

## 1. MANEJO DE ARCHIVOS

Muchos de los programas que se crean con Visual Basic .NET necesitan interactuar con datos del exterior, procesarlos para luego mostrarlos en un formulario, guardarlos en archivos de texto, en una hoja de Excel, en un archivo de Word, enviarlos a la red o simplemente imprimirlos en papel, etc. Se usan archivos para conservar a largo plazo grandes cantidades de datos. Los datos guardados en archivos se conocen como datos persistentes. Los computadores guardan los archivos en dispositivos de almacenamiento secundario como discos magnéticos, ópticos y cintas magnéticas.

Para procesar archivos en Visual Basic .NET se debe hacer uso del espacio de nombres<sup>5</sup> **System.IO**. El espacio de nombres **System.IO** contiene enumeraciones para la apertura de archivos, el acceso a los archivos, el uso compartido de archivos, además, de clases para las operaciones de rutas de acceso y la manipulación de flujos de datos.

### 1.1 Clase File

Se puede utilizar la clase **File** para operaciones como copiar, mover, cambiar el nombre, crear, abrir, eliminar y anexar texto a archivos de texto plano. También con la clase **File** se puede obtener y definir atributos del archivo o información relacionada con la hora y fecha de creación, el acceso y la escritura en un archivo.

**Stream** es la clase base de todas las secuencias de flujos de datos. Una secuencia es una abstracción de una secuencia de bytes, como un archivo, un dispositivo de entrada/salida, un canal de comunicación interprocesos o un socket TCP/IP. La clase **Stream** y sus clases derivadas proporcionan una visión genérica de diferentes tipos de entrada y salida, aislando al programador de los detalles específicos del sistema operativo y sus dispositivos subyacentes.

Algunos métodos de la clase **File** son:

**Tabla 1.1 Métodos de la clase File.**

Método	Proceso
CreateText(ruta)	Crea o abre un archivo para escribir texto.
Create(ruta)	Crea un archivo en la ruta especificada
AppendText	Anexa texto a un archivo de texto existente.
Delete	Elimina un archivo existente.
Copy	Copia un archivo
Exists	Determina si existe un archivo específico.
Open	Abre un archivo de texto existente
Move	Mueve un archivo a un sitio específico.
GetCreateTime	Devuelve la hora y la fecha de la creación de un archivo.
Replace	Reemplaza el contenido de un archivo en otro.

---

<sup>5</sup> Se usan para agrupar clases y otros tipos de datos que estén relacionados entre sí.

### 1.1.1 Ejemplo clase File

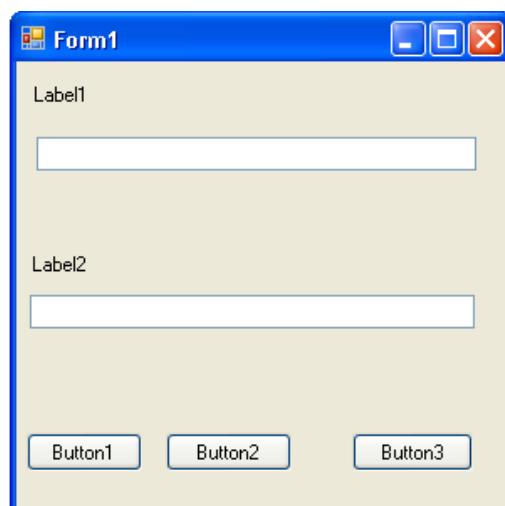
Hacer un nuevo proyecto llamado **GuardarábrirConFile** y realizar un programa que permita a un usuario guardar información en un archivo de texto plano, recuperar la información de éste, así como adicionarle información utilizando la clase **File**.

**NOTA:** Si lo considera necesario puede revisar el **Anexo A**, donde se explica brevemente la creación, la interfaz de usuario, el establecimiento de propiedades a los controles, la escritura de código, la ejecución de un nuevo proyecto Windows Forms.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 2 **Label**, 2 **TextBox**, 3 **Button**.

**Figura 1.1** Interfaz de usuario (**GuardarábrirConFile**).



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Establezca las siguientes modificaciones a las propiedades en los controles:

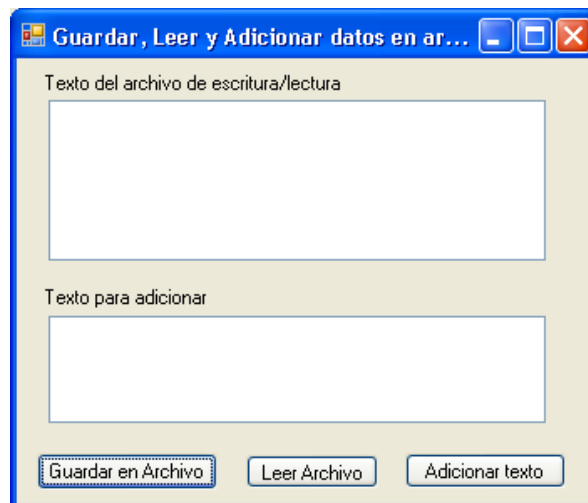
**Tabla 1.2** Propiedades de los controles del proyecto **GuardarábrirConFile**.

Control	Propiedad	Valor
Label1	Name	etiquetaver
	Text	Texto del archivo de escritura/lectura.
Label2	Name	etiquetaadicion
	Text	Texto para adicionar
TextBox1	Name	texto
	Text	En blanco
	Multiline	true
TextBox2	Name	textoadicional
	Text	En blanco
	Multiline	true
Button1	Name	botonguardar

	Text	Guardar en archivo
Button2	Name	botonabrir
	Text	Abrir archivo
Button3	Name	botonadicionar
	Text	Adicionar texto
Form1	Name	formulario
	Text	Guardar, leer y adicionar datos en archivo de texto.

La interfaz de usuario queda como se muestra en la siguiente figura:

**Figura 1.2 Interfaz de usuario, propiedades modificadas (GuardarAbrirConFile).**



- **Escribir código**

- Antes de la apertura de la clase se debe importar el espacio de nombres **System.IO**. Dé doble clic sobre el formulario y busque **Public class formulario** y antes de este código escriba **imports System.IO**. El código queda de la siguiente manera:

```
imports System.IO
Public Class formulario
.....
End Class
```

Se importa el espacio de nombres **System.IO** para poder manipular los métodos de la clase **File**.

