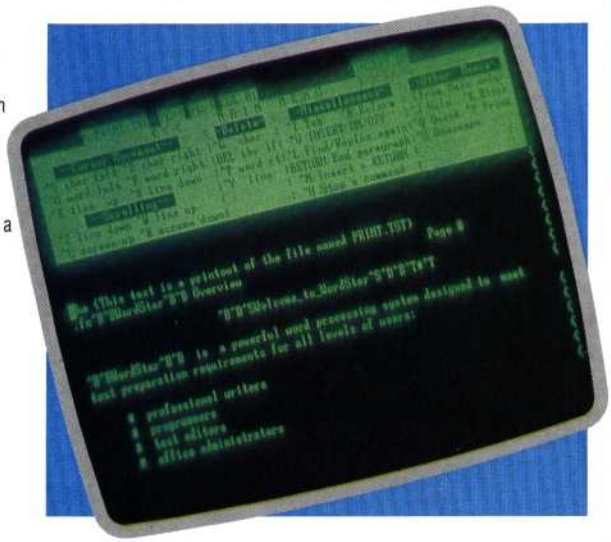
GUIDELINES

As more memory becomes available on microcomputers, the techniques used to guide the user through the workings of a program can become more sophisticated. Here we discuss the design and implementation of general-purpose 'help' routines that may be incorporated into your own programs.

Memory is now cheap. The next generation of home computers, which may have a *minimum* of 128 Kbytes of RAM, will leave most of us with far more memory than even our most ambitious programs will ever require. Throughout the history of computing, a shortage of memory has been the major excuse for failing to provide users with sufficient instructions, sensible error messages or on-line help. Now there is no excuse.

There are three main user aids that can be provided within a program: instructions, 'help' pages and 'signposts'. Instructions take two forms. They can be given in a single block at the beginning of the program, or they may be supplied as required throughout the program (as prompts for user input, for instance). Ideally, both should be available to the user.

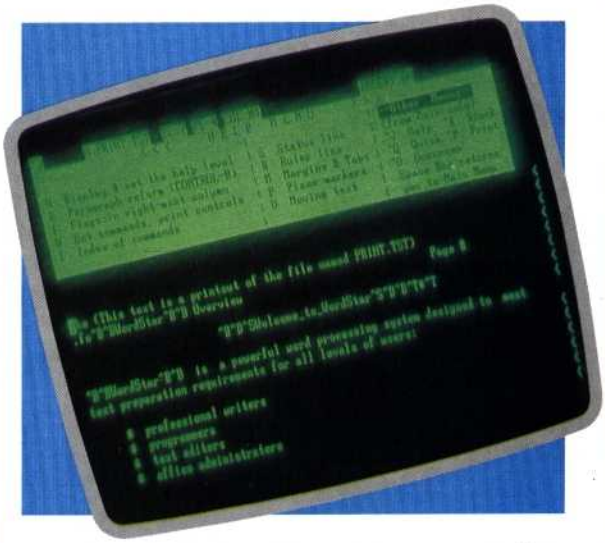
In their simplest form, instructions may simply be a page — or several pages — of text explaining in clear English how to use the program. The text



Words Of Advice
Micropro Wordstar provides a top-selling example of command-driven software with 'on-line help'. The on-screen help menu can be abbreviated or removed by the user, but an enormously detailed Help file structure is always available at a single keypress

can be held in strings or DATA statements within the program, and will be displayed when required by a call to a subroutine written for this purpose. At the start of the main program, the user is asked if instructions are needed; if they are, the subroutine is called. Thereafter, other routines that accept user input should be tailored so that a specified input ('?' is common, or you could use 'I') triggers a call to the instructions subroutine. It is a good idea to create a standard 'display instructions' command, and modify any library input routines to accept it. Don't forget to modify any prompts used in your routines so that 'Press any key for more...' becomes 'Press any key for more (or "I" for instructions)'. This will give you a standard format that will be used in all your programs.

But instructions need not be text-only. Diagrams may be included, and the instruction routines can be developed to give examples and allow the user to practise and learn. Such instruction routines are common in programs that run scientific experiments — here the user may be required to perform a specified task to a particular level of skill before being allowed to progress to the main program. Such 'teaching' routines are not easy to develop because they must simulate the



behaviour of the rest of the program, as well as evaluating the user's performance. It is well worth the attempt, however, as designing this type of routine will give an indication of the problems that the main program will present to the user.

In a similar fashion, 'help' pages may be called up to explain the operation of particular parts of the program. This facility is found in many systems, where it is available to explain the use of commands — the Unix operating system, for example, allows the entire user manual to be accessed as on-line help! Providing help in your own programs need be no more difficult than supplying instructions: at each appropriate point, simply allow the user to enter a help request instead of the usual input — when this happens the program should call the relevant help routine. A complex program is likely to require a large