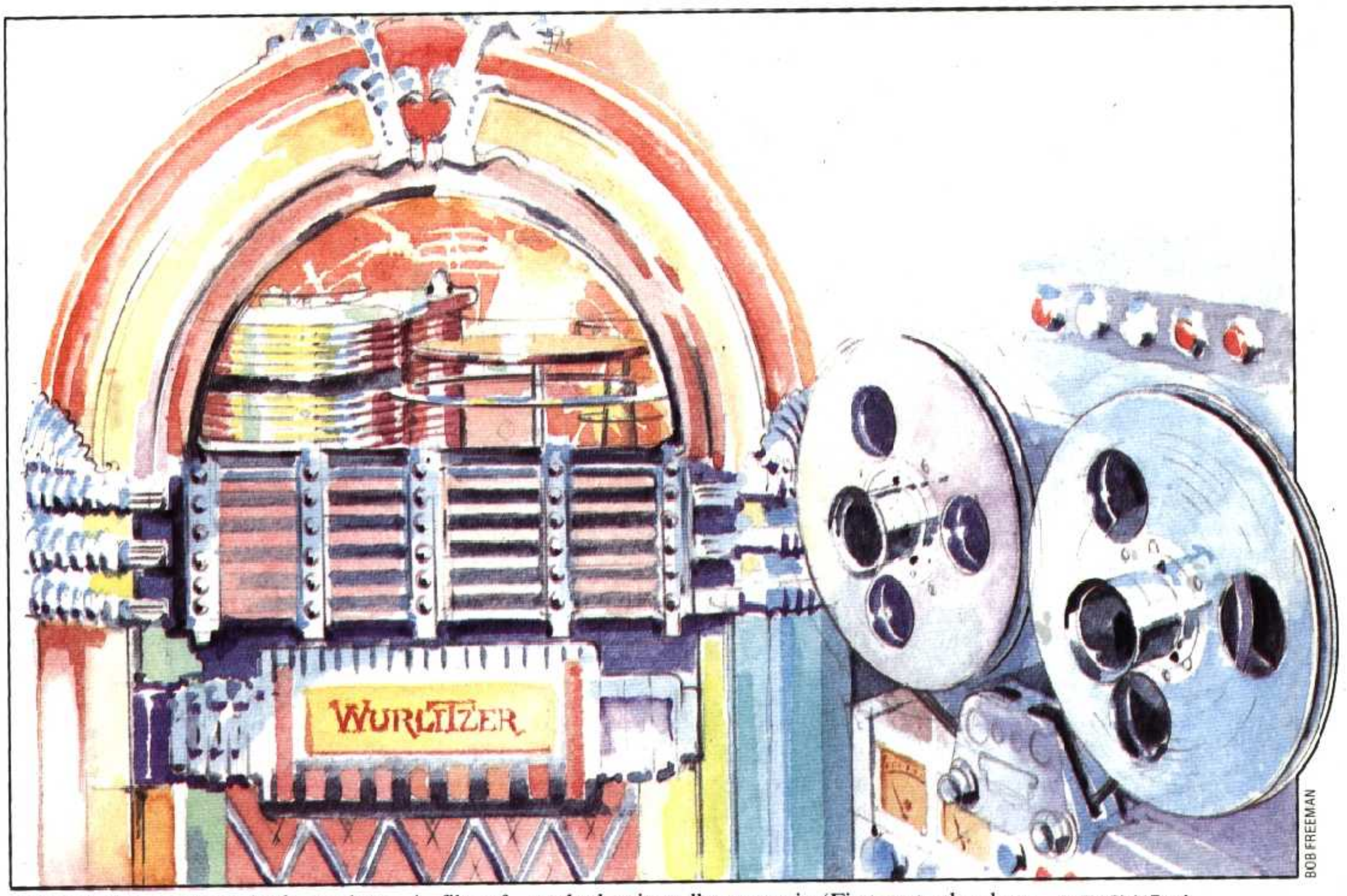

changes the way we look at data. A file of recognisable names and addresses has become a block of anonymous data. Computers do not need to know what a data item means, only where it is, and what to do with it.

The data in the array Shops( ) is in alphabetic order, but this is unlikely to be the most economical order in which to visit the shops. Suppose the computer works out that the best delivery schedule is:

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

This schedule might be stored in another array, but that would mean the same information is stored twice in the memory. Micro owners will know that RAM is limited, and it might be inconvenient or impossible to duplicate data in that way, so another method is needed.

If the actual data are replaced by their position numbers in the array Shops( ), then the delivery schedule looks like this:

```
BLOCKNAME: Deliveries
1)  4
2)  1
3)  3
4)  2
5)  6
6)  5
```

and what it really means is, 'First go to the shop whose details are stored in Shops(4), then go to Shops(1), then to Shops(3) . . .', and so on. The only significant information in the schedule is the order in which to visit the shops, so this is all that needs to be stored in the new array, Deliveries.

Deliveries( ) is now an index to the array Shops( ) for the purpose of deliveries. When printing this schedule the computer will use the numbers in the array Deliveries( ) to print the names and addresses from the array Shops( ) in the correct order.

In this simple exercise, information — the shops' names and addresses — has been manipulated but not changed by the different data structures imposed upon it. A data structure does not change the content of the data, but gives significance by associating it in an ordered way with other data.

Just as we can re-arrange the array Shops( ) by re-indexing it according to the array Deliveries( ), so can we construct other indexes to serve other purposes. When we discussed databases (see page 124), we noted that certain information could be selected by reference to pointers included in each individual record. In this way we can 'embed' in each record of the file Shops( ), a pointer that would indicate its place in the delivery schedule. We could further extend the record to include, for instance, a pointer into a file of standing orders — Atkinsons, for example, always have 48 white loaves, 12 wholemeal loaves, and so on. The production department could then run through the file extracting information relating only to the number of loaves that they need to bake.

On The Right Track
A juke box contains 200 songs on 100 record discs. It costs £2,000, and weighs 80kg (176lbs). To select any song, press three keys. Average time from SELECT to PLAY is 15 seconds. A reel of magnetic tape on a tape recorder may contain the same 200 songs. The tape recorder costs £500, and weighs 10kgs (22lbs). To select any song, rewind the tape, press PLAY, and wait. Average time from SELECT to PLAY is 1,500 seconds.

A juke box is a direct access device: it is fast, fairly specialised, and expensive. A tape recorder is a sequential access device: slow, much less specialised, but reasonably cheap. A cassette recorder connected to a microcomputer is a sequential access device, while a floppy disk drive is a direct access device, even though it may be used to store sequential files