1. Impresión.

1.1 Introducción.

Vamos a ver la impresión desde el entorno de programación, no vamos a ver Crystal Report.

La conclusión después de lo visto con ésta versión de VB Studio Net 2005 es que hemos mejorado los recursos con respecto a la versión seis, aunque lo que se ha hecho es legalizar lo que a nivel de calle era habitual, y como consecuencia de ello el objeto PictureBox, que antes se usaba para lo que ahora hace el PrintDocument, ha perdido casi todas sus capacidades.

En esta versión nos aparece una categoría de objetos en la caja de herramientas denominados Impresión.

Estos objetos son:

PageSetupDialog
PrintDialog
PrintDocument
PrintPreviewControl
PrintPreviewDialog

El que dirige la impresión PrintDocument y el resto los que recogen los resultados, por lo tanto:

Como director de orquesta	PrintDocument
Como receptor para la impresora	PrintDialog
Como receptor para una vista previa	PrintPreviewDialog
Configurar parámetros desde usuario	PageSetupDialog
Como receptor para gestionar nosotros la vista previa	PrintPreviewControl.

Una vez hecha una primera aproximación al tema hay que avanzar unas cuantas advertencias para entender el ejemplo que tenemos.

El funcionamiento de una impresión está controlado por el objeto PrintDocument en su evento PrintPage.

Es decir que las tareas de impresión se han de escribir en éste evento para que se ejecuten, no es como antes que se escribía un procedimiento para realizar un listado y desde él se llamaba a un procedimiento en el que se realizaba la tarea de imprimir, método Print, sea en el objeto Printer o en un PictureBox, y se gestionaban los cambios de página.

Ahora hay que gestionar los cambios de página ejecutando las líneas que siguen

```
e.HasMorePages = True
Exit Sub
```

Esto provoca una salida del procedimiento de evento y un retorno al mismo, pero el retorno es al principio, no al punto en el que se ejecuta la salida, por lo que hay que tener presente, que es lo que se hace en el inicio del procedimiento, pues si no puede ser que nunca acabemos el listado, porque siempre se inicie desde el principio, o bien que se generen errores de apertura de archivos, o posicionamientos inesperados, o parecidos.

Por lo tanto hay que realizar todas las tareas previas al inicio fuera de ese procedimiento para evitar su duplicado y consecuente error.

Por otro lado tampoco sirve realizar algunas tareas en el procedimiento desde el que se llama al objeto PrintDocument, pues como es un evento, no se ejecuta en el orden de llamada, sino después de que se a ejecutado el procedimiento, por lo que si se

```
Apertura de archivos
Llamada a PrintDocument
Cerrado de archivos
```

cuando se ejecuta el PrintDocument, los archivos están cerrados.

Cuando estamos usando el objeto PrintPreviewDialog, no se puede colocar el objeto en tiempo de diseño en el formulario si deseamos que se pueda ejecutar el listado más de una vez, ya que la segunda vez da error por que el objeto no está disponible.

La solución es crear una instancia de ese objeto en tiempo de ejecución en el programa, de esa forma queda resuelto el problema.

En fin esta es la entrada, pero a pesar de todo, una vez controlado estas incidencias, la impresión funciona.

1.2 Paso a paso.

Desde el interior hacia el exterior.

Vamos a ver la impresión de dentro a fuera, empezando por la impresión de una línea de detalle

Los pasos que hemos contemplado son:

Inicio de listado
Cabeceras
Detalle
Fin de página
Fin de listado

El procedimiento de impresión que vamos a ver está basado en el uso de un vector en el que se almacenan los datos de los campos que se van a imprimir.

Estos datos son el título y su coordenada X, en su mínima expresión, esto se puede enriquecer, pero para el ejemplo es suficiente.

Por lo que el primer paso es ver como cargar ese vector y calcular la coordenada X de cada campo.

Este procedimiento también es susceptible de mejoras, pero se trata de ver un ejemplo sencillo y no complicar el tema.

1.2.1 Definición.

Estructura del vector para su creación.

En esta estructura se pueden añadir todos aquellos parámetros que permiten potenciar el diseño del listado a nivel de campos, color, coordenada Y para casos especiales, subcabeceras, etc.

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

Crear el vector, a nivel de formulario, conviene que tenga un campo de más, a nulos, porque ese campo puede ser, es, el final del campo anterior, es decir el último, pero nos da la coordenada X exacta para un posible enmarcado, o final de un subrayado, etc.…

```
Dim TextoCabecera(6) As CabecDetalle
```

Estilos que se van a utilizar en el listado.

La definición se puede realizar a nivel de módulo.

Estos datos se pueden capturar de un archivo de personalización.

```
' Estilos de impresión
 Public Est_Lin_Iden As Font = New Font("Arial", 6)
 Public Est_Titulo As Font = New Font("Arial", 14, FontStyle.Italic)
 Public Est_Cabecera As Font = New Font("Arial", 12)
 Public Est_Lin_Det As Font = New Font("Arial", 10, FontStyle.Regular)
 Public Est_Pie As Font = New Font("Arial", 12)
 Public Est_Fin As Font = New Font("Arial", 12)
```

Procedimiento que calcula las coordenadas para su uso posterior.

