

1. Archivos.

1.1 Introducción.

Disponemos de dos tipos de archivos, Secuenciales y Random.

Las condiciones son las ya conocidas, pero cambia la sintaxis y las instrucciones para su uso.

Cambian los nombres, pero sigue todo igual, afortunadamente.

Aparece la posibilidad de bloqueo de los archivos para otros procesos, en lectura, o en grabación o ambas, en el momento de la apertura.

1.2 Secuencial.

Se sigue abriendo para entrada, salida y para adición.

Las instrucciones son las mismas, cambian las instrucciones de apertura y cierre.

El ejemplo que sigue graba y lee un archivo con los meses del año.

En la declaración del Canal, se usa la función FreeFile para obtener un número de canal disponible.

```
Module Ejemplo
Sub main()
  Dim Canal As Int16 = FreeFile()
  Dim Nombre As String = "Prueba.Txt"
  Dim Valor As String
  Dim Meses() As String = {"Enero", "Febrero", "Marzo", _
                          "Abril", "Mayo", "Junio", _
                          "Julio", "Agosto", "Septiembre", _
                          "Octubre", "Noviembre", "Diciembre"}

  Console.WriteLine("Probamos grabar un archivo secuencial ")
  FileOpen(Canal, Nombre, OpenMode.Output)
  For Each Valor In Meses
    WriteLine(Canal, Valor)    \ con marca fin de registro
                              \ sin marca Write a solas
    Console.WriteLine("Grabado {0} ", Valor)
  Next
  ' Cierre del Archivo
  FileClose(Canal)

  Console.WriteLine()
  Console.WriteLine("Probamos a leer un archivo secuencial ")
  FileOpen(Canal, Nombre, OpenMode.Input)
  Do While Not EOF(Canal)
    Input(Canal, Valor)
    Console.WriteLine("Leído {0} ", Valor)
  Loop
  ' Cierre del Archivo
  FileClose(Canal)
End Sub
End Module
```

Vemos ahora el ejemplo para adición de registros, solo es necesario cambiar el parámetro de la instrucción FileOpen, OpenMode.Output, por OpenMode.Append.

```
Module Ejemplo
Sub main()
  Dim Canal As Int16 = FreeFile()
  Dim Nombre As String = "Prueba.Txt"
  Dim Valor As String
  Dim Meses() As String = {"Enero", "Febrero", "Marzo", _
                          "Abril", "Mayo", "Junio", _
                          "Julio", "Agosto", "Septiembre", _
                          "Octubre", "Noviembre", "Diciembre"}

```

```

Console.WriteLine("Probamos añadir registros al archivo secuencial ")
FileOpen(Canal, Nombre, OpenMode.Append)
For Each Valor In Meses
    WriteLine(Canal, Valor)
    Console.WriteLine("Grabado {0} ", Valor)
Next
' Cierre del Archivo
FileClose(Canal)

Console.WriteLine()
Console.WriteLine("Probamos a leer un archivo secuencial ")
FileOpen(Canal, Nombre, OpenMode.Input)
Do While Not EOF(Canal)
    Input(Canal, Valor)
    Console.WriteLine("Leído {0} ", Valor)
Loop
' Cierre del Archivo
FileClose(Canal)
End Sub
End Module

```

La instrucción FileOpen, como se puede ver en el ejemplo recibe tres parámetros.
 El canal por el que se trabaja con el archivo.
 El nombre del archivo y el modo de apertura del mismo.

La lectura es con la instrucción Input(Canal, Variable, Variable,)
 La grabación es con la instrucción Write(Canal, Variable, Variable,)

Las variables han de estar en consonancia al tipo de dato grabado.
 El formato de archivo obtenido es un fichero Ascii delimitado, con comillas comas.

1.3 Random.

Las características de los archivos a la hora de grabarlos no cambian, los pasos a seguir son los mismos.

Conviene tener presente que con la instrucción FileGet, si el entorno está con el modo Option Strict On, dará errores y hay que desactivarlo, sin embargo FilePut, funciona correctamente, esperemos que lo soluciones pronto.

Como en el caso de los secuenciales cambian el nombre de las instrucciones, aunque al final es igual.
 El primer paso es definir la estructura del registro a utilizar.

Como podemos ver en el ejemplo se define un tipo de usuario con los campos definidos como públicos y cuando son de tipo cadena, usando delante el argumento <VbFixedString(longitud de campo)>.

En el ejemplo lo declaramos de doce, para así poder utilizar el relleno por la derecha de blancos.

