Syllabus Content:

10.3 Files

- show understanding of why files are needed
- use Pseudocode for file handling:
    - OPENFILE <filename> FOR READ/WRITE/APPEND // Open file (understand the difference between various file modes)
    - READFILE <filename>,<string> // Read a line of text from the file
    - WRITEFILE <filename>,<string> // Write a line of text to the file CLOSEFILE <filename> // Close file
    - EOF(<filename>) // function to test for the end of the file
- Write pseudocode to handle text files that consist of one or more lines

10.3 Files

Data need to be stored permanently. One approach is to use a file. For example, any data held in an array while your program is executing will be lost when the program stops. You can save the data out to file and read it back in when your program requires it on subsequent executions.

A text file consists of a sequence of characters formatted into lines. Each line is terminated by an end-of-line marker.
The text file is terminated by an end-of-file marker.

Note: you can check the contents of a text file (or even create a text file required by a program) by using a text editor such as NotePad.

PSEUDOCODE:

```
OPENFILE <filename> FOR WRITE // open the file for writing
WRITEFILE <filename >, <stringValue> // write a line of text to the file
CLOSEFILE // close file
```

```
DECLARE line1, line2, line3 : STRING
OPENFILE "file.txt" FOR WRITE // open the file for reading
    WHILE NOT EOF "file.txt"
        INPUT line1
        INPUT line2
        INPUT line3
        WRITEFILE "file.txt" , line1              // write a line of text to the file
        WRITEFILE "file.txt" , line2
        WRITEFILE "file.txt" , line3
    END WHILE
CLOSEFILE "file.txt"                              // close file
```

Or you can write lines in File like this too

```
OPENFILE  "file.txt" FOR WRITE                            // open the file for reading
            WRITEFILE "file.txt" , "Good Morning"         // write a line of text to the file
            WRITEFILE "file.txt" , "Good Day"
            WRITEFILE "file.txt" , "Good night"
CLOSEFILE  "file.txt"                                     // close file
```

PSEUDOCODE:

```
OPENFILE <filename> FOR READ // opens the file for read
WRITEFILE <filename >, <stringValue> // write a line of text to the file
CLOSEFILE // close file
```

```
DECLARE line1, line2, line3 : STRING  // no need to re-declare variables in OPENFILE, just
use value of variable to output on screen during OPENFILE operation.

OPENFILE  "file.txt" FOR READ // open the file for reading
      WHILE NOT EOF ("file.txt")
            READFILE "file.txt" , line1 // write a line of text to the file
            READFILE "file.txt" , line2
            READFILE "file.txt" , line2
      END WHILE
CLOSEFILE  "file.txt  // close file
```

Or

```
DECLARE line1, line2, line3 : STRING // variables already been declared during WRITEFILE
                                     // no need to re-declare variables in OPENFILE, just use
value of variable to output on screen during OPENFILE operation.

OPENFILE  "file.txt" FOR READ // open the file for reading

      WHILE NOT EOF ("file.txt")
            READFILE "file.txt"
            OUTPUT  line1                         // write a line of text to the file
            READFILE "file.txt"
             OUTPUT line2
            READFILE "file.txt"
            OUTPUT line2
      END WHILE
CLOSEFILE  "file.txt"                             // close file
```

PSEUDOCODE:

```
OPENFILE <filename> FOR APPEND // opens the file for read
WRITEFILE <filename >, <stringValue> // write a line of text to the file
CLOSEFILE // close file
```

```
DECLARE line1, line2, line3 : STRING        // declaration of variables

OPENFILE  "file.txt" FOR APPEND             // open the file for appending/editing

          line1 = "Sunday"
          line2 = "Monday"
          line2 = "Tuesday"

          WRITEFILE "file.txt" , line1                  // write a line of text to the file
          WRITEFILE "file.txt" , line2
          WRITEFILE "file.txt" , line2

CLOSEFILE  "file.txt"                                   // close file
```

The end-of-file (EoF) marker

If we want to read a file from beginning to end we can use a conditional loop. Text files contain a special marker at the end of the file that we can test for. Testing for this special end-of- file marker is a standard function in programming languages. Every time th is function is called it will test for this marker.

