to subroutines to be added later. If your BASIC doesn't feature MERGE either, then you will have to type in the various modules as they are written, and save them together.

The program blocks in the table have been merged as a 'trial run' in the listing printed here, to illustrate the pitfalls of the 'try it and see' approach encouraged by languages such as BASIC. Our program would not run properly because the flow of control through it has not yet been thought through carefully enough. There is no point in typing in the whole of this program just to find out that it will not work, but if you have saved all the routines from earlier parts of the course, and if your BASIC has the RENUM command, you can try renumbering and then MERGEing to produce a similar listing.

The first block in the main program is INITIL, which is supposed to initialise variables, dimension arrays, read in files, assign the data to the arrays, set flags and so on.

The *INITIL* subroutine is broken down into *CREARR* (to create the arrays), *RDINFL* (to read in files and assign the data to the appropriate arrays) and *SETFLG* (to set flags etc.).

When all this has been done, the program moves on to *GREETS*, a subroutine to print a greeting message on the screen. The last part of this routine waits for the user to press the space bar for the program to continue.

The program then goes on to *CHOOSE*. This comprises two parts: the first presents a menu of the options offered by the address book program; the second accepts the choice input from the keyboard and assigns the (numeric) value to a variable called CHOI.

The value of this variable is used by the next program block, EXECUT, to select one of nine further program blocks. All of these, except EXPROG, will need to return to *CHOOSE* after they have been executed so that the user has the opportunity of selecting another option. This will not be required if 9 (EXPROG) has been selected because this option is supposed to terminate the operation of the program.

The chief problem with this program as it stands is that the control flow is not correct. INITIL insists that we read in a file from mass storage whether a file exists or not. If the program is being run for the first time, no records will have been entered and there will be no data files on the tape or disk. Any attempt to open and read a non-existent file will result in an error message and the program will not work.

What is required is to have the *RDINFL* routine called only by one of the EXECUT modules, and then only once each time the program is run. This suggests that there should be an INFL flag, originally set to 0, that will be set to 1 once the file has been read in. If it has been set to 1, it will inhibit further attempts to read in. ADDREC will then always search through the arrays to locate the first empty element and will write the information there. This record will almost certainly not be in the proper sort sequence, so there should be a RMOD flag which will be set to 1 when executing The RMOD flag should also be set to 1 if MODREC or DELREC are executed. You can try writing the relevant code to achieve this, or if you simply want to run the program change line 1310 to RETURN.

Adding a record, deleting one or modifying one all mean that the sequence of records is likely to be out of order, so any module (FNDREC, for example) should first check RMOD to see if any changes have been made. If they have, we could either insist on a sort before a search is made, or put up with an inefficient search through a pile. EXPROG will automatically check RMOD and call the sort routine if it is set (to 1) before saving the data in the file on tape or disk.

The 'human interface' aspects of the program, which we mentioned earlier, can be broken down into the following categories:

User interface
User image
Error recovery
Security
Adaptability

User interface refers to the way the user of the program communicates with the program. We have opted for the use of menus throughout (rather than commands). Many people prefer commands, but the important point is that, whatever form of intercommunication is used, it should be consistent. Similar commands should not do different things in different parts of the program. If they do, the user has to read each menu carefully before each choice is made and 'reflexes' cannot be built up.

As our program stands, it is poor in this respect: the greeting message is terminated by hitting the space bar; the options menu is terminated automatically by hitting any of the number keys from 1 to 9; and the data entry in ADDREC is terminated (for each field) by hitting the RETURN key. This kind of inconsistency may be acceptable in a 'home-brew' program, but should be considered unacceptable in commercial software.

User image refers to the way the user perceives the operation of the program. It is considerably influenced by the quality of the user interface. Most of the operations going on inside the computer are completely hidden from the computer operator. The only way the operator can form an idea of what's going on in the program is from the visual input he receives from the screen in response to the inputs from the keyboard. The user image we would want from our address book program would be that of an actual, physical address book. Similarly, the user image desirable from a word processing program would be that of a piece of 'paper' (on the screen) upon which we type. In this case, ideally, bold type would appear bold on the screen, underlined type would be underlined, and justified type (type with a straight right margin) would be justified on the screen.

A perfect user image is seldom possible — no