BASICALLY BASIC

Graham Hall, B. Sc.

BASIC Functions

One of the features of the BASIC language which makes it suitable for such a wide range of applications, is the set of pre-written instructions that can perform commonly used operations. These instructions are called 'Functions'. The functions to be described are the more common ones available on most personal computer systems. The list is not exhaustive, and the way in which they are used in the particular version of BASIC used by your machine may be slightly different from the description here. You should check the users reference manual, for your computer for the complete description of functions available in your machines version of BASIC.

There are three types of functions: math, print and string. Math functions perform mathematical operations, print functions cause operations on terminal output and string functions perform operations on quoted strings or string variables. String functions will be described later in this series.

Math Functions

There are two types of math function - arithmetic and trigonometric. Table 1 lists and summarises the common math functions which are fully described below.

ABS Function

To use the ABS (Absolute value) function the function keyword is followed by an argument enclosed in parenthesis, on a statement line. The result returned is the magnitude, regardless of sign, of the argument. For example the absolute value of both +15 and -15 is 15. The following program demonstrates this:

```
10 PRINT ABS(15), ABS(-15), ABS(0-10)
20 END RUN
15 15 10
```

The argument of the function can be any valid BASIC expression, variable or constant. The ABS function is useful for scientific applications where the magnitude of a number is required and not its sign.

EXP Function

The EXP (Exponential) function computes the value of the mathematical constant 'e' raised to the power of the numeric argument specified to the function within parentheses. The constant 'e' is the base of natural logarithms (to six significant digits, e = 2.71828). For example, to raise e to the power of two the EXP function can be written in a program as: 10 PRINT EXP(2). The inverse of the EXP function is the LOG function, which computes the logarithm to the base e of an argument (this is known as the 'natural logarithm'). This relationship can be demonstrated by combining the EXP and LOG functions:

```
10 PRINT "EXP (2) = " : EXP(2)
20 PRINT "LOG(2) = " ; LOG(2)
30 PRINT "LOG (EXP(2)) = '' ; LOG (EXP(2))
40 PRINT "EXP (LOG(2)) = " ; EXP (LOG(2))
50 END RUN
EXP(2) = 7.38906 LOG(2) = 0.69315 LOG(EXP(2)) = 2 EXP(LOG(2)) = 2
```

Lines 30 and 40 of the above function program are the inverse of each other. They illustrate how the argument to a function can be another function. The EXP function is useful for scientific applications.

INT Function

The argument to the INT (Integer) function can be any valid constant, variable, function or expression. The value returned is the largest integer less than or equal to the argument, i.e. the integer part of the argument is separated from the fractional part. For example:

INT (3.142) = 3, INT (0.69) = 0 and INT (-4.15) = -5

Note the value returned for a negative argument is a greater negative integer.

LOG Function

The LOG (Logarithm) function computes the natural logarithm 24 (logarithm to the base e) of its argument. The argument can be any positive constant, variable, function or expression which evaluates to a positive number. An error message will be printed if the argument is equal to or less than zero.

Some versions of BASIC also have a logarithm function which determines logarithms to the base ten (common logarithm). However, if your version of BASIC does not include this facility it is possible to convert the arguments natural logarithm to another base by dividing the natural logarithm of the argument by the natural logarithm of the new base. For example to find the logarithm of five to the base ten (common logarithm):

```
10 PRINT "LOGARITHM TO BASE 10 of 5 = " ;LOG(5)/LOG(10)
20 END RUN
LOGARITHM TO BASE 10 of 5 = 0.69897
```

RND Function

The RND (Random) function generates a random number between zero and one (but not including zero or one). The argument of the RND function can be any positive integer, zero or negative integer. The way the argument determines the operation of the RND function differs for different versions of BASIC. Usually, any positive integer within parentheses gives a new random number each time the RND function is used in a program. With zero or a negative integer specified, the same random numbers are generated. This is useful for debugging a program because if the numbers generated by the RND function is varied for each execution, program errors would be difficult to find. Some versions of BASIC do not require the RND function to be specified with an argument. In this case a RANDOMISE statement is included in a program before the RND statement, if different random numbers are required for each program execution. The random numbers generated by the RND function can be modified by expressions involving other BASIC functions and operators. For example, to generate a random integer between one and one hundred:

```
10 PRINT INT(100 * RND(l)) + 1
```

The argument of the INT function is '100 * RND(l)' which is an expression consisting of a constant, 100, multiplying a random number generated by the RND function. The result of this combination of functions is a random integer between zero and ninety-nine. The range, is adjusted from one to a hundred by adding the constant one to each number generated.

The RND function is especially useful for games programs and simulations.