El archivo generado, no tiene delimitadores de campos ni de registros, Random estándar.

```

Structure RegRandom
    <VbFixedString(12)> Public NombreMes As String
End Structure

```

Este es otro ejemplo de estructura de archivo random.

```

Public Structure RegistroRandom
    <VbFixedString(15)> Public Nom As String
    <VbFixedString(15)> Public Dire As String
    <VbFixedString(15)> Public Pobl As String
    <VbFixedString(2)> Public Prov As String
    <VbFixedString(9)> Public Tele As String
End Structure

```

El siguiente paso es declarar luego la variable registro que vamos a utilizar en el acceso al archivo.

```
Dim Registro As RegRandom
```

Y una variable en este caso Int16 para marcar la posición en el archivo.

```
Dim Posicion As Int16
```

A la apertura del archivo se le han añadido varios parámetros que no se han usado, excepto el de longitud de registro, y además el modo de apertura se indica que es Random.

```
FileOpen(Canal, Nombre, OpenMode.Random, , , Len(Registro))
```

El siguiente paso es asignar los datos a la variable registro, rellena con blancos por la derecha, lo hemos hecho usando el objeto Strings con su método Lset.

```
Registro.NombreMes = [Strings].LSet(Valor, 12)
```

A continuación se incrementa el contador, sistema abreviado

```
Posicion += 1
```

Y utilizamos la instrucción FilePut para grabar, el contenido es el mismo que antes con el Put.

```
FilePut(Canal, Registro, Posicion)
```

Luego sin cerrar el archivo, inicializamos posición, o usamos el For Next, y leemos el archivo entero, y funciona.

La lectura es con FileGet, que funciona igual que el Get.

```
FileGet(Canal, Registro, Posicion)
```

Y ahora todo el ejemplo completo.

```
Module Ejemplo
  Structure RegRandom
    <VBFixedString(12)> Public NombreMes As String
  End Structure

  Sub main()
    Dim Registro As RegRandom
    Dim Canal As Int16 = FreeFile()
    Dim Nombre As String = "PruebaRandom.Txt"
    Dim Posicion As Int16
    Dim Valor As String

    Dim Meses() As String = {"Enero", "Febrero", "Marzo", _
                             "Abril", "Mayo", "Junio", _
                             "Julio", "Agosto", "Septiembre", _
                             "Octubre", "Noviembre", "Diciembre"}

    Console.WriteLine("Probamos añadir registros al archivo ")
    FileOpen(Canal, Nombre, OpenMode.Random, , , Len(Registro))
    For Each Valor In Meses
      Posicion += 1
      Registro.NombreMes = [Strings].LSet(Valor, 12)
      FilePut(Canal, Registro, Posicion)
      Console.WriteLine("Grabado {0} ", Valor)
    Next

    Console.WriteLine()
    Console.WriteLine("Probamos a leer el archivo ")
  End Sub
End Module
```

```

For Posicion = 1 To 12
    FileGet(Canal, Registro, Posicion)
    Console.WriteLine("Leído {0} ", Registro.NombreMes)
Next

' Cierre del Archivo
FileClose(Canal)

End Sub
End Module

```

1.4 Otras actuaciones con archivos.

Además de leer los archivos de datos, a veces hay que borrar o cambiar de nombre alguno de los archivos, los métodos imprescindibles son los que figuran en el ejemplo.

```

Module Ejemplo
    Structure RegRandom
        <VBFixedString(12)> Public NombreMes As String
    End Structure

    Sub main()
        Dim Registro As RegRandom
        Dim Canal As Int16 = FreeFile()
        Dim Nombre As String = "PruebaRandom.Txt"
        Dim Posicion As Int16

        FileOpen(Canal, Nombre, OpenMode.Random, , , Len(Registro))

        Console.WriteLine(FileSystem.CurDir()) ' Directorio Actual

        ' Devuelve la fecha del archivo
        Console.WriteLine(FileSystem.FileDateTime("C:\Autoexec.bat"))

        ' Devuelve la longitud en bytes del archivo, 144 = 12 * 12
        Console.WriteLine(FileSystem.FileLen("PruebaRandom.txt"))
        ' Devuelve la longitud en bytes del archivo, 144 = 12 * 12
        Console.WriteLine(FileSystem.LOF(Canal)) ' 144, 12 * 12

        ' Posición actual del puntero
        Console.WriteLine(FileSystem.Loc(Canal))

        ' Cambia de nombre un archivo
        FileSystem.Rename(Viejo, Nuevo)

        ' Borrado de directorio
        FileSystem.Rmdir(Direccion)

        ' Cierre del archivo
        FileClose(Canal)
        ' Borra el archivo
        FileSystem.Kill("PruebaRandom.Txt")

    End Sub
End Module

```

Además hay más métodos disponibles.

