Comencemos a programar con

# VBA - Access

## Entrega 18

# Trabajar con ficheros II

Eduardo Olaz

## Trabajando con Ficheros

En el capítulo anterior hemos visto la forma de buscar y renombrar ficheros, obtener datos sobre algunas de sus propiedades, crear carpetas, etc.

Un fichero informático no es más que una serie de datos Byte que están almacenados en un soporte magnético u óptico y que podemos manipular.

En esta entrega vamos a ver procedimientos básicos para manejar ficheros y los métodos para almacenar, exportar o recuperar datos.

Comenzaré con el acceso a ficheros de tipo **Secuencial**, es decir ficheros en los que para llegar a una posición concreta del mismo es preciso recorrerse todas las posiciones anteriores.

## Instrucción Open

Para poder acceder a un fichero lo primero que hay que hacer es abrirlo.

La instrucción **Open** es la que se encarga de activar las funciones de Entrada/Salida en un fichero.

Su sintaxis es

```
Open RutaDeAcceso For Modo [Access TipodeDeAcceso]
[Bloquear] As [#]NúmeroArchivo [Len=LongitudRegistro]
```

**RutaDeAcceso** es el nombre del fichero, y opcionalmente su ruta completa

Por ejemplo **Datos.txt** o `C:\Comercial\`**Datos.txt**

En el primer caso buscará en la carpeta definida por defecto; carpeta que podemos establecer o averiguar con las funciones **CurDir** y **ChDir**, tal y como explicamos en la entrega anterior. En el segundo lo hará en la carpeta `C:\Comercial`.

Si no existiera esa carpeta generaría el **error 76**: *No se ha encontrado la ruta de acceso*.

**Modo** indica la forma como vamos a acceder al fichero. Este parámetro es obligatorio. Si no se indicara definiría **For Random**. Se utiliza una de estas palabras clave:

| | |
|---|---|
| **Append** | Añadir datos secuencialmente a partir del final. |
| **Binary** | Acceso a ficheros binarios sin longitud fija. |
| **Input** | Acceso en modo lectura secuencial. |
| **Output** | Acceso en modo escritura secuencial. |
| **Random** | Acceso en modo aleatorio directo por número de registro. |

**TipodeDeAcceso**. Parámetro opcional que indica los tipos de actuación permitidas. Se utiliza una de estas palabras clave:

| | |
|---|---|
| **Read** | Permite efectuar operaciones de lectura. |
| **Write** | Operaciones de escritura. |
| **Read Write** | Lectura y Escritura. |

**Bloquear**. Especifica las acciones permitidas para otros procesos concurrentes

| | |
|---|---|
| **Shared** | Fichero compartido. |

| | |
|---|---|
| **Lock Read** | Bloqueado para Lectura. |
| **Lock Write** | Bloqueado para Escritura. |
| **Lock Read Write** | Bloqueado para Lectura y Escritura. |

**NúmeroArchivo**. Parámetro obligatorio que especifica un número entero como manejador de archivo. Cuando se efectúan operaciones de lectura y escritura hay que hacer mención a este número. Su valor debe estar entre **1** y **511**.

> *La función* **FreeFile***, que veremos a continuación, nos permite obtener un número libre para el manejador del archivo.*

**LongitudRegistro**. Opcional. Permite definir el número de caracteres almacenados en el buffer para el *Acceso Secuencial* a un fichero, y la longitud de un registro completo en el modo de *Acceso Aleatorio*. Estos dos modos de acceso los veremos en los apartados siguientes. Su valor debe ser inferior a **32.768**.

## Función FreeFile

La Función **FreeFile**, nos devuelve un valor del tipo **Integer** que podremos utilizar con la instrucción **Open** para abrir, leer, escribir y cerrar ficheros.

Su sintaxis es

```
FreeFile [(NúmeroIntervalo)]
```

**NúmeroIntervalo** es un parámetro opcional que puede tomar el valor **0** por defecto y **1**.

Si se pasa como parámetro 0, devuelve un número libre para el manejador de ficheros, en el rango de **1** a **255**. Si se pasa el parámetro 1, el rango va de 256 a 511.

El rango **256** a **511** se suele usar para ficheros que van a ser compartidos.