SGN Function

The SGN (Sign) function returns a result which depends on the sign of the argument. The argument can be any valid constant, variable or expression. If the argument evaluates to a positive value the SGN function returns a 1; a zero would be returned if the argument evaluates to zero; otherwise a -1 is returned when the argument evaluates to a negative value. An example of this is:

```
10 ON SGN (x) + 2 GOTO 200,300,400
```

Here the constant two is added to the result returned by the SGN function. The result is now a positive integer (1,2 or 3) which is used as an argument to the ON GOTO statement to make a branch.

If X is negative, the result of SGN (X) +2 is 1, which directs program control to line 200. If X is 0, the result of SGN (X) +2 is 2, which directs program control to line 300. A branch is made to line 400 when X is positive because then the result of SGN (X) +2 is 3.

SQR Function

The SQR (Square Root) function determines the square root of its argument. The argument can be a positive constant, variable or expression. An error message will be returned if the argument specified is negative. However, some versions of BASIC convert a negative argument to its absolute value and return the square root of the converted value. For example, PRINT SQR(4) will output a 2 to the terminal. The command PRINT SQR(-4) will cause an error message to be output unless the absolute value of the argument is taken by the SQR function.

Trigonometric Functions

ATN Function

The ATN (Arctangent) function is a trigonometric function that accepts a numeric argument. The ATN function computes and returns the principal value of the arctangent of the argument in radians (or angular measure). This will be within the range: $-\pi 2 < ATN(x) < \pi / 2$

The relationship between the radian and degree is:

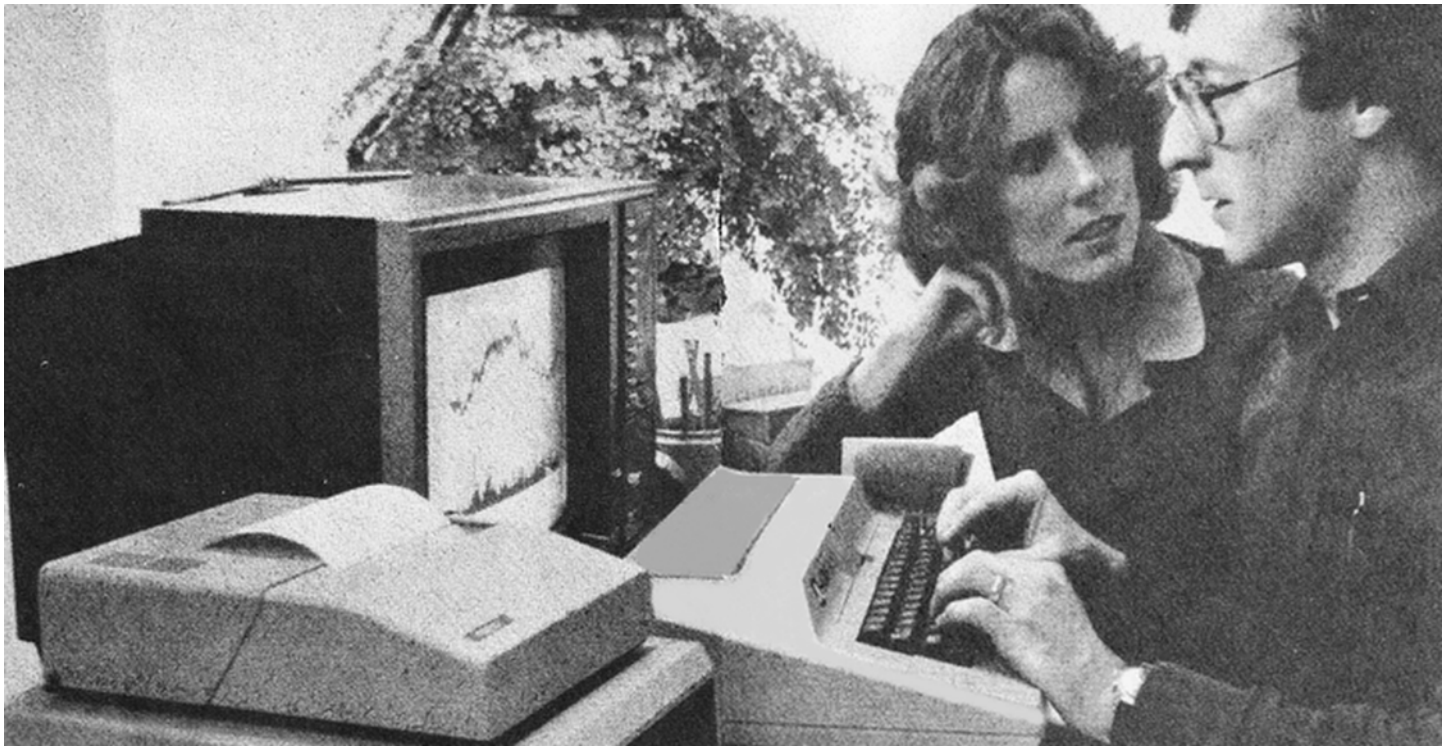
1 degree = π / 180 radians, where π represents the circular constant 3.1415927.

