

Conceptos básicos.

- **PROYECTO.** Un proyecto es un conjunto de módulos de formulario, módulos estándar, módulos de clase y archivos de recursos que componen una aplicación. La ventana Proyecto muestra todos los archivos de una aplicación.
- **FORMULARIO.** Un formulario incluye los controles y el código asociado a dicho formulario. Puede compartir código en todo el proyecto colocando el código en un módulo de formulario o en un módulo estándar y declarando el procedimiento como Public.
- **CONTROLES O COMPONENTES.** Los controles son herramientas como cuadros, botones y etiquetas que se disponen en un formulario para permitir la entrada de datos o para presentar resultados. También hacen más atractivos los formularios. Para dibujar controles en un formulario usar la Caja de herramientas.
- **PROPIEDADES.** Usando la ventana Propiedades se definen las propiedades de formularios y controles. Las propiedades especifican los valores iniciales de las características, tales como tamaño, nombre y posición. La ventana Propiedades enumera todas las propiedades y los valores del control o módulo seleccionado actualmente.
- Para hacer que una aplicación responda a las acciones del usuario o a los eventos del sistema, hay que escribir un **código** en lenguaje Visual Basic para los formularios y los controles orientado a los objetos.

Concepto de código:

Es el texto escrito en un programa. El programa se divide en distintos apartados o subrutinas generalmente como respuesta a un evento y se divide en procedimientos (subrutinas) o funciones.

Procedimiento: Bloque de código independiente que se puede ejecutar desde otros bloques de código

Existen dos tipos de procedimientos: las *funciones* y las *subrutinas* (también denominadas *sub*).

Función: Son procedimientos que además devuelven un valor al procedimiento desde donde se le llamó

Subrutina: Simplemente ejecuta código como respuesta a un evento o acción y no devuelven ningún valor

Reglas de nomenclatura al declarar procedimientos o variables:

- Deben comenzar por una letra
- No pueden contener puntos
- No puedes superar los 255 caracteres
- Nombres de formularios y módulos no sobrepasar 40 caracteres
- No pueden tener el mismo nombre que una palabra clave
- Comentarios en el código se inician con el signo apóstrofe (')

Versiones de Visual basic

- VB clásica: Para versiones de Visual Basic 4, 5 ó 6
- VB.NET: Para versiones de Visual Basic 2005 – 2008 ó 2010 permite desarrollar en la plataforma.NET pero algunos comandos del código han cambiado.

Eventos o acciones (sobre un control):

Lista de eventos (Acciones sobre los controles)	
Evento	Acción
Click	Al hacer clic con el mouse (o el teclado)
DragDrop	Al soltar un control arrastrado con el mouse
DragOver	Al Arrastrar un control con el mouse.
GotFocus	El botón obtiene el foco.
KeyDown	Presionar una tecla mientras el botón tiene el foco.
KeyPress	Presionar una tecla y devolver su valor ASCII.
KeyUp	Liberar una tecla mientras el botón tiene el foco.
LostFocus	El botón pierde el foco.
MouseDown	Presionar el botón del mouse sobre el botón.
MouseMove	Mover el puntero del mouse por encima del botón.
MouseUp	Liberar el botón del mouse en el botón.

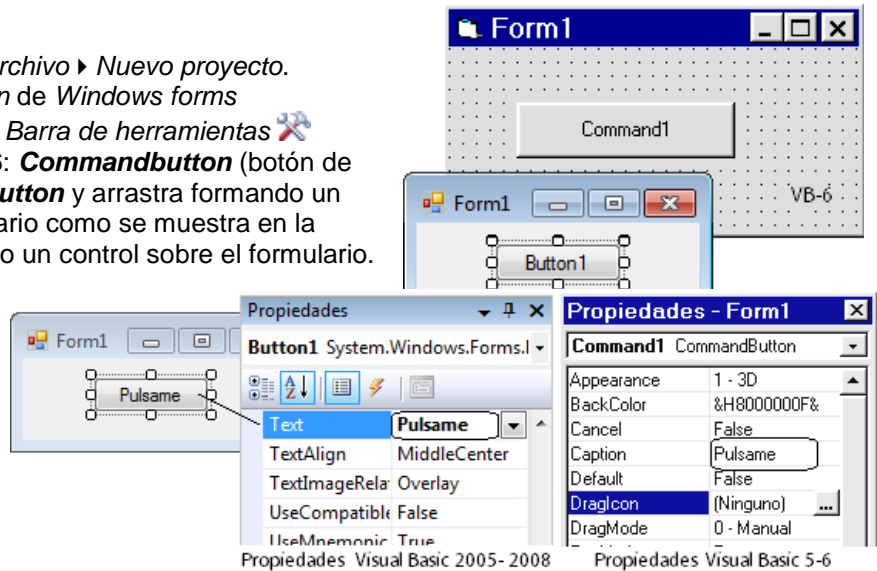
## Ejercicio 1.

Para empezar, elige del menú: *Archivo* ▶ *Nuevo proyecto*.  
Para VB.NET: Escoge: *Aplicación de Windows forms*

1. **Poner un control:** De la *Barra de herramientas* pulsa sobre el botón VB6: **Commandbutton** (botón de comando) o en VBnet: **Button** y arrastra formando un recuadro sobre el formulario como se muestra en la imagen. Ya hemos puesto un control sobre el formulario.

2. **Modificar sus propiedades:**

En la ventana de *Propiedades* (F4), busca la propiedad *Caption* o *Text* (según la versión) y a su derecha escribe: "Pulsame" para cambiar la propiedad del título del botón.



3. **Escribir la acción:** Pulsa *doble clic* sobre el botón para escribir el código de su acción. Asegúrate que el procedimiento es al hacer clic. Verás que ya se ha escrito el principio y final de la subrutina. Escribe dentro la línea resaltada en la imagen:

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Button1.Text = "Muy bien"
    End Sub
End Class
```

```
Objeto: Command1 Procedimiento: Click
Private Sub Command1_Click()
    Command1.Caption = "Muy Bien"
End Sub
```

4. **Comprobar:** Para comprobar su funcionamiento pulsa en el botón ▶ *Iniciar* depuración (*Run*)

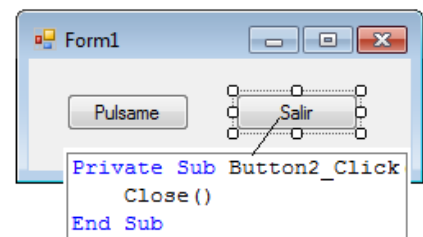
## Ejercicio 2. Salir

Un botón para salir: Repite los mismos pasos para añadir un segundo botón (*Command2* o *button2*) que tenga el título **Salir** y que al pulsar clic se cierre la ventana:

- Version VB6: *Unload Me*
- Version VB .NET: *Close()*

5. **Guardar el proyecto:**

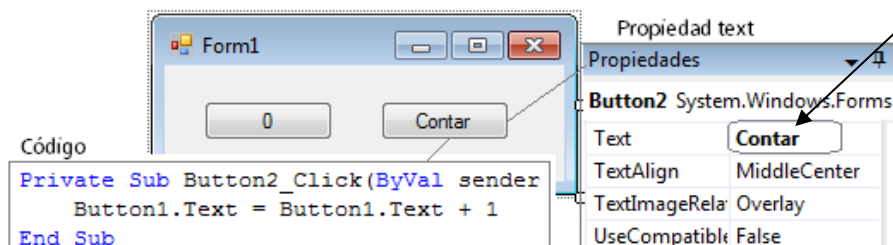
Elige del menú: *Archivo* ▶ *Cerrar proyecto y guardarlo* con el nombre: **Ejercicio1**.



## Ejercicio 3. Contador

Veremos como la acción de un botón puede cambiar la propiedad de otro botón.

- Crea un nuevo proyecto, (*Archivo* ▶ *Nuevo proyecto*) y añade dos botones al formulario: *Button1* y *Button2*



En el *Button1* cambia la propiedad *text* (o *caption*) a "0"  
En el *Button2* cambia el texto del botón a "Contar"  
Pulsa *doble clic* sobre el botón 2 y añade el código de la figura.

- Comprueba y guarda el ejercicio en tu carpeta con el

nombre: **Ejercicio2**

### Resumen. Código para cambiar la propiedad de un control:

- 1º. Se escribe el nombre del control (ejemplo *Command1*)
- 2º. Se pone un punto (.)
- 3º. Se escribe el nombre de la propiedad que queremos cambiar (ejemplo: *Button1.text*)
- 4º. Se iguala al valor que le queremos asignar (ejemplo: *Button1.Text = "hola"*)

### **Ejercicio 4. Mostrar una ventana**

SHOW 1 Muestra una ventana en modo modal (continúa ejecutándose el programa principal)  
 SHOW 0 Muestra una ventana en modo no modal (el programa principal se detiene)  
 SHOW Muestra la ventana actual

▪ Recupera el ejercicio anterior (Archivo ► Abrir proyecto) *Ejercicio2* y Abre el formulario *Form1.vb*  
 Vamos a crear una ventana de información Acerca de...

▪ **Poner el botón:** Añade un nuevo botón *Button3* cuya propiedad *Text (Caption)* sea: *Acerca de...*

▪ **Crear la ventana:**

- VB6: Elige del menú:  
   *Insertar ► Formulario*
  - VB-NET: *Proyecto ► Agregar*  
   - *Windows forms*
- Se mostrará el nuevo formulario *Form2*.

▪ Cambia las propiedades del formulario:

- En la ventana **Propiedades:**
- Name: *Acercade*
  - Text (caption): *Acerca de...*

Añade en su interior un control **Label** cuya propiedad *Text (Caption)* diga: "Programado en Ofimega"

Cambia su propiedad *Font* para que el tipo de letra sea más grande.

Añade luego un botón de comando que diga "Aceptar" para cerrar la ventana y cuyo código sea:

```
Private Sub Command1_Click() : Close() //VB6: Unload Me
```


Luego en la ventana principal (Form1) pulsaremos doble clic sobre el botón Acercade y escribiremos el código:

```
Private Sub Acercade_Click() : Acercade.Show() //VB6: Acercade.Show(1)
```

- Para comprobar su funcionamiento pulsa en el botón *Iniciar* ( ▶ ).
- Si funciona bien guarda el proyecto con el nombre: *Ejercicio4*

### **Ejercicio 5. Crear un Menú**

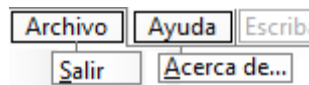
Continuaremos con el ejercicio anterior (Recupéralo si es necesario) y sitúate en el Form1 principal.

- VB6: Elige del menú: *Herramientas ► Editor de menús*.
- VB.NET: Añade el control del cuadro de herramientas: *MenuStrip* 

Crearás los siguientes ítems en el menú:

Una vez creado pulsa *Aceptar*.

*Nota:*



Para que aparezca Salir con la S subrayada debes escribir en la propiedad text: **&Salir** y se utiliza para resaltar la letra "Hotkey" o carácter que activa ese elemento al pulsar la tecla "caliente".

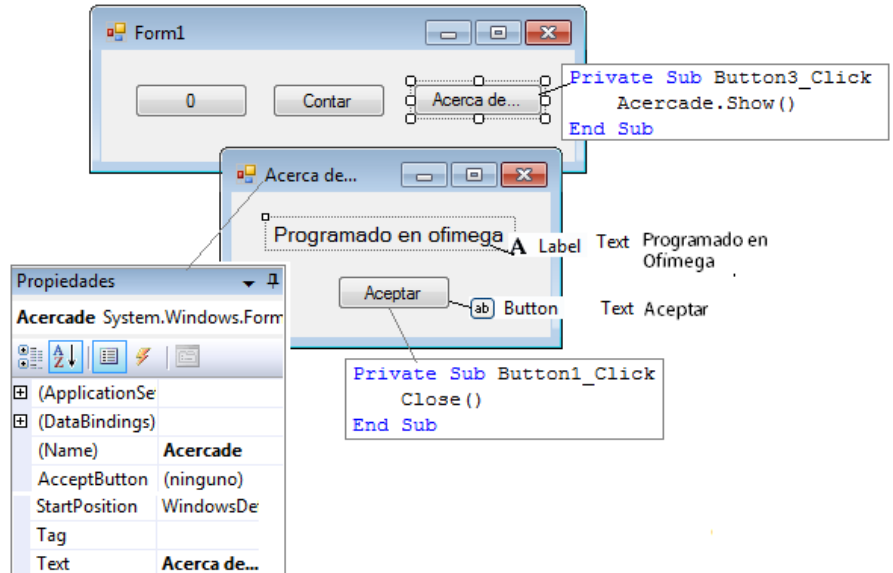
En el Form1, pulsa *doble clic* sobre el elemento **Salir** para escribir el código de la acción: `Close()`

En el Form1, pulsa *doble clic* sobre el elemento **Acerca de...** y **escribe el** Código:

- VB6: `Button3_Click`
- VB.NET: `Button3_Click(AcercaDeToolStripMenuItem, e)`

Para no repetir la misma acción que ya estaba escrita en *Button3* hace una llamada a esa subrutina pasando el parámetro *sender* (el nombre del objeto que lo activa)

Para comprobar su funcionamiento pulsa en el botón *Iniciar* ( ▶ ). Si funciona bien guarda el proyecto.



## **Ejercicio 6. Mostrar mensajes**

**MsgBox (Función):** Muestra un mensaje en un cuadro de diálogo, espera a que el usuario elija un botón y devuelve el valor correspondiente al botón elegido por el usuario.

Sintaxis: **MsgBox ( mensaje, botones, título, archivoAyuda, contexto )**

Significado de los argumentos:

mensaje	Expresión de cadena
botones	Expresión numérica que corresponde los botones
título	Expresión de cadena que se muestra en la barra de título
archivoAyuda	Expresión de cadena que identifica el archivo de Ayuda
contexto	Expresión numérica que es igual al número de contexto de Ayuda

**INPUTBOX** Pedir información. Ej: dato=**Inputbox**(mensaje , título, texto\_defecto, posx, posy, ayuda)

### **Ejercicio:**

1. Crea un nuevo proyecto. (Escoge del menú: *Archivo ▶ Nuevo proyecto* de Windows forms)
2. Añade un control label cuya propiedad caption sea: "¿Quién ganará este año la liga?"
3. Añade **dos** botones de comando cuyos títulos (caption o text) sean Barça y Real Madrid respectivamente.

Evento para el botón Barça:

```
Private Sub Command1_Click()
    MsgBox ("No lo creo") ' -> El texto del mensaje se escribe entre comillas
End Sub
```

Evento para el botón Real Madrid:

```
Private Sub Command2_Click()
    Dim Mensa, tit ' -> Declara una variable llamada Mensa y otra Tit
    Mensa = "Es probable" ' -> Asigna un texto a la variable Mensa
    Tit = "Respuesta" ' -> Asigna texto al título Tit
    MsgBox Mensa, , Tit ' -> Muestra el mensaje con el texto de Mensa y el
    título de la ventan Tit
End Sub
```

Ejemplo que oculta el formulario (Hide) y lo muestra de nuevo (Show)

Crea un tercer botón llamado Betis cuyo código sea:

```
Private Sub Betis_Click ()
    Dim Msg ' -> Declara variable.
    Hide ' -> Esconde el formulario.
    Msg = "Con el Betis no hay juego, pulsa Aceptar para continuar"
    MsgBox Msg ' -> Muestra el mensaje.
    Show ' -> Muestra el formulario de nuevo.
End Sub
```

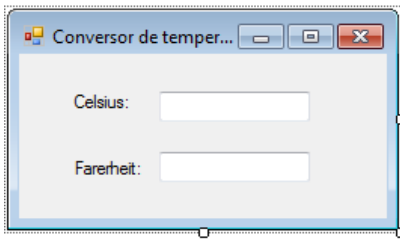
Cierra

y guarda el proyecto con el nombre: Mensajes

### **Ejemplo de mensaje con botones y retorno de información**

```
Msg = " ¿Desea continuar?" ' Define el mensaje.
Estilo = vbYesNo + vbCritical + vbDefaultButton2 ' Define los botones.
Titulo = "MsgBox de demostración" ' Define el título.
Ayuda = "DEMO.HLP" ' Define el archivo de ayuda.
Ctxt = 1000 ' Define el contexto del tema.
Respuesta = MsgBox(Msg, Estilo, Titulo, Ayuda, Ctxt) ' Muestra el mensaje.
If Respuesta = vbYes Then ' El usuario eligió el botón Sí.
    MiCadena = "Sí" ' Ejecuta una acción.
Else ' El usuario eligió el botón No.
    MiCadena = "No" ' Ejecuta una acción.
End If
```

## Ejercicio 7. Cambio de temperaturas.



Esta aplicación de ejemplo convierte temperaturas de Fahrenheit a Celsius y viceversa.

Hay tres pasos en la creación de una aplicación:

- 1º. Crear la interfaz ventana-formulario con sus componentes.
- 2º. Establecer las propiedades de cada control o componente.
- 3º. Escribir el código de acción para cada uno.

1º. Crear la interfaz. (Poner la sartén y los ingredientes)

Creas un nuevo proyecto. (Escoge del menú: *Archivo* ▶ *Nuevo proyecto* de Windows forms).

En VB.net puedes asignar el nombre a la aplicación: **Temperaturas**. Aparecerá un formulario vacío.

A continuación, de la Caja de herramientas selecciona dos cuadros de texto (textbox) que aceptan la entrada del usuario y presentan texto. Uno para las temperaturas Fahrenheit y otro para las temperaturas Celsius.

Para describir el contenido de cada cuadro de texto, añadiremos 2 etiquetas (labels) al formulario.

El usuario no puede modificar las etiquetas pero sí puede escribir en los *TextBoxes*.

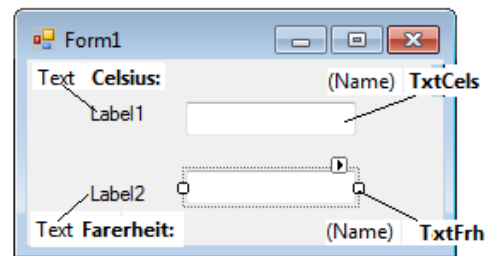
2º. Definir las propiedades. (Preparar los ingredientes)

Las propiedades de un control se cambian en la ventana Propiedades que muestra las propiedades y los valores de un control (puedes pulsar F4 para verla).

Propiedad Name: Selecciona el primer cuadro de texto, vamos a cambiar la propiedad Name. (El nombre lo usa el programador en los controles para referirse a ellos pero no se ven) En este caso, el nombre de este cuadro de texto será: **TxtCels** (es común usar la notación húngara)

Propiedad Caption o Text: Indica el rótulo o título que se muestra un control. En VB clásico se llama Caption y en VB.NET se llama text. Cambia la propiedad Caption o Text de la etiqueta Label1 a: **Celsius**.

- Cambia la propiedad *Name* del text2 a **TxtFahr**
- Cambia la propiedad *caption* o *text* de la etiqueta Label2 a **Fahrenheit**



3º. Escribir el código. (Cocción)

Ahora necesitamos agregar código para hacer que la aplicación responda a las acciones del usuario.

Para convertir temperaturas usaremos las siguientes fórmulas:

$$\text{Cels} = (\text{Fahr} - 32) * 5/9$$
 Para convertir en grados Celsius o centígrados  

$$\text{Fahr} = (\text{Cels} * 9/5) + 32$$
 Para convertir en grados Fahrenheit

A continuación, pulsa doble clic sobre el objeto TextCels. Busca el evento o método KeyPress e introduce el código del procedimiento de evento que se muestra abajo.

Objeto: <b>txtCels</b>	Proc: <b>KeyPress</b>	TxtCels VB.net	<b>KeyPress</b>
<pre>Private Sub txtCels_KeyPress(KeyAscii As Integer)     ' keyascii 13 = Tecla ENTRAR     If (KeyAscii = 13) Then         txtFahr.Text = Val(txtCels.Text * 9 / 5) + 32     End If End Sub</pre>		<pre>Public Class Form1     Private Sub TextCels_KeyPress(ByVal sender As Object, ByVal e As EventArgs)         If e.KeyChar = ChrW(13) Then             TxtFrh.Text = Val(TxtCels.Text * 9 / 5 + 32)         End If     End Sub End Class</pre>	

Si se pulsa la tecla INTRO (código ASCII=13) entonces el texto de txtFahr valdrá:  $\text{Fahr} = \text{Val}(\text{txtCels.text} * 9/5) + 32$

Se debe poner **Val** para convertir el texto en número y poder operar numéricamente con él.

- Ahora asociaremos un procedimiento de evento similar al otro cuadro de texto TxtFahr:  $\text{txtCels.Text} = \text{Val}(\text{txtFahr.Text} - 32) * 5 / 9$
- Para finalizar de trabajar sobre la aplicación, guardar el proyecto. Al elegir Guardar proyecto en el menú Archivo se pedirá un nombre al proyecto.
- Por último verificar la aplicación eligiendo Iniciar en el menú Ejecutar o haciendo clic en el botón Iniciar (>) de la barra de herramientas.
- Cuando ejecutes la aplicación, al escribir un número y presionar la tecla INTRO, se producirá la conversión de Celsius a Fahrenheit o de Fahrenheit a Celsius.

- Para crear un archivo ejecutable del proyecto, usa el comando Crear archivo EXE en el menú Archivo.
- Habrás creado tu primer programa en Visual Basic.

### **Mejora: Tratamiento de errores:**

La instrucción **On Error GoTo línea**

Se utiliza para prevenir el bloqueo del programa y saltar a la subrutina *línea* en el caso de error.

En este caso *línea* es llamada Fallo.

Activa el tratamiento de errores que se produce si los usuarios escriben un valor no numérico en uno de los cuadros de texto.

*Ejercicio:* Añade el código derecho a los eventos keypress de los Textbox anteriores