## Instrucción Print #

La Instrucción **Print #**, graba los datos tal y como se mostrarían, por ejemplo en la **Ventana Inmediato**, usando la sentencia **Print**.

Su sintaxis es

```
Print #NúmeroArchivo,[ListaAGrabar]
```

**NúmeroArchivo** es un entero entre **1** y **511**.

Se aconseja usar uno devuelto por la función **FreeFile**.

**ListaAGrabar** (Opcional) es una lista de datos que queremos incorporar al fichero abierto con **Open** .

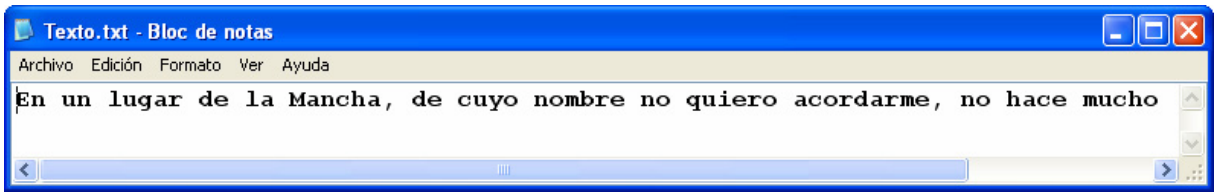
Este procedimiento graba un párrafo con las primeras frases de *Don Quijote de la Mancha*, en el fichero `Texto.txt`.

Finalmente cierra el fichero mediante **Close #NúmeroArchivo**.

```
Public Sub UsoDePrint()
    Dim lngFichero As Long
    Dim strFichero As String

    strFichero = CurrentProject.Path
```

```
    ' Definimos como carpeta actual la del fichero mdb
    ChDir (strFichero)
    strFichero = strFichero & "\Texto.txt"

    lngFichero = FreeFile
    ' Abrimos el fichero, como de Escritura _
      y si no existe lo crea
    Open strFichero For Output As #lngFichero
    ' Escribimos en el fichero
    Print #lngFichero, "En un lugar de la Mancha, ";
    Print #lngFichero, "de cuyo nombre no quiero acordarme,";
    Print #lngFichero, " no hace mucho que vivía . . ."

    Close #lngFichero
End Sub
```

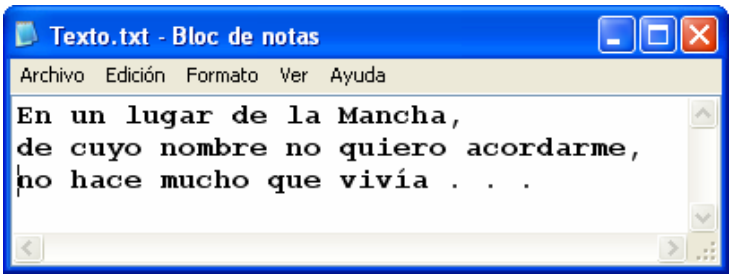
Ejecute este procedimiento y abra el archivo, por ejemplo con el **Bloc de Notas**.




Como puede apreciar, las diferentes partes del texto han sido colocadas una detrás de la otra. Esto ha sido así, porque he colocado un punto y coma después de cada texto.

```
    Print #lngFichero, "de cuyo nombre no quiero acordarme,";
```

Si no se hubiera colocado el punto y coma, cada tramo de texto se hubiera grabado en un párrafo diferente.




**Print#** añade detrás de cada texto un *Retorno de Carro* **VbCr** y un *Salto de Línea* **vbLf**, salvo que encuentre un punto y coma, o una coma tras el texto a grabar.

El conjunto **vbCr** y **vbLf** equivale a la constante **vbCrLf**.

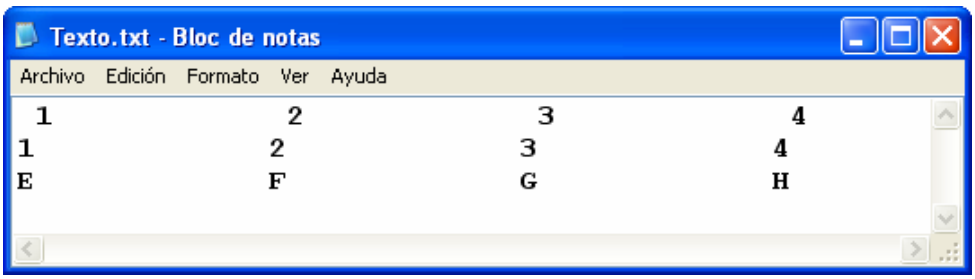
Si en lugar de poner un punto y coma, tras la expresión a grabar, pusiéramos una coma, **Print#** añadiría un **tabulador**.