The function will return FALSE if the end of the file is not yet reached and will return TRUE if the end -of-file marker has been reached. In **pseudocode** we call this function EOF(). We can use the construct REPEAT ... UNTIL EOF().

If it is possible that the fi le contains no data, it is better to use the construct WHILE NOT EOF()

For example, the following pseudocode statements read a text file and output its contents:

```
OPENFILE "Test .txt" FOR READ
WHILE NOT EOF("Test.txt")
READFILE "Test. txt", TextString
OUTPUT TextString
ENDWHILE
CLOSEFILE "Test. txt"
```

```
PROCEDURE MakeNewFile (OldFile, NewFile, Status : STRING)
DECLARE Line1, Line2, Line3 : STRING
DECLARE NumCopied, NumRecs : INTEGER
        NumRecs ← 0
        NumCopied ← 0
OPENFILE OldFile FOR READ
OPENFILE NewFile FOR WRITE
                WHILE NOT EOF(OldFile)
                        READFILE OldFile, Line1
                        READFILE OldFile, Line2
                        READFILE OldFile, Line3
                        NumRecs ← NumRecs + 1
                        IF Line3 <> Status THEN
                                WRITEFILE NewFile, Line1
                                WRITEFILE NewFile, Line2
                                WRITEFILE NewFile, Line3
                                        NumCopied ← NumCopied + 1
                        END IF
                END WHILE
        OUTPUT "File " , OldFile , " contained " , NumRecs ,__" employee details"
        OUTPUT Numcopied , " employee sets of details were __ written to file", NewFile
CLOSEFILE OldFile
CLOSEFILE NewFile
END PROCEDURE
```

Accessing special Folders in VB

Locations of files can vary from machine to machine or user to user. The exact location of my Documents folder changes depending on who has logged on.
VB.net uses special system variables to hold the current users file locations, such as my documents, desktop, My Music, etc.
To get access to the variables, you must import the **system.environment** library.

NOTE: Not all locations are available due to system security

```
Option Explicit On
Imports System.Environment
Module Module1
    Dim mydocs As String
    Dim mymusic As String
    Dim myfavorites As String
    Sub main()
        mydocs = GetFolderPath(SpecialFolder.MyDocuments)
        mymusic = GetFolderPath(SpecialFolder.MyMusic)
        myfavorites = GetFolderPath(SpecialFolder.Favorites)
        Console.WriteLine(mydocs)
        Console.WriteLine(mymusic)
        Console.WriteLine(myfavorites)
        Console.ReadLine()
    End Sub
End Module
```

Using folders

To access sub-directories, concatenate the system folder path with the folder path and/or file name:

```
Option Explicit On
Imports System.Environment
Module Module1
    Dim mydocs As String
    Dim myfiles As String
    Sub main()
        mydocs = GetFolderPath(SpecialFolder.MyDocuments)
        myfiles = mydocs & "\textfiles"
        Console.WriteLine(myfiles)
        Console.ReadLine()
    End Sub
End Module
```

Opening a Text File in Visual Basic

The first step in working with files in Visual Basic is to open the file.

This is achieved using the Visual Basic ***FileStream*** class. The ***FileStream*constructor** accepts the file name to be opened as the first parameter, followed by a number of other parameters defining the mode in which the file is to be opened. These fall into the categories of ***FileMode***, ***FileAccess*** and ***FileShare***. The options available as listed in the following tables:

FileMode Options

Mode	Description
Append	If the file exists it is opened. Any writes are appended to the end of the file. Requires *FileAccess.Write* mode
Create	Creates a new file, removing old file if it already exists
CreateNew	Creates a new file and returns error if file already exists
Open	Opens an existing file. Returns error if file does not exist
OpenOrCreate	If file already exists it is opened, otherwise a new file is created
Truncate	Opens an existing file and deletes all existing content

FileAccess Options

Mode	Description
Read	Opens the file for reading only.
ReadWrite	Opens the file for both reading and writing
Write	Opens the file to writing only