```
On Error GoTo Fallo
txtFahr.Text = (txtCels.Text * 9 / 5) + 32
Exit Sub
Fallo:
If Err.Number = 13 Then 'error de tipo
txtFahrenheit.Text = "No se pudo hacer la conversión"
Else
Err.Raise Err.Number
End If
```

### **On Error Resume Next**

Hace que la ejecución continúe en la instrucción que sigue a la instrucción que produjo el error en tiempo de ejecución

**On Error GoTo 0:** Deshabilita el control de errores en el procedimiento actual.

### **Ejercicio de Funciones. Hipotenusa**

Una **función** puede ser llamada desde cualquier control y devuelve un valor que puede ser recogido por otro control.

Crema un formulario similar al ejercicio anterior con el siguiente aspecto:

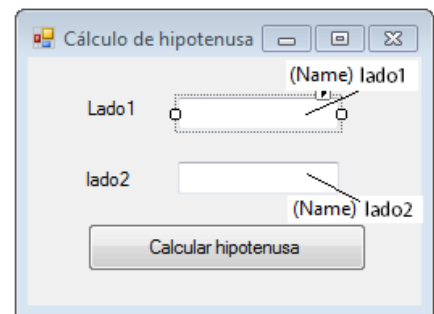
Escribe la siguiente función antes de la línea de Endclass:

```
Function hypotenuse(ByVal side1 As Single, ByVal side2 As Single) As Single
Return Math.Sqrt((side1 ^ 2) + (side2 ^ 2))
End Function
```

Esta función recoge dos valores numéricos sencillos llamados *side1* y *side2* y nos devuelve otro valor numérico sencillo (single)

En el botón Button2 añade el siguiente código:

```
Private Sub Button1_Click()
Button1.Text = hypotenuse(Val(lado1.Text), Val(lado2.Text))
End Sub
```



En el texto del botón se mostrará el resultado de la función a la que se le han dado los valores que tenían los cuadros de texto *lado1* y *lado2*.

Comprueba su funcionamiento y guarda el proyecto con el nombre: **hipotenusa**

### **Identificadores: Variables y constantes.**

Una variable es un nombre que representa a un valor del tipo numérico o de texto que suele cambiar temporalmente.

**Alcance de las variables:**

**Variables Locales:** Dim Nombre As tipo (sólo existen en el procedimiento)


Static Nombre As tipo (valen en la aplicación)

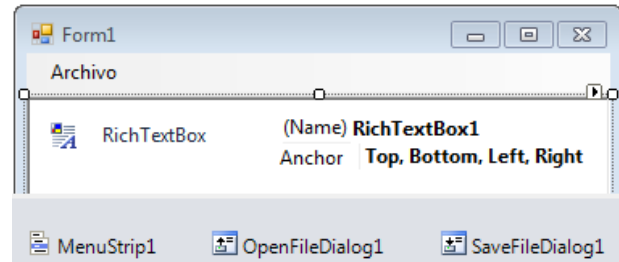
**Variables privadas:** Private Nombre as Tipo (Son reconocidas además en todos los procedimientos del mismo módulo)

**Variables Públicas:** Public Nombre As tipo (Son reconocidas el cualquier procedimiento de cualquier módulo)

**Constantes:** (Public/Private) Const Nombre As tipo = Expresión  
Un dato repetitivo que siempre contiene el mismo valor.

## Ejercicio Procesador de texto en VB.net

- Elige del menú: *Archivo* ▶ *Nuevo proyecto*. - Aplicación de Windows forms: **Editor1**
- Poner los controles básicos: De la Barra de herramientas  añade los controles de la imagen:
  - 1 RichTextBox ajustado al borde de la ventana
  - 1 MenuStrip: Al que añadiremos los elementos:
    - Archivo: Nuevo – Abrir y guardar.
    - 2 ventanas de dialogo: Una para abrir y otra para guardar




- Pulsa doble clic sobre el elemento **Archivo** ▶ **Abrir** de tu formulario. Esto nos creará la subrutina `AbrirToolStripMenuItem` y permitirá introducir el código siguiente:

```
Private Sub AbrirToolStripMenuItem_Click(.....
    With OpenFileDialog1
        .Title = "Abrir archivo de texto enriquecido"
        .Filter = "Archivo de texto enriquecido (*.rtf)|*.rtf"
        If .ShowDialog() = DialogResult.OK And .FileName <> ""
            Then RichTextBox1.LoadFile(.FileName)
        End If
    End With
End Sub
```

- Pulsa doble clic sobre el elemento **Archivo** ▶ **Guardar** de tu formulario. Esto nos creará la subrutina `GuardarToolStripMenuItem` y permitirá introducir el código siguiente:

```
Private Sub GuardarToolStripMenuItem_Click(....
    With SaveFileDialog1
        .Title = "Guardar archivo de texto enriquecido"
        .Filter = "Archivo de texto enriquecido (*.rtf)|*.rtf"
        If .ShowDialog() = DialogResult.OK And .FileName <> ""
            Then RichTextBox1.SaveFile(.FileName)
        End If
    End With
End Sub
```

- Guarda el proyecto con el nombre: **Editor1**
- y comprueba el código pulsando  o [F5].

### Añadir el código al menú Edición:

- Al *MenuStrip* le añadiremos los elementos: Edición: Cortar – Copiar y Borrar
- Programaremos las opciones del menú "Edición", tal como sigue:
  - En Copiar: `Clipboard.SetDataObject(RichTextBox1.SelectedText)`
  - En Cortar: `Clipboard.SetDataObject(RichTextBox1.SelectedText) - RichTextBox1.SelectedText = ""`
  - En Pegar: `Dim iData As IDataObject = Clipboard.GetDataObject()  
If iData.GetDataPresent(DataFormats.RTF) Then  
RichTextBox1.SelectedText = iData.GetData(DataFormats.Text)`

## Tema variables. Tipos y conversión de variables

- Cómo se inicializa una variable:  
Las públicas: Poner en el apartado de **declaraciones** del programa principal:  
DIM nombre AS STRING – Para crear una variable llamada NOMBRE del tipo texto  
DIM numero AS INTEGER – Para crear una variable llamada NUMERO del tipo numérica
- Ámbito de las variables:  
Públicas: Public NUMERO as integer  
Locales: Dim NUMERO as integer (Poner dentro de la misma Subrutina)  
Número = 10 –>Variable local  
Form1.numero = 20 –>Variable a nivel de módulo  
Module1.numero = 30 –>Variable global
- Tipos de variables:

Visual Basic / Access	Basic	Descripcion
Integer - entero	%	2 bytes de -32.000 a 32.000
Long - entero largo	&	4 bytes de - a + 2mil millones
Single - Decimal sencillo	i	Real 4 bytes: +-1,7 E 308
Double - Decimal doble	#	8 bytes doble precisión +-1,7 E 308
Currency - Moneda	@	+-300.000 4 decimales fijo
String - Texto	\$	cadena de texto
Byte - entero corto	-	+255 positivo (no resta)
Boolean - lógico sí/no	-	True o False
Date - Fecha	-	Numérica de Fecha y hora
Variant - Comodín	-	general variable

- Conversión de variables:

Ejemplo:  
Private Sub Command2\_Click()  
Dim numero As Integer  
numero = 2  
Command2.Caption = CStr(numero)  
End Sub

### Funciones de conversión de variables

Cbool – Boolean  
Cbyte – Byte  
Cint – Integer  
CDBl – Double  
CStr – String  
Cdate – Date  
CVar – Variant  
Ccur – Currency

Convertir código ASCII en carácter: **ASC**

Convertir carácter en código ASCII: **CHR**

Ejemplo: Para que mostrar el número 32 : PRINT 32

Para mostrar el carácter ASCII 32: PRINT CHR(32)

El tipo de variable Byte sólo llega hasta 256 números pero es más rápida y ocupa menos memoria.

## Estructuras de control: de decisión y de bucle

Son estructuras muy usadas para evaluar un dato o repetir una secuencia:

- Condicionales: *Simples:* escoge una opción de 2 según la condición: If then  
*Múltiples:* Escoge varias opciones. Select case.
- Cíclicas: Bucles repetitivos: finitos o infinitos.

<u>If ... then ... Else:</u>	<u>Select Case:</u>	<u>Do ... Loop:</u>	<u>For ... Next:</u>
If valor=1 then Instrucciones Else if valor =2 Instrucciones Else Instrucciones End If	Select Case valor Case 1,2,3 Instrucciones Case 4,5,6 Instrucciones Case Else Instrucciones End Select	A=5 Do While A>1 Print A A=A-1 Loop	For x=1 to 20 Step 2 Print x Next

## Ejercicio 9. Cambio de señal luminosa

Tema: Botones uso de IF - ELSEIF - ENDIF

Vamos a diseñar un semáforo que cambie de color cada vez que se pulse el botón.

Debemos dibujar tres imágenes casi idénticas de tres semáforos, cada uno de ellos encendido de un color y luego solaparlas una encima de la otra de manera que parezca que solo hay una.

Estas imágenes son muy fáciles hacer con Paint. Copiando y pegando puedes hacer las tres.

Imagen1;  
Name:imgYellow

Imagen2;  
Name: imgRed

Imagen3;  
Name: imgGreen

Name: Cerrar  
On click: Unload Me  
' Descarga este formulario.

Name: Cambio  
Procedimiento: On click:  
ChangeSignal ' Va al procedure cambio señal.

```
Private Sub ChangeSignal()
' CAMBIA DE COLOR EL SEMAFORO
If imgGreen.Visible = True Then
imgGreen.Visible = False
imgYellow.Visible = True
Elseif imgYellow.Visible = True Then
imgYellow.Visible = False
imgRed.Visible = True
Else
imgRed.Visible = False
imgGreen.Visible = True
End If
End Sub
```

Este ejemplo convierte texto introducido en un control TextBox a mayúsculas. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control TextBox, y después presione F5 e introduzca algo en el control TextBox.

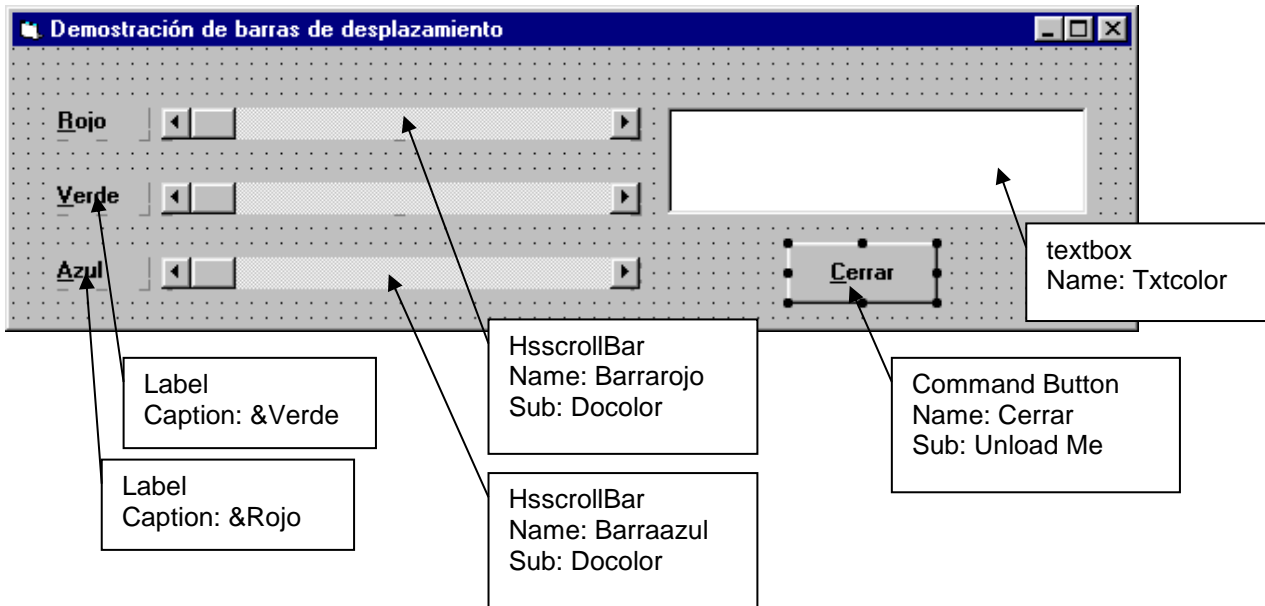
```
Private Sub Text1_KeyPress (KeyAscii As Integer)
Char = Chr(KeyAscii)
KeyAscii = Asc(UCCase(Char))
End Sub
```

## Ejercicio Controles con barras y colores

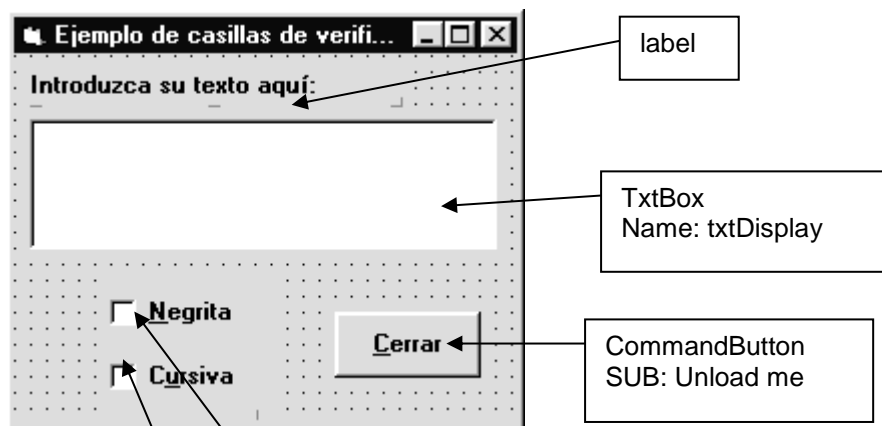
**RGB (Función): RGB (rojo, verde, azul)** → Devuelve un número entero que representa un color RVA.

- Crea un proyecto Nuevo en VB: Archivo ▶ Nuevo proyecto - Aplicación de Windows forms: **Colores**
- Añade de la *Barra de herramientas* los componentes que se muestran en la figura: Puedes sustituir los 3 componentes *HscrollBar* por *TrackBars*
- Pulsa en *Ver código* [F7] y en el objeto general crea la siguiente subrutina:

```
Private Sub DoColor()
    txtColor.BackColor = RGB(Barrarojo.Value, Barraverde.Value, Barraazul.Value)
End Sub
```



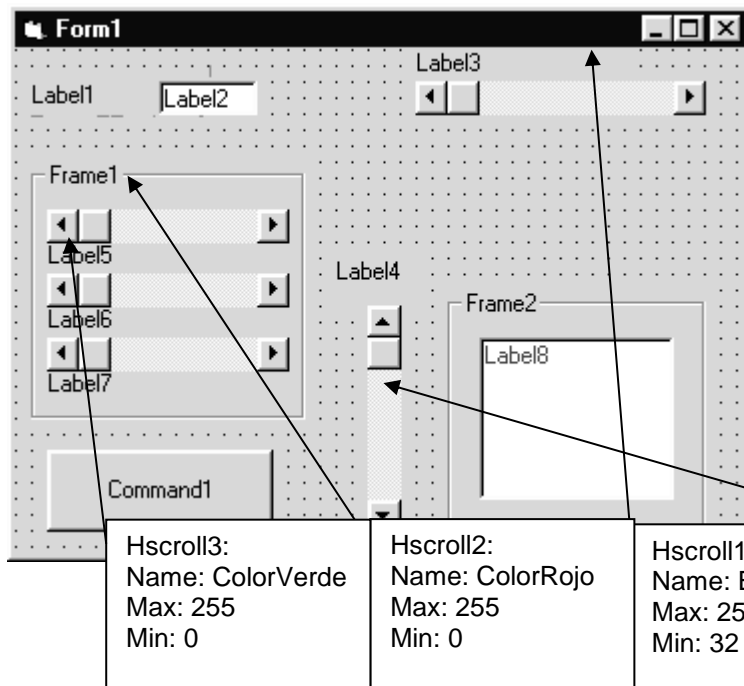
## Ejercicio Casillas de verificación.