Por ejemplo, la expresión:

```
    Print #lngFichero, 1, 2, 3, 4
    Print #lngFichero, "1", "2", "3", "4"
```

```
Print #lngFichero, "E", "F", "G", "H"
```

Grabaría un fichero tal y como se puede apreciar en la siguiente imagen




Nótese que deja un espacio de tabulación entre cada carácter.

Podemos apreciar que la primera sentencia **Print #**, deja un espacio en blanco delante de cada uno de los números.

En cambio la segunda, que graba los números como caracteres, no lo hace.

Con **Print #** se pueden usar, no sólo tabuladores mediante comas, sino que podemos usar las funciones **Tab** y **Spc.**

Los datos grabados con **Print #**, normalmente se leerán con **Line Input #** ó **Input #**. Estas instrucciones las veremos más adelante en esta misma entrega.

## Función Tab

La Función **Tab**, marca el número de columna en el que se empezará a grabar el primer carácter de la lista de datos o la expresión a grabar mediante la instrucción **Print #** o el método **Print**.

Sintaxis: **Tab**[(**n**)]

**n** representa el número de columna.

Por ejemplo si ejecutamos este código obtendremos un fichero como el de la imagen:


```
Public Sub UsoDeTab()
    Dim lngFichero As Long
    Dim strFichero As String

    strFichero = CurrentProject.Path
    ChDir (strFichero)
    strFichero = strFichero & "\FicheroConTab.txt"

    lngFichero = FreeFile(1)
```

```
    Open strFichero For Output As #lngFichero
    Print #lngFichero, Tab(1); "Nombre"; Tab(10); "Eduardo";
    Print #lngFichero, Tab(25); "Apellido1"; Tab(35); "Olaz"
    Close #lngFichero
End Sub
```

Como podemos apreciar los 4 datos se han grabado en las posiciones que hemos especificado: **1**, **10**, **25** y **35**.

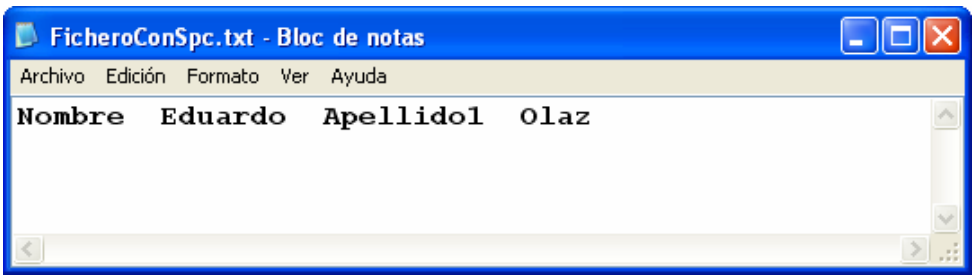
## Función Spc

La Función **Spc**, especifica el número de espacios en blanco antes de grabar el primer carácter de la lista de datos o expresión, mediante **Print #** o el método **Print**.

Sintaxis: **Spc**(**n**)

El parámetro n es obligatorio y marca el número de espacios.

```
Public Sub UsoDeSpc()
    Dim lngFichero As Long
    Dim strFichero As String

    strFichero = CurrentProject.Path
    ChDir (strFichero)
    strFichero = strFichero & "\FicheroConSpc.txt"

    lngFichero = FreeFile(1)
    Open strFichero For Output As #lngFichero
    Print #lngFichero, "Nombre"; Spc(3); "Eduardo";
    Print #lngFichero, Spc(3); "Apellido1"; Spc(3); "Olaz"
    Close #lngFichero
End Sub
```




**Spc** toma en cuenta la anchura de la línea, si ésta se ha definido mediante la instrucción **Width #**. Si no se ha definido esta anchura, **Spc** funciona de forma semejante a la función **Space**.

Para obtener más información sobre estas funciones, consulte la ayuda de Access.