FileShare Options

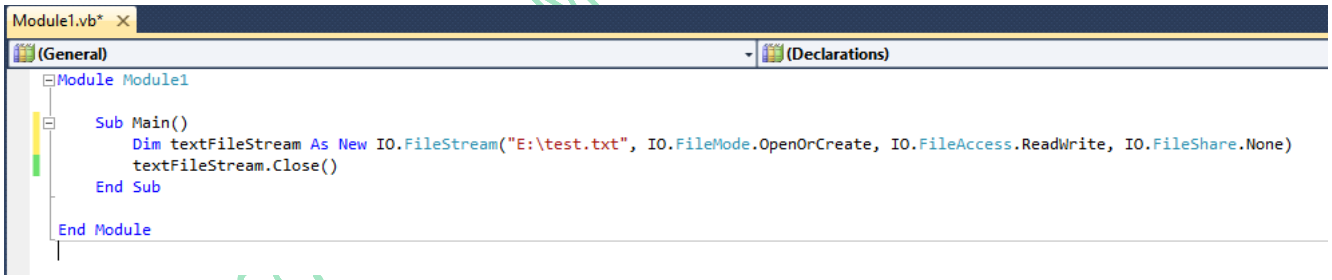
Mode	Description
None	The file cannot be opened by any other program until it is closed by the current program
Read	Other programs may simultaneously open and read from the file, but not write to it.
ReadWrite	Other programs may simultaneously open and read and write from/to the file.
Write	Other programs may simultaneously open and write to the file, but not read from it.

With the above options in mind, the following code excerpt opens **'C:\Temp\text.txt'** in ***FileMode.OpenOrCreate*** with ***FileAccess.ReadWrite*****permission** and no file sharing, and then closes it:

VB Code for creating text file

```
Module Module1
    Sub Main()
       Dim textFileStream As New IO.FileStream("E:\test.txt", IO.FileMode.OpenOrCreate,
       IO.FileAccess.ReadWrite, IO.FileShare.None)
       textFileStream.Close()
    End Sub
End Module
```

This code will create a text file in E drive with the name test.txt



```
Module Module1

    Sub Main()
        Dim textFileStream As New IO.FileStream("E:\test.txt", IO.FileMode.OpenOrCreate, IO.FileAccess.ReadWrite, IO.FileShare.None)
        textFileStream.Close()
    End Sub

End Module
```

