QBASIC

Introduction

- BASIC (Beginners All Purpose Symbolic Instruction Code) is one of the easiest high level language.
- BASIC was developed in 1964 by Professors John Kemeny and Thomas Kurtz.
- QBASIC (Quick Beginners All Purpose Symbolic Instruction Code) is a high level programming language published by Microsoft in 1991 AD.

Features Of QBASIC

- It is simple and easy to learn.
- It allows us to write and run programs immediately.
- It allows us to break lengthy programs into modules.
- Supports structured programming.
- Debugging can be easily done.

Elements of QBASIC Programming

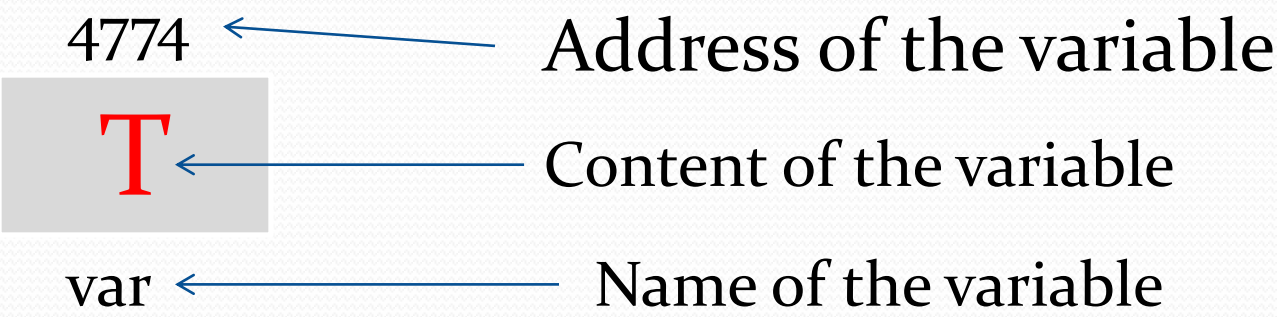
1) Character Set
2) Variable
3) Constant
4) Operator and Expression
5) Keywords (Reserved Words)

Character Set

- Character set is a set of valid characters that a language can recognize.
- A character represents any letter, digit, or any other sign.
- QBASIC has following character set:
- Alphabets : A to Z (small and capital)
- Numbers : 0 to 9
- Special Characters : + - * / = ^ % # ( ) ! : & ; < > $?

Variable

- Variables are the storage locations in the computer's memory.
- The data stored in a variable is the variable's value.
- Variable's value change during the execution of program.



Types of Variable

1. Numeric Variable : The numeric variable has a number as its value . It must begin with an alphabet and reaming characters may be alphabet or digits or both . Examples : area,a1,L1,L2,etc.
2. String variable : The string variable has a string of character or alphanumeric as its value . It must begin with an alphabet and end with a dollar($) sign. Example : BOOK1$,A$,a$,etc.

Constant

- Constant is a data item whose value does not change during execution of a program.
- It is also called literal.
- There are two types of constant :

1) Numeric constant
2) String constant

1) Numeric Constant : It is a sequence of positive or negative numbers on which mathematical operations can be performed. Examples : 23,-5,80,etc.
2) String Constant : It is a sequence of characters which may include numbers , letters and certain characters enclosed in quotation marks.

Operator and Expression

- Operators : operators are the symbol, which are used in arithmetic operation, logical expression and string expression. It helps to convert one or more values into a single value.
- Expression : A combination of an operator and its operands is referred to as an expression.

Sum=21+6

Since (21+6) has a value, it is an expression. Its value 27, is stored in the variable sum.

Expression do not have to be in form of mathematical operation. Example: number=3, 3 is expression.

Types of Operators

1) Arithmetic operators : The operators that operate on numeric constants and variables are called Arithmetic operators. They are used to perform various mathematical operation. Examples: +,-,*,/,Mod,^
2) Relational operators : They are used to compare two values of same type , either both numeric or both string. Example: =,<,>,<=,>=,<>
3) Logical operators : They are used to connect two or more relational expressions to evaluate a single value as True or False. Example: AND,OR,NOT

Keywords (Reserved Words)

- Keywords are those words which have special meaning in QBASIC.
- They are reserved for special purpose and must not be used as normal identifiers names.
- Some of keywords are REM, CLS , INPUT, LET, PRINT and END

QBASIC Statements

- QBASIC statement is a meaningful expression or an instruction in a source language.
- These statements are stored in the memory and executed only when the command RUN is given.
- It is either executable(tell what BASIC to do during execution of program) or non-executable (do not cause any program action).
- The statements can be divided into four categories : declaration statement, assignment statement, input/output statement and control statement.

Some QBASIC statements

- REM statement : Rem statement is a non-executable statement which is used include explanatory remarks to be inserted in program.
- It is useful to explain what a program does and what specific lines of code do.
- The general format of REM statement is :
- REM <remarks> or ‘<remarks>

CLS statement

- CLS statement is used to clear the output screen.
- It is generally given before the start of any program so that there is a fresh screen and left over from previous program is clear completely.
- The general format of CLS statement is: CLS

LET statement

- LET is an assignment statement used to assign the value of an expression to a variable.
- It is an optional keyword i.e., the equal sign is sufficient when assigning to a variable name.
- The general format of LET statement is :

LET <variable> = <expression>

Example: LET distance =300 , Length=3

INPUT statement

- It is used to accept the input from the keyboard during program execution.
- It facilitates the use of same program for various set of data to obtain different results in different execution.
- The general format of INPUT statement is:

INPUT [“definer”;/,] list of variables

Example: INPUT “Enter length of rectangle”;len

QB64 x64

File Edit View Search Run Debug Options Tools

```
1  Rem "This program calculates the area of circle"
2
3  PI = 3.14159
4
5  Let R = 8
6
7  ' Calculating the area of circle
8
9  Area = PI * (R ^ 2)
10
11 'Displaying area of circle
12
13 Print "The area of circle is :"; Area
14
15 End
```

READ statement

- It is used to read values from DATA statement and assign them to variables.
- The purpose of DATA statement is to store the numeric and string constants that are accessed by the program's READ statement.
- The general format of READ…DATA is:

READ [variable1,variable2,…..]

