3 More On Qbasic

In this chapter, you will learn about :

- *Introduction* • *Remark Statement (REM)* • *Loops in QBASIC*
- *Different Types of Loops* • *Modulus Operator (MOD) Function*
- *String Operators* • *QBASIC Library Functions* • *String Functions*
- *Mathematical Functions* • *Graphics in QBASIC*

INTRODUCTION

QBASIC is a programming language which has been evolved from BASIC language. It is a popular language, developed to teach fundamental concepts to the beginners. It is a fourth-generation program language which allow programmers to place explanatory comments directly in the program. This is an effective way of documenting the program.

Being a fourth-generation language, its advantage is that it is a result orient language and can be used by non-programing personnel such as the user.

REMARK STATEMENT (REM)

The REM statement is used for writing a remark in QBASIC Program. This program does not execute a line or a statement if it starts with the REM statement. The advantages of using this statement are as follows :

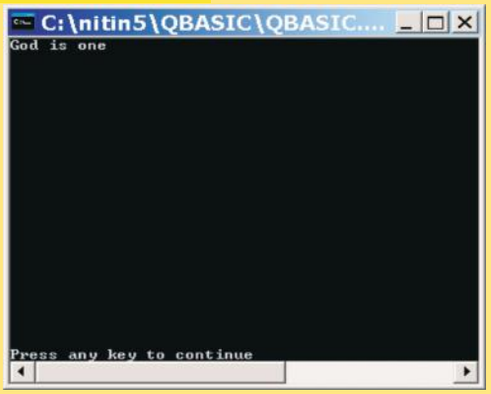
- Through it the logic of the program can be better understood.
- It helps in modifying and upgrading the program.
- Also the readability of the source code can be increased.

Syntax : REM <any remark or comments of the programmer>

Program :

```
REM My First Program
CLS
Print"GOD is one"
END
```

Output : (Fig. 7.1)



LOOPS IN QBASIC

A set of instructions is a program, which is repeated a certain number of times, until a condition is met or infinitely; it is also called loop. Loop is a process that breaks the normal sequence that

is top-down sequence of a program and executes the given set of instructions once or more. A given instruction in a program can be easily repeated by using a loop.

Suppose, we have a program that contains 7 statements. Now, in a normal top-down sequence, the statements will be executed one by one from 1 to 7. But if we use a loop statement after the 3rd statement asking to repeat the statements 5, 6 and 7, the program will be executed as per the instructions. Thus, a loop statement alters the normal sequence.

Loops are divided into two categories :

1. Unconditional loop 2. Conditional loop

We can convert an unconditional loop with the help of IF-----THEN-----ELSE statements.

DIFFERENT TYPES OF LOOPS

GOTO Statement

The GOTO statement is used to unconditionally transfer control from one point in the program to another. The control can be transferred either forward or backward.

Know More

The first character of a Label should be an alphabet and the name can consist of alphabets, numeric digits and the underscore (–) character.

Syntax

```
GOTO <Lable 1 Line number>
```

Study the following example carefully :

Program 1

```
PRINT "PANDIT ";
GOTO Name1 (unconditional loop)
Name2: (label)
PRINT "JAWAHAR"
Name3: (label)
PRINT "LAL"
Name1: (label)
PRINT "NEHRU"
END
```

Output :

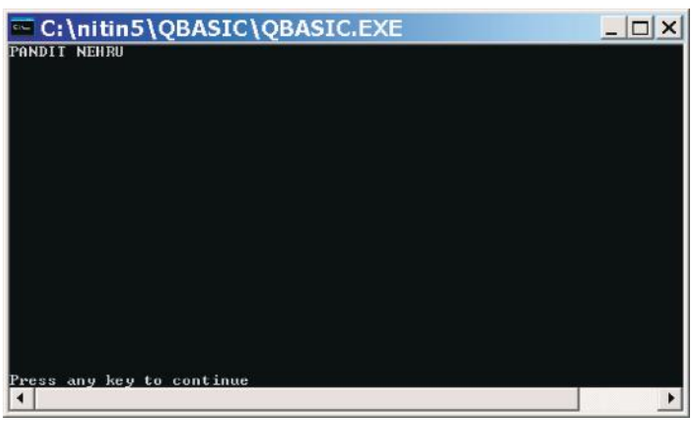


Fig. 7.2

In the above example. 'Name1', 'Name2' and 'Name3' are labels used to identify three parts of the program. The loop 'GOTO Name1' makes the execution control jump to 'PRINT NEHRU', ignoring labels 'Name2' and 'Name3'. The program ends with the END. Thus, we see the final output as PANDIT NEHRU.

