Organise Your Program

We sort a program using built-in functions to rearrange information

This week's program illustrates how relatively complex programs can be broken down into simple sub-programs or subroutines that can be written and tested separately.

Apart from the advantage of being separately testable, the use of subroutines allows the development of the program to follow a logical progression. There are many approaches to writing a program in BASIC. One of the commonest is 'trial and error': you sit down at the computer and start entering lines of BASIC without having thought out carefully what is required to make the program work. This method leads to badly structured programs that will often not work the first time. If the structure of the program is not clear, it is not easy to find the mistakes or 'bugs'.

A much better approach is to sit down with a notebook and work out the structure of the program first, in steps of ever-increasing refinement, until a correct and working program can be written. A flow diagram will also help (see page 104). Let's see how this is done.

Problem: Write a program that will input a number of names, the forename followed by surname. Now reverse the order of each name so that the surname comes first, followed by a comma and a space, and by the forename. It will then sort the names into alphabetical order and print them out.

For example, if the names BILL JONES and FRED ASHTON were entered (in that order), the program would print out:

```
ASHTON, FRED
JONES, BILL
```

Before even attempting to write a program to do this, first write down the desired input and output in the most general terms:

```
Step 1
Input names in random order, first name first
Output names in alphabetical order, surnames
first
```

This simply clarifies what we want to be done. This is an essential first step to a properly organised program. The next step is to refine the stages in the first step and make sure that the program still works. Do not, at this stage, get into too much detail. Simply write down in a little more detail, the stages involved:

```
Step 2
Find out number of names to be input
Enter names
Reverse names
Sort names
Print names
```

Look at the above list and check that it will work. Can you see anything wrong with it? Are there any flaws in the logic? If not, you are ready to go on to the next stage of refinement.

The stages we arrived at in Step 2 are small enough and simple enough to be written separately as small sub-programs. Sub-programs are called subroutines in BASIC. Let's give the subroutines names to make them easier to identify. Subroutine 1, to find out the number of names to be input, can be called FINDNUM. Subroutine 2, to enter the names, can be called ENTER. Subroutine 3, to reverse the names, can be called REVERSE. Subroutine 4, to sort the names, can be called SORT. Finally, Subroutine 5, to print the names can be called PRINTNAMES.

```
Step 3.1 FINDNUM
Prompt the user to input required number
Get the required number N
Use N to set up string array
Step 3. 2 ENTER
If number of names is less than N,
prompt user to input another name
Add name to string array
Step 3.3 REVERSE
Find length of string (name)
Find 'space' in string
Put characters in string up to 'space'
into temporary string variable
Put characters in string from 'space' to
end into another temporary variable
Add comma space to end of variable
Assign second followed by first
temporary variables to original string
Step 3. 4 SORT
Compare first item in array with next item
If first item is bigger than next one
(i.e. higher in the alphabet), swap
Compare second item with third
Swap, if necessary
Repeat until all pairs are compared
Go back to beginning of array and
repeat comparison of pairs until no swaps
have taken place
```

Note: This sort routine is exactly the same as the one used in the previous part of the Basic Programming course. The 'swap' part will be dealt with as a subroutine called from within the SORT subroutine.