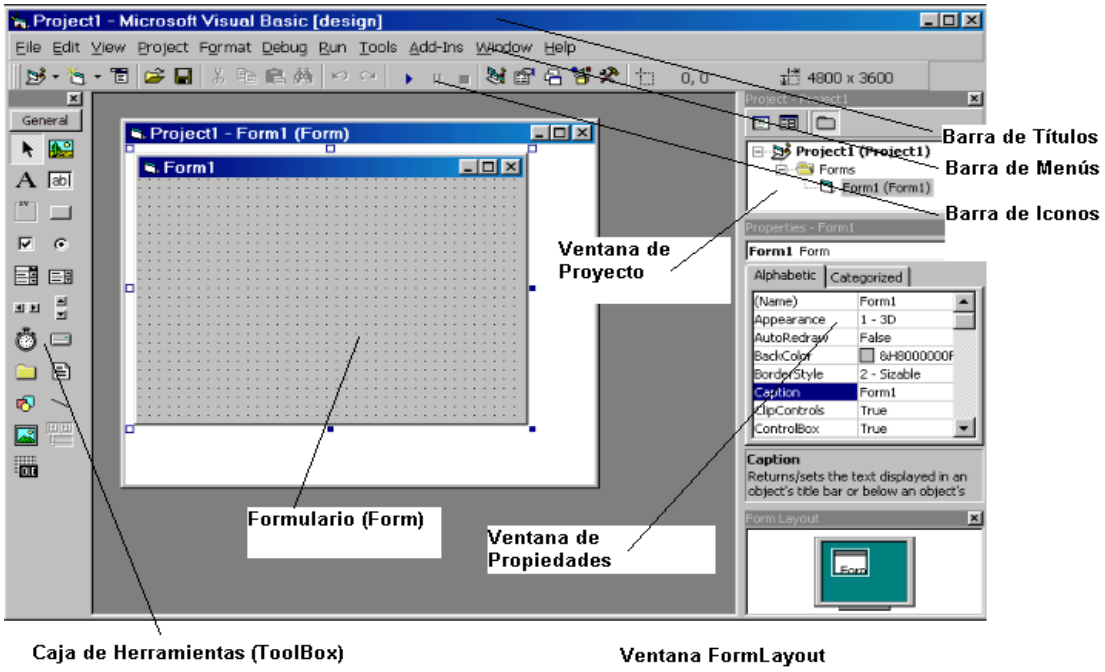
Visual Basic 6.0

I.- Introducción

a) Ejecuta el “Visual Basic”, es decir:
Clic en [Inicio]
Cursor en “Programas”
Cursor en “Visual Basic 6.0”
Clic en “Visual Basic 6.0”

Selecciona el icono o opción: “**Standard EXE**, y clic en [Open]

b) Observa los diferentes elementos de la pantalla:

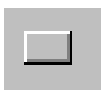


Todo este conjunto de elementos determinan lo que se llama un “Entorno integrado de desarrollo” o IDE (Integrated Development Environment), en nuestro caso se trata del **IDE del Visual Basic**.

c) Vamos a hacer nuestro primer programa (deberíamos hablar de **proyecto** en lugar de “programa”), para ello:

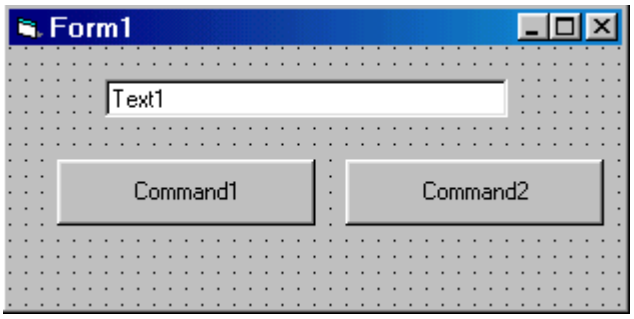
1.- Hemos de **colocar los “controles”** en el formulario:
- Clic en el icono **TextBox** del cuadro de controles:

- “Marca” un pequeño recuadro en el “Form1”
- Clic en el icono **CommandButton** del cuadro de controles:



- “Marca” un pequeño recuadro en el “Form1”
- Inserta otro botón en el formulario, es decir:
  * Clic en el icono **CommandButton**
  * Marca un pequeño recuadro en el Form1

Mueve y cambia el tamaño de los controles de forma que te quede aproximadamente de la siguiente manera:



2.-Vamos a **establecer las propiedades** de los controles (el formulario se considera un control más)

- Selecciona el "form" (clic en el formulario o selecciona "Form1 Form" en la lista desplegable de la "Ventana Propiedades")

- En la propiedad **"Caption"**, borra el texto que aparece por defecto (Form1) y escribe: **Mi primer programa**

- En la propiedad **"Name"**, escribe: frmProg01

- Selecciona el cuadro de texto **Text1** y establece las siguientes propiedades:
  Name = txtSaludo
  Text = borra el texto que aparece por defecto

- Selecciona el primer botón: **Command1** y
  Caption = &Saludo
  Name = cmdSaludo

- Selecciona el segundo botón: **Command2** y
  Caption = &Borrar
  Name = cmdBorrar

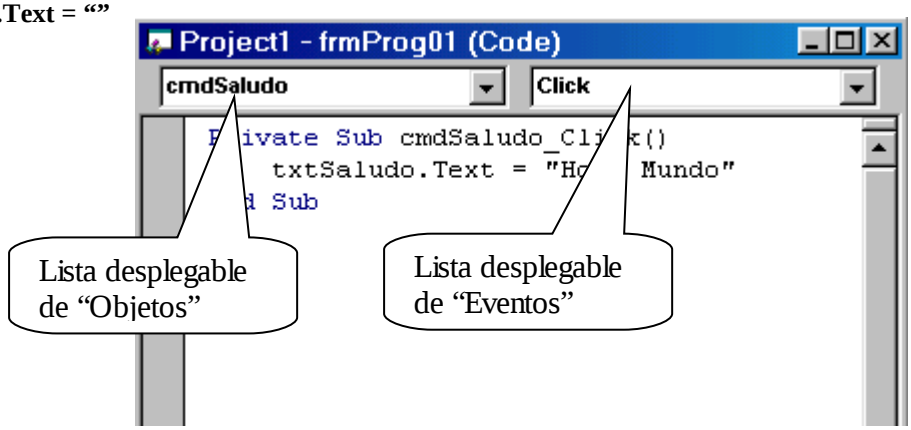
3.- Vamos a escribir el **código ...**

- Haz un doble clic en [Saludo] y escribe:

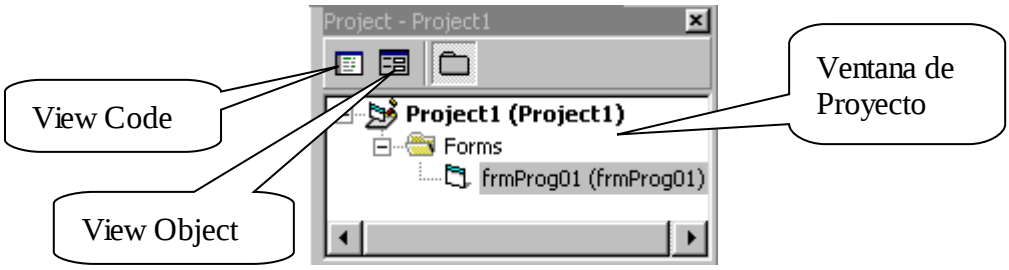
```
Private Sub cmdSaludo_Click()
        txtSaludo.Text = "Hola Mundo"
End Sub
```

- En la lista desplegable de objetos (la de la izquierda), selecciona **cmdBorrar** y escribe:

```
Private Sub cmdBorrar_Click()
txtSaludo.Text = ""
End Sub
```



- "Cierra" la ventana **Project1 – frmProg01 (Code)**, para volver al formulario, o haz clic en el icono **"View Object"** de la **"Ventana de Proyecto"**:



d) Vamos a grabar nuestro proyecto ...

- Menú File
  Save frmProg01 As …
  * En "Save in" sitúate en *TuCarpeta*
  * En "File name" escribe: **Prog01.frm**
  [Save]
- Menú File
  Save Project As …
  * Sitúate en *TuCarpeta*
  * Como nombre del fichero, escribe: **Prog01.vbp**

e) Vamos a ejecutar nuestro proyecto, para comprobar que funciona y cómo funciona:

- Clic en el icono **"Start"**: 

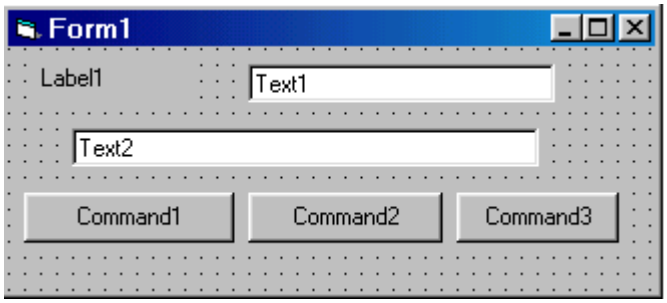
- En estos momentos decimos que estamos en **"tiempo de ejecución"** (antes estábamos en "tiempo de diseño")
  * Prueba el funcionamiento del programa.
  * Observa lo que sucede si pulsas [Alt][S] o [Alt][B]

Cuando te hayas cansado de jugar, "cierra" el formulario "Mi primer programa" (clic en la "X" del extremo superior derecho de la ventana "Mi primer programa"), de esta forma volvemos a **"tiempo de diseño"**.

f) Vamos a hacer otro programa:
  Menú File
    New Project
      Selecciona **"Standard EXE"**
      [Ok]

Paso 1: Insertar controles
- Inserta:

La etiqueta **Label1**corresponde al icono **"Label",** del cuadro de controles:



Paso 2: Establecer propiedades

- Form1
  Caption = Nuestro segundo programa
  Name = frmProg02

- Label1
  Caption = Escribe tu nombre
  Name = lblNombre

- Text1
  Name = txtNombre
  Text =

- Text2
  Name = txtRespuesta
  Text =

- Command1
  Name = cmdSaludo
  Caption = &Saludo

- Command2
  Name = cmdBorrar
  Caption = &Borrar

- Command3
  Name = cmdSalir
  Caption = S&alir

Paso 3: Escribir el "Código"

Accede a la ventana de código (clic en el icono "View Code" de la ventana de proyecto) y escribe los siguientes **procedimientos de evento**:

```
Private Sub cmdSaludo_Click()
    txtRespuesta.Text = "Hola " + txtNombre.Text
End Sub

Private Sub cmdBorrar_Click()
    txtNombre.Text = ""
    txtRespuesta.Text = ""
    txtNombre.SetFocus
End Sub

Private Sub cmdSalir_Click()
    End
End Sub
```

Paso 4: Grabar el programa

- Menú File
  Save frmProg02 As ...
  Prog02.frm (en *TuCarpeta*)

- Menú File
  Save Project As ...
  Prog02.vbp (en *TuCarpeta*)

g) Ejecuta el programa para comprobar que funciona.
Prueba las opciones: [Alt][a], [Alt][B] y [Alt][S]

Para saber más I

El Visual Basic
Visual Basic es una excelente herramienta de programación que permite crear aplicaciones propias para Windows. Este programa permite crear ventanas, botones, menús, etc de una forma fácil e intuitiva. El lenguaje de programación (el lenguaje que hemos de utilizar en el "código"), es el mítico BASIC.

"Visual Basic" es un lenguaje de programación visual, también llamado lenguaje de cuarta generación. Esto quiere decir, que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla.

"Visual Basic" es un **programa basado en objetos**, aunque **no orientado a objetos** como C++ o Java. La diferencia está en que VB utiliza objetos con sus propiedades y métodos, pero carece de los mecanismos de herencia y polimorfismo propios de los verdaderos lenguajes orientados a objetos como Java y C++

Formularios y Controles
Formulario o ficha o form en VB es una ventana. Es también un control, llamado "contenedor" porque contiene normalmente otros controles.
Control es cada uno de los elementos gráficos de una aplicación típica de windows: ventanas (forms), cajas de diálogo, botones, barras de desplazamiento, etc.

Objetos y Propiedades
Los formularios y los distintos tipos de controles son entidades genéricas de las que puede haber varios ejemplares concretos en cada programa.
En "programación orientada a objetos" (en VB deberíamos decir "basada en objetos"), se llama **clase** a estas entidades genéricas, mientras que se llama **objeto** a cada ejemplar de una clase determinada.

- El "TextBox" es una clase
Los TextBox1, TextBox2, TextBox3 de mi programa, son objetos de la clase "TextBox"

- Cada objeto consta de un conjunto de **propiedades**: casi todas pueden establecerse en "tiempo de diseño" y casi todas en "tiempo de ejecución"

- En tiempo de diseño hemos establecido la propiedad **Name** del **Form**, simplemente escribiendo en la ventana de propiedades
- En tiempo de ejecución, hemos establecido la propiedad **Text** de un **TextBox**, con una línea de código del tipo:

```
txtRespuesta.Text = "Hola " + txtNombre.Text
```

Observa de qué forma accedemos a una propiedad en "tiempo de ejecución":

NombreDelObjeto.NombreDeLaPropiedad

Atención con el llamado "operador punto"

- Cada objeto además de propiedades, tiene métodos (funciones). Los métodos no son más que "posibilidades" a acciones que pueden realizar los objetos correspondientes.

Por ejemplo: **txtNombre.SetFocus**
"SetFocus" es un método del TextBox (y de otros controles), que hace: "sitúa el cursor en el objeto

Observa que la "notación" correspondiente a un "método" es la misma que para una propiedad.

En definitiva: el estudio del Visual Basic significa, entre otras cosas, el estudio de las propiedades y métodos de los diferentes objetos (controles).

Nombres de Objetos (notación húngara)
Cada objeto tiene un **"name"**.
Por defecto, dicho nombre hace referencia al tipo de control, por ejemplo el primer formulario de una aplicación tiene por defecto el nombre **Form1**, el segundo **Form2**, etc.

Los nombres que aparecen por defecto no son los más adecuados. Hay una convención ampliamente aceptada (para los programadores de Microsoft es obligatoria), que es la siguiente:
"Se utilizan 3 letras minúsculas que indican el tipo de control, seguidas de otras letras (la primera mayúscula), que hacen referencia al uso que se va a dar a dicho control".
Así:

cmdBorrar:	CommandButton que sirve para "borrar"
txtSaludo:	TextBox que sirve para "saludar"

De momento conocemos los prefijos de:

CommandButton	**cmd**
Form	**frm**
Label	**lbl**
TextBox	**txt**

Ya irán saliendo los que aparecen a continuación:

CheckBox	**chk**	ComboBox	**cbo**
DriveListBox	**drv**	DirListBox	**dir**
HorizontalScrollBar	**hsb**	FileListBox	**fil**
ListBox	**lst**	Frame	**fra**
OptionButton	**opt**	Image	**img**
Shape	**shp**	Line	**lin**
Timer	**tmr**	Menu	**mnu**
PictureBox	**pct**		

Eventos
Las acciones del usuario o del sistema sobre nuestro programa se llaman **EVENTOS**.
Cada vez que se produce un evento sobre un determinado control, Visual Basic ejecuta un **procedimiento o función** que realiza la acción o acciones programadas por el programador para este evento concreto.

Se denominan **procedimientos de evento** y su nombre por ejemplo: **cmdAceptar_ Click,** corresponde a la acción de hacer clic en el botón cmdAceptar

Proyectos y otros ficheros
Un programa en Visual Basic es un **proyecto** que consta de diferentes elementos, básicamente:

Proyecto (.vbp) * **Formularios** con su código (se dice que el código está escrito en el **módulo** del formulario): ficheros con extensión **frm**.

* **Módulos estándar** (contienen funciones y procedimientos generales): ficheros con extensión **bas**.

Para crear un fichero ejecutable (extensión EXE) deberemos utilizar la opción:
Menú File
Make

II.- El Lenguaje BASIC

a) Crea un nuevo proyecto:
Menú File
New Project
Standard EXE

b) Accede al **módulo del formulario,** es decir:
clic en el icono **"View Code"** de la **"Ventana de Proyecto"**

- Selecciona en la lista desplegable de objetos: **Form** y en la lista de eventos: **Activate**
Aparecerá el "esqueleto" del **procedimiento de evento**:Form_Activate

- Escribe:

```
Private Sub Form_Activate()
    Print "Hola tio/tia"
End Sub
```

- Cierra la ventana de código (módulo del formulario) y **ejecuta** el programa.
Supongo que está clara la utilidad de la sentencia: **Print "mensaje"**

c) Vamos a crear un **módulo estándar** (será un fichero con extensión BAS), haz lo siguiente:
Menú Project
Add Module
[Open]
En la nueva ventana **Project1 – Module1 (Code)**, que corresponde a un módulo estándar, escribe:

```
Public Sub Proced01()
    Form1.Print "Nuestro primer procedimiento general"
End Sub
```

Cierra la ventana **Project1 – Module1(Code)**

d) Observa la **ventana de proyecto**:
Nuestro proyecto consta de un formulario (Form1) y un módulo estándar (Module1).
En el **Module1** tenemos escrito un **procedimiento** de nombre Proced01.
En el **módulo del Form1** tenemos escrito un **procedimiento de evento** de nombre: Form_Activate.

- Edita el procedimiento de evento **Form_Activate**:
  - Selecciona el **Form1** en la ventana de proyecto.
  - Clic en el icono **"View Code"** de la Ventana de Proyecto

- Borra la línea: **Print "Hola tio/tia"**, y en su lugar escribe: **Proced01**

- "Cierra" la ventana de código.

e) Ejecuta el programa.
Si todo funciona correctamente aparecerá el mensaje: "Nuestro primer procedimiento general", en el Form1.

En efecto, observa:
- Al ejecutar el programa, se activa el "form"
- Por lo tanto se ejecuta el procedimiento **Form_Activate**
- La única sentencia del procedimiento anterior, es **Proced01**

- Por lo tanto se ejecuta el procedimiento: Proced01, que se encuentra en el módulo estándar **Module1**
- Como la única sentencia del **Proced01** es **Form1.Print "Nuestr ..."**
- Aparece en el form, el mensaje correspondiente.

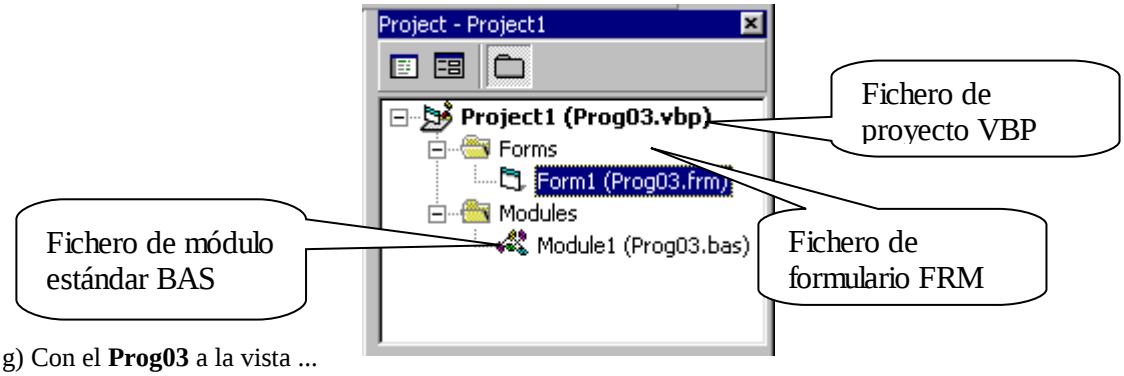
f) Vamos a grabar el proyecto, pero de la siguiente forma:
- Clic en el icono **"Save Project"**:

- Observa que primero nos pregunta el nombre que queremos dar a nuestro módulo estándar (fichero con extensión BAS):
  - Sitúate en *TuCarpeta*
  - File name = **Prog03.bas**

- A continuación nos pregunta el nombre que queremos dar a nuestro formulario (fichero con extensión FRM):
  - Sitúate en *TuCarpeta*
  - File name = **Prog03.frm**

- Por último nos pregunta el nombre que queremos dar a nuestro proyecto (fichero con extensión VBP):
  - Sitúate en *TuCarpeta*
  - File name = **Prog03.vbp**

Observa la **Ventana de Proyecto**:



g) Con el **Prog03** a la vista ...

- Inserta en el formulario un **CommandButton**: Command1, sitúalo en el ángulo inferior derecho del form.

- Accede al **módulo del formulario** y escribe el siguiente procedimiento de evento:

```
Private Sub Command1_Click()
        Print "HOLA MUNDO"
End Sub
```

- Ejecuta el programa y pruébalo (haz clic en el [Command1] varias veces).

Es importante que tengas clara la diferencia entre los programas **Command1_Click** y **Form_Activate** que son **procedimientos de evento**, y se encuentran en el módulo de formulario (fichero con extension **frm**) y el programa **Proced01**, que es un **procedimiento Sub** (procedimiento estándar), y se encuentra en un módulo estándar (fichero con extensión BAS)
Observa de qué forma podemos ejecutar el procedimiento **Proced01**

h) Continuando con el **Prog03** ...

- Escribe en el **Prog03.bas**:

```
Public Sub Proced02()
   Dim nom As String
   nom = InputBox("Cuál es tu nombre?")
   Form1.Print "Hasta luego " + nom
End Sub
```

- Inserta en el "form" otro botón, será el **Command2**, y escribe el siguiente procedimiento de evento:

```
Private Sub Command2_Click()
        Proced02
End Sub
```

- Ejecuta el programa para visualizar lo que hemos conseguido.

- Grabálo todo con el mismo nombre ...

- Clic en el icono **"Save Project"**

- Estudio del Proced2():

Dim nom As String
Declaramos una variable de nombre "nom" y tipo "texto" (string). Ya iremos viendo los diferentes tipos de variable en Visual Basic.

nom = InputBox("Cuál es tu nombre?")
Aparece una ventana, con el mensaje **"Cuál es tu nombre?"**. Cuando contestamos a la pregunta y pulsamos [Return] o clic en [Ok], lo que hemos escrito se "guarda" en la variable de nombre "nom".

Form1.Print "Hasta luego " + nom
Se escribe en el "form" la frase "Hasta luego" y a continuación el valor de la variable "nom".

Observa el operador de **concatenación +**, que sirve para unir el mensaje "Hasta luego" con el valor de "nom".
Quizás seria mejor utilizar el otro **operador de concatenación**, que es el ampersand (&), para no confundir el operador de unión "+" con el correspondiente operador de sumar.

i) Continuando con el **Prog03** ...

- Escribe en el módulo estándar:

```
Public Sub Proced03()
   Dim pobla As String
   Dim tele As String
   pobla = InputBox("Cuál es tu población?")
   tele = InputBox("Y teléfono?")
   Form1.Print "Población= " & pobla
   Form1.Print "Teléfono= " & tele
   Form1.Print
   Form1.Print "Que te vaya bien"
End Sub
```

- Accede al módulo del formulario y cambia el procedimiento de evento **Command2_Click()** de forma que al hacer clic en el [Command2] se ejecute el **Proced03**

- Ejecuta nuestro programa para comprobar que funciona ...

- Ejecútalo varias veces para descubrir la utilidad de la sentencia: **Form1.Print**

- Observa el uso del operador de concatenación **&**

- Graba el programa con el mismo nombre **Prog03**

j) Continuando con el **Prog03** ...

- Sitúate en el módulo estándar y escribe el siguiente procedimiento:

```
Public Sub Proced04()
   Dim n1 As Integer, n2 As Integer
   n1 = InputBox("Escribe un número entero")
   n2 = InputBox("Escribe otro número entero")
   Form1.Print "La suma de " & n1 & " y " & _
      n2 & " es = " & n1 + n2
End Sub
```

- Accede al módulo del formulario y cambia el procedimiento de evento **Command1_Click()**, de forma que al hacer clic en el [Command1] se ejecute el **Proced04**

- Ejecuta nuestro programa para comprobar que nuestro **"Proced04"** funciona correctamente.

- **Estudio del Proced04**

Dim n1 As Integer, n2 As Integer
Declaramos dos variables enteras (integer = entero), es decir n1 y n2 contendran números enteros.

Una sentencia BASIC se puede distribuir entre varias líneas, siempre y cuando al final de la línea escribamos el símbolo de subrayado precedido de un espacio en blanco.

Observa como no hay ningún problema en concatenarlas diferentes partes de una frase con variables y operaciones entre variables incluidas.

- Graba el programa con el mismo nombre **Prog03**

k) Sitúate en el módulo estándar y escribe el siguiente procedimiento:

```
Public Sub Proced05()
   Dim bas As Double
   Dim alt As Double
   Dim are As Double
   bas = InputBox("", "Base del Triángulo")
   alt = InputBox("", "Altura del triángulo")
   are = bas * alt / 2
   MsgBox ("El área del triángulo es = " & are)
   Form1.Print "Base = " & bas
   Form1.Print "Altura = " & alt
   Form1.Print "Área Triángulo= " & are
End Sub
```

- Accede al módulo del formulario y cambia el procedimiento de evento **Command2_Click()**, de forma que al hacer clic en el [Command2] se ejecute el **Proced05**

- Ejecuta nuestro programa para comprobar que nuestro **"Proced05"** funciona correctamente (utiliza números decimales).

Estudio del Proced05:

- **Double** es otro tipo de datos de VB que indica números decimales.

- **InputBox(" mensaje1" , "mensaje2")**
El "mensaje2" será el título de la ventana correspondiente al InputBox. Y "mensaje1" es el texto que aparece en el interior de la ventana.

- **MsgBox("mensaje")**
Es una ventana que nos muestra el "mensaje" y el programa queda inmovilizado hasta que pulsamos [Return] o clic en el [Ok] de dicha ventana.
Observa pues, que **MsgBox** es una alternativa al **Print**

- Graba el programa con el mismo nombre **Prog03**

l) Crea un nuevo proyecto **Standard EXE**

- Inserta un **CommandButton** en el ángulo inferior derechodel form, de propiedades:
Name = cmdMsgBox
Caption = MsgBox

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdMsgBox_Click()
  Dim nom As String
  Dim Respuesta As Integer
  nom = "Pepito"
  MsgBox ("Hola " & nom)
  MsgBox "Hola " & nom
  MsgBox "Mira el título", , "Pongo el título que quiero"
  MsgBox "Observa este" & vbCrLf & "texto que ocupa" & _
    vbCrLf & "tres líneas", , "Titulo"
  MsgBox "Mira el icono de" & vbCrLf & "pregunta", _
    vbQuestion, "Icono Interrogacion"
  MsgBox "Otro icono", vbCritical, "Icono Critico"
  MsgBox "otro", vbExclamation, "Icono Exclamación"
  MsgBox "otro mas", vbInformation, "Icono Información"
  Respuesta = MsgBox("Observa que al incluir más" & _
    vbCrLf & "de un botón, en el MsgBox" & _
    vbCrLf & "pongo paréntesis y utilizo" & vbCrLf _
    & "una variable, que recogerá" & _
    vbCrLf & "el botón que hemos pulsado", vbYesNo + _
    vbQuestion, "Dos botones")
  Form1.Print "Dos botones = " & Respuesta
  Respuesta = MsgBox("tres botones", vbYesNoCancel + _
    vbInformation, "Con icono información")
  Form1.Print "tres botones " & Respuesta
```

```
    Respuesta = MsgBox("tres botones pero" & vbCrLf & _
       "el activo es el segundo", vbAbortRetryIgnore _
       + vbCritical + vbDefaultButton2, "Icono crítico")
    Form1.Print "Tres botones pero ..." & Respuesta
End Sub
```

- Ejecuta el programa para comprobar que funciona, convendria ejecutarlo varias veces observando detenidamente lo que va sucediendo.

- Graba el programa con el nombre **Prog04.frm** (el formulario) y **Prog04.vbp** (el proyecto).

Estudio del cmdMsgBox_Click()

- El primer "MsgBox":
  MsgBox("Hola" & nom)
  es el tipo de cuadro que ya habíamos utilizado en el apartado anterior.

- Si observamos el segundo:
  MsgBox "Hola" & nom
  llegamos a la conclusión que tanto da poner o no poner paréntesis.

- Recuerda que en Visual Basic podemos escribir líneas de programa distribuyéndolas en varias líneas, sin más que escribir el **símbolo de subrayado** (tecla del "menos") **precedido de un espacio en blanco**.

- **vbCrLf** es una constante simbólica de VB que "obliga a un retorno de carro o nueva línea", con su uso conseguimos distribuir el texto en varias líneas.

- El primer argumento del **MsgBox** es el texto que aparece en el cuadro. El tercer argumento es el texto que aparece como título del cuadro (igual que sucedía con el InputBox)

- En el segundo argumento del "msgbox" podemos incluir un icono determinado y/o varios botones y/o activar por defecto un botón determinado. Todo esto se consigue utilizando constantes simbólicas de VB o su valor numérico equivalente como aparece en las siguientes tablas:

Constantes para los iconos	Valor Numérico	Significado
vbCritical	16	Icono crítico
vbQuestion	32	Icono pregunta
vbExclamation	48	Icono exclamación
vbInformation	64	Icono información

Constantes para los botones	Valor Numérico	Significado
vbOKOnly (defecto)	0	[Aceptar]
vbOKCancel	1	[Aceptar][Cancelar]
vbAbortRetryIgnore	2	[Anular][Reintentar][Ignorar]
vbYesNoCancel	3	[Sí][No][Cancelar]
vbYesNo	4	[Sí][No]

Constantes para activar botón	Valor Numérico	Significado
vbDefaultButton1 (defecto)	0	Activa el primer botón
vbDefaultButton2	256	Activa el segundo botón
vbDefaultButton3	512	Activa el tercer botón

- El hecho de incluir botones no tiene sentido si no recogemos el botón pulsado en una variable (de aquí el uso de la variable **respuesta** en nuestro procedimiento). En este caso hemos de escribir el **MsgBox** con paréntesis necesariamente.
  Los números o constante simbólica que devuelven los diferentes botones son los siguientes:

Botón	**Devuelve el número**	**Constante**
Aceptar	1	vbOK
Cancelar	2	vbCancel
Anular	3	vbAbort
Reintentar	4	vbRetry
Ignorar	5	vbIgnore
Sí	6	vbYes
No	7	vbNo

-Graba el programa con el mismo nombre **Prog04**

m) Inserta otro **CommandButton** de propiedades:
Name = cmdInputBox
Caption = InputBox

- Accede al módulo del formulario y escribe el siguiente procedimiento de evento:

```
Private Sub cmdInputBox_Click()
   Dim Respuesta As String
   Respuesta = InputBox("Primera línea" & vbCrLf _
      & "segunda línea", "título del inputbox")
   Respuesta = InputBox("haz clic en [Cancel]", _
      "A ver que pasa si cancelo")
   Form1.Print "Al pulsar cancelar resulta = " & Respuesta
   Respuesta = InputBox("Aparece un valor por defecto", _
      "título", "esto aparece por defecto")
   Respuesta = InputBox("situo la ventana", _
      "1200 twips a la derecha y 1400 hacia abajo", _
      "coordenadas 1200x1400", 1200, 1400)
   Respuesta = InputBox("otra posición", , , 50, 75)
End Sub
```

- Ejecuta el programa para comprobar que funciona, convendría ejecutarlo varias veces observando detenidamente lo que va sucediendo.

Estudio del cmdInputBox_Click()

Observa la sintaxis completa de la función “inputbox” :

variable = InputBox(mensaje1, mensaje2, mensaje3, num1, num2)

mensaje1 = el texto que aparece en el interior del cuadro
mensaje2 = el texto que aparece como título del cuadro.
mensaje3 = el texto que aparece por defecto, escrito.
num1 = coordenada horizontal en twips del extremo superior izquierdo del cuadro.
num2 = coordenada vertical en twips del extremo superior izquierdo del cuadro.

Si ante un cuadro InputBox, hacemos click en el botón [Cancelar], el valor de la "variable" es nula.

1 cm = 566 twips
1 pixel = 15 twips

-Graba el programa con el mismo nombre **Prog04**

n) Inserta otro CommandButton, encima del anterior, de propiedades:
Name = cmdRectangulo
Caption = Rectángulo

- Accede al módulo del formulario y escribe el siguiente procedimiento de evento:

```
Private Sub cmdRectangulo_Click()
   Dim a As Double, b As Double
   Dim s As Double, salida As String
   b = InputBox("Base del Rectángulo")
   a = InputBox("Altura del rectángulo")
   salida = ""
   salida = salida + "ÁREA DE UN RECTÁNGULO" + vbCrLf
   salida = salida + "Base del Rectángulo = " + Str(b) + vbCrLf
   salida = salida + "Altura del Rectángulo =" + Str(a) + vbCrLf
   salida = salida + vbCrLf
   s = a * b
   salida = salida + "Área del Rectángulo = " + Str(s)
   Form1.Print salida
   MsgBox salida
End Sub
```

- Ejecuta el programa, observando detenidamente lo que sucede.

Estudio del cmdRectangulo_Click()

- Observa de qué forma **acumulamos** muchos datos en un solo **Print** o un solo **MsgBox** (ésta será la forma de proceder, cuando necesitemos una "salida" con mucha información).
- Declaramos una variable tipo texto de nombre "salida", que inicializamos a nada: **salida = ""**
- Acumulamos a la variable **salida** todo lo que queremos: **salida = salida + lo que sea**
  "lo que sea" ha de ser una cadena, por esta razón hemos de utilizar la función incorporada a VB: **Str**
  Str(número) = cadena de texto
  Si en lugar de utilizar el operador "+", utilizamos el "&", no es necesario utilizar la función de conversión **Str**, ya que en éste caso el operador (&), "une" los diferentes elementos, sin importar el "tipo".
- Para acabar un solo **Form1.Print** o un **MsgBox**.

o) Inserta otro CommandButton, encima del anterior, de propiedades:
Name = cmdContadores
Caption = Contadores

- Accede al módulo del formulario y escribe el siguiente procedimiento de evento:

```
Private Sub cmdContadores_Click()
   Dim x As Integer
   Dim s As String
   s = ""
```

```
    x = InputBox("Escribe un número entero")
    s = s + "Valor inicial de x" + Chr(9) + Str(x) + Chr(10)
    x = x + 3
    s = s + "Valor de x después de pasar por el contador x=x+3" + Chr(9) + _
            Chr(9) + Str(x) + Chr(10)
    x = x - 2
    s = s + "Después de x=x-2" + Chr(9) + Str(x) + Chr(10)
    x = x * 3
    s = s + "Después de x=x*3" + Chr(9) + Str(x) + Chr(10)
    Form1.Print s
    MsgBox s
End Sub
```

- Ejecuta varias veces el programa, observando detenidamente lo que sucede

Estudio del cmdContadores_Click()

- Chr(9) es equivalente a pulsat la tecla de tabulación
  Chr(código numérico) = caracter Ascii que corresponde al número introducido.
- Chr(10) es una alternativa a **vbCrLf**
- El "contador" es un instrumento muy utilizado en programación.
  Por ejemplo:
  x = x + 3 significa "El nuevo valor de x es el anterior valor de x + 3 unidades"
  En definitiva: la sentencia x=x+3, incrementa en 3 unidades el valor de x

- Graba el programa con el mismo nombre:
  Prog04.frm
  Prog04.vbp

p) Crea un nuevo proyecto (Menú File – New Project – Estándar EXE)

- Inserta un CommandButton en el ángulo inferior derecho del form, de propiedades:
  Name = cmdIf
  Caption = If

- Accede al módulo del formulario y escribe el siguiente procedimiento de evento:

```
Private Sub cmdIf_Click()
    Dim num As Double
    num = InputBox("", "Escribe un número")
    If num < 100 Then
        MsgBox "El número es menor de 100"
    Else
        MsgBox "El número no es menor de 100"
    End If
End Sub
```

- Ejecuta varias veces el programa, observando detenidamente lo que sucede.

La estructura de programación If – Then – Else

```
If condición Then
        Sentencia1
        Sentencia 2
        ...
        ...
Else
        Sentencia3
        Sentencia4
        ...
        ...
End If
```

Si se cumple la "condición" entonces se ejecutarán las sentencias 1, 2, etc. En caso contrario, es decir si no se cumple la condición, se ejecutarán las sentencias 3, 4, etc.
La cláusula "Else" es opcional.

q) Inserta otro CommandButton encima del anterior de propiedades:
Name = cmdIfBis
Caption = IfBis

- Accede al módulo del formulario y escribe el siguiente procedimiento de evento:

```
Private Sub cmdIfBis_Click()
  Dim sexo As String * 1
  sexo = InputBox("Escribe tu sexo (solo la inicial: H o V)")
  If sexo = "H" Or sexo = "h" Then
    MsgBox "Hola chica"
  ElseIf sexo = "V" Or sexo = "v" Then
    MsgBox "Hola chico"
  Else
    MsgBox "Hola sexo ambiguo"
  End If
End Sub
```

- Ejecuta varias veces el programa, observando detenidamente lo que sucede.

Estudio del cmdIfBis_Click()

- String*1
  Significa "String", es decir texto pero de longitud 1, es decir un sólo carácter
- Or
  Evidentemente significa "o".
- Observa la estructura If entera:

```
If condición1 Then
  Sentencia1
  Sentencia2
  …
  …
ElseIf condición2 Then
  Sentencia3
  Sentencia4
  ...
  ...
ElseIf condici'on3 Then
  Sentencia5
  Sentencia6
```

```
    …
    …
  Else
    Sentencia7
    Sentencia8
    …
    …
  End If
```

r) Inserta otro CommandButton encima del anterior, de propiedades:
Name = cmdRepetir
Caption = Repetir

- Accede al módulo del form y escribe el siguiente procedimiento de evento:

```
Private Sub cmdRepetir_Click()
   Dim a As String * 1
   a = InputBox("Quieres continuar (S/N)?")
   If a = "S" Or a = "s" Then
      MsgBox "El programa continua"
   End If
End Sub
```

- Ejecuta varias veces el programa, observando lo que sucede

- Graba el programa como:
Prog05.frm
Prog05.vbp

s) Crea un nuevo proyecto (Menú File – New Project – Estándar EXE)

- Inserta un CommandButton en el ángulo inferior derecho del form, de propiedades:
Name = cmdForTo
Caption = For To

- Accede al módulo del formulario y escribe el siguiente procedimiento de evento:

```
Private Sub cmdForTo_Click()
   Dim num As Integer
   Dim i As Byte
   Dim nom As String, salida As String
   num = InputBox("Cuántas veces quieres que te salude?")
   nom = InputBox("Cuál es tu nombre?")
   For i = 1 To num
      salida = salida & "Hola " & nom & vbCrLf
   Next
   Form1.Print salida
   MsgBox salida
End Sub
```

- Ejecuta varias veces el programa, observando detenidamente lo que sucede.

La estructura de programación: For – To – Next

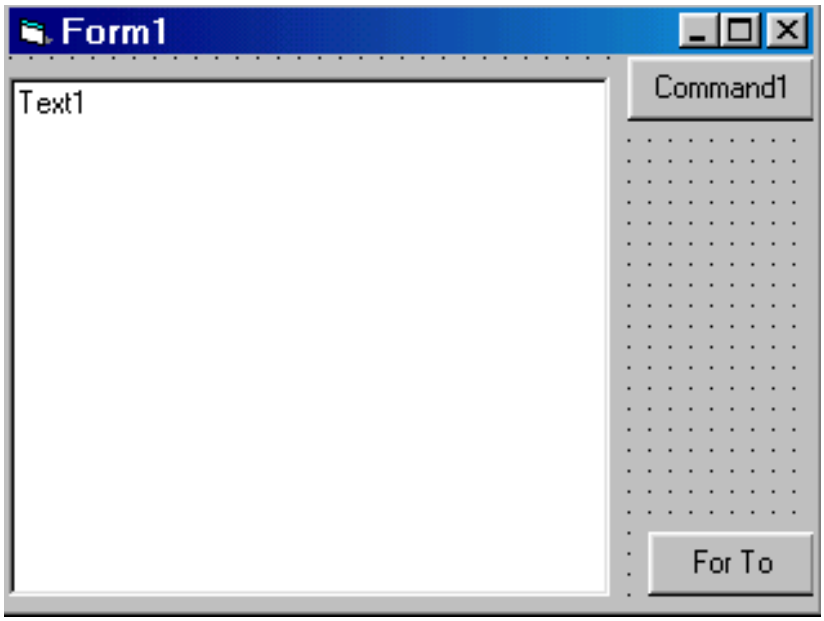
```
For variable = primer valor To último valor
    Sentencia1
    Sentencia2
    Sentencia3
    ...
    ...
Next
```

Desde el "primer valor" hasta el "último valor" se repetirá la ejecución de las sentencias 1, 2, 3, etc.

- **Byte** es un tipo de datos que representa un número entero de 0 a 255. Encambio "Integer" también es un número entero pero de –32768 a +32767

Habrás observado que si el número de veces que deseamos el "saludo" es relativamente grande, por ejemplo 100 veces, está claro que el formulario tiene un problema de espacio y también el **MsgBox** Vamos a intentar solucionar el problema en el siguiente programa ...

- Inserta en el "form" un **TextBox** y un **CommandButton** aproximadamente de la siguiente forma:



- Cambia las siguientes propiedades del **Text1**:
  Multiline = True
  ScrollBars = 2 - Vertical

- Escribe el siguiente procedimiento de evento:

```
Private Sub Command1_Click()
   Dim num As Integer
   Dim i As Byte
   Dim nom As String, salida As String
   num = InputBox("Cuántas veces quieres que te salude?")
   nom = InputBox("Cuál es tu nombre?")
   For i = 1 To num
      salida = salida & Str(i) & "  - Hola " & nom & vbCrLf
   Next
   Text1.Text = salida
End Sub
```

- Ejecuta el programa para un número de “saludos” tan grande como quieras.

En definitiva:
Cuando necesitemos una “salida” con mucha información, utilizaremos un **TextBox** con “Multiline = True” y “ScrollBars = 2 Vertical”, en lugar del **Form1.Print** o **MsgBox**

- Graba el programa como: **Prog06.frm**
Prog06.vbp

t) Crea un nuevo proyecto (Menú File – New Project – Estándar EXE)

- Inserta un CommandButton en el ángulo inferior derecho del form, de propiedades:
Name = cmdDoWhile
Caption = Do While

- Accede al módulo del formulario y escribe el siguiente procedimiento de evento:

```
Private Sub cmdDoWhile_Click()
   Dim contador As Integer
   contador = 1
   Do While contador <= 5
      Form1.Print "Pepe"
      contador = contador + 1
   Loop
End Sub
```

- Ejecuta varias veces el programa observando detenidamente lo que sucede.

La estructura de programación Do While - Loop

```
Do While condición
   Sentencia1
   Sentencia2
   …
   …
Loop
```

Mientras se cumpla la “condición” se repetirá la ejecución de las sentencias 1, 2, etc.

- Inserta otro CommandButton encima del anterior de propiedades:
Name = cmdOtroDoWhile
Caption = Otro Do While

- Escribe el siguiente procedimiento:

```
Private Sub cmdOtroDoWhile_Click()
   Dim pregunta As String * 1
   pregunta = "S"
   Do While pregunta = "S" Or pregunta = "s"
      Form1.Print "Pepe"
      pregunta = InputBox("Quieres continuar?")
   Loop
End Sub
```

- Ejecuta el programa para probarlo.

- Grábalo como **Prog07.frm**
Prog07.vbp

u) Crea un nuevo proyecto (Menú File – New Project – Standard EXE)

- Inserta un módulo estándar (Menú Project – Add Module) y escribe el siguiente procedimiento:

```
Public Sub Case1()
   Dim dia As Integer
   dia = InputBox("Escribe un número del 1 al 7")
   Select Case dia
      Case 1
         Form1.Print "1 - Lunes"
      Case 2
         Form1.Print "2 - Martes"
      Case 3
         Form1.Print "3 - Miercoles"
      Case 4
         Form1.Print "4 - Jueves"
      Case 5
         Form1.Print "5 - Viernes"
      Case 6
         Form1.Print "6 - Sábado"
      Case 7
         Form1.Print "7 - Domingo"
      Case Else
         Form1.Print "Esto no es un número del 1 al 7"
   End Select
End Sub
```

- En el form inserta un CommandButton en elángulo inferior derecho, con las propiedades:
Name = cmdCase
Caption = Case

- Accede al módulo del formulario y escribe el siguiente procedimiento de evento:

```
Private Sub cmdCase_Click()
   Case1
End Sub
```

– Ejecuta el programa varias veces, observando detenidamente lo que sucede.

- **La estructura de programación “Select Case”**

```
Select Case variable
    Case a
            Sentencia1
            Sentencia2
            ...
            ...
    Case b
            Sentencia3
            Sentencia4
```

```
                ...
                ...
        Case c
                Sentencia5
                Sentencia6
                ...
                ...
        Case Else
                Sentencia8
                Sentencia9
                ...
                ...
End Select
```

Viene a ser una generalización de la estructura **If – Then**: según el valor de la "variable", se ejecutarán unas sentencias u otras.

- Accede al módulo estándar y escribe el siguiente procedimiento:

```
Public Sub Case2()
   Dim x As String
   x = InputBox("Escribe VARÓN o HEMBRA, según tu sexo")
   Select Case x
      Case "VARÓN"
         Form1.Print "Qué tal guapo?"
      Case "HEMBRA"
         Form1.Print "Qué tal guapa?"
      Case Else
         Form1.Print "Qué tal sexo ambiguo?"
   End Select
End Sub
```

- En el "form" inserta otro CommandButton, encima del anterior con las propiedades:

Name = cmdOtroCase
Caption = Otro Case

- Accede al módulo del formulario y escribe el siguiente procedimiento de evento:

```
Private Sub cmdOtroCase_Click()
   Case2
End Sub
```

- Ejecuta el programa para comprobar que funciona

- Grábalo como **Prog08.bas, Prog08.frm, Prog08.vbp**

Programa que nos suma todos los números que queramos

- Nuevo Proyecto (Menú File – New Project – Standard EXE)

- Inserta en el "form" un CommandButton con las propiedades:
  Name = cmdSumaMucho
  Caption = Suma Mucho

- Inserta el siguiente procedimiento de evento:

```
Private Sub cmdSumaMucho_Click()
   Dim num As Double, total As Double
   Dim s As String
   num = InputBox("Escribe un número")
   s = s & "La suma de los números:" & vbCrLf
   s = s & num & Chr(9)
   Do While num <> 0
      total = total + num
      num = InputBox("Escribe un nuevo valor (Escribe 0 para terminar)")
      If num <> 0 Then
         s = s & num & Chr(9)
      End If
   Loop
   s = s & vbCrLf & vbCrLf
   s = s & " es = " & total
   Form1.Print s
   MsgBox s
End Sub
```

- Ejecuta el programa

- Grábalo como **Prog09.frm, Prog09.vbp**

Resolución de una ecuación de 2 grado

- Nuevo Proyecto

- CommandButton de propiedades:
  Name = cmdEcuacion2
  Caption = Ecuación de 2 grado

- Procedimiento de evento:

```
Private Sub cmdEcuacion2_Click()
   Dim a As Double, b As Double, c As Double
   Dim dis As Double, x1 As Double, x2 As Double
   Dim x As Double
   a = InputBox("Coeficiente de x^2= ")
   Form1.Print "Coeficiente de x^2= " & a
   If a = 0 Then
      Form1.Print "No es una ecuación de 2 grado"
   Else
      b = InputBox("Coeficiente de x = ")
      Form1.Print "Coeficiente de x = " & b
      c = InputBox("Termino independiente= ")
      Form1.Print "Termino independiente = " & c
```

```
        dis = b ^ 2 - 4 * a * c
        If dis = 0 Then
           x = (-b) / (2 * a)
           Form1.Print "La ecuación tiene una solución = " & x
        End If
        If dis < 0 Then
           Form1.Print "Las soluciones son imaginarias"
        End If
        If dis > 0 Then
           x1 = (-b + Sqr(dis)) / (2 * a)
           x2 = (-b - Sqr(dis)) / (2 * a)
           Form1.Print "x1 = " & x1
           Form1.Print "x2 = " & x2
        End If
     End If
  End Sub
```

- Ejecuta el programa para los siguientes casos:
  - a = 0
  - a = 1 ; b = 1 ; c = 1
  - a = 1 ; b = -4 ; c = 4
  - a = 1 ; b = 1 ; c = -6
  - Pruébalo también para valores decimales.

- La función incorporada al VB, **Sqr**, calcula la raíz cuadrada.

- Graba el programa como **Prog10.frm, Prog10.vbp**

Escalas de temperatura Celsius y Farenheit

- Nuevo Proyecto

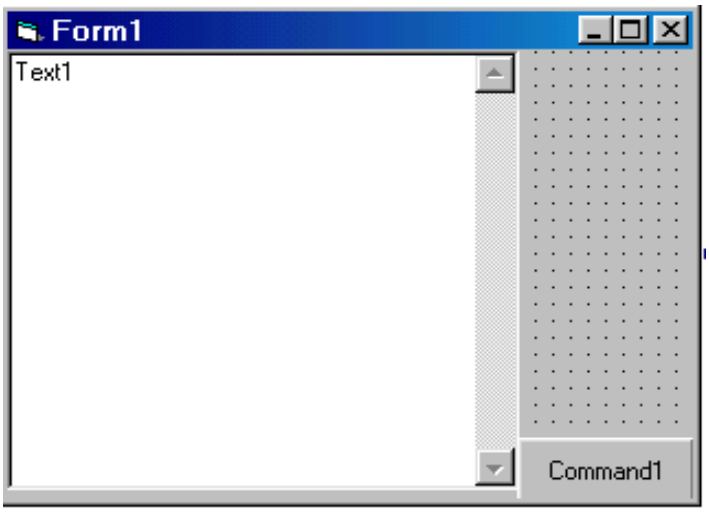
- **CommandButton**, de propiedades:

  Name = cmdTemperaturas
  Caption = Temperaturas

- **TextBox** de propiedades:

  Multiline = True
  ScrollBars = 2 - Vertical

- Distribución:



- Procedimiento de evento:

```
Private Sub cmdTemperaturas_Click()
   Dim contador As Integer
   Dim farenheit As Integer
   Dim celsius As Integer
   Dim s As String
   s = s & "Temperaturas Farenheit y Celsius" & vbCrLf
   s = s & "==============================" & vbCrLf
   For contador = -10 To 1000
      celsius = 10 * contador
      farenheit = 32 + (celsius * 9) / 5
      ' La fórmula anterior transforma la temperatura
      ' de grados centígrados a farenheit
      s = s & "celsius= " & celsius & _
         Chr(9) & "farenheit= " & farenheit _
         & vbCrLf
      If celsius = 0 Then
         s = s & "Atención: Punto de Congelación del agua" & vbCrLf
      End If
      If celsius = 100 Then
         s = s & "Atención: Punto de Ebullición del agua" & vbCrLf
      End If
      If celsius = 150 Then
         Exit For
      End If
   Next
   Text1.Text = s
End Sub
```

- Ejecuta el programa y observa …

- **Comentarios para el programador**:
Visual Basic interpreta que todo lo que está a la derecha del carácter “apóstrofo” en una línea cualquiera del programa es un **comentario** y no lo tiene en cuenta para nada.

- **Exit For**
Observa de qué forma, salimos del bucle **For – To – Next** si nos interesa

- Observa también que al utilizar el símbolo de concatenación “&” (en lugar de la “+”), no es necesario “convertir” los números a texto (Str).

Suma y Producto de los múltiplos de 2 inferiores a 30

- Nuevo Proyecto

- CommandButton de propiedades:
  Name = cmdMultiplos2
  Caption = Multiplos de 2

- TextBox de propiedades:
  Multiline = True
  ScrollBars = 2-Vertical

- Distribución: igual que el ejercicio anterior

- Procedimiento de evento:

```
Private Sub cmdMultiplos2_Click()
   Dim m2 As Integer, sum As Integer
   Dim pro As Double, s As String
   sum = 0: pro = 1
   s = s & "Múltiplo de 2  -  Suma Parcial  -  Producto Parcial" _
      & vbCrLf
   For m2 = 2 To 30 Step 2
      sum = sum + m2
      pro = pro * m2
      s = s & Chr(9) & m2 & Chr(9) & sum & _
         Chr(9) & pro & vbCrLf
   Next
   s = s & vbCrLf & vbCrLf
   s = s & "Suma Total = " & sum & vbCrLf
   s = s & "Producto total = " & pro
   Text1.Text = s
End Sub
```

- Ejecuta el programa

- Observa de qué forma podemos incluir dos senténcias o más en una misma línea: basta escribir dos puntos para separarlas.

- La cláusula **Step** (paso) indica el salto que debe hacer el índice correspondiente al **For**
De esta forma: **For x=2 To 15 Step 3**
Determinaria los valores: x = 2
x = 5
x = 8
x = 11
x = 14

- Graba el programa como **Prog12.frm, Prog12.vbp**

Cálculo del factorial de un número

- Nuevo Proyecto: Standard EXE

- CommandButton de propiedades: Name = cmdFactorial
Caption = Factorial

- Procedimiento de evento:

```
Private Sub cmdFactorial_Click()
   Dim num As Integer, fact As Double
   Dim i As Integer
   num = InputBox("Cálculo del factorial del número = ")
   fact = 1
   For i = 1 To num
      fact = fact * i
   Next
   Form1.Print "El factorial de " & num & " es " & fact
   MsgBox "El factorial de " & num & " es " & fact
End Sub
```

- Ejecuta el programa

- Recuerda que el factorial de un número "x" es **x! = x(x-1)(x-2)(x-3) ... 3 2 1**

- Graba el programa como **Prog13.frm, Prog13.vbp**

Programa que calcula los 10 primeros múltiplos del número que queramos, por último nos da la suma y el producto de todos ellos.

- Nuevo Proyecto: Standard EXE

- **CommandButton** de propiedades: Name = cmd10Multiplos
Caption = 10Múltiplos

- Procedimiento de evento:

```
Private Sub cmd10Multiplos_Click()
   Dim salida As String, num As Integer
   Dim mult As Double, sum As Double
   Dim prod As Double, i As Integer
   sum = 0: prod = 1
   num = InputBox("Múltiplos de que número?")
   For i = 1 To 10
      mult = num * i
      salida = salida & mult & Chr(9)
      sum = sum + mult
      prod = prod * mult
   Next
   salida = salida & vbCrLf
   salida = salida & "Suma = " & sum & vbCrLf
   salida = salida & "Producto = " & prod & vbCrLf
   Form1.Print salida
   MsgBox salida
End Sub
```

- Grábalo como **Prog14.frm, Prog14.vbp**

Tabla de valores de la función $y = x^2 - 5x + 10$

- Nuevo Proyecto: Standard EXE

- **CommandButton** de propiedades: Name = cmdFuncion
Caption = Tabla Valores

- **TextBox** de propiedades: Multiline = True
  ScrollBars = 2-Vertical

- Procedimiento de evento:

```
Private Sub cmdFuncion_Click()
  Dim x1 As Double, x2 As Double
  Dim incr As Double, i As Double
  Dim s As String, y As Double
  s = "Tabla de Valores de y=x^2 -5x+ 10" & vbCrLf
  x1 = InputBox("Escribe el menor valor de x de la tabla")
  x2 = InputBox("Escribe el mayor valor de x de la tabla")
  If x1 >= x2 Then
    MsgBox "No tiene sentido lo que quieres hacer"
  Else
    incr = InputBox("Escribe el incremento de x")
    If incr <= 0 Then
      MsgBox "No tiene sentido lo que pretendes hacer"
    Else
      For i = x1 To x2 Step incr
        y = i * i - 5 * i + 10
        s = s & "x = " & i & Chr(9) & _
          "y = " & y & vbCrLf
      Next
      Text1.Text = s
    End If
  End If
End Sub
```

- Ejecuta el programa para probarlo.

- Grábalo como **Prog15.frm, Prog15.vbp**

Cálculo de la hipotenusa de un triángulo rectángulo. El programa tiene la opción de volver a empezar

- Nuevo Proyecto: Standard EXE

- **CommandButton** de propiedades: Name = cmdPitagoras
  Caption = Pitágoras

- Procedimiento de Evento:

```
Private Sub cmdPitagoras_Click()
  Dim opc As String * 1, hipo As Double
  Dim cat1 As Double, cat2 As Double
  opc = "S"
  Do While UCase(opc) = "S"
    Form1.Cls
    cat1 = InputBox("Escribe el valor de un cateto")
    Form1.Print "Cateto 1 = " & cat1
    cat2 = InputBox("Escribe el valor del otro cateto")
```

```
        Form1.Print "Cateto 2 = " & cat2
        Form1.Print
        hipo = Sqr(cat1 * cat1 + cat2 * cat2)
        Form1.Print "Hipotenusa = " & hipo
        opc = InputBox("Quieres calcular otra hipotenusa (S/N)")
      Loop
    End Sub
```

- Grábalo como **Prog16.frm, Prog16.vbp**

- Observa:
  - **Form1.Cls**
    Borra el contenido del "Form"
  - **Ucase(cadena)**
    Es una función incorporada al Visual Basic que transforma todas las letras de "cadena" a mayúsculas.
    Escribir: **Ucase(opc)="S"**, es equivalente a **opc="S" Or opc="s"**, pero más corto.

Programa que calcula el número "e"

- Nuevo Proyecto

- **CommandButton** de propiedades: Name = cmdE
  Caption = E

- **TextBox** de propiedades: Multiline = True
  ScrollBars = 2 - Vertical

- Procedimiento de evento:

```
Private Sub cmdE_Click()
  Dim s As String, i As Double
  s = "El numero 'e' " & vbCrLf
  For i = 1 To 15
    s = s & "n = " & i & Chr(9) & _
      "e = " & (1 + 1 / i) ^ i & vbCrLf
  Next
  s = s & vbCrLf
  For i = 100 To 5000000000# Step 10000000
    s = s & "n = " & i & Chr(9) & _
      "e = " & (1 + 1 / i) ^ i & vbCrLf
  Next
  s = s & vbCrLf
  s = s & "Verdadero valor de 'e' = " & Exp(1)
  Text1.Text = s
End Sub
```

- Grábalo como **Prog17.frm, Prog17.vbp**

Programa que nos da la "nota" cualitativa a partir de la cuantitativa
El programa nos pide el número total de preguntas y el número de respuestas acertadas. A partir de aquí y utilizando la estructura **Select Case**, el programa nos da la nota cualitativa.

- Nuevo Proyecto: Standard EXE

- **CommandButton** de propiedades: Name = cmdNota
  Caption = Nota

- Procedimiento de evento:

```
Private Sub cmdNota_Click()
   Dim num As Integer, notanum As Integer
   Dim bien As Integer, notacual As String, s As String
   num = InputBox("Escribe el número total de preguntas")
   bien = InputBox("Escribe el número de respuestas acertadas")
   notanum = 10 * bien / num
   Select Case notanum
      Case 0 To 1
         notacual = "Muy Deficiente"
      Case 2 To 3
         notacual = "Deficiente"
      Case 4
         notacual = "Insuficiente"
      Case 5
         notacual = "Suficiente"
      Case 6
         notacual = "Bien"
      Case 7 To 8
         notacual = "Notable"
      Case 9 To 10
         notacual = "Excelente"
   End Select
   s = "Nota Cualitativa" & vbCrLf & vbCrLf
   s = s & "Número de preguntas = " & num & vbCrLf
   s = s & "Número de correctas = " & bien & vbCrLf
   s = s & "Nota cuantitativa= " & notanum & vbCrLf
   s = s & "Nota cualitativa : " & notacual
   MsgBox s
End Sub
```

- Ejecuta varias veces el programa, para comprobar que funciona.

- Grábalo como **Prog18.frm, Prog18.vbp**

Para saber más II

Introducción

Un programa informático, está constituido en un sentido general por **variables**, que contienenlos datos con los que se trabaja y por **algoritmos**, que son las sentencias que operan sobre estos datos. Estos datos y algoritmos suelen estar incluidos dentro de **funciones** o **procedimientos**.

Los lenguajes de alto nivel son más o menos comprensibles para el usuario, pero no para el **procesador** (ordenador). Para que éste pueda ejecutarlos es necesario traducirlos a su propio lenguaje máquina. Al paso del lenguaje de alto nivel al lenguaje máquina se le denomina **compilación**.

En VB esta etapa no se aprecia tanto como en otros lenguajes donde el programador tiene que indicar al ordenador explícitamente que realice dicha compilación.

Los programas de Visual Basic se dice que son **interpretados** y no compilados, ya que el código no se convierte a código máquina, sino que hay otro programa que durante la ejecución **interpreta** las líneas de código que ha escrito el programador.

Proyectos y Módulos

Un proyecto en VB es el conjunto de todos los ficheros necesarios para que un programa funcione: dicha información se almacena en un fichero con extensión **vbp** (Visual Basic Project)..

El caso más simple de un proyecto, es un único formulario y constará de dos ficheros: el que define el proyecto (*.vbp) y el que define el formulario (*.frm).

Los **Módulos** pueden ser de tres tipos
- Módulo asociado a un formulario (*.frm)
- Módulo estándar: contienen únicamente líneas de código basic (*.bas)
- Módulos de clase: contienen agrupaciones de código y datos denominados clases (*.cls)

El código está formado por pequeños bloques de programas, que pueden ser de tres tipos:
- Procedimientos de evento
- Procedimientos estándar
- Funciones (las veremos en el siguiente capítulo)

Además el módulo, puede contener una **parte General**, formada por variables comunes a todos los procedimientos del módulo.

Ámbito de las variables y procedimientos

- **Ámbito local**

Un módulo puede contener variables, procedimientos y funciones públicosy privados
- Los Públicos

Son aquellos a los que puede acceder libremente desde cualquier punto del proyecto. Es necesario preceder al nombre de la variable, procedimiento o función de la palabra: PUBLIC

Ejemplos: Public x As Integer
Public Sub Pepe1(x As Integer, …)

Para utilizar un elemento “Public” desde otro módulo: si estamos por ejemplo en el módulo 2:
Modulo1.Variable1
Call Modulo1.Procedimiento1(...)

- Los Private

No son accesibles desde ningún otro módulo distinto de aquel en el que se haya declarado.

Se llama **variable LOCAL** a una variable definida dentro de un procedimiento o función. Las variables locales solo son accesibles en el procedimiento o función donde están declaradas.

Static o Dim?
Una variable local es reinicializada (a 0 si es númerica y a " " si es texto, por defecto) cada vez que se entra en el procedimiento o función. Es decir, una variable local no conserva su valor entre una llamada al procedimiento y la siguiente. Para conseguir que el valor de la variable se conserve hay que declarar la variable como STATIC
Es decir:Dim x As Integer
Static y As Integer
"x" inicialmente es siempre 0, encambio "y" depende del último valor que ha tomado.

- **Ámbito global**
Se puede acceder a una variable o procedimiento **GLOBAL** desde cualquier parte de la aplicación.
Para conseguir un "elemento" (variable, procedimiento o función) GLOBAL, hay que declararlo anteponiendo la palabra PUBLIC en la parte general de un módulo estándar *.bas o de un módulo de formulario *.frm

Variables
Una variable es un nombre que designa una zona de memoria y contiene un valor de un tipo determinado.
Las variables pueden cambiar su valor a lo largo de la ejecución de un programa.
La variable que no cambia su valor a lo largo de la ejecución de un programa se le llama "constante".
Declaración de Constantes:

```
Const x=459                    ' Por defecto las constantes son Private
Public Const sal="Hola"
Private Const x As Integer=5
Const s="Adiós", h As Double=5.0792
```

VB tiene sus propias constantes, para investigarlas, basta que hagas:
Menú View
Object Browser
Observarás que muchas de ellas empiezan por el prefijo **vb**

Tipos de datos

- **Boolean** True o False

- **Byte** 0 a 255 entero

- **Integer** -32768 a 32767 entero

- **Long** -2147483648 a 2147483647 entero

- **Single** $-3.40\ 10^{38}$ a $3.40\ 10^{38}$ decimal

- **Double** $-1.79\ 10^{308}$ a $1.79\ 10^{308}$ decimal

- **Courrency** $-9.22\ 10^{14}$ a $9.22\ 10^{14}$ decimal

- **String** cadena de 0 a 65500 caracteres

- **Date** fecha

- **Variant** depende del valor de la variable, es decir, si la variable es una fecha será tipo Date, si es un número natural inferior a 255 será tipo Byte, etc.

Observando el tipo **Variant**, podriamos llegar a la conclusión de olvidarnos de todos los tipos de datos, excepto el propio Variant, pero hay un problema: el tipo Variant ocupa y de mucho, mucha memoria.

Declaración de Variables

Dim o Static nombreVariable As TipoVariable

Ejemplos:

```
Dim radi As Double, super As Single
Dim nom As String
Dim x As String*5
Static a As Integer, b As Integer
```

Si escribimos: **Dim x, y As String**
"x" será Variant, e "y" es String
Lo correcto seria: **Dim x As String, y As String** o **Dim x As String**
Dim y As String

Es un buen hábito de programación la declaración de los tipos de variable que se van a utilizar en un procedimiento, antes de que vayan a ser utilizadas. Esto aumenta la legibilidad de los programas.

El VB no nos obliga a declarar préviamente las variables que utilizamos en un programa (por defecto todas las variables no declaradas son "Variant"), a diferencia de otros lenguajes de programación como el C++ o el Java.
Seria interesante **obligar** al VB a la declaración de variables, ya que el error típico de programación consiste en cambiar el nombre de una variable por error; si el VB nos obligara a declarar todas las variables, detectaríamos inmediatamente el error.

Para obligar a la declaración previa de variables, basta escribir en la "parte General" del módulo la sentencia: **Option Explicit**

De todas formas, podemos conseguir que el VB lo haga por nosotros con:

```
Menú Tools
    Options...
        Solapa: Editor
        Activa la opción: Require Variable Declaration
```

Operadores

Aritméticos

^	exponenciación
\	división entera
mod	resto de la división entera

Cuando en una expresión aritmética intervienen operandos de diferentes tipos, el resultado se expresa en la misma precisión que la del operando que la tiene más alta.
De menor a mayor: Integer, Long, Single, Double, Currency

Concatenación
& o +

Relacionales
Los opradores relacionales también conocidos como operadores de "Comparación", comparan dos expresiones dando un resultado True, False o Null
= , <> , < , > , <= , >=

Lógicos
Not (no), and (y), or (o), xor ("or" exclusivo)

Algunas funciones incorporadas

Funciones de caracteres
- **Len(x)** número de caracteres de la cadena "x"
- **Lcase(x)** convierte a minúsculas
- **Ucase(x)** convierte a mayúsculas
- **Str(x)** convierte el número x a cadena de texto.
- **Val(x)** función contraria a la anterior

Funciones matemáticas
- **Abs(x)** valor absoluto
- **Int(x)** parte entera
- **Exp(x)** exponencial
- **Log(x)** logaritmo neperiano
- **Round(x,n)** redondeo
- **Rnd()** número aleatorio
- **Sin(x)** seno
- **Cos(x)** coseno
- **Tan(x)** tangente
- **Sqr(x)** raíz cuadrada

El arte de programar: Los Algoritmos
Un "algoritmo" es en un sentido amplio una "secuencia de pasos o etapas que conducen a la realización de una tarea".
Los primeros algoritmos nacieron para resolver problemas de tipo matemático. Antes de escribir un programa de ordenador, hay que tener muy claro el algoritmo, es decir, cómo se va a resolver el problema considerado.

Ejemplo:
Algoritmo de Euclides para calcular el m.c.d. de dos números
mcd(38,48)
1.- necesitamos saber el número mayor
2.- divido el mayor entre el menor, 48/36 = 1 y resto12
3.- divido el divisor de antes entre el resto (de antes): 36/12 = 3 y resto 0
4.- continuamos dividiendo **divisor/resto**, hasta que la división sea exacta
5.- el m.c.d es el último resto distinto de cero, en nuestro caso 12

En general: mcd(a,b)

```
1.-     if a<b then
            aux = a
            a = b
            b = aux
        end if
        A partir de este momento a es el mayor y b el menor
2.-     if a mod b = 0 then
            resto = b
        end if
3.-     do while a mod b <> 0
            resto = a mod b
            a = b
            b = resto
        loop
4.-     mcd = resto
```

Ejercicios II

1) Haz un programa que funcione de la siguiente forma:
- El programa nos pide que escribamos dos números positivos menores de 57
- El programa nos da como resultado el producto de los dos números
- Si los números no son positivos o son mayores de 57, el programa nos lo dice
- El programa nos pregunta al final si queremos volver a empezar

Graba el programa con el nombre **Ejer01** (es decir Ejer01.frm / Ejer01.vbp), en *TuCarpeta*.

2) Escribe un programa que nos vaya pidiendo números. Si escribimos el número 9999 se acaba, por último el programa nos da como resultado el número de números introducidos, exceptuando el 9999.
Graba el programa con el nombre **Ejer02** en *TuCarpeta*

3) Haz un programa que escriba todos los múltiplos de 23 inferiores a 1000 y por último nos dé la suma de todos ellos.
Graba el programa con el nombre **Ejer03**

4) Haz un programa que sirva para hacer una tabla de valores de la función y=sen(7x-5)
- El programa nos pide los dos valores de "x" (valores máximo y mínimo de la tabla)
- El programa nos pide el incremento (variación) de la "x"

Graba el programa con el nombre **Ejer04**

5) Haz un programa que sirva para calcular un cateto de un triángulo rectángulo a partir del otro cateto y la hipotenusa, de la siguiente forma:
- El programa nos pide el valor de la hipotenusa.
- El programa nos pide el valor de un cateto.
- Si el cateto es mayor que la hipotenusa, el programa nos da un mensaje de error y se acaba.
- El programa nos da como resultado el valor del otro cateto y nos pregunta si queremos volver a empezar.

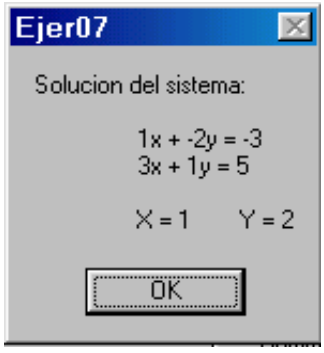
Graba el programa con el nombre **Ejer05**

6) Haz un programa que sirva para resolver ecuaciones de 2 grado del tipo $ax^2 + bx = 0$
Graba el programa con el nombre **Ejer06**

7) Haz un programa que sirva para resolver sistemas de ecuaciones del tipo:

$$\left.\begin{array}{l} ax + by = c \\ dx + ey = f \end{array}\right\}$$

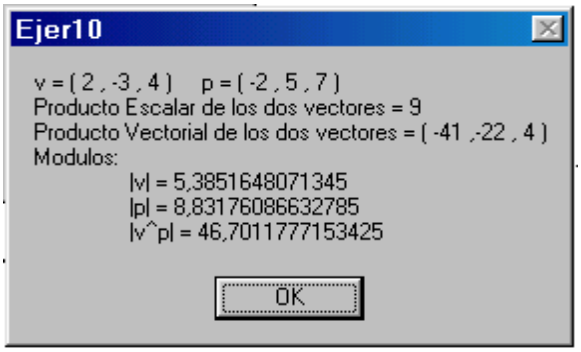
Graba el programa con el nombre **Ejer07**.
x = (ce – bf)/(ae – bd) ; y = (af – dc)/(ae – bd)
Prueba el funcionamiento del programa para el caso a = 1; b=-2 ; c=-3 ; d=3 ; e=1 ; f=5; si todo funciona correctamente: x = 1, y = 2
La "salida" deberia ser de la siguiente forma:



8) Haz un programa que escriba los 15 primeros múltiplos de 7, su suma y su producto. El programa ha de tener la posibilidad de volver a empezar.
Graba el programa con el nombre **Ejer08**

9) Haz un programa que sirva para calcular el área de un triángulo o el área de un rectángulo o el área de un círculo. El programa ha de tener la posibilidad de volver a empezar.
Graba el programa con el nombre **Ejer09**

10) Haz un programa tal que: dados dos vectores del espacio calcule su producto escalar, producto vectorial y además nos dé el módulo de los dos vectores y también el módulo del producto vectorial
Graba el programa con el nombre **Ejer10**
v = (a,b,c) p=(d,e,f)
Producto Escalar = ad + de + cf
Producto Vectorial = (bf-ec,dc-af,ae-bd)
Módulo de v = $\sqrt{a^2+b^2+c^2}$
La salida deberia ser:



11) Haz un programa que nos pida un número y dé como resultado la tabla de multiplicar del número introducido.
Graba el programa con el nombre **Ejer11**

12) Haz un programa que calcule el número "e" mediante el desarrollo en serie:
e = 1 + 1/(1!) + 1/(2!) + 1/(3!) + 1/(4!) + ... + 1/(50!)
Graba el programa con el nombre **Ejer12**

III .- Arrays y Funciones

a) Haz un nuevo proyecto tipo “Standard EXE”

- Inserta en el ángulo inferior derecho del “form” un **CommandButton** de propiedades:
  Name = cmdArray1
  Caption = Array1

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdArray1_Click()
   Dim a(1 To 3) As Double
   Dim i As Integer
   For i = 1 To 3
      a(i) = InputBox("Escribe un número")
   Next
   Form1.Print "Los números que hay en la matriz son:"
   For i = 1 To 3
      Form1.Print "a(" & i & ") = " & a(i)
   Next
End Sub
```

- Ejecuta el programa observando detenidamente lo que sucede.

Estudio del cmdArray1_Click()

* **Dim a(1 To 3) As Double**
Definimos lo que se llama “array” o “matriz” o “vector” o “arreglo” de una dimensión. Y no es más que una variable que consta de 3 números **Double**:
Para acceder a los elementos de nuestro “array” hemos de utilizar la notación:
  a(1) = primer elemento, a(2) = segundo elemento, a(3) = tercer elemento, ...

b) El programa anterior es muy bonito, pero no sirve para nada, vamos a hacer un programa un poco más complicado ...

Observa la siguiente tabla:

	Lunes	**Martes**	**Miercoles**	**Jueves**	**Viernes**
Inicio	8	10,5	6	9	7
Fin	14	17	13,5	13	18

Resulta que cada día de la semana hacemos una jornada laboral distinta, en el ejemplo de la tabla (que representa una semana determinada), el lunes empezamos a trabajar a las 8 y terminamos a las 2 de la tarde, el martes empezamos a trabajar a las 10 y media y terminamos a las 17h, etc.

Vamos a hacer un programa para introducir y guardar los datos de dicha tabla.

- Inserta un nuevo CommandButton en el form de propiedades:
  Name = cmdArray2
  Caption = Array2

- Escribe el siguiente procedimiento:

```
Private Sub cmdArray2_Click()
   Dim a(1 To 2, 1 To 5) As Double
```

```
    Dim i As Byte, j As Byte
    For j = 1 To 5
      For i = 1 To 2
        a(i, j) = InputBox("Escribe la hora de inicio " & _
         "y después la hora de finalización" & _
         vbCrLf & "para cada día de la semana " & _
         "empezando por el lunes y acabando" & _
         vbCrLf & "en el viernes")
      Next
    Next
    Form1.Print "Los valores de la matriz son"
    For j = 1 To 5
      For i = 1 To 2
        Form1.Print a(i, j)
      Next
    Next
  End Sub
```

- Graba el proyecto como **Prog19.frm**
Prog19.vbp

- En esta ocasión trabajamos con un **array bidimensional** o matriz de dos dimensiones:
a(1 To 2 , 1 To 5)
Sus elementos:
a(1,1), a(1,2), a(1,3), a(1,4), a(1,5), a(2,1), a(2,2), a(2,3), a(2,4), a(2,5)

El programa **cmdArray2_Click**, hemos de reconocer que está muy bien para utilizar matrices bidimensionales y ciclos anidados, pero es completamente inútil.

Vamos a modificar el programa anterior para que nos calcule el "número total de horas trabajadas a la semana" ...

c) Inserta un nuevo CommandButton en el form, de propiedades: Name = cmdArray3
Caption = Array3

Escribe el siguiente procedimiento de evento:

```
  Private Sub cmdArray3_Click()
    Dim a(1 To 2, 1 To 5) As Double
    Dim i As Byte: Dim j As Byte
    Dim suma As Double
    Dim diaria As Double
    For j = 1 To 5
      For i = 1 To 2
        a(i, j) = InputBox("Introduce los valores igual que antes")
      Next
    Next
    suma = 0
    For j = 1 To 5
      diaria = a(2, j) - a(1, j)
      suma = suma + diaria
      Form1.Print "Una jornada = " & diaria
    Next
    Form1.Print "Toda la semana = " & suma
  End Sub
```

- Ejecuta el programa.
Si todo funciona correctamente en el formulario aparecerá:
Una jornada = 6

Una jornada = 6,5
Una jornada = 7,5
Una jornada = 4
Una jornada = 11
Toda la semana = 35

- Graba el proyecto con el mismo nombre: **Prog19.frm**
Prog19.vbp

d) Crea un nuevo proyecto tipo “Standard EXE”
- Inserta un CommandButton en el ángulo inferior derecho del form, de propiedades:
Name = cmdMedia
Caption = Media

- Inserta un TextBox, que ocupe practicamente todo el formulario de propiedades:
Multiline = True
ScrollBars = 2-Vertical

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdMedia_Click()
   Dim s As String, num As Integer, sum As Double, i As Byte
   Dim x() As Double
   s = "PROMEDIO DE UN NUMERO CUALQUIERA DE ELEMENTOS" & _
         vbCrLf
   sum = 0
   num = InputBox("Escribe el numero de elementos")
   ReDim x(num)
   For i = 1 To num
      x(i) = InputBox("Numero?")
      s = s & x(i) & vbCrLf
      sum = sum + x(i)
   Next
   s = s & vbCrLf
   s = s & "El promedio de todos estos numeros es " & sum / num
   Text1.Text = s
End Sub
```

- Ejecuta el programa varias veces para probarlo

- Graba el programa con el nombre: **Prog20.frm**
Prog20.vbp

- Observa de qué forma definimos un **array dinámico**: al principio del programa, no sabemos el número de elementos (Dim x() As Double); cuando ya sabemos el número de elementos: **Redim x(variable)**

e) **Programa que calcula, dada una serie de 5 números, la media aritmética, las desviaciones respecto a la media, la desviación media , la varianza y la desviación típica**

- Crea un nuevo proyecto.

- Inserta un CommandButton en el ángulo inferior derecho del formulario, de propiedades:
Name = cmdEstadistica
Caption= Estadística

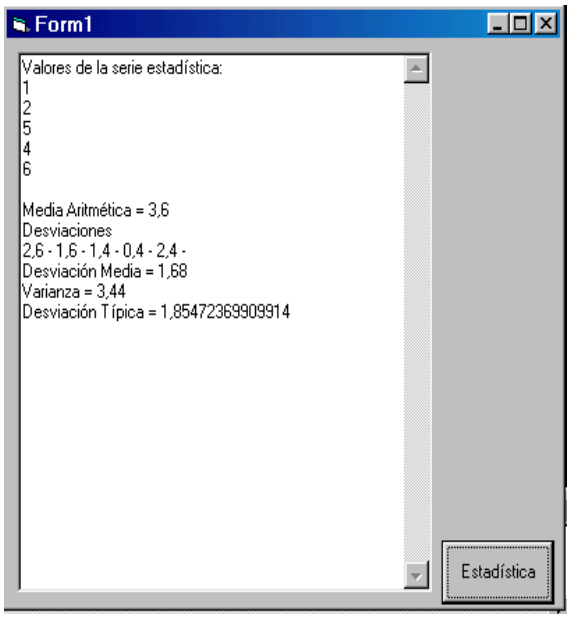
- Inserta un TextBox, que ocupe practicamente todo el formulario, de propiedades:
  Multiline = True
  ScrollBars = 2-Vertical

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdEstadistica_Click()
   Dim s As String, sum As Double, med As Double
   Dim num(1 To 5) As Double
   Dim desv(1 To 5) As Double
   Dim sum1 As Double, sum2 As Double, i As Byte
   Dim desmed As Double, vari As Double, dt As Double
   s = "": sum = 0: sum1 = 0: sum2 = 0
   s = s & "Valores de la serie estadística:" & vbCrLf
   For i = 1 To 5
      num(i) = InputBox("Número ?")
      s = s & num(i) & vbCrLf
      sum = sum + num(i)
   Next
   med = sum / 5
   s = s & vbCrLf: s = s & "Media Aritmética = " & med & vbCrLf
   s = s & "Desviaciones" & vbCrLf
   For i = 1 To 5
      desv(i) = Abs(num(i) - med)
      sum1 = sum1 + desv(i)
      sum2 = sum2 + desv(i) * desv(i)
      s = s & desv(i) & " - "
   Next
   desmed = sum1 / 5
   vari = sum2 / 5
   dt = Sqr(vari)
   s = s & vbCrLf
   s = s & "Desviación Media = " & desmed
   s = s & vbCrLf
   s = s & "Varianza = " & vari & vbCrLf
   s = s & "Desviación Típica = " & dt
   Text1.Text = s
End Sub
```

- Ejecuta el programa varias veces.

- Pruébalo para el caso:



- Grábalo como **Prog21.frm**
Prog21.vbp

f) Crea un nuevo proyecto

- Inserta un módulo estándar (Menú Project – Add Module)
En la nueva ventana correspondiente al módulo estándar escribe:

```
Public Function MEDIA(n1 As Integer, n2 As Integer) As Double
   MEDIA = (n1 + n2) / 2
End Function
```

- Inserta en el "form" un CommandButton de propiedades:
Name = cmdFuncion1
Caption = Función 1

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdFuncion1_Click()
   Form1.Print MEDIA(2, 3)
End Sub
```

- Ejecuta el programa, observando lo que sucede.

- Para ver mejor la utilidad de nuestra función, inserta otro CommandButton de propiedades:
Name = cmdFuncion2
Caption = Función 2

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdFuncion2_Click()
   Dim x As Integer, y As Integer
   x = InputBox("Escribe un número entero")
   y = InputBox("Escribe otro número entero")
   Form1.Print "El PROMEDIO de " & x & " y " & _
     y & " es " & MEDIA(x, y)
End Sub
```

- Ejecuta el programa unas cuantas veces, observando lo que sucede

- Graba el programa como **Prog22.bas**
Prog22.frm
Prog22.vbp

g) Escribe en el módulo **Prog22.bas** la siguiente función:

```
Public Function Raiz4(n As Double) As Double
   Raiz4 = Sqr(Sqr(n))
 End Function
```

- Inserta en el "form" un nuevo CommandButton de propiedades:
Name = cmdFuncion3
Caption = Función 3

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdFuncion3_Click()
   Form1.Print "La raíz cuarta de 625 es " & Raiz4(625)
   Form1.Print "La raíz cuarta de 72,81 es " & Raiz4(72.81)
   Dim x As Double
   x = InputBox("Escribe un número positivo")
   Form1.Print "La raíz cuarta de " & x & " es " & _
     Raiz4(x)
End Sub
```

- Ejecuta el programa unas cuantas veces.

- Para solucionar los posibles cálculos con números negativos, corrige la "function" de forma que nos quede:

```
Public Function Raiz4(n As Double) As Double
   If n < 0 Then
      MsgBox "No se puede calcular la raíz " & _
        "cuarta de un número negativo"
   Else
      Raiz4 = Sqr(Sqr(n))
   End If
End Function
```

- Inserta en el form un nuevo CommandButton de propiedades:
  Name = cmdFuncion4
  Caption = Función 4

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdFuncion4_Click()
   Dim num As Double
   num = InputBox("Introduce un número")
   Form1.Print "La raíz cuarta de " & num & " es " _
     & Raiz4(num)
End Sub
```

- Ejecuta el programa

- Grábalo con el mismo nombre: **Prog22.bas**
  Prog22.frm
  Prog22.vbp

h) Crea un nuevo proyecto del tipo "Standard EXE"

- Inserta un CommandButton en el formulario de propiedades:
  Name = cmdAreas
  Caption = Áreas

- Accede al **módulo** del formulario y escribe, en la parte "general" del módulo, las siguientes funciones:

```
Public Function AYUDA()
   Form1.Print "Escribe R para calcular el área de un Rectángulo"
   Form1.Print "Escribe T para un triángulo"
End Function

Public Function Rectangulo(bas As Double, alt As Double) As Double
   Rectangulo = bas * alt
End Function

Public Function Triangulo(bas As Double, alt As Double) As Double
```

```
    Triangulo = bas * alt / 2
End Function
```

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdAreas_Click()
    Dim opcion As String * 1
    Dim al As Double, ba As Double
    opcion = InputBox("Que opción? R/T/A")
    Select Case opcion
        Case "R"
            ba = InputBox("Base del rectángulo")
            al = InputBox("Altura del rectángulo")
            Form1.Print "El área del rectángulo es = " & Rectangulo(ba, al)
        Case "T"
            ba = InputBox("Base del triángulo")
            al = InputBox("Altura del triángulo")
            Form1.Print "El área del triángulo es = " & Triangulo(ba, al)
        Case Else
            AYUDA
    End Select
End Sub
```

- Ejecuta el programa

- Grábalo como **Prog23.frm**
Prog23.vbp

- Observa pues que no hay ningún tipo de problema para incluir las funciones en un módulo de formulario, es decir no es imprescindible el uso de un módulo estándar para escribir las funciones.

i) Crea un nuevo proyecto del tipo “Standard EXE”

- Inserta **tres** CommandButton, en el ángulo inferior derecho del form.

- Escribe los siguientes procedimientos de evento.

```
Private Sub Command1_Click()
    Cls
    Dim I As Integer
    For I = 1 To 5
        Print Rnd
    Next
End Sub

Private Sub Command2_Click()
    Cls
    Dim Tiradas As Integer, NumeroDeCaras As Integer
    Dim NumeroDeCruces As Integer
    Dim Increible As Integer
    Dim I As Integer
    Dim Resultado As Single
    Tiradas = InputBox("Escribe el numero de tiradas")
    For I = 1 To Tiradas
        Resultado = Rnd
        Select Case Resultado
            Case Is < 0.5
                NumeroDeCaras = NumeroDeCaras + 1
            Case 0.5
```

```
            Print "Moneda de canto"
            Increible = Increible + 1
          Case Else
            NumeroDeCruces = NumeroDeCruces + 1
        End Select
      Next
      Print "Caras = " & NumeroDeCaras
      Print "Cruces = " & NumeroDeCruces
      If Increible > 0 Then
        Print "Monedas de canto"
      End If
    End Sub

    Private Sub Command3_Click()
      Cls
      Randomize
      Print "10 tiradas aleatorias de un dado de parchis"
      For I = 1 To 10
        Print Fix(6 * Rnd) + 1
      Next
    End Sub
```

- Ejecuta el programa varias veces

- Grábalo como **Prog24.frm**
Prog24.vbp

- **Estudio del Prog24: La función RND**
Cada vez que el ordenador procesa una línea que contiene las sentencias **Print Rnd**, en el "form" aparece un número aleatorio entre 0 y 1. El número puede ser cero, pero nunca puede ser 1

$0 \leq Rnd < 1$

En realidad la serie de números del **Rnd** es pseudoaleatoria, porque cada vez es la misma. Ejecuta varias veces el **Command1_Click** y lo observarás (deberás cerrar cada vez el **Prog24).**

Para conseguir que la función **Rnd** genere verdaderos números aleatorios, en el programa hemos de anteponer la sentencia **Randomize**, que es lo que sucede en el **Command3_Click**

Observa:

x = Rnd	$0 \leq x < 1$	x número decimal
y = 6*Rnd	$0 \leq y < 6$	y número decimal
z = Int((y-x)*Rnd+x)	$x \leq z < y$	z número entero

j) Crea un nuevo proyecto tipo "Standard EXE"

- Inserta en el "form" dos CommandButton de propiedades:

Name = cmdTipo1	Name = cmdTipo2
Caption = Función Tipo 1	Caption = Función Tipo 2

- Escribe en la parte "General" del módulo del "form", las siguientes funciones:

```
    Public Function mediageo1()
      Dim x As Integer, y As Integer
      x = InputBox("Escribe un número entero")
      y = InputBox("Escribe otro número entero")
      MsgBox "La media geométrica de " & x & _
        " y " & y & " es " & Sqr(x * y)
    End Function
```

```
Public Function mediageo2(a As Integer, b As Integer) As Double
   mediageo2 = Sqr(a * b)
End Function
```

- Escribe los siguientes procedimientos de evento:

```
Private Sub cmdTipo1_Click()
   ' Este programa sólo contiene una
   ' llamada a una función
   mediageo1
End Sub

Private Sub cmdTipo2_Click()
   Dim x As Integer, y As Integer
   x = InputBox("Escribe un número entero")
   y = InputBox("Escribe otro número entero")
   MsgBox "La media geométrica de " & x & _
     " y " & y & " es " & mediageo2(x, y)
End Sub
```

- Grábalo como **Prog25.frm**
Prog25.vbp

- Ejecútalo varias veces

El primer tipo de función (mediageo1) se dice que es una **función sin retorno de parámetros**
El segundo tipo (mediageo2) es una **función que retorna parámetros**

k) **Programa que determina si un número es primo, utilizando una función**

- Nuevo proyecto del tipo Standard EXE

- Inserta un CommandButton de propiedades:
Name = cmdPrimo
Caption = Primo

- En la parte "General" del módulo del formulario, escribe la función:

```
Public Function Primo(x As Integer) As String
   Dim resto As Integer, i As Integer, opc As Integer
   opc = 0
   For i = 2 To x - 1
      resto = x Mod i
      If resto = 0 And x <> 2 Then
         opc = 1
         Exit For
      End If
   Next
   If opc = 1 Then
      Primo = "N"
   Else
      Primo = "S"
   End If
End Function
```

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdPrimo_Click()
   Dim num As Integer
   num = InputBox("Escribe un número entero")
   If Primo(num) = "S" Then
      Form1.Print "El número " & num & " es primo"
   Else
      Form1.Print "El número " & num & " no es primo"
   End If
End Sub
```

- Ejecuta varias veces el programa para comprobar si funciona

- Grábalo como **Prog26.frm**
Prog26.vbp

l) **Programa que calcula el M.C.D. de dos números, utilizando una función**

- Nuevo Proyecto

- Inserta un CommandButton de propiedades:
Name = cmdMCD
Caption = M.C.D.

- En la parte "General" del módulo del formulario, escribe la función:

```
Public Function MCD(a As Integer, b As Integer) As Integer
   Dim resto As Integer, aux As Integer
   If a < b Then
      aux = a
      a = b
      b = aux
   End If
   If a Mod b = 0 Then
      resto = b
   End If
   Do While a Mod b <> 0
      resto = a Mod b
      a = b
      b = resto
   Loop
   MCD = resto
End Function
```

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdMCD_Click()
   Dim x As Integer, y As Integer
   x = InputBox("Escribe un número")
   y = InputBox("Escribe otro número")
   Form1.Print "El MCD de " & x & " y " & y & " es " & MCD((x), (y))
End Sub
```

- Ejecútalo varias veces: MCD(5 , 25) = 5
MCD(7 , 3) = 1
MCD(720 , 300) = 60

- Grábalo como **Prog27.frm**
Prog27.vbp

m) **Números Aleatorios**

- Nuevo Proyecto

- CommandButton de propiedades: Name = cmdAleatorios
Caption = Aleatorios

- Inserta un TextBox, que ocupe prácticamente todo el form, de propiedades:
Multiline = True
ScrollBars = 2-Vertical

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdAleatorios_Click()
   Dim s As String, i As Byte
   Dim x As Double, y As Double
   Randomize
   s = "NÚMEROS ALEATORIOS" & vbCrLf & vbCrLf & vbCrLf
   s = s & "50 números aleatorios entre 0 y 1" & vbCrLf & vbCrLf
   For i = 1 To 50
      s = s & Rnd & " - "
   Next
   s = s & vbCrLf & vbCrLf & vbCrLf
   s = s & "50 números aleatorios enteros entre 3 y 7" & vbCrLf
   s = s & "incluido el 3 y excluido el 7" & vbCrLf
   s = s & vbCrLf
   For i = 1 To 50
      s = s & Int((7 - 3) * Rnd + 3) & " - "
   Next
   s = s & vbCrLf
   s = s & vbCrLf
   s = s & "50 números aleatorios enteros entre 3 y 7" & vbCrLf
   s = s & "incluidos el 3 y el 7 " & vbCrLf
   s = s & vbCrLf
   For i = 1 To 50
      s = s & Int((7 + 1 - 3) * Rnd + 3) & " - "
   Next
   s = s & vbCrLf
   s = s & vbCrLf
   s = s & "50 números aleatorios enteros entre los dos que tú quieras" & vbCrLf
   s = s & "incluidos los dos extremos " & vbCrLf
   x = InputBox("Escribe el menor")
   y = InputBox("Escribe el mayor")
   s = s & vbCrLf
   s = s & "concretamente entre " & x & " y " & y & vbCrLf & vbCrLf
   For i = 1 To 50
      s = s & Int((y + 1 - x) * Rnd + x) & " - "
   Next
   Text1.Text = s
End Sub
```

- Ejecuta el programa.

- Grábalo como **Prog28.frm**
Prog28.vbp

- Fórmulas Generales:
Int((y-x)*Rnd+x) números enteros aleatorios entre x e y, incluido x, excluido y
Int((y+1-x)*Rnd+x) números enteros aleatorios entre x e y, incluidos los dos

n) **Adivinanzas**

Vamos a hacer un programa que nos pregunte un número entero del 1 al 10, y el usuario del programa tiene 5 tentativas para adivinarlo.

- Nuevo Proyecto

- Inserta un CommandButton de propiedades: Name = cmdAdivinanza
Caption = Adivinanza

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdAdivinanza_Click()
   Dim x As Integer, num As Integer
   Dim i As Byte, control As Byte
   i = 0: control = 0: Randomize
   x = Int((10 + 1 - 1) * Rnd + 1)
   Do While i < 5
      i = i + 1
      num = InputBox("Escribe un entero del 1 al 10, intento " & i)
      If num = x Then
         Form1.Print "Lo has acertado en " & i & " tentativas"
         i = 5
         control = 1
      End If
   Loop
   If control = 0 Then
      Form1.Print "Lo siento pero se han acabado tus vidas, el número era " & x
   End If
End Sub
```

- Ejecuta el programa

- Grábalo como: **Prog29.frm**
Prog29.vbp

o) **Programa que nos pregunta 5 sumas aleatoriamente y al final nos da la "nota"**

- Nuevo proyecto

- Inserta un CommandButton de propiedades:
Name = cmdSumasAleatorias
Caption = Sumas Aleatorias

- Escribe el siguiente procedimiento de evento:

```
Private Sub cmdSumasAleatorias_Click()
```

```
    Dim x As Integer, y As Integer, z As Integer
    Dim nota As Byte, i As Byte
    Randomize
    For i = 1 To 5
       x = Int((9 + 1 - 1) * Rnd + 1)
       y = Int((9 + 1 - 1) * Rnd + 1)
       z = InputBox(x & " + " & y & " = ")
       If z = x + y Then
          Form1.Print x & " + " & y & " = " & z
          Form1.Print Chr(9) & "Muy bien"
          nota = nota + 1
       Else
          Form1.Print x & " + " & y & " = " & z
          Form1.Print "Lo siento pero es falso"
          Form1.Print x & " + " & y & " = " & (x + y)
       End If
    Next
    Form1.Print vbCrLf & "Tu nota es " & (2 * nota)
End Sub
```

- Ejecuta el programa

- Grábalo como **Prog30.frm**
Prog30.vbp

Para saber más III

Funciones y Procedimientos

En VB se dintingue entre “funciones” y “procedimientos Sub”, la diferencia entre ambos es que una función tiene algún valor de retorno (en general).

Por tanto, un procedimiento “Sub” es un segmento de código independiente del resto, que una vez llamado por el programa, ejecuta un número determinado de instrucciones.

Para llamar desde un formulario a un procedimiento o función **“Public”** definido en otro formulario es necesario preceder su nombre por el del formulario en que está definido.
Sin embargo, si se desea llamar a un “proc” o “fun” definido en un módulo estándar (*.bas), no es necesario precederlo del nombre del módulo (exceptuando el caso de que coincida con el nombre de otro procedimiento)

Funciones

```
[Static][Private] Function NombreFunción(arg1 As Tipo1, …) As Tipo2
        sentencias
        NombreFunción = expresión
        [Exit Function]
        [sentencias]
        [NombreFunción = expresión]
End Function
```

La llamada a la función: **variable = NombreFunción(arg1 ...)**

Procedimientos “Sub”

```
[Static][Private] Sub NombreProcedimiento (arg1 As Tipo1, …)
        sentencias
        Exit Sub
        Sentencias
End Sub
```

La llamada puede ser:
- Call NombreProcedimiento (argumentos)
- NombreProcedimiento (argumentos)

Argumentos por referencia y por valor

Pasar un argumento **por referencia** en una función o procedimiento Sub, implica que se pasa la variable original, de modo que el procedimiento puede modificar su valor.

Pasar **por valor** implica crear una nueva variable dentro de lafunción y pasarle una copia del valor de la variable externa. Si se modifica el valor de la variable copia, la variable original queda inalterada.

Cuando en la “llamada” se ponen como argumentos, constantes numéricas o expresiones, los valores se pasan por **valor**.

Por defecto los argumentos se pasan **por referencia**. Si expresamos el argumento entre paréntesis en la llamada es un **argumento pasado por valor**.

Ejemplos:

```
Raiz(4)  por valor
Raiz(n)  por referencia
Raiz((n))        por valor
```

Es decir, en este último caso, lo que se "pasa" es una copia de "n",si el procedimiento o función cambia este valor, el cambio afecta sólo a la copia y no a la propia variable "n"

Otra forma de expresar un **argumento por valor**:

```
Function raiz(ByVal n As Double) As Double
…
…
```

Procedimientos Recursivos

Se dice que una función o pocedimiento son **recursivos**, si se llaman a sí mismos.

Ejemplo:

```
Function factorial(n As Integer) As Long
        If n = 0 Then
                factorial = 1
        Else
                factorial = n * factorial(n-1)
        End If
End Function
```

Procedimientos con argumentos opcionales

```
Private Sub x(num As Double, Optional n = 3 As Integer)
        …
        …
        …
End Sub
```

Para llamarla, por ejemplo:

```
x(7 , 5)
x(7)      será igual a x(7 , 3)
```

Número indeterminado de argumentos

```
Public Function maximo(ParamArray numeros())
        For Each x in numeros
                …
                …
                …
                maximo = x
        Next
End Function
```

Los argumentos se especifican en forma de array de dimensión indeterminada **numeros()**
Es necesario la palabra reservada: **ParamArray**

La estructura FOR EACH - NEXT

```
For Each variable In grupo
    …
    …
    …
Next variable
```

Es una construcción similar al bucle **For**, con la diferencia de que la variable no toma valores a partir de un mínimo, sino a partir de los elementos de un array (o colección de objetos).
Esta construcción es muy útil cuando no se sabe el número de elementos que tiene el array o colección.

Arrays

Un array permite referirse a una serie de elementos del mismo tipo con un mismo nombre, y hacemos referencia a un elemento de la serie utilizando uno o más índices.

VB permite definir arrays de una o más dimensiones (hasta 60) y de cualquier tipo de datos

- Arrays estáticos: la dimensión siempre es la misma
- Arrays dinámicos: la dimensión se puede modificar durante la ejecución del programa.

Arrays estáticos

Dim vector(19) As Double	Una dimensión, 20 elementos = vector(0), vector(1), vector(2), ... vector(19)
Dim pepe(3,1 To 6) As Integer	Dos dimensiones, 4x6 = 24 elementos pepe(0,1), pepe(0,2), ... , pepe(3,6)
Public cadena(1 To 2) As String	12 palabras

Arrays Dinámicos
El espacio necesario para un array estático se asigna al iniciarse el programa y permanece fijo durante su ejecución.

```
Dim Matriz() As Integer
…
….
…
ReDim Matriz(f,c)
...
...
```

Es decir redimensionamos el array mediante variables que contienen los valores adecuados y la palabra reservada **ReDim**.

La ayuda inteligente al escribir código

"IntelliSense" es la sofisticada tecnología de Microsoft que nos permite ahorrar trabajo cuando estamos escribiendo código.

Habrás observado que al escribir código, en muchas ocasiones aparecen unos pequeños cuadros con información sobre la orden que estamos escribiendo.

Veamos cómo funciona esta ayuda inteligente. Tiene tres componentes:

1) Información rápida
Siempre que escribimos una palabra reservada, seguida de un espacio o de un paréntesis, aparece una nota en pantalla que contiene la sintáxis del elemento escrito. Un ejemplo sería cuando escribimos **MsgBox**

2) Lista de propiedades y métodos
Cuando escribimos el nombre de un objeto y el punto, aparece un cuadro que contiene todas las propiedades y métodos del objeto en cuestión.
Un ejemplo sería cuando escribimos **Form1** y un punto. Basta seleccionar la propiedad o método de la lista que aparece y pulsar [Tab]

3) Lista de constantes
La tercera posibilidad de Intellisense es que aparece un listado con todas las constantes incorporadas de VB, según el objeto y propiedad.
Lo habrás observado al comenzar a escribir vb...; basta seleccionar una de las contantes y pulsar [Tab]

Si te molesta la “ayuda inteligente”, basta que pulses [Esc] cuando aparece.

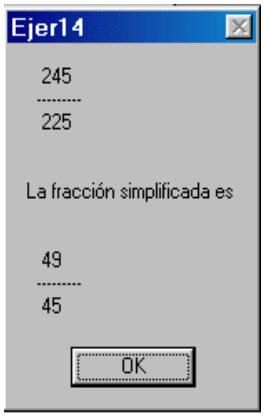
Ejercicios III

1) Haz un programa de nombre **Ejer13** (Ejer13.frm – Ejer13.vbp), que calcule el mínimo común múltiplo de dos números utilizando la función MCD del "Prog27" y sabiendo que **mcm(x,y) = x*y / MCD(x,y)**

2) Haz un programa de nombre **Ejer14**, que sirva para simplificar una fracción numérica, debes utilizar la función MCD del ejercicio anterior.

Observa: $\frac{a}{b}=\frac{a/MCD(a,b)}{b/MCD(a,b)}$

La "salida" deberia ser de la forma:



3) Haz un programa de nombre **Ejer15**, que sirva para sumar o restar dos fracciones y después simplifique el resultado.

Observa: $\frac{a}{b}+\frac{c}{d}=\frac{a(mcm(b,d)/d)+c(mcm(b,d)/d)}{mcm(b,d)}$

La "salida" deberia ser:

4) Haz un programa de nombre **Ejer16**, que sirva para calcular el módulo de un vector en el espacio, utilizando una función.

5) Haz un programa de nombre **Ejer17**, que sirva para calcular el área de un triángulo en el espacio, utilizando la función del ejercicio anterior

Recuerda:

A = (a1,a2,a3), B = (b1,b2,b3), C = (c1,c2,c3)

AB = (b1-a1,b2-a2,b3-a3), AC = (c1-a1,c2-a2,c3-a3)

Área del triángulo ABC: mitad del producto vectorial de AB y AC (consulta el **Ejer10**)

Comprúebalo para el siguiente caso:

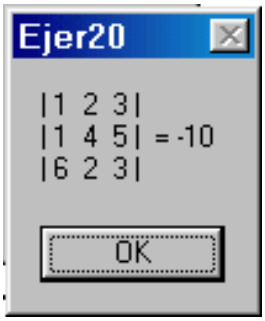
7) Haz un programa de nombre **Ejer19**, tal que:
- El programa nos pregunta cuántas multiplicaciones queremos hacer.
- El programa nos las pregunta aleatoriamente.
- Al final el programa nos da la nota cualitativa.

8) Haz un programa de nombre **Ejer20**, que calcule un determinante de tercer orden.

Recuerda:

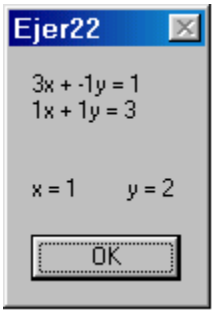
$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = aei + dch + bfg - gec - hfa - dbi$$

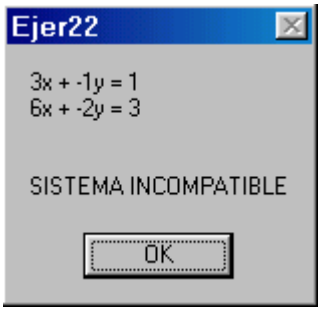
Compruébalo para el caso:



10) Haz un programa de nombre **Ejer22**, que resuelva un sistema de dos ecuaciones con dos incógnitas por el método de Cramer.
Crea una función que calcule un determinante de 2º orden

Compruébalo para los casos:





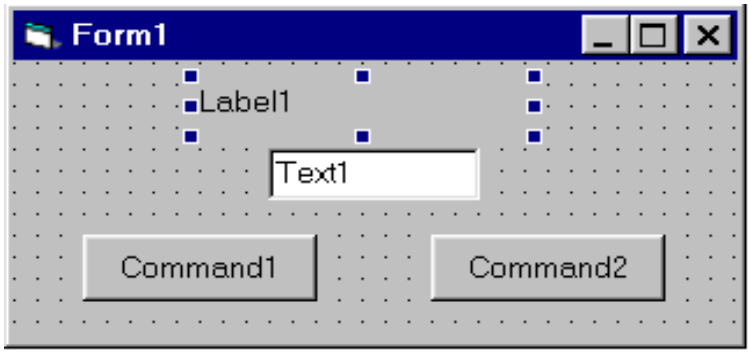
11) Haz un programa de nombre **Ejer23**, que calcule los 50 primeros términos de la sucesión de termino general: (3n + 1) / (2n – 1)

IV.- Label, TextBox, CommandButton

Continuando con el estudio del VB, vamos a ver en los próximos capítulos, los diferentes controles que podemos utilizar en “nuestros programas”.

a) Haz un nuevo proyecto.

- Incluye en el **Form1** los siguientes controles:



- Cambia las siguientes propiedades:

  - **Form1**
    Name: frmClave
    Caption: Contraseña

  - **Label1**
    Name: lblClave
    Caption: Introduce la clave secreta
    Alignment: 2 (Center)

  - **Text1**
    Name: txtContraseña
    MaxLength: 6
    PasswordChar: *
    Text: borra el texto que aparece por defecto

  - **Command1**
    Name: cmdAceptar
    Default: True
    Caption: &Aceptar

  - **Command2**
    Name: cmdCancelar
    Caption: &Cancelar
    Cancel: True

- Escribe los siguientes procedimientos de evento:

```
Private Sub cmdAceptar_Click()
  If UCase(txtContraseña.Text) <> "PEPITO" Then
    MsgBox "No has acertado ", vbCritical
    End
  Else
```

```
        MsgBox "Muy bien, es correcto", vbExclamation
        End
    End If
End Sub


Private Sub cmdCancelar_Click()
    End
End Sub
```

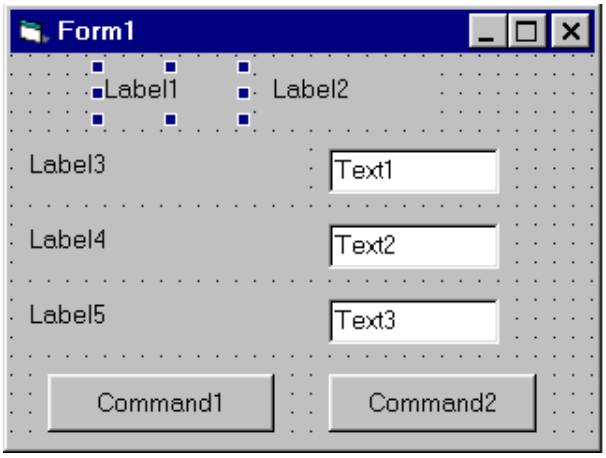
- Graba el formulario con el nombre **Prog31.frm** y el proyecto como **Prog31.vbp**

- Ejecuta unas cuantas veces el programa observando detenidamente lo que sucede.

- Notas:
  - Está claro que la contraseña correcta es **Pepito**
  - La función **Ucase(argumento)**, convierte el argumento en mayúsculas. Gracias a esta función podemos introducir la contraseña en mayúsculas o minúsculas.
  - Propiedad **Alignment: 2 (Center)**, centra el texto en el recuadro.
  - Propiedad **Default del CommandButton**: establece un valor que determina el botón predeterminado, si "Default = True". Si un botón es **Default = True**, automáticamente el resto de botones son "Default = False".
  - Propiedad **Cancel** de un CommandButton: determina el botón cancelar de un formulario. En un formulario sólo puede haber un botón de comando con la propiedad **Cancel = True**.
  - **PasswordChar = *.** La mejor forma de descubrir lo que hace esta propiedad, es escribir un caracter distinto y volver a ejecutar el proyecto.
  - Si como propiedad **Caption** de un CommandButton, escribimos **&Aceptar**, en el botón aparece **Aceptar**. Y en "tiempo de ejecución", es equivalente hacer click en el botón, que pulsar las teclas [ALT][A]. Pruébalo, ejecutando de nuevo el proyecto.

b) Haz un nuevo proyecto.

- Incluye en el **Form1** los siguientes controles:



- Cambia las siguientes propiedades:

  - **Form1**
    Name: frmFechas

Caption: Fechas

- **Label1**
  Name: lblEtiq1
  Caption: Hoy es

- **Label2**
  Name: lblHoy

- **Label3**
  Name: lblEtiq2
  Caption: Primero del mes que viene

- **TextBox1**
  Name: txtPrimeroMesViene

- **Label4**
  Name: lblEtiq3
  Caption: Escribe una fecha

- **TextBox2**
  Name: txtFecha
  Text: borra el texto que aparece por defecto

- **Label5**
  Name: lblEtiq4
  Caption: 1º del Mes Siguiente

- **TextBox3**
  Name: txtSiguiente

- **CommandButton1**
  Name: cmdVale
  Caption: &Vale

- **CommandButton2**
  Name: cmdOtra
  Caption: &Otra

- Escribe el siguiente código:

  - Procedimiento de evento:

```
Private Sub Form_Load()
   lblHoy.Caption = Date
End Sub
```

    Date es una función incorporada de VB que devuelve la fecha del sistema. Es decir, estamos haciendo que al ejecutar el programa (Form_Load), se escriba en el segundo label (lblHoy) automáticamente la fecha de hoy.

  - Crea un **módulo estándar**, y escribe la siguiente **función**:

```
Public Function PrimeroMes()
   PrimeroMes = DateSerial(Year(Now), Month(Now) + 1, 1)
End Function
```

    Antes de continuar escribiendo código, veamos que hará la función **PrimeroMes**

New: es una función incorporada de VB que devuelve la fecha y hora del sistema.
Year (fecha): es otra función incorporada, que devuelve el año correspondiente a la "fecha".
Month(fecha) +1: es otra función incorporada que devuelve el mes (más 1), es decir el mes siguiente a "fecha"
DateSerial(año, mes, día) es otra función incorporada que nos da la fecha correspondiente a "día", "mes" y "año".

En definitiva, la función "PrimeroMes" nos dará la fecha correspondiente al día 1 del mes siguiente a la fecha del sistema.

- En el módulo estándar anterior, escribe la siguiente función:

```
Public Function PrimeroMesCualquiera(Cual As Date) As Date
   PrimeroMesCualquiera = DateSerial(Year(Cual), Month(Cual) + 1, 1)
End Function
```

Si comparas la función con la anterior, llegarás a la conclusión de que **PrimeroMesCualquiera** es una generalización de **PrimeroMes**, en el sentido de que "funciona" con cualquier fecha(Cual).

- Escribe los siguientes procedimientos de evento:

```
Private Sub cmdVale_Click()
   txtSiguiente.Text = PrimeroMesCualquiera(txtFecha.Text)
End Sub

Private Sub cmdOtra_Click()
   txtFecha.Text = ""
   txtSiguiente.Text = ""
End Sub
```

- Corrige el procedimiento **Form_Load()**, de forma que nos quede:

```
Private Sub Form_Load()
   lblHoy.Caption = Date
   txtPrimeroMesViene.Text = PrimeroMes
End Sub
```

- Graba el módulo con el nombre **Prog32.bas**, el formulario como **Prog32.frm** y el proyecto como **Prog32.vbp**

- Sólo queda ejecutar varias veces el programa para ver si funciona.

c) Haz un nuevo proyecto.

Se trata de hacer un programa que calcule el factorial de un número, recuerda que el factorial de un número es el producto **1*2*3*4…** hasta llegar al número. Por ejemplo, el factorial de 5 es **1*2*3*4*5** es decir **120**.
A diferencia de los proyectos anteriores vamos a hacerlo poco a poco…

En primer lugar hemos de tener claro el "código fundamental del programa". Inserta un módulo estándar a nuestro proyecto y escribe el siguiente procedimiento:

```
Public Sub Factorial()
   Dim i As Double
   Dim num As Double
```

```
    Dim factur As Double
    num = InputBox("Escribe el número: ")
    factur = 1
    For i = 1 To num
       factur = factur * i
       Form1.Print factur
    Next
    Form1.Print "FACTORIAL = " & factur
End Sub
```

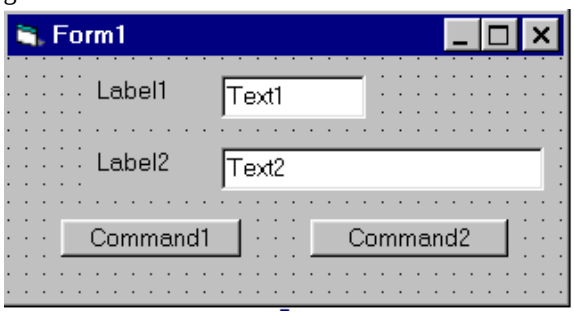
- Vamos a ver si funciona. Escribe el procedimiento de evento:

```
Private Sub Form_Activate()
      Factorial
End Sub
```

Ejecuta varias veces el programa, para asegurarte que funciona.

d) Continuando con el "proyecto anterior", vamos a hacerlo **visual…**

- Coloca en el formulario los siguientes controles:



- Cambia las siguientes propiedades:

Propiedades	Name	Caption	Text
Form1	FrmFactorial	Factorial de un número:	----------
Label1	LblNumero	Número:	----------
Label2	LblFactorial	Factorial:	----------
Text1	TxtNumero	----------	(borra el texto)
Text2	TxtFactorial	----------	(borra el texto)
Command1	CmdCalcular	Calcular	----------
Command2	cmdBorrar	Borrar	----------

- Vamos a "adaptar" el procedimiento **Factorial()** del apartado anterior, para nuestro formulario…

  Escribe los siguientes procedimientos de evento:

```
Private Sub cmdCalcular_Click()
   Dim i As Double, factur As Double
   factur = 1
   For i = 1 To txtNumero.Text
      factur = factur * i
   Next
   txtFactorial.Text = factur
End Sub
```

```
Private Sub cmdBorrar_Click()
   txtNumero.Text = ""
   txtFactorial.Text = ""
End Sub
```

- **Borra** el procedimiento de evento **Form_Activate**

- Graba el formulario con el nombre **Prog33.frm** y el proyecto como **Prog33.vbp**. No es necesario que grabes el módulo.

- Ejecuta y prueba nuestro proyecto.
  Si intentas calcular el factorial de un número mayor de 170, se producirá un error de desbordamiento.

e) Vamos a hacer una serie de "mejoras técnicas"…

- Cuando "borramos" los campos **Número** y **Factorial** nos gustaría que el cursor de escritura se coloque en el campo **"Número"**.

  Nada más fácil. Corrige el procedimiento de evento **cmdBorrar_Click()** de forma que quede:

```
Private Sub cmdBorrar_Click()
   txtNumero.Text = ""
   txtFactorial.Text = ""
   frmFactorial.txtNumero.SetFocus
End Sub
```

- Ejecuta de nuevo nuestro proyecto y prueba el funcionamiento del botón [Borrar].

  SetFocus es un método común a muchos controles, en nuestro caso un **TextBox**, que sirve para colocar el "foco" en el control correspondiente.

- El uso del ratón cuando ejecutamos el proyecto es bastante incómodo, nos gustaría utilizar únicamente el teclado.

  Veamos:

  - Corrige en primer lugar la propiedad **Caption** de "cmdCalcular" y "cmdBorrar" por **&Calcular** y **&Borrar** respectivamente. De esta forma evitamos el uso del ratón para hacer click en dichos botones.
  - Podríamos evitar el uso de [ALT][C] o click en [Calcular] de la siguiente forma:

  Escribe el siguiente procedimiento de evento:

```
Private Sub txtNumero_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then cmdCalcular_Click
End Sub
```

  La cosa se complica ¿verdad?. Veamos que es más fácil de lo que parece:

  KeyPress: evento que consiste en pulsar una tecla.

  KeyAscii: variable entera que corresponde a la tecla que pulsamos (es la tecla que pulsamos en código ASCII). Por ejemplo la tecla [Return], tiene por código ASCII el número 13

txtNumero_KeyPress: al pulsar una tecla cuando estamos en el "textbox" correspondiente al "número".

If KeyAscii = 13 Then cmdCalcular_Click
Si la tecla que pulsamos es [Return] entonces ejecuta el procedimiento "cmdCalcular_Click".

En definitiva: si cuando acabamos de escribir un número (en el TextBox correspondiente), pulsamos [Return], en el campo correspondiente aparecerá el factorial del número introducido.

Pruébalo ejecutando de nuevo nuestro proyecto

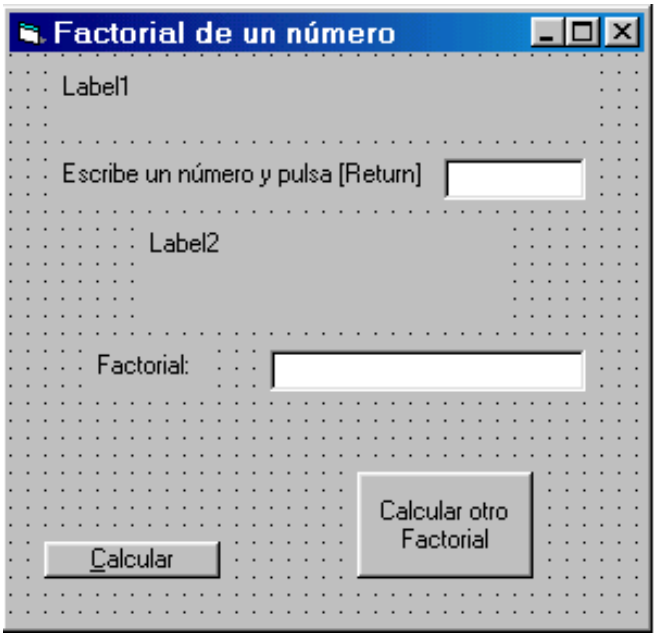
- Graba de nuevo el formulario y el proyecto con los mismos nombres (Prog33)

f) Vamos a mejorar "estéticamente" el programa del "factorial"…

Cambia las siguientes propiedades:

	Caption	**Visible**
lblNumero	Escribe un número y pulsa [Return]	----------
lblFactorial	----------	False
txtFactorial	----------	False
cmdCalcular	----------	False
cmdBorrar	Calcular otro factorial	False

Añade dos **Label** más y cambia la posición y tamaño de los controles de forma que te quede aproximadamente de la siguiente forma:



- Modifica el procedimiento de evento:

```
Private Sub cmdCalcular_Click()
   Dim i As Double, factur As Double
   factur = 1
   For i = 1 To txtNumero.Text
```

```
    factur = factur * i
  Next
  Label2.Visible = True
  Label2.Caption = "El factorial de " & txtNumero.Text & _
    " es el resultado de multiplicar: " & _
    " 1*2*...* " & txtNumero.Text
  lblFactorial.Visible = True
  lblFactorial.Caption = "El factorial de " & txtNumero.Text & _
    " es: "
  txtFactorial.Visible = True
  txtFactorial.Text = factur
  cmdBorrar.Visible = True
End Sub
```

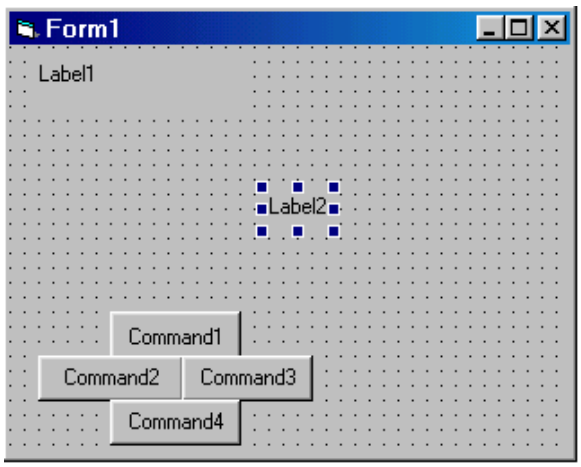
- Cambia las siguientes propiedades:

	Visible	Caption	BackColor	BorderStyle	Font	AutoSize
Label1	---------	FACTORIAL DE UN NÚMERO	Rojo	1-FixedSingle	Arial Negrita 12	True
Label2	False	----------	----------	----------	----------	----------

- Graba el formulario con el nombre **Prog34.frm** y el proyecto como **Prog34.vbp**

g) Aunque la finalidad de un label es el mostrar un texto en modo estático, las posibilidades que nos ofrece el VB nos permiten utilizar un label para otros fines…

- Haz un nuevo proyecto. Inserta en el formulario los siguientes controles:



- Cambia las siguientes propiedades:

	Name	Caption	BackColor
Label1	LblColor	Color primer plano	Rojo
Label2	LblMueve	(borra el texto)	----------
Command1	CmdArriba	Arriba	----------
Command2	CmdIzquierda	Izquierda	----------
Command3	CmdDerecha	Derecha	----------
Command4	cmdAbajo	Abajo	----------

- Escribe los siguientes procedimientos de evento:

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, _
        X As Single, Y As Single)
  lblColor.ForeColor = RGB(Rnd * 256, Rnd * 256, Rnd * 256)
  lblColor.BackColor = RGB(Rnd * 256, Rnd * 256, Rnd * 256)
End Sub

Private Sub cmdArriba_Click()
  lblMueve.Move lblMueve.Left, lblMueve.Top - 30
End Sub

Private Sub cmdAbajo_Click()
  lblMueve.Move lblMueve.Left, lblMueve.Top + 30
End Sub

Private Sub cmdDerecha_Click()
  lblMueve.Move lblMueve.Left + 30
End Sub

Private Sub cmdIzquierda_Click()
  lblMueve.Move lblMueve.Left - 30
End Sub
```

- Graba el formulario con el nombre **Prog35.frm** y el proyecto como **Prog35.vbp**

- Ejecuta y prueba nuestro proyecto.

- Estudiemos el programa:
  - Evento **MouseMove** del **Form** (Form_MouseMove)
    Se produce por el sólo hecho de **mover** el ratón por el formulario. Observa que dicho evento posee una serie de parámetros: Button, Shift, X, Y; argumentos que no utilizamos en nuestro procedimiento.
  - **RGB**(argumento1, argumento2, argumento3)
    Es una función incorporada a VB, que devuelve un color.
    Argumento1 = componente rojo (de 0 a 255)
    Argumento2 = componente verde (de 0 a 255)
    Argumento3 = componente azul (de 0 a 255)
  - **Rnd**
    Es otra función incorporada VB, que devuelve un número aleatorio entre 0 y 1 (distinto de 1)
  - **Rnd*256**
    Devuelve un número aleatorio entre 0 y 255
  - **Move**
    Es un método común a muchos controles que permite **mover** el control
    Objeto.Move argumento1, argumento2, argumento3, argumento4
    Argumento1 = movimiento horizontal (columna)
    Argumento2 = movimiento vertical (línea)
    Argumento3 = anchura del objeto
    Argumento4 = altura del objeto
    De hecho el método Move nos permite fijar la posición y dimensiones del objeto sin necesidad de manipular las propiedades Left, Top, Width y Height.

h) Vamos a ver un ejemplo que muestre las posibilidades de las etiquetas para presentar texto al usuario con muy distintos formatos.

Se trata de una ventana que contiene una etiqueta con borde, color y fuente de tamaño distinto al inicial. Además, al hacer clic sobre ella se intercambian los colores del texto y el fondo.

Crea un nuevo proyecto ...

1. Añadir al formulario por defecto un control de etiqueta y un botón de comando.

2. Cambia las siguientes propiedades:
3.

	Objeto	Propiedad	Valor
Form1	Name	frmEtiqueta	
	Caption	Ejemplo de etiquetas	
Command1	Name	cmdSalir	
	Caption	&Salir	
Label1	Name	lblDemo	
	Caption	VISUAL BASIC	
	BackColor	&H00FF0000& (Azul)	
	ForeColor	&H0000FFFF& (Amarillo)	
	FontSize	18	
	AutoSize	True	
	BorderStyle	1 (Fixed Single)	

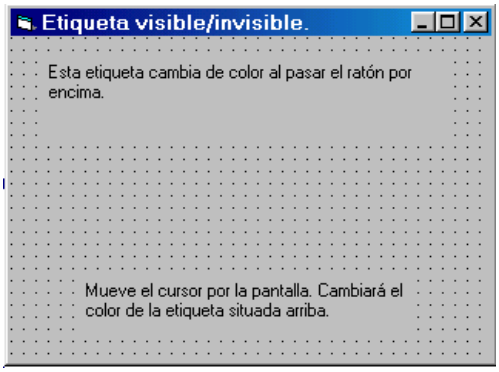
3. Escribe los siguientes procedimientos de evento:

```
Private Sub lblDemo_Click()
  Dim Temporal
  'Intercambiar los colores del texto y el fondo
  Temporal = lblDemo.BackColor
  lblDemo.BackColor = lblDemo.ForeColor
  lblDemo.ForeColor = Temporal
End Sub

Private Sub cmdSalir_Click()
  End
End Sub
```

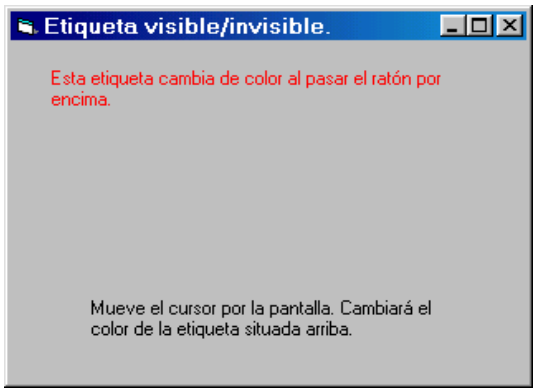
Graba el programa como **Prog36.frm**
Prog36.vbp

Y pruébalo.

i) Crea un nuevo proyecto con dos etiquetas de “caption”:



De forma que al ejecutar el programa resulte de la siguiente forma:



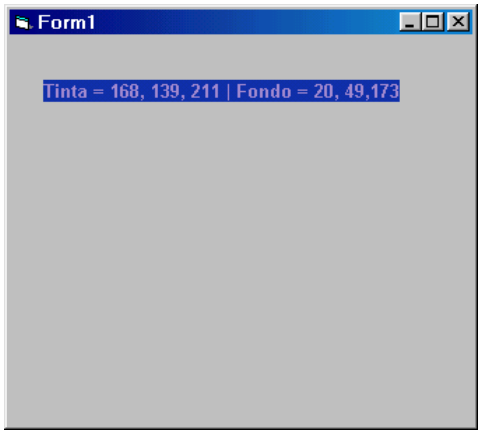
En definitiva deberás escribir los siguientes procedimientos de evento:

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Label1.ForeColor = &H80000018
End Sub

Private Sub Label1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Label1.ForeColor = &HFF&
End Sub
```

Graba el programa como **Prog37.frm, Prog37.vbp**, y ejecútalo para probarlo.

j) Haz un nuevo proyecto con un “label” de caption:

De forma que al mover el ratón por el formulario (Form_MouseMove), la etiqueta cambie de color de forma aleatoria y además indique el código numérico de los colores de fondo y primer plano.
Es decir:

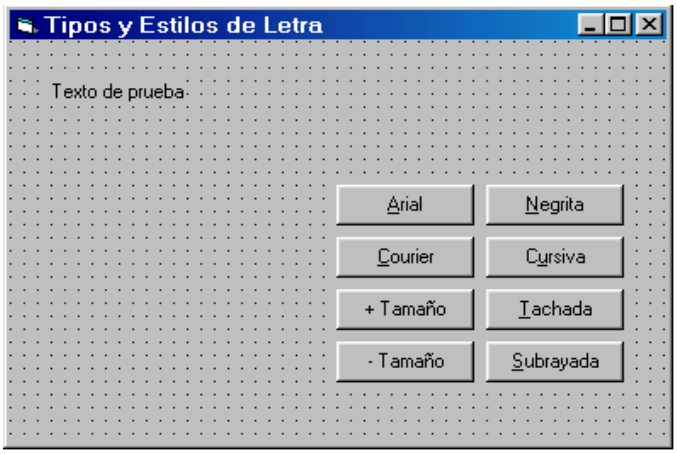


Deberás considerar el código:

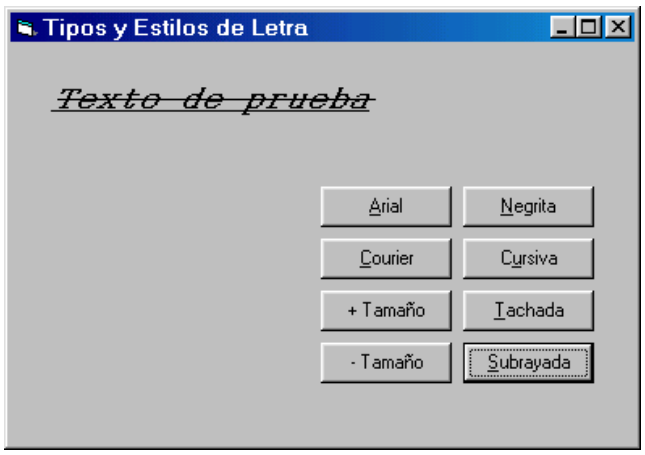
```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Dim RojoT As Byte, VerdeT As Byte, AzulT As Byte
   Dim RojoF As Byte, VerdeF As Byte, AzulF As Byte

   RojoT = Int(Rnd * 256)
   VerdeT = Int(Rnd * 256)
   AzulT = Int(Rnd * 256)
   RojoF = Int(Rnd * 256)
   VerdeF = Int(Rnd * 256)
   AzulF = Int(Rnd * 256)

   EtiquetaColor.ForeColor = RGB(RojoT, VerdeT, AzulT)
   EtiquetaColor.BackColor = RGB(RojoF, VerdeF, AzulF)

   EtiquetaColor.Caption = "Tinta = " & RojoT & ", " & VerdeT & ", " & AzulT
   EtiquetaColor.Caption = EtiquetaColor.Caption & " | Fondo = " & RojoF & ", " & VerdeF &
"," & AzulF
End Sub
```

Graba el programa resultante como: **Prog38.frm, Prog38.vbp**

k) Haz un nuevo proyecto con los siguientes elementos:



De forma que al pulsar en los botones correspondientes, aparezca en el “label”, el o los aspectos de letra considerados, por ejemplo:

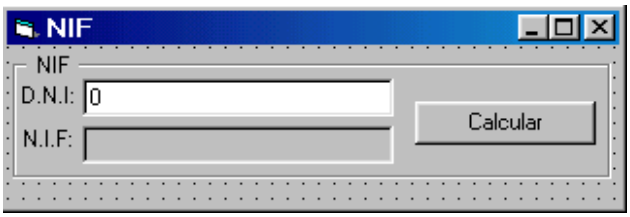


El código que deberás escribir es:

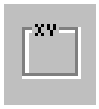
```
Private Sub BotonArial_Click()
   Texto.FontName = "Arial"
End Sub

Private Sub BotonAumentarTamaño_Click()
   Texto.FontSize = Texto.FontSize + 4
End Sub

Private Sub BotonCourier_Click()
   Texto.FontName = "Courier"
End Sub

Private Sub BotonCursiva_Click()
   Texto.FontItalic = Not Texto.FontItalic
End Sub

Private Sub BotonNegrita_Click()
   Texto.FontBold = Not Texto.FontBold
End Sub

Private Sub BotonReducirTamaño_Click()
   Texto.FontSize = Texto.FontSize - 4
End Sub

Private Sub BotonSubrayada_Click()
   Texto.FontUnderline = Not Texto.FontUnderline
End Sub

Private Sub BotonTachada_Click()
   Texto.FontStrikethru = Not Texto.FontStrikethru
End Sub
```

Graba el programa como **Prog39.frm, Prog39.vbp** y pruébalo.

l) Haz un nuevo proyecto con el siguiente contenido:

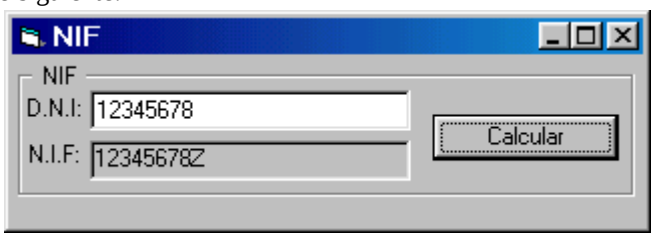


El recuadro “NIF”, debes hacerlo con el control **Frame**:



Se dice de este control que es un contenedor, porque “contiene otros controles”, se utiliza para embellezer y organizar los diferentes elementos en el “form”;

Queremos conseguir lo siguiente:



Es decir, a partir del DNI, queremos calcular el NIF.

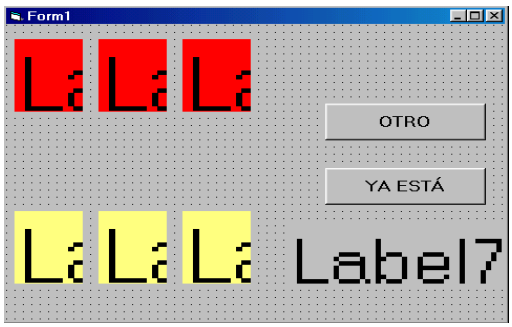
Deberás escribir el siguiente código:

```
Public Function Nif(Dni As Long) As String
Nif = Dni & Right(Left("TRWAGMYFPDXBNJZSQVHLCKEO", ((Dni Mod 23) + 1)), 1)
End Function

Private Sub Command1_Click()
On Error Resume Next
Select Case Len(Text1)
Case Is < 8
MsgBox "Debe Introducir 8 caracteres", 16, "Error"
Case Else
Label3 = Nif(Text1)
End Select
End Sub


Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
Select Case KeyCode
Case 13
Command1_Click
End Select
End Sub
```

Graba el programa como **Prog40.frm, Prog40.vbp**, y pruébalo.

m) Nuevo Proyecto

- Inserta:
Tres labels donde tendremos 3 números aleatorios
Tres labels donde pondremos **(arrastrándolos con el ratón)** los tres números anteriores ordenados de menor a mayor
Otro Label donde aparecerá “Bien” o “Mal”, según si el orden correspondiente es correcto o no.
Dos Botones



Propiedades: los tres primeros “labels”, **DragMode = 1**

- Código:

```
Dim n1 As Integer
Dim n2 As Integer
Dim n3 As Integer
Private Sub azar()
   n1 = Int(Rnd * 10) + 1
   n2 = Int(Rnd * 10) + 1
   n3 = Int(Rnd * 10) + 1
   If n1 <> n2 And n2 <> n3 And n1 <> n3 Then
      Label1.Caption = n1
      Label2.Caption = n2
      Label3.Caption = n3
   Else
      Exit Sub
   End If
End Sub

Private Sub Command1_Click()
   Label4.Caption = ""
   Label5.Caption = ""
   Label6.Caption = ""
   Label7.Caption = ""
   Label1.Visible = True
   Label2.Visible = True
   Label3.Visible = True
   Call azar
End Sub

Private Sub Command2_Click()
   If Val(Label4.Caption) < Val(Label5.Caption) And Val(Label5.Caption) < Val(Label6.Caption)
Then
      Label7.Caption = "Bien"
   Else
      Label7.Caption = "Mal"
   End If
End Sub
```

```
Private Sub Form_Load()
   Label4.Caption = ""
   Label5.Caption = ""
   Label6.Caption = ""
   Label7.Caption = ""
   Randomize
   Call azar
End Sub

Private Sub Label4_DragDrop(Source As Control, X As Single, Y As Single)
   Label4.Caption = Source
   Source.Visible = False
End Sub

Private Sub Label5_DragDrop(Source As Control, X As Single, Y As Single)
   Label5.Caption = Source
   Source.Visible = False
End Sub
Private Sub Label6_DragDrop(Source As Control, X As Single, Y As Single)
   Label6.Caption = Source
   Source.Visible = False
End Sub
```

- Grábalo como **Prog41.frm, Prog41.vbp** y pruébalo,es decir **arrastra con el ratón** (evento DragDrop), los números que aparecen en los tres “labels” superiores en los tres labels inferiores, de forma que los tres números esten ordenados de menor a mayor. Y no hagas como yo en el siguiente ejemplo:

n) Crea un **nuevo proyecto** con los siguientes objetos y propiedades:

- Form1:
  Caption: LOTERÍA
- Label1:
  Caption: Escribe un número:
- Text1:
  Text: (nada)
- Text2:
  Text: (nada)
  Enabled: False
- Label2:
  Caption: Ver el elemento nº:
- Text3:
  Text: (nada)
- Text4:

Text: (nada)
Enabled: False
Font: Tamaño= 24

Pretendemos simular un juego de lotería, que funciona de la siguiente forma: el programa sortea 6 números del 1 al 49 y nosotros hemos de adivinar uno de éstos números.

- El primer problema es tener guardados los 6 números del sorteo…

  - Accede a la ventana de código y
    Objeto: **(General)**
    Procedimiento: **(declaraciones)**

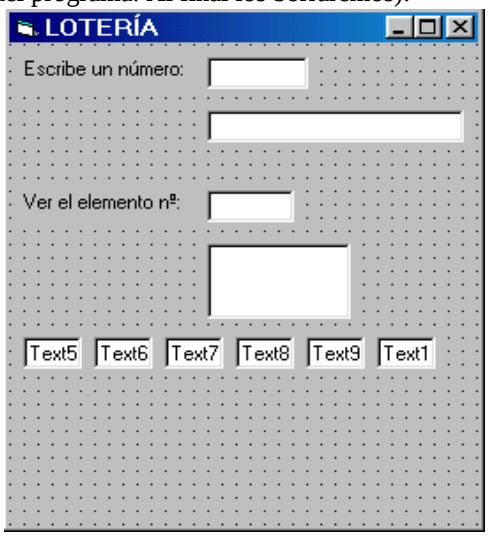
    Escribe:

    ```
    Dim Lotería(5) As Integer
    ```

    Es decir: declaramos una variable matricial (Lotería) de una dimensión, con 6 valores: Lotería(0), Lotería(1), …, Lotería(5).

- Antes de continuar, coloca en el formulario **6** TextBox, donde aparecerán los 6 números del sorteo (es simplemente para comprobar el funcionamiento del programa. Al final los borraremos):

Tendremos aproximadamente:



Accede a la ventana de código con las opciones:
Objeto: **Form**
Procedimiento: **Load**
Y escribe el siguiente procedimiento:

```
Private Sub Form_Load()
   Dim i, j As Integer
   Dim NúmeroSorteado As Integer
   Dim Trabajar, HaSalido As Integer
   Randomize
   For i = 0 To 5
      Trabajar = True
      Do While Trabajar
         NúmeroSorteado = Int(Rnd * 49) + 1
         HaSalido = False
         For j = 0 To i
            If NúmeroSorteado = Lotería(j) Then HaSalido = True
         Next j
         If Not HaSalido Then
```

```
            Lotería(i) = NúmeroSorteado
            Trabajar = False
          End If
        Loop
    Next i
    Text5.Text = Lotería(0)
    Text6.Text = Lotería(1)
    Text7.Text = Lotería(2)
    Text8.Text = Lotería(3)
    Text9.Text = Lotería(4)
    Text10.Text = Lotería(5)
End Sub
```

Vamos a intentar entender el funcionamiento del programa:

- Definimos 5 variables:

  - **i**: es el índice de un ciclo **For-To-Next** de i= 0 hasta i=5 a partir del cual guardaremos en **Lotería(i)** los números sorteados.

  - **j**: es el índice de otro ciclo **For-To-Next** de j=0 hasta j=i, que servirá para investigar si algún valor de "Lotería" está repetido.

  - **NúmeroSorteado**: Representa el número aleatorio de 1 hasta 49 (NúmeroSorteado = Int(Rnd*49) + 1

  - **HaSalido:** Controlará si el número sorteado ya ha salido

  - **Trabajar**: Controlará el ciclo **Do While – Loop**

-

```
For i=0 To 5
…….
…….
…….
Next i
```

Es decir: Para i=0 hasta i=5

```
Trabajar=True
Do While Trabajar
……..
……..
……..
Loop
```

  - NúmeroSorteado = número aleatorio entre 1 y 49

  - Si el valor de "NúmeroSorteado" es igual a algún valor de **Lotería()** entonces **Ha Salido = True** y **NúmeroSorteado** vuelve a tomar un valor aleatorio entre 1 y 49

  - Si el valor de "NúmeroSorteado" no ha salido, el valor de **Lotería()** es igual a dicho número y salimos del **Do While – Loop** (Trabajar = False).

- Por último: en los 6 últimos "TextBox" del formulario aparecen los 6 valores de la lotería.

- Ejecuta el programa varias veces para comprobar que los 6 valores de la lotería son distintos.

Al escribir un número del 1 al 6, en el cuadro de texto **"Text3"**, nos interesa que aparezca en el **"Text4"** el número correspondiente de la **Lotería**…

- Accede a la ventana de código con las opciones:
  Objeto: **Text3**
  Procedimiento: **Change**

  Y escribe el siguiente procedimiento:

```
Private Sub Text3_Change()
   Dim i, Elemento As Integer
   I = Val (Text3)
   If i>0 And i<7 Then
            Elemento = Lotería(i-1)
            Text4 = Str(Elemento)
   End If
End Sub
```

- Observa:

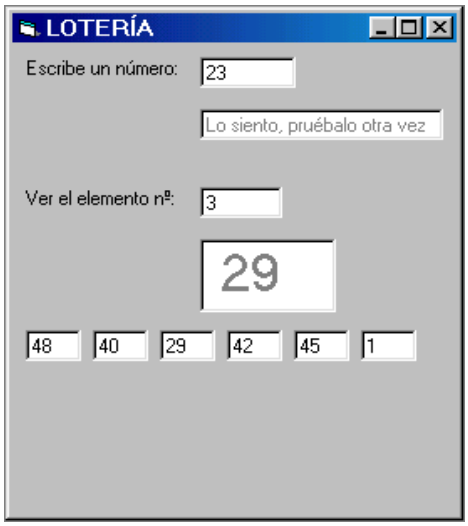
  - La variable "i" es el número que escribimos en el **Text3** (i = Val(Text3))

  - La variable **"Elemento"** es el valor correspondiente del vector **Lotería()**:
    Si i=1 entonces Elemento= Lotería(i-1)= Loteria(0)
    Si i=2 entonces Elemento= Lotería(i-1)= Loteria(1)
    Etc.

  - Dicho valor de "Elemento" se escribe en el **Text4**: Text4 = Str(Elemento)

- Ejecuta el programa y escribe en el campo **"