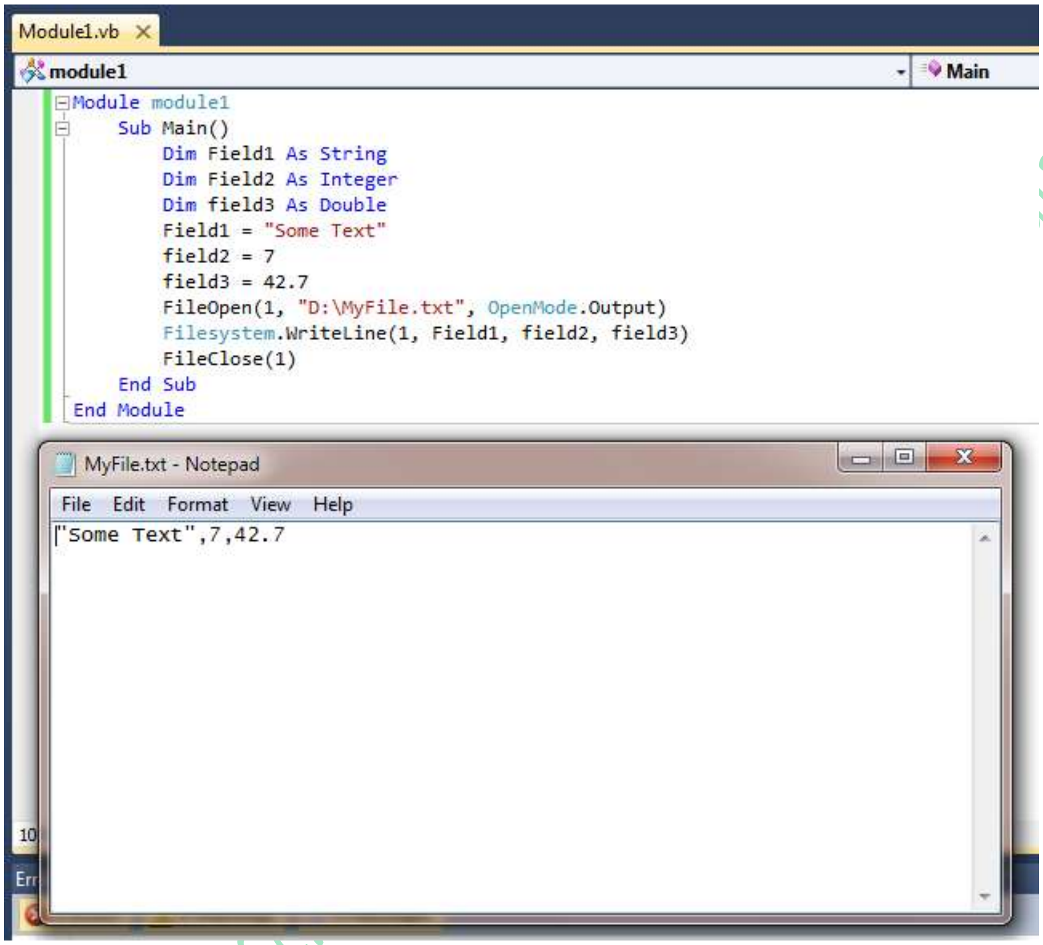
Creating CSV files with WRITELINE

The comma-separated values (CSV) file format is a file formats used to store tabular data in which numbers and text are stored in plain textual form that can be read in a text editor, spreadsheet or Database.
Lines in the text file represent rows of a table, and commas in a line separate what are fields in the tables row.
The following example used the WriteLine statement to create a CSV file with 3 variables:

```
Module module1
    Sub Main()
        Dim Field1 As String
        Dim Field2 As Integer
        Dim field3 As Double
        Field1 = "Some Text"
        field2 = 7
```

```
            field3 = 42.7
            FileOpen(1, "E:\MyFile.txt", OpenMode.Output)
            Filesystem.WriteLine(1, Field1, field2, field3)
            FileClose(1)
        End Sub
    End Module
```

The above code created a CSV file in E drive with these values in it.



```
Module module1
    Sub Main()
        Dim Field1 As String
        Dim Field2 As Integer
        Dim field3 As Double
        Field1 = "Some Text"
        field2 = 7
        field3 = 42.7
        FileOpen(1, "D:\MyFile.txt", OpenMode.Output)
        Filesystem.WriteLine(1, Field1, field2, field3)
        FileClose(1)
    End Sub
End Module
```

```
"Some Text",7,42.7
```

NOTE: Strings are enclosed in quotes, numbers are not enclosed in quotes
For other ways of manipulating CSV files, see page 82

Closing file

```
FileClose(1)
```

Close filenumber 1

PSEUDOCODE:

```
OPENFILE <filename> FOR WRITE // open the file for writing
WRITEFILE <filename >, <stringValue> // write a line of text to the file
CLOSEFILE // close file
```

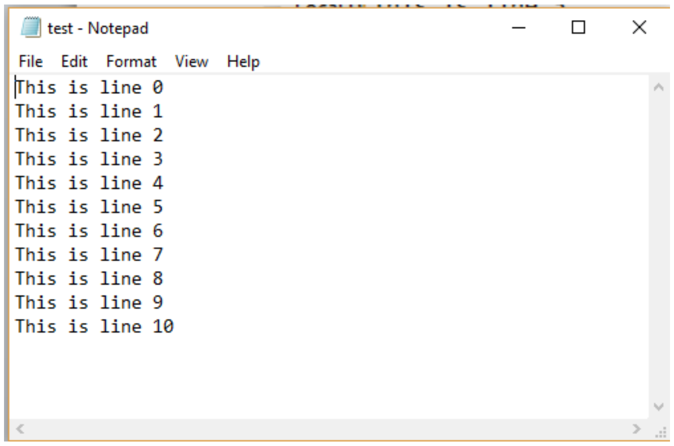
Writing to a File with Visual Basic

Once a file has been opened with the appropriate options, it can be written to using the Visual Basic ***StreamWriter*** class. The ***StreamWriter*constructor** takes a ***FileStream*** as the sole parameter.

The ***Write()*** and ***WriteLine()*** methods of the ***StreamWriter*** class are then used to write to the file. ***Write()*** writes the text with no new line appended to the end of each line. ***WriteLine()*** on the other hand, appends a new line to end of each line written to the file.