To convert the result of the ATN function from radian measure to degrees the value returned is multiplied by 180 / π (i.e. 57.2957795). For example, the command PRINT ATN (10)

prints the arctangent of 10 in radians. The command

```
PRINT ATN (10) * 57.2957795
```

will print on the terminal the arctangent of 10 in degrees.



ABS(X) Returns the absolute value of X.
ATN(X) Returns the arctangent of the value X in radian measure.
COS(X) Returns the cosine of the radian value X.
EXP(X) Returns the constant V (2.72828) raised to the power of X. INT(X) Returns the largest integer value of X.
LOG(X) Returns the natural logarithm of the value X.
RND(X) Returns pseudo random numbers.
SGN(X) Returns an indication of the sign of X.
SIN(X) Returns the sine of the radian value X.
SQR(X) Returns the square root of X.
TAB(X) Positions output to the terminal beginning at column X on the output line.
TAN(X) Returns the tangent of the radian value X.

Table 1. Common Math and Print Functions.

The ATN Function is the inverse of the TAN (Tangent) function. This is shown by the following short program:

```
10 PRINT “ATN(10) = ” ; ATN(10)
20 PRINT “TAN(ATN(10)) = ” ; TAN(ATN(10))
30 END
```

The expression TAN(ATN(10)) will evaluate to 10 because of the inverse relationship between the ATN and TAN functions.

COS Function

The COS (Cosine) function requires an angular argument in radian measure and returns the cosine of the angle. To convert an angle from degrees to radian measure so that it can be used as an argument to the COS function, the angle in degrees is multiplied by the ratio π / 180 (i.e. 0.0174533).

For example, the command PRINT COS (45 * 0.0174533) will print the cosine of the 45 degree angle.

SIN Function

The SIN (Sine) function requires an angular argument in radian measure and returns the sine of the angle. The same method as described above can be used to convert an angle from degrees to radians. For example, the command PRINT SIN(45* 0.0174533) will print the sine of the 45 degree angle.

TAN Function

The TAN (Tangent) function requires an angular argument in radian measure and returns the tangent of the angle. This is the inverse of the arctangent (ATN) function previously described. For example, to print the tangent of a 30 degree angle the command PRINT TAN (30 * 0.0174533) can be used.

Print Functions
TAB Function

The TAB (Tabular) function is used to position output on the terminal. The argument that the TAB function is numeric and moves the start of printing to the specified column. For example, the command PRINT TAB (20); "MESSAGE" outputs the string MESSAGE beginning at the twenty-first column. On most personal computer systems an output line is divided into 72 columns numbered from 0.

User Defined Functions

BASIC allows the programmer to name and formulate a function using the DEF (Define) statement. The DEF statement has the following format:

line number DEF FN n (x) = expression

For example, you may wish to define a function which would return the area of a circle given the radius. Such a function would be;

10 DEF FN A (x) = 3.1415927 * X ^2 (since area of a circle = π x radius squared).

The function name consists of three letters. The first two F and N must always be present but the last can be any letter from A to Z. This allows defined functions in one program (although some systems allow a letter immediately followed by a single number as a function identifier).

The argument to the function is the ‘dummy variable’ within the parenthesis (x). This reserves memory space for the arguments given to the function later in the program. Any legal variable name can be used as a dummy variable.

The expression on the right hand side of the equal sign is the calculation the function is to perform. The following program demonstrates how the function would be defined and then used to calculate the area of ten circles:

```
10 REM - CALCULATE AREA OF CIRCLES OF RADIUS 7-10 cm
20 DEF FN A (x) 3.1415927 * X ^ 2
30 PRINT TAB (5); "RADIUS (CM)’’; TAB (20); "AREA (CM 2)”
40 FOR X = 1 to 10
50 PRINT TAB (8); X; TAB (22); FN A (x)
60 NEXT X
70 END
```

The program is composed of the following lines:

Line 10 - The REM statement outlines the programs function. The characters following REM are ignored by the computer.

Line 20- This is the function definition. The function is named FN A and is equal to the expression π X 12, where X is the functions argument. It is advisable to place all function definitions at the beginning of a program so that the program is easier to read and follow.

Line 30 - The TAB function is used to format the output to the terminal. The ,heading ‘RADIUS (CM)' will begin at the sixth output line column and ‘AREA (CM 2)' will begin at the twenty-first column.

Line 40-The FOR statement initialises the loop index to one and sets the range to ten.

Line 50 - The PRINT statement outputs the value of X (underneath the heading ‘RADIUS (CM)’ because of the formatting of the TAB function) followed by the value of the area of the circle of radius X computed by the FN A function defined on line 20.

Line 60 - The next statement increments the value of X by one and directs program execution back to line 40.

Line 70 - The END statement signifies program completion.