Write the data to the file
CLOSE the file
```

Whenever the OPEN statement is used it is necessary to state whether we are writing data from the computer to the file (an output) or reading data from the file into the computer (an input). In the BBC Micro, this is done using the OPENOUT and OPENIN statements. The equivalent in Microsoft BASIC is OPEN "O" and OPEN "I". A short program fragment to write the data above into a file (in Microsoft BASIC) would be:

```
100 OPEN "O", #1, "TEMP.DAT"
110 PRINT #1,1,13.6,2,9.6,3,11.4,4,10.6,5,
    11.5,6,11.1,7,10.9
120 CLOSE #1
```

The word OPEN in line 100 makes the file available to the program. OPEN is followed by "O" to indicate that data will go out from the program to be stored in the file. This is followed by #1, which tells the computer that we'll be referring to this as file number 1 in our program. Each file is given an arbitrary number that will subsequently be used with the INPUT# or PRINT# statements when we want to read or write data to that file. Finally, we have the filename in double quotation marks. We've called our file TEMP.DAT to indicate that it contains temperature readings, and is a data file rather than a program.

A complete Microsoft BASIC program to enter the data into a file and subsequently read it out and print it is given below:

```
100 OPEN "O", #1, "TEMP.DAT"
110 PRINT# 1,1,13.6,2,9.6,3,11.4,4,10.6,5,11.5,6,
    11.1,7,10.9
120 CLOSE #1
130 REM LINES 130 & 140 ARE 'DUMMY' LINES TO
140 REM REPRESENT INTERVENING PROGRAM
150 OPEN "I", #1, "TEMP.DAT"
160 FOR X = 1 TO 7
170 INPUT #1, DAY, TEMP
180 PRINT "DAY ";DAY,TEMP
190 NEXT X
200 CLOSE #1
210 END
```

This opens a file, numbered #1 and named TEMP.DAT, writes data into it using the PRINT# statement and then CLOSEs the file. Later in the program the same file is opened using both the number and the filename (the number does not need to be the same as when the file was created, but the number used in the PRINT# or INPUT# statements must be the same as the one assigned to the filename when the file was opened). INPUT #1 in line 170 indicates that the input will come from a file numbered #1 (that is, the file TEMP. DAT) and not from the keyboard.

We shall leave this look at file handling for the moment and return to the address book program and some of the components involved in the INITIALISE subsection of the program. First, let's look at the amount of memory space required for a single record in the address book file (the word 'file' here is being used in the database sense of being the set of all related records, not in the operating system sense of being a named group of data stored on tape or disk).

The use of fixed-length fields is somewhat wasteful of memory space, but makes the programming a lot simpler. If we allow one whole line for each field, with 40 characters to a line, all of this will be saved in an array even if most of the line consists of blank spaces. In some versions of BASIC, however, when string arrays are DIMensioned, each element can be up to 256 characters long. The dimensioning merely sets the number of elements in the array, not the size of each element.

If you have a BASIC that can handle multi-dimensional arrays it would be possible to use a separate dimension for each of the fields, but many versions of BASIC cannot do this so we shall explore alternative approaches. The simplest method is to use a separate string array for each of