El cálculo se basa en la fuente que se va a utilizar después en el momento de imprimir la línea de detalle.

```
Private Sub ConfigCabecera(ByRef Textocabecera() As CabecDetalle)
  Dim Fuente As Font
  Dim Lapiz As New System.Drawing.SolidBrush(System.Drawing.Color.Black)
  Dim AnchoString 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 = ""

  ' Formato del texto
  Formato.FormatFlags = StringFormatFlags.MeasureTrailingSpaces
  ' Margen lateral
  Cx = Clng(Hoja.DefaultPageSettings.Margins.Left)
  ' Fuente a utilizar
  Fuente = Est_Lin_Det
  ' Bucle de cálculo
  While X < UBound(Textocabecera)
    Textocabecera(X).Cx = Cx
    ' Ancho del texto
    AnchoString                                                          =
Grafico.MeasureString(StrDup(Len(Textocabecera(X).Texto), "n"), Fuente)
    Cx = CLng(Cx + AnchoString.Width)
    X = X + 1
  End While
End Sub
```

El ancho del campo se calcula a partir de una letra de referencia generando una cadena con ella, en el ejemplo se ha usado la “n”, que ni es muy ancha ni muy estrecha. Hay que tener presente que se usan fuentes del tipo TrueType, no de ancho fijo.

Este procedimiento debe llamarse desde el evento Load, de esa forma solo se ejecuta una vez.

1.2.2 Línea de detalle.

En este procedimiento se recibe el registro de datos del archivo que se va a imprimir, y se utiliza para colocar cada campo en su sitio las coordenadas X Y de rigor,

La coordenada X es la que se ha calculado antes en la fase de configuración.

La coordenada Y se va calculando para cada fase del listado.

En el caso de éste procedimiento, se basa en la suma de la altura de la fuente utilizada a una variable llamada Cy, que se pasa ByRef.

Este incremento puede alterarse y de esa forma cambiar el espacio interlinea.

El valor de esa variable es el que se usará como referencia para el control de cambio de página al superar un determinado valor del alto del objeto.

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

    Dim Fuente As Font
    Dim Pincel As New System.Drawing.SolidBrush(System.Drawing.Color.Black)

    Fuente = Est_Lin_Det
    With Registro
      e.Graphics.DrawString(.Expe, Fuente, Pincel, TextoCabecera(0).Cx, Cy)
      e.Graphics.DrawString(.Nomb, Fuente, Pincel, TextoCabecera(1).Cx, Cy)
      e.Graphics.DrawString(.Ape1, Fuente, Pincel, TextoCabecera(2).Cx, Cy)
      e.Graphics.DrawString(.Ape2, Fuente, Pincel, TextoCabecera(3).Cx, Cy)
      e.Graphics.DrawString(.Domi, Fuente, Pincel, TextoCabecera(4).Cx, Cy)
    End With
    ` Incremento de la coordenada Y
    Cy = Cy + Fuente.Height
  End Sub
```

1.2.3 Cabeceras.

La cabecera puede crearse todo lo detallada que se desee, en el ejemplo se ha utilizado

Línea de identificación
Título
Cabecera de detalle.

```
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 secuencial")    ' Título del listado
  Cabecera(Cy, TextoCabecera, e)  ' Cabecera de detalle
End Sub
```

Los procedimientos llamados desde este procedimiento son los siguientes

```
Public Sub LineaIden(ByRef Cy As Long, _
                     ByRef ContPag As Integer, _
                     ByVal e As System.Drawing.Printing.PrintPageEventArgs)
  Dim Fuente As Font
  Dim Pincel As New System.Drawing.SolidBrush(System.Drawing.Color.Black)
  Dim Alto As Long = e.PageSettings.PaperSize.Height
  Dim Ancho As Long = e.PageSettings.PaperSize.Width

  Dim Cx As Long          ' Coordenada horizontal
  Dim Texto As String

  Fuente = Est_Lin_Iden
  ContPag = ContPag + 1
  Cy = CLng(Alto * 0.02)      ' Avance de línea con respecto al borde
superior
  Cx = CLng(Ancho * 0.05)
  Texto = "Archivos secuenciales             Pag: " & ContPag & "  " & Now
  e.Graphics.DrawString(Texto, fuente, Pincel, Cx, Cy)
  Cy = Cy + fuente.Height  ' Avance de línea
End Sub
```

```
 Public Sub Titulo(ByRef Cy As Long, _
                   ByVal e As System.Drawing.Printing.PrintPageEventArgs, _
                   ByVal Texto As String)

   Dim Fuente As Font
   Dim Pincel As New System.Drawing.SolidBrush(System.Drawing.Color.Black)
   Dim Ancho As Long = e.PageSettings.PaperSize.Width
   Dim Cx As Long

   Fuente = Est_Titulo
   Cx = CLng(Ancho * 0.05)
   Cy = Cy + fuente.Height
   e.Graphics.DrawString(Texto, fuente, Pincel, Cx, Cy)
   Cy = Cy + fuente.Height
 End Sub
```