```
If ChkBold.Value = vbChecked Then
    txtDisplay.font.Bold = True
Else
    txtDisplay.Font.Bold = False
End If
```

```
If ChkItalic.Value = vbChecked Then
    txtDisplay.Font.Italic = True
Else
    txtDisplay.Font.Italic = False
End If
```

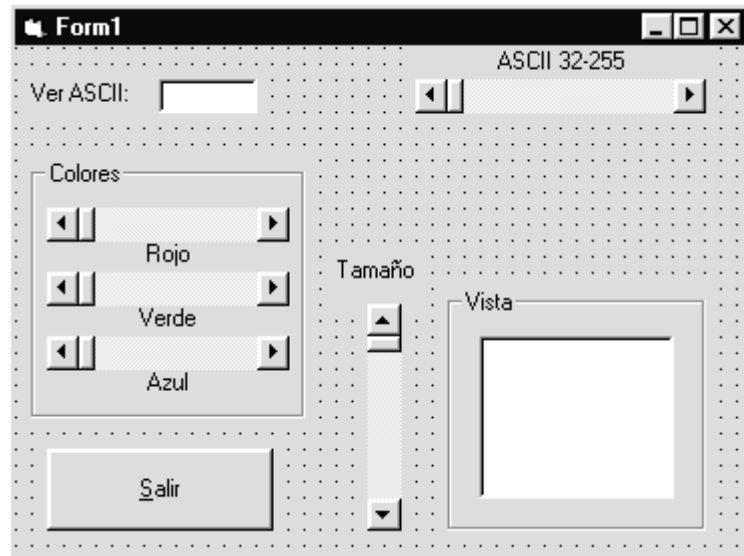
**Ejercicio Resumen de controles más usuales.**



Introduce los controles que se muestran en la figura y cambia las siguientes **propiedades** entre otras para que quede como en la 2ª figura:

- Label 8: Nombre: VerAscii  
Backcolor: Blanco  
BorderStyle: 1-Fixed single
- Command1: Caption: &Salir
- Label 1: Nombre: Carácter
- Label2: Nombre: ASCII  
Backcolor: blanco  
BorderStyle: 1-Fixed single
- Label5: Nombre: rojo  
Caption: Rojo  
Alignment: 2-Center
- Frame1: Nombre: colores  
caption: colores

Hscroll3: Name: ColorVerde Max: 255 Min: 0	Hscroll2: Name: ColorRojo Max: 255 Min: 0	Hscroll1: Name: Barracodigo Max: 255 Min: 32	Hscroll5: Name: TamCaracter Max: 80 Min: 6
---	--	---	---



Cambiar **label2** (ASCII) por **text1** (ASCII)

```
Private Sub Form_Load()
    VerAscii.FontName = "Times New Roman" 'tipo de letra
End Sub

Private Sub TamCaracter_Change()
    VerAscii.FontSize = TamCaracter.Value 'tamaño
    Label4.Caption = "Tamaño-" + Str(TamCaracter.Value)
End Sub
```

```
Private Sub Colorrojo_Change()
    VerAscii.ForeColor = RGB(Colorrojo.Value, Colorverde.Value, Colorazul.Value)
    Rojo.Caption = "Rojo -" + Str(Colorrojo.Value)
End Sub

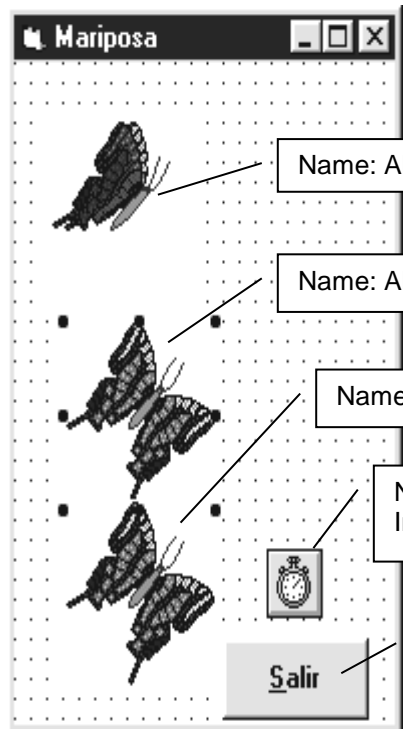
Private Sub Barracodigo_Change()
    ASCII.Text = Chr(Barracodigo.Value)
    VerAscii.Caption = Chr(Barracodigo.Value)
    Label3.Caption = "Valor Ascii-" + Str(Barracodigo.Value)
End Sub

Private Sub ASCII_Change()
    VerAscii.Caption = Chr(Barracodigo.Value)
    If ASCII.Text <> "" Then Barracodigo.Value = Asc(ASCII.Text)
End Sub
```

*Nota: Idem colorverde y colorazul*

## Ejercicio . Mariposa.

A cada impulso del timer la imagen1 se intercambia con las otras dos y avanza. Las otras dos imágenes deben estar ocultas y el timer activado (enabled).



Name: Alasabiertas

Name: Alascerradas

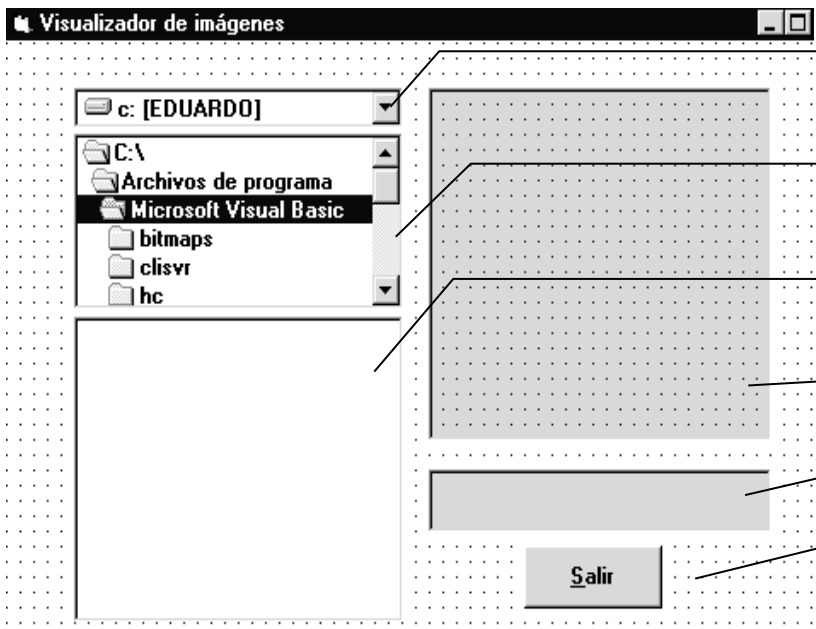
Name: Imagen1

Name: Timer1  
Interval: 200

Name: Command1  
Caption: &Salir

```
Private Sub Timer1_Timer()
    Static Flag As Integer
    Imagen1.Move Imagen1.Left + 20, Imagen1.Top - 5
    If Flag = 1 Then
        Imagen1.Picture = alasabiertas.Picture ' Muestra la
        imagen de la mariposa con las alas abiertas.
        Flag = 0 ' Alterna el valor.
    Else
        Imagen1.Picture = alascerradas.Picture ' Muestra
        la imagen de la mariposa con las alas cerradas.
        Flag = 1
    End If
End Sub
```

## Ejercicio . Visor de imágenes



Drive1  
Name: Drive1

Dir1  
Name: Dir1

File1  
Name: File1

Image  
Name: Open

Label: Label1

Command1 - Salir

```
Private Sub File1_DblClick()
    ' en (C:\) , Path tiene una barra (\) al final
    ' en cualquier otro nivel, no hay barra (\) final.
    If Right(file1.Path, 1) <> "\" Then
        label1.Caption = file1.Path & "\" & file1.filename
    Else
        label1.Caption = file1.Path & file1.filename
    End If
    Form1.open.Picture = LoadPicture(label1.Caption)
End Sub
```

```
Private Sub Form_Load()
    Drive1.Drive = App.Path
    Dir1.Path = App.Path
End Sub

Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub

Private Sub Dir1_Change()
    file1.Path = Dir1.Path
End Sub
```

**Animación. Protector de pantalla**

Picture: Bola.bmp

Ventana: Form1 - Name: DemoForm

Timer: Timer1 – Interval: 1

CommanButton1: Empezar – Timer1.enabled=true

CommanButton2: Parar – Timer1.enabled=false

Private Sub Form\_Load(): mover = 1Private Sub Timer1\_Timer()

Select Case Mover

Case 1

' Mueve el gráfico a la izquierda y hacia arriba 20 twips usando el método Move.

Bola.Move Bola.Left - 20, Bola.Top - 20

' Si el gráfico alcanza el borde izquierdo del formulario, se mueve a la derecha y hacia arriba.

If Bola.Left &lt;= 0 Then

Mover = 2

' Si el gráfico alcanza el borde superior del formulario, se mueve a la izquierda y hacia abajo.

Elseif Bola.Top &lt;= 0 Then

Mover = 4

End If

Case 2

' Mueve el gráfico a la derecha y hacia arriba 20 twips.

Bola.Move Bola.Left + 20, Bola.Top - 20

' Si el gráfico alcanza borde derecho del formulario, se mueve a la izquierda y hacia arriba. Se determina el borde derecho del formulario restando el ancho del gráfico del ancho del formulario.

If Bola.Left &gt;= (DemoForm.Width - Bola.Width) Then

Mover = 1

' Si el gráfico alcanza el borde superior del formulario, se mueve a la derecha y hacia abajo.

Elseif Bola.Top &lt;= 0 Then

Mover = 3

End If

Case 3

' Mueve el gráfico a la derecha y hacia abajo 20 twips.

Bola.Move Bola.Left + 20, Bola.Top + 20

' Si el gráfico alcanza el borde derecho del formulario, se mueve a la izquierda y hacia abajo.

If Bola.Left &gt;= (DemoForm.Width - Bola.Width) Then

Mover = 4

' Si el gráfico alcanza el borde inferior del formulario, se mueve a la derecha y hacia arriba. Se determina el borde inferior del formulario restando la altura del gráfico de la altura del formulario menos 680 twips debido a la altura de la barra de título la barra de menús.

Elseif Bola.Top &gt;= (DemoForm.Height - Bola.Height) - 680 Then

Mover = 2

End If

Case 4

' Mueve el gráfico a la izquierda y hacia abajo 20 twips.

Bola.Move Bola.Left - 20, Bola.Top + 20

' Si el gráfico alcanza el borde izquierdo del formulario, se mueve a la derecha y hacia abajo.

If Bola.Left &lt;= 0 Then

Mover = 3

' Si el gráfico alcanza el borde inferior del formulario, se mueve a la izquierda y hacia arriba.

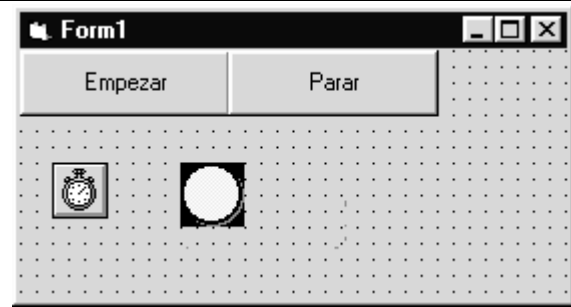
Elseif Bola.Top &gt;= (DemoForm.Height - Bola.Height) - 680 Then

Mover = 1

End If

End Select

End Sub



**Ejercicios de caracteres. Key Press**

- **KeyPress:** Ocurre cuando el usuario presiona y libera una tecla

Sintaxis: **Private Sub Form\_KeyPress(valorascii As Integer)**

valorascii : Es un entero que devuelve un código de tecla numérico del estándar ANSI

- **ASC(cadena)** Devuelve el código de carácter correspondiente a la primera letra de una cadena de caracteres.

```
Ejemplo:      MiNúmero = Asc("A")      ' Devuelve 65.
              MiNúmero = Asc("a")      ' Devuelve 97.
              MiNúmero = Asc("Apple")   ' Devuelve 65.
```

- **CHR(función)** Devuelve el carácter asociado con el N° ASCII

```
Ejemplo:      MiCaracter = Chr(65)    ' Devuelve A.
              MiCaracter = Chr(97)    ' Devuelve a.
              MiCaracter = Chr(62)    ' Devuelve >.
              MiCaracter = Chr(37)    ' Devuelve %.
```

- **UCASE(cadena)** Devuelve una cadena que se ha convertido en mayúscula.

```
Minusculas = "Hola a todos"          ' La Cadena por convertir.
Mayusculas = UCase(Minusculas)      ' Devuelve "HOLA A TODOS".
```

**Ejercicio:**

Este ejercicio convierte texto introducido en un control TextBox a mayúsculas. Para probar este ejemplo, pega el código en la sección Declaraciones de un formulario que contenga un control TextBox, y después presiona F5 e introduce algo en el control TextBox.

```
Private Sub Text1_KeyPress (KeyAscii As Integer) 'KeyAscii será el N° Ascii de la tecla
Char = Chr(KeyAscii)                            ' Char es una variable, CHR devuelve el carácter
KeyAscii = Asc(UCase(Char))                     ' Devuelve el código de carácter de la primera letra
End Sub
```

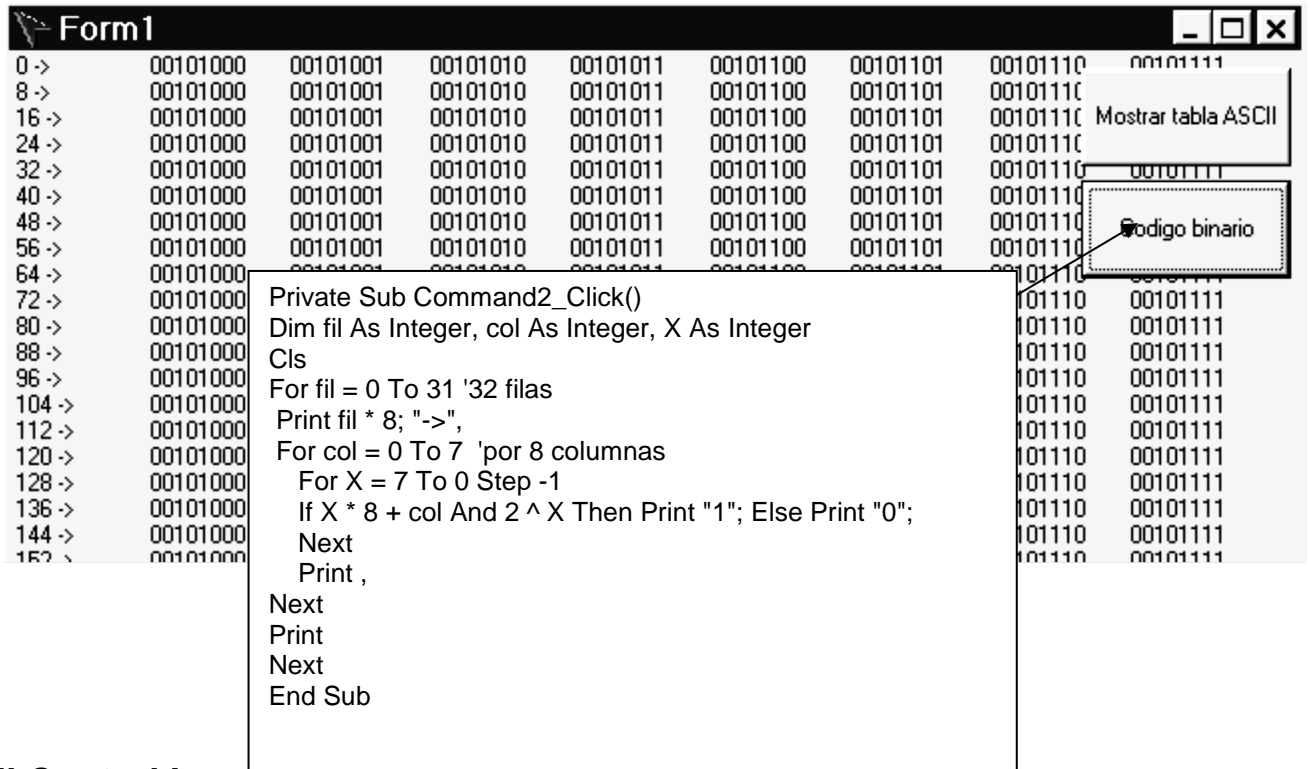
**Ejercicio: Crea tu propia tabla de caracteres ASCII. (Matriz multidimensional)**

Vamos a ordenar desde el número 32 al 255 en grupos de ocho la tabla ASCII, ya que los anteriores no son imprimibles.

- Crear formulario nuevo con un botón de comando.
- Doble clic sobre el botón y escribir el código siguiente:

```
Dim I As Byte
Dim J As Byte
Cls
For I = 4 To 31 'Desde el 32 hasta el 255
  For J = 0 To 7 'una fila de ocho
    Print I * 8 + J; "="; Chr(I * 8 + J),
  Next
Print " " 'Para que salte a la siguiente
línea
Next
End Sub
```

**Ampliación del ejercicio anterior:** Crear un segundo botón con el siguiente código:



```

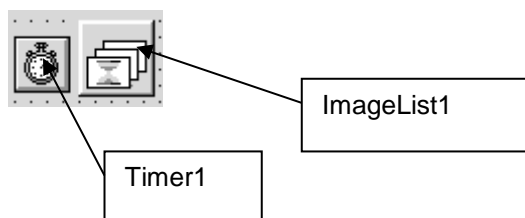
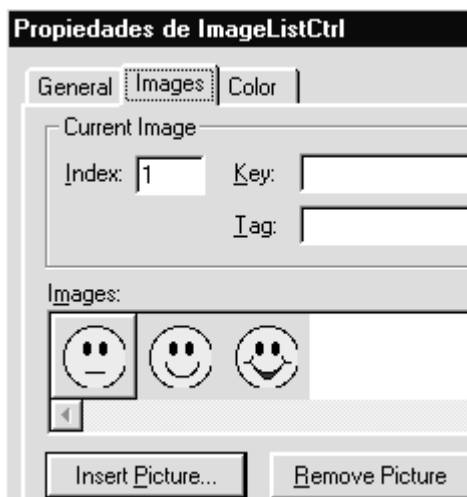
Private Sub Command2_Click()
Dim fil As Integer, col As Integer, X As Integer
Cls
For fil = 0 To 31 '32 filas
Print fil * 8; "->",
For col = 0 To 7 'por 8 columnas
For X = 7 To 0 Step -1
If X * 8 + col And 2 ^ X Then Print "1"; Else Print "0";
Next
Print ,
Next
Print
Next
Print
Next
End Sub

```

## El Control ImageList

### Ejercicio de Animación. Cambiar el Icono de la aplicación cada 3 segundos

1. Abrir el ultimo proyecto realizado.
2. Insertar el componente **Image List** (Si no existe, añadirlo de: Herramientas – controles personalizados Microsoft Common Controls 6.0). Este control No funciona por sí solo.
3. En la propiedad personalizado: Imágenes – Insertar Picture: e insertar 3 imágenes de iconos(ejemplos: Archivos de programa\microsoft Visual basic\icons\Misc\Face01 – Face02 – Face03
4. En las declaraciones del apartado general añadir el código: **Dim x As Integer**
5. Al cargar el formulario añadir el código: **x = 0**
6. Insertar un control Timer. Propiedad Interval: 300 y añadirle el siguiente código al timer:



```

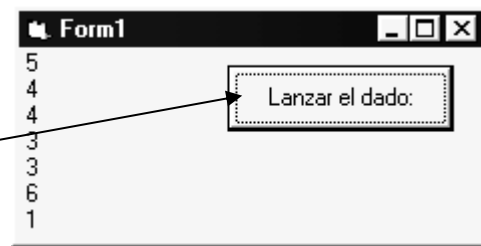
Private Sub Timer1_Timer()
x = x + 1
Form1.Icon = ImageList1.ListImages(x).Picture
If x = 3 Then x = 0 ' al llegar a la última pone la 1ª
End Sub

```

## Funciones aleatorias

Funciones aleatorias: Utilizar el Randomize del timer  
 Función **Rnd** = valor comprendido entre 0 y 1 que podemos multiplicar por una escala y obtener la parte entera con la función (Cint).

