Dean area, we might want the program to search for every occurrence of 'Cinderford' in the TOWN field. The program could search through the TOWN fields and note the location of each occurrence of Cinderford. All that would then be necessary, to print the names and addresses of all these friends, would be to retrieve all the elements having the same number from all the arrays for each 'Cinderford' record. Using this approach, there would be no need to inspect the INDEX field, and the technique has the merit of being a relatively simple operation.

In the next instalment we will look at some of the problems involved in searching through lists to find specific items.

## Exercise

■ Assume that records with the following fields will be adequate for our computerised address book:

NAME field
STREET field
TOWN field
COUNTY field
PHONE NUMBER field

Suppose that one of the options offered by a menu in the computerised address book is:

5. CREATE A NEW ENTRY

You type 5 and the program branches to the part where new records are created (you may assume that there are no entries in the address book yet). Since the program is to be fully menu-driven, you will always be prompted for the entries expected — with prompts such as ENTER THE NAME, ENTER THE STREET and so on. Here is a list of the expected results:

1. An element in an array for the name
2. An element in an array for the street
3. An element in an array for the town
4. An element in an array for the county
5. An element in an array for the phone number

Your task is to develop this, through a process of top-down programming using a pseudo-language, to a point where direct conversion into BASIC becomes possible. The pseudo-language can follow your own rules; we only suggest that you use capital letters for keywords such as IF, LOOP and so on, and small letters for descriptions in ordinary English of the operations to take place.

## Basic Flavours

SPECTRUM

**Step 3**

```
  10 INPUT "INPUT FULL NAME";F$
  15 LET COUNT=0
  20 FOR L=1 TO LEN F$
  30 LET C$=F$(L)
  40 IF C$=" " THEN LET COUNT=L
  50 NEXT L
  60 PRINT"LAST SPACE IS IN
     POSITION";COUNT
  70 STOP
9990 DEF FN M$(X$,P,N)=X$(P TO
     P+N-1)
9991 DEF FN L$=X$(    TO N)
9992 DEF FN R$=X$(LEN X$-N+1 TO    )
```

In this programming project, the string functions MID$, LEFT$, RIGHT$ will be much used. Their equivalents in Sinclair BASIC are:

| | |
|---|---|
| LEFT$(F$,N) | replace by F$(  TO N) |
| RIGHT$(F$,N) | replace by F$(LEN(F$)-N+1 TO  ) |
| MID$(F$,P,N) | replace by F$(P TO P+N-1) |
| MID$(F$,P,1) | replace by F$(P) |

Note that string variable names on the Spectrum cannot be more than one letter long (plus the "$").

**Step 4**

```
 5 LET S$=""
10 LET F$="TOM BROWN"
20 LET COUNT=4
30 FOR L=COUNT+1 TO LEN F$
40 LET S$=S$+F$(L)
50 NEXT L
60 PRINT "SURNAME IS  ";S$
70 STOP
```

**Step 5**

```
 5 LET C$=""
10 FOR L=1 TO COUNT -1
20 LET T$=F$(L)
30 LET CHAR=CODE T$
40 IF CHAR>64 THEN LET C$=C$+CHR$
   CHAR
50 NEXT L
60 STOP
```

In this fragment, the problem of single letter string variable names has arisen: F$ is the Spectrum equivalent of the variable FULLNAME$, so C$ has to stand in for the variable FORENAME$.

**Part of subroutine X**

```
FOR L=1 TO LEN W$
LET C$=W$(L)
IF C$="." THEN GOTO 1550
NEXT L
```

**Part of subroutine Y**

```
FOR Q=1 TO LIMIT
LET A(L)=P(Q)
NEXT Q
```

VARIABLES

Of the most popular home computers, only the BBC Micro supports long variable names such as FULLNAME$. The Spectrum allows long numeric variable names, but only single letter string variable names. The Dragon 32, Vic-20, and Commodore 64 support long variable names, but only the first two characters are significant, so that FULLNAME$ is valid, but refers to the same memory location as FUJIYAMA$: both have the same first two characters.

On the Oric-1 variable names cannot be more than two characters (first a letter then a number or a letter), while the Lynx allows only single letter variable names, though both lower- and upper-case letters are valid and distinct.