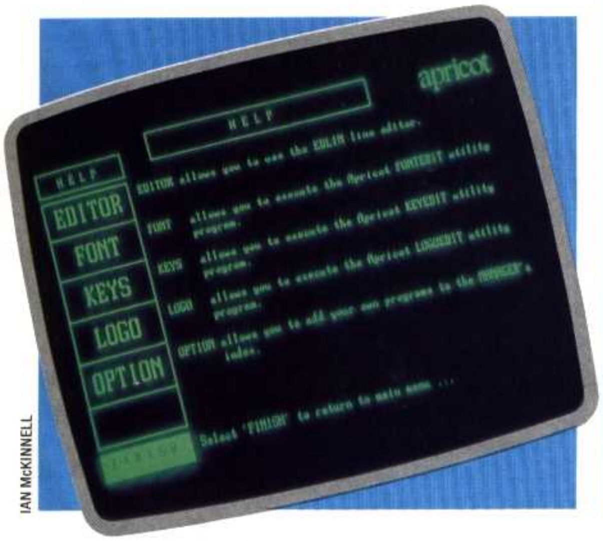
number of help pages, so again a general help routine is desirable. This may require the user to input a number to identify the particular help page required. On disk-based systems, the help pages may be held on disk as separate files. The help routine will then create the appropriate file name from the user's input, read in the file and display it on the screen.

Both help and instructions routines may well take up more than just a single page of information. If this is the case, your display routine should be designed in such a way that the user is able to move backwards and forwards through the pages at will. You should also ensure that the user can leave the routine at any stage and return to the exact point at which the main program was left — it



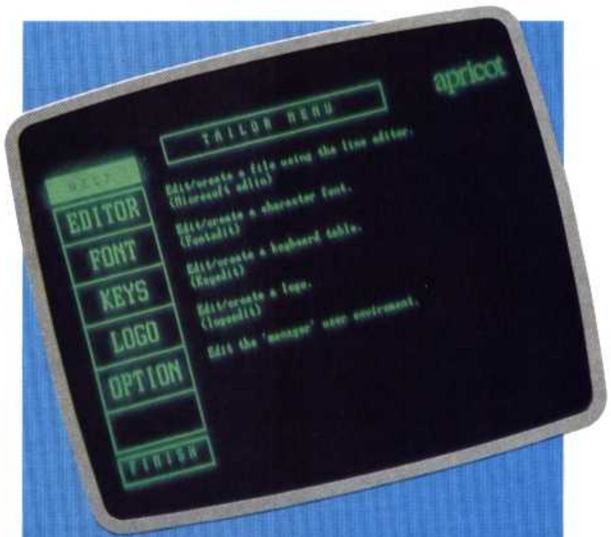
```
HELP
EDITOR allows you to use the EDLIN line editor.
FONT   allows you to execute the Apricot FONTEDIT utility
       program.
KEYS   allows you to execute the Apricot KEYEDIT utility
       program.
LOGO   allows you to execute the Apricot LOGOEDIT utility
       program.
OPTION allows you to add your own programs to the MANAGER's
       index.

Select 'FINISH' to return to main menu ...
```

IAN McKINNELL

is very frustrating to go through 10 pages of redundant information each time the instructions are required! If a prompt had been given, it will now have been lost so it must be repeated. The help routine should set a flag that tells the calling routine that it must go back to the last instruction before the help call, first clearing the flag.

A common metaphor for user interactions with complex programs is to think of the user navigating through a tangled network of logic. The newcomer to the program will not understand its structure and can easily become disoriented and lost. Thus 'signposts' are needed to guide the user. A menu is the clearest example; this operates like a road sign that shows the possible exits from a junction. Systems such as Apple's Macintosh and Lisa work in a similar way, using icons instead of menu options.

Some directions are more important than others. In a command-based system, there may be dozens of possible commands. However, not all of these will be relevant or even possible at a given point in the program. If the number of options is small, it is useful to display a line or two to explain what they are. Some options — such as QUIT — must be available at all times, so it is a good idea to keep these on permanent display. UNDO, SAVE and other application-specific commands may also be constantly available. A common technique is to program such commands onto function keys and display a single-line message to show each key's function. It is always good practice to display a



```
TAILOR MENU
Edit/create a file using the line editor.
(Microsoft edlin)
Edit/create a character font.
(Fontedit)
Edit/create a keyboard table.
(Keyedit)
Edit/create a logo.
(Logoedit)
Edit the 'manager' user environment.
```

Good Management
The ACT Apricot's Manager software guides the user through a suite of unfriendly utility programs by its hierarchical menu system. Help is an option on every menu, and consists of an explanation of the other menu items. This is a good example of classic menu-driven software supported by large Help files

signpost that indicates the way out of a program — this instils confidence in first-time users, whose major concern is often to find the emergency exit!

Some experimental systems have been developed that can monitor a user's performance and adjust the level of help given accordingly. Commercial programs with this feature are still a long way off, but it is possible to use simple techniques to achieve at least a part of this goal. If the user is asked to give his or her name each time the program is run, then a file can be kept of users and their skill levels. These levels can be calculated (from the number of times a particular user has run the program, say, or from the highest score achieved if the program in question is a game) and updated at the end of the program run. As the skill level increases, the type of help and signposting supplied will change, becoming briefer and less intrusive. The user might also be asked to choose the level of help required, as in the Wordstar word processing package. Ideally, both alternatives would be used.

Incorporating help can be a valuable guide to improving a program's performance. Once a help routine has been designed (such as the one provided here) it is a simple matter to modify it to record which help pages were used and how often they were needed. This gives a clear indication of the trouble-spots in the program.