aprenderaprogramar.com

Ficheros de texto con Visual Basic. System.IO StreamWriter, StreamReader. Write, Read, AppendText, WriteLine, ReadLine. Ejemplos prácticos. (CU00329A-2)

Sección: Cursos

Categoría: Curso Visual Basic Nivel I

Fecha revisión: 2029

Autor: Mario R. Rancel

Resumen: Entrega nº28 del Curso Visual Basic Nivel I

29

OPERAR CON FICHEROS

Es frecuente necesitar guardar información en ficheros o recuperar información desde ficheros con Visual Basic. La forma de hacerlo en general es " Abrir -- > Operar -- > Cerrar", pero la sintaxis concreta a utilizar depende de la versión de Visual Basic que estemos empleando:

Para las versiones menos recientes de Visual Basic: consulta la entrega CU00329A-1 de este curso (http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61)

Para las versiones más recientes de Visual Basic: describimos a continuación cómo debe procederse.

Nos vamos a centrar en describir **de forma práctica** como podemos operar con ficheros con el propósito de generar programas que nos permitan ver aplicaciones algorítmicas interesantes. No vamos a explicar todas las posibles formas de operar con ficheros ni todas las instrucciones relacionadas.

ABRIR UN FICHERO PARA ESCRIBIR EN ÉL

Para manipular información de un fichero, en concreto para guardar datos en él, lo primero que hemos de hacer es abrir el fichero. El proceso a seguir es:

Abrir el fichero --> Manipular datos (borrar lo existente y escribir, o añadir a continuación de lo existente) --> Cerrar el fichero

Open --> Manipular datos --> Close

La sintaxis que emplearemos será la siguiente:

```
Dim nombreInterno As New System.IO.StreamWriter("RutaDelFichero", TrueoFalse)
.
.
.
nombreInterno.Close()
```

nombreInterno: es el nombre que queramos darle a la representación interna que va a usar el programa del fichero. No es el nombre del fichero propiamente dicho, sino un nombre elegido por nosotros para referirnos al fichero.

System.IO.StreamWriter: es el tipo de objeto que el programa va a usar internamente para guardar la información asociada a un fichero que se abre para escritura (write).

RutaDelFichero: indica la ruta en que se localiza el fichero. Por ejemplo "C:\Users\Lenovo\Desktop\misdatosnet.dat" es una ruta. Si no especificamos una ruta completa sino una ruta como "misdatosnet.dat" el fichero se guardará en el directorio por defecto que esté empleando Visual Basic.

TrueoFalse: valor booleano donde especificamos si queremos tener activada (True) la identificación del juego de caracteres con que está codificado el archivo, o no tenerla activada (False). Nosotros mantendremos este parámetro como False.

Dim NombreInterno As New…: es la sintaxis con la que creamos el objeto que representa a nuestro fichero.

Concretando la escritura damos a continuación ejemplos prácticos:

a) Escritura de datos en archivo secuencial (creará el archivo si no existe).

```
Dim myFileToWrite As New System.IO.StreamWriter("C:\Users\Asus\Desktop\misdatosnet.dat", False)

        .
        .
        .
myFileToWrite.Close()
```

ABRIR UN FICHERO PARA LEER INFORMACIÓN DESDE ÉL

Para extraer información de un fichero, en concreto para recuperar datos desde él, lo primero que hemos de hacer es abrir el fichero. El proceso a seguir es:

Abrir el fichero --> Manipular datos (extraer información existente) --> Cerrar el fichero

Open --> Manipular datos --> Close

La sintaxis que emplearemos será la siguiente:

```
Dim nombreInterno As New System.IO.StreamReader("RutaDelFichero", TrueoFalse)
.
.
.
nombreInterno.Close()
```

nombreInterno: es el nombre que queramos darle a la representación interna que va a usar el programa del fichero. No es el nombre del fichero propiamente dicho, sino un nombre elegido por nosotros para referirnos al fichero.

System.IO.StreamReader: es el tipo de objeto que el programa va a usar internamente para guardar la información asociada a un fichero que se abre para extracción de datos (lectura, read).

RutaDelFichero: indica la ruta en que se localiza el fichero. Por ejemplo "C:\Users\Lenovo\Desktop\misdatosnet.dat" es una ruta. Si no especificamos una ruta completa sino una ruta como "misdatosnet.dat" el fichero se guardará en el directorio por defecto que esté empleando Visual Basic.