- En modo diseño del formulario, seleccione el objeto **botonguardar**, dé doble clic para abrir el editor de código **botonguardar\_Click** y escriba el siguiente código:

```
Try
Dim escribir As StreamWriter
escribir = File.CreateText("c:\datosentexto.txt")
escribir.Write(texto.Text)
escribir.Close()
texto.Text = ""
```

```

    MsgBox("Texto Guardado", MsgBoxStyle.Information)
Catch ex As Exception
    MsgBox("Error al guardar el archivo", MsgBoxStyle.Critical)
End Try

```

Se define un bloque **Try-Catch** para atrapar errores y las acciones a seguir. Aunque no es necesario, es una buena práctica cuando se están realizando operaciones de entrada/salida para evitar salidas anormales del sistema. En el bloque **Try** se establece el código que realiza una tarea específica y que en un caso dado puede generar un error; en el bloque **Catch** irá el código que realizará las acciones a seguir en caso de error.

Dentro del bloque **Try** se crea un objeto llamado **escribir** de tipo **StreamWriter** (flujo de escritura). A dicho objeto se le asigna la creación del archivo de texto “**c:\datosentexto.txt**” por medio del método **createText** de la clase **File**. Utilizando el método **write** y enviándole como parámetro el contenido del control llamado **texto**, se guarda la información en el archivo especificado. Por otro lado, se cierra el archivo utilizando el método **close()** y se limpia el objeto **texto**. Por último se muestra una caja de mensajes con el mensaje “**Texto Guardado**”. En el bloque **Catch** se captura en el objeto **ex** de tipo **Exception** el error que se pueda generar y se mostrará el mensaje “**Error al guardar el archivo**”.

- c) Seleccione el objeto **botonabrir**, dé doble clic para abrir el editor de código **botonabrir\_Click** y escriba el siguiente código:

```

Try
    Dim leer As StreamReader
    leer = File.OpenText("c:\datosentexto.txt")
    texto.Text = leer.ReadToEnd
    leer.Close()
Catch ex As Exception
    MsgBox("Error al leer el archivo", MsgBoxStyle.Critical)
End Try

```

Dentro del bloque **Try** se crea un objeto llamado **leer** de tipo **StreamReader** (flujo de lectura). A dicho objeto se le establece la apertura del archivo de texto “**c:\datosentexto.txt**” por medio del método **openText** de la clase **File**. Utilizando el método **readToEnd** se lee el archivo de texto desde el inicio hasta el final y se le asigna su contenido al objeto **texto**. Por otro lado, se cierra el archivo utilizando el método **close ()**. En el bloque **Catch** si se genera algún error se mostrará el mensaje “**Error al leer el archivo**”.

- d) Seleccione el objeto **botonadicionar**, dé doble clic para abrir el editor de código **botonadicionar\_Click** y escriba el siguiente código:


```

Try
    texto.Text = " "
    Dim adicionartexto As StreamWriter
    adicionartexto = File.AppendText("C:\datosentexto.txt")
    adicionartexto.WriteLine(textoadicional.Text)
    adicionartexto.Close()
    textoadicional.Text = ""
Catch ex As Exception
    MsgBox("Error al adicionar inf. al archivo", MsgBoxStyle.Critical)
End Try

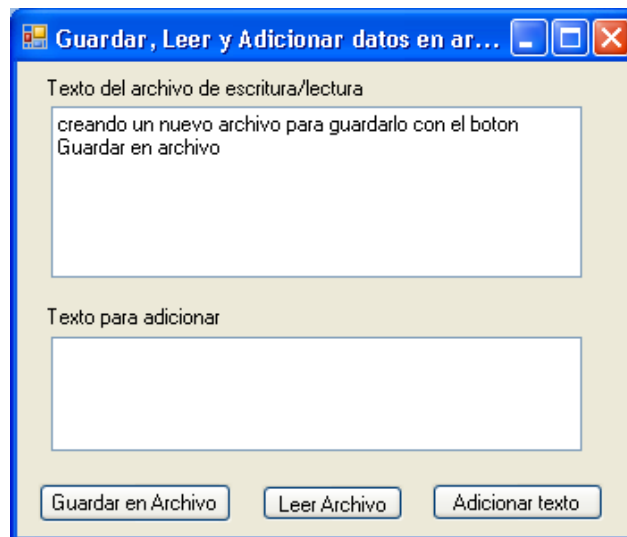
```

Dentro del bloque **Try** primero se coloca el objeto **texto** en blanco y luego se crea un objeto llamado **adicionartexto** de tipo **StreamWriter** (flujo de escritura). A dicho objeto se le asigna la propiedad de adición del texto al archivo "c:\datosentexto.txt" por medio del método **AppendText** de la clase **File**. Utilizando el método **WriteLine** y enviándole como parámetro el contenido del control llamado **textoadicional**, se adiciona la información al final del archivo especificado. Por otro lado se cierra el archivo utilizando el método **close ()** y se limpia el objeto **textoadicional**. En el bloque **Catch** si se genera algún error, se mostrará el mensaje "**Error al adicionar inf. al archivo**".

- **Ejecutar el proyecto**

Para ejecutar el proyecto pulse la tecla **F5** o el icono , se visualizará la figura 1.2. Al escribir "*creando un nuevo archivo para guardarlo con el botón Guardar en archivo*" en el objeto **texto**, dicha figura quedaría así:

**Figura 1.3 Ejecución aplicación AbrirGuardarConFile.**



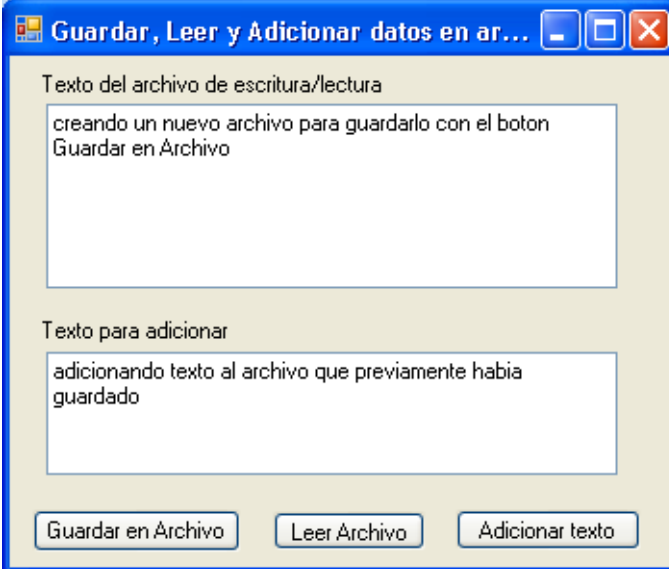
Al pulsar el botón **Guardar en Archivo**, se creará en **C:\** el archivo **datosentexto.txt** y el objeto **texto** se limpiará. Mostrándose la siguiente caja de texto:

**Figura 1.4 Mensaje de texto guardado en el archivo datosentexto.txt.**



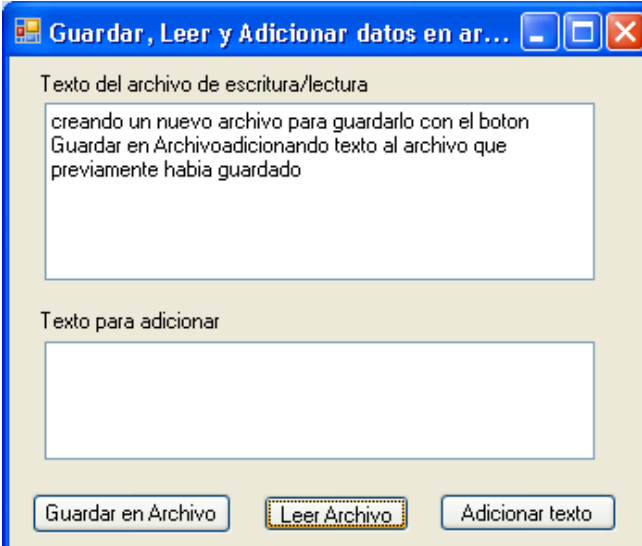
Pulse el botón **Aceptar** para cerrar dicha caja. Al pulsar el botón **Leer Archivo**, se visualizará nuevamente la figura 1.3. Si se adiciona el texto “*adicionando texto al archivo que previamente había guardado*” en el objeto **textoadicional**. El formulario deberá ser semejante a la siguiente figura:

**Figura 1.5 Formulario con el texto original y el texto adicional.**



Ahora pulse el botón **Adicionar Texto**, y nuevamente pulse el botón **Leer Archivo** para visualizar la siguiente figura:

**Figura 1.6 Lectura del archivo con el texto adicional.**



## 1.2 Clases StreamWriter y StreamReader

Las clases **StreamWriter** y **StreamReader** permiten las operaciones con archivos de texto plano. Para hacer uso de estas clases es necesario incluir el espacio de nombres **System.IO**. La Clase **StreamReader** (flujo de lectura) es una opción más para la manipulación de archivos de texto plano. Esta clase, entre otros, contiene el método **ReadToEnd** cuyo objetivo es leer un archivo desde una posición inicial hasta el final.

La Clase **StreamWriter** (flujo de escritura) está diseñada para la salida de caracteres. Esta clase contiene entre otros el método **Write** para escribir información en el archivo.

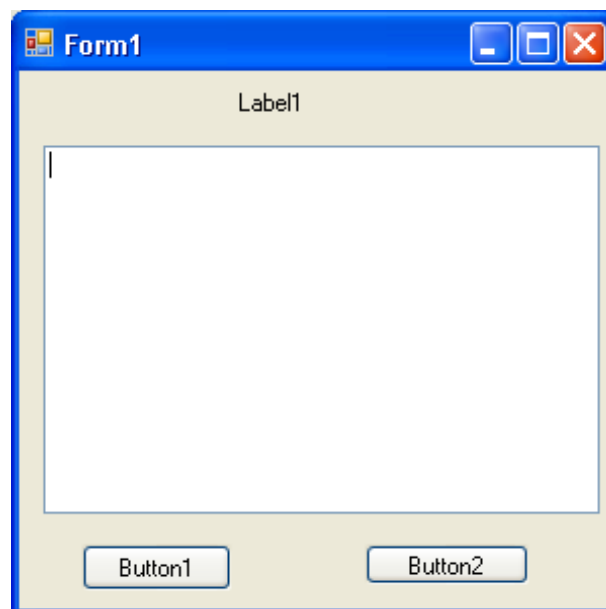
### 1.2.1 Ejemplo clases StreamWriter y StreamReader

Crear un proyecto llamado **GuardarAbrirArchivoTexto** y hacer un programa que permita a un usuario guardar información en un archivo de texto plano, así como poder abrirlo y visualizar el contenido de este, utilizando las clases **StreamWriter** y **StreamReader**.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 1 **Label**, 1 **TextBox**, 2 **Button**.

**Figura 1.7 Interfaz de usuario (GuardarAbrirArchivoTexto).**



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Para el caso del ejemplo, establezca las siguientes modificaciones a los controles:

**Tabla 1.3 Propiedades de los controles del proyecto GuardarAbrirArchivoTexto.**

Control	Propiedad	Valor
Label1	Name	titulo
	Text	Guardar y abrir un archivo de texto.
	Font – Bold	true
TextBox1	Name	texto
	Text	En blanco
	Multiline	true
Button1	Name	botonguardar
	Text	Guardar archivo de texto
Button2	Name	botonabrir
	Text	Abrir archivo de texto
Form1	Name	formulario
	Text	Guardar y abrir archivos de texto.

La interfaz de usuario queda como se muestra en la siguiente figura:

**Figura 1.8 Interfaz de usuario modificada (GuardarAbrirArchivoTexto).**



- **Escribir código**

- Seleccione el objeto **botonguardar**, dé doble clic para abrir el editor de código y escriba el siguiente código:

```
Try
    Dim archivo As New System.IO.StreamWriter("./archivotexto.txt")
    archivo.Write(texto.Text)
    archivo.Close()
    texto.Text = ""
Catch ex As Exception
    MsgBox("No se pudo guardar la informacion", MsgBoxStyle.Critical)
End Try
```

Se define la variable **archivo** asignándosele un espacio de memoria de tipo **System.IO.StreamWriter** (), al cual se le envía como parámetro el nombre del archivo (archivotexto.txt). Dicho archivo estará ubicado en la carpeta



`\bin\debug` del proyecto. Utilizando el método **Write** se escribe el contenido del objeto **texto** en el archivo. Por otro lado se cierra el archivo utilizando el método **close ()** y se limpia el objeto **texto**.

- b) Seleccione el objeto **botonabrir**, dé doble clic para abrir el editor de código y escriba el siguiente código:

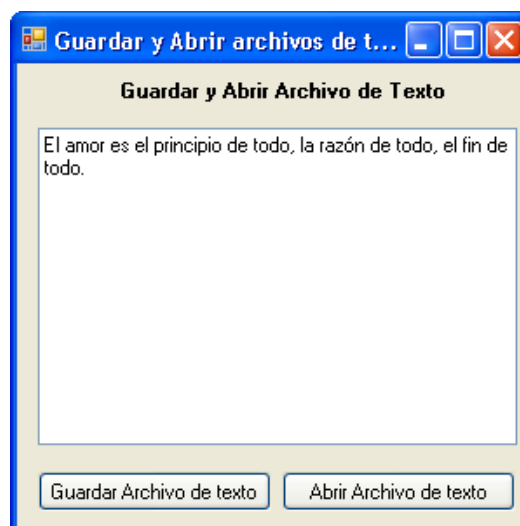
```
Try
    Dim miruta As String = ("./archivotexto.txt")
    Dim archivo As New System.IO.StreamReader(miruta)
    texto.Text = archivo.ReadToEnd
    archivo.Close()
Catch ex As Exception
    MsgBox("No se pudo guardar la informacion", MsgBoxStyle.Critical)
End Try
```

Se definen las variables **miruta** de tipo **String**, la cual se inicializa con la ruta y el nombre del archivo “**archivotexto.txt**” y la variable **archivo** asignándosele un espacio de memoria de tipo **System.IO.StreamWriter ()**. Utilizando el método **ReadToEnd** se lee el contenido del archivo y se le asigna al objeto **texto**. Por último se cierra el archivo.