```
Private Sub Command1_Click()
  Print Cint(Rnd * 6)
End Sub
```



## Funciones trigonométricas y de tiempo

Tener en cuenta que los ángulos estarán expresados en radianes y no en grados. Para pasar de grados a radianes multiplicar por PI y dividir por 180.

**Time:** Devuelve la hora actual del sistema.

Ej: MiHora = #4:35:17 PM# Asigna una hora. Time = MiHora  
 Establece MiHora en la hora del sistema.

**Date:** Devuelve la fecha actual del sistema. Ej: MiFecha = Date

**Now:** Devuelve la fecha y la hora actuales de acuerdo a la configuración del sistema de su PC.

**Format:** Se usa para formatear un dato.

MiCadena = Format(MiHora, "h:m:s")

-> Devuelve "17:4:23".

MiCadena = Format(MiHora, "hh:mm:ss AMPM")

-> Devuelve "05:04:23 PM".

MiCadena = Format(MiFecha, "dddd, mmm d aaaa")

-> Devuelve "Miércoles, 27 de Ene de 1993".

Hour(horadigital): Devuelve un número entero entre 0 y 23, inclusive, que representa la hora del día.

Day(fecha):

### Funciones trigonométricas

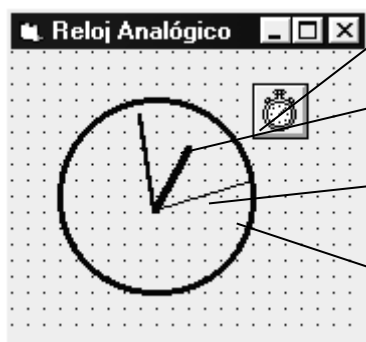
Sin	–	Seno
Cos	–	Coseno
Tan	–	Tangente
Atn	–	Arcotangente



```
Private Sub Reloj_Timer()
  ' Una vez por segundo actualizamos la etiqueta
  label
  Etiqueta.Caption = Format(Now, "hh:mm:ss")
End Sub
```

### Ejercicio reloj Digital.

### Ejercicio reloj analógico:



Timer: Name: reloj

Horas: Line Name: Horas  
 X1: 1065 X2: 1320 Y1: 1185 Y2: 720

Segundo: Line Name: segundos  
 X1: 1065 X2: 1800 Y1: 1185 Y2: 960

Name: Shape1  
 Shape: shape=3 circle

```
Private Sub Reloj_Timer()
  Const Radianes = 3.1415927 / 180 ' constante para conversión
  ' Posicionamos las tres agujas
  Horas.X2 = 2500 + Sin(Hour(Now) * 30 * Radianes) * 1000
  Horas.Y2 = 2500 - Cos(Hour(Now) * 30 * Radianes) * 1000
  Minutos.X2 = 2500 + Sin(Minute(Now) * 6 * Radianes) * 1500
  Minutos.Y2 = 2500 - Cos(Minute(Now) * 6 * Radianes) * 1500
  Segundos.X2 = 2500 + Sin(Second(Now) * 6 * 3.1415927 / 180) * 2000
  Segundos.Y2 = 2500 - Cos(Second(Now) * 6 * 3.1415927 / 180) * 2000
End Sub
```

## Ejercicio. Juego. Control de una nave

Propiedades del form1:  
Windows State: Maximized  
Border style: None

Progressbar  
Name: energia

Image  
Name: Nave

Picture box 1  
Align al top

Timer1  
Interval: 1  
Enabled:false

ImageList1

Image  
Name: Navearri

Image

Image  
Name: Naveder

Energía: [Progress Bar]

General Images Color

Current Image  
Index: 7 Key: [ ]  
Tag: [ ]

Images:

Arranque Paro [Left Arrow] [Right Arrow] [Up Arrow] [Down Arrow]

Toolbar1 con imagelist1  
Align: bottom

ImageList1

Procedimiento para la barra de botones  
Private Sub Toolbar1\_ButtonClick  
Select Case Button.Index  
Case 1: ' boton arranca  
Timer1.Enabled = True  
Case 2: ' boton para  
Timer1.Enabled = False  
Case 3: ' derecha  
izqui = izqui - 5  
nave.Picture = naveder.Picture  
Case 4: ' izquierda  
izqui = izqui + 5  
nave.Picture = naveiz.Picture  
Case 5:  
arri = arri - 5  
nave.Picture = navearri.Picture  
Case 6:  
arri = arri + 5  
nave.Picture = navearri.Picture  
Case 7:  
Unload Me ' sale  
End Select  
energía.Value = energía.Value - 1  
maniobras = maniobras + 1  
'estado.Panels(1) = "Maniobras: " + maniobras  
End Sub

Declaraciones generales:  
Dim izqui, arri As Integer

Procedimiento para el timer 1  
Private Sub Timer1\_Timer()  
If nave.Left > form1.width Then nave.Left = 1  
If nave.Top > form1.top Then nave.Top = 1  
If nave.Left < 1 Then nave.Left = form1.width-1  
If nave.Top < 1 Then nave.Top = form1.top-1  
nave.Left = nave.Left + izqui  
nave.Top = nave.Top + arri  
End Sub

Private Sub Form\_Load()  
izqui = 0  
arri = 0  
energía.Value = 100  
End sub

Ampliaciones:

1. Añadir dibujo para escenario de fondo y si la nave aterriza en un punto exacto premiar
2. Parar si la energía llega a cero
3. Añadir medidores de potencia

## Efecto Protector de pantalla. pixels aleatorios

Ejercicio: Crear un formulario en blanco y poner un componente Timer1 en un formulario cuya propiedad Interval sea 1 y su acción a cada pulso sea la siguiente:

Funciones empleadas:

- Método PSET(Xpos,Ypos),RGB(R,G,B): Crea un punto de un color especificado.
- RND: Devuelve un número aleatorio comprendido entre 0 y 1.
- RGB(R,G,B): Devuelve un número entero que representa un color RVA.
- ScaleWith: Devuelven o establecen el número de unidades de medida horizontal (ScaleWidth) y vertical (ScaleHeight) del interior de un objeto al utilizar métodos gráficos o al colocar controles.

```
Private Sub Timer1_Timer()  
    ' Crea colores RGB aleatorios.  
    R = 255 * Rnd  
    G = 255 * Rnd  
    B = 255 * Rnd  
    'crea posiciones aleatorias X e Y  
    xpos = Rnd * ScaleWidth  
    ypos = Rnd * ScaleHeight  
    PSet (xpos, ypos), RGB(R, G, B)
```

## Acceso a bases de datos con el control Data

Usar el motor de base de datos Jet: Los Objetos de acceso a datos (DAO) y el control Data utilizan el motor de base de datos Microsoft Jet para tener acceso a las bases de datos. En Visual Basic puedes utilizar el control **Data** para crear aplicaciones de bases de datos para una gran variedad de formatos de base de datos o usar el acceso a datos directo a objeto **DAO**.

### Elementos de una tabla

En una base de datos Jet, las filas de las tablas se denominan registros y las columnas campos.

**Clave principal:** cada tabla tiene una clave principal. La clave principal es un campo o una combinación de campos que es exclusiva para cada fila de la tabla.

**Índices:** Los índices de tabla de la base de datos son listas ordenadas en las que se puede buscar más rápidamente que en las propias tablas.

### Propiedades del control Data:

Para especificar qué datos desea

recuperar, debe establecer las propiedades `DatabaseName` y `RecordSource` de un control Data. Además, puede establecer las siguientes propiedades y métodos:

- `Connect` Especifica el tipo de base de datos que se va a abrir
- `Exclusive` Si `Exclusive = True`, ninguna otra aplicación podrá abrir la base de datos.
- `ReadOnly` Si puede actualizar la base de datos
- `Recordset` Es un objeto que contiene el conjunto de registros devuelto por el control Data
- `BOFAction` y `EOFAction`: determinan qué acción realiza el control Data cuando las propiedades `BOF` o `EOF` del `Recordset` son `True`.
- `Refresh`: Actualiza el objeto `Recordset`

### El objeto Recordset.

¿Qué es un `Recordset`? Todo el conjunto de registros al que hace referencia un control Data se denomina conjunto de registros o `Recordset`. El `Recordset` se almacena en la memoria, transfiriéndose al disco si es necesario. Ejemplo de métodos para el `Recordset`:

- Método `AddNew`: para agregar un nuevo registro a un conjunto de registros: `Data1.Recordset.AddNew`
- Método `UpdateRecord` del control Data: para guardar el registro actual en una base de datos:
- Método `CancelUpdate` del control Data: para cancelar un método `AddNew` o `Edit`: `Data1.CancelUpdate`
- El método `Delete`: para quitar un registro de una base de datos. El registro eliminado seguirá siendo el actual hasta que el usuario se desplace a otro registro diferente, como se muestra en el siguiente código:

```
Sub cmdUpdate_Click ()
    Data1.UpdateRecord
End Sub
```

```
Sub cmdDelete_Click ()
    Data1.Recordset.Delete
    Data1.Recordset.MoveNext
    If Data1.Recordset.EOF Then
        Data1.Recordset.MoveLast
    End If
End Sub
```

### Eventos:

`Validate`: para comprobar los datos antes de guardar un registro en la base de datos. ejemplo:

```
Private Sub Data1_Validate (Action As Integer, Save As Integer)
    Dim iResponse As Integer
    If Save = True Then
        iResponse = MsgBox ("¿Guardar cambios?", vbYesNo)
        If iResponse = vbNo Then
            Save = False
            Data1.UpdateControls ' Actualiza campos.
        End If
    End If
End Sub
```



**Evento Reposition:** para modificar la apariencia de un formulario o realizar una acción necesaria cuando se desplace a un nuevo registro.

El código siguiente ilustra cómo mostrar el número de registro actual:

```
Private Sub Data1_Reposition()
    Data1.Caption = "Número de registro " & Data1.Recordset.AbsolutePosition + 1
End Sub
```

#### Mostrar un mensaje de error personalizado:

```
Private Sub Data1_Error(DataErr As Integer, Response As Integer)
    If DataErr = 3022 Then 'Error de clave duplicada
        MsgBox "Escriba un número de Id. de empleado único"
        txtEmpID.SetFocus
        Response = 0
    Else
        Response = 1 'muestra el mensaje de error estándar
    End If
End Sub
```

## Base de datos de teléfonos

Pasos a seguir para crear una aplicación en VB con acceso a datos:

1. Crear una base de datos mediante Access (o desde VB elegir del menú: Complementos → Administrador de datos), que contenga al menos una tabla con los distintos campos de información: Nombre, apellidos, teléfono, etc. Y guardar esta base de datos con un nombre. Ej: **Teléfonos.mdb**. Luego salir de Access
2. Entrar en visual Basic, crear en un nuevo proyecto un formulario de Visual basic y añadirle un control **Data**, que proporciona la manera más sencilla de tener acceso a información de bases de datos existentes.
3. Después de agregar el control Data al formulario, se debe buscar su propiedad **DatabaseName** para especificar la base de datos a la que desea tener acceso. Ej: C:\Mis documentos\Teléfonos.mdb
4. Después hay que definir la propiedad **RecordSource** para presentar la lista de tablas y consultas almacenadas en la base de datos.
5. Agregar un **cuadro de texto** al formulario. Para que el cuadro de texto presente información de nuestra base de datos, primero debemos asociarlo al control Data1 en su propiedad **Datasource**.
6. Después definir el campo que el cuadro de texto va a presentar mediante la propiedad **DataField**. Ej: Nombre.
7. A continuación, agregar una **etiqueta** y definir su propiedad Caption como Nombre.
8. Repetir con el resto de los campos añadiendo los otros controles de texto. Cuando se ejecuta la aplicación, los cuadros de texto presentarán la información de campos que ha especificado. Haciendo clic en los botones del control Data nos desplazaremos a través de los registros de la base datos.

(Todos estos pasos los podemos generar agregando el complemento diseñador de formularios de datos a Visual Basic)

Botón agregar:  
Data1.Recordset.AddNew

Botón eliminar:  
Data1.Recordset.Delete

Botón refrescar:  
Data1.Refresh

Botón actualizar:  
Data1.UpdateRecord

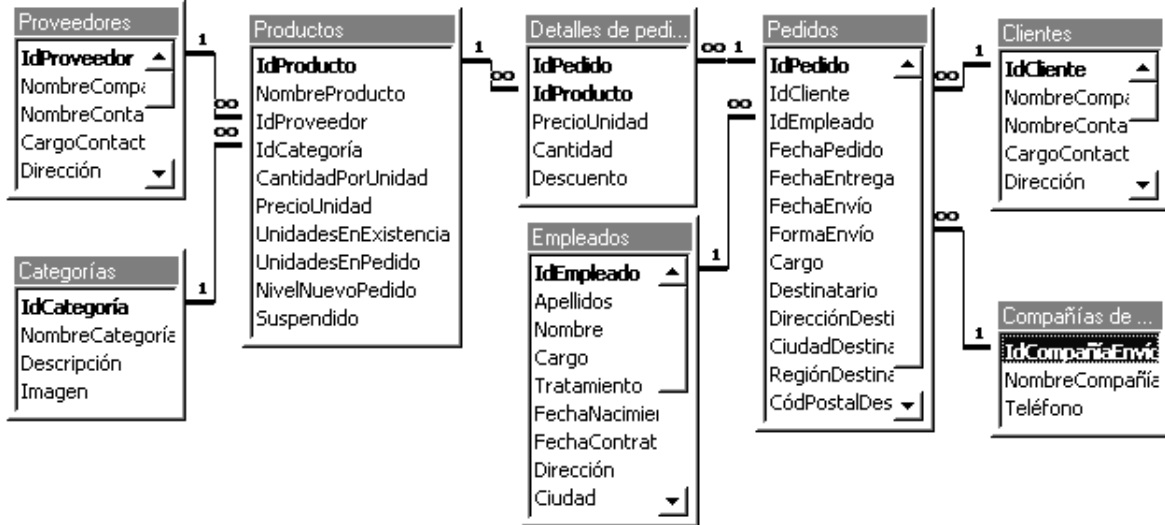
Botón cerrar:  
Unload Me

TextBox:  
Datasource: Data1  
Datafield: E-Mail

Data1:  
DatabaseName: telefonos.mdb

## Ejercicio de Acceso a la base de datos Neptuno.

En este ejercicio utilizará el Asistente para formularios de datos con el fin de crear un formulario de entrada de datos Maestro/Detalles sirviéndose de las tablas Pedidos y Detalles de pedidos de la base de datos Neptuno, que se encuentra en la carpeta de Access o se suministra junto con este ejercicio y que contiene las siguientes tablas relacionadas:



### Crear el formulario Pedidos con un asistente:

1. Crea un nuevo proyecto EXE estándar. en VB 4.0 ó y mediante el Administrador de complementos, carga el Asistente para formularios de datos. En Visual basic 6.0 elige Nuevo proyecto – Asistente para aplicaciones formulario para acceso a datos.
2. Selecciona las opciones del Asistente para formularios de datos según se muestra en la siguiente tabla.

Opción	Valor
Formato de base de datos	Access
Nombre de la base de datos	<Carpeta de instalación de VB>\Nwind.mdb
Distribución del formulario	Maestro/Detalles
Origen de registros principal	Tabla Pedidos
Campos seleccionados	IdPedido, IdCliente, IdEmpleado, FechaPedido, Destinatario
Origen de registros de detalle	Tabla Detalles de pedidos
Campos de detalle	Todos los campos de la tabla Detalles de pedidos
Campo para vincular Maestro y Detalles	IdPedido
Controles disponibles	Todos
Nombre del formulario	frmPedidos

5. En el cuadro de diálogo Propiedades del proyecto, establece Objeto inicial a frmPedidos.

6. Ejecuta la aplicación. Guarda el formulario con el nombre **Pedidos.frm** y guarda el proyecto con el nombre **Pedidos.vbp**

	IdPedido	IdProducto	PrecioUnidad	Cantidad	Descu
▶	10251	22	16,8	6	
	10251	57	15,6	15	
	10251	65	16,8	20	
*					

**Usar el control DBCombo:**

Mejoraremos el formulario Pedidos reemplazando el cuadro de texto IdEmpleado por un control de cuadro combinado enlazado a datos (DBCombo). Esto permitirá a los usuarios seleccionar empleados por el nombre sin que tengan que conocer sus Id.

**Para agregar un control Data**

1. Abrir el formulario Pedidos.
2. Agregar un nuevo control Data al formulario.
3. Asignar a la propiedad Name el valor datEmpleados.
4. Asignar a la propiedad DatabaseName el valor <Carpeta de instalación de VB>\Nwind.mdb.
5. Asignar a la propiedad RecordSource el valor Empleados.
6. Asignar a la propiedad Visible el valor False.

El control Data está oculto porque sólo sirve para proporcionar información al cuadro combinado.

**Para agregar un control DBCombo**

1. Dentro del formulario Pedidos, reemplazar el cuadro de texto **IdEmpleado** por un control de cuadro combinado enlazado a datos. (Agregar el control al cuadro de herramientas antes de utilizarlo).

2. Establecer las propiedades RowSource y ListField del cuadro combinado enlazado a datos de modo que contengan el apellido del empleado. Asignar a la propiedad **RowSource** el valor datEmpleados y a propiedad **ListField** el valor Apellidos.

3. Asignar a la propiedad DataSource valor del control Data creado por el Asistente para formularios de datos y a las propiedades DataField y BoundColumn el valor IdEmpleado. Comprobar que funciona.

The screenshot shows a form titled 'Pedidos' with several input fields and a data grid. The fields are: IdPedido (10251), IdCliente (VICTE), IdEmpleado (Leverling), FechaPedido (8/8/94), and Destinatario (Victuailles en stock). Below the fields is a data grid with columns: IdPedido, IdProducto, PrecioUnidad, Cantidad, and Descu. The grid contains three rows of data. At the bottom of the form are buttons for 'Agregar', 'Eliminar', 'Renovar', 'Actualizar', and 'Cerrar'. A status bar at the bottom indicates 'Registro: 4'.

	IdPedido	IdProducto	PrecioUnidad	Cantidad	Descu
▶	10251	22	16,8	6	
	10251	57	15,6	15	
	10251	65	16,8	20	

la  
el

**Usar el evento Validate:**

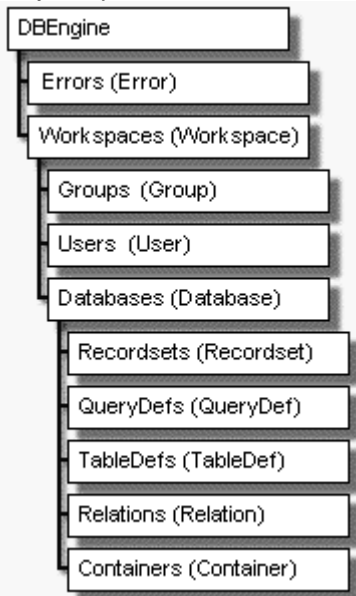
Utilizaremos el evento **Validate** del control Data para confirmar los cambios realizados en la tabla Pedidos.

1. Modifica el evento Validate para el control Data de la tabla Pedidos.
2. Si se ha realizado alguna modificación en el registro actual, utiliza un cuadro de mensajes para preguntar al usuario si desea guardar los cambios.
3. Si el usuario responde que No a la solicitud de guardar los cambios, elimina los cambios mediante el argumento Save. Ejecuta la aplicación.