Let us look at another example (Program 2) where GOTO is used with the IF...THEN statement to create a conditional loop. Here, PARA 2 and PARA 3 are the labels.

Know More

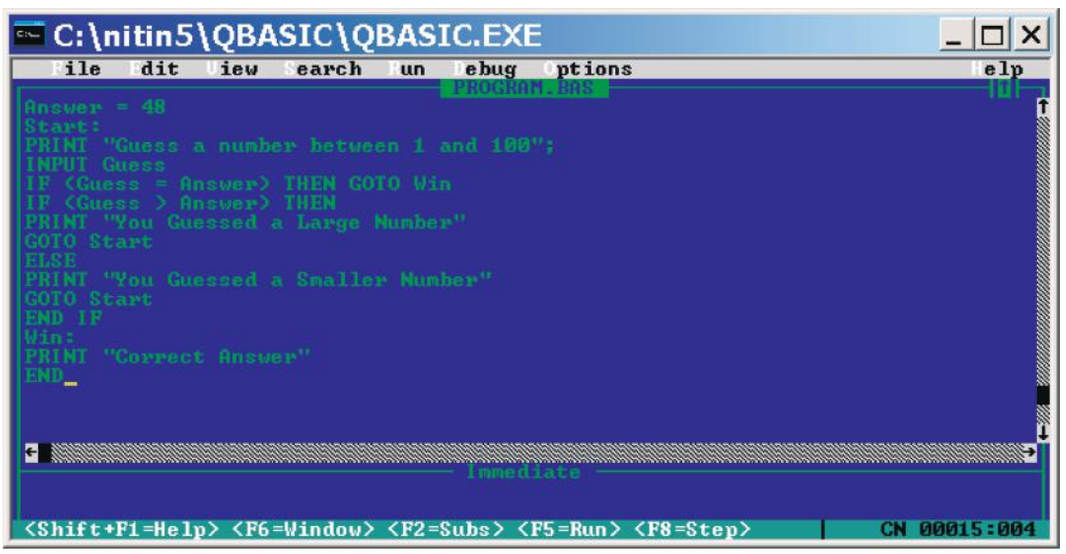
Any command after the End command is not executed.

Program 2

```
Statement 1
PARA 2 : (label)
Statement 2
Statement 3
Statement 4
PARA 3 : (label)
IF <condition 1> THEN GOTO END (GOTO +IF...THEN statement = conditional
loop)
Statement 5
GOTO PARA 2 (unconditional loop)
END
```

In the above example, the GOTO statement along with IF...THEN statement becomes a loop. If 'condition 1' is true, the program follows the line after GOTO and jumps to END. If not, it proceeds to execute statement 5 after which the program execution again returns to the PARA 2 label.

The following example will illustrate this more clearly. It is a number guessing game in which the user will input his guess. The user wins when he guesses the correct number.



GOSUB Statement

The GOSUB statement is similar to the GOTO statement. The SUB part represents a subroutine or subprogram. When the execution reaches the GOSUB line, it jumps to the part where the subroutine is given. We use the RETURN statement with the GOSUB statement. When the program

execution reaches the RETURN statement, it automatically goes back to the line rule after the GOSUB statement and executes it. Using the GOSUB statement, we need not write the same set of codes again and again.

Let us understand this with the help of an example.

Program

```
GOSUB Wish (loop)
PRINT "AFTERNOON" (statement 1)
GOSUB Wish (loop)
PRINT "NIGHT" (statement 2)
END
Wish : (label indicating the subroutine)
PRINT "GOOD"
RETURN
```

Output :

Know More

We can also use Line numbers as labels in a program. We can write a program that has line numbers for each line. The line numbers do not have to be in sequence.

