few editing commands) or re-typing the correct line. Whenever the BASIC interpreter encounters an error in syntax or logic it stops the execution of the program and reports the error. Fixing the bugs is as simple as trying a new line in place of the erroneous line, and typing RUN<CR> again.

Basic's Disadvantages

The BASIC programming language has a number of disadvantages, however, some of which are subtle and some glaring. Because it is interpreted (although a few compiled versions of BASIC do exist) it runs very slowly. If speed is not very critical (as in a program to calculate your current bank balance, for example) the slowness of interpreted BASIC will be of no consequence. If, on the other hand, speed is of the essence (as in a screen animation program using graphics, or a 'clock' used to time reactions in a laboratory experiment) interpreted BASIC is likely to be far too slow.

If you need speed in your programs, there are two routes to follow: programming in either machine code or Assembly language (see page 448) — a difficult and time consuming process — or programming in a compiled language such as PASCAL or FORTH. Compiled languages are not difficult to learn, but the source code (the original program) is almost sure to contain bugs, which the compiler will find when it tries to compile the program. These are difficult to rectify, compared with bugs in BASIC. After corrections have been made to the source code, the program will have to be compiled all over again. Most compilers take two or three 'passes' through the source code, and each pass is likely to result in error messages, each of which will have to be corrected before the program can be re-compiled.

Producing a correctly compiled program is likely to be a far more time consuming processs than achieving a working program in interpreted BASIC. On the other hand, BASIC is likely to lead the novice programmer from the 'straight and narrow' by allowing bad programming techniques that highly structured languages such as PASCAL would reject. BASIC allows the programmer to write very careless programs, full of GOTOs for example, and these bad habits can make the transition to more advanced languages difficult.

What Next After Basic?

BASIC is a flexible language, and one that is not difficult to learn. It has excellent string handling facilities, but is slow and fails to take full advantage of the power of a home computer. On the other hand, more modern languages, such as PASCAL and FORTH, offer programming facilities either difficult or impossible in BASIC.

PASCAL was also devised as a teaching language, and specifically designed to encourage the development of well constructed, 'structured' programs. PASCAL is a compiled language, which means that users encounter numerous errors picked up by the compiler (after the source code has been written and before the compiled 'object code' can be run), and this can be very frustrating. Novice PASCAL programmers also tend to find the restraints of the language, such as the need to declare all variables at the beginning of the program (and to state what type they are — real, integer, etc.), to be an impediment to free and flexible programming.

On the other hand, PASCAL demands that the programmer thinks through the logic of the program properly before writing. Programs in PASCAL are likely to throw up numerous syntactical errors in the source code. But they are also more likely to be well designed and less likely to contain fundamental logical errors.

FORTH has recently become a very popular alternative to BASIC as a programming language on home micros. Although FORTH is not as difficult to learn as Assembly or machine code language, it must be said that it is far less 'intuitive' than either BASIC or PASCAL. Even so, FORTH has many unique merits that make it a contender as the programmer's second language.

Although FORTH is a high level language, it runs nearly as fast as machine code, owing to the unique way it works. Whereas languages such as BASIC have a fixed number of statements and commands, FORTH users can define their own vocabulary.

The keyword PRINT in BASIC means that any character following it enclosed in double quotes will be printed on the screen. Nothing the programmer does can alter this. In FORTH, PRINT could be defined to produce, say, a listing on the screen of the hexadecimal equivalents of the ASCII codes, printed in a vertical column, of the characters in a string.

FORTH gives the programmer the power to define any word to mean whatever is wanted and to produce the desired results whenever it is used from then on. Not only is FORTH extremely flexible in this way, but it also produces programs that can be compiled to object code (see page 184) which are nearly as compact and fast running as machine language programs.

Although there are many programming languages available, most hobbyists moving on from BASIC will be inclined to choose from Assembly language, PASCAL and FORTH. Very briefly, the advantages and disadvantages of each can be summarised thus:

BASIC
Easy to learn
Easy to remember
Easy to de-bug
Slow in execution
Uses lots of memory
Does not encourage structured programming

Assembly Language:
Not very easy to learn
Not very easy to remember