BASICALLY BASIC

Graham Hall, B. Sc.

A regular series to teach BASIC for use at home and in business or scientific applications. No previous programming experience required. Continuation of our superb series from Electronics & Music Maker.

Subroutines

Often it is required to perform the same series of operations at different points in a BASIC program. This can be achieved by writing the statements as a subroutine within the main program. When the statements are to be executed the subroutine is referenced (or'called') by the main program. After the statements of the subroutine have been executed, control is automatically directed back to the main program statement that immediately follows the call to the subroutine.

The GOSUB and RETURN Statements

The GOSUB statement transfers control of the program to a subroutine and the RETURN statement returns control from that subroutine back to normal program execution.

The general form of the GOSUB statement is:

GOSUB line number

where the line number is the first line of the subroutine. The first line of a subroutine can be any legal BASIC statement including the REM statement. It is good programming practice to make the first line of a subroutine a REM statement followed by a comment which outlines the function of the subroutine.

The last statement of any subroutine must be a RETURN statement. The RETURN statement causes control to transfer back to the main program line following the GOSUB statement. The RETURN statement is not followed by a line number - when the subroutine is entered the computer 'remembers' the line which called the subroutine. The RETURN statement instructs the computer to use this value to return to the statement after the GOSUB.

The use of the GOSUB and RETURN statements to implement a subroutine is best illustrated by considering a BASIC program:

```
10 REM PROGRAM - SORT LISTS OF NUMBERS
15 DIM X(10)
20 PRINT"INITIAL LIST 1"
30 FOR 1=1 TO 10 40 READ X(I)
50 PRINT X(I);
60 NEXT I
70 REM CALL SUBROUTINE - SORT LIST ONE
80 GOSUB 600
90 REM NOW READ SECOND LIST OF NUMBERS
100 PRINT:PRINT"INITIAL LIST 2"
110 FOR 1=1 TO 10 120 READ X(I)
130 PRINT X(I);
140 NEXT I
150 REM CALL SUBROUTINE- SORT LIST TWO
160 GOSUB 600
170 REM NEXT TWO LINES ARE DATA
180 DATA -10,6,3,50,-7,0,41,32,9,15
190 DATA 18,-4,-50,61,11,18,22,-1,99,6
200 END
600 REM SUBROUTINE-SORT LIST IN ARRAY X 610 FOR 1 = 1 TO
9 620 FOR J=I + 1 TO 10
630 IF X(I)>X(J)THEN LET T=X(I):LET X())=X(J):LET X(J)=T
640 NEXT J
650 NEXT I
660 PRINT:PRINT''SORTED LIST"
670 FOR 1=1 TO 10 680 PRINT X(I);
690 NEXT I 700 RETURN
```

In this example, the subroutine begins at line 600 and ends with a RETURN statement at line 700. The subroutine uses a simple sort algorithm to sort the numbers stored in the array named X. It then prints the list of sorted numbers before returning control back to the main program.

The program is composed of the following lines:

Line 10 - The REM statement identifies the program. The characters following REM are ignored by the computer.

Line 20 - The PRINT statement outputs the message contained in quotation marks to the terminal.