```
Public Sub Cabecera(ByRef Cy As Long, _
                    ByVal TextoCab() As CabecDetalle, _
                    ByVal e As System.Drawing.Printing.PrintPageEventArgs)

  Dim Fuente As Font
  Dim Pincel As New System.Drawing.SolidBrush(System.Drawing.Color.Black)
  Dim Lapiz As New Pen(Color.Black, 3)

  Dim Ancho As Long = e.PageSettings.PaperSize.Width

  Dim X As Integer

  Fuente = Est_Cabecera
  Cy = Cy + Fuente.Height     ' Separación del título
  While X < UBound(TextoCab)
    e.Graphics.DrawString(TextoCab(X).Texto,         Fuente,         Pincel,
TextoCab(X).Cx, Cy)
    X = X + 1
  End While
  Cy = Cy + Fuente.Height     ' Avance de línea
  Cy = Cy + 10                ' Línea de subrayado
  e.Graphics.DrawLine(Lapiz,  CInt(Ancho  *  0.05),  Cy,  CInt(Ancho  *  0.95),
Cy)
End Sub
```

En estos procedimientos se definen las variables Alto y Ancho con respecto al objeto “e”, propiedades de tamaño de la hoja definida para la impresión, para de esa forma usarlas como referencia proporcional en la ubicación de los datos.

1.2.4 Fin de página.

El procedimiento es igual a otros, no tiene nada significativo.

```
Public Sub PiePagina(ByVal Cy As Long, _
                     ByVal e As System.Drawing.Printing.PrintPageEventArgs)

  Dim Alto As Long = e.PageSettings.PaperSize.Height
  Dim Ancho As Long = e.PageSettings.PaperSize.Width
  Dim Fuente As Font
  Dim Pincel As New System.Drawing.SolidBrush(System.Drawing.Color.Black)
  Dim Cx As Long
  Dim Texto As String

  Fuente = Est_Pie
  Cy = CLng(Alto * 0.95)     ' Situación con respecto al alto de la página
```

```
  Cx = CLng(Ancho * 0.05)      ' Situación con respecto al ancho de la
página
  Texto = "Final de página"
  e.Graphics.DrawString(Texto, Fuente, Pincel, Cx, Cy)
  Cy = Cy + Fuente.Height
End Sub
```

1.2.5 Inicio de listado.

Lo único significativo es que la coordenada Cy no se tiene en cuenta, por eso se declara internamente, ya que se supone que se usa una página completa para este paso.

```
Public Sub InicioImpresion _
                   (ByVal e As System.Drawing.Printing.PrintPageEventArgs)
  Dim Alto As Long = e.PageSettings.PaperSize.Height
  Dim Ancho As Long = e.PageSettings.PaperSize.Width
  Dim Fuente As New Font("Arial", 18, FontStyle.Italic)
  Dim Pincel As New System.Drawing.SolidBrush(System.Drawing.Color.Black)
  Dim Cx As Long
  Dim Cy As Long
  Dim Texto As String

  Cy = CLng(Alto * 0.5)      ' Situación con respecto al alto de la página
  Cx = CLng(Ancho * 0.05)    ' Situación con respecto al ancho de la página
  Texto = "Inicio de listado"
  e.Graphics.DrawString(Texto, Fuente, Pincel, Cx, Cy)
End Sub
```

El inicio de listado, no se puede llamar desde el evento BeginPrint, pues no disponemos del mismo tipo de objeto, y además se activa antes de imprimir la primera página.

Podemos abrir archivos en este punto.

1.2.6 Fin de listado.

La filosofía es similar al de inicio.

```
Public Sub FinImpresion(ByRef Cy As Long, _
                    ByVal e As System.Drawing.Printing.PrintPageEventArgs)
    Dim Alto As Long = e.PageSettings.PaperSize.Height
    Dim Ancho As Long = e.PageSettings.PaperSize.Width
    Dim Fuente As Font
    Dim Pincel As New System.Drawing.SolidBrush(System.Drawing.Color.Black)
    Dim Cx As Long
    Dim Texto As String

    Fuente = Est_Fin
    Cy = CLng(Alto * 0.95)   ' Situación con respecto al alto de la página
    Cx = CLng(Ancho * 0.05)  ' Situación con respecto al ancho de la página
    Texto = "Final de listado"
    e.Graphics.DrawString(Texto, Fuente, Pincel, Cx, Cy)
    Cy = Cy + Fuente.Height    ' Avance de línea
  End Sub
```

1.2.7 Bucle del listado.

Finalizada la parte de los procedimientos ahora toca el procedimiento de evento del listado.

En el bucle se realiza la llamada a todos los procedimientos del listado.

Siguiendo el orden, primero definir las variables de Cabecera

Después se ejecuta el bucle, condicionado a no fin de archivo.

En éste bucle al haberse inicializado las variables Cabec y Pie, la primera vez solo se ejecuta la cabecera.

En la cabecera, cuando la variable de contador de páginas vale cero, se ejecuta el inicio de impresión.

Cuando se ejecuta la acción de pié de página, la línea que sigue, es la que provoca que se retorne al procedimiento para continuar con su ejecución cuando ya se ha realizado el cambio de página.