In this example, the subroutine 'Wish' has been used for two different parts of the program (statements 1 and 2). Every time the execution control reaches the RETURN statement, it returns to the line after the last executed GOSUB statement. Hence, we gt the output as 'GOOD AFTERNOON' and 'GOOD NIGHT' in the order.

FOR...TO...NEXT statement

The FOR...TO...NEXT is a conditional loop used to repeatedly execute a statement or a block of statements for the specified number of times. We assign a value to the variable and the program loops until that variable is equal to or greater than the number specified after 'TO'. This is a better method to control a loop execution.

Syntax :

```
FOR (counter variable)=(start) TO (end) STEP (increment)
Statement 1
Statement 2
NEXT (counter variable)
```

The counter variable after FOR defines the function the loop will perform. We can use both numeric and text variables.

- **Start** is the initial value and **End** is the final value of the variable.
- **Increment** is the amount the variable is incremented each time through the loop. The STEP part is optional. In the absence of STEP, the counter variable is incremented by 1 each time during the execution of the loop.

Let us write a program using the FOR...TO...NEXT loop to input a number and then display the multiplication table of that number.

Program

```
REM Multiplication Table
INPUT "Enter number:", N
FOR I = 1 TO 10 STEP 1
P = N*I
PRINT N; "X"; I; "="; P
NEXT I
```

Output :

Fig. 7.5

First, let us identify the parts of the FOR statement. 'I' is the counter variable which has 1 as the start value. The end value is 10 and the counter variable will increase by 1, which is the STEP value.

The statements after the FOR loop are executed. Then, the current value of the counter variable is increased by 1. It means the new value of 'I' will be 2. This process continues till the value of 'I' is equal to the end value 10. As the loop is executed using the counter variable 'I', the multiplication table is displayed. Thus, if we input N as 3. We well see the output as :

3 x 1 = 3

3 x 2 = 6 and so on.

SELECT...CASE loop

The SELECT...CASE loop is an alternative to the nested IF ...THEN...ELSE statements. The output in both the cases is the same. It is easier to use the SELECT...CASE command in programs.

Let us use the same program with both IF...THEN...ELSE and SELECT...CASE commands and compare the results.

Program using IF...THEN...ELSE	**Program using SELECT.... CASE**
PRINT "Menu"	PRINT "Menu"
PRINT "1. First"	PRINT "1. First"
PRINT "2. Second"	PRINT "2. Second"
PRINT "3. Third"	PRINT "3. Third"
PRINT "Enter Your Choice"	PRINT "Enter Your Choice"
INPUT CHOICE	INPUT CHOICE

```
IF CHOICE = 1 THEN
PRINT "FIRST"
ELSE IF CHOICE = 2
THEN PRINT "SECOND"
ELSE IF CHOICE = 3
THEN PRINT "THIRD"
ELSE PRINT "Invalid Entry"
END IF
```

```
SELECT CASE CHOICE
CASE 1
PRINT "FIRST"
CASE 2
PRINT "SECOND"
CASE 3
PRINT "THIRD"
CASE ELSE
PRINT "Invalid Entry"
END SELECT
```

The SELECT...CASE statement can in click any number of alternative statements for execution.

The SELECT...CASE structure is simple than the nested IF...THEN...ELSE structure. In both these examples. "FIRST", "SECOND" and "THIRD" will be printed if the value of CHOICE is entered as 1, 2 and 3 respectively. Otherwise, "Invalid Entry" will be printed.

WHILE...WEND loop

Unlike the FOR...NEXT loop, which is executed for a specified number of times, the WHILE... WEND loop executes itself as long as the condition stated with it is true. This loop is useful when the number of time the loop is to be repeated is not known in advance. WHILE...WEND loops start with the statement WHILE and ends with the statement WEND.

While executing the program, the computer will check if the variable value initialised meets the given condition. If the condition is met, the loop is executed till the time the condition is satisfied.