- **Ejecutar el proyecto**

Al ejecutarse el proyecto en el entorno de desarrollo de Visual Basic.NET/2008, se visualizará la figura 1.8. Al adicionársele el texto “*El amor es el principio de todo, la razón de todo, el fin de todo*” en el objeto llamado **texto**, el formulario presentaría el siguiente aspecto:

**Figura 1.9 Ejecución aplicación GuardarábrirArchivoTexto.**



Al pulsar el botón con la etiqueta “**Guardar Archivo de texto**”, se creará un archivo de texto llamado **archivotexto.txt** y el objeto **texto** quedara en blanco. Al pulsar el botón **Abrir Archivo de texto**, se cargará en el objeto **texto** el contenido del archivo de texto.

### 1.3 Controles OpenFileDialog y SaveFileDialog

Los controles **OpenFileDialog** y **SaveFileDialog** del cuadro de herramientas de Visual Basic .NET /2008, son cuadros de diálogo que permiten abrir y guardar archivos, respectivamente. El control **OpenFileDialog** representa un cuadro de diálogo para seleccionar un archivo que será abierto. El control **SaveFileDialog** representa un cuadro de diálogo para guardar un archivo nuevo o sobrescribir en un archivo existente. Estos controles obtienen su funcionamiento de la clase abstracta **FileDialog**. La clase **FileDialog** es un cuadro de diálogo modal; por tanto, al mostrarse, bloquea el resto de la aplicación hasta que el usuario haya elegido un archivo o cancelado la operación.

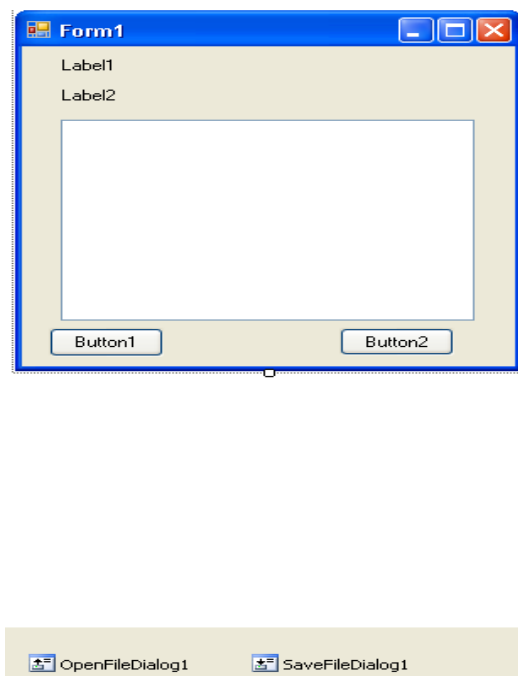
#### 1.3.1 Ejemplo con los controles OpenFileDialog y SaveFileDialog

Realizar un proyecto llamado **GuardarAbrirConControles** y diseñar un programa que permita a un usuario guardar información en un archivo de texto plano, así como poder abrirlo y visualizar el contenido de éste, utilizando los controles **OpenFileDialog** y **SaveFileDialog**.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 2 **Label**, 1 **TextBox**, 2 **Button**, 1 **OpenFileDialog** y 1 **SaveFileDialog**.

**Figura 1.10** Interfaz de usuario (GuardarAbrirArchivoConControles)



- **Establecer las propiedades de los objetos de la interfaz de usuario**

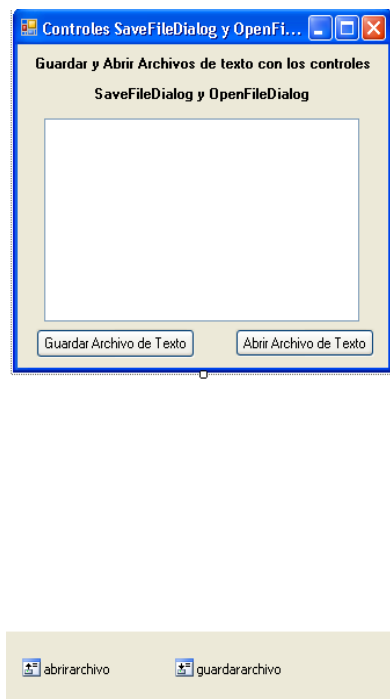
Establezca las siguientes modificaciones a los controles:

**Tabla 1.4 Propiedades de controles proyecto GuardarábrirArchivoConControles.**

Control	Propiedad	Valor
Label1	Name	titulo1
	Text	Guardar y abrir archivo de texto con los controles.
	Font – Bold	true
Label2	Name	titulo2
	Text	SaveFileDialog y OpenFileDialog.
	Font – Bold	true
TextBox1	Name	texto
	Text	En blanco
	Multiline	true
SaveFileDialog1	Name	guardarárchivo
OpenFileDialog1	Name	abrirárchivo
Button1	Name	botonguardar
	Text	Guardar archivo de texto
Button2	Name	botonabrir
	Text	Abrir archivo de texto
Form1	Name	formulario
	Text	Controles SaveFileDialog y OpenFileDialog.

