Braving The Elements

Subscripted variables, unlike their simple counterparts, can contain any number of elements

In our earlier program for calculating the number of days to Christmas we encountered a new type of variable called a 'subscripted' variable. These differ from ordinary or 'simple' variables in that they can have any number of compartments or elements within the box. Simple variables recognise two letters or letters followed by a digit from 0 to 9 (some versions of BASIC allow whole words to be used as variable names). A, B, B1, C3 and R2 are all simple variables. Subscripted variables look like this: A(6), B(12) or X(20). The subscript is the number in brackets. The examples we have given would be read as: 'A sub six', 'B sub twelve' and 'X sub twenty'.

If we think of a simple variable as being a box with a name or label on it, we can think of a subscripted variable as a box containing a specified number of internal elements. If we want a variable with 12 elements, we create it initially using the BASIC DIM statement, like this: DIM A(12). Any letter of the alphabet may be used.

Assigning values to simple variables is straightforward, using either LET or INPUT statements, like LET A = 35, LET B1 = 365 or INPUT C3. Values can be entered in the elements of a subscripted variable in the same way. Let's see how we would assign values to a subscripted array. ('Array' is an alternative name for a set of subscripted variables.) For example:

```
10 DIM A(5)
```

creates a subscripted variable with five elements. We can now assign a value to each element:

```
20 LET A(1) = 5
30 LET A(2) = 10
40 LET A(3) = 15
50 LET A(4) = 20
60 LET A(5) = 100
```

To find out how these variables differ from simple variables, let's assign values to a few simple variables:

```
70 LET X = 5
80 LET Y = 6
90 LET Z = 7
```

Try entering all these on your computer and then check the contents of each variable using the PRINT command. Many of the statements in BASIC also function as commands. After you have entered the statements above, check them by LISTing them and then type RUN. Now you can type PRINT X<CR>. You should see 5 instantly displayed on the screen. Next type PRINT Y. The computer will respond to this PRINT command by displaying 6 on the screen. If you want to check the elements in the subscripted variable, type PRINT A(1) to find out the value of the first element in the array. The computer should respond by printing 5 on the screen. Try PRINTing the values of A(3) and A(5).

The important difference between subscripted variables and ordinary variables is that the subscript can itself be a variable. To see what this means, type PRINT A(X). The screen will respond with the figure 100. Why?

Look at the list you have typed in and check the value of variable X. It is 5. A(X) is equivalent to A (the value of variable X) and this is equivalent to A(5). Typing PRINT A(X) is therefore exactly equivalent to typing PRINT A(5). What value would you expect if you typed PRINT A(Y — X)? Before actually trying it, see if you can work out the answer.

Assigning Values

If there are only a few simple variables, the LET statement is the simplest way of assigning values to them. Subscripted variables may well have a large number of elements in the array, so let's see what the alternative methods of entering the values are:

```
10 DIM A(5)
20 PRINT "INPUT THE VARIABLES"
30 INPUT A(1)
40 INPUT A(2)
50 INPUT A(3)
60 INPUT A(4)
70 INPUT A(5)
```

This method is just as tiresome to type in as using LET statements, though it would certainly work. If we know exactly how many variables there are (in this case there are five) it is easier to use a FOR-NEXT loop, like this:

```
10 DIM A(5)
20 FOR X = 1 TO 5
30 INPUT A(X)
40 NEXT X
```

This program would expect five values to be typed on the computer keyboard when the program was run. The RETURN key would have to be pressed after each figure had been entered. If we know beforehand what the values in the variable are, it is