Classified Information

When designing any program that involves the storage or manipulation of information, it is important to pay close attention to structure, so that items of data needn't be duplicated

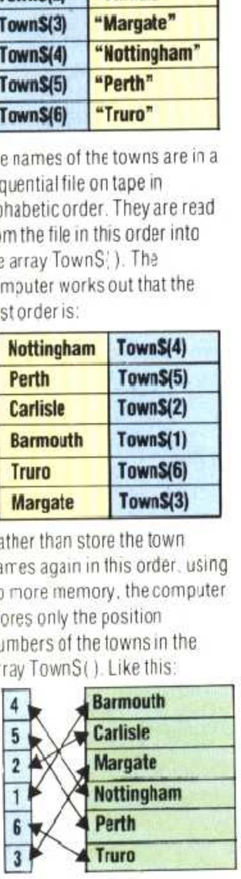
Shortest Route
A Midlands firm has interests in six towns, and has to send a lorry around them every month. The firm's computer can work out the most efficient route.

Town$(1)	"Barmouth"
Town$(2)	"Carlisle"
Town$(3)	"Margate"
Town$(4)	"Nottingham"
Town$(5)	"Perth"
Town$(6)	"Truro"

The names of the towns are in a sequential file on tape in alphabetic order. They are read from the file in this order into the array Town$( ). The computer works out that the best order is:

Nottingham	Town$(4)
Perth	Town$(5)
Carlisle	Town$(2)
Barmouth	Town$(1)
Truro	Town$(6)
Margate	Town$(3)

Rather than store the town names again in this order, using up more memory, the computer stores only the position numbers of the towns in the array Town$( ). Like this:



The list of numbers is an index to the array Town$( ). Other such indexes to the array could be created, each giving a different order to the names. The advantage of indexing like this is that the original file does not have to be sorted or duplicated: only the indexes need be sorted and stored

Anyone who has ever used a card index system (a library catalogue, for example) knows how useful it can be. If you have ever dropped the card box you know that the same cards out of order become maddeningly useless. A library contains a vast amount of information, but unless it's arranged in some sort of order its value as an information system is almost nil.

The essence of an information system is not the information itself but, rather, its organisation. Take, for example, this (fictional) item in a trade directory:

Smith, J., 15 High Street, Middletown.

On its own, its value as information is limited; if, however, you know that it comes from the Ironmongers' section of the directory, then it has a new significance that comes from the structure in which it is placed.

The simplest data structure is in the form of a file: a collection of data with common features. The name of the file reveals something about the information in it, and putting all that information together under that name makes it much easier to use. The file can be treated as one large unit of information, or as a particular grouping of smaller units. A book is a file; Mrs Beeton's autobiography is usually read as a whole, while her cookery book is usually read as a collection of individual recipes.

If the file is large, finding a particular piece of information may mean starting with the first item of the file, and scanning each item in turn until the right one is found. This is called sequential search or serial access, and a file arranged this way is a sequential file. A television programme is a sequential file of information, and so is a human conversation.

Sequential files are common because they are useful and cheap to implement, and because in many ways they mirror human methods of thought. However, they can become unwieldy and slow to use, so they are frequently divided internally into sub-files, which can be found directly without searching through the whole file. Books were once simple sequential files, but the invention of chapters, page numbers, and indexing transformed them. The chapters are sub-files of the book, and the pages are sub-files of both the chapters and the book.

A file that does not require sequential searching is called a direct access file. An album of songs on magnetic tape is a sequential file, while the same album on a long playing record is a direct access file: finding a song on the tape requires starting at the beginning and winding forward, while any track on the LP can be accessed directly by moving the pick-up arm across the LP to the start of the track.

Direct access depends on knowing where things are: in a book the index tells you what is where. Knowing exactly where things are means work (and, therefore, costs money). Indexing a book is an extra task for the author or publisher, and the information in the book may not warrant this expense; novels, for example, are not indexed, whereas textbooks usually are.

Computers process large quantities of information at speed, using a variety of data structures. Data has to be stored permanently on magnetic tape or disk in some structured way — typically in a sequential file — but in quite different ways in the computer's main memory.

Suppose that a bakery has the addresses of all its shops in a sequential file on tape, and wants the computer to print delivery schedules for the drivers who take bread to the shops. The file on tape might look like this:

```
Atkinsons   22 High Street
Brown & Co   108 Alma Road
Edwards   49 Barking Lane
Wilson Bros   7 High Street
Wrights   65 Lower Road
Youngers   31 Parsons Hill
```

When the file is read from tape into main memory, each name and address will be stored in a numbered location of its own, and all these locations together form a block of memory with its own name, so the file in memory looks like this:

```
BLOCKNAME: Shops
1) Atkinsons   22 High Street
2) Brown & Co   108 Alma Road
3) Edwards   49 Barking Lane
4) Wilson Bros   7 High Street
5) Wrights   65 Lower Road
6) Youngers   31 Parsons Hill
```

Now the data items can be accessed individually by naming the block and the location within it. Shops(4), for example, contains Wilson Bros, 7 High Street. This structure is called an array (see page 194), the data structure most commonly used by computers for internal data processing. It is like a book with one piece of information on each page. Notice that this simple structure immediately