La interfaz de usuario queda como se muestra en la siguiente figura:

**Figura 1.11 Interfaz de usuario modificada (GuardarábrirArchivoconControles).**



- **Escribir código**

- Selecione el objeto **botonguardar**, dé doble clic para abrir el editor de código y escriba el siguiente código:

```

guardararchivo.Filter = "Archivos de texto (*.txt)| *.txt"
guardararchivo.ShowDialog()
If guardararchivo.FileName <> "" Then
    Try
        Dim guardar As IO.StreamWriter
        guardar = New IO.StreamWriter(guardararchivo.FileName)
        guardar.Write(texto.Text)
        guardar.Close()
        texto.Text = ""
    Catch ex As Exception
        MsgBox("No se pudo guardar el archivo")
    End Try
Else
    MsgBox("Archivo no contiene información")
End If

```

En el anterior código se utiliza la propiedad **Filter** del control **guardararchivo** para que cuando se abra el cuadro de diálogo solamente se visualicen los archivos con extensión **.txt**. El método **ShowDialog** permite abrir el cuadro de diálogo. Utilizando la estructura **if** se pregunta si el nombre de archivo es diferente de vacío utilizando la propiedad **FileName**. Si se cumple la condición se crea un objeto llamado **guardar** de tipo **IO.StreamWriter** al cual se le asigna espacio de memoria enviándole como parámetro el nombre del archivo por intermedio de la propiedad **FileName** del control **guardararchivo**. Por otro lado, se utiliza el método **Write** para guardar el contenido del objeto **texto**; luego se cierra el objeto con el método **close ()** y por último se limpia el objeto **texto**. Si el nombre del archivo esté en blanco se mostrará un mensaje informando que el archivo esta vacío o no contiene información.

- b) Seleccione el objeto **botonabrir**, dé doble clic para abrir el editor de código y escriba el siguiente código:

```

abrirarchivo.Filter = "Archivos de texto (*.txt)| *.txt"
abrirarchivo.ShowDialog()
If abrirarchivo.FileName <> "" Then
    Try
        Dim verarchivo As New IO.StreamReader(abrirarchivo.FileName)
        texto.Text = verarchivo.ReadToEnd
        verarchivo.Close()
    Catch ex As Exception
        MsgBox("No se pudo abrir el archivo")
    End Try
Else
    MsgBox("Archivo no contiene información")
End If

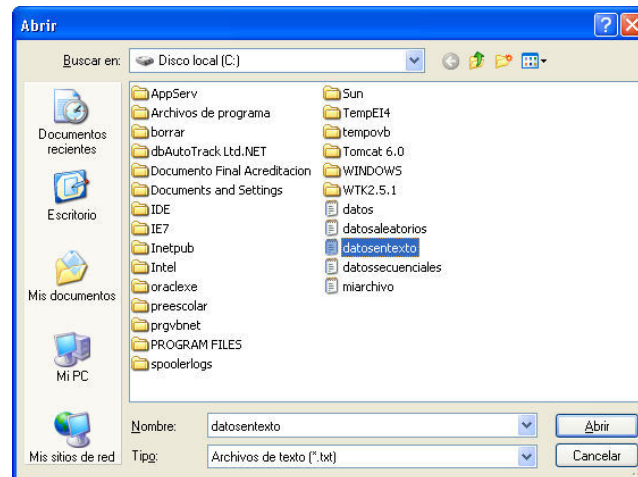
```

Utilizando la estructura **if** se pregunta si el nombre de archivo es diferente de vacío. Si se cumple la condición se crea un objeto llamado **verarchivo** de tipo **IO.StreamReader** al cual se le asigna espacio de memoria enviándole como parámetro el nombre del archivo por intermedio de la propiedad **FileName** del control **abrirarchivo**. Por otro lado, se utiliza el método **ReadToEnd** para leer el archivo y asignarle el contenido al objeto **texto**, luego se cierra el objeto con el método **close ()**. Si, por el contrario, no contiene información, se mostrará un mensaje informando que el archivo esta vacío.

- **Ejecutar el proyecto**

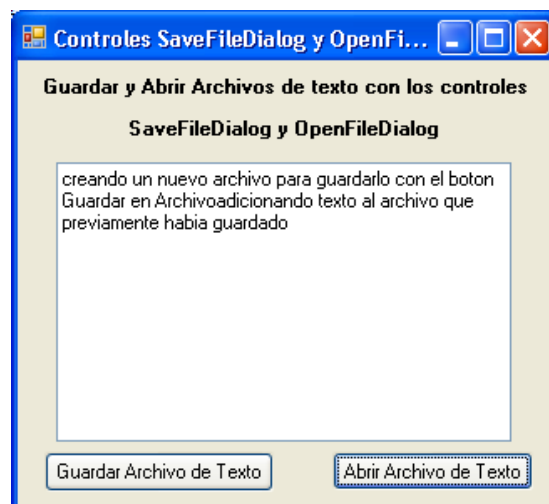
Al ejecutarse el proyecto en el entorno de desarrollo de Visual Basic.NET se visualizará la figura 1.11. Al pulsar el botón **Abrir Archivo de Texto** se visualizará el cuadro de diálogo de **Abrir**. Se debe seleccionar un archivo de texto y pulsar el botón del cuadro de diálogo **Abrir**.

**Figura 1.12 Cuadro de Diálogo control abrirarchivo.**



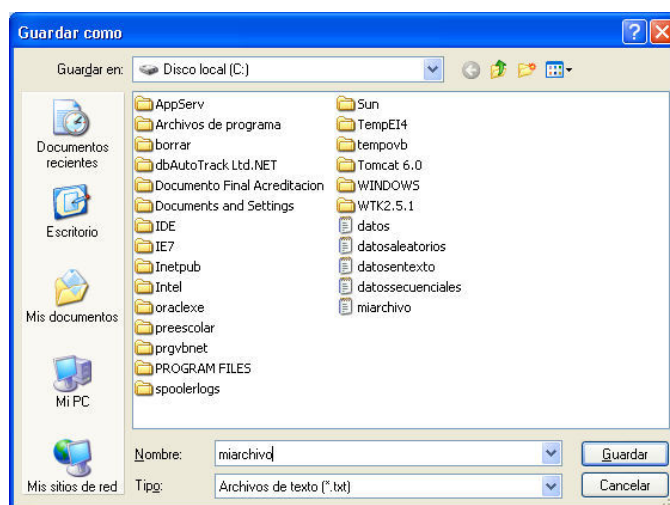
El formulario presentaría un aspecto similar:

**Figura 1.13 Formulario con archivo de texto abierto.**



Al pulsar el botón **Guardar Archivo de Texto** se visualizará el cuadro de diálogo de **Guardar como**, donde se deberá digitar el nombre del archivo a guardar y pulsar el botón del cuadro de diálogo “**Guardar**”. Si el archivo ya existe, preguntará si desea sobrescribirlo.

**Figura 1.14 Cuadro de diálogo control guardar archivo.**



## 1.4 Archivos secuenciales

Otra forma de guardar o leer información de un archivo de texto plano es utilizar archivos secuenciales. Los archivos secuenciales se denominan de esta manera porque para guardar y leer la información se realiza desde el principio hasta el final del archivo, es decir, para acceder a un dato que se encuentra en la mitad del archivo es necesario empezar a recorrerlo desde el principio hasta encontrar el dato, o sea en forma secuencial. El acceso secuencial funciona mejor cuando se desea procesar archivos únicamente de texto y no archivos en los que los datos se dividen en una serie de registros.

