establish early, each time the address book program is used, and one that is likely to need updating several times during the use of the program. It is therefore going to be one of our 'global' variables, and establishing its value will need to be part of an 'initialisation' routine. This can be done every time the program is run, or a 'flag' can be created which indicates whether or not the value of POSITION has changed since the program was last run. The latter approach is not difficult, but at this stage it creates an unnecessary complication. We'll keep things simple and find the value of POSITION as one of the early tasks whenever the program is run.

Let's revise the activities we want the computerised address book to do and see if we can move towards an overall program strategy. This time we'll be slightly more rigorous and assume that each of the activities will be dealt with as a separate subroutine (the name of which will be indicated by being enclosed in asterisks).

| | |
|---|---|
| 1. Find record (from name) | *FINDREC* |
| 2. Find names (from incomplete name) | *FINDNAMES* |
| 3. Find record (from town) | *FINDNMTWN* |
| 4. Find records (from initial) | *FINDINIT* |
| 5. List records (all) | *LISTRECS* |
| 6. Add record | *ADDREC* |
| 7. Change record | *MODREC* |
| 8. Delete record | *DELREC* |
| 9. Exit program (save) | *EXPROG* |

We now know, in broad terms, what the desired 'inputs' and 'outputs' of the program are, so we can already start thinking in terms of a main program. All the detailing can be done through the process of top-down programming and coded in the various subroutines. We know that several things will need to be initialised, including the value of POSITION. We know that, as the program is to be menu-driven, we will be presented with a set of choices whenever the program is run. We also know that, whatever our response to the choices presented, we will want one of them at least to be executed.

So the body of the main program can already take shape:

```
MAIN PROGRAM
BEGIN
  INITIALISE (procedure)
  GREET (procedure)
  CHOOSE (procedure)
  EXECUTE (procedure)
END
```

In BASIC it would look like this (with line numbers substituted for the subroutine names):

```
10 REM H.C.C. ADDRESS BOOK PROGRAM
20 GOSUB *INITIALISE*
30 GOSUB *GREET*
40 GOSUB *CHOOSE*
50 GOSUB *EXECUTE*
60 END
```

The *GREET* subroutine or procedure would display a greeting on the screen for a few seconds, followed by the menu. The greeting could, perhaps, look like this:

```
                *WELCOME TO THE*
             *HOME COMPUTER COURSE*
          *COMPUTERISED ADDRESS BOOK*
(PRESS THE SPACE-BAR WHEN READY TO CONTINUE)
```

In response to the request to press the space bar, the program will branch to the *CHOOSE* subroutine and the user will be presented with a screen like this:

```
             *DO YOU WISH TO*
1. FIND A RECORD (from a name)
2. FIND NAMES (from part of a name)
3. FIND RECORDS (from a town)
4. FIND RECORDS (from an initial)
5. LIST ALL RECORDS
6. ADD A RECORD
7. CHANGE A RECORD
8. DELETE A RECORD
9. EXIT AND SAVE
              *CHOOSE 1 TO 9*
            *FOLLOWED BY RETURN*
```

At this point, the program will branch to the appropriate subroutine depending on the number entered. The structure of the program is now beginning to take shape. All options except number 9 (to EXIT and SAVE) will need to end with an instruction to return to the *CHOOSE* subroutine. But there are many details of the internal organisation of the data that we have not considered. We will come to these later.

Let's assume that we are running the program, that it already has all the records in it that we need, and that we want to search for a full record by inputting a name only. This calls for option 1 — FIND A RECORD (*FINDREC*). Before we attempt to design this part of the program, let's consider some of the problems involved in computerised search routines.

## Searching

Textbooks on programming techniques tend to deal with searching and sorting together. Readers may recall that we have already touched on sorting in a program designed to sort names into alphabetical order (see page 134). Both sorting and searching raise interesting points about how data is organised — in a computer or any other information system.

If a 'manual' address book comprised a notebook without a thumb index, and if entries were added when new names and addresses were thought of, without being sorted into alphabetical order, we would have a data structure known as a 'pile'. A pile is a set of data collected in the order in which it arrives. It is obvious that a pile is the least effective way of organising data. Every time you want to find someone's address and telephone number you have to look through the whole address book. The same is usually true of