```
Dim iResponse As Integer
If Save = True Then 'los datos han cambiado
    'ignora los cambios si el usuario no quiere guardar
    iResponse = MsgBox("¿Guardar cambios?", vbYesNo)
    If iResponse = vbNo Then
        Save = False
    End If
End If
```

## Acceso a datos mediante DAO:

Para tener acceso y manipular datos directamente mediante programa (sin control data) debes comprender la jerarquía de DAO.



Por ejemplo, la instrucción siguiente abre una base de datos:

```
Set dbMydb = DBEngine.Workspaces(0).OpenDatabase("Mibd.mdb")
```

Donde:

Objeto	Descripción
Workspace	Contiene las bases de datos abiertas.
Database	Una base de datos abierta.
Recordset	Los registros de una tabla o los registros que resultan de ejecutar una consulta.
QueryDef	Una definición almacenada de una consulta.
TableDef	Una definición almacenada de una tabla.

Para tener acceso a datos mediante programa con DAO, primero debe utilizar el objeto **DBEngine** para abrir un espacio de trabajo.

**DBEngine** es el objeto de nivel superior dentro del modelo de objetos DAO.

Contiene y controla todos los demás objetos de la jerarquía de DAO.

El código siguiente muestra el uso de la propiedad Version y el método

```
MsgBox "El número de versión de DAO es " & DBEngine.Version
DBEngine.RepairDatabase "C:\MIBD.MDB"
```

RepairDatabase:

Objeto **Workspace**: define una sesión para el usuario y determina cómo interactúa su aplicación con los datos. Si abre una base de datos sin especificar un objeto Workspace, se utilizará DBEngine.Workspaces(0) como valor predeterminado. El ejemplo de código siguiente utiliza el método CreateWorkspace del objeto DBEngine para crear un nuevo espacio de trabajo que usará el motor de base de datos Microsoft Jet para interactuar con los datos:

```
Dim wspNew as Workspace
Set wspNew = DBEngine.CreateWorkspace ("NewJetWorkspace", "Admin", "", dbUseJet)
```

Utilizar la instrucción **Set** para asignarla a una base de datos o un Recordset.

Para abrir una base de datos:

```
Dim dbMydb As Database
Set dbMydb = OpenDatabase ("C:\Archivos de programa\DevStudio\VB\Nwind.mdb")
```

Para crear un recorsSet:

```
Dim dbMydb As Database
Dim recEmployees As Recordset
Set dbMydb = OpenDatabase ("C:\Archivos de programa\DevStudio\VB\Nwind.mdb")
Set recEmployees = dbMydb.OpenRecordset ("Empleados")
```

Para mostrar registros:

```
txtEmpID.Text = recEmployees("IdEmpleado") 'o puede utilizar una sintaxis alternativa
```

```
txtEmpID.Text = recEmployees![IdEmpleado]
```

Modos de desplazarse a través de un recordset

```
recEmployees.Move +5 'Avanza 5 filas
recEmployees.Move -6 'Retrocede 6 filas
```

```
recEmployees.MoveNext
If recEmployees.EOF Then
    recEmployees.MoveLast
    Beep
End If
```

RecordCount: MsgBox "El Recordset contiene " & recEmployees.RecordCount & " registros."

AbsolutePosition: lblStatus.Caption = "Está en el registro " & rs.AbsolutePosition + 1

Bookmark: Es un marcador para guardar una ubicación y volver a ella:

```
Dim varBookmark As Variant
```

```
varBookmark = recEmployees.Bookmark 'Guardar ubicación
```

```
' [Aquí iría el código]
```

```
recEmployees.Bookmark = varBookmark 'Volver a la ubicación guardada
```

### Modificar datos con DAO:

Actualizar registros: Para actualizar un registro primero debe invocar al método Edit, llenar los campos del registro con los datos apropiados y, después, invocar al método Update.

```
Private Sub cmdUpdate_Click ()
```

```

recEmployees.Edit
recEmployees("Apellidos") = txtLName.Text
recEmployees.Update

```

```
End Sub
```

Agregar registros: Utiliza el método AddNew para agregar un nuevo registro al Recordset. El registro no se guardará realmente en la base de datos hasta que ejecute el método Update.

```

Sub cmdAdd_Click ()
    ClearFields
    recEmployees.AddNew

```

```
End Sub
```

Eliminar registros con Delete:

```

Private Sub cmdDelete_Click()
    recEmployees.Delete
    recEmployees.MoveNext
    If recEmployees.EOF Then
        recEmployees.MoveLast
    End If
    FillFields 'Procedimiento escrito por el usuario

```

```
End Sub
```

Usar DAO con el control Data:

En una misma aplicación puede combinar DAO y el control Data. Esto es útil si desea utilizar controles enlazados en una aplicación y seguir teniendo la misma flexibilidad de uso que ofrece DAO.

Puede asignar a la propiedad Recordset del control Data un objeto Recordset creado mediante el método OpenRecordset, como se muestra en el ejemplo siguiente:

```

Dim recEmployees As Recordset
Set recEmployees = dbMydb.OpenRecordset ("Empleados")
Set datEmployees.Recordset = recEmployees ' Establece el control Data

```

Buscar registros en un Dynaset:

```

recEmployees.FindFirst "[IdEmpleado] = 5"
recEmployees.FindFirst "[Apellidos] Like 'A*'" ' Uso del comodín *

```

Buscar un registro mediante SQL:

```

strSQL = "Select * From Empleados " & "Where [Apellidos] = " & "" & txtLastName.text & ""
set recEmployees = dbMydb.OpenRecordset (strSQL, dbOpenDynaset)

```

En la mayoría de los casos, los métodos Find son más lentos que crear un Recordset mediante SQL.

Buscar en una tabla:

Utilice el método Seek para buscar un registro en un Recordset de tipo table

```

rs.Index = "Persona"
rs.Seek "=", "Davolio", "Nancy"

```

Tipos de recorddset:

Table: corresponde a una única tabla y se puede modificar. Sólo se carga en memoria el registro actual

Dynaset: es un conjunto dinámico de registros que puede contener campos de una o más tablas de una base de datos. En general, las tablas son el tipo de Recordset más rápido y los dynaset son el tipo más flexible.

**Usar consultas:**

Consulta de selección: Para utilizar una consulta de selección almacenada, basta con proporcionar el nombre de la consulta en el método OpenRecordset del objeto Database.

```
Set recProducts = dbMydb.OpenRecordset("Productos pedidos")
```

También puede utilizar el método OpenRecordset del objeto QueryDef para recuperar los resultados de la consulta.

```

Dim qryProducts As QueryDef
Set qryProducts = dbMydb.QueryDefs("Productos pedidos")
Set recProducts = qryProducts.OpenRecordset

```

Establecer un parámetro: Si la consulta almacenada requiere parámetros, utilice la colección Parameters del objeto QueryDef para establecer parámetros:

```
Set qryProducts.Parameters("Fecha de inicio") = CDate(txtBeginDate.text)
```

Crear una consulta nueva: Utilice el método CreateQueryDef del objeto Database

```
Set qryNewQuery = dbMydb.CreateQueryDef ("Mi consulta", "Select * From Empleados")
```

consulta de acciones: es una instrucción SQL que actualiza, elimina o inserta registros en una base de datos. Utilice el método Execute del objeto Database para ejecutar una consulta de acciones.

```
dbMydb.Execute "Actualizar productos", dbFailOnError
```

```
Msgbox "Esta consulta cambió " & dbMydb.RecordsAffected & " registros."
```

### Usar SQL:

SELECT: Utilice la instrucción SELECT de SQL para recuperar registros. La sintaxis de SELECT es la siguiente:

```
SELECT lista_campos
FROM nombres_tabla
WHERE condiciones_búsqueda
ORDER BY lista_campos
```

El ejemplo siguiente recupera únicamente los campos Apellidos e IdEmpleado en los que IdEmpleado es mayor que 5, en orden descendente según el Id. de empleado:

```
SELECT [Apellidos], [IdEmpleado]
FROM Empleados
WHERE [IdEmpleado] > 5
ORDER BY [IdEmpleado] DESC
```

SELECT con múltiples tablas: La cláusula INNER JOIN especifica que desea obtener los registros de la tabla Categorías cuyo Id. de categoría coincida con el mismo IdCategoría en la tabla Productos.

```
strSQL = "SELECT Categorías.[NombreCategoría], " & "Productos.[NombreProducto] " & _
"FROM Categorías " & "INNER JOIN Productos ON " & "Productos.[IdCategoría] =
Categorías.[IdCategoría]"
```

WHERE: Utilice la cláusula WHERE para limitar la selección:

'Basic Where:

```
strSQL = "SELECT * FROM Empleados WHERE [Apellidos] = " & "" & txtLName.text & """
```

'Where In

```
strSQL = "SELECT Empleados.[Apellidos] FROM Empleados " & "WHERE Empleados.Región in
('NY','WA')"
```

'Where Between

```
strSQL = "SELECT [IdPedido] FROM Pedidos WHERE " & "([FechaPedido] BETWEEN #01/01/93# AND
#31/01/93#)"
```

```
strSQL = "SELECT [IdPedido] FROM Pedidos WHERE " & "([FechaPedido] BETWEEN #" &
CDate(txtStartDate.TEXT) & "# AND #" & CDate(txtEndDate.TEXT) & "#)"
```

ORDER BY: Utilice la cláusula ORDER BY para crear un Recordset en un orden determinado:

```
SELECT * FROM Empleados ORDER BY [Apellidos] DESC
```

### SELECT de SQL:

Puede crear dinámicamente una cadena SQL en tiempo de ejecución y utilizar el método OpenRecordset para crear un Recordset basado en la instrucción SQL.

```
Dim strSQL As String
```

```
strSQL = "Select * From Empleados"
```

```
Set recEmployees = dbMydb.OpenRecordset (strSQL)
```

UPDATE de SQL: La instrucción Update de SQL modifica registros existentes. La opción dbFailOnError hace que la actualización se deshaga si se produce un error durante la actualización. Además, se invoca al controlador de errores.

```
strSQL = "UPDATE [Detalles de pedidos] SET [Descuento] = 1 " & "WHERE [Descuento] = 0"
```

```
dbMydb.Execute strSQL, dbFailOnError
```

INSERT de SQL: para insertar un nuevo registro.

```
strSQL = "INSERT INTO EMPLEADOS " & "([Nombre], [Apellidos]) " & "VALUES ('Juan', 'Gracia')"
```

```
dbMydb.Execute strSQL, dbFailOnError
```

## Ejercicio de Acceso a datos mediante DAO: Neptuno.

### Parte 1: Crear y desplazarse por un Recordset.

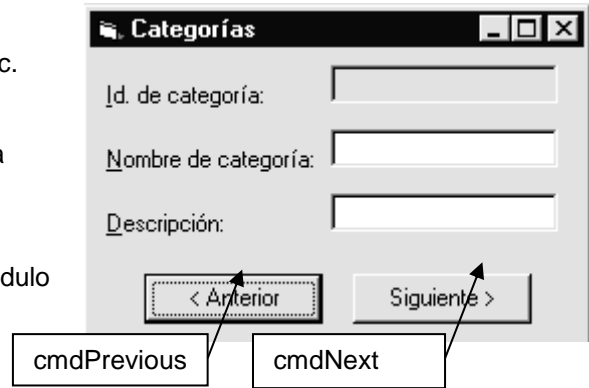
Para crear el formulario Categorías

1. Crea un nuevo proyecto EXE estándar de Visual Basic.
2. Asigna al formulario predeterminado el nombre frmCategories.
3. Agrega controles al formulario, como se muestra en la siguiente ilustración.

Para abrir una base de datos y crear un Recordset

En el formulario Categorías, declara dos variables de módulo como se muestra en el siguiente ejemplo:

```
Dim dbCurrent As Database
Dim recCategories As Recordset
```



En el procedimiento de evento **Unload**, cierra (con Close) la base de datos. Prueba la aplicación.

```
Private Sub Form_Load()
    Set dbCurrent = OpenDatabase("../Neptuno.mdb")
    Set recCategories = dbCurrent.OpenRecordset("Categorías")
    recCategories.MoveFirst
    FillFields
End Sub

Sub FillFields()
    ' este procedimiento poblará los campos del formulario.
    ' será llamado desde múltiples procedimientos.
    lblCategoryID.Caption = recCategories.Fields("IdCategoría")
    txtCategoryName.Text =
recCategories.Fields("NombreCategoría")
    txtDescription.Text = recCategories.Fields("Descripción")
End Sub
```

```
Private Sub cmdPrevious_Click()
    recCategories.MovePrevious
    If recCategories.BOF Then
        Beep
        recCategories.MoveFirst
    Else
        FillFields
    End If
End Sub
```

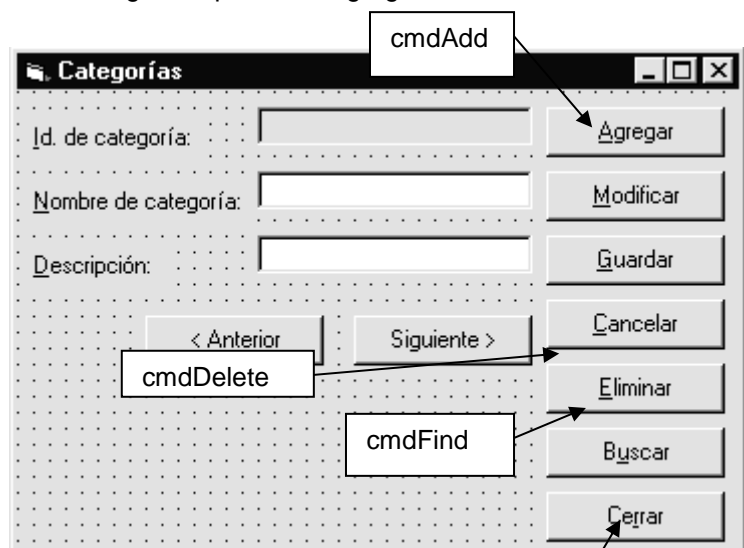
```
Private Sub cmdNext_Click()
    recCategories.MoveNext
    If recCategories.EOF Then
        Beep
        recCategories.MoveLast
    Else
        FillFields
    End If
End Sub
```

### Ejercicio Neptuno. Parte 2. Agregar y modificar registros.

Para modificar el formulario: Modifica el formulario Categorías; para ello, agrega los botones de comando que se muestran en la siguiente ilustración.

Nota Como el campo IdCategoría es de tipo Autonumérico, contendrá automáticamente un nuevo valor en cuanto ejecute el método AddNew. Entonces puede poner el valor de este campo en la etiqueta lblCategoryID. Por ejemplo, ejecute el siguiente ejemplo de código inmediatamente después del método AddNew.

```
lblCategoryID.Caption =
recCategories("IdCategoría")
```



Aquí se muestra el código para los botones de Agregar, Modificar, Guardar, Eliminar:

Cancelar y

```
Private Sub cmdAdd_Click() ' agregar un nuevo registro
    recCategories.AddNew
    lblCategoryID.Caption = recCategories.Fields("IdCategoría")
    txtCategoryName.Text = ""
    txtDescription.Text = ""
    ButtonEditAddMode
    txtCategoryName.SetFocus

End Sub

Private Sub cmdEdit_Click() ' modificar
    recCategories.Edit
End Sub

Private Sub cmdSave_Click() ' guardar
    recCategories.Fields("NombreCategoría") = txtCategoryName.Text
    recCategories.Fields("Descripción") = txtDescription.Text
    recCategories.Update
    recCategories.Bookmark = recCategories.LastModified
End Sub

Private Sub cmdCancel_Click() ' cancelar
    recCategories.CancelUpdate
    FillFields
End Sub

Private Sub cmdDelete_Click() ' borrar
    recCategories.Delete
    recCategories.MoveNext
    If recCategories.EOF Then
        recCategories.MoveLast
    End If
    FillFields
End Sub
```

Para escribir código para el botón Buscar, utilizamos la función InputBox para pedir a los usuarios una cadena que represente un fragmento del campo Descripción.

```
Private Sub cmdFind_Click() ' buscar
    Dim strSQL As String
    Dim strAnswer As String
    strAnswer = InputBox("Introduzca una parte de la descripción", "Buscar registros")
    strSQL = "Select * from categorías where [Descripción] like " & _
        "*" & strAnswer & "*"
    ' ejecuta la consulta
    Set recCategories = dbCurrent.OpenRecordset(strSQL)
    If recCategories.RecordCount = 0 Then
        'no se han encontrado registros
        MsgBox "No se han encontrado registros. Se muestran todos los registros."
        Set recCategories = dbCurrent.OpenRecordset("Categorías")
    Else
        'al menos ha encontrado un registro
        recCategories.MoveFirst
        FillFields
    End If
End Sub
```

## Parte 4. Usar consultas.

Para consultar los diez productos más caros

1. Crea un nuevo formulario.
2. La consulta guardada Los diez productos más caros de la base de datos Neptuno.mdb devuelve los diez productos más caros. En el procedimiento de evento Form\_Load del nuevo formulario, crea un Recordset basado en esta consulta y muestra el resultado en un control DBGrid, como se muestra en la siguiente ilustración:

DiezProductosMasCaros	PrecioUnidad
Vino Côte de Blaye	263,5
Salchicha Thüringer	123,79
Buey Mishi Kobe	97
Mermelada de Sir Rodney	81
Langostinos tigre Carnarv	62,5
Raclet de queso Courdav.	55
Manzanas secas Manjim	53
Tarta de azúcar	49,3

```
Private Sub Form_Load()
    Dim dbCurrent As Database
    Set dbCurrent = DBEngine.Workspaces(0).OpenDatabase("C:\Archivos de programa\DevStudio\VB\Nwind.mdb")
    Set datProducts.Recordset = dbCurrent.OpenRecordset("Los diez productos más caros")
End Sub
```

sugerencia: Con un control Data oculto en el formulario y asignar a la propiedad DataSource del control DBGrid el control Data. Después, puede hacer que la propiedad Recordset del control Data sea igual al conjunto de registros devuelto por la consulta almacenada.

Modificaciones opcionales:

Desactivar los botones durante la ejecución:

Cuando un usuario se desplaza por el formulario, desactivar los cuadros de texto y los botones Guardar y Cancelar de forma que no se realicen cambios a los datos a menos que el usuario haga clic en el botón Modificar o Agregar. Cuando un usuario haga clic en el botón Modificar o Agregar, activar los cuadros de texto y los botones Guardar y Cancelar. Desactivar todos los demás botones del formulario.

```
Private Sub cmdEdit_Click()
    ButtonEditAddMode
    recCategories.Edit
End Sub

Sub ButtonEditAddMode()
    cmdSave.Enabled = True
    cmdCancel.Enabled = True
    cmdAdd.Enabled = False
    cmdEdit.Enabled = False
    cmdDelete.Enabled = False
    cmdFind.Enabled = False
    cmdClose.Enabled = False
    cmdPrevious.Enabled = False
    cmdNext.Enabled = False
    txtCategoryName.Enabled = True
    txtDescription.Enabled = True
End Sub
```

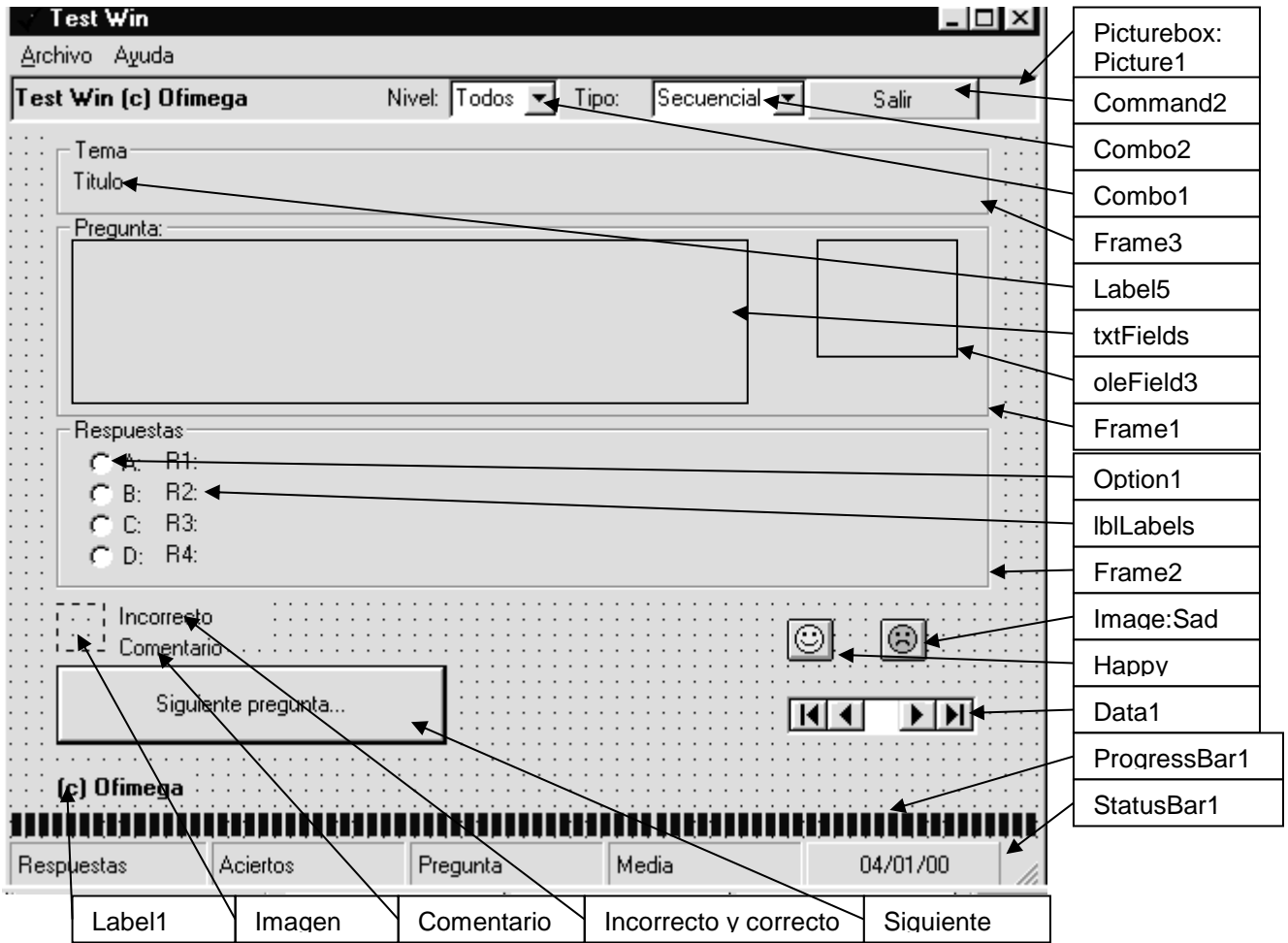
País	Apellidos	Nombre	FechaEnvío	IdPedido	ImporteVent
Reino Unido	Suyama	Michael	3/01/95	10350	642,0
EE.UU.	Davolio	Nancy	2/01/95	10357	1167,6
EE.UU.	Peacock	Margaret	2/01/95	10360	7390,
EE.UU.	Davolio	Nancy	3/01/95	10361	2046,2
EE.UU.	Peacock	Margaret	4/01/95	10363	447,
EE.UU.	Davolio	Nancy	4/01/95		
EE.UU.	Leverling	L Janet	2/01/95		

Utilizar una consulta de parámetros:

Cree un nuevo formulario que utilice la consulta de parámetros Ventas de empleado por país para mostrar los registros de ventas comprendidos en un intervalo de fechas, como se muestra en la siguiente ilustración. Los nombres de los parámetros para la consulta son Fecha inicial y Fecha de fin.

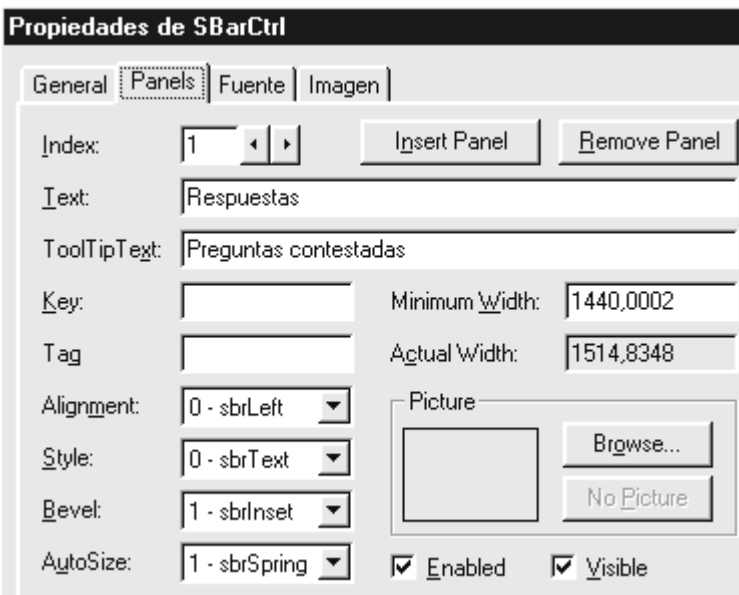
```
Private Sub cmdExecute_Click()
    Dim strSQL As String
    Dim dbCurrent As Database
    Dim recSales As Recordset
    Dim qrySales As QueryDef
    Set dbCurrent = DBEngine.Workspaces(0).OpenDatabase _
        ("C:\Archivos de programa\DevStudio\VB\Nwind.mdb")
    Set qrySales = dbCurrent.QueryDefs("Ventas de empleado por país")
    qrySales.Parameters("Fecha de inicio") = CDate(txtBegin)
    qrySales.Parameters("Fecha de fin") = CDate(txtEnd)
    Set recSales = qrySales.OpenRecordset()
    Set datSales.Recordset = recSales
End Sub
```

## Ejercicio completo. Test Win



Una vez agregados los controles al formulario, estableceremos las propiedades de los controles más importantes:

- La barra de estado: StatusBar1.



Name: StatusBar1  
 Insertaremos los 5 paneles que se muestran en la barra de estado, en la propiedad personalizado:  
 Tool Tip text: es la descripción contextual del mouse, ejemplo:  
 Preguntas contestadas  
 y el texto y estilo de cada panel es el siguiente:  
 Repuestas: estilo Sbrtext  
 Aciertos: estilo Sbrtext  
 Preguntas: estilo Sbrtext  
 Media: estilo Sbrtext  
 Fecha: estilo Sbrdata

- ProgressBar1: Máximo: 100, Mínimo: 0, Alineación: al bottom

- Optionbutton: copiar y pegar creando matriz de cuatro controles cuya caption sea: A:, B, C, y D

**Base de datos:** A continuación crear la base de datos del Test de preguntas en Access (por ejemplo) cuyos campos sean los siguientes:

Nombre del campo	Tipo de datos
Nivel	Númérico
Título	Texto
Texto	Memo
R1	Texto
R2	Texto
R3	Texto
R4	Texto
SOL	Texto
COMEN	Texto
Imagen	Objeto OLE

Guarda la tabla con el nombre TEST1 y la base de datos con el nombre TEST.MDB en la carpeta donde desarrolles el programa.

Rellena al menos unas preguntas de dos o tres niveles para luego probar su funcionamiento.

Una vez creada la tabla y algunos registros, salir de Access Volver a Visual basic para asignar los campos necesarios a los controles:

**Data1:**            databasename:C:\Archivos de programa\

Test\test.mdb - RecordSource: Test1-Visible: False

**Label5:** Caption: Título: Datasource: Data1 – Datafield: Título – Autosize: True.

**Textbox txtFields:** (aquí pondremos el campo memo ) ->: Datasource: Data1 - Multiline: true

**Label Comentario:** Datasource: Data1 - Field:COMEN

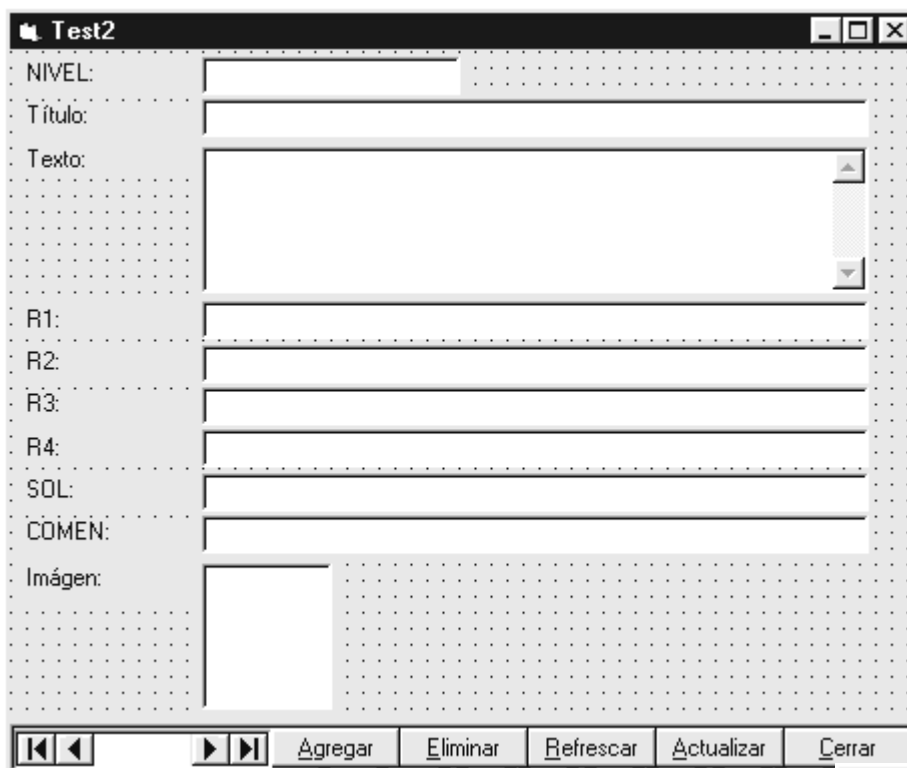
**Ole oleField3:** Datasource: Data1 - Datafield: Imagen – Class: Paint.Picture.

**lblLabels:** Datasource: Data1 - Datafield: R1 – Caption R1 Idem para las cuatro labels de las respuestas.

Antes de continuar para editar y añadir las preguntas crearemos el formulario **TESTFORM** y el de contraseña **CONTRA** como se muestra en el proyecto:

 <b>contra.frm</b>	contra
 <b>test1.frm</b>	Form2
 <b>testform1.FRM</b>	testform

Formulario **TESTFORM** :



Añade el complemento diseñador de formulario de datos y/o elige la opción del menú complementos de Visual basic. esta opción te permite crear el formulario Test Form automáticamente, y quedará aproximadamente como el de la figura:

Para generar el formulario Visual basic te pedirá abrir la base de datos correspondiente:

TEST.MDB tabla TEST1 y añadir todos los campos disponibles de la lista.

Para que no puedan cambiar las preguntas antes crearemos un formulario de contraseña de paso:

Formulario **CONTRA:**

```
Private Sub Text1_Change()
If UCase(Text1.Text) = "OFIMEGA" Then
Command1.Enabled = True
'UCase es para convertir a mayúsculas
'es mejor que command1 default=true para aceptar
con intro y passwordchar: *
End Sub
```



Al diseñar el formulario el botón VALE debe estar deshabilitado (enabled=false) para que al escribir correctamente la contraseña se active y 'es mejor que command1 default=true para aceptar o entrar con Enter.

Aquí se muestra el código de ambos botones:

**Botón Vale:**

```
Private Sub Command1_Click()
testform.Show
Hide ' Esconde el formulario.
End Sub
```

**Botón cancelar:**

```
Private Sub Command2_Click()
Unload Me
End Sub
```

Código del formulario principal:

### Crear una consulta SQL

**Combo1:** Sirve para elegir el nivel de las preguntas

por lo tanto en la propiedad LIST se añadirá: 0 1 2 3 4 5 6 etc..

Este control Combo crea y ejecuta una Consulta de selección (Query) que actua como filtro para cada nivel. esta es la sintaxis de una consulta típica:

**SELECT** (campos de la tabla) **FROM** (nombre de tabla) **WHERE** (condición o filtro)

Este es el ejemplo de una consulta que filtra el nivel 1:

```
Qry = "Select TEST1.NIVEL, TEST1.Titulo,TEST1.Texto,Test1.imagen,
TEST1.R1,TEST1.R2,TEST1.R3,TEST1.R4,TEST1.SOL,TEST1.COMEN From TEST1 WHERE HERE
Test1.Nivel=1"
```

Si el combo1 se elige y no se modifica , a veces no se activa por eso se añade al evento Combo1\_Click():  
Combo1\_Change

Como la consulta es muy larga se divide en varias líneas. Este es el código copeto:

```
Private Sub Combo1_Change()
Dim Qry As String
'Genera la consulta
Qry = "Select TEST1.NIVEL, TEST1.Titulo,TEST1.Texto,Test1.imagen,"
Qry = Qry + "TEST1.R1,TEST1.R2,TEST1.R3,TEST1.R4,TEST1.SOL,TEST1.COMEN"
Qry = Qry + " From TEST1 WHERE (Test1.Nivel=" + Combo1.Text + ")"
Data1.RecordSource = Qry
'Refresca el control Data1
Data1.Refresh
End Sub
```

**Combo2:** No tiene código, sólo elige la opción de la lista: Secuencial o aleatoria, si es secuencial, el botón siguiente salta de pregunta en pregunta y si es aleatoria salta un número del 1 al 6 (como en un dado).

**Menú:** Elegir del menú de Visual Basic: Herramientas – Editor de menús.

&Archivo	Crear la estructura de menús de la imagen.
...&Editar preguntas...	El código para Editar: <b>contra.Show</b> (muestra la ventana contraseña)
...&Imprimir	El código Imprimir: <b>Form2.PrintForm</b> 'imprime la imagen del formulario form2
...&Salir	El código para salir: <b>Unload me</b>
A&yuda	El código para ayuda contenido: Muestra la ayuda. Para crear un archivo de ayuda se crea un archivo del tipo RTF y se compila con el programa CHW.EXE (ver tema ficheros de ayuda). El código para ayuda Acerca de... mostrar una ventana típica de Acerca de...
...&Contenido...	
...&Acerca de...	

(ya se explicó)

```
Private Sub Contenido_Click()
With CommonDialog1 'Abre la ayuda y muestra contenidos
.HelpCommand = cdlHelpContents
.HelpFile = "c:\archivos de programa\test\ayuda.hlp "
.ShowHelp
End With
End Sub
```

A continuación se muestran las tres subrutinas básicas: Al cargar el formulario, al elegir una opción de respuesta y al pulsar el botón siguiente pregunta.

```

Private Sub Form_Load()
Show
siguiente.Enabled = False
Combo1.SetFocus
Imagen.Picture = Happy.Picture
Frame2.Enabled = True
End Sub

Private Sub siguiente_Click()
Dim dado As Integer
If Combo2.Text = "Secuencial" Then
    Data1.Recordset.MoveNext
    Label1.Caption = "Elige una respuesta"
Else
    dado = Int((6 * Rnd) + 1) ' Genera un valor
aleatorio entre 1 y 6.
    Data1.Recordset.Move dado
    Label1.Caption = "Has sacado un " + Str(dado)
End If
StatusBar1.Panels(3) = "Núm pregunta: " +
Str(Data1.Recordset.AbsolutePosition + 1)
siguiente.Enabled = False
Comentario.Visible = False
Frame2.Enabled = True
Option1(0).Value = False
Option1(1).Value = False
Option1(2).Value = False
Option1(3).Value = False
incorrecto.Visible = False
correcto.Visible = False
End Sub
Private Sub Combo1_Change()
Dim Qry As String
'Genera la consulta
Qry = "Select TEST1.NIVEL,
TEST1.Titulo,TEST1.Texto,Test1.imagen,"
Qry = Qry +
"TEST1.R1,TEST1.R2,TEST1.R3,TEST1.R4,TE
ST1.SOL,TEST1.COMEN"
Qry = Qry + " From TEST1 WHERE
(Test1.Nivel=" + Combo1.Text + ")"
Data1.RecordSource = Qry
'Refresha el control Data1
Data1.Refresh
'ProgressBar1.Max =
Int(Data1.Recordset.RecordCount)
'StatusBar1.Panels(5) = "Total:" +
Str(Data1.Recordset.RecordCount)
End Sub

```

```

Private Sub Option1_Click
Dim letra As String ' variable letra tipo texto
Dim comenta, solu As String
Dim numero, nota As Integer
numero = Index 'índice matriz de opción
Select Case numero
    Case 0: letra = "A"
    Case 1: letra = "B"
    Case 2: letra = "C"
    Case 3: letra = "D"
    Case Else
        letra = "Sin respuesta"
End Select
If Data1.Recordset.EOF Then MsgBox ("Fin
de respuestas")
Exit Sub ' no continua
End If
solu = Data1.Recordset.Fields("Sol").Value
preguntas = preguntas + 1
StatusBar1.Panels(1) = "Respuestas:" +
Str(preguntas)
ProgressBar1.Value = preguntas
If letra = solu Then
    aciertos = aciertos + 1
    correcto.Visible = True
    incorrecto.Visible = False
    StatusBar1.Panels(2) = "Aciertos:" +
Str(aciertos)
    Imagen.Picture = Happy.Picture
Else
    Beep
    incorrecto.Caption = "Incorrecto, la
respuesta correcta era la " + solu
    incorrecto.Visible = True
    correcto.Visible = False
    Imagen.Picture = Sad.Picture
End If
Frame2.Enabled = False
nota = aciertos * 10 / preguntas
StatusBar1.Panels(4) = "Media: " + Str(nota)
Comentario.Visible = True
siguiente.Enabled = True
Label1.Caption = "Pulsa siguiente pregunta"
End Sub

```

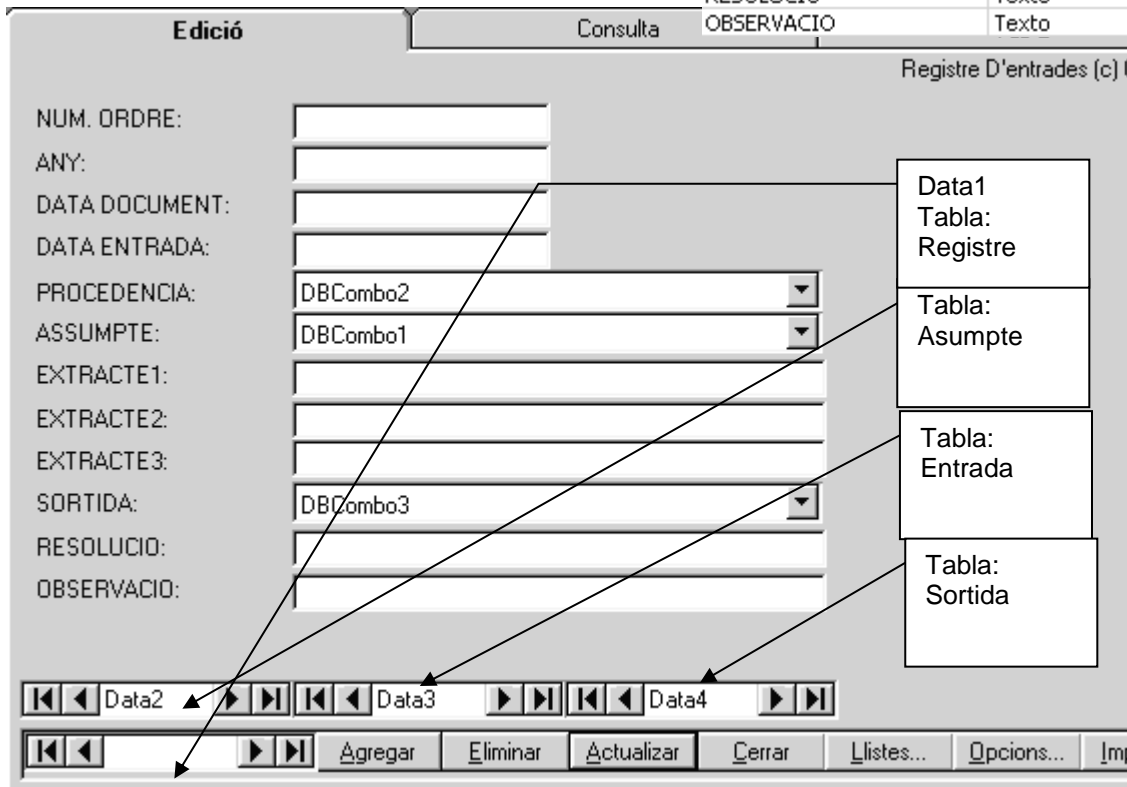
# PROGRAMA DE CONTROL DE REGISTROS

1.- Crear la base de datos Registro.Mdb con las siguientes tablas:

Nombre del campo	Tipo de datos
NUM_ORDRE	Númérico
ANY	Númérico
DATA_DOCUM	Fecha/Hora
DATA_ENTRA	Fecha/Hora
PROCEDENCI	Texto
ASSUMPTE	Texto
EXTRACTE1	Texto
EXTRACTE2	Texto
EXTRACTE3	Texto
SORTIDA	Texto
RESOLUCIO	Texto
OBSERVACIO	Texto

Registro, Asumpte, Procedencia, Sortida y Clau.

La tabla Registre tiene los siguientes campos:



```

Private Sub Actualizar_Click()
    Dim numero
    Data1.Recordset.MoveLast
    numero =
    Data1.Recordset.Fields("Num_ordre").Value
    estado.Caption = "Último registro es el " + Str(numero)
    Data1.UpdateRecord
    Data1.Recordset.MoveLast
End Sub
    
```

```

Private Sub Agregar_Click()
    Dim numero As Integer
    Dim ano As Integer
    Data1.Recordset.MoveLast
    numero = Data1.Recordset.Fields("Num_ordre").Value
    ano = Data1.Recordset.Fields("Any").Value
    Data1.Recordset.AddNew
    Data1.Recordset.Fields("Num_ordre").Value = numero + 1
    Data1.Recordset.Fields("Any").Value = ano
    Data1.Recordset.Fields("DATA_DOCUM").Value = Date
    Data1.Recordset.Fields("DATA_ENTRA").Value = Date
    Data1.UpdateRecord
    Data1.Recordset.MoveLast
    estado.Caption = "Se añadió un registro"
End Sub
    
```

```

Private Sub Cerrar_Click()
    Unload Me
End Sub
    
```

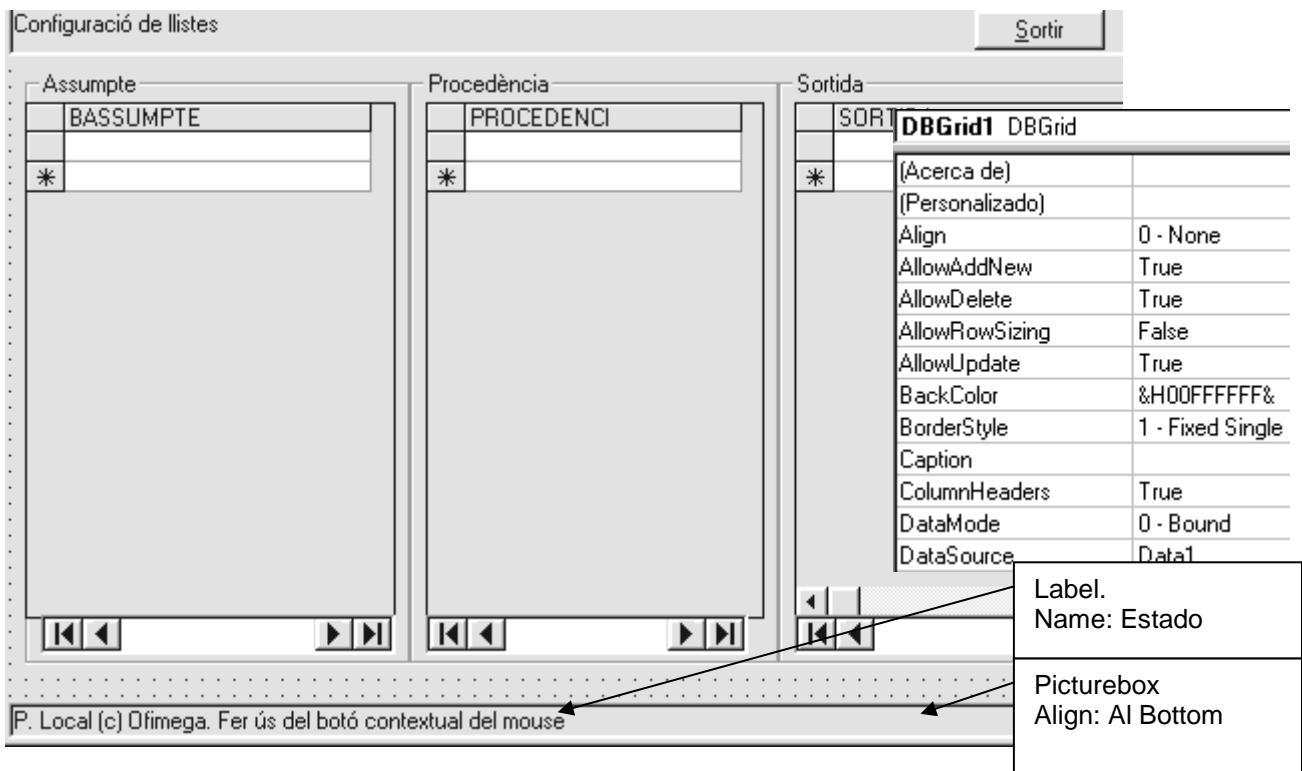
```

numero = Data1.Recordset.Fields("Num_ordre").Value
Msg = " ¿Desea eliminar el registro" + Str(numero) + "?" ' Define el mensaje.
Estilo = vbYesNo + vbCritical + vbDefaultButton2 ' Define los botones.
Titulo = "Eliminar registro" ' Define el título.
Respuesta = MsgBox(Msg, Estilo, Titulo)
If Respuesta = vbYes Then ' El usuario eligió Sí.
    Data1.Recordset.Delete
    Data1.Recordset.MoveNext
    
```

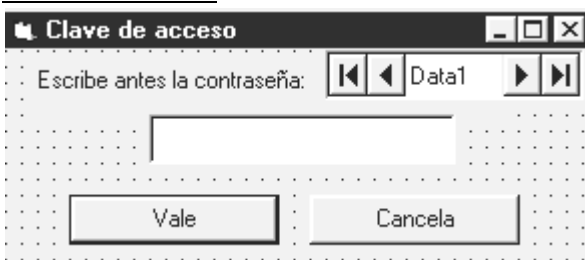
Height	315
HelpContextID	0
Index	
IntegralHeight	True
Left	2160
ListField	PROCEDENCI
Locked	False
MatchEntry	0 - Basic Matching
MouseIcon	(Ninguno)
MousePointer	0 - Default
Name	DBCombo2
RowSource	Data3

```
Private Sub Llistes_Click()
frmconfig.Show
End Sub
```

Formulario FrmConfig: Contiene 3 Frames, 3 Dbgrids, 3 Datas, 2 PictureBox y 1 CommdadButton



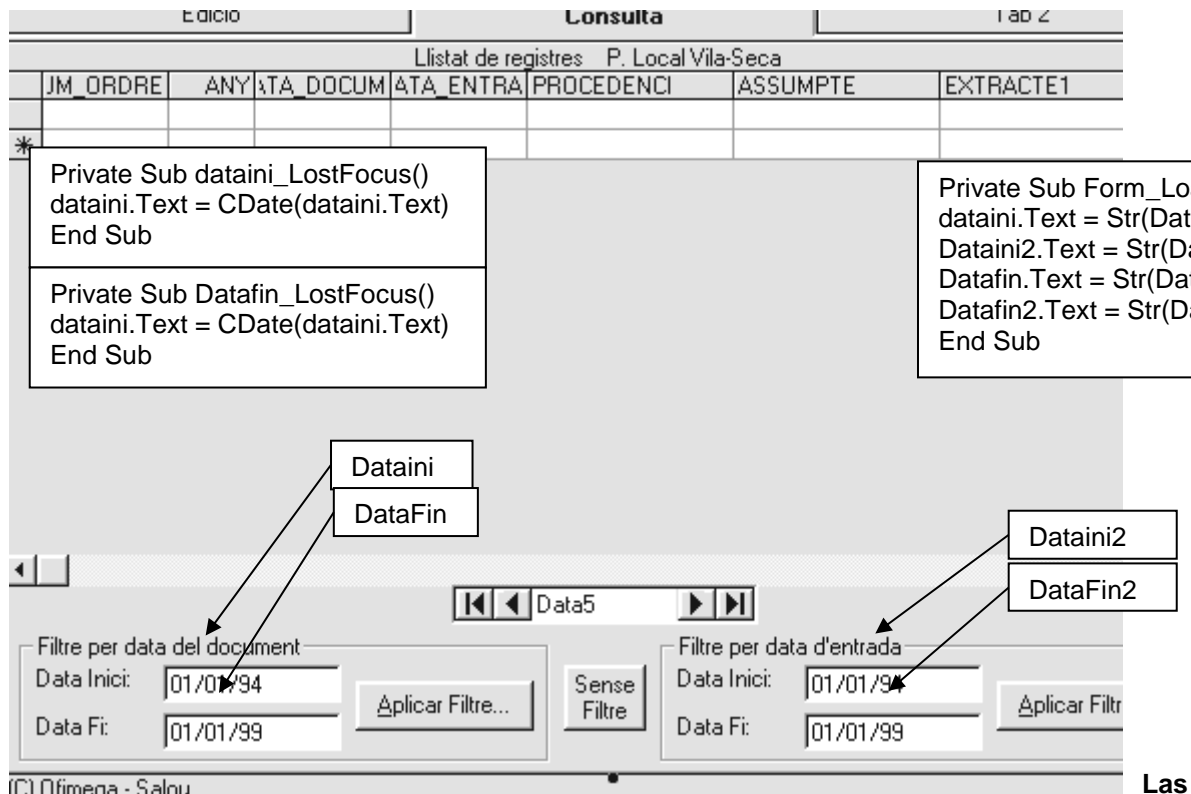
Formulario Contra:



Aparece al inicio del programa. Habilita el botón **Vale** al comparar con un texto Ucase() de la tabla clau que se puede modificar desde el botón de opciones

Formulario Principal.Sstab2: Consultas.

Contiene 1 Dbgrid, 2 Frames, 4 Textbox, 4 Labels, 1 Control data5 y tres botones.



(f) Ofimena - Salou

Las

**consultas SQL.** Código de los botones de filtro 1 y 2:

```
Private Sub Filtre1_Click()
Dim Qry As String
Qry = "SELECT PLOCAL.NUM_ORDRE, PLOCAL.DATA_DOCUM, PLOCAL.DATA_ENTRA,"
Qry = Qry + " PLOCAL.PROCEDENCI, PLOCAL.ASSUMPTE, PLOCAL.EXTRACTE1,"
PLOCAL.EXTRACTE2,"
Qry = Qry + " PLOCAL.EXTRACTE3, PLOCAL.SORTIDA, PLOCAL.RESOLUCIO,
PLOCAL.OBSERVACIO"
Qry = Qry + " From PLOCAL WHERE PLOCAL.DATA_DOCUM>#" + dataini.Text + "# and
PLOCAL.DATA_DOCUM<#" + Datafin.Text + "#"
Data5.RecordSource = Qry
Data5.Refresh
End Sub
```

```
Private Sub Filtre2_Click()
Dim Qry As String
Qry = "SELECT PLOCAL.NUM_ORDRE, PLOCAL.DATA_DOCUM, PLOCAL.DATA_ENTRA,"
Qry = Qry + " PLOCAL.PROCEDENCI, PLOCAL.ASSUMPTE, PLOCAL.EXTRACTE1,"
PLOCAL.EXTRACTE2,"
Qry = Qry + " PLOCAL.EXTRACTE3, PLOCAL.SORTIDA, PLOCAL.RESOLUCIO,
PLOCAL.OBSERVACIO"
Qry = Qry + " From PLOCAL WHERE PLOCAL.DATA_ENTRA>#" + Dataini2.Text + "# and
PLOCAL.DATA_ENTRA<#" + Datafin2.Text + "#"
Data5.RecordSource = Qry
Data5.Refresh
End Sub
```

## Ficheros de ayuda

Cómo crear un fichero de ayuda. Primero se crea el fichero en formato RTF (que permite subrayados y pies de página entre otros efectos) y luego se compila con el programa HCW (Help Compiler Windows) que se incorpora a partir de la versión 5 en adelante (caso de no tenerlo, es exactamente el mismo que el de Borland Delphi)

### 1.- Crear el fichero ayuda.rtf:

Como procesador usaremos Word 97. Ateniéndose a los siguientes efectos de formato:

Subrayado: vínculo del tema de ayuda.

Doble subrayado: ventana flotante

Texto oculto: marcador del tema de ayuda

En nota a pie: # = marcador del tema; K = Palabra clave; \$ = Nombre del tema a buscar

Entrar en Word 97.

Ver en modo normal (no diseño de página)

Insertar nota a pie con la marca personal # (no autonumeración) y escribir como nota: ID\_AYUDA

Insertar otra nota a pie con la marca \$ y escribir en la nota: Contenido del fichero de ayuda

Escribir en el documento normal:

Ayuda de Win Test

Este fichero contiene información para el uso del programa Wintest

En subrayado doble y con un tabulador:

DescripciónID\_DESCRIPCION→Formato fuentes texto oculto

Modificar preguntasID\_MODIFICAR → Formato fuentes texto oculto

Insertar salto de página (Control+Intro) y en la nueva página insertar nota a pie (#) llamada ID\_DESCRIPCION

escribir en el documento normal el texto de la descripción

Insertar salto de página (Control+Intro) y en la nueva página insertar nota a pie (#) llamada ID\_MODIFICAR y escribir en el documento normal el texto de modificar.

El documento debe quedar similar a este:

```
# $ Ayuda de Win Test
Este fichero contiene información para el uso del programa Wintest
  Descripción
  Modificar preguntas
----Salto de página-----
# Descripción: Este es un ejemplo de un archivo de ayuda creado en formato RTF desde Word con notas a
pie de página
----Salto de página-----
# Modificar: Esta es la página de modificar y en su nota a pie de página se encuentra el marcador
ID_MODIFICAR
-----notas a pie-----
# ID_AYUDA
$ Contenido del fichero de ayuda
# ID_DESCRIPCION
# ID_MODIFICAR
```

Guardar el archivo con formato RTF y con el nombre: AYUDA.RTF.

### 2.- Compilar el fichero:

Ejecutar el programa HCW.EXE. Elegir : File – New – Help Project

Añadir el Fichero: File... Add "Ayuda.rtf" y Pulsar el botón Save and compile

**3.- Enlazar la aplicación de Visual basic con el fichero de ayuda:**

Método 1: Llamar a la API de Windows (Public declare function Winhelp Lib "user32" ) y escribir el código:

```
Private Sub mnuHelpContents_Click()  
    Dim R as integer  
    App.Helpfile = "Ayuda.hlp"  
    R= Winhelp(hwnd,(App.HelpFile), &H3,0)  
End Sub
```

Método 2: Insertar un control de cuadro de diálogo cuya propiedad Helpfile sea "Ayuda.hlp" y un menú de ayuda cuyo código sea:

```
Private Sub mnuHelpContents_Click()  
    With CommonDialog1 'Abre la ayuda y muestra la pantalla de contenidos  
        .HelpCommand = cdlHelpContents  
        .HelpFile = "C:\.....\ayuda.hlp "  
        .ShowHelp  
    End With  
End Sub
```

**Ejemplo añadir datos Mediante RecordSet**

```

Private Sub Form_Load()
    ' Establecemos la conexión
    Dim Conexion As New Connection
    With Conexion
        .Provider = "Microsoft.Jet.OLEDB.3.51"
        .ConnectionString = "Data Source=E:\Libros\PBVisualBasic6\Ejemplos\22\Fotografias.mdb"
    .Open
    End With

    ' Creamos un Recordset
    Dim Resultado As New Recordset
    ' abrimos la tabla Objetivos
    Resultado.Open "Objetivos", Conexion, , _
        adLockOptimistic, adCmdTable
    Resultado.AddNew ' añadimos el registro
    ' insertamos los nuevos datos
    Resultado!Nombre = "Zoom240"
    Resultado!Tipo = "Bayoneta"
    Resultado!mm = "240"
    Resultado.Update ' y los escribimos
    ' cerramos la tabla
    Resultado.Close

    Unload Me
End Sub

```

**Ejemplo consulta de datos.**

```

Private Sub Form_Load()
    ' Establecemos la conexión
    Dim Conexion As New Connection
    With Conexion
        .Provider = "Microsoft.Jet.OLEDB.3.51"
        .ConnectionString = "Data Source=E:\Libros\PBVisualBasic6\Ejemplos\22\Fotografias.mdb"
    .Open
    End With
    ' Creamos un Recordset
    Dim Resultado As New Recordset
    ' abrimos la tabla Objetivos
    Resultado.CursorLocation = adUseClient
    Resultado.Open "Fotografias", Conexion, , _
        adLockReadOnly, adCmdTable
    Dim Cadena As String

    Resultado.Find "Tema='Monumentos'" ' Fijamos el filtro hasta que lleguemos al final
    Do While Not Resultado.EOF
        Cadena = Cadena & Resultado!Tema & _
            " -> " & Resultado!titulo & vbCrLf
        Resultado.Find "Tema='Monumentos'", 1 ' buscar siguiente
    Loop
    MsgBox Cadena

    Resultado.Close ' cerramos la tabla

    Unload Me
End Sub

```

**Códigos de tecla de Visual Basic para los eventos Keydown, Keyup y keypress**

Constante	Valor	Descripción
vbKeyLButton	0x1	Botón izquierdo del mouse.
vbKeyRButton	0x2	Botón derecho del mouse.
vbKeyCancel	0x3	Tecla CANCEL.
vbKeyMButton	0x4	Botón medio del mouse.
vbKeyBack	0x8	RETROCESO.
vbKeyTab	0x9	Tecla TAB.
vbKeyClear	0xC	SUPR.
vbKeyReturn	0xD	ENTRAR.
vbKeyShift	0x10	MAYÚS.
vbKeyControl	0x11	CTRL.
vbKeyMenu	0x12	MENU.
vbKeyPause	0x13	PAUSA.
vbKeyCapital	0x14	BLOQ MAYÚS.
vbKeyEscape	0x1B	ESC.
vbKeySpace	0x20	BARRA ESPACIADORA.
vbKeyPrior	0x21	RE PÁG.
vbKeyNext	0x22	AV PÁG.
vbKeyEnd	0x23	FIN.
vbKeyHome	0x24	INICIO.
vbKeyLeft	0x25	FLECHA IZQUIERDA.
vbKeyUp	0x26	FLECHA ARRIBA.
vbKeyRight	0x27	FLECHA DERECHA.
vbKeyDown	0x28	FLECHA ABAJO.
vbKeySelect	0x29	SELECT.
vbKeyPrint	0x2A	IMPRIMIR PANTALLA.
vbKeyExecute	0x2B	EXECUTE.
vbKeySnapshot	0x2C	SNAPSHOT.
vbKeyInsert	0x2D	INS.
vbKeyDelete	0x2E	SUPRIMIR.
vbKeyHelp	0x2F	AYUDA.
vbKeyNumlock	0x90	BLOQUEO NUMÉRICO.

Desde KeyA hasta KeyZ son iguales a sus equivalentes ASCII: 'A' hasta 'Z'

Constante	Valor	Descripción
vbKeyA	65	Tecla A.
vbKeyB	66	Tecla B.
vbKeyC	67	Tecla C.
vbKeyD	68	Tecla D.
vbKeyE	69	Tecla E.
vbKeyF	70	Tecla F.
vbKeyG	71	Tecla G.
vbKeyH	72	Tecla H.
vbKeyI	73	Tecla I.
vbKeyJ	74	Tecla J.
vbKeyK	75	Tecla K.
vbKeyL	76	Tecla L.
vbKeyM	77	Tecla M.
vbKeyN	78	Tecla N.
vbKeyO	79	Tecla O.
vbKeyP	80	Tecla P.
vbKeyQ	81	Tecla Q.
vbKeyR	82	Tecla R.
vbKeyS	83	Tecla S.
vbKeyT	84	Tecla T.
vbKeyU	85	Tecla U.
vbKeyV	86	Tecla V.
vbKeyW	87	Tecla W.
vbKeyX	88	Tecla X.
vbKeyY	89	Tecla Y.
vbKeyZ	90	Tecla Z.

**Teclas del teclado numérico**

Constante	Valor	Descripción
vbKeyNumpad0	0x60	Tecla 0.
vbKeyNumpad1	0x61	Tecla 1.
vbKeyNumpad2	0x62	Tecla 2.
vbKeyNumpad3	0x63	Tecla 3.
vbKeyNumpad4	0x64	Tecla 4.
vbKeyNumpad5	0x65	Tecla 5.
vbKeyNumpad6	0x66	Tecla 6.
vbKeyNumpad7	0x67	Tecla 7.
vbKeyNumpad8	0x68	Tecla 8.
vbKeyNumpad9	0x69	Tecla 9.
vbKeyMultiply	0x6A	MULTIPLICACIÓN (*).
vbKeyAdd	0x6B	MÁS (+).
vbKeySeparator	0x6C	INTRO.
vbKeySubtract	0x6D	MENOS (-).
vbKeyDecimal	0x6E	PUNTO DECIMAL (.).
vbKeyDivide	0x6F	DIVISIÓN (/).

Constante	Valor	Descripción
vbKeyF1	0x70	F1.
vbKeyF2	0x71	F2.
vbKeyF3	0x72	F3.
vbKeyF4	0x73	F4.
vbKeyF5	0x74	F5.
vbKeyF6	0x75	F6.
vbKeyF7	0x76	F7.
vbKeyF8	0x77	F8.
vbKeyF9	0x78	F9.
vbKeyF10	0x79	F10.
vbKeyF11	0x7A	F11.
vbKeyF12	0x7B	F12.
vbKey0	48	Tecla 0.
vbKey1	49	Tecla 1.
vbKey2	50	Tecla 2.
vbKey3	51	Tecla 3.
vbKey4	52	Tecla 4.
vbKey5	53	Tecla 5.
vbKey6	54	Tecla 6.
vbKey7	55	Tecla 7.
vbKey8	56	Tecla 8.
vbKey9	57	Tecla 9.

<b><u>Resumen de instrucciones VB</u></b>		
<b>A</b> Access (Open) Alias (Declare) Any (Declare) AppActivate Append (Open)	Get GoSub...Return GoTo	Rollback
<b>B</b> Base (Option Base) Beep BeginTrans Binary (Option Compare)	<b>I</b> If...Then...Else Input #	<b>RSet</b>
<b>C</b> Call Case (Select Case) ChDir ChDrive Close CommitTrans CompactDatabase Compare (Option Compare) Const	<b>K</b> Kill	<b>S</b>
<b>D</b> Date Declare DefBool (Deftype) DefCur (Deftype) DefDate (Deftype) DefDbl (Deftype) DefInt (Deftype) DefLng (Deftype) DefObj (Deftype) DefSng (Deftype) DefStr (Deftype) DefVar (Deftype) DeleteSetting Dim Do...Loop	<b>L</b> Let Lib (Declare) Line Input # Load Local (On Error) Lock...Unlock Loop (Do...Loop) LSet	Select Case
<b>E</b> Each (For Each...Next) Else (If...Then...Else) Elseif (If...Then...Else) End EndIf (If...Then...Else) Erase Error Exit Explicit (Option Explicit)	<b>M</b> Mid MkDir	SendKeys
<b>F</b> FileCopy For Each...Next For...Next FreeLocks Function	<b>N</b> Name	Set
<b>G</b>	<b>O</b> On Error On GoSub On Goto Open Option Base Option Compare Option Explicit Option Private Output (Open)	SetAttr
	<b>P</b> Preserve (ReDim) Print # Private Property Get Property Let Property Set Public Put	SetDataAccessOption
	<b>R</b> Random (Open) Randomize Read (Open) ReDim RegisterDatabase Rem RepairDatabase Reset Resume Return (GoSub...Return) Rmdir	SetDefaultWorkspace
	<b>T</b> Text (Option Compare) Then (If...Then) Time Tipo de datos definido por el usuario (Type) Type	Static
	<b>U</b> Unload Unlock (Lock...Unlocks) Until (Do...Loop)	Stop
	<b>W</b> While...Wend Width # With Write #	Sub

## BASIC – QBASIC PARA DOS

Visual Basic está basado en instrucciones del lenguaje **Qbasic** derivadas de **Basic** derivado de **Básica**. Qbasic viene incluido en el antiguo sistema operativo DOS.

Atención: Si se va a ejecutar desde Windows 9X no se debe instalar el sistema operativo DOS para que funcione. Para instalar el programa simplemente copiar los archivos Qbasic.exe y Qbasic.hlp del antiguo sistema operativo DOS, preferentemente las versiones: 5.0 6.0 o 6.2.

Para ejecutarlo desde Windows puedes crear un archivo BAT con el EDIT.COM del DOS similar a este:  
E incluso un acceso directo al mismo.  
Al entrar en el programa entrarás en una pantalla de edición azul en la que se observa la referencia básica que se puede desctivar pulsando ESC.

```
Archivo BASIC.BAT:
@ ECHO OFF
CD\
CD DOS
QBASIC
```

Utiliza la ayuda – índice para conocer las palabras clave y su sintaxis (modo general de uso) en el siguiente cuadro se muestra un resumen por tipos:

Palabras clave según tarea: Tarea de programación	Palabras clave incluidas en esta lista de QBasic
Controlar flujo del programa	DO...LOOP, END, EXIT, FOR...NEXT, IF...THEN...ELSE, GOSUB...RETURN, GOTO, ON...GOSUB, ON...GOTO, SELECT CASE, STOP, SYSTEM
Declarar constantes y variables	CONST, DATA, DIM, ERASE, OPTION BASE, READ, REDIM, REM, RESTORE, SWAP, TYPE...END TYPE
Definir y llamar procedimientos	CALL, DECLARE, EXIT, FUNCTION, RUN, SHELL, SHARED, STATIC, SUB
Entrada/salida de dispositivos	CLS, CSRLIN, INKEY\$, INP, INPUT, KEY (asignación), LINE INPUT, LOCATE, LPOS, LPRINT, LPRINT USING, OPEN COM, OUT, POS, PRINT, PRINT USING, SPC, SCREEN (función), TAB, VIEW PRINT, WAIT, WIDTH
Presentar imágenes gráficas	CIRCLE, COLOR, GET (gráficos), LINE, PAINT, PALETTE, PCOPY, PMAP, POINT, PRESET, PSET, PUT (gráficos), SCREEN (instrucción), VIEW, WINDOW
Comandos del sistema de DOS	CHDIR, KILL, MKDIR, NAME, RMDIR
Entrada/salida de archivos	CLOSE, EOF, FILEATTR, FREEFILE, GET (archivos), INPUT, INPUT\$, LINE INPUT, LOC, LOCK, LOF, OPEN, PUT (archivos), SEEK (función), SEEK Instrucción, UNLOCK, WRITE
Manejo de la memoria	CLEAR, FRE, PEEK, POKE
Manipulación de cadenas	ASC, CHR\$, HEX\$, INSTR, LCASE\$, LEFT\$, LEN, LSET, LTRIM\$, MID\$ (función), MID\$ (instrucción), OCT\$, RIGHT\$, RSET, RTRIM\$, SPACE\$, STR\$, STRING\$, UCASE\$, VAL
Realizar cálculos matemáticos	ABS, ASC, ATN, CDBL, CINT, CLNG, COS, CSNG, CVDMBF, CVSMBF, EXP, INT, LOG, RANDOMIZE, RND, SGN, SIN, SQR, TAN, TIME\$ (función)
Crear desvíos para eventos y errores	COM, ERDEV, ERDEV\$, ERL, ERR, ERROR, KEY (desviar eventos), ON COM, ON ERROR, ON KEY, ON PEN, ON PLAY, ON STRIG, ON TIMER, PEN, PLAY (desviar eventos), RESUME, RETURN, STRIG, TIMER (función), TIMER (instruc.)

## 1.- EJERCICIO DE PREGUNTAS Y RESPUESTAS

Escribe las siguientes líneas de código en la pantalla de edición de Qbasic.

Recuerda que el apóstrofe son comentarios y se pueden omitir aunque siempre es bueno ponerlos para documentar tu programa o para recordar un concepto.

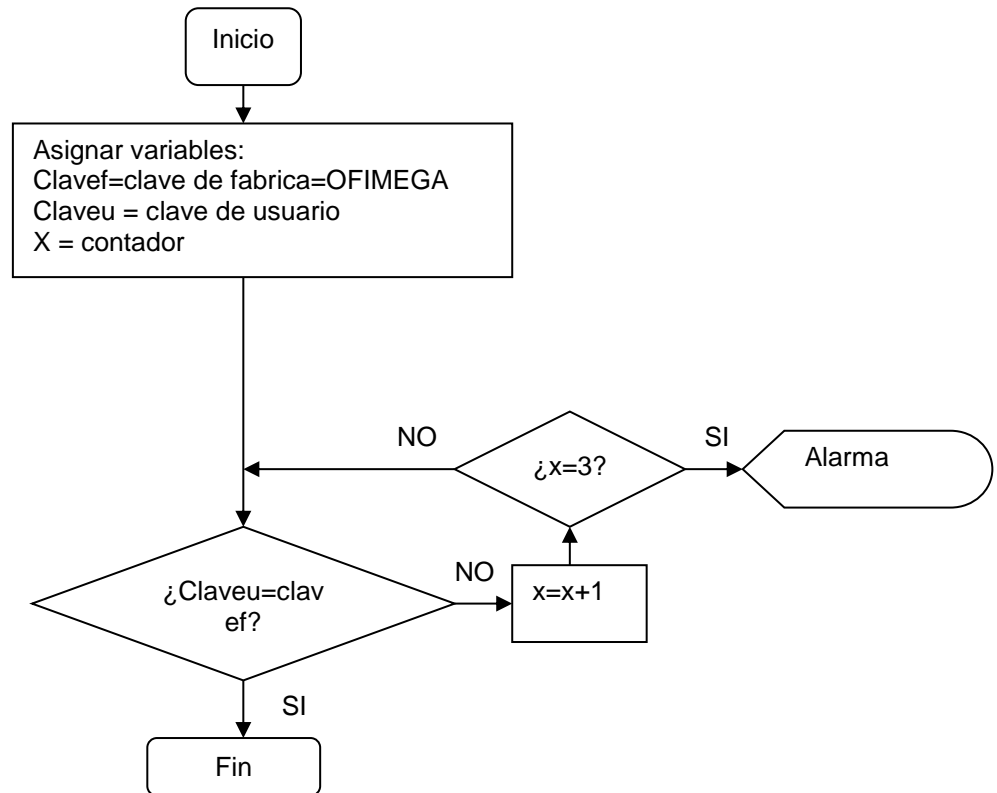
Una vez finalizado. Si este no tiene errores elige del menú Ejecutar- iniciar para comprobar su funcionamiento o pulsa la combinación de teclas Mays-+F5. Correrás el programa (run).

ejercicio de qbasic adivina un numero	
CLS	'borra pantalla
SCREEN 1	'modo pantalla color
COLOR 2, 4	
PRINT "hola"	
INPUT "como te llamas ", x\$	'pide variable x\$ tipo carácter
PRINT "encantado "; x\$	
GOSUB musica	'manda a subrutina musica
INPUT "quieres jugar conmigo ", y\$	'pide variable y\$ tipo caracter
IF y\$ = "no" OR y\$ = "NO" THEN GOTO final	'si y\$=no entonces manda a final
Pregunta:	'etiqueta pregunta
INPUT "cuantos años tengo ", a#	'pide variable tipo numerica
IF a# = 20 GOTO acierto	'si es 20 manda a acierto
IF a# > 20 THEN PRINT "debe ser menor"	
IF a# < 20 THEN PRINT "debe ser mayor"	
GOSUB aviso	'suena tono y vuelve
GOTO pregunta	
Acierto:	'etiqueta acierto
PRINT "has acertado"	
GOSUB musica	
final:	'etiqueta final
PRINT "adios"	
END	'termina el programa
musica:	'eyiqueta musica
PLAY "cdefgab"	'suenan tonos c=do d=re e=mi etc...
RETURN	'vuelve
aviso:	'etiqueta aviso
BEEP	
RETURN	'vuelve

- Presta atención a las etiquetas que es a donde se desvía la secuencia del programa.
- El envío a una subrutina o procedimiento (GOSUB etiqueta) hace que después vuelva con el RETURN a la siguiente línea desde donde se llamó.
- Los envíos GO TO se pueden evitar y sustituir por GOSUB's o por bucles Do While

**2.- EJERCICIO CLAVE DE ACCESO.**

Este ejercicio está representado en el flujograma o diagrama de flujos siguiente:



```

' ejercicio de CLAVE DE ACCESO
inicio:
  COLOR 2, 4          'color texto= 2   color fondo=4
  CLS                 'limpia pantalla
  x% = 1              'variable x% será el contador
  clavef$ = "OFIMEGA" 'Clave de fábrica correcta
  DO WHILE x% < 4    'Hazlo mientras contador sea menor a 4
    claveu$ = "*****" 'Crea la variable clave de usuario
    INPUT "Introduzca la clave:", claveu$ ' Pide clave de usuario
    IF claveu$ = clavef$ THEN 'si es correcta...
      GOTO fin 'manda a fin
    ELSE 'si no...
      x% = x% + 1 'incrementa el contador
      GOSUB fallo 'mana a subrutina fallo
    END IF 'Fin del If en varias instrucciones
  LOOP 'Vuelta del Do While

alarma: 'Etiqueta alarma
  FOR y = 1 TO 10 'Contador de uno a 10 (bucle finito)
    PLAY "c" 'tono do
    PRINT "ALARMA" 'muestra alarma
    PLAY "b"
    PRINT "ALARMA"
  NEXT 'repite el siguiente
GOTO inicio 'vuelve al principio

fin: 'etiqueta fin
  PLAY "ceg" 'tono de correcto
  PRINT "CORRECTO"
  END

fallo: 'etiqueta fallo
  BEEP 'tono incorrecto

```