….

…..

DATA [constant1,constant2,….]

CONST statement

- CONST statement is a non-executable statement that declares symbolic constants to be used in place of numeric or string values.
- The general format of CONST statement is :

CONST constantname = expression

Example: CONST pi=3.141

PRINT statement

- It is used to display data on the screen.
- It prints constants, variables or expression.
- A question mark(?) can be used instead of the word PRINT.
- The general format of PRINT statement is:

PRINT [list of expression] [,|;]

or

? [list of expression] [,|;]

Example: PRINT “Sum of two numbers=”;sum

TAB function

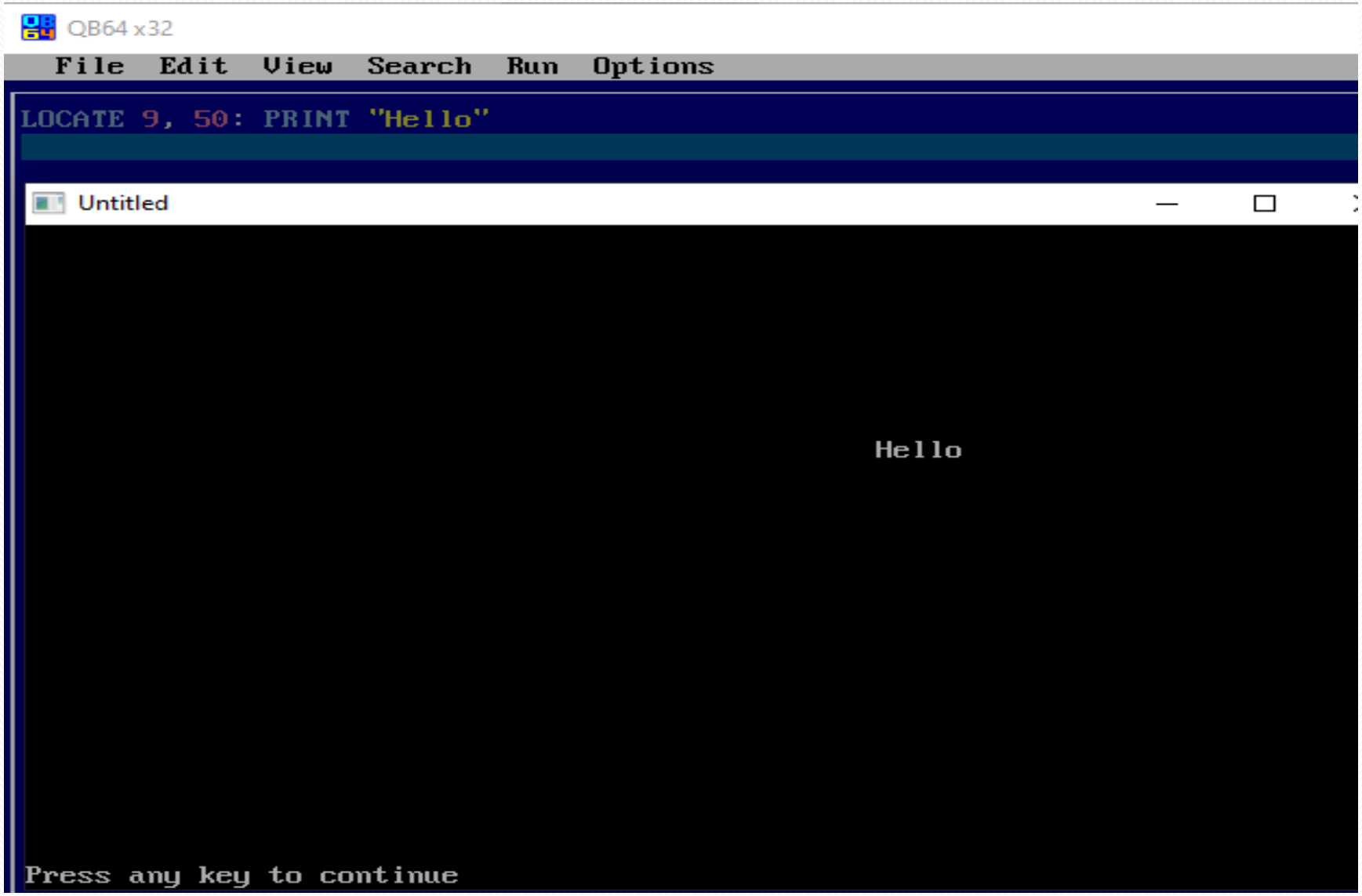
- It is used to space the position.
- It is only used in PRINT,LPRINT and PRINT# statement.
- The general format is : TAB(n)

LOCATE statement

- It is used to move the cursor to the specified position on the screen and determine the height of the cursor.
- The general format is:

LOCATE [row][,[col][,[cursor][,[start][,stop]]]]

Example: LOCATE 9,50 : PRINT ”Hello”


QB64 x32
File Edit View Search Run Options
LOCATE 9, 50: PRINT "Hello"
Untitled
Hello
Press any key to continue

QB64 x32

File Edit View Search Run Options

```
REM"WAP TO CALCULATE THE AREA OF TRIANGLE"
CLS
INPUT "Enter the height"; h
INPUT "Enter the breadth"; b
area = 1 / 2 * (h * b)
PRINT "The area of triangle is"; area
END
```

Untitled

```
Enter the height? 4
Enter the breadth? 5
The area of triangle is 10
```

File Edit View Search Run Options

Untitled*

```
REM"Write a program that converts centigrade temperature into Fahrenheit temperature. Where f=(9/5)*c+32"
CLS
INPUT "Enter the temperature in centigrade"; c
f = c * (9 / 5) + 32
PRINT "The temperature in fahrenheit is"; f
END
```

Untitled

```
Enter the temperature in centigrade? 5
The temperature in fahrenheit is 41
```

QB64 x32

File Edit View Search Run Options

```
CLS
INPUT "Enter first number"; a
INPUT "Enter second number"; b
q = a \ b
PRINT "Quotient="; q
res = a / b
print"Result=";res
r = a MOD b
PRINT "Remainder="; r
END
```

Untitled

```
Enter first number? 5
Enter second number? 2
Quotient= 2
Result= 2.5
Remainder= 1
```

```
CLS
INPUT "Enter first numbers"; a
INPUT "Enter second numbers"; b
sum = a + b
diff = a - b
product = a * b
div = a / b
PRINT "sum="; sum; "Difference="; diff; "Product="; product; "Division="; div
END
```

Untitled

```
Enter first numbers? 5
Enter second numbers? 2
sum= 7 Difference= 3 Product= 10 Division= 2.5
```

END statement

- It denotes the end of program.
- It must be written as the last statement of the program.
- The general format of END statement is :

  END

Control statement

- Computer program executes its statement from beginning to end in sequential order. This type of program is called in-line program.
- But some programs do not execute all their statements in strict order from beginning to end.
- The flow of jump control jumps from one part of program to another, depending upon the calculations performed in the program.
- Program statement that cause jumps are called control statements.

Types of control statement

1) Branching statement: It allows program to transfer the control to some other specified statement instead of the sequential execution.

Types of Branching statement:

a) Unconditional Branching statement :It is statement that transfers the control unconditionally (without testing any condition) from one statement to another statement in program. GOTO statement is the simplest unconditional branching statement.

GOTO statement

- It is used for unconditional transfer of execution from one part of program to other. It does not depend on any test of condition.
- The general format is :

GOTO [line number | line label]

QB64 x64

File Edit View Search Run Debug Options Tools

```
x = 1
top: 'A label is a name followed by colon.
y = x * x
Print y
x = x + 1
GoTo top
End
```

2) Conditional Branching: It is a statement that allows the selective execution of statements based on a particular condition. On the basis of a given condition a selected segment of a program is executed.

It is also called decision-making statement.

The conditional branching include IF…THEN and SELECT CASE statement.

IF…THEN statement

- It is used for making decisions as well as comparison.
- It allows branching depending upon the value of an expression.
- The different form of IF…THEN statement are:

1)IF…THEN statement

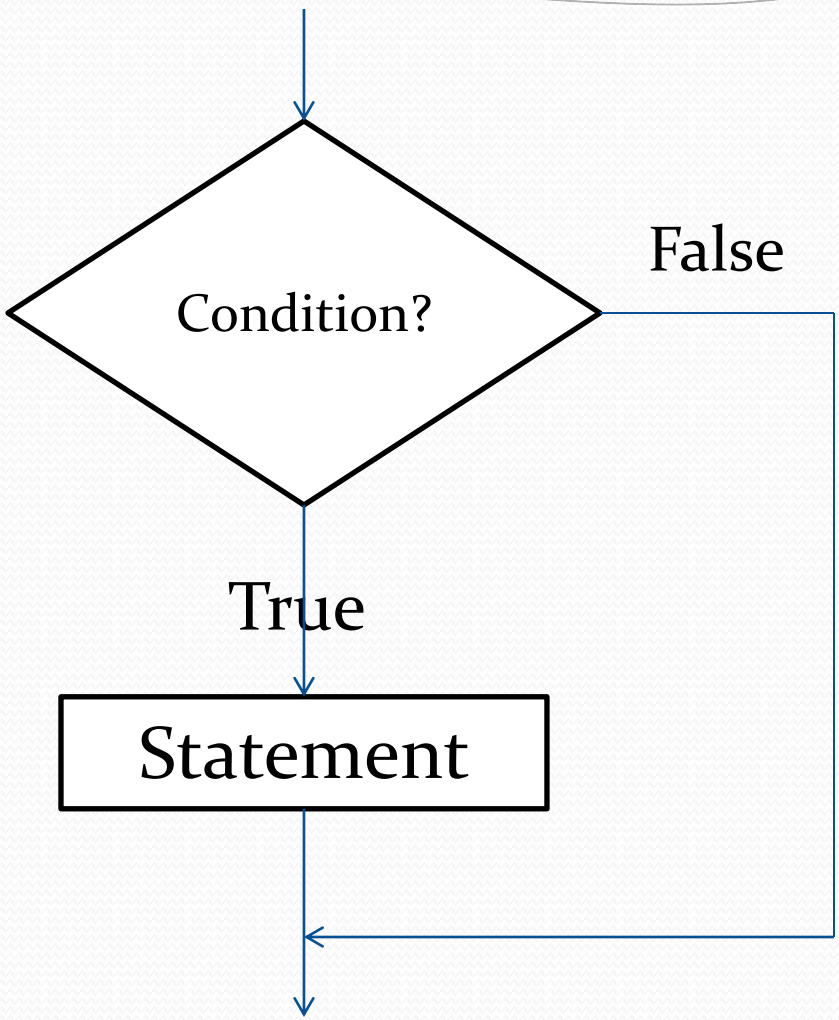
2)IF…THEN…ELSE statement

3)IF…ELSEIF…ENDIF statement

1) IF…THEN statement

- It is a conditional branching statement. If the condition is true, the statement given next to THEN will be executed otherwise next executable statement following IF…THEN statement will be executed.
- The general format is:

IF<condition> THEN <statement>

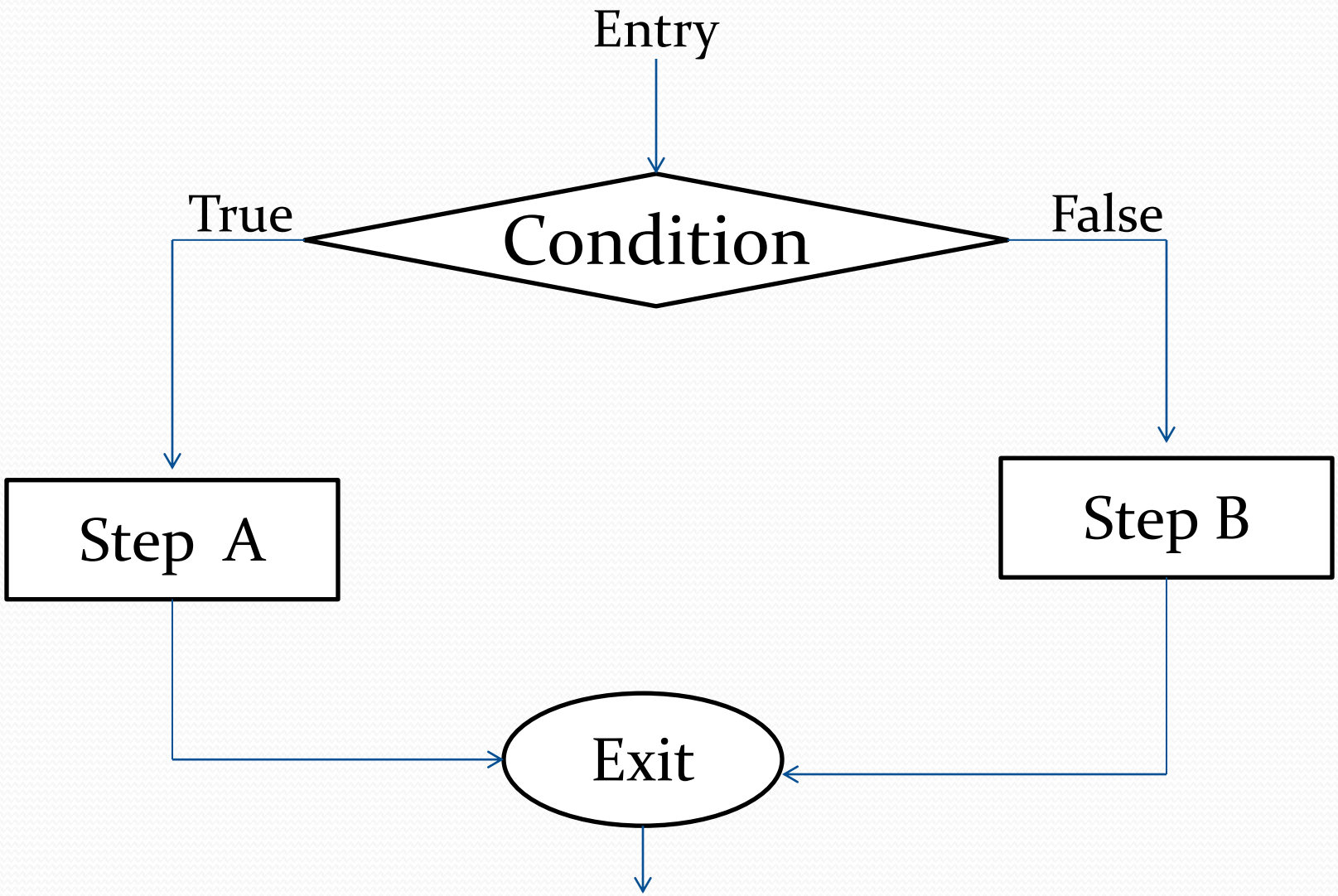

Condition?
False
True
Statement

```
CLS
INPUT “Enter any number”; num
IF num > 0 THEN PRINT “positive number”
IF num < 0 THEN PRINT “negative number”
IF num=0 THEN PRINT “Zero”
END
```

IF…THEN…ELSE statement

- It is an extension of IF…THEN statement.
- It tests a particular condition and asks the computer true or false question only.
- If condition is true , the computer follows the command given after THEN.
- If condition is false , the computer altogether ignore the command given after THEN and moves to ELSE part to execute the command started there.
- The general format is:

IF <condition> THEN <statement> ELSE <statement>


Entry
True
Condition
False
Step A
Step B
Exit

```
CLS
INPUT "Enter number"; num
IF num > 1 THEN
    PRINT "Number is greater than 1"
ELSE
    PRINT ; num; " is less than 1"
END IF
END
```

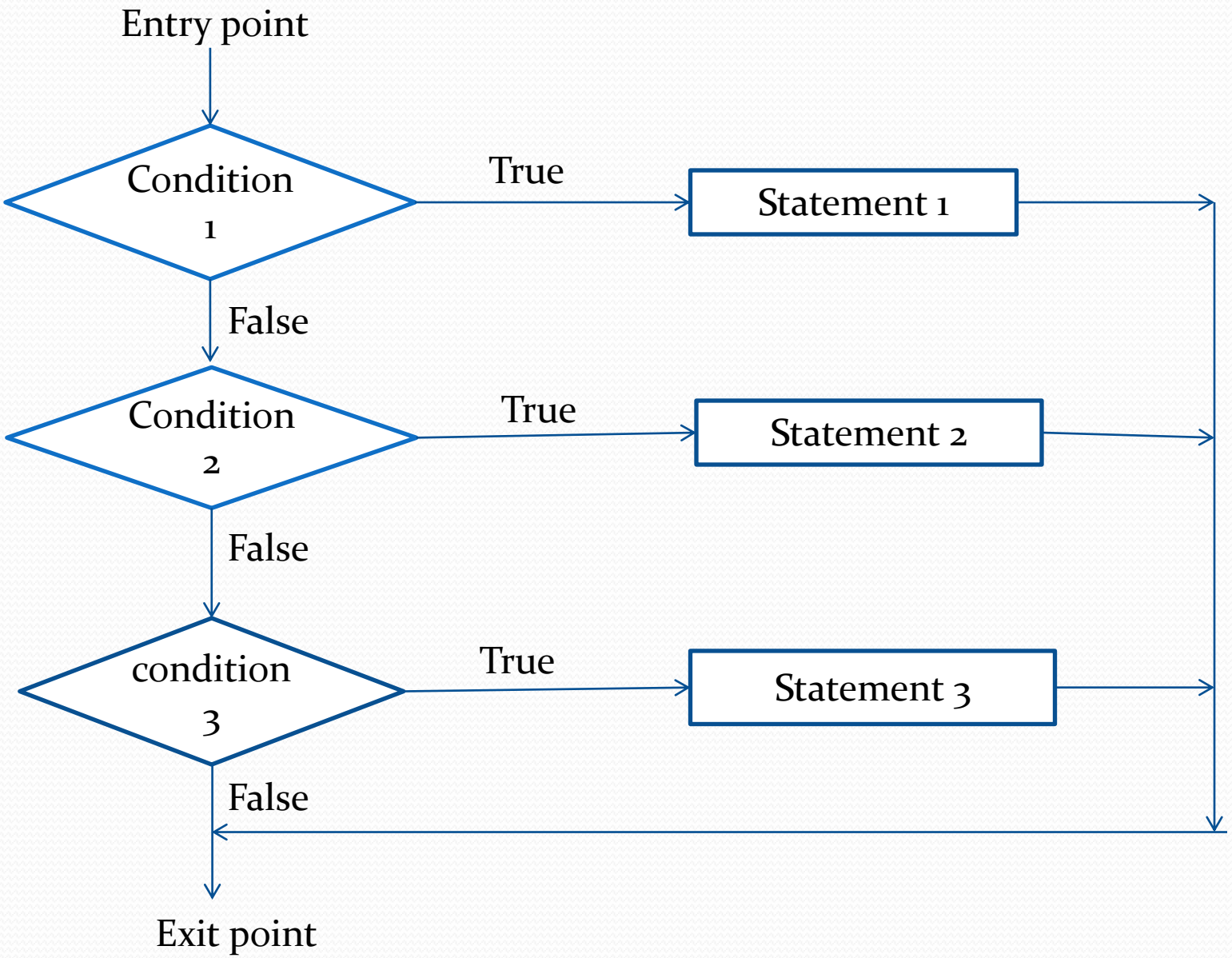
Q) WAP to input two different number and find the greatest number.

Q) WAP to input a number and check whether the given number is odd or even.

IF…ELSEIF…ENDIF statement

- It is a chain of IF statement.
- The executable statements following the last ELSE statement are executed if all the conditions on the IF and ELSEIF lines are false.
- The general format is:

```
IF [condition] THEN
        [statementblock-1]
ELSEIF [condition] THEN
        [statementblock-2]
ELSE
        [statementblock-3]
END IF
```


Entry point
Condition 1
True
Statement 1
False
Condition 2
True
Statement 2
False
condition 3
True
Statement 3
False
Exit point

```
CLS
INPUT "Enter your age:";age
IF age>18 THEN
        PRINT "He can drive"
ELSEIF age<18 THEN
        PRINT "He can't drive"
ELSEIF age=18 THEN
        PRINT "Visit driving centre"
ELSE
        PRINT "something error"
ENDIF
END
```

Q) WAP to input a number and check whether given number is positive, negative or zero.

Q) WAP to find the greatest number among three number.

Q) WAP to input a percentage and check whether he/she scores distinction, first div, second div, third div or fail.