Program

```
REM Variable
Number =2
REM WHILE <condition>
WHILE NUMBER <10>
REM Statement 1
PRINT "New number :" : Number
REM Statement 2
Number = Number + 2
REM END of WHILE loop
WEND
```

Know More

The WHILE statement evaluates both the conditions numerical or logical.

Remember :

- A condition is an expression that will return a TRUE (non-zero) or FALSE (zero) answer.
- The loop body can consist of more than one QBASIC statement.
- We can use the REM statement to insert our comments in a program.

DO...WHILE and DO...UNTIL loops

The DO...LOOP is another control statement that repeats a block of statements WHILE a condition **is true** or UNTIL a condition **becomes true**. In this case, the loop is executed at least once, irrespective of whether the condition is true or not.

It is preferable to use the DO loop instead of the FOR...NEXT loop when the number of repetitions is not specified and is fully dependant upon a condition.

First, we type a variable that is to initialised. Then, we should clearly indicate whether we want to use WHILE or UNTIL with the loop. Also, we should enter a condition along with the WHILE or UNTIL line. This should be followed by the loop body, which consists of a list of statements. Type LOOP at the end of the program to close it.

Syntax 1

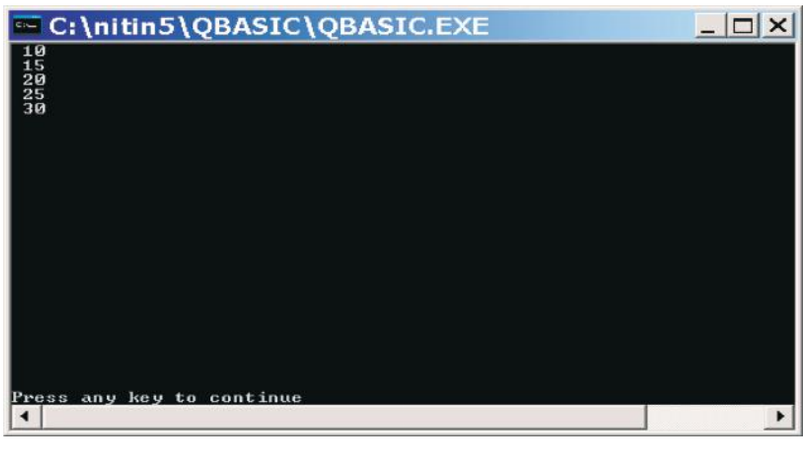
```
Variable
DO[{WHILE/UNTIL}<condition>]
Statement 1
Statement 2
LOOP
```

Syntax 2

```
Variable
DO
Statement 1
Statement 2
LOOP[{WHILE/UNTIL}<condition>]
```

Let us look at an example that uses the UNTIL Statement.

```
Program
Number = 10
DO UNTIL Number = 35
PRINT Number
Number = Number + 5
LOOP
```



Output :

MODULUS OPERATOR (MOD) FUNCTION

MOD is an arithmetic operator which divides one number by another and returns the remainder. It is very useful to check the divisibility of integers.

Syntax : R = A MOD B (where, R, A and B are integer type numeric variables)

Let us study an example to understand this more clearly.

Program

```
A = 20

B = 3

C = 5

R = A MOD B

PRINT "Remainder =" ; R

IF (A MOD C = 0) THEN

PRINT C; "is a factor of "; A

END IF
```

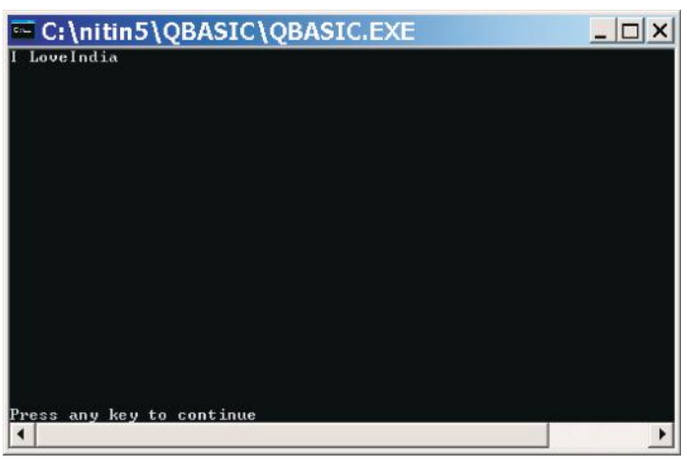
In the above example, the MOD function checks the divisibility of the numeric variables A, B and C, which are integers. Here, we see that since A is perfectly divisible by C, the modulus (remainder) is shown as 0. If the modulus is not zero. We can say the number is not divisible.