Cuando se abre un archivo para acceso secuencial, se debe especificar si se va a escribir en el archivo (**Input**), si se va a leer el archivo (**Output**) o si se va a adicionar información al archivo (**Append**). Para acceder a un archivo secuencial se utiliza la función **FileOpen**. Cuando se abre un archivo secuencial para **Input**, el archivo ya debe existir; de lo contrario, se producirá un error. No obstante, cuando se intenta abrir un archivo que no existe para **Output** o **Append**, la instrucción **FileOpen** primero crea el archivo y a continuación lo abre. Una vez que se abre un archivo para una operación **Input**, **Output** o **Append**, debe cerrarse con la instrucción **FileClose** antes de volver a abrirlo para realizar otro tipo de operación.

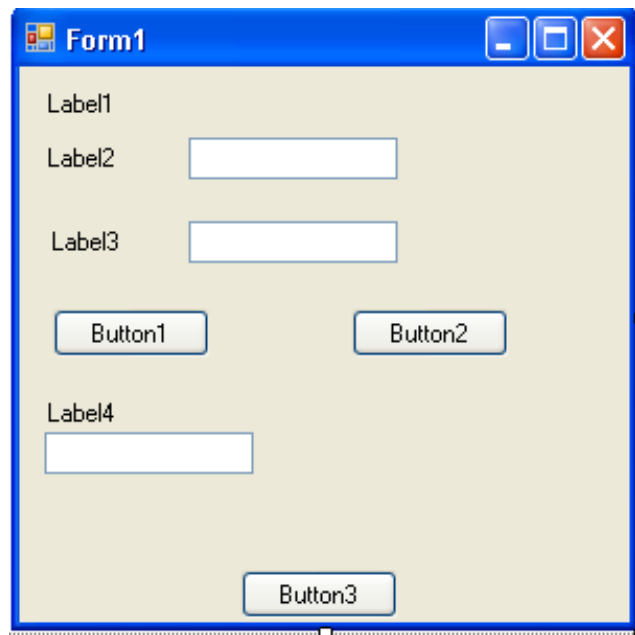
### 1.4.1 Ejemplo de archivos secuenciales

Crear un proyecto llamado **ArchivosSecuenciales** y realizar un programa que permita a un usuario guardar información en un archivo de texto plano, leer la información de éste, así como adicionarle más información a dicho archivo utilizando archivos secuenciales.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 4 **Label**, 3 **TextBox**, 3 **Button**.

**Figura 1.15 Interfaz de usuario (ArchivosSecuenciales).**



- **Establecer las propiedades de los objetos de la interfaz de usuario**

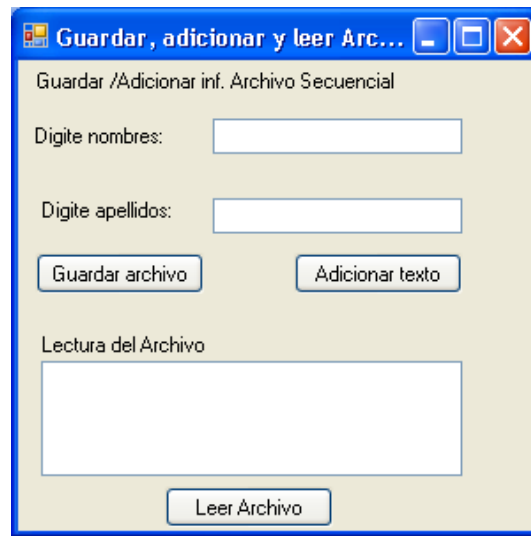
Establezca las siguientes modificaciones a los controles:

**Tabla 1.5 Propiedades de controles proyecto ArchivosSecuenciales.**

Control	Propiedad	Valor
Label1	Name	titulo
	Text	Guardar/adicionar inf. archivo secuencial.
Label2	Name	etiquetanombre
	Text	Digite nombres:
Label3	Name	etiquetaapellido
	Text	Digite apellidos:
Label4	Name	mostrar
	Text	Lectura del archivo:
TextBox1	Name	textonombre
	Text	En blanco
TextBox2	Name	textoapellido
	Text	En blanco
TextBox3	Name	textolectura
	Text	En blanco
	Multiline	true
Button1	Name	botonguardar
	Text	Guardar archivo
Button2	Name	botonadicionar
	Text	Adicionar texto
Button3	Name	botonabrir
	Text	Leer archivo
Form1	Name	formulario
	Text	Guardar, adicionar y leer archivo secuencial.

La interfaz de usuario queda como se muestra en la siguiente figura:

**Figura 1.16 Interfaz de usuario modificada (ArchivosSecuenciales).**



- **Escribir código**

- a) Seleccione el objeto **botonguardar**, dé doble clic para abrir el editor de código y escriba el siguiente código:

```
Try
    FileOpen(1, "c:\datossecuenciales.txt", OpenMode.Output)
    PrintLine(1, textonombre.Text, textoapellido.Text)
    textonombre.Text = ""
    textoapellido.Text = ""
    FileClose(1)
Catch ex As Exception
    MsgBox("No se pudo guardar el archivo")
End Try
```

En el anterior código se utiliza la función **FileOpen** con los parámetros: 1, un número de archivo libre, "c:\datossecuenciales.txt", nombre del archivo, **OpenMode.OutPut**, modo de apertura del archivo (salida). Con el método **PrintLine** se imprime en una línea del archivo el contenido de los objetos **textonombre** y **textoapellido**, luego se limpian dichos objetos y por último se cierra el archivo.

- b) Seleccione el objeto **botonadicionar**, dé doble clic para abrir el editor de código y escriba el siguiente código:

```
Try
    FileOpen(1, "c:\datossecuenciales.txt", OpenMode.Append)
    PrintLine(1, textonombre.Text, textoapellido.Text)
    textonombre.Text = ""
    textoapellido.Text = ""
    FileClose(1)
Catch ex As Exception
    MsgBox("No se pudo adicionar el archivo")
End Try
```



Lo único que cambia con respecto a la opción de **Guardar archivo** es el modo de apertura del archivo (**OpenMode.Append**), se cambia **OutPut** por **Append** (adicionar). Este modo de apertura permite guardar más información al final del archivo especificado.

