Topic: Basic Function

A function is a structure that simplifies a complex operation into a single step. A function is therefore a many-to-one (or sometimes one-to-one) relation. The set of values at which a **function** is defined is called its domain, while the set of values that the **function** can produce is called its range. Functions act as 'black boxes', they accept input, value or values and process them in a defined manner to produce or return an output value. As long as you know how and when to use a particular function, you need not to be bothered about how it actually works.

Consider, for example the process of taking the square root of a number value. If you have to define this program, anytime you need a square root, your program would require extra code and you would probably get tired of entering the same code over and over again.

True Basic has a built in function that compresses the entire square root operation into a single step. Using the SQR function, your program can easily find the square root of any number greater than or equal to zero.

BASIC Programming

In 1964, John G. Kemeny and Thomas E. Kurtz designed the original **BASIC language** at Dartmouth College in New Hampshire.

B – Beginners

A – All purpose

S – Symbolic

I – Instructional

C – Code

Basic Arithmetic Expressions

Basic arithmetic expressions and algebraic expressions are similar with little different examples of basic arithmetic expressions. Arithmetic expressions are

composed of a combination of constants, variables, operation symbols, and functions. An expression may be very simple or quite complex, but it will result in a single value. Whether an expression is simple or complex, the calculations must be performed in a specific order. To ensure the computer will correctly evaluate and calculate arithmetic expressions, you have to learn to code them using the rules of BASIC. In order to use arithmetic expressions efficiently, you must be able to evaluate and convert conventional mathematical expressions into proper BASIC expressions.

I = {P*R*T}/100

Basic arithmetic expressions and their algebraic expressions

ALGEBRAIC EXPRESSION	BASIC ARITHMETIC EXPRESSION
B + D	B + D
A – B	A – B
B C	B / C
D^2	D ^2
C + B÷D	C + B* D
B = B x H	B – B* D
P = ax – bx + c/2	P = a*x – b*x + c/2
$a(b-d)^3 \div d+1$	$a* \{b-d\}^3/d+1$
$B - \frac{(-a+b)}{2a}$	B = {-a+b}/2*a
$b^2 – 5bc -3b$	SQR {b} ^ c^{-5}* b* – 3b
$\underline{(b +c)}$ +d Sin b	{b+c} + d/Sin {b}

Sign	**Arithmetic Expression**	**Basic Expression**	**Name**
{ }	14 + 2	{14 + 2}	Bracket

Of	½ of 5	5 * 2	Exponential
/	24 ÷ 4	24/4	Division
x	6 x 5	6 * 5	Multiplication
+	2 + 3	2 + 3	Addition
–	3 – 2	3 – 2	Subtraction
√	√16	SQR {16}	Square root

Parentheses Rule

There are cases where the precedence rule may cause a problem. For example:

The BASIC expression LET Y = A/B+ C would produce undesired results, because A would be divided by B and the result added to C. The solution to this problem is in the use of *parentheses.* If we let parentheses override the order of precedence (but maintain the order of precedence within the parentheses), the result will be satisfactory. Now let's examine the previous example using *parentheses.*

Example:

Using parentheses

With the parentheses, B is added to C and the sum of this operation is divided into A, giving the correct result.

Sometimes more than one set of parentheses may be needed to tell the BASIC language in what order to execute the arithmetic operations.

Example:

Parentheses inside parentheses

The BASIC expression to accomplish this would be:

For this expression to give us the correct results, A and B must be added first, then the sum divided by C, and finally that result is squared. When parentheses within parentheses are used, the innermost parentheses will be evaluated first.

Addition has a lower precedence than either division or exponentiation; therefore, A + B must be in the inner parentheses. Division has a lower precedence than exponentiation, so (A+B)/C also must be enclosed in parentheses, ((A+B)/C), to ensure it is performed next.

The important thing to remember is the parentheses maybe used to over-ride the normal order of precedence. The *parentheses rule* says:

. Computations inside parentheses are performed first.

. If there are parentheses inside parentheses, the operations inside the inner pair are performed first.

SUMMARY

Simple problems can be solved with BASIC by using only two or three instructions. These are the END, PRINT and LET statements. To use these effectively, you must know how they work and what rules must be followed in using them.

The END statement, which must be the last statement in every BASIC program, has two functions. It indicates to the compiler that there are no more BASIC statements for it to translate and it terminates execution of the program.

The PRINT statement is used to instruct the computer to output something either on the terminal or the printer. The standard print line in BASIC is divided into print zones or fields of 16 spaces each.

The two punctuation marks used in PRINT statements are the comma and semicolon. A comma used as a separator in a PRINT statement causes standard spacing and a semicolon causes packed spacing.

Information enclosed in quotation marks in a PRINT statement will be printed exactly as it appears in the program.

The LET statement can be used to assign a constant value to a variable name, a variable to a variable name, or the results of an expression to a variable name. The equal sign in a LET statement does not indicate algebraic equality, rather it means be assigned the value of. The value assigned by a LET statement is stored in the computer's memory; therefore, it can be referenced by its variable name.

Both the PRINT and LET statements may contain expressions with arithmetic operations. These arithmetic operations must be specified by the appropriate operation symbol. Should you forget to include the symbol, the computer will not insert it for you, but will give you an error message.

Constants and variables are used to refer to numeric values or character strings. A constant is a whole or decimal number or character string whose value does not change. A variable name is an arbitrary name you select and you and the computer use to refer to a value stored in the computer's memory. This value may vary during execution of the program, but can contain only one value at a time.