## Instrucción Width #

La instrucción **Width #**, especifica el ancho de línea de salida a un archivo abierto mediante la instrucción Open.

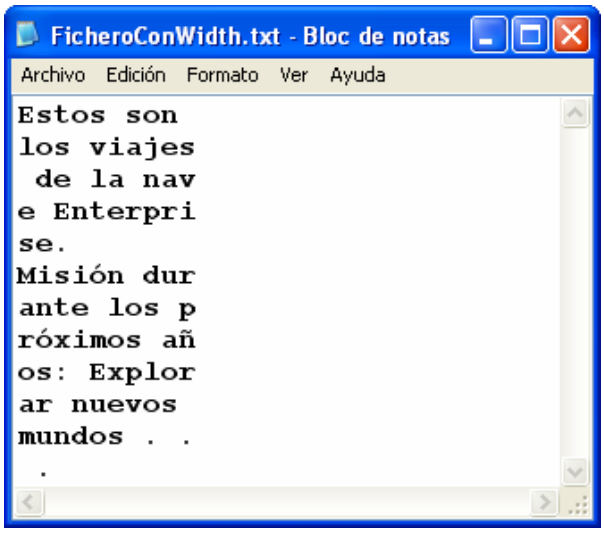
Sintaxis: **Width # NúmeroArchivo, Ancho**

El parámetro **Ancho** es obligatorio y puede ser un número ó una expresión numérica cuyo resultado esté entre **0** y **255**.

Si ejecutamos el siguiente código:

```
Public Sub UsoDeWidth()
    Dim lngFichero As Long
    Dim i As Long
    Dim strTexto As String
    Dim strFichero As String
    Dim strCaracter As String * 1

    strFichero = CurrentProject.Path
    ChDir (strFichero)
    strFichero = strFichero & "\FicheroConWidth.txt"
    strTexto ="Estos son los viajes de la nave Enterprise." _
        & vbCrLf _
        & "Misión durante los próximos años: " _
        & "Explorar nuevos mundos . . ."
    lngFichero = FreeFile(1)
    Open strFichero For Output As #lngFichero
    ' Establece el ancho de la línea a 10.
    VBA.Width# lngFichero, 10

    For i = 1 To Len(strTexto)
        ' Graba el texto carácter a carácter
        strCaracter = Mid$(strTexto, i, 1)
        Print #lngFichero, strCaracter;
    Next i
Close #lngFichero
End Sub
```

Obtendremos el siguiente resultado:




La utilización combinada de estos procedimientos, nos permite la construcción de diversos formatos de ficheros de texto y datos.

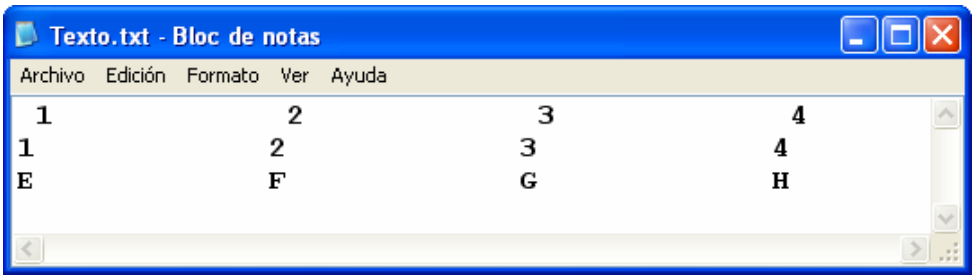
Especialmente las funciones **Tab**, **Spc** y la instrucción **Width**, posibilitan la elaboración de ficheros cuya característica sea la definición de campos de longitudes y posiciones establecidas, como ocurre por ejemplo con los ficheros para el intercambio de datos bancarios.

## Instrucción Write #

La instrucción **Write #**, es semejante a **Print#**, y sirve también para escribir datos en un fichero secuencial.

La diferencia fundamental con **Print #** es que **Write #**, escribe las cadenas marcándolas con el carácter Comilla doble como delimitador, además separándolos con una coma, y los números los escribe separándolos con comas.