File Edit View Search Run Options

```
CLS
INPUT "Enter three numbers"; a, b, c
IF a > b AND a > c THEN
    PRINT a; "is greater"
ELSEIF b > a AND b > c THEN
    PRINT b; "is greater number"
ELSE
    PRINT ; c; " is  greater number"
END IF
END
```

Untitled

```
Enter three numbers? 6,2,1
 6 is greater
```

File Edit View Search Run Options

```
CLS
INPUT "Enter percentage"; per
IF per >= 80 AND per <= 100 THEN
    PRINT "Distinction"
ELSEIF per >= 60 AND per < 80 THEN
    PRINT "First Division"
ELSEIF per >= 50 AND per < 60 THEN
    PRINT "Second Division"
ELSEIF per >= 40 AND per < 50 THEN
    PRINT "Third  Division"
ELSEIF per < 40 THEN
    PRINT "Fail"
ELSE
    PRINT "please enter value between 0 to 100"
END IF
END
```

Untitled

```
Enter percentage? 88
Distinction
```

Looping Statement

- Loop is a control structure that causes a statement or group of statements to repeat, based on a condition.
- The looping statement allow flexibility to the programmer in controlling the number of times a specific instruction is to be repeated.
- The three types of looping statement are FOR…NEXT, WHILE…WEND and DO…LOOP.

FOR…NEXT statement

- It is used to execute a series of instruction for a given number of times.
- FOR statement is placed at beginning of loop and NEXT statement at the end.
- The criteria of FOR…NEXT are starting value, final value and Increment/Decrement.
- The general format is:

FOR <counter variable >=x to y<STEP z>

<program statement>

NEXT<counter variable>

- A counter variable is used to keep count of the number of times the loop is repeated.
- The initial value(x) and final value(y) for the counter variable are given with FOR statement.
- If the STEP clauses in not included, the default value for STEP ,i.e., 1 is used.
- The next part generate a new value for the loop variable on the basis of STEP value. If STEP value is 3, then it adds 3 to previous value of loop variable.
- After generation of new value for the loop variable it returns the program control back to the FOR part of the FOR…NEXT statement.

File Edit View Search Run Options

```
CLS
FOR i = 1 TO 5 'by default STEP 1
    PRINT i;
NEXT i
END
```

Untitled

```
1  2  3  4  5
```

File Edit View Search R

```
CLS
FOR i = 5 TO 1 STEP -1
    PRINT i
NEXT i
END
```

Untitled

```
5
4
3
2
1
```

Q) WAP to print ‘n’ natural number.

Q) Program to display “My country Nepal” 15 times.

Q) WAP to display numbers from 20 to 5.

Q)Program to generates first ten even numbers.

Q) WAP to calculate sum of 10 natural numbers.

Q) WAP to calculate sum of n-natural numbers.

Q) WAP to calculate the factorial of a given number.

Q) WAP to print Fibonacci series.

- What is Fibonacci series?

The Fibonacci sequence is a sequence where the next term is the sum of the previous two terms. The first two terms of the Fibonacci sequence are 0 followed by 1.

The Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

File Edit View Search Run Options

```
REM program to  find the print of first n natural numbers.
CLS
INPUT "Enter the number": n
FOR i = 1 TO n
    PRINT i
NEXT i
END
```

Untitled

```
Enter the number? 5
 1
 2
 3
 4
 5
```

Q)WAP to find the product of first n natural number.

```
REM To find the product of first n natural numbers
CLS
INPUT "Enter the number"; n
p = 1
FOR i = 1 TO n
    p = p * i
NEXT i
PRINT "The product upto "; n; "="; p
END
```

REM“program to find factorial of a number.”

```
CLS
INPUT "Enter a Number: ", num
fact = 1
FOR i = 1 TO num
    fact = fact * i
NEXT i
PRINT "Factorial of ";n;" is ";fact
END
```

WHILE…WEND statement

- It is an entry-controlled loop.
- If the condition is true, all the statements are execute and when the WEND statement reached, control is reached to WHILE statement which evaluates the condition again.
- The general format is:

```
WHILE [condition]
        <program statement>
         INC/DEC
WEND
```

```
CLS
i=1
WHILE i<=10
        PRINT i
         i=i+1
WEND
END
```

Q) WAP to find the sum of digits of a given number.

Q) WAP to find the reverse of a given number.

Q) WAP to check the given number is palindrome or not palindrome.

Q) WAP to check the given number is Armstrong or not Armstrong.

File Edit View Search Run Options

```
REM"WAP to find sum of digits of given number"
CLS
INPUT "enter number"; num
WHILE num <> 0
    r = num MOD 10
    s = s + r
    num = num \ 10
WEND
PRINT "sum of digit of number is"; s
END
```

Untitled

```
enter number? 456
sum of digit of number is 15
```

File Edit View Search Run

```
CLS
INPUT "enter number"; num
WHILE num <> 0
    r = num MOD 10
    s = s * 10 + r
    num = num \ 10
WEND
PRINT "reversed number is"; s
END
```

Untitled

```
enter number? 123
reversed number is 321
```

- What is Armstrong number?

The number which is formed by the sum of cubes of its own digits is Armstrong number.

Example: 153, 370, 371, 407, etc.

153 = 1^3 + 5^3 + 3^3
= 1 + 125 + 27
= 153

- What is Palindrome number?

A palindromic number (also known as a numeral palindrome or a numeric palindrome) is a number that remains the same when its digits are reversed.