1.5 Listados.

Los listados han sufrido cambios considerables.

No hacemos referencia al Crystal Report, pues en definitivas cuentas es un asistente, no muy complicado de ver, y que los alumnos en cualquier momento se pueden mirar.

Hablamos desde el listado con los recursos disponibles en Visual Studio, con los objetos PrintDocument, PrintPreviewDialog, ...

El principal es que el hecho de realizar un listado para que éste se obtenga por impresora o pantalla, es indistinto, ha de realizarse en el evento PrintPage del objeto PrintDocument.

Este objeto, es el que digamos realiza la labor de impresión, facilitando el resultado del mismo al objeto impresora o al de vista previa, que son los que se encargan de su visualización en el destino correspondiente.

El principal apartado a tener en cuenta es el sistema de cambio de página, el cual, salvo error por nuestra parte, es ..., sin comentarios.

Lo más parecido a lo visto, es la salida por excepción del lenguaje RPG, aunque claro, nada que ver con la misma, esa funcionaba bien, claro, tampoco con el lenguaje, ni con su diseñador.

El sistema seguido para el cambio de página es el realizar una salida del procedimiento de evento PrintPage, indicando previamente en el objeto "e", con su propiedad HasMorePages a True, que quedan más páginas que imprimir.

```
If Cambio then
    e.HasMorePages = true
    Exit Sub
End If
```

Esto es estupendo, a primera vista solo, pues se supone, que se lanza el cambio de página y retorna al punto de ruptura, pero no es así, vuelve al inicio del procedimiento, no saben más los chicos de Bill, y eso te obliga a tomar precauciones.

Por ejemplo el archivo no se puede abrir en el evento PrintPage, sino dará errores, o se perderá el posicionamiento del puntero, etc..

El siguiente paso es pensar que lo abres en la llamada al procedimiento, en el evento Click del objeto Button, pero tampoco, porque entonces, como son eventos, no se ejecutan como procedimientos, sino que finaliza uno y empieza el otro, por lo que si abres y cierras en el Button, cuando llegas al PrintPage, el archivo está cerrado.

Por lo tanto, de momento:

Con archivos secuenciales, hay que abrir el archivo al cargar el Formulario, en el evento BeginPrint, o establecer código de control para saber si el archivo está abierto o no.

Con archivos random, hay menos pegas, pues abrir el archivo a nivel de Form, o de aplicación, según necesidades o costumbres, y solo preocuparse que la variable de posicionamiento en el archivo esté definida a nivel de formulario, fuera del procedimiento de evento, PrintPage, que se encarga del listado, y tema resuelto.

Vemos un ejemplo sobre un archivo secuencial.

Primero las estructuras y los procedimientos que se usan en el ejemplo.

```
Public Structure Archivo
    Public Exped As String
    Public Nomb As String
    Public Apel As String
    Public Ape2 As String
    Public Domic As String
End Structure

Public Structure CabecDetalle
    Public Texto As String
    Public Cx As Long
End Structure

Private Sub ConfigCabecera(ByRef Textocabecera() As CabecDetalle)
```

```

Dim Fuente As Font
Dim Lapiz As New System.Drawing.SolidBrush(System.Drawing.Color.Black)
Dim Ancho As New SizeF
Dim Formato As New System.Drawing.StringFormat
Dim Cx As Long
Dim Grafico As Graphics = Me.CreateGraphics
Dim X As Integer
Fuente = Est_Lin_Det
' Campos del listado
Textocabecera(0).Texto = "Código   "
Textocabecera(1).Texto = "Nombre       "
Textocabecera(2).Texto = "Apellido 1   "
Textocabecera(3).Texto = "Apellido 2   "
Textocabecera(4).Texto = "Domicilio    "
Textocabecera(5).Texto = ""

' Se crea para poder calcular el margen.
Me.VistaPreviaDialogo = New PrintPreviewDialog
' Formato del texto
Formato.FormatFlags = StringFormatFlags.MeasureTrailingSpaces
' Margen lateral
Cx = CLng(VistaPreviaDialogo.Size.Width * 0.1)
' Fuente a utilizar
Fuente = Est_Lin_Det ' New Font("Arial", 12, FontStyle.Italic)
' Bucle de cálculo de coordenadas Cx
While X < UBound(Textocabecera)
    Textocabecera(X).Cx = Cx
    ' Ancho del texto
    Ancho = Grafico.MeasureString(StrDup(Len(Textocabecera(X).Texto),
"n"), Fuente)
    Cx = CLng(Cx + AnchoString.Width)
    X = X + 1
End While
End Sub