- c) Seleccione el objeto **botonabrir**, de doble clic para abrir el editor de código y escriba el siguiente código:

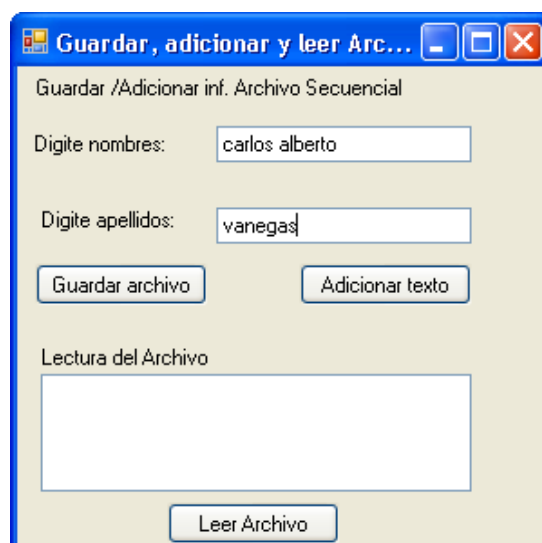
```
Try
    FileOpen(1, "c:\datossecuenciales.txt", OpenMode.Input)
    Dim linea As String
    Do Until EOF(1)
        linea = LineInput(1)
        textolectura.Text = textolectura.Text & linea & vbCrLf
    Loop
    FileClose(1)
Catch ex As Exception
    MsgBox("No se pudo abrir el archivo")
End Try
```

Se utiliza la función **FileOpen** con los parámetros: 1, un número de archivo libre, "c:\datossecuenciales.txt", nombre del archivo, **OpenMode.Input** modo de apertura del archivo (entrada). Se define una variable llamada **linea** de tipo **String**, la cual servirá para guardar cada línea del archivo. Por otro lado se crea un ciclo (**Do Until – Loop**) que recorrerá el archivo hasta el final (**EOF**) línea por línea y utilizando el método **LineInput** () se le asignará la información leída al objeto **textolectura**, por último se cierra el archivo.

- **Ejecutar el proyecto**

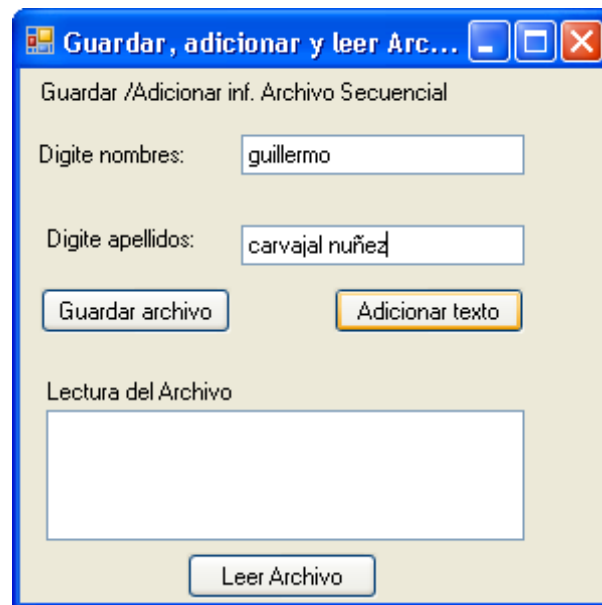
Al ejecutarse el proyecto se visualizará la figura 1.16. Al escribir un nombre y un apellido en las respectivas cajas de texto y pulsar el botón **Guardar archivo**, se guardará dicha información en el archivo "c:\datossecuenciales.txt". El formulario con la información quedaría de la siguiente forma:

**Figura 1.17 Formulario con información de nombre y apellido.**



Si se desea adicionar más información al archivo de texto, escriba nuevamente los nombres y apellidos correspondientes y pulse el botón **Adicionar texto**, se podría obtener la siguiente pantalla:

**Figura 1.18** Formulario con información para adicionar.



Guardar, adicionar y leer Arc...

Guardar /Adicionar inf. Archivo Secuencial

Digite nombres: guillermo

Digite apellidos: carvajal nuñez

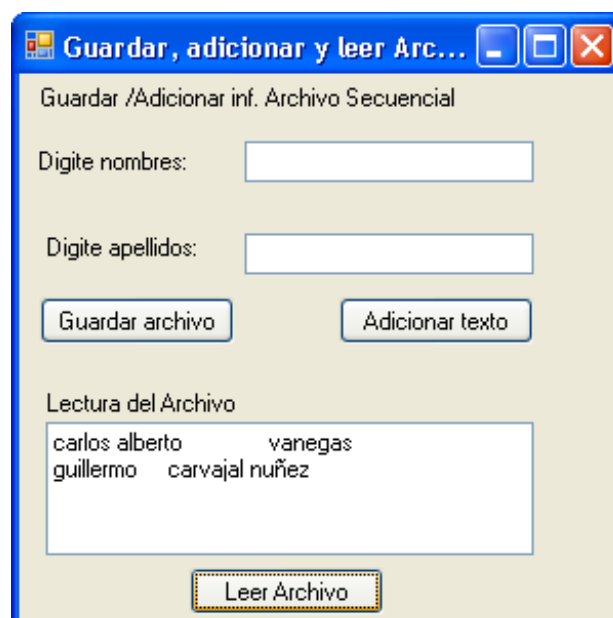
Guardar archivo Adicionar texto

Lectura del Archivo

Leer Archivo

Al pulsar el botón **Leer Archivo** se leerá el contenido del archivo de texto, como se muestra en la siguiente figura.

**Figura 1.19** Formulario con la información del archivo.



Guardar, adicionar y leer Arc...

Guardar /Adicionar inf. Archivo Secuencial

Digite nombres:

Digite apellidos:

Guardar archivo Adicionar texto

Lectura del Archivo

carlos alberto vanegas  
guillermo carvajal nuñez

Leer Archivo

## 1.5 Archivos binarios

También es posible guardar o leer información desde un archivo con acceso binario. Con los archivos de acceso binario se puede almacenar información ya sea numérica, de cadena o de ambas. Este tipo de archivo no requiere campos de longitud fija. No obstante, es necesario conocer cómo se escribieron exactamente los datos en el archivo para poder recuperarlos correctamente. Por ejemplo, si almacena una serie de productos y unas cantidades, debe tener en cuenta que el primer campo (producto) es texto y el segundo (cantidades) es numérico.

Para abrir un archivo para acceso binario, se recurre a **OpenMode.Binary** con la instrucción **FileOpen**. Una vez abierto, para escribir en el archivo se utiliza las funciones **FilePut** y **FilePutObject**. Para leer un archivo se manejan las funciones **FileGet** y **FileGetObject**.

