aprenderaprogramar.com

Open, Close, Freefile. Write# y Input#. Abrir y cerrar ficheros con Visual Basic. Leer y guardar datos. Ejemplos prácticos. (CU00329A-1)

Sección: Cursos

Categoría: Curso Visual Basic Nivel I

Fecha revisión: 2029

Autor: Mario R. Rancel

Resumen: Entrega nº28-1 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: describimos a continuación cómo debe procederse.

Para las versiones más recientes de Visual Basic: consulta la entrega CU00329A-2 de este curso (http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61).

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.

OPEN, CLOSE, FREEFILE. INSTRUCCIONES WRITE# Y INPUT#. EJEMPLOS DE USO.

Para manipular información de un fichero, por ejemplo guardar datos en él o leer datos desde él, lo primero que hemos de hacer es abrir el fichero. El proceso a seguir es:

Abrir el fichero → Manipular datos (extraer, guardar, modificar, etc.) → Cerrar el fichero

Open → Manipular datos → Close

A su vez, cada vez que accedemos a un fichero usamos un número como identificador. Podemos considerar que el identificador es una línea de teléfono o canal entre el ordenador y el fichero. Así, para establecer una llamada (abrir un fichero) tenemos que buscar una línea que esté libre. El número de líneas es grande, pero no infinito. Por tanto, podemos abrir muchos archivos al mismo tiempo, pero no todos los que queramos.

La sintaxis que emplearemos será la siguiente:

```
Open [ruta de acceso] For [tipo de acceso] As [número de archivo]
        .
        .
        .
Close
```

Open: es la instrucción que da lugar a la apertura del fichero.

Ruta de acceso: es la ruta del archivo que queremos abrir para manipular. Se entiende que dicho archivo se encuentra en una unidad de disco (por ejemplo *C:\problema3.dat* corresponde al fichero *problema3.dat* que se encuentra en la unidad de disco C:). Si el archivo no existe al tratar de abrirlo para escribir datos será creado automáticamente, pero si tratamos de abrirlo para extraer datos aparecerá el mensaje "Error 53: Archivo no encontrado".

For [tipo de acceso]: indica qué acceso utilizamos para manipular el fichero. Vamos a considerar únicamente ficheros secuenciales, con lo que el tipo de acceso podrá ser ***Output*** (para escribir datos en ficheros, eliminando la información previa que pudiera existir), ***Append*** (para escribir datos, conservando la información previa contenida en el fichero) ó ***Input*** (para leer datos contenidos en un fichero).

As [número de archivo]: indicará el "número de línea" que vamos a emplear para la comunicación. Estableceremos un número de línea utilizando la función *FreeFile*, que devuelve un tipo *Integer* que indica el siguiente canal disponible para usar un *Open*.

Close: da lugar al cierre del fichero y a que el número de archivo o línea de comunicación quede libre para una posterior ocasión.

Concretando la escritura nos quedamos con:

a) Escritura de datos en archivo secuencial (no conserva información contenida en archivo).

```
Canal = Freefile
Open "C:\misdatos.dat" For Output As Canal
        .
        .
        .
Close
```

b) Escritura de datos en archivo secuencial (conserva información contenida en archivo).

```
Canal = Freefile
Open "C:\misdatos.dat" For Append As Canal
        .
        .
        .
Close
```

c) Lectura de datos de un archivo secuencial.

```
Canal = Freefile
Open "C:\misdatos.dat" For Input As Canal
        .
        .
        .
Close
```

Ya tenemos definido cómo abrimos y cerramos la comunicación con un archivo. Ahora tenemos que ver cómo manipulamos los datos. Para ello nos valemos de las instrucciones *Write#* y *Input#.* Existen distintas variantes para el uso de estas instrucciones. Vamos a considerar únicamente el caso de escritura y extracción de una variable por línea. La sintaxis que emplearemos será:

a) Para escribir en el fichero:

```
Write #[Número de archivo], [Dato a escribir]
```

El dato a escribir puede ser un texto entrecomillado, un valor numérico o una variable numérica o tipo texto.

b) Para extraer datos del fichero:

```
Input #[Número de archivo], [Nombre de variable]
```

El nombre de variable ha de corresponder a una variable tipo texto o tipo numérico que habremos declarado previamente. Es ideal saber qué tipo de dato contiene el fichero para hacer una extracción usando una variable adecuada. Si no dispusiéramos de esta información podríamos usar variables tipo *Variant,* pero no es lo más deseable.

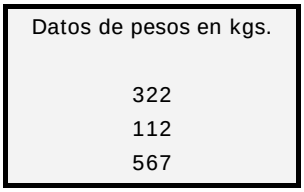
Consideremos un programa a modo de ejemplo (ver el código incluido a continuación al mismo tiempo que se leen las explicaciones):

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*). El valor de Canal se establece a través de *FreeFile*, con lo que hay una asignación automática de "línea de comunicación". A continuación se abre el archivo C:\misdatos.dat (en caso de que no existiera previamente el archivo es creado) para escritura (Output). A través de *Write#* se escriben cuatro líneas y se cierra la comunicación.

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

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim Canal%, i%
Dim Encabezado As String
Dim Dato(3) As Integer
Dim Datoextraido(3) As Integer
Dim Textoextraido As String

Private Sub Form_Load()
Show
Encabezado = "Datos de pesos en kgs"
Dato(1) = 322
Dato(2) = 112
Dato(3) = 567

Canal = FreeFile
Open "C:\misdatos.dat" For Output As Canal
Write #Canal, "Datos de pesos en kgs"
Write #Canal, Dato(1)
Write #Canal, Dato(2)
Write #Canal, Dato(3)
Close Canal

Canal = FreeFile
Open "C:\misdatos.dat" For Input As Canal
Input #Canal, Textoextraido
Input #Canal, Datoextraido(1)
Input #Canal, Datoextraido(2)
Input #Canal, Datoextraido(3)
Close Canal

Label1.Alignment = 2
Label1.FontBold = True
Label1 = vbCrLf & Textoextraido & vbCrLf & vbCrLf
For i = 1 To 3
   Label1 = Label1 & Datoextraido(i) & vbCrLf
Next i
```

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



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

```
"Datos de pesos en kgs."
322
112
567
```

La escritura mediante *Write#* de valores tipo texto queda delimitada por un entrecomillado que es eliminado en el momento de la extracción.

El tamaño del archivo es de *40 bytes* desglosados en:

- *23 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: CU00329A-2 y 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