Assembly Lines

We can now bring together the sub-programs that will process our computerised address book, and examine ways of making the program more user-friendly

Although many details of the address book program have yet to be finalised, the overall structure should now be becoming clear. At this point in the development of a program of any size it is as well to draw a block diagram of the program and to think through the flow of activities in the program.

This is also the point at which the program writer should think about the 'human interface' and 'user image' aspects of the program. These important concepts and practices are often not given the attention they deserve, even in commercial software.

'Human interface', simply defined, means the 'ergonomics' of the software, or how easy it is to use. 'User image' is concerned with how the user perceives the program being used. We shall examine these concepts with regard to our program as developed so far, and determine to what extent we will be able to implement them.

The table shows the main blocks of the program that have been considered so far. As a convention, and simply to keep things tidy, we have used names with six characters for procedures or subroutines, seven characters (including $) for string arrays, four characters for simple numeric variables and five characters (including $) for simple string variables that are global. Local variables (in loops, for example) will usually be single letters.

MAIN PROGRAM BLOCKS

	CREARR	(creates arrays and initialises variables)
INITIL	RDINFL	(reads in file from tape or disk)
	SETFLG	(sets flags and modifies variables)
GREETS		(prints greeting message)
CHOOSE	CHMENU	(prints options menu)
	INCHOI	(assigns option to CHCE)
	FNDREC	(locates and prints a record)
	FNDNMS	(locates names from partial input)
	FNDTWN	(locates records for specific town)
	FNDINT	(locates names from initials)
EXECUT	LSTREC	(lists all records)
	ADDREC	(adds a new record)
	MODREC	(modifies existing record)
	DELREC	(deletes a record)
	EXPROG	(saves file and exits program)

Each of the major program blocks in the second column needs to be further broken down into sub-units, and the sub-units will need to be further refined until we have enough detail to write the actual code in BASIC. The processes involved in this form of 'stepwise refinement' have been illustrated for many of the blocks in earlier parts of the Basic Programming course.

Assuming that all or most of the program modules have been worked out, coded into BASIC, and individually tested, how can they be linked together to form a complete program? The best way to tackle this problem is to save each module on tape or disk, giving it the same filename used in the program development notes. Thus ADDREC can be written and tested as far as possible, and then saved under the filename ADDREC. Normally, when a program is loaded from tape or disk, we use the LOAD command, followed by the filename, as in LOAD "ADDREC". This has the effect, however, of clearing everything in memory, so if we load ADDREC and then subsequently load MODREC, the whole of the ADDREC program will disappear.

Fortunately, there is a partial solution. The MERGE command loads a program from tape or disk without erasing any program already in memory. But there is one important proviso. If any of the program line numbers in the MERGEd program are the same as line numbers in the program already in memory, the new line numbers will overwrite the old line numbers and cause chaos. Versions of BASIC with the RENUM command can get round this by renumbering the lines in a program module before they are saved, so that when they are merged there will be no conflict.

Unfortunately, many versions of BASIC on home computers do not have the RENUM command, and therefore careful planning of the line numbers from the beginning will be necessary. When a full chart of all the major program modules has been worked out (as we have partially done in the table), starting line numbers for each block can be assigned. Parts of the program likely to have extensive modifications or changes, such as the main program or file-handling parts of the program, should have increments of 50 or even 100 in order to leave plenty of room for additions. Program modules less liable to modification, such as the GREETS routine, can have line number increments of 10. Putting lots of blank REMs in the program not only makes the program easier to read, but also allows additional statements or calls