En el ejemplo con Print #, nos generaba el siguiente fichero:

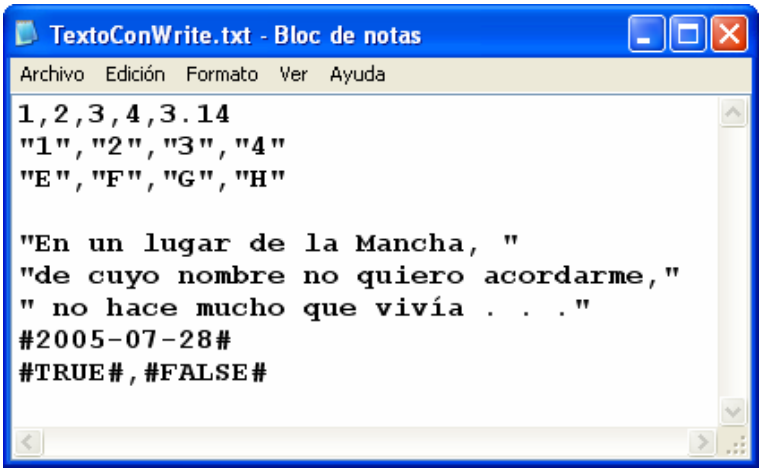



Vamos a cambiar ahora las instrucciones **Print #** por **Write #**

```
Public Sub UsoDeWrite()
    Dim lngFichero As Long
    Dim strFichero As String

    strFichero = CurrentProject.Path
    ' Definimos como carpeta actual la del fichero mdb
    ChDir (strFichero)
    strFichero = strFichero & "\TextoConWrite.txt"
```

```
    lngFichero = FreeFile
    ' Abrimos el fichero, como de Escritura _
      y si no existe lo crea
    Open strFichero For Output As #lngFichero
    ' Escribimos en el fichero

    Write #lngFichero, 1, 2, 3, 4, 3.14
    Write #lngFichero, "1", "2", "3", "4"
    Write #lngFichero, "E", "F", "G", "H"
    Write #lngFichero,
    Write #lngFichero, "En un lugar de la Mancha, "
    Write #lngFichero, "de cuyo nombre no quiero acordarme,"
    Write #lngFichero, " no hace mucho que vivía . . ."
    Write #lngFichero, " no hace mucho que vivía . . ."
   Write #lngFichero, Date
   Write #lngFichero, True, False
   Close #lngFichero
End Sub
```

El resultado será:




Podemos observar que efectivamente nos crea un fichero con los elementos separados por comas. Otro aspecto interesante es que, tanto las fechas como los valores booleanos, los delimita con almohadillas **#**.

Si hubiera que grabar un dato cuyo valor fuera **Null**, grabaría **#Null#**.

Un dato de Error, lo grabaría con el siguiente formato **#ERROR códigoerror#**

Si hubiera que grabar un dato cuyo valor fuera **Empty**, no grabaría nada en el fichero.

Los datos grabados con **Print #**, normalmente se leerán con **Line Input #** ó **Input #**. Estas instrucciones las veremos a continuación.

## Instrucciones Input # y Line Input #

La instrucción **Input**, lee datos de un archivo secuencial abierto y asigna esos datos a sus correspondientes variables.

Su sintaxis es:

**Input # NúmeroArchivo, *ListaDeVariables***

La lista de variables es una lista de ellas, separadas por comas.

La instrucción **Line Input #**, presenta una variación frente a **Input**.

Su sintaxis es:

**Line Input #númeroarchivo, VariableCadena**

**VariableCadena** es una variable de tipo **String** en la que se almacenarán los datos de una línea del fichero.

Veamos cómo funciona todo esto:

```
Public Sub LecturaFichero()
    Dim lngFichero As Long
    Dim strFichero As String
    Dim strLinea As String
    Dim strCliente As String
    Dim datAperturaCuenta As Date
    Dim curSaldoCuenta As Currency

    strFichero = CurrentProject.Path
    ' Definimos como carpeta actual la del fichero mdb
    ChDir (strFichero)
    strFichero = strFichero & "\ClientesBanco.dat"

    lngFichero = FreeFile
    ' Abrimos el fichero, como de Escritura _
      y si no existe lo crea
    Open strFichero For Output As #lngFichero
    ' Primero creamos el fichero ClientesBanco.dat _
      que luego leeremos
    Write #lngFichero, _
        "Pedro Rodríguez", #5/25/1983#, 100000.25
    Write #lngFichero, _
        "Juan Gorostiza", #12/30/2001#, 43235.37
    Write #lngFichero, _
        "Jon Estarriaga", #7/7/2004#, -105.85
    Write #lngFichero, _
        "Andrea Olazábal", #2/14/1995#, 128.36
```

```
    Close #lngFichero

    ' Ahora ya tenemos el fichero creado
    ' Primero Vamos a leerlo con Line Input
    lngFichero = FreeFile
    ' Abrimos el fichero, como de lectura
    Open strFichero For Input As #lngFichero
    ' Leemos las líneas del mismo
    ' Realiza el bucle hasta el final del fichero.
    ' EOF indica que hemos llegado al final
    Debug.Print
    Debug.Print "Fichero " & strFichero
    Debug.Print
    Debug.Print "Datos leídos con Line Input #"
    Do While Not EOF(lngFichero)
        ' Lee la línea y se la asigna a la variable String
        Line Input #lngFichero, strLinea
        Debug.Print strLinea
    Loop
    Close #lngFichero

    lngFichero = FreeFile
    ' Ahora vamos a leer el fichero con Input
    Open strFichero For Input As #lngFichero
    Debug.Print
    Debug.Print "Datos leídos con Input"
    ' Para ello deberemos asignárselos a sus variables
    Do While Not EOF(lngFichero)
        ' Lee la línea y se la asigna a las variables
        Input #lngFichero, _
             strCliente, datAperturaCuenta, curSaldoCuenta
        Debug.Print strCliente, _
                   datAperturaCuenta, _
                   curSaldoCuenta
    Loop
   Close #lngFichero
End Sub
```

Al ejecutar este código, nos mostrará en la ventana **[Inmediato]** lo siguiente:

```
Fichero C:\Curso_VBA\Capítulo_18\ClientesBanco.dat

Datos leídos con Line Input #
"Pedro Rodríguez",#1983-05-25#,100000.25
"Juan Gorostiza",#2001-12-30#,43235.37
"Jon Estarriaga",#2004-07-07#,-105.85
"Andrea Olazábal",#1995-02-14#,128.36

Datos leídos con Input
Pedro Rodríguez              25/05/1983     100000,25
Juan Gorostiza               30/12/2001     43235,37
Jon Estarriaga               07/07/2004    -105,85
Andrea Olazábal              14/02/1995     128,36
```

En el código y en el resultado, podemos apreciar la diferencia fundamental entre **`Line Input #`** e **`Input #`** la lectura de una línea completa, en el primer caso, y de los datos individuales en el segundo.

Lo que hemos visto hasta ahora se refiere a los llamados **Ficheros de acceso Secuencial**, en los que la información se va leyendo desde el inicio hasta el final, de manera secuencial.

Ya desde los tiempos de **BASICA**, **GWBasic** ó **QBasic**, existían los llamados **Ficheros de acceso aleatorio**, en los que se podía grabar o leer una información en una posición concreta.

Aunque está considerado como un formato “obsoleto”, vamos a analizarlo “por encima”.

## Ficheros de acceso Aleatorio

Para que VBA pueda leer un registro concreto de un fichero de acceso Aleatorio, primero debe conocer en qué posición del fichero debe empezar la lectura o escritura.

Para ello debe tener la siguiente información

- Qué tamaño tiene cada registro (los registros y los campos deben ser de una longitud determinada)
- Qué nº de registro debe leer o escribir

El tamaño del registro se especifica en el momento de la apertura del fichero.

```
Open "Fichero" For Random As #Numero Len = TamañoRegistro
```

**`Random`** especifica que se va a abrir el fichero en modo de acceso Aleatorio (modo por defecto de **`Open`** si no se especifica otro modo).

**`Len`** indica el tamaño del registro.

Para grabar los datos en un fichero de este tipo, debemos indicarle a VBA en qué posición vamos a grabarlos.

Si no indicamos una posición de registro, escribirá al final del fichero.

Para leer los datos de un fichero da acceso aleatorio se utiliza la instrucción **`Get`**; para escribir se usa la instrucción **`Put`**.

## Instrucción Get

La instrucción **Get**, lee datos de un archivo abierto y asigna esos datos a una variable.