```

```

Private Sub LineaDetalle( _
    ByVal e As System.Drawing.Printing.PrintPageEventArgs, _
    ByRef Cy As Long, _
    ByVal Registro As Archivo)

    Dim Fuente As New System.Drawing.Font("Times New Roman", 12,
FontStyle.Bold)
    Dim Pincel As New System.Drawing.SolidBrush(System.Drawing.Color.Black)

    With Registro
        e.Graphics.DrawString(.Exped,Fuente, Pincel, Textocabecera(0).Cx, Cy)
        e.Graphics.DrawString(.Nomb, Fuente, Pincel, Textocabecera(1).Cx, Cy)
        e.Graphics.DrawString(.Apel1, Fuente, Pincel, Textocabecera(2).Cx, Cy)
        e.Graphics.DrawString(.Ape2, Fuente, Pincel, Textocabecera(3).Cx, Cy)
        e.Graphics.DrawString(.Domic,Fuente, Pincel, Textocabecera(4).Cx, Cy)
    End With
    Cy = Cy + Fuente.Height
End Sub

```

```

Private Sub Cabeceras(_
    ByRef Cy As Long, _
    ByVal e As System.Drawing.Printing.PrintPageEventArgs)
    LineaIden(Cy, ContPag, e)           ' Línea de identificación
    Titulo(Cy, e, "Listado archivo ")  ' Título del listado
    Cabecera(Cy, TextoCabecera, e)    ' Cabecera de detalle
End Sub

```

El procedimiento que sigue es el evento PrintPage de un objeto del tipo PrintDocument, que en el ejemplo se denomina Hoja,

Este procedimiento se llama desde el evento clic de un objeto Button.

```

Private Sub Hoja_PrintPage(_
    ByVal sender As System.Object, _
    ByVal e As System.Drawing.Printing.PrintPageEventArgs) _
    Handles Hoja.PrintPage

    Dim Cy As Long           ' Coordenada vertical
    Dim Cabec As Boolean = True ' Obtención de cabeceras
    Dim Pie As Boolean = False ' Activar cambio de página
    Dim Registro As Archivo

    While Not EOF(Canal)
        If Pie Then
            PiePagina(Cy, e)
            e.HasMorePages = True
            Exit Sub
        End If
        If Cabec Then
            Cabeceras(Cy, e)
            Cabec = False
        End If
        ' lectura de un registro
        With Registro
            Input(Canal, .Exped) ' código
            Input(Canal, .Nomb)
            Input(Canal, .Ape1)
            Input(Canal, .Ape2)
            Input(Canal, .Domic)
        End With
        ' Línea de detalle
        LineaDetalle(e, Cy, Registro)
        ' Control de fin de página
        Pie = Cy > e.MarginBounds.Height
    End While
    FinImpresion(Cy, e)
    e.HasMorePages = False
    FileClose(Canal)
End Sub

```

La llamada desde el objeto Button, puede ser algo así.

Primero se asigna valor a unas propiedades de un objeto PrintDialog, llamado DialogoImpresora.

Después se visualiza el objeto y se espera la respuesta del usuario, y si esta es afirmativa se lanza el listado.

```

DialogoImpresora.AllowSelection = False
DialogoImpresora.AllowSomePages = False
DialogoImpresora.PrintToFile = False
DialogoImpresora.PrinterSettings.Copies = 1
DialogoImpresora.Document = Hoja
If DialogoImpresora.ShowDialog() = Windows.Forms.DialogResult.OK Then
    Hoja.Print()
End IF

```

Y en el evento Load del formulario llamaremos al procedimiento de configuración, y o bien aquí o en el evento BeginPrint abriremos el archivo.

```

ConfigCabecera(TextoCabecera)
Canal = FreeFile()
FileOpen(Canal, "Archivo.Sec", OpenMode.Input)

```

El listado con un archivo random, es más fácil, pues no hay la complejidad de perder los punteros en las entradas y salidas del procedimiento.

Todo lo anterior es valido, excepto el procedimiento de evento del objeto PrintDocument, Hoja, y también que la variable que controla el posicionamiento en el archivo está en el formulario.