TrueoFalse: valor booleano donde especificamos si queremos tener activada (True) la identificación del juego de caracteres con que está codificado el archivo, o no tenerla activada (False). Nosotros mantendremos este parámetro como False.

Dim NombreInterno As New…: es la sintaxis con la que creamos el objeto que representa a nuestro fichero.

Concretando la escritura damos a continuación ejemplos prácticos:

Lectura de datos en archivo secuencial (habrá un error si el archivo no existe).

```
Dim myFileToWrite As New System.IO.StreamWriter("C:\Users\Asus\Desktop\misdatosnet.dat", False)
            .
            .
            .
myFileToWrite.Close()
```

Ya tenemos definido cómo abrimos y cerramos la comunicación con un archivo. Ahora tenemos que ver cómo manipulamos los datos.

ESCRITURA EN FICHEROS

Para escribir en ficheros disponemos de varias instrucciones o posibilidades. Vamos a explicar una forma sencilla de escribir sobre ficheros, basadas en los métodos Write para escribir datos, o WriteLine, para escribir datos línea a línea.

La sintaxis que emplearemos para escribir línea a línea en el fichero será:

```
nombreInterno.WriteLine (datoAEscribir)
```

Donde **nombreInterno** es el nombre interno que estamos usando en nuestro programa como representación simbólica del fichero y **datoAEscribir** es aquello que queremos escribir en el fichero (podrá ser un texto entrecomillado, o una variable alfanumérica o numérica. En el caso de una variable, en el fichero se escribirá el contenido de la variable).

La sintaxis que emplearemos para escribir sin diferenciar líneas será:

```
nombreInterno.Write (datoAEscribir)
```

EXTRAER DATOS DE FICHEROS

Para extraer información desde ficheros disponemos de varias instrucciones o posibilidades. Vamos a explicar una forma sencilla de escribir sobre ficheros, basándonos en los métodos Read para leer datos carácter a carácter, o ReadLine, para extraer el contenido de líneas completas.

La sintaxis que emplearemos para extraer línea a línea en el fichero será:

```
variableDondeExtraemos = nombreInterno.ReadLine ()
```

Donde **variableDondeExtraemos** es el nombre de la variable donde va a quedar almacenada la información contenida en la línea del fichero. Si sabemos el tipo de información contenida en el fichero podemos extraer al tipo de variable adecuada. Por ejemplo, un número entero podremos extraerlo a una variable Integer, o un texto a una variable tipo String. **nombreInterno** es el nombre interno que estamos usando en nuestro programa como representación simbólica del fichero. No siempre habrá que extraer el contenido a una variable. Por ejemplo podríamos escribir MsgBox ("Extraido " & nombreInterno.ReadLine(), o usarlo de diferentes maneras sin guardar la información en una variable. No obstante, hemos preferido escribirlo de esta manera porque es el uso más habitual.

La sintaxis que emplearemos para extraer carácter a carácter será:

```
variableDondeExtraemos = variableDondeExtraemos & chr(nombreInterno.Read ())
```

El método Read nos devuelve el valor entero que representa el carácter extraído desde el fichero. Por ejemplo a la letra A le corresponde el número 65. Para obtener el carácter A desde el número correspondiente escribimos chr(65). En este caso la sintaxis que hemos indicado es para añadir a una variable tipo String un carácter extraido desde el fichero. Este no es la única forma posible de utilizar el método Read(), es sólo un ejemplo. Normalmente la extracción carácter a carácter irá combinada con el uso de bucles.

Consideremos un programa a modo de ejemplo (ver el código indicado a continuación al mismo tiempo que se lee la explicación):

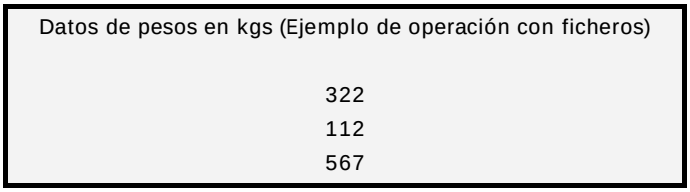
El código comienza con la declaración de las variables que van a intervenir en el programa. Se asignan valores a las variables Encabezado (tipo texto) y Dato(1), (2) y (3) (tipo Integer). A continuación se abre el archivo en la ruta especificada (en caso de que no existiera previamente el archivo es creado) para escritura (usamos un StreamWriter). A través del método WriteLine se escriben cuatro líneas y se cierra la comunicación con el método Close().

