COMPUTER PROGRAMMING

WORKING WITH FILES

DR. USMAN AKMAL

DEPARTMENT OF CIVIL ENGINEERING
UNIVERSITY OF ENGINEERING AND TECHNOLOGY, LAHORE

CONTENTS

WORKING WITH FILES

QBASIC uses sequential file-processing commands. We will learn how to create, modify and manage sequential files on the disk. By using the files in QBASIC one could process large amount of data and store it for later use.

The OPEN Statement

Following is the syntax of the OPEN statement;

```
OPEN filename$ [FOR mode] AS [#]filenumber
```

filename$ **It must be a string variable or constant consisting of a valid file name.**

The OPEN Statement

mode It has one of the following three values;

- **APPEND**
- **INPUT**
- **OUTPUT**

✓ ***mode* refers to the way your program uses the file.**

✓ ***To create the file (or to overwrite the exiting file), open the file in OUTPUT mode.***

✓ ***To read data from a file, open the file in INPUT mode.***

✓ ***APPEND mode allows to add the data to the end of a file.***

The OPEN Statement

`filenumber`

- ✓ It must be a whole number that links the file to a number used throughout the program having value from 1 to 255.
- ✓ More than one file can be opened in a program (up to 255).
- ✓ Instead of typing complete file name, one could refer to the file using assigned ***`filenumber`***.
- ✓ The pound sign (#) is optional before the ***`filenumber`***.

The OPEN Statement [EXAMPLES]

EXAMPLE 1

Consider it is required to create a sequential file that stores the data of students. The following `OPEN` statement will be used for this purpose;

```
OPEN "ClassData.dat" FOR OUTPUT AS #1
```

EXAMPLE 2

If the data of students is already created and is required in the program to read stored data. Then the following `OPEN` statement will be used;

```
OPEN "ClassData.dat" FOR INPUT AS #1
```

The OPEN Statement [EXAMPLES]

EXAMPLE 3

If it is required to add more students data in existing file, then to add data to the end of the file, following form of `OPEN` statement will be used;

```
OPEN “ClassData.dat” FOR APPEND AS #1
```

The CLOSE Statement

After using a file, you must close the file with the `CLOSE` statement. One should close all the files in QB program when they are no longer required.

Following is the syntax of the `CLOSE` statement;

```
CLOSE [[#]filenumber] [,[#]filenumber]
```

- ✓ **All files can be closed in a program by putting `CLOSE` on a line by itself.**
- ✓ **If `CLOSE` is followed with one or more integer *`filenumbers`* separated by commas, QBASIC closes the files associated with those numbers.**

The CLOSE Statement [Examples]

EXAMPLE 1

To close an output file associated with the number 1, any one of the following will work;

```
CLOSE #1
```

```
CLOSE 1
```

EXAMPLE 2

To close files associated with file numbers 2 and 5, any one of the following will be used;

```
CLOSE #2, #5
```

```
CLOSE #2, 5
```

```
CLOSE 2, #5
```

```
CLOSE 2, 5
```

The CLOSE Statement [Examples]

EXAMPLE 3

To close all files in the program, use following;

```
CLOSE
```

Creating Sequential Files using PRINT # and WRITE # Statements

After opening a file, it is required to write into it through coding.

Most data going to a file comes from user input, calculations, DATA statements, or a combination of these.

The PRINT # Statement sends output data to a sequential file. The format of PRINT # statement is;

```
PRINT #filenumber, expressionlist
```

filenumber	**must be the number of the open file to which data to be saved.**

Creating Sequential Files using PRINT # and WRITE # Statements

`expressionlist`

It is one or more variables, constants, expressions, or a combination of each separated by commas and semicolons.