```
 e.HasMorePages = True
```

Salvados estos pasos iniciales ya solo queda ir leyendo e ir imprimiendo línea de detalle.

El control de página se realiza con la altura del objeto de impresión y activando la variable “Pie” para realizar el pié de página y salto de página cuando proceda.

Al finalizar el evento se lanza el fin de listado y el cierre de archivos, si es que procede.

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

    Dim Cabec As Boolean = True
    Dim Pie As Boolean = False

    While Not EOF(Canal)
      If Pie Then
        PiePagina(Cy, e)
        e.HasMorePages = True
        Exit Sub
      End If
      If Cabec Then
        If ContPag = 0 Then
          LineaIden(Cy, ContPag, e)    ' Línea de identificación
          InicioImpresion(e)
          e.HasMorePages = True
          Exit Sub
        End If
        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
      LineaDet(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
```

1.2.8 Llamada al Bucle.

La llamada se realiza desde el evento clic de un button.

En el ejemplo es un listado por pantalla con el PrintPreviewDialog.

```
 Me.VistaPreviaDialogo = New PrintPreviewDialog
```

```
 Canal = FreeFile()
 FileOpen(Canal, "Archivo.sec", OpenMode.Input)
 VistaPreviaDialogo.MdiParent = MenuPrincipal  ' MDI
 VistaPreviaDialogo.Document = Hoja
 VistaPreviaDialogo.Show() ' o   VistaPreviaDialogo.ShowDialog
```

El significado del código es el siguiente:

La línea siguiente crea una instancia del objeto PrintPreviewDialog, de esa forma lo que se consigue es que aunque se repita el proceso varias veces, siempre hay una instancia disponible del mencionado objeto, ya que si se cierra la vista previa una vez vista, al ejecutarse de nuevo el programa da el error de no estar disponible por haberse descargado.

```
Me.VistaPreviaDialogo = New PrintPreviewDialog
```

La línea anterior está relacionada con

```
' Objeto de impresión en pantalla
Friend WithEvents VistaPreviaDialogo As PrintPreviewDialog
```

que figura en la definición de variables a nivel de formulario.

La línea siguiente, abre archivos, eso ya dependerá del momento, en el ejemplo como se ha hecho con un archivo secuencial, si deseamos repetirlo, ésta es una forma, o bien abrir archivos en el evento BeginPrint.

```
Canal = FreeFile()
FileOpen(Canal, "Archivo.sec", OpenMode.Input)
```

En ésta línea lo que se consigue es que el formulario de vista previa esté contenido en el formulario de nuestra aplicación, y no sea un elemento independiente. Si se maximiza lo hace contenido dentro de nuestro MDI, que es el nombre de nuestro formulario,

```
VistaPreviaDialogo.MdiParent = Principal
```

La asignación del objeto PrintDocument, Hoja, es realmente lo que permite que el listado se ejecute y se asigne su resultado al objeto PrintPReviesDialog, VistaPreviaDialogo, si no se asigna el objeto a esta propiedad, no se produce el enlace entre uno y otro objeto.

```
VistaPreviaDialogo.Document = Hoja
```

Una vez realizada la asignación del objeto PrintDocument en la propiedad Document, el método Show, permite que se pueda visualizar el resultado del listado.

```
VistaPreviaDialogo.Show()
```

1.2.9 Definición de variables del programa.

El tema de definición de variables ya es un apartado muy personal, así que cada cual decida.

Si que hay que comentar que en esta versión de Vb Net, no existe el contador de páginas de la versión seis de VB, por lo tanto hay que gestionarla por parte nuestra, estableciendo los incrementos de página en el lugar adecuado, nosotros lo hemos colocado en el procedimiento de la línea de identificación.

Hemos buscado en la documentación de VB Net, y figura claro que no lo gestionan, parece raro, pero es así.

```
Dim ContPag As Integer              ' Contador de páginas
Dim TextoCabecera(6) As CabecDetalle
Dim Cy As Long                      ' Coordenada vertical
Dim Canal As Integer
Dim Registro As Archivo
' Objeto de impresión en pantalla, así o con Dim
```

```
Friend WithEvents VistaPreviaDialogo As PrintPreviewDialog
```

1.2.10 Eventos.

El objeto PrintDocument, posee varios eventos, dos de ellos que se activan antes y después del inicio del listado.

En esos dos eventos se puede colocar la apertura y cierre de archivos, en lugar de hacerlo como se ha colocado en el ejemplo anterior, y queda más acorde con lo que es la idea de los programas en VB.

Por lo que podrían quedar como sigue en el ejemplo siguiente.

```
Private Sub Hoja_BeginPrint( _
           ByVal sender As Object, _
           ByVal e As System.Drawing.Printing.PrintEventArgs) _
    Handles Hoja.BeginPrint
  Canal = FreeFile()
  FileOpen(Canal, "Archivo.sec", OpenMode.Input)
End Sub
```

```
Private Sub Hoja_EndPrint( _
           ByVal sender As Object, _
           ByVal e As System.Drawing.Printing.PrintEventArgs) _
    Handles Hoja.EndPrint
  FileClose(CanalMalum)
End Sub
```

Y quitar las correspondientes líneas de código del ejemplo anterior.