Example : 121, 55,767,etc

QB64 x32

File Edit View Search Run Options

```
CLS
INPUT "Enter any number"; N
A = N
S = 0
WHILE N <> 0
    R = N MOD 10
    S = S * 10 + R
    N = N \ 10
WEND
IF A = S THEN
    PRINT A; "is palindrome"
ELSE
    PRINT A; "is not palindrome"
END IF
END
```

Untitled

```
Enter any number? 121
 121 is palindrome
```

DO…LOOP statement

- It causes a set of program statement to execute repeatedly until certain condition are met or as long as certain condition are true.
- The syntax of DO…LOOP statement can take either of the two forms:
- Syntax1:

```
DO
        <statement block>
LOOP WHILE/UNTIL <boolean expression>
```

Syntax2:

```
DO WHILE/UNTIL <boolean expression>
                <statement block>
LOOP
```

```
CLS
i=0
DO
    PRINT i
    i=i+1
LOOP WHILE i<=10
END
```

Using WHILE ... WEND

```
CLS
INPUT "ENTER ANY NUMBER"; N
A = N
S = 0
WHILE N <> 0
   R = N MOD 10
   S = S + R ^ 3
   N = N \ 10
WEND
IF A = S THEN
   PRINT A; "IS ARMSTRONG"
ELSE
   PRINT A; "IS NOT ARMSTRONG"
END IF
END
```

- **Using DO WHILE ..... LOOP**

```
CLS
INPUT "Enter a number"; n
b = n
DO WHILE n <> 0
    r = n MOD 10
    a = a + r ^ 3
    n = n \ 10
LOOP
IF a = b THEN
    PRINT "It is armstrong number";
ELSE
    PRINT "It is not armstrong number";
END IF
END
```

Library Function in QBASIC

- A function is a built-in formula or a ready-made program that helps us to perform a certain task such as mathematical, financial, logical, etc. A function manipulates data passes to it and returns either a string or a numeric value. There are two types of functions in QBASIC Programming.
- User-defined function
- Built-in function/Library function

- **User-defined function:** It is created by the programmer to perform the operations as per the requirements. It can be a numeric or string function.
- **Built-in function:** It is a pre-defined program which is provided by QBASIC to perform some task easily. It gives many more built-in functions for manipulating strings, numbers. It makes our work easy. It is also known as Library functions. There are two types of built in function :Mathematical Function and String Library Function

Mathematical Function

- **SQR :** It calculate and return the square root of non-negative number.
- Example:

```
CLS
PRINT SQR(144)
END
```

Output:12

- **MOD :** It returns reminder, when one number is divide by another number.
- Example:

```
CLS
PRINT 9 MOD 5
END
```

Output:

4

- **INT :** It returns an integer of a given number.
- Example:

```
CLS n = 27.5
PRINT INT(n)
END
```

- Output:

27

- **CINT :** It returns nearest rounding numeric value from -32768 to 32767 as argument.
- Example:

```
CLS
PRINT CINT(3.49)
PRINT CINT(3.51)
END
```

- Output

3
4

- **SGN :** It returns the sign of numeric value. If the number is positive it returns 1, if the number is negative, it returns -1 and if the number is 0 it returns 0.
- Example:

```
CLS
PRINT SGN(15)
PRINT SGN(-15)
PRINT SGN(0)
END
```

- Output:

```
1
-1
0
```

- **ABS :** It returns corresponding positive value.
- Example:

```
CLS
PRINT (-15.3)
END
```

- Output:

```
15.3
```

- **SIN :** It returns the sign value of given number.
- **COS :** It is used to obtain the cosine value of a number.
- **TAN :** It returns the tangent of a number.
- Example:

```
CLS
PRINT SIN(1.5)
PRINT COS(1.5)
PRINT TAN(1.5)
END
```

- Output:

.997495
.0707372
14.10142

String Library Function

- UCASE$
- LCASE$
- LEFT$
- RIGHT$
- MID$
- LTRIM$
- LEN
- VAL
- ASC
- CHR$
- STR$
- STRING$
- SPACE$
- TAB
- DATE$ and TIME$

UCASE$

- It converts string values to uppercase.
- Example:

```
CLS
 x$ = "teach school"
PRINT UCASE$(x$)
END
```

Output:

TEACH SCHOOL

LCASE$

- It converts string value to lowercase.
- Example:

```
CLS
x$ = "TEACH SCHOOL"
PRINT LCASE$(x$)
END
```

Output:

teach school

LEFT$

- it extract and return the number of characters from the let of a string.
- Example:

```
CLS
 x$ = “computer”
 PRINT LEFT$(x$, 5)
END
```

- Output:

compu

RIGHT$

- It extract and return the numbers of characters from the right of a string.

- Example:

```
CLS
x$ = "computer"
PRINT RIGHT$(x$, 4)
 END
```

- Output:

uter

MID$

- It is used to pick up the required strings from the string.

- Example:

```
CLS
a$ = "computer"
 PRINT MID$(a$, 3, 5)
END
```

- Output:

mpute

LRTIM$

- LRTIM$ is used to remove the spaces from the left of the string and RTRIM$ is used to remove the spaces from the right of the string.
- Example:

```
CLS
 a$ = “computer”
 b$ = “science”
 PRINT a$ + b$
PRINT RTRIM$(a$) + LTRIM$(b$)
END
```

- Output:

```
computer        science
computerscience
```

LEN

- It returns the length of a give string.
- Example:

```
CLS
b$ = "nepal"
PRINT LEN(b$)
END
```

- Output:

5

VAL

- If both string are started with number value. This function can perform mathematical calculations among them.
- Example:

```
CLS
a$ = “10 boys”
b$ = “20 girls”
 total = VAL(a$) + VAL(b$)
PRINT "Total students = "; total
END
```

- Output:

Total students = 30

ASC

- It returns ASCII value of a character.
- Example:

```
CLS
x$ = "A"
PRINT ASC(x$)
END
```