- ✓ **The only difference between `PRINT` and `PRINT #` is *`#filenumber`*, which redirects the output to a file rather than to the screen.**
- ✓ **It is important to remember that `PRINT #` prints data to a file exactly as the data would appear on-screen with the regular `PRINT` statement.**

Creating Sequential Files using PRINT # and WRITE # Statements

- ✓ **This means that positive numbers have a space before the number where the invisible plus sign is, semicolons make the data items print right next to one another, and the comma prints in next print zone on the disk (each print zone is 14 characters wide, just as on-screen).**

EXAMPLE PROGRAM SHOWING USE of OPEN, PRINT # and WRITE # STATEMENTS

```
INPUT "Number of Students = ", sn
DIM RollNo AS INTEGER
DIM StudentName AS STRING
DIM UrduMarks AS INTEGER, EngMarks AS INTEGER, MathMarks AS INTEGER
OPEN “SecA.dat" FOR OUTPUT AS #1
PRINT #1, “RollNo”, “Name”, “Urdu”, “English”, “Math”

FOR J = 1 TO sn
    CLS
    PRINT
    INPUT "Enter Student Roll No. = ", RollNo
    INPUT "Enter Student Name = ", StudentName
    INPUT "Enter Marks in Urdu = ", UrduMarks
    INPUT "Enter Marks in English = ", EngMarks
    INPUT "Enter Marks in Math = ", MathMarks
    PRINT #1, RollNo, StudentName, UrduMarks, EngMarks, MathMarks
NEXT J
CLOSE #1
PRINT
PRINT "DATA HAS BEEN SAVED IN THE FILE"
```

OPEN, PRINT # and WRITE # STATEMENTS

```
DIM RollNo AS INTEGER
DIM StudentName AS STRING
DIM UrduMarks AS INTEGER, EngMarks AS INTEGER, MathMarks AS INTEGER

OPEN "Data2.dat" FOR OUTPUT AS #1
INPUT "Enter Student Roll No. = ", RollNo
INPUT "Enter Student Name = ", StudentName
INPUT "Enter Marks in Urdu = ", UrduMarks
INPUT "Enter Marks in English = ", EngMarks
INPUT "Enter Marks in Math = ", MathMarks
WRITE #1, RollNo, StudentName, UrduMarks, EngMarks, MathMarks
CLOSE #1

OPEN "Data2.dat" FOR INPUT AS #2
INPUT #2, RollNo, StudentName$, UrduMarks, EngMarks, MathMarks
Total = UrduMarks + EngMarks + MathMarks
Percent = Total / 300 * 100
CLOSE #2

OPEN "Data2.dat" FOR APPEND AS #3
PRINT #3, "Total Marks = "; Total
CLOSE #3
```

READ and DATA Statements

```
READ grade
READ firstName$, lastName$
READ fullName$, age, weight, hometown$


DATA 87.5
DATA “Ali”, “Ahmad”
DATA “Ali Ahmad”, 32, 70, “Lahore”
```

READ and DATA Statements

```
CLS
FOR J = 1 TO 5
   READ RollNo%, StudentName$, MathMarks%, EngMarks%, UrduMarks%
   Total = MathMarks% + EngMarks% + UrduMarks%
   PRINT RollNo%, StudentName$, Total

NEXT J
DATA 1,"A",66,66,66
DATA 2,"B",77,77,77
DATA 3,"C",88,88,88
DATA 4,"D",99,99,99
DATA 5,"E",66,66,66
```

END OF LECTURE

OPEN, PRINT # and WRITE # STATEMENTS

```
CLS
GOTO Here
First:
PRINT "A"
GOTO Final
There:
PRINT "B"
GOTO First
Here:
PRINT "C"
GOTO There
Final:
END
```

```
CLS
GOTO 100
300 PRINT "A"
GOTO 400
200 PRINT "B"
GOTO 300
100 PRINT "C"
GOTO 200
400 END
```