STRING OPERATORS

You have learnt about strings in QBASIC. The '+' operator is used to join two strings in QBASIC. This is known as concatenation.

Program

```
REM STRING Operation

CLS

A$ = "I"

B$ = "Love"

C$ = " India"

D$ = A$ + " " + B$ + C$

PRINT D$

END
```



Output :

While concatenating two or more strings, we should be careful about the spaces between the strings. In the above example, we put space between A$ and B$, but no space between B$ and C$. The result can be seen in the output screen.

The '+' is the only arithmetic operator that can be used for String operations in QBASIC. However, there are many in-built library functions for strings. We will discuss some of them here.

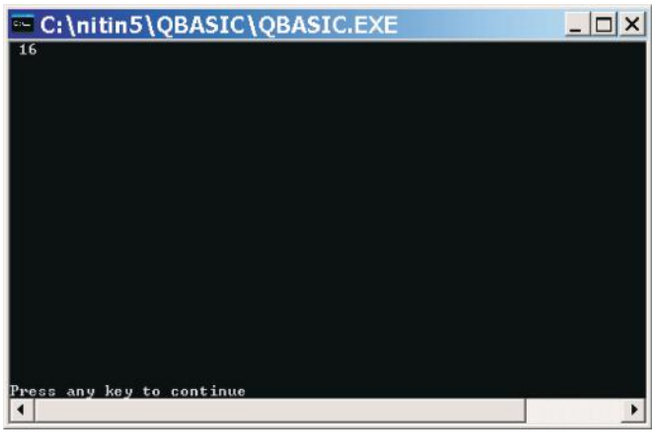
QBASIC LIBRARY FUNCTIONS

QBASIC has some predefined in-built library functions that have a specific use. These functions are like small procedures or subroutines. These in-built library functions can be divided into two categories :

- String Functions
- Mathematical Functions

STRING FUNCTIONS

Let us discuss some basic String functions.



1. LEN Function

This function is used to find the length of the String. The LEN function also count the spaces in the string.

Program

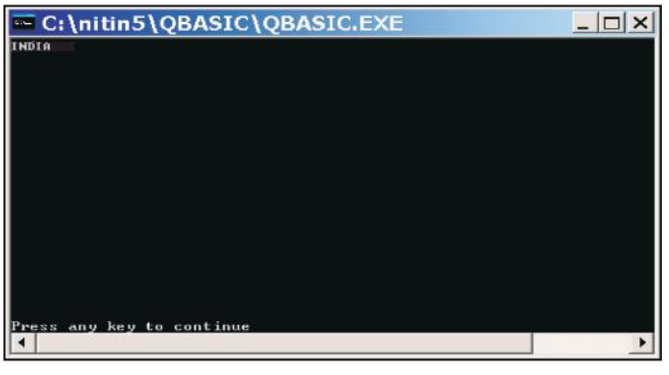
```
Name$ ="I Love India"
PRINT LEN (Name$)
```

2. LEFT$ Function

This string function is used to print the specified number of characters from the left side of the String. Here again, the spaces are also counted as characters.

Program

```
Name$ = "I Love India"
PRINT LEFT$(Name$, 6)
```



3. RIGHT$ Function

This function prints the specified number of characters from the right side of the String, including the spaces.

Program

```
Name$ = "I Love India"
PRINT RIGHT$ (Names$, 6)
```

4. MID$ Function

This function prints the specified number of characters starting any where in between the string, as mentioned.

Program

```
Name$ = "I Love India"

PRINT MID$ (Name$, 3, 6)
```

Here, the computer extracts six characters starting from the third character of the string, which is 'I'.