También se podría haber declarado estática, pero hemos optado por ésta visión.

Por lo tanto la definición de variables a nivel de formulario queda:

```

Dim Posicion As Long           ' Posición en archivo random
Dim ContPag As Integer         ' Contador de páginas
Dim TextoCabecera(4) As CabecDetalle ' Vector de campos
Private WithEvents VistaPreviaDialogo As PrintPreviewDialog ' Objeto
Dim Registro as RegistroRandom

```

La variable registro es del siguiente tipo de estructura, o de usuario.

```

Public Structure RegistroRandom
    <VBFixedString(15)> Public Nom As String
    <VBFixedString(15)> Public Dire As String
    <VBFixedString(15)> Public Pobl As String
    <VBFixedString(2)> Public Prov As String
    <VBFixedString(9)> Public Tele As String
End Structure

```

El procedimiento de impresión , el evento PrintPage :

```

Private Sub Hoja_PrintPage( _
    ByVal sender As System.Object, _
    ByVal e As System.Drawing.Printing.PrintPageEventArgs) _
    Handles Hoja.PrintPage

    Dim Cy As Long           ' Coordenada vertical
    Dim Cabec As Boolean = True
    Dim Pie As Boolean = False

    While Posicion <= CLng(Campo02.Text)
        FileGet(Canal, Registro, Posicion)
        Select Case Registro.Nom <> Space(15)
            Case True ' registro ocupado
                If Pie Then
                    PiePagina(Cy, e)
                    e.HasMorePages = True
                End If
            Case False
                Exit Sub
        End Select
        Posicion = Posicion + 1
    End While
End Sub

```

```

        End If
        If Cabec Then
            Cabeceras(Cy, e)
            Cabec = False
        End If
        ' Línea de detalle
        LineaDet(e, Cy)
        ' Control de fin de página
        Pie = Cy > e.MarginBounds.Height
    End Select
    ' Incremento de posición
    Posicion = Posicion + 1
End While
FinImpresion(Cy, e)
e.HasMorePages = False
End Sub

```

Y la llamada a éste procedimiento se hace desde un objeto Button, en su evento click.

El ejemplo que sigue está hecho con un PrintPreviewDialog, en lugar de con la impresora como el anterior.

En éste ejemplo hay que matizar, que se crea el ejemplo PrintPreviewDialog cada vez que se lanza el listado, pues de esa forma al descargarse cuando se cierra la vista de su contenido, si se vuelve a lanzar el listado volvemos a tener una referencia valida del mismo, sino dará error.

Por eso no se crea en tiempo de diseño, adjunto al formulario.

Por lo tanto la llamada desde el evento Click del Button queda como sigue:

```

' Inicializar la variable posición
Posicion = CLng(Campo01.Text)
' Crear el objeto cada vez que se hace el listado.
Me.VistaPreviaDialogo = New PrintPreviewDialog
' hacer que esté integrado en el formulario menú.
VistaPreviaDialogo.MdiParent = Principal
' asignar el objeto que contiene el listado
VistaPreviaDialogo.Document = Hoja
' visualizar el resultado
VistaPreviaDialogo.Show()

```

Y en el evento Load del formulario, se abre el archivo random, aunque se puede hacer en otro sitio.

Para la ejecución de éste ejemplo es necesario que la opción de configuración Option Strict esté en Off.

Desconocemos porque da error la instrucción FileGet, ya que su equivalente FilePut, funciona correctamente bajo esa configuración, y la consulta que se ha pasado a Microsoft, aún esta en trámite, y no ha sido respondida.

1.6 Cargar un archivo en un DataGrid.

Los datos de un archivo también se pueden visualizar en un objeto DataGrid, no tiene porque ser de uso exclusivo para las bases de datos.

Este objeto es de unas prestaciones bastante amplias, y contempla un sin fin de posibilidades, adjuntamos un procedimiento de configuración, pero evidentemente, es ampliable en características y solo es un ejemplo, hemos intentado que amplio.

```

Private Sub ConfigDataGrid()

    ' Formato del DataGrid

    Dim X As Integer
    With DataGrid
        ' Definir la fuente
        .Font = Est_Lin_Det
        ' Tamaño
    End With