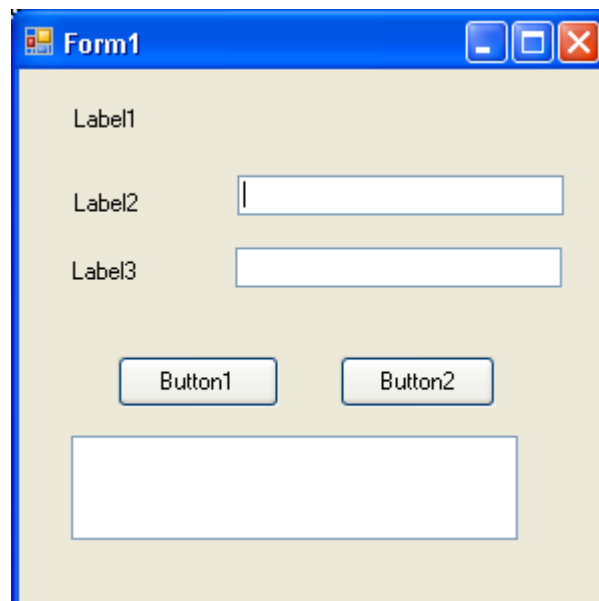
### 1.5.1 Ejemplo de archivos binarios

Realizar un proyecto llamado **ArchivosBinarios** y hacer un programa que permita a un usuario guardar información en un archivo de texto plano, leer la información de este, utilizando archivos binarios.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 3 **Label**, 3 **TextBox**, 2 **Button**.

**Figura 1.20** Interfaz de usuario (ArchivosBinarios).



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Establezca las siguientes modificaciones a los controles:

**Tabla 1.6 Propiedades de controles proyecto ArchivosBinarios.**

Control	Propiedad	Valor
Label1	Name	titulo
	Text	Guardar/adicionar inf. archivo binario.
Label2	Name	etiquetaproducto
	Text	Digite producto:
Label3	Name	etiquetacantidad
	Text	Digite cantidad:
TextBox1	Name	textoproducto
	Text	En blanco
TextBox2	Name	textocantidad
	Text	En blanco
TextBox3	Name	textolectura
	Text	En blanco
	Multiline	true
Button1	Name	botonguardar
	Text	Guardar archivo
Button2	Name	botonabrir
	Text	Leer archivo
Form1	Name	formulario
	Text	Guardar/leer archivo binario.

La interfaz de usuario queda como se muestra en la siguiente figura:

**Figura 1.21 Interfaz de usuario modificada (ArchivosBinarios).**



- **Escribir código**

a) Definir una estructura llamada **Articulo** después de **Public class formulario**:

```
Structure Articulo
    <VBFixedString(30)> Dim producto As String
    <VBFixedString(10)> Dim cantidad As String
End Structure
```

Se define una estructura (**Structure**) llamada **Articulo** que contendrá dos tipos de datos **String producto** y **cantidad**. Se utiliza el atributo **VBFixedString ()** para definir la longitud máxima de los datos.

- b) Seleccione el objeto **botonguardar**, dé doble clic para abrir el editor de código y escriba el siguiente código:

```
Dim nombrearchivo As String
Dim numeroarchivo As Short
Dim inventario As Articulo
inventario.producto = textoproducto.Text
inventario.cantidad = textocantidad.Text
numeroarchivo = FreeFile()
nombrearchivo = "c:\datosaleatorios.txt"
FileOpen(numeroarchivo, nombrearchivo, OpenMode.Binary)
FilePut(numeroarchivo, inventario)
FileClose(numeroarchivo)
```

Se crean las variables **nombrearchivo** de tipo **String**, **numeroarchivo** de tipo **Short** e **inventario** de tipo **Articulo**. Se le asigna la información capturada en los objetos **textoproducto** y **textocantidad** a los datos de la estructura **Articulo** utilizando la instancia **inventario**. La variable **numeroarchivo** es inicializada con un número de archivo libre utilizando la función **FreeFile ()**, como también se le asigna a la variable **nombrearchivo** la ruta y el nombre del archivo donde se guardará la información. Se abre el archivo en modo binario utilizando la función **FileOpen** que recibe como parámetros: un número de archivo libre (**numeroarchivo**), el nombre del archivo (**nombrearchivo**), la apertura del archivo binario (**OpenMode.Binary**). Con **FilePut** se escriben los valores que contiene la estructura **inventario** en el archivo binario y por último se cierra el archivo.

- c) Seleccione el objeto **botonabrir**, dé doble clic para abrir el editor de código y escriba el siguiente código:

```
Dim nombrearchivo As String
Dim numeroarchivo As Short
numeroarchivo = FreeFile()
nombrearchivo = "c:\datosaleatorios.txt"
FileOpen(numeroarchivo, nombrearchivo, OpenMode.Binary)
textolectura.Text = New String("c", 50)
FileGet(numeroarchivo, textolectura.Text)
FileClose(numeroarchivo)
```

Como se puede apreciar, las primeras cinco líneas son exactamente iguales a la opción de guardar. Se le asigna al objeto **textolectura** en su propiedad **text** un espacio de memoria de tipo **char** ("c"), con un espacio para 50 caracteres. Se utiliza la función **FileGet** para leer el contenido del archivo, éste recibe como parámetros: el numero del archivo (**numeroarchivo**) y el objeto **textolectura**, el cual contendrá el contenido del archivo de texto. Por último se cierra el archivo.

- **Ejecutar el proyecto**

Al ejecutarse el proyecto en el entorno de desarrollo de Visual Basic.NET/2008, se visualizará la figura 1.21. Al escribir un producto y una cantidad en las respectivas cajas de texto y pulsar el botón **Guardar Archivo**, se guardará dicha información en el archivo c:\datosaleatorios.txt”. Al pulsar el botón **Leer Archivo**, el formulario con la información leída quedaría de la siguiente forma:

**Figura 1.22 Formulario con información de un producto y una cantidad.**



## **1.6 Importar y exportar datos de una hoja de Excel**

Hasta el momento se ha trabajado con archivos de texto plano en donde se ha podido guardar, adicionar y recuperar información. Con Visual Basic .NET también es posible manipular archivos con otro tipo de formato; por eso en este aparte se exportará datos desde un proyecto de Visual Basic .NET a un archivo de Excel y se importarán datos de un archivo de Excel a un proyecto de Visual Basic .NET.

### **16.1 Ejemplo importar y exportar datos de Excel**

Dentro de un proyecto llamado **ImportarExportarExcel**, hacer un programa que permita a un usuario exportar y/o importar datos de una hoja de Excel. Cuando se importe de Excel los datos se deberán mostrar en un control DataGridView de Visual Basic .NET, así como se deberá solicitar el rango de datos a importar. Al exportar se deberán enviar todos los datos que contenga el control DataGridView a un archivo de Excel.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 1 **Label**, 1 **TextBox**, 2 **Button** y 1 **DataGridView**.