5. STR$ Function

This function shows us the String representation of the number mentioned in the program. The result will be in the String format. Hence, we won't be able to do calculations with the number.

Know More

If we assign a string value to a non-string variable an error message is displayed.

Program

```
Number = 99

PRINT STR$ (Number)
```

Here, the number 99 is changed into a string value 99. The string "99" cannot be used in any arithmetic expression or calculation.

6. VAL Function

The VAL function is the opposite of the STR$ function. It converts the numeric representation of the string into a numeric digit. This numeric digit can be used in any arithmetic expression and calculation.

Program

```
Number$ = "99"

PRINT VAL (Number$)
```

7. ASC Function

All character used in computer have their respective numeric value, which is known as the ASCII value. The ASC function shows the ASCII value of a character.

Program

```
C$ = "B"

PRINT ASC(C$)
```

8. SPACE$ Function

This function gives us a string that has the specified number of spaces.

Program

```
A$ = "I"

B$ = "Love"

C$ = "India"

PRINT A$ + SPACE$(2) + B$ + SPACE$(5) + C$
```

MATHEMATICAL FUNCTIONS

In QBASIC there are some in-built library functions to execute mathemtical calculations as well. Let us look at some of them :

1. RND Function

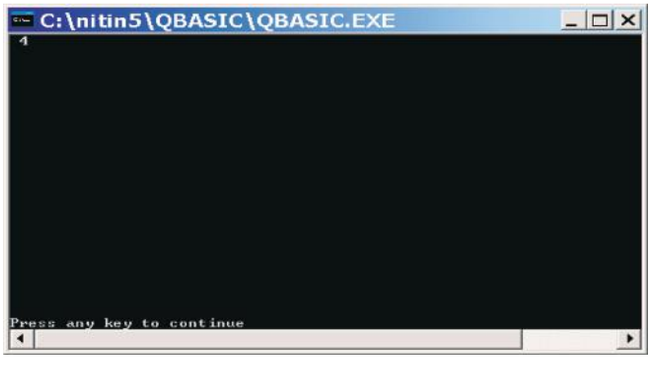
The RND function returns a random number between 0 and 1. This will be a decimal number that can go up to 7 decimal places.

Program

```
R = 9 + (RND * 31)

PRINT "A Random number between 9 and 40 is", R
```

Suppose the RND function returns a number–71 (a value between 0 and 1). The product of RND * 31 = 23.21. The INT value of 23.21 is 23, which becomes 33 when 10 is added to it. Here, the INT function has been used to round off the decimal part of the final random number.

2. SQR Function



This function returns the square root of a numeric expression or numeric variable.

Program

```
Number 1 = 16

PRINT SQR (Number 1)
```

GRAPHICS IN QBASIC

We have learnt in previous classes how to draw figures in LOGO using procedures and primitives. QBASIC also allows us to draw different shapes and figures in different colors using special statements.

COLOR Code

In QBASIC, colors are represented by numbers. Some of the colors with their respective color code are :

0 = black	4 = red	8 = grey	12 = light red
1 = blue	5 = magenta	9 = light blue	13 = light magenta
2 = green	6 = brown	10 = light green	14 = yellow
3 = cyan	7 = white	11 = light cyan	15 = bright white

LINE Statement

As the name suggests, this statement is used to draw a line from one point to another. It can also be used to draw a rectangular box on a specified diagonal point.

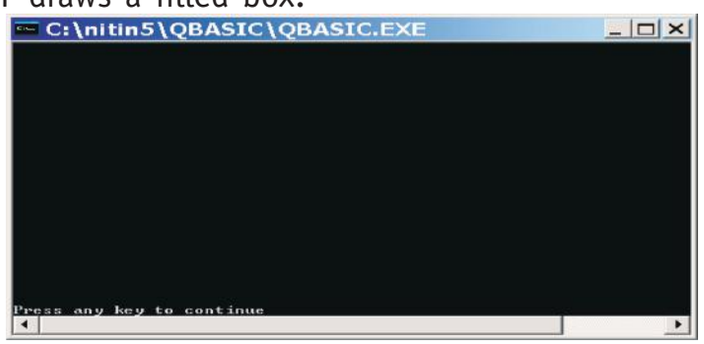
Syntax : LINE (X1, Y1) – (X2, Y2), C, B/BF

Let us study the different parts of this syntax.