El ejemplo que se ha expuesto es un ejemplo sencillo, y dadas las posibilidades de cada uno de los objetos disponibles, se puede mejorar, pero solo pretende ser un punto de partida.

1.3 Los objetos de impresión.

Los objetos de impresión ya los hemos expuesto, ahora vamos a ver las capacidades más importantes de los tres más usados.

1.3.1 PrintDocument.

Como en temas anteriores, su contenido es muy amplio y no tiene sentido su exposición completa aquí, así que éste es el link a este objeto, y ahí tenemos todos los datos.

http://msdn2.microsoft.com/es-es/library/system.drawing.printing.printdocument_members(vs.80).aspx

Vamos a abordar lo más interesante, creemos.

1.3.1.1 Eventos.

Algunos eventos ya los hemos nombrado en el ejemplo anterior, son estos en orden de ejecución.

BeginPrint
QueryPageSettings
PrintPage
EndPrint
Disposed

BeginPrint, PrintPage y EndPrint, ya se ha visto de forma clara cual es su misión.

1.3.1.1.1 QueryPageSettings.

El evento QueryPageSettings, se activa antes del inicio de impresión de cada página, por lo que se puede utilizar para cambiar las características de las propiedades de la hoja que se va a iniciar su impresión.

Es posible imprimir cada página de un documento con una configuración de página diferente. La configuración de página se establece modificando las propiedades individuales de la propiedad QueryPageSettingsEventArgs.PageSettings, o estableciendo el valor de la propiedad PageSettings.

Los cambios que se hacen en esta propiedad solo afectan a la pagina en impresión, no a la predeterminada del documento.

Si la propiedad Cancel se establece a True, se puede cancelar el listado.

1.3.1.1.2 Disposed.

El evento Disposed se produce cuando se ha liberado el objeto.

Es el útlimo paso antes de desaparecer.

Depende de cómo se escriba el programa, o se usen los métodos se dara o no esta circunstancia.

1.3.1.2 Propiedades.

La más útil de todas creemos que es DefaultPageSettings y PrinterSettings, bueno útiles serán todas cuando se han desarrollado.

DefaultPageSettings
DocumentName
OriginAtMargins
PrintControler
PrinterSettings
Site.

1.3.1.2.1 DefaultPageSettings.

Obtiene o establece la configuración de página que se utiliza como predeterminada para todas las páginas que se van a imprimir.

Su gestión debe realizarse en el evento QueryPageSettings.

1.3.1.2.2 PrinterSettings

Dispone de una colección de propiedades referente a la gestión de la impresora que se está utilizando en la impresión.

Por ejemplo el número de copias que se va a imprimir.

El nombre de la impresora que se está usando, la resolución, tamaño margenes y orientación del papel, etc....

1.3.1.3 Métodos.

El más imortante es el Print que es el que nos lanza la impresión, desencadena el PrintPage, cuando usamos la impresora.

1.3.2 PrintDialog.

Este objeto es el estandar que aparece en los programas al realizar la impresión de un documento.

Su uso permite al usuario que elija la impresora de destino, o el número de copias que desea imprimir, etc..

Se puede personalizar a la hora de su uso ya que hay algunas opciones que pueden habilitarse o no.

Un ejemplo de uso de este objeto puede ser:

```
 DialogoImpresora.ShowNetwork = True
 DialogoImpresora.ShowHelp = True
 DialogoImpresora.AllowCurrentPage = False
 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()
```

Con la última línea

```
If DialogoImpresora.ShowDialog() = Windows.Forms.DialogResult.OK Then
Hoja.Print()
```

Si el usuario pulsa aceptar se lanza el listado.

No difiere mucho con respecto a la versión anterior, aunque en ésta versión por ejemplo si se piden varias copias de un listado las lanza el, no es necesario que las gestionemos nosotros, como en la versión anterior.

Para acceder a los datos de sus opciones podemos utilizar el siguiente link.

http://msdn2.microsoft.com/es-es/system.windows.forms.printdialog_members(VS.80).aspx

1.3.3 PrintPreviewDialog.

Este objeto nos permite olvidar la gestión que había que hacer en la versión seis de todo lo referente a la impresión por pantalla utilizando varios picturebox para poder implementar una vista previa, ahora nos lo encontramos hecho, ¡ bien !.

Para acceder a todas sus posibilidades pulsar en el link.

http://msdn2.microsoft.com/es-es/library/system.windows.forms.printpreviewdialog_members(VS.80).aspx

El funcionamiento es en realidad muy sencillo, si olvidamos todo lo referente al PrintDocument claro, y no deseamos complicarnos la vida con todas sus posibilidades.

Este es un ejemplo de cómo puede usarse.

Lo principal es no olvidar como se ha planteado en el ejemplo, es decir el objeto se crea en el programa, y no está insertado en tiempo de diseño, con eso se consigue, que se pueda crear una instancia cada vez que se lanza una impresión y no se genera el error de objeto no disponible cuando se repite el listado.

