CHAPTER SIX
READING AND WRITING FILES
Files

A **file** is a collection of data stored in a disk with a specific name and a directory path. When a file is opened for reading or writing, it becomes a **stream**.

The stream is basically the sequence of bytes passing through the communication path. There are two main streams: the **input stream** and the **output stream**. The **input stream** is used for reading data from file (read operation) and the **output stream** is used for writing into the file (write operation).

Related term of files

- **Record**: one logical section of a file that holds a related set of data. If the file contains Student information, a record would hold the information on one student: name, address, studentID, etc. If there are 5,000 students registered, the file contains 5,000 records.
- **Field**: part of a record that defines a specific information. In the Student record, FirstName, LastName, StudentID, are fields. The field is the lowest element in the file. Even if the information consists of one character, Sex is M or F, it is still considered a separate field. The field is the equivalent of the variable - we call it a variable when it is used to store data in memory and call it a field when it stores in a file.
- **I/O**: stands for Input/Output. Whenever you work with a file you have to have ways of reading data from the file (that's Input) and ways of writing data to the file (that's Output). I/O operations consist of all those commands that let you read and write files.

Types of files

There are basically three types of files you can work with:

- **Sequential file**: this is a file where all the information is written in order from the beginning to the end. To access a given record you have to read all the records stored before it. It is in fact like

listening to a tape - you can go forward or back but you can't jump directly to a specific song on the tape.

- **Random file**: a file where all records are accessible individually. It is like a CD where you can jump to any track. This is useful when there is a large quantity of data to store and it has to be available quickly.
- **Binary files**

Binary Files are useful for

√ Storing numbers, programs and images

√ Where no defined file structure is present

❖ Access Mode of file

For Mode in the Open statement indicates how the file will be used. There are five access modes:

- **Input:** open for sequential input; the file will be read sequentially starting at the beginning.
- **Output:** open for sequential output; records will be written sequentially starting at the beginning; if the file does not exist, it is created; if it does exist, it is overwritten.
- **Random:** open for random read and write; any specific record can be accessed.
- **Append:** sequential output to the end of an existing file; if the file does not exist it is created; it does not overwrite the file.
- **Binary:** open for binary read and write; access is at **byte level.**

Learning how to create program to read and write text files

6.1 Introduction

To be able to open a file and read the data from storage unit of a computer, such as a hard drive as well as able to save the data into the storage unit are important functions of a computer program. In fact, the ability to store, retrieve and modify data makes a computer a powerful tool in database management.

In this Chapter, we will learn how to manage data that is stored as a text file. Using text file is an easy way to manage data, although it is not as sophisticated as full-fledged database management software such as SQL Server, Microsoft Access and Oracle. Visual Basic 2010 allows the user to create a text file, save the text file as well as read the text file. It is relatively easy to write code for the above purposes in VB2010. Reading and writing to a text file in VB2010 required the use of the StreamReader class and the StreamWriter class respectively. **StreamReader** is a tool that enables the streaming of data by moving it from one location to another so that the user can read it.

For example, it allows the user to read a text file that is stored in a hard drive. On the other hand, the **StreamWriter** class is a tool that can write data input by the user to a storage device such as the hard drive.

6.2 Reading a Text File

In order to read a file from the hard disk or any storage device, we need to use the

StreamReader class. To achieve that, first we need to include the following statement in

the program code:

Imports System.IO

This line has to precede the whole program code as it is higher in hierarchy than the StreamReader Class. In Fact, this is the concept of object oriented programming where **StreamReader** is part of the namespace System.IO. You have to put it on top of the whole program (i.e. above the Public Class Form 1 statement). The word import means we import the namesapce System.IO into the program. Once we have done that, we can declare a variable of the streamReader data type with the following statement:

Dim FileReader As StreamReader

If we don't include the Imports System.IO, we have to use the statement

Dim FileReader As IO.StreamReader each time we want to use the StreamReader class.

Now, start a new project and name it in whatever name you wish, we named it TxtEditor
here. Now, insert the OpenFileDialog control into the form because we will use it to read the file from the storage device. The default name of the OpenFileDialog control is OpenFileDialog1, you can use this name or you can rename it with a more meaningful name. The OpenFileDialog control will return a DialogResult value that can determine whether the user clicks the OK button or Cancel button. We will also insert a command button and change its displayed text to 'Open'. The user can use it to open and read a certain text file. The following statement will accomplish the task above.

```
Dim results As DialogResult
results = OpenFileDialog1.ShowDialog
If results = DialogResult.OK Then
'Code to be executed if OK button was clicked
Else
'Code to be executed if Cancel button was clicked
End If
```

Next, we insert a textbox ,name it TxtEditor and set its Multiline property to true. It is used for displaying the text from a text file. We also insert a button and name it BtnOpen. In order to read the text file, we need to create a new instant of the streamReader and connect it to a text file with the following statement:

```
FileReader = New StreamReader(OpenFileDialog1.FileName)
```

In addition, we need to use the ReadToEnd method to read the entire text of a text file
and display it in the text box. The syntax is:

```
TxtEditor.Text = FileReader.ReadToEnd( )
```

Lastly, we need to close the file by using the Close() method. The entire code is shown
in the box below:

The Code

```
Imports System.IO
Public Class Form1
Private Sub BtnOpen_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnOpen.Click
Dim FileReader As StreamReader
Dim results As DialogResult
results = OpenFileDialog1.ShowDialog
If results = DialogResult.OK Then
FileReader = New StreamReader(OpenFileDialog1.FileName)
TxtEditor.Text = FileReader.ReadToEnd()
FileReader.Close()
End If
End Sub
```

6.3 Writing to a Text File

Writing a text file means storing the text entered by the user via the textbox into a storage device such as a hard drive. It also means saving the file. To accomplish this task, we need to deploy the **StreamWriter** Class. You also need to insert the SaveFileDialog control into the form as it is used to save the data into the storage unit like a hard drive. The default name for the SaveFileDialog control is SaveFileDialog1.

We also insert another button and name it as BtnSave. The Code is the same as the code for reading the file, you just change the StreamReader to StreamWriter, and the method from ReadToEnd to Write. The code is shown overleaf.

The code

```
Imports System.IO
Public Class Form1
Private Sub BtnSave_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
Dim FileWriter As StreamWriter
Dim results As DialogResult
results = SaveFileDialog1.ShowDialog
```

```
If results = DialogResult.OK Then
FileWriter = New StreamWriter(SaveFileDialog1.FileName, False)
FileWriter.Write(TxtEditor.Text)
FileWriter.Close()
End If
End Sub
```

When you click the save button, the program will prompt you to key in a file name and the text will be save as a text file. Finally, you can combine the two programs together and create a text editor that can read and write text file.

Summary

In this chapter, you learned how to create program to read and write files

_ In section 6.1, you learned the basic principle of handling reading and saving
data to storage unit of a computer.

_ In section 6.2, you learned how to write code to read text file. You learned how
to import the namespace System.IO and include the streamReader class in your
code.

_ In section 6.3, you learned how to design program to write text file to the
storage unit of the computer. You learned how to include StreamWriter class in
your code so that your program can write file to the computer.