- (X1, Y1) is the coordinate indicating the starting point of the line whereas (X2, Y2) specifies where the line will end. Here, X is the distance across the top left corner and Y is the distance down the left corner.
- C represents the color code. This is optional. In the absence of a color code, white is taken as the default color.
- The command B or BF draws a rectangle, where (X1, Y1) – (X2, Y2) is the imaginary diagonal. The B option draws an empty box whereas BF draws a filled box.

Program

```
SCREEN 9

REM To draw a red line

LINE (3, 3) - (9, 9), 4

REM To return to text mode

SCREEN 0
```



The above program will draw a red line of the said length. The line will start a little inside the top left corner and move diagonally to end at the ninth pixel across and ninth pixel down (Fig. 7.15).

If we add B at the end, an empty box with a diagonal of the specified length will be drawn.

PSET Statement

This command is used to draw a single point (pixel) on the screen with a specified color. The syntax is as follows :

Syntax : PSET (X, Y), C

Here, X and Y denote the coordinates of the point. The letter C is the color code, which is optional. Look at the following program :

```
SCREEN 9

PSET (50, 50), 1

SCREEN 0
```

The output will be a single pixel in blue colour at the specified location on the screen.

CIRCLE Statement

We use this statement to draw circles. We can also use this command to draw an arc.

The syntax is as follows :

Syntax : CIRCLE (X, Y), R, C, SA, EA

Let us study the parts of the syntax :

- X and Y represent the coordinates of the centre of the circle.
- R is the radius of the circle.
- C is the color code (optional).
- SA denotes the starting angle if you want to draw an arc.
- EA denotes the end angle of the arc.

Points to Remember

- QBasic is a programming language evolved from BASIC language.
- It allows programmers to place explanatory command directly into the program.
- The REM statement is used for writing a remark in QBasic program.
- A set of instructions is a program, which is repeated a certain number of time, until a condition is met; it is also called loop.
- Loops are divided into two categories.
- The GOTO statement is used to transfer one point in the program to another.
- The FOR---TO---NEXT is a conditional loop used to execute a statement.
- LEN function is used to find the length of the String.
- The VAL function is the opposite of the STR$ function.
- LINE statement is used to draw a line from one point to another.
- CIRCLE statement is used to draw circles.

A. Tick (✓) the correct option :

1. This statement is used for writing a remark in QBASIC program.
   (a) LINE ☐ (b) REM ☐ (c) CIRCLE ☐
2. LEN function is used to find the _____ of the string.
   (a) length ☐ (b) width ☐ (c) height ☐
3. This function returns the square root of a numeric variable.
   (a) RND ☐ (b) ASC ☐ (c) SQR ☐
4. This statement is used to draw circles.
   (a) PSET ☐ (b) CIRCLE ☐ (c) LINE ☐

B. Fill in the blanks :

1. The _____ function returns a random number between 0 and 1.
2. QBASIC is a _____ generation program language.
3. _____ is an arithmetic operator.
4. Loops are divided into _____ categories.
5. _____ is the initial value of the variable.

C. Tick (✓) for the correct statements and cross (✗) for the incorrect ones :

1. The REM statement is used for writing a formula in QBASIC. ☐
2. The SELECT.....CASE structure is simple than the nested IF....THEN....ELSE structure. ☐
3. The ASC function shows the ASCII value of a character. ☐
4. In QBASIC number are represented by colors. ☐

D. Answer the following Questions :

1. What do you mean by a loop in a programming language ?
2. What do you mean by GOTO statement ?
3. Why do we use FOR...TO...NEXT statement ?
4. What do you mean by concatenation ?
5. Why do we use REM statement in a QBASIC program ?
6. What is a GOSUB statement ?

ACTIVITY

- Write the complete syntax of the following graphics command. Execute them properly as well :

(a) LINE (b) CIRCLE