Output:

65

dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
12	c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
13	d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
14	e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
15	f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
27	1b	033	ESC	59	3b	073	;	91	5b	133	[	123	7b	173	{
28	1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	|
29	1d	035	GS	61	3d	075	=	93	5d	135	]	125	7d	175	}
30	1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

www.alpharithms.com

CHR$

- It returns character value of a give ASCII code.
- Example:

```
CLS
FOR i = 65 TO 90
    PRINT i; "="; CHR$(i)
NEXT i
END
```

- Output

```
65 = A
66 = B
 .
 .
90 = Z
```

STR$

- It converts numeric value to string value, which cannot be used for mathematical calculations.
- Example:

```
CLS
 a = 50
 b = 20
PRINT "Before using STR$ : "; a + b
PRINT "After using STR$ : "; STR$(a) + STR$(b)
END
```

- Output:

```
Before using STR$ : 70
After using STR$ : 50 70
```

STRING$

- It returns a string of a specified length made up of a repeating character.
- Example:

```
CLS
PRINT STRING$(5, 97)
END
```

- Output:

```
aaaaa
```

SPACE$

- It is used to put blank spaces.
- Example:

```
CLS
PRINT "Class"; SPACE$(6); "Roll.No"
END
```

- Output:

```
Class      Roll.NO
```

TAB

- It is used to put Tab.
- Example:

```
CLS x$ = "Name"
y$ = "Address"
PRINT TAB(5); x$; TAB(10); y$
END
```

- Output:

```
Name    Address
```

DATE$ and TIME$

- DATE$ and TIME$ returns the current date and time respectively.
- Example:

```
CLS
PRINT DATE$
PRINT TIME$
END
```

```
CLS
n$="NEPAL"
FOR i=1 to LEN(n$)
        x$=RIGHT$(n$,i)
        PRINT x$
NEXT i
END
```

Output:

```
L
AL
PAL
EPAL
NEPAL
```

WAP to reverse a given string

```
CLS
INPUT "Enter any string ";N$
a=LEN(N$)
FOR i=a TO 1 STEP-1
      X$=MID(N$,I,1)
      W$=W$+X$
NEXT I
PRINT "Reversed string is " ; W$
END
```

WAP to check a given number is palindrome or not.

```
CLS
INPUT "Enter any string ";N$
a=LEN(N$)
FOR i=a TO 1 STEP-1
      X$=MID(N$,I,1)
      W$=W$+X$
NEXT I
IF N$=W$ THEN
         PRINT "palindrome"
ELSE
        PRINT "Not palindrome"
END
```

WAP to count vowels in a given string.

```
CLS
INPUT "Enter any string ";n$
FOR i=1 TO LEN(n$)
      x$=MID(n$,I,1)
      x$=LCASE(x$)
      IF x$="a" OR x$="e" OR x$="i" OR x$="o" OR x$="u"
      THEN v=v+1
       END IF
NEXT I
PRINT "Total no. of vowel is ";v
END
```

WAP to make odd no. character capital and even no. character smaller.

```
CLS
INPUT "Enter any string "; n$
FOR i=1 TO LEN(n$)
      x$=MID$(n$ ,i,1)
      IF I MOD 2= 0 THEN
         w$=w$+LCASE$(x$)
      ELSE
         w$=w$+UCASE$(x$)
      ENDIF
NEXT I
PRINT w$
END
```

Nested Loop in QBASIC

- A nested loop is a loop within a loop, an inner loop within the body of an outer one. This repeats until the outer loop finishes.
- QBASIC allows to use one loop inside another loop.

Example

```
CLS
FOR i = 1 TO 4
      PRINT "This is an outer loop "; i
       FOR j = 1 TO 3
            PRINT "Inner Loop "; j
       NEXT j
       PRINT   //new line
NEXT i
END
```

```
CLS
FOR i = 1 TO 5
        FOR j = 1 TO i
                PRINT j;
        NEXT j
        PRINT
NEXT i
END
```

WAP to print the following pattern:

5
55
555
5555
55555

```
CLS
n=5
FOR i=1 TO 5
        PRINT ; n
        n=n*10+5
NEXT i
END
```

WAP to print the following pattern:

1
12
123
1234
12345

```
CLS
n=5
FOR i=1 TO 5
        FOR J=1 to i
                PRINT J;
         NEXT J
         PRINT
NEXT i
END
```

WAP to print the following pattern:

```
NEPAL
 EPA
  P
```

```
CLS
n$="NEPAL"
l=LEN(n$)
FOR i=1 TO 3
        x$=MID$(n$,i,l)
        PRINT TAB(i); x$
        l=l-2
NEXT i
END
```

Array in QBASIC

- Array is a variable which stores different values of the same data type. It is useful to organize multiple variables.
- To declare an array DIM (dimension) statement is used.
- Example: DIM numbers(10)

Example

```
CLS
DIM num(5)
num(0) = 2
num(1) = 4
num(2) = 5
num(3) = 10
num(4) = 6
num(5) = 9
PRINT num(0); num(1); num(2); num(3); num(4); num(5)
PRINT num(3) + num(5)
END
```

- Output:

2 4 5 10 6 9
19

WAP to take 5 numbers from the users and print the sum of numbers.

```
CLS
DIM num(4)
sum = 0
PRINT "Enter any 5 numbers"
FOR i = 0 TO 4
        INPUT " :: "; num(i)
        sum = sum + num(i)
NEXT i
PRINT "The Sum = "; sum
END
```