Es decir el uso en un evento click sería este:

```
Me.VistaPreviaDialogo = New PrintPreviewDialog
VistaPreviaDialogo.MdiParent = MenuPrincipal  ' Formulario MDI
VistaPreviaDialogo.Document = Hoja
VistaPreviaDialogo.Show()
```

Y a nivel de formulario crearíamos el objeto con la línea:

```
' Objeto de impresión en pantalla.
Friend WithEvents VistaPreviaDialogo As PrintPreviewDialog
' Con Dim, una u otra.
Dim VistaPreviaDialogo As PrintPreviewDialog
```

Solo una de las dos.

Después esta línea

```
VistaPreviaDialogo.MdiParent = MenuPrincipal  ' MDI
```

Lo que consigue es que el objeto quede incrustado en nuestro formulario principal, y al maximizarse se integre en él, lo que se asigna es el nombre del formulario MDI.

```
VistaPreviaDialogo.Document = Hoja
```

Esta línea es la que asigna el resultado de la impresión al objeto y la que ejecuta el método Show, la que permite visualizar los resultados.

Desde ese objeto se puede imprimir después lo que se visualiza.

El sistema utilizado en el ejemplo permite que se puedan crear simultáneamente varias instancias de un mismo listado, y puede servir para comparar listados.

1.4 Impresora / Pantalla.

Si deseamos realizar un formulario en el que figure la posibilidad de obtener el listado indistintamente en pantalla o en impresora, el código del evento Click en el button sería algo parecido a esto:

```
Select Case Opcion00.Checked
  Case True   ' activada vista previa
    Me.VistaPreviaDialogo = New PrintPreviewDialog
    VistaPreviaDialogo.Activate()
    VistaPreviaDialogo.MdiParent = Principal
    VistaPreviaDialogo.Document = Hoja
    VistaPreviaDialogo.Show()
  Case Else   ' Activado impresora
    DialogoImpresora.ShowNetwork = True
    DialogoImpresora.ShowHelp = True
    DialogoImpresora.AllowCurrentPage = False
    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 Select
```

El resto del código que se ha expuesto no sufre ningún cambio, ya que el objeto que realiza la impresión, el printdocument Hoja, se asigna según se ejecute una u otra parte del código a la impresora o al objeto de vista previa.

1.5 Con un archivo Random.

Crearíamos una variable, Posición por ejemplo, con la que direccionar la lectura en el achivo Random a nivel de formulario, no del evento, ya que sino perderíamos su valor continuamente, o utilizar Static.

```
Dim Posición as Long
```

Después en el evento Click del button aceptar, asignaríamos el valor inicial del listado en el ejemplo en un textbox llamado Campo01.text.

```
Posicion = CLng(Campo01.Text)
```

El ejemplo se supone con un archivo random en el que los registros usados están con un valor distinto de blanco.

Y ya una vez en el evento PrintPage podría quedar algo parecido a esto:

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

  Dim Cabec As Boolean = True
  Dim Pie As Boolean = False

  While Posicion <= CLng(Campo02.Text)
    FileGet(CanalAgenda, Reg, Posicion)
    Select Case Reg.Nom <> Space(15)
      Case True    ' registro ocupado
        If Pie Then
          PiePagina(Cy, e)
          e.HasMorePages = True
          Exit 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
```

1.6 Para una base de datos.

En este ejemplo se han seguido los mismos criterios que en los anteriores a la hora de estructurar el programa usando los mismos pasos y procedimientos.

Se han adaptado a las necesidades de una base de datos y al uso de su objeto.

Se ha incorporado código en los eventos del objeto PrintDocument.

Se ha modificado el bucle principal del evento para evitar la perdida de datos en el Read del objeto Reader.

El objeto utilizado en este ejemplo es un DataReader, es el adecuado, pues es más rápido que otros objetos de acceso a datos de VB, aunque se puede hacer con otros.

Vamos a colocar las diferencias con respecto a lo ya conocido.

A nivel de formulario creamos los siguientes objetos.

```
Dim TextoCabecera(6) As CabecDetalle
Dim ContPag As Integer    ' Contador de páginas

Dim Conexion As System.Data.OleDb.OleDbConnection
Dim Reader As OleDb.OleDbDataReader
Dim Comando As System.Data.OleDb.OleDbCommand
Dim RecordSet As System.Data.DataTable
```

No se ha usado el constructor New, porque así se genera una nueva instancia cada vez que se repita el listado.

El procedimiento siguiente es el de configurar cabeceras que no ha sufrido ningún cambio.

A continuación el de línea de detalle, sus cambios son mínimos.