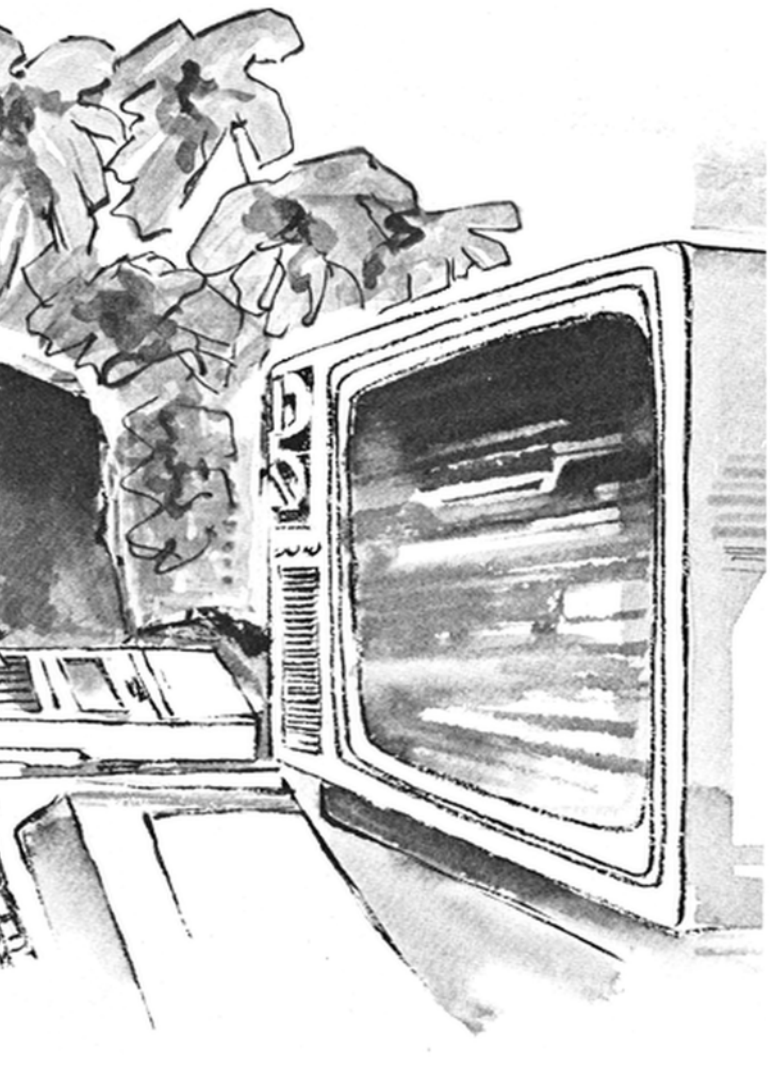
Lines 30, 40, 50 and 60 - The FOR statement initialises the variable I to one and sets the limit of the loop to ten, its corresponding NEXT statement is on line 60. Each time the loop is executed the READ statement on line 40 takes a number from the DATA statement in line 180 and assigns it to the array named X in the position specified by the subscript (the subscript is the variable I within the parentheses after the array name). The PRINT statement on line 50 outputs the number to the terminal. The semicolon after the array element directs the computer to print each number on the same line. When the loop has been executed ten times, program control proceeds to line 70. The array X now contains the ten numbers specified as data in line 180.

Line 80 - The GOSUB statement directs the program control to the subroutine beginning at line 600. The statements of the subroutine are now executed until the RETURN statement on line 700 is encountered. This RETURN statement directs the control back to the main program line following the GOSUB statement (line 90).

Lines 100, 120,130 and 140 - These statements are the same as lines 30-60 and read into the array X the next ten numbers from the data in line 190. (These statements could also have been written as another subroutine. The main program would then consist of four subroutine calls - first a call to a subroutine to read the numbers into the array and then a call to the subroutine to sort the numbers. Then these two subroutines would be called again to repeat the process for the next set of data.)

Line 160 - The GOSUB statement accesses the same subroutine on line 600 as referenced by line 80. However, when the RETURN statement of the subroutine is executed this time, control shifts to line 170 because the GOSUB at line 160 was the last statement in the main program to be executed.

Lines 180 and 190 - These DATA statements contain the values that are provided for the READ statement.



Line 200 - The END statement signifies the finish of the main program. The statements after the END statement are those of the subroutine which sorts the array.
Line 600 - This is the first statement of the subroutine. Note that it is a REM statement followed by a comment which explains the function of the subroutine. It is given a high line number to make it easier to distinguish between subroutine statements and those of the main program.
Lines 610, 620, 630, 640 and 650 - These statements perform the sort operation on the array X. The IF THEN statement on line 630 compares adjacent array elements, if the first element is greater in magnitude than the second, the elements are exchanged. In order to perform this exchange the original value of the first element is stored in the variable T. The value of the second element is then put into the first element. Now the original value of the first element remembered in T, is assigned to the second element. If the comparison is false the exchange is not performed.