Se vuelve a abrir comunicación para lectura de datos (usamos un StreamReader), se asignan los datos a las variables Textoextraido y Datoextraido(1), (2) y (3) y se cierra la comunicación.

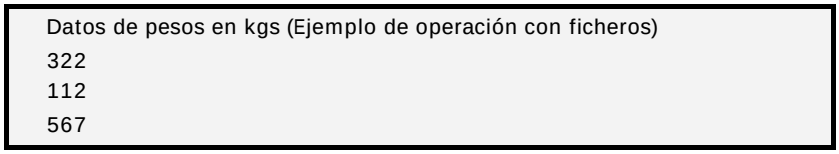
Ejecuta este programa (haz las adaptaciones que sean necesarias en lo relativo a la ruta en tu computador o a la versión de Visual Basic que estés empleando) y comprueba que se crea el archivo con el contenido esperado.

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
    Dim Encabezado As String
    Dim Dato(3) As Integer
    Dim Datoextraido(3) As Integer
    Dim Textoextraido As String

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load

        Dim myFileToWrite As New
System.IO.StreamWriter("C:\Users\Toshiba\Desktop\misdatosnet.dat", False)
        Encabezado = "Datos de pesos en kgs"
        Dato(1) = 322
        Dato(2) = 112
        Dato(3) = 567

        myFileToWrite.WriteLine(Encabezado & " (Ejemplo de operación con ficheros)")
        myFileToWrite.WriteLine(Dato(1))
        myFileToWrite.WriteLine(Dato(2))
        myFileToWrite.WriteLine(Dato(3))
        myFileToWrite.Close()

        Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Toshiba\Desktop\misdatosnet.dat", False)
        Textoextraido = myFileToRead.ReadLine()
        Datoextraido(1) = myFileToRead.ReadLine()
        Datoextraido(2) = myFileToRead.ReadLine()
        Datoextraido(3) = myFileToRead.ReadLine()
        myFileToRead.Close()

        Label1.Font = New Font("Arial", 12, FontStyle.Bold)
        Label1.TextAlign = ContentAlignment.MiddleCenter
        Label1.Text = vbCrLf & Textoextraido & vbCrLf & vbCrLf
        For i = 1 To 3
            Label1.Text = Label1.Text & Datoextraido(i) & vbCrLf
        Next i
    End Sub
End Class
```

La información extraída del archivo se muestra en pantalla, donde aparece:



Si abrimos directamente el archivo *misdatosnet.dat* con un visor como el bloc de notas, el contenido que apreciamos es el siguiente:



El tamaño del archivo es de *75 bytes* desglosados de forma aproximada en:

- *58 bytes* al texto de la primera línea.
- *2 bytes* al salto de línea y retorno de carro de la primera línea.
- *3 bytes* a la segunda línea.
- *2 bytes* al salto de línea y retorno de carro de la segunda línea.
- *3 bytes* a la tercera línea.
- *2 bytes* al salto de línea y retorno de carro de la tercera línea.
- *3 bytes* a la cuarta línea.
- *2 bytes* al salto de línea y retorno de carro de la cuarta línea.

Se comprueba lo indicado en relación al acceso secuencial: a pesar de guardar variables tipo *Integer*, el espacio ocupado no es el correspondiente a este tipo de variables (*2 bytes*) sino el equivalente a que fueran un texto en el archivo (*1 byte* por carácter).

Existen numerosos términos para el manejo de ficheros que no vamos a abordar.

Algunos términos tienen distinto significado o efecto en función del tipo de acceso a fichero que se utilice o en función de la versión de Visual Basic que se utilice.

La señal de Final de Archivo (EOF) nos resultará útil para extraer datos desde un archivo hasta llegar al final del mismo (punto donde finalizan los datos) la estudiaremos en el apartado correspondiente a "Herramientas de programación con Visual Basic".

Próxima entrega: CU00330A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61