```
Private Sub LineaDet(ByVal e As System.Drawing.Printing.PrintPageEventArgs, _
                     ByRef Cy As Long, _
                     ByVal Reader As OleDb.OleDbDataReader, _
                     ByVal Fuente As Font)

Dim Pincel As New System.Drawing.SolidBrush(System.Drawing.Color.Black)
With Reader
 e.Graphics.DrawString(.Item(0).ToString, Fuente, Pincel, TextoCabecera(0).Cx, Cy)
 e.Graphics.DrawString(.Item(1).ToString, Fuente, Pincel, TextoCabecera(1).Cx, Cy)
 e.Graphics.DrawString(.Item(2).ToString, Fuente, Pincel, TextoCabecera(2).Cx, Cy)
 e.Graphics.DrawString(.Item(3).ToString, Fuente, Pincel, TextoCabecera(3).Cx, Cy)
 e.Graphics.DrawString(.Item(4).ToString, Fuente, Pincel, TextoCabecera(4).Cx, Cy)
End With
Cy = Cy + Fuente.Height
End Sub
```

La diferencia es "`Reader.Item(0).ToString`", que es una forma de hacer referencia a cada uno de los campos que se han cargado en la instrucción SQL generada para leer los datos de la tabla de la base de datos, y que se recibe como parámetro en el procedimiento.

En lugar de utilizar `Item(0) es más seguro utilizar Item("NombreDeCampo")` entre comillas.

El siguiente paso será el de las cabeceras, que sigue igual en sus procedimientos.
Fin de página, Inicio de listado y Fin de listado igual.

Los cambios aparecen en el momento que hemos de crear el bucle de lectura, ya que la fuente de datos no tiene nada que ver con un archivo, random o secuencial, así que vamos a ver los cambios.
La llamada al listado se hará igual, desde el evento Clic de un button.
El uso del objeto PrintPreviewDialog también es igual.

```
Private Sub Button1_Click(ByVal sender As Object,_
                          ByVal e As System.EventArgs) _
            Handles Button1.Click
  VistaPrevia = New PrintPreviewDialog
  VistaPrevia.MdiParent = Principal
  VistaPrevia.Document = Hoja
  VistaPrevia.Show()
End Sub
```

En el evento BeginPrint crearemos el objeto con los datos de la tabla, mediante una consulta SQl.
La consulta podrá ser almacenada en la base de datos, o como el ejemplo generada en el programa.

```
Private Sub Hoja_BeginPrint(ByVal sender As Object, _
                    ByVal e As System.Drawing.Printing.PrintEventArgs) _
            Handles Hoja.BeginPrint
  CrearDataReader()
End Sub
```

```
Private Sub CrearDataReader()
  Dim CadenaSQL As String
  Dim Condicion As String
  Dim Clave As String
  CreaRangos(Condicion, Clave)
  CadenaSQL = "Select Exped, Ape1, Ape2, Nombre, Domic From Alumnos " &
_
                " Where " & Condicion & _
                " Order by " & Clave & " ;"
  ContPag = 0     ' Contador de páginas
  Try
    Conexion = New System.Data.OleDb.OleDbConnection
    Comando = New System.Data.OleDb.OleDbCommand

    ' Abrir la base de datos.
    ConfigConexion(Conexion)
    Conexion.Open()
    ' Tipo de comando a ejecutar
    Comando.CommandType = CommandType.Text
    ' Contenido del comando
    Comando.CommandText = CadenaSQL '  "Usp_Alumnos"   ' CadenaSQL
    ' Conexión a utilizar, configurada previamente.
    Comando.Connection = Conexion
    ' Ejecución de SQL
    Reader = Comando.ExecuteReader
  Catch Ex As Exception
    MsgBox(Ex.Message, MsgBoxStyle.Information, "Crear RecordSet")
  End Try
End Sub
```

El procedimiento que a partir de los datos introducidos en el formulario genera la SQL es este:

```
Private Sub CreaRangos(ByRef Condicion As String, _
                       ByRef Clave As String)
```

```
    Dim Desde As String
    Dim Hasta As String

    Select Case Opcion01.Checked  ' código
      Case True
        Select Case Campo01.Text = ""
          Case True
            Desde = "000000"
          Case Else
            Desde = Format(CLng(Campo01.Text), "000000")
        End Select
        Select Case Campo02.Text = ""
          Case True
            Hasta = "999999"
          Case Else
            Hasta = Format(CLng(Campo02.Text), "000000")
        End Select
        Condicion = "(((Alumnos.Exped) >= '" & Desde & "' And " & _
                    " (Alumnos.Exped) <= '" & Hasta & "' ))"
        Clave = "Alumnos.Exped"
      Case Else
        Select Case Campo01.Text = ""
          Case True
            Desde = CStr(StrDup(30, " "))
          Case Else
            Desde = Campo01.Text
        End Select
        Select Case Campo02.Text = ""
          Case True
            Hasta = CStr(StrDup(30, "Z"))
          Case Else
            Hasta = Campo02.Text
        End Select
        Condicion = "(((Alumnos.Ape1) >= '" & Desde & "' And " & _
                    " (Alumnos.Ape1) <= '" & Hasta & "' ))"
        Clave = "Alumnos.Ape1"
    End Select
  End Sub
```

Ya tenemos la instrucción SQL creada y solo hay que hacer que se ejecute.
Vamos a explicar las siguientes líneas de código del CrearDataReader.