In the following code excerpt a **StreamWriter** object is created using the **FileStream**, and a For loop writes 11 lines to the file:1

```
Module module1
    Sub Main()
       Dim textFileStream As New IO.FileStream("E:\test.txt", IO.FileMode.OpenOrCreate,
       IO.FileAccess.ReadWrite, IO.FileShare.None)

        Dim myFileWriter As New IO.StreamWriter(textFileStream)
        Dim intCounter As Integer

        For intCounter = 0 To 10
            myFileWriter.WriteLine("This is line " & CStr(intCounter))
        Next intCounter

        myFileWriter.Close()
        textFileStream.Close()
    End Sub
End Module
```



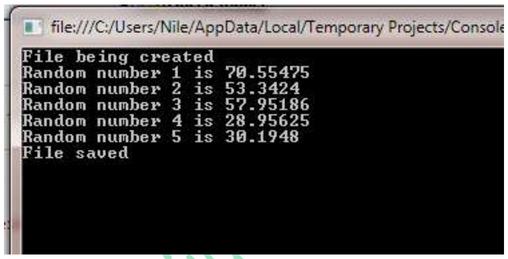
```
This is line 0
This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
This is line 6
This is line 7
This is line 8
This is line 9
This is line 10
```

Writing to a text file

Writing to a text file usually means creating a text file.
The following code examples demonstrate how to open, write to and close a file called **sampleFile.TXT** in each of the three languages. If the file already exist s, it is overwritten as soon as the file handle is assigned by the **'open file'** command.

VB Code with Text Files:

```
 Module module1
    Sub main()
        Dim FileHandle As IO.StreamWriter 'The file is accessed th ro ugh an object (see
        Dim LineOfText As String 'called a StreamWriter.
        FileHandle = New
        IO.StreamWriter("SampleFile . TXT")
        FileHandle.WriteLine(LineOfText)
        FileHandle.Close()
    End Sub
End Module
```



file:///C:/Users/Nile/AppData/Local/Temporary Projects/Console

```
File being created
Random number 1 is 70.55475
Random number 2 is 53.3424
Random number 3 is 57.95186
Random number 4 is 28.95625
Random number 5 is 30.1948
File saved
```

StreamWriter with text files

Two objects **StreamReader** and **StreamWriter** are used to read and write data in a text file. Both of these commands are stored in the System.IO library, so you will need to import it into your program.
The following line needs to be added B **System.IO** by adding **before** the Module definition

```
Imports System.IO
```

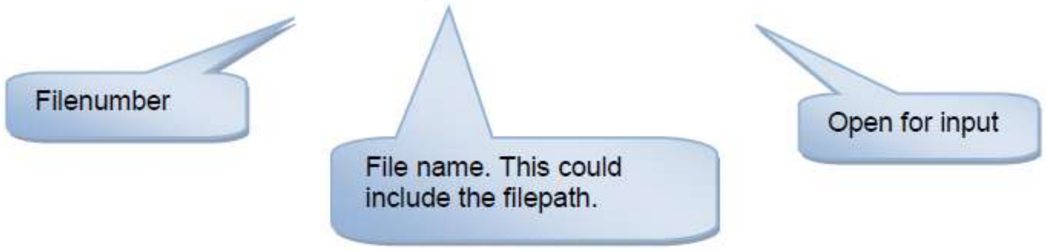
```
Option Explicit On
Imports System.IO
Imports System.Environment
Module module1
    'create a variable to write a stream of characters to a text file
    Dim CurrentFileWriter As StreamWriter
    Sub Main()
        Dim FileName, TextString As String
        Dim Count As Integer
        FileName = GetFolderPath(SpecialFolder.MyDocuments) & "text.txt"
        CurrentFileWriter = New StreamWriter(FileName)
        Console.WriteLine("File being created")
        CurrentFileWriter.WriteLine("File ceated on " & Now())
        For Count = 1 To 5
            TextString = Rnd() * 100
            Console.WriteLine("Random number " & Count & " is " & TextString)
            CurrentFileWriter.WriteLine("Random number " & Count & " is " & TextString)
        Next
        CurrentFileWriter.Close() ' close file
        Console.WriteLine("File saved")
        Console.ReadLine()
    End Sub
End Module
```

Files using Channels

The FILEOPEN command opens a file for input or output. It used the concept of having a filenumber to link the program to the physical file.

