Fully Functional

There are built-in functions in Basic, which means that a lot of the programming has already been done for you. Knowing how to use them adds power to your computing

Suppose you wanted to calculate the square root of a number in one of your programs. There are a number of ways this could be done. The crudest and least satisfactory way would be to create a table of square root values and to use this to give you the value wanted for a particular number. You probably learnt how to do this in school. An alternative method is to use the square root 'function', built in to most versions of BASIC. Here, the arithmetic of the operation is taken care of by BASIC without the programmer having to worry about it. Let's see how it works:

```
10 REM THIS PROGRAM FINDS THE SQUARE ROOT
20 REM OF A NUMBER
30 PRINT "INPUT THE NUMBER YOU WANT TO"
40 PRINT "FIND THE SQUARE ROOT OF"
50 INPUT N
60 LET A = SQR(N)
70 PRINT "THE SQUARE ROOT OF ";N;" IS ";A
80 END
```

Type in this short program and see that it does indeed give you the square root of any number you type in. Let's look at the rules of how to use this 'square root' function.

A 'function' in BASIC is generally a command word (SQR in this case) followed by brackets that enclose the expression to be operated on. In this program, N is the number input from the keyboard. It is the number we want the square root of. Line 60 says 'let the square root of N be assigned to the variable A'. Line 70 prints out the value of A.

The expression inside the brackets is called the 'argument' of the function and does not always have to be a variable: it is equally possible to use actual numbers. Type this in and see what happens when you run it:

```
10 PRINT SQR(25)
20 END
```

You will see that this works just as well. Similarly, we can use more complex arguments inside the brackets. Try this one:

```
10 LET A = 10
20 LET B = 90
30 LET C = SQR(A+B)
40 PRINT C
50 END
```

This little program can be shortened by combining lines 30 and 40 like this:

```
10 LET A = 10
20 LET B = 90
30 PRINT SQR(A+B)
40 END
```

The way to think of functions is as short programs built in to BASIC that are available for the programmer to use at any time. Most versions of BASIC have quite a large number of functions as well as the facility of allowing the programmer to define new ones for use within a program. Later, we will see how this is done. Here we will look at a few more of the commonly available functions. They come in two varieties: numeric functions, in which the argument (the part inside the brackets) is a number, numeric variable or numeric expression, and string functions, in which the argument is a character string or expression made up from character strings. First we'll look at a few of the numeric functions.

Previously, on page 77, we used a program that calculated the number of tiles needed to tile a room. A small 'bug' in this program was that the answer could well involve fractions of a tile. 988.24 could represent a possible result of running this program. At times like this we want a way of rounding the answer to the nearest whole number. Whole numbers are referred to mathematically as 'integers' and one of the functions in BASIC will 'return' the integer part of any number. Here's how it works:

```
10 PRINT "INPUT A NUMBER CONTAINING A
   DECIMAL FRACTION"
20 INPUT N
30 PRINT "THE INTEGER PART OF THE NUMBER
   IS ";
40 PRINT INT(N)
50 END
```

If this program is run and the number you input is 3.14, the program will print on the screen:

```
THE INTEGER PART OF THE NUMBER IS 3
```

Of course, if we are dealing with tiles, we would then need to add 1 to the answer, to make sure that we bought more than the required amount, not less.

On other occasions, we may want to find the 'sign' of a number to see if it is negative, zero or positive. To do this, most BASICS incorporate a SGN function. Try this:

```
10 PRINT "INPUT A NUMBER"
20 INPUT N
30 LET S = SGN(N)
40 IF S = -1 THEN GOTO 100
```