```
  Conexion = New System.Data.OleDb.OleDbConnection
  Comando = New System.Data.OleDb.OleDbCommand
```

Estas líneas crean objetos nuevos a partir de la definición a principio de programa, de esa forma se puede repetir el listado sin perder referencia a los objetos en cuestión.

```
    ' Abrir la base de datos.
    ConfigConexion(Conexion)
    Conexion.Open()
```

El primer paso es configurar la conexión, que es el siguiente procedimiento.

```
Public Sub ConfigConexion( _
            ByRef Conexion As System.Data.OleDb.OleDbConnection)
  Dim Ruta As String
```

```
  Dim Proveedor As String
  Proveedor = "Provider=Microsoft.Jet.OLEDB.4.0; "
  Ruta = "Data source=C:\BaseDatos.Mdb;"
  Conexion.ConnectionString = Proveedor & Ruta
End Sub
```

Este procedimiento es una versión muy reducida, pero es válido.

Lo siguiente es la consulta SQL.
Se ejecuta mediante un objeto comando.
En el ejemplo es de tipo texto, no almacenado, que es otra posibilidad, y sin parámetros.
Esos temas los abordaremos en otro apartado.

```
  ' Tipo de comando a ejecutar, de texto
  Comando.CommandType = CommandType.Text
```

Después asignar la cadena SQL al objeto comando, y éste se asigna a la conexión abierta antes.

```
  ' Contenido del comando
  Comando.CommandText = CadenaSQL
  ' Conexión a utilizar, configurada previamente.
  Comando.Connection = Conexion
```

Ahora hay que ejecutar la consulta, se ejecuta y se asigna a un objeto del tipo DataReader.

```
  ' Ejecución de SQL
  Reader = Comando.ExecuteReader
```

Y con eso ya hemos obtenido los datos, ahora hay que usarlos, ¿quién? el evento PrintPage del objeto Hoja, lo vemos.

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

  '   Ejemplo con un datareader

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

  Fuente = Est_Lin_Det   ' estilo línea de detalle
  Try
    If Reader.HasRows Then
      If Cabec Then
        If ContPag = 0 Then
          LineaIden(Cy, ContPag, e)   ' Línea de identificación
          InicioImpresion(e)
          e.HasMorePages = True
          Exit Sub
        End If
        Cabeceras(Cy, e)
        Cabec = False
      End If
      While Reader.Read
        ' Línea de detalle
        LineaDet(e, Cy, Reader, Fuente)
        ' Control de fin de página
        If Cy > e.MarginBounds.Height Then
```

```
          PiePagina(Cy, e)
          e.HasMorePages = True
          Exit Sub
        End If
      End While
    End If
  Catch ex As Exception
    MsgBox(ex.Message, MsgBoxStyle.Information, "Evento Print Page")
  End Try
  FinImpresion(Cy, e)
  e.HasMorePages = False
End Sub
```

Ahora vamos a comentar los cambios.

Como se puede ver no son muchos,

Como curiosidad, que lo es, es que el bucle de lectura no tiene instrucción de lectura, si no que la misma condición del While ejecuta una lectura, lo cual al principio te despista, pues pierdes registros y no se te ocurre por donde.

Toda la parte inicial es idéntica.

El inicio de cabeceras se ha sacado del bucle, sino cada vez que se entra y se sale se pierde un registro, así eso no sucede.

Si se elimina la página de identificación de proceso, se podría mantener la estructura anterior. Ya que no se ejecutaría dos veces la salida del procedimiento, y no se perdería el registro.

Está todo condicionado a que el objeto Reader tenga filas,

```
  If Reader.HasRows Then
```

Si las tiene entonces se ejecuta el apartado de cabeceras.

```
 If Cabec Then
   If ContPag = 0 Then
     LineaIden(Cy, ContPag, e)    ' Línea de identificación
     InicioImpresion(e)
     e.HasMorePages = True
     Exit Sub
   End If
   Cabeceras(Cy, e)
   Cabec = False
 End If
```

Una vez solucionado lo de las cabeceras ejecutamos el bucle.

```
 While Reader.Read
   ' Línea de detalle
   LineaDet(e, Cy, Reader, Fuente)
   ' Control de fin de página
   If Cy > e.MarginBounds.Height Then
     PiePagina(Cy, e)
     e.HasMorePages = True
     Exit Sub
   End If
 End While
```

Y todo incluido en un Try Catch y condicionado a que hayan registros en el Reader.

Es breve y conciso, solo hay de nuevo que la llamada a la línea de detalle incluye el objeto Reader.

Finalizada la ejecución del evento del objeto Hoja, se ejecutará el evento EndPrint.

```
Private Sub Hoja_EndPrint(ByVal sender As Object,_
                  ByVal e As System.Drawing.Printing.PrintEventArgs) _
       Handles Hoja.EndPrint
```

```
  Conexion.Close()
  Conexion.Dispose()
  Comando.Dispose()
  Reader.Close()
End Sub
```

En este evento se cierra la conexión y liberan recursos, de esa forma se puede volver a lanzar el listado con otros parámetros.

Y en el Load del formulario se llamará al ConfigCabecera.