Su sintaxis es:

**Get #NúmeroArchivo,*[NumeroRegistro]*,*Variable***

La variable suele ser una variable **Type** de tipo registro.

```
Open "Clientes.dat" For Random As #lngFichero _
        Len = Len(Datos)
Get #lngFichero, 2, Datos
Close #lngFichero
```

Tras esta instrucción, se habrán pasado los datos del registro Nº 2 a la variable **Datos**.

El numero de registro es opcional

## Instrucción Put

La instrucción **Put**, escribe datos en un archivo abierto , volcándolos desde una variable.

Su sintaxis es:

**Put #NúmeroArchivo,*[NumeroRegistro]*,*Variable***

La variable suele ser una variable **Type** de tipo registro.

Si quisiéramos grabar los datos contenidos en la variable **Datos**, en el registro Nº 2 de la tabla **Clientes.dat**, podríamos hacerlo así:

```
Open "Clientes.dat" For Random As #lngFichero _
        Len = Len(Datos)
Put #lngFichero, 2, Datos
Close #lngFichero
```

El procedimiento **PruebaFicheroRandom** graba 5 registros en **Clientes.dat** y luego los lee, mostrando los datos en la ventana de depuración:

```
Public Type ClienteBanco
    Nombre As String * 15
    Apellido1 As String * 15
    Apellido2 As String * 15
    Cuenta As Long
    Saldo As Currency
End Type

Public Sub PruebaFicheroRandom()
    GrabaFicheroRandom
    LeeFicheroRandom
End Sub
```

```
Public Sub GrabaFicheroRandom()
    Dim Cliente As ClienteBanco
    Dim i As Long
    Dim strNumero As String
    ' Rnd genera un número aleatorio entre 0 y 0.999999...
    ' Randomize Timer inicializa cada vez una secuencia _
      Diferente con Rnd
    Randomize Timer
    For i = 1 To 5
        strNumero = "_" & Format(i, "000")
        With Cliente
            .Nombre = "Nombre" & strNumero
            .Apellido1 = "Apellido1" & strNumero
            .Apellido2 = "Apellido2" & strNumero
            .Cuenta = Format(100000 * Rnd, "00000")
            .Saldo = Int(1000000 * Rnd) / 100
        End With
        GrabaCliente Cliente, i
    Next i
End Sub

Public Sub LeeFicheroRandom()
    Dim Cliente As ClienteBanco
    Dim i As Long
    For i = 1 To 5
    LeeCliente Cliente, i
        Debug.Print
        With Cliente
        Debug.Print "Cliente " _
                & CStr(i) & ": " _
                & Trim(.Nombre) & " " _
                & Trim(.Apellido1) & " " _
                & Trim(.Apellido2)
        Debug.Print "Cuenta " & .Cuenta
        Debug.Print "Saldo " & Format(.Saldo, "#,##0.00 €")
    End With
    Next i
End Sub

Public Sub GrabaCliente( _
```

```
                Datos As ClienteBanco, _
                Optional ByVal Posicion As Long = -1)
    On Error GoTo HayError
    Dim lngFichero As Long
    Dim strFichero As String
    lngFichero = FreeFile
    Open "Clientes.dat" For Random As #lngFichero _
        Len = Len(Datos)
    'Grabamos los datos
    If Posicion >= 0 Then
        Put #lngFichero, Posicion, Datos
    Else
        Put #lngFichero, , Datos
    End If
    Close #lngFichero
Salir:
    Exit Sub

HayError:
    MsgBox "Error al grabar " & strFichero _
            & vbCrLf _
            & Err.Description
    Resume Salir
End Sub

Public Sub LeeCliente( _
                Datos As ClienteBanco, _
                Optional ByVal Posicion As Long = -1)
    On Error GoTo HayError
    Dim lngFichero As Long
    Dim strFichero As String

    lngFichero = FreeFile
    Open "Clientes.dat" For Random As #lngFichero Len =
Len(Datos)
    'Leemos los datos
    If Posicion >= 0 Then
        Get #lngFichero, Posicion, Datos
    Else
        Get #lngFichero, , Datos
```

```
    End If
    Close #lngFichero
Salir:
    Exit Sub

HayError:
    MsgBox "Error al leer " & strFichero _
           & vbCrLf _
           & Err.Description
    Resume Salir
End Sub
```