```

```

.Height = Me.Height * 0.5
.Width = Me.Width * 0.9
' Ubicación
.Left = Me.Width * 0.05
.Top = Me.Height * 0.5

' Cómo se activa la edición en la celda
.EditMode = DataGridViewEditMode.EditOnEnter
' Modo de selección
.SelectionMode = DataGridViewSelectionMode.FullRowSelect
' Multiselección
.MultiSelect = False
' Cómo se determina el tamaño de las filas
.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None
' Ajustes del encabezado de las columnas,
.ColumnHeaderHeightSizeMode = _
    DataGridViewColumnHeadersHeightSizeMode.DisableResizing
'
' de las filas
.RowHeaderWidthSizeMode = _
    DataGridViewRowHeadersWidthSizeMode.DisableResizing

' Añadir filas, no disponible
.AllowUserToAddRows = False
' Borrar filas, no disponible
.AllowUserToDeleteRows = False
' Activar posibilidad de cambiar columnas de sitio
.AllowUserToOrderColumns = True
' Activar posibilidad de cambiar tamaño de las columnas
.AllowUserToResizeColumns = True
' Cambiar tamaño de las filas
.AllowUserToResizeRows = False
' Solo lectura
.ReadOnly = True
' Color de fondo
.BackgroundColor = Color.White
'
' del texto
.ForeColor = Color.Black
'
' de las líneas
.GridColor = Color.Green
' Estilo del borde
.BorderStyle = BorderStyle.Fixed3D
'
' de bordes de cabecera
.ColumnHeadersBorderStyle = DataGridViewHeaderBorderStyle.Raised
'
' de celdas de cabecera
.RowHeadersBorderStyle = DataGridViewHeaderBorderStyle.Raised
'
' del borde las celdas
.CellBorderStyle = DataGridViewCellBorderStyle.Single
' Mostrar la columna que contiene los encabezados de la fila
.RowHeadersVisible = True

' Estilos

' Asignar color de fondo para la selección para todas las celdas
.DefaultCellStyle.SelectionBackColor = Color.White
' Asignar color de primer plano para la selección para todas las
celdas
.DefaultCellStyle.SelectionForeColor = Color.Black

' Asignar Estilo de selección de fila de cabecera por defecto
' de forma que el valor no se superponga a los valores
' de estilo de las celdas.

```

```

.RowHeadersDefaultCellStyle.SelectionBackColor = Color.Empty

' Asignar el color de fondo para todas las filas
' y para forma alternativa
.RowsDefaultCellStyle.BackColor = Color.AntiqueWhite
.AlternatingRowsDefaultCellStyle.BackColor = Color.Aqua

' Asignar la fila y columna de estilo de cabecera
.ColumnHeadersDefaultCellStyle.ForeColor = Color.AntiqueWhite
.ColumnHeadersDefaultCellStyle.BackColor = Color.BlueViolet
.RowHeadersDefaultCellStyle.BackColor = Color.Black

' Crear las columnas del datagrid
.ColumnCount = 6
.Columns(0).Name = "Exped."
.Columns(1).Name = "Nomb"
.Columns.Item(2).Name = "Dire"
.Columns.Item(2).HeaderText = "Dirección"
.Columns.Item(3).Name = "Pobl"
.Columns.Item(3).HeaderText = "Población"
.Columns.Item(4).Name = "Prov"
.Columns.Item(4).HeaderText = "Provincia"
.Columns.Item(5).Name = "Telef"
.Columns.Item(5).HeaderText = "Teléfono"

' Acciones de ajuste
' Tamaño de las columnas
For X = 0 To DataGrid.ColumnCount - 1
.AutoResizeColumn(X)
Next

' Ajuste del alto de los encabezados
.AutoResizeColumnHeadersHeight()
End With
End Sub

```

El siguiente paso es cargar los datos en el mismo.

```

Private Sub CargaDataGrid()
With DataGrid
.Rows.Add(Posicion, Reg.Nom, Reg.Dire, Reg.Pobl, Reg.Prov, Reg.Tele)
End With
End Sub

```

El siguiente la llamada desde el bucle de listado, en éste caso, un archivo random.

```

Private Sub BucleDatagrid()
While Posicion <= CLng(Campo02.Text)
FileGet(CanalAgenda, Reg, Posicion)
Select Case Reg.Nom <> Space(15)
Case True ' registro ocupado
' Línea de detalle
CargaDataGrid()
End Select
' Incremento de posición
Posicion = Posicion + 1
End While
End Sub

```

Solo queda la llamada desde el evento Click de un objeto Button.

```
' Inicializar la variable posición  
Posicion = CLng(Campo01.Text)  
' Ejecutar el bucle  
BucleDataGrid
```