The FOR loop initialised on line 630 is between the FOR and the NEXT statements of the loop initialised on line 620. This is a 'nested loop'. The loop index variables I and J are used as subscripts to reference the elements of the array X. Initially I is one and J is initialised, to two (I + 1). Thus the first element of the array is compared with the second by the IF statement on line 630 and exchanged if it is greater in magnitude. Now line 640 increments J by one to three and the first element of the array, which may have originally been the second element depending on the result of the first comparison, is compared with the third element and exchanged if necessary. This continues until J reaches ten at which point the first position in the array contains the smallest number of the list. Now I is incremented to two and J is initialised at three. The second element of the array is compared with the third and exchanged if greater in magnitude. This comparison and exchange process continues for each element in the array. When I reaches nine the elements of the array have all been compared with each other and arranged into ascending numeric order.
Line 660 - This starts a new line and outputs a message to the terminal.
Lines 670, 680 and 690 - The FOR loop index I is used to reference each element of the array which is printed by the PRINT statement on line 680. This is similar to lines 30-60 except that new data is not read in.
Line 700 - The RETURN statement redirects the program control back to the statement in the main program immediately following the GOSUB statement which called the subroutine.

When the program is run the output to the terminal will be:

```
INITIAL LIST 1
-10 6 3 50 -7 0 41 32 9 15
SORTED LIST
-10 -7 0 3 6 9 15 32 41 50
INITIAL LIST 2
18 -4 -50 61 11 18 22 -1 99
6 SORTED LIST
-50 -4 -1 6 11 18 18 22 61 99
```

The use of subroutines enables programs to be constructed in a modular way. This makes testing and de-bugging easier, since each subroutine can be individually tested using a test program, before it is combined with the main program.

Nested Subroutines

A 'nested subroutine' call is a call to a subroutine from within a subroutine. Subroutines can be nested to several levels (the limit on the level of nesting is dependent on the version of BASIC being used). Each GOSUB must have a corresponding RETURN statement. The following simple program illustrates how subroutines can be nested:

```
10 REM NESTED SUBROUTINES
20 GOSUB 200
30 PRINT "LINE 30"
40 END
200 PRINT "SUBROUTINE 1"
210 GOSUB 500
220 RETURN
500 PRINT "SUBROUTINE 2"
510 GOSUB 800
520 RETURN
800 PRINT "SUBROUTINE 3"
810 RETURN
```

The GOSUB statement on line 20 calls the subroutine starting at line 200, line 210 prints out the message identifying the subroutine and then the GOSUB on line 210 calls the second subroutine. This subroutine prints a message and then calls the third subroutine. The RETURN statement on line 810 directs the program to line 520, the statement immediately following the GOSUB which called the third subroutine. Line 520 is a RETURN statement which directs program control back to line 220 which in turn directs control back to line 30 of the main program. The output when the program is run would be:

```
SUBROUTINE 1
SUBROUTINE 2
SUBROUTINE 3
LINE 30
```

The ON GOSUB Statement

The ON GOSUB statement is similar to the ON GOTO statement described previously, in that it is used to implement a semi-conditional branch. The ON GOSUB statement has the general form:
Line number ON variable or equation GOSUB list of line numbers
For example, 10 ON X GOSUB 600, 800, 1000
Based on the integer value of the variable X, the program branches to one of the list of alternative subroutines. The selection is made in numeric order. So, for the above example:
if X is 1 - the program branches to the subroutine at line 600
if X is 2 - the program branches to the subroutine at line 800
if X is 3 - the program branches to the subroutine at line 1000
When the RETURN statement of the subroutine which is referenced is reached, the program control is directed back to the statement in the main program which immediately follows the ON GOSUB statement.

The number of alternative lines to be branched to is only limited by however many can be written on one program line. If the value of the variable or equation is zero, negative or larger than the number of alternative lines, program execution will stop and an error message will be output by the computer.