Simply Obeying Orders

Your computer will do exactly what you want when you 'talk' to it in the right way — and it won't make mistakes

Other Languages

BASIC is used by more people on more microcomputers than any other programming language. But BASIC is by no means the only one. Before the days of microcomputers, when most computing was done on room-sized mainframe computers, scientists and engineers used a language called FORTRAN. In the world of micros, other popular languages include PASCAL, FORTH, and LOGO

Pascal

Like BASIC, PASCAL was developed primarily as a teaching language for student programmers. It is held in high regard by teachers of programming because it encourages the writing of well thought out and elegant programs. PASCAL is usually supplied on floppy disk and tends to be expensive. A low-cost cassette version is available for the Spectrum for under £30. The PASCAL language can be used for writing large and sophisticated programs

Forth

Programs written in the FORTH language look much less like English than BASIC or PASCAL. FORTH is also more difficult to learn. It has the advantage of great power in the sense that complex programs can be written in a few lines. FORTH allows you to define your own commands, whereas in BASIC they are pre-defined

Logo

LOGO is a relatively new language becoming popular in education. It has the great advantage of being simple enough for even quite young children to learn. It can help teach programming techniques and also encourages a logical approach to program design from an early stage. LOGO uses 'turtle' graphics which allow pictures to be easily produced on the screen. A mechanical turtle (see page 34) can also be connected to the computer. Simple commands typed in on the keyboard can move the turtle and make it draw lines and shapes

It is perfectly possible for anyone to use a computer — at home or at work — without knowing anything at all about how the computer works. Starting here, THE HOME COMPUTER COURSE begins a step-by-step series that explains, from the beginning, all you need to know to be able to create your own computer programs successfully.

Many people find that after a while, the pre-packaged programs and games they've bought for their computer start to become a little boring and they wonder if they can modify them or even write their own. But a computer can do nothing by itself. It must be given a list of instructions telling it in minute detail exactly what to do and how to go about achieving it. These instructions form what is called a *program* and the art of creating them is called *programming*.

There is nothing difficult about programming. You don't even have to be good at maths, unless, of course, you want to write programs to perform mathematical tasks. All you need to begin with is to understand BASIC.

Your First Language

Most home computers are provided with a built-in computer language called BASIC. As its name implies, it is designed to enable beginners to learn the rudiments of programming quickly and easily. Like any human language, BASIC has it own grammar, vocabulary and syntax, although the vocabulary is far smaller than that of English. BASIC uses a number of short English words that are easily recognisable and simple to learn. As a general-purpose language it is suitable for both the novice and the more experienced user.

But one drawback with the language is that over the years, different computer manufacturers have tended to include their own modifications. The result is that there are a large number of variations in BASIC, particularly regarding the commands for controlling the more recently developed aspects of the machine — such as colour, graphics and sounds. Any variations of BASIC which occur in the most popular computers are shown in the 'Basic Flavours' box in each lesson.

Because of the variations in BASIC from computer to computer, it is nearly impossible to write a BASIC program of any complexity that will run on every computer. Fortunately, however, the language has a common core, which is usually the same in all machines. We'll start by concentrating on that core, and as the course progresses we will work steadily towards more complex programs.

The Initial Steps

Let's begin by writing a small program and seeing what happens. This one will show the computer apparently making a mistake. Switch on the computer and type in the program exactly as shown, including all the spaces. The <CR> at the end of each line is to remind you to hit Carriage Return. On your computer, this key may be labelled RETURN, ENTER or even ↵.

```
10 REM COMPUTERS NEVER MAKE
   MISTAKES<CR>
20 PRINT "TYPE IN A NUMBER"<CR>
30 INPUT A<CR>
40 LET A = A + 1<CR>
50 PRINT "I THINK THE NUMBER YOU TYPED
   WAS ";<CR>
60 PRINT A<CR>
70 END<CR>
```

After you have typed it all in, type LIST<CR>. The program you just typed should reappear on the screen. LIST is an instruction to the computer to 'print' a listing of the program in memory. If the program appeared on the screen properly after typing LIST, we could try to RUN it. If you make a mistake when typing in the program, don't worry. After you have LISTed the program, simply retype any line containing a mistake. Don't forget the line number. Try typing

```
25 REM HERE IS ANOTHER 'REM' LINE<CR>
```

and then LIST the program again. To get rid of the line, type the line number alone, followed by <CR>. When you are satisfied the program has been typed correctly, you can 'run' it by typing RUN<CR>. Try this and you should see on the screen:

```
TYPE IN A NUMBER
```

Go ahead and type a number. Try 7. (Use numerals — the computer won't recognise 'seven' as 7 unless we specially program it to do so.) If you typed in 7, the screen should look like this:

```
I THINK THE NUMBER YOU TYPED WAS 8
```

Did the computer really make a mistake, or was it simply obeying orders? If we look at the program line by line we can see what each instruction made the computer do. Here's the first line:

```
10 REM COMPUTERS NEVER MAKE MISTAKES
```