Tras ejecutarse este código, mostrará algo así como:

```
Cliente 1: Nombre_001 Apellido1_001 Apellido2_001
Cuenta 73549
Saldo 5.571,35 €

Cliente 2: Nombre_002 Apellido1_002 Apellido2_002
Cuenta 41781
Saldo 2.964,22 €

Cliente 3: Nombre_003 Apellido1_003 Apellido2_003
Cuenta 96618
Saldo 9.775,93 €

Cliente 4: Nombre_004 Apellido1_004 Apellido2_004
Cuenta 6879
Saldo 8.564,24 €

Cliente 5: Nombre_005 Apellido1_005 Apellido2_005
Cuenta 99736
Saldo 5,18 €
```

## Función Seek

La instrucción **Seek**, devuelve un valor del tipo **Long** que nos indica el número de registro de la posición actual dentro de un fichero abierto.

Su sintaxis es:

**Seek(#NúmeroArchivo)**

Por ejemplo. Tras ejecutarse este código, **Seek** devolverá el valor 4.

En el caso de los ficheros de acceso **Random**, Seek devuelve la posición siguiente al último registro al que se ha accedido para lectura o escritura.

```
Public Sub PruebaSeek()
    Dim lngFichero As Long
    Dim Datos As ClienteBanco
    lngFichero = FreeFile
    Open "Clientes.dat" For Random As #lngFichero _
          Len = Len(Datos)
    Get #lngFichero, 2, Datos
    Debug.Print "La posición actual es el registro " _
                    & CStr(Seek(lngFichero))
    Close #lngFichero
End Sub
```

El resultado en la ventana de depuración será:

```
La posición actual es el registro 3
```

## Ejemplos de apertura de ficheros con OPEN

```
Open "Datos.txt" For Input As #1
```

Abre el fichero `Datos.txt` en modo secuencial para **Lectura**.

```
Open "Datos.app" For Output As #2
```

Abre el fichero `Datos.app` en modo secuencial para **Escritura**.

```
Open "DatosBinarios.dat" For Binary Access Write As #3
```

Abre el fichero `DatosBinarios.dat` en modo binario para sólo **Escritura**.

Para ver las posibilidades de manejar ficheros en modo Binario, consulte la ayuda de VBA.

```
Open "Datos.app" For Random As #4 Len = Len(Cliente)
```

Abre el fichero `Datos.app` en el modo de acceso **Aleatorio**.

Se supone que la estructura de los registros es idéntica a la estructura del tipo **`Cliente.`**

```
Open "Datos.txt" For Output Shared As #2
```

Abre el fichero `Datos.txt` en modo secuencial para **Escritura**.

Sin embargo, al no estar bloqueado, instrucción **`Shared,`** otro proceso puede manejarlo simultáneamente.

```
Open "Datos.txt" For Output Lock As #2
```

Abre el fichero `Datos.txt` en modo secuencial para **Escritura**.

El fichero quedará bloqueado, instrucción **`Lock`**, para el acceso simultáneo desde cualquier otro proceso.

```
Open "Datos.txt" For Output Lock Write As #2
```

Abre el fichero `Datos.txt` en modo secuencial para **Escritura**.

El fichero quedará bloqueado, instrucción **`Lock Write`**, para la escritura desde cualquier otro proceso.

## Nota sobre esta entrega

En esta entrega se han explicado formas de acceso a ficheros en lectura y escritura, que han sido clásicas en las versiones Basic de Microsoft, prácticamente desde sus inicios.

Aunque puedan considerarse como “obsoletas” su conocimiento es muy interesante para enfrentarse a casos especiales en los que se requiera elaborar o leer ficheros de datos específicos. Sería el caso definido en los cuadernos bancarios, Ficheros de texto con apuestas de quinielas de fútbol, etc.

Además podemos encontrarnos con aplicaciones “antiguas” que nos obliguen a leer sus datos, escritos en formatos especiales.

No he pretendido realizar un repaso exhaustivo de estos procedimientos, por lo que para la ampliación de conceptos remito a la ayuda de VBA o al MSDN de Microsoft.

Afortunadamente el mundo cambia y Microsoft nos ha aportado herramientas mucho más potentes que las aquí expuestas, como la utilización de **Schema.ini** que veremos en la siguiente entrega.