Reading Files (Input)

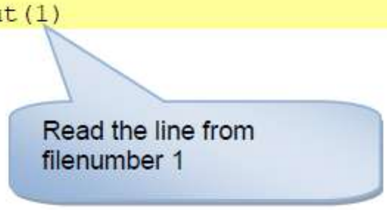
```
        FileOpen(1, "MyFile.txt", OpenMode.Input)
```



Reading a line of text

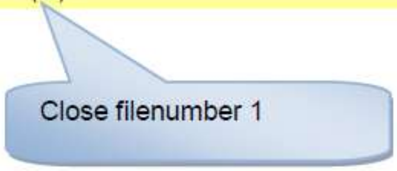
To read a line of text from the opened file

```
DIM LineFromFile as string

LineFromFile = LineInput(1)
```



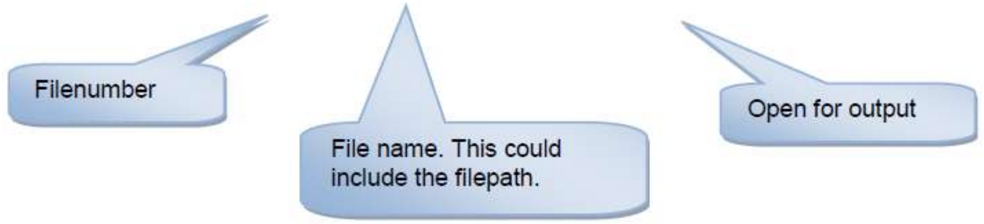
Closing file

```
FileClose(1)
```

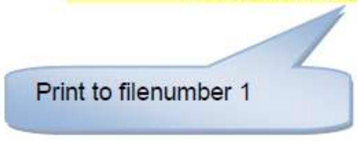


Writing a line of Text

```
        FileOpen(1, "MyFile.txt", OpenMode.output)
```



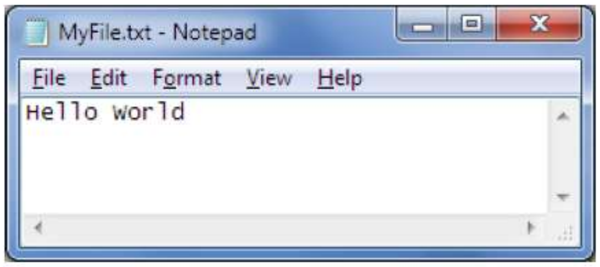
```
Dim LineofText As String
LineofText = "Hello World"
PrintLine(1, LineofText)
```



Printing a line of text

The PrintLine writes a string to a text file opened with a filenumber.

The above code will produce the following text file:

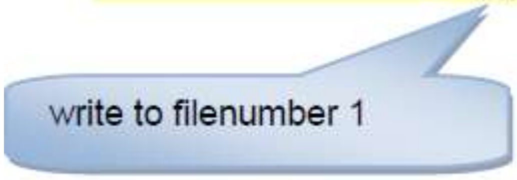


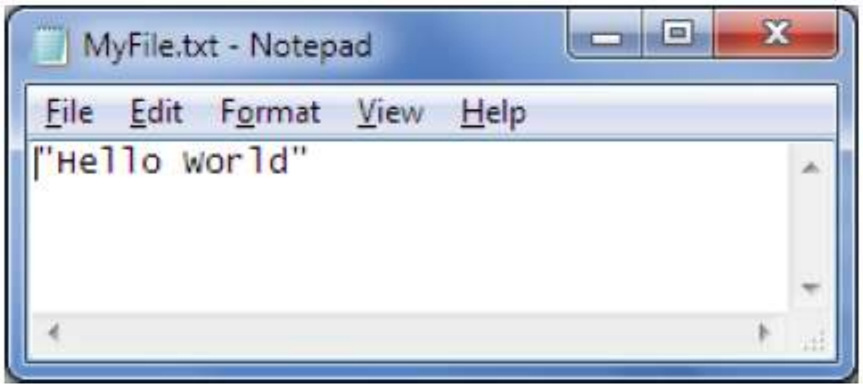
```
Hello world
```

Writing a line of text

The **Writeline** writes to a **textfile** opened with a **filenumber** BUT the string is enclosed in quotes

```
Dim LineofText As String
LineofText = "Hello World"
Writeline(1, LineofText)
```





```
"Hello world"
```