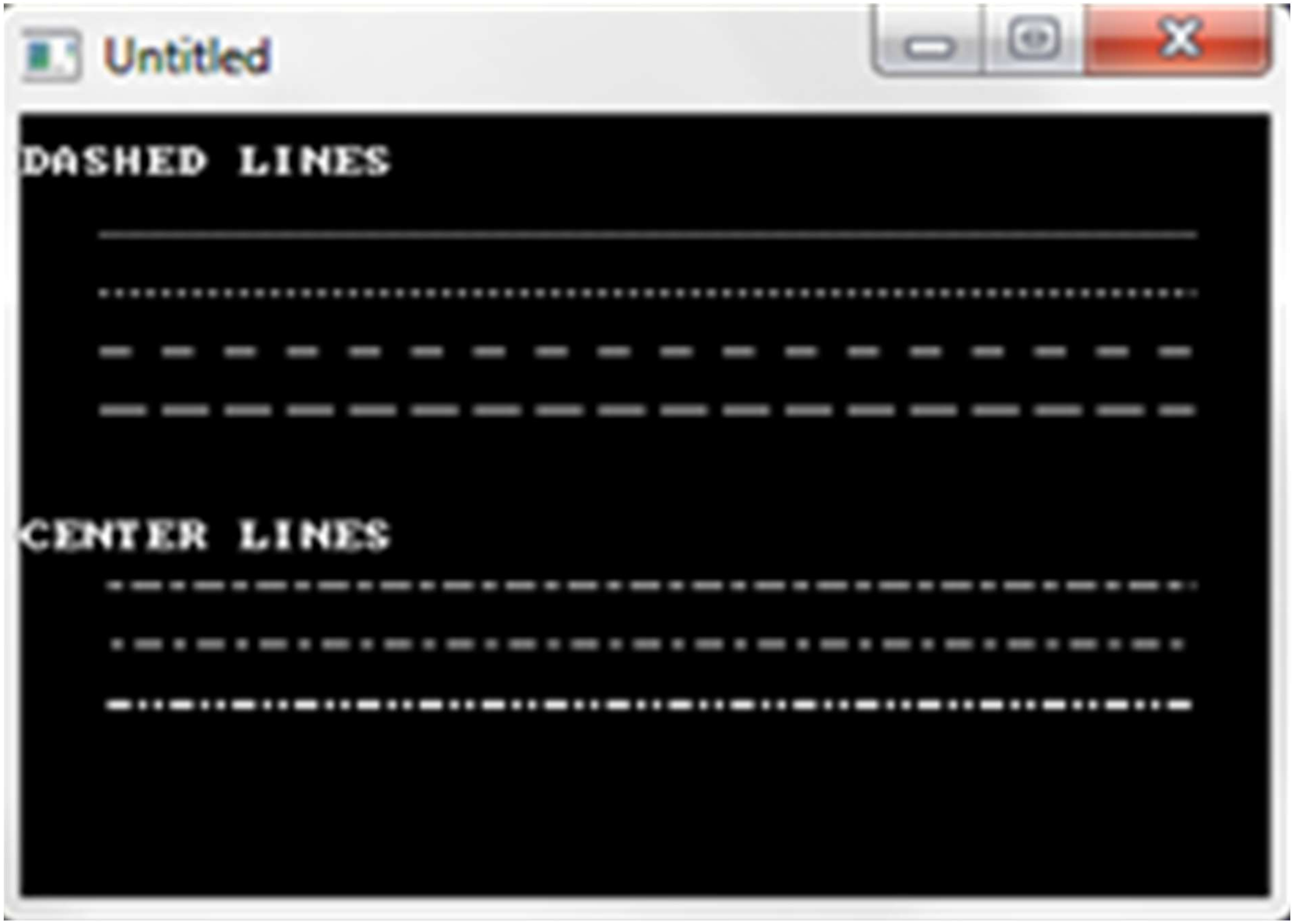
```
CLS
PRINT "123456789012345678901234567890"
PRINT 3, 4

123456789012345678901234567890
+3            +4
```

```
SCREEN 7
LOCATE 2: PRINT "DASHED LINES"
LINE (20, 30)-(300, 30), , , &HAAAA '1010101010101010
LINE (20, 45)-(300, 45), , , &HCCCC '1100110011001100
LINE (20, 60)-(300, 60), , , &HFF00 '1111111100000000
LINE (20, 75)-(300, 75), , , &HFFF0 '1111111111110000
LOCATE 14: PRINT "CENTER LINES"
LINE (20, 120)-(300, 120), , , &HF3FC '1111001111111100
LINE (20, 135)-(300, 135), , , &HE3F8 '1110001111111000

LINE (20, 150)-(300, 150), , , &HFCCC '1111110011001100
LINE (20, 151)-(300, 151), , , &HFCCC '1111110011001100

DO
LOOP WHILE INKEY$ = ""
```



```
DASHED LINES

CENTER LINES
```