Step 3.5 PRINTNAMES

Print each item in the array until all items have been printed

Each of the steps needed to build this program has now been worked out in a reasonable amount of detail. The SORT routine has only been sketched roughly since it was dealt with fully in the last instalment of the course. And SWAP, which is 'called' from within this subroutine, has been left out completely. Let's now see how easy it is to convert programs worked out in English into a program in BASIC.

Step 4
1. FINDNUM

The three lines in Step 3.1 translate directly into BASIC statements. The user is prompted by a PRINT statement, the number is found by using an INPUT statement and the array is dimensioned by using the DIM statement:

```
PRINT "HOW MANY NAMES DO YOU WISH TO
  ENTER?"
INPUT N
DIM A$ (N)
RETURN
```

The variable N now contains the maximum number of names to be entered. The DIM statement dimensions a string array. String variables contain strings of alphanumeric characters instead of numbers. A string variable name always ends with a 'dollar' sign. A$ alone could only contain one string. DIM A$ (N) creates an array that can contain 'N' strings. Subscripted variables have been dealt with earlier in the course.

The RETURN statement transfers control back to the main program at the line following the subroutine call. Values assigned to variables in the subroutine will be 'carried back' to the main program and can be used elsewhere in the program, even in other subroutines.

2. ENTER

As long as the number of names entered is less than N, the user needs to be prompted to enter a name and this name must be added to the string array. This calls for creating a FOR-NEXT loop; we know that the first name in the array will be its first element, and that the last one will be the Nth, so:

```
FOR X = 1 TO N
PRINT "ENTER NAME"
INPUT A$(X)
NEXT X
RETURN
```

That should suffice to enter all the names into the array. But sharp readers will have spotted what happens when we come to reverse the order of the first and last names in the REVERSE subroutine. Each element (name) in the array will have to be pulled out again, then reversed, and then put back in the array. Rather than complicate and lengthen the program by doing that, it would be simpler to call the REVERSE subroutine from within the ENTER subroutine after each name has been typed in. The name can then be reversed before it is assigned to the array. To do that, we just have to add one line, thus:

```
FOR X = 1 TO N
PRINT "ENTER NAME"
INPUT A$(X)
GOSUB [REVERSE]
NEXT X
RETURN
```

All the names in the array will now be in reversed order (surname first, followed by forename) and will therefore be ready for sorting.

3. REVERSE

To reverse the order of names, we need to know where the 'space' is separating the first name from the surname. When we know where the space is, we can use various functions to pull out parts of the string and assign those parts to other strings. Functions in BASIC are commands that perform a predefined operation on the value following the function name. This part is always in brackets. Many functions are 'built in' but it is also possible to define your own. A typical 'built in' function is SQR ( ). This function 'returns' the square root of the value inside the brackets. So: LET A = SQR (9): PRINT A will print a 3.

REVERSE uses the functions LEN (to find the length of the string), INSTR (to locate the position of the space), LEFT$ (to remove a specified number of characters from the left of the string) and

Programs Within A Program

The main program this time is very short. All the real work is done in the sub-programs (called subroutines in BASIC). Each of the steps needed to make the program work are separated and written as short 'mini-programs'. These are then simply linked together by the main program.

When the program is run, each time a GOSUB statement is encountered, the program branches to the specified subroutine line number and that section of the program is then executed. The end of the subroutine is indicated by the RETURN statement. On reaching this, the program returns to the point immediately after the GOSUB that called the subroutine.

Subroutines can be 'nested' within subroutines. The ENTER subroutine calls another subroutine called REVERSE, and SORT sometimes calls another subroutine called SWAP.

Breaking down a problem into separate subroutines linked by a simple main program makes the development and testing of programs far easier