PSEUDOCODE:

```
OPENFILE <filename> FOR READ // open the file for reading
READFILE <filename >, <stringValue> // read a line of text to the file
CLOSEFILE // close file
```

Reading From a File in Visual Basic

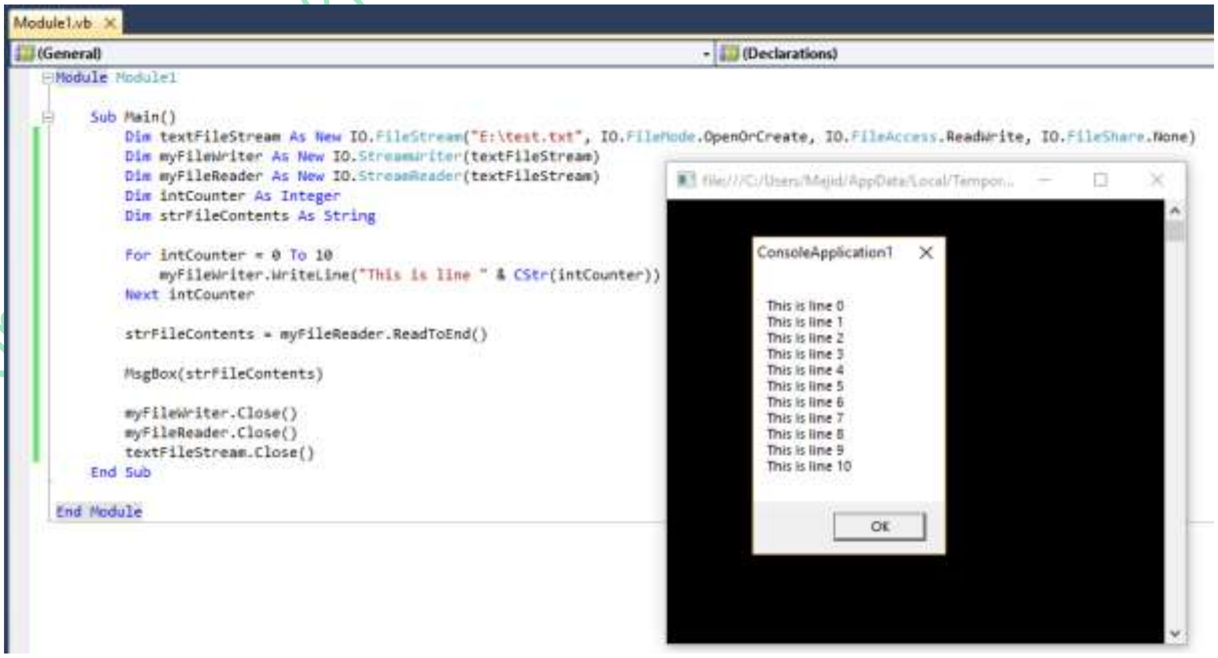
Now that we have created and written to a file the next step is to read some data from the file. This is achieved using the Visual Basic ***StreamReader*** object.

The ***StreamReader ReadLine()*** method can be used to read the next line from the file stream including the new line. The ***Read()*** method reads a line from the file but removes the new line.

The ***ReadToEnd()*** method can be used to read from the current line in the file to the end of the file.

The following code excerpt further extends our example to read the data back from the file after it has been written and display the contents in a MessageBox:

```
Dim textFileStream As New IO.FileStream("E:\test.txt", IO.FileMode.OpenOrCreate,
IO.FileAccess.ReadWrite, IO.FileShare.None)
Dim myFileWriter As New IO.StreamWriter(textFileStream)
Dim myFileReader As New IO.StreamReader(textFileStream)
Dim intCounter As Integer
Dim strFileContents As String
        For intCounter = 0 To 10
            myFileWriter.WriteLine("This is line " & CStr(intCounter))
        Next intCounter
strFileContents = myFileReader.ReadToEnd()
MsgBox(strFileContents)
myFileWriter.Close()
myFileReader.Close()
textFileStream.Close()
```

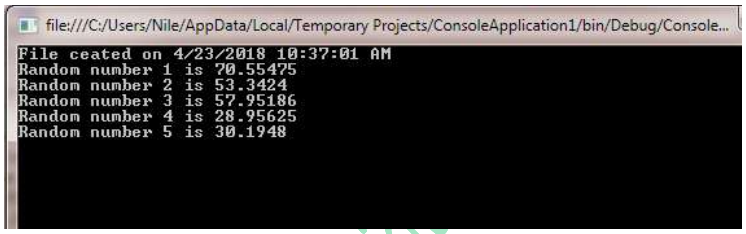


```
Module Module1

    Sub Main()
        Dim textFileStream As New IO.FileStream("E:\test.txt", IO.FileMode.OpenOrCreate, IO.FileAccess.ReadWrite, IO.FileShare.None)
        Dim myFileWriter As New IO.StreamWriter(textFileStream)
        Dim myFileReader As New IO.StreamReader(textFileStream)
        Dim intCounter As Integer
        Dim strFileContents As String

        For intCounter = 0 To 10
            myFileWriter.WriteLine("This is line " & CStr(intCounter))
        Next intCounter

        strFileContents = myFileReader.ReadToEnd()

        MsgBox(strFileContents)

        myFileWriter.Close()
        myFileReader.Close()
        textFileStream.Close()
    End Sub

End Module
```

```
This is line 0
This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
This is line 6
This is line 7
This is line 8
This is line 9
This is line 10
```

StreamReader with text files in VB

The StreamReader can either read the contents of the whole file into a variable, or read one line at a time.
ReadToEnd reads the entire file into a variable
ReadLine reads a single line (up to the CR code)


file:///C:/Users/Nile/AppData/Local/Temporary Projects/ConsoleApplication1/bin/Debug/Console...

```
File ceated on 4/23/2018 10:37:01 AM
Random number 1 is 70.55475
Random number 2 is 53.3424
Random number 3 is 57.95186
Random number 4 is 28.95625
Random number 5 is 30.1948
```

```
Option Explicit On
Imports System.IO
Imports System.Environment
Module Module1
    Dim CurrentFileReader As StreamReader
    Sub Main()
        Dim FileName, TextString As String
        TextString = ""
        FileName = GetFolderPath(SpecialFolder.MyDocuments) & "text.txt"
        CurrentFileReader = New StreamReader(FileName) 'opens the file
        If File.Exists(FileName) Then
            TextString = CurrentFileReader.ReadToEnd
        Else
            Console.WriteLine("File does not exist")
        End If
        CurrentFileReader.Close() ' close file
        Console.WriteLine(TextString)
        Console.ReadLine()
    End Sub
End Module
```

Appending to a text file

Sometimes we may wish to add data to an existing file rather than creating a new file. This can be done in Append mode. It adds the new data to the end of the existing file.
The following pseudocode statements provide facilities for appending to a file:

PSEUDOCODE:

```
OPENFILE <filename> FOR APPEND // open the file for append
WRITEFILE <filename >, <stringValue> // write a line of text to the file
CLOSEFILE // close file
```

```
Dim FileHandle As IO.StreamWriter 'The file is accessed through a StreamWriter.The extra
parameter True tells the system to append to the object.
        FileHandle = New
        IO.StreamWriter(" SampleFile . TXT", True) '
        FileHandle.WriteLine(LineOfText)
        FileHandle.Close()
```

VB Code

The following code examples demonstrate how to output the contents of a file in each of the VB.

```
Dim LineOfText As String
Dim FileHandle As System.IO.StreamReader
FileHandle = New
System.I(O.StreamReader("Test. txt"))

Do Until FileHandle.EndOfStream
     LineOfText = FileHandle.ReadLine()
     Console.WriteLine(LineOfText)
Loop

FileHandle.Close()
```

References:

- Cambridge International AS & A level Computer Science Course book by Sylvia Langfield and Dave Duddell
- Visual Basics Console Cook Book
- https://www.youtube.com/watch?v=snFPNd13XyA
- https://www.techotopia.com/index.php/Working_with_Files_in_Visual_Basic
- VB.NET Console Book by ***Dough Semple***