Finishing Touches

By removing the anomalies caused by stringing together the modules, and adding a few more facilities, our address book program is now complete

In the last instalment of the course, readers were left with the problem of working out why running the address book program, then adding a record (using *ADDREC*), then locating a record (using *FINDREC*), and then exiting from the program (using *EXPROG*) would result in the added record not being saved. The problem arose through the use of the variable RMOD as a flag to indicate that a record had been modified (implying that the file might be out of order). The *SRTREC* subroutine would sort the file into alphabetical order, and then set RMOD to 0 on the assumption that the file is in order. Executing *EXPROG* checked to see if the file was in order (RMOD = 0) and didn't bother to save the file if it was in a sorted condition.

Adding a record (using *ADDREC*) would set RMOD to 1 (since a record had been modified, i.e. a new record had been added), but *SRTREC* would set RMOD to 0, indicating that the file had been sorted. What is really needed, however, irrespective of whether the file has been sorted or not, is a flag that signals that a record has been modified and a separate flag to show if the file is in a sorted condition or not. Then, subroutines that need to know that the file is sorted can check the 'sorted' flag, and subroutines that need to know if any record has been modified can check the 'modified' flag.

Suitable names for the two flags would be RMOD, to show if a record has been modified, and SRTD, to show if the file has been sorted.

When the program was presented on page 399, line 1230 contained the statement LET SVED = 0. The SVED variable has not been used so far, but when the line was included, it was realised that RMOD alone would not be enough. The variable name SVED was chosen with the idea that certain conditions would have to be true before a save (to tape or disk) would be necessary.

A more appropriate name for this flag would be SRTD (to indicate that the file is in a sorted condition). The original line 1230 has been changed to:

```
1230 LET SRTD = 1
```

There are now four possible states regarding the condition of the data file. These are:

RMOD	SRTD	
0	0	Not modified, not sorted (illegal)
1	0	Modified, not sorted
0	1	Not modified, sorted
1	1	Modified, sorted

RMOD=0 and SRTD=0 is illegal because the program ensures that the data file is always sorted before it is saved. When the program is run, RMOD is set to 0 (line 1220) to indicate that no modifications have taken place, and SRTD is set to 1 (line 1230) to indicate that the file is sorted.

Any operation that modifies a record (such as *ADDREC*, *DELREC* or *MODREC*) sets RMOD to 1 and this flag is not reset by any subsequent operation. SRTD, which is initially set to 1, is reset to 0 by any activity that might mean the data has become out of order (such as in *MODREC* if the name field is altered). Any activity that needs to assume the data is sorted (such as *FINDREC*) always checks SRTD and calls the sort routine if SRTD = 0. By using these two flags, instead of just RMOD, the program is able to terminate without saving the data file if no modifications have taken place during the current run of the program. It will not be 'tricked into' terminating without saving if a sort takes place after a record modification.

The other variable not used so far is CURR. This variable is used to save the 'current' position in the array of a record after one has been located by the search routine. CURR is not cleared after a value has been assigned to it; it is used to carry information about the target record to other routines in the program. The end of the *FINDREC* (search) routine has been modified in lines 3320 and 3330 to set the value of CURR: to 0 if the search failed to find the target record; and to MID if the search was successful.

Line 13340 branches to the *NOTREC* subroutine if CURR is 0. This displays a message saying that the record has not been found and displays the search key ,NAMFLD$(SIZE). *NOTREC* returns to the main menu after the space bar has been pressed. *NOTREC* could be modified quite easily to give the user the opportunity to:

```
PRESS RETURN TO TRY AGAIN OR
SPACE BAR TO CONTINUE
```

It might appear that the easiest way to achieve this would be to call *FINDREC* again if RETURN were pressed. However, calling a subroutine from within itself, whilst not illegal in BASIC, 'confuses' the return address and will cause the subroutine to be repeated again even when you don't want it to. There are ways of getting round this problem, but the programming starts to get a bit tricky!

An easier way would be to have used a flag (such as NREC for not record) and reset it in *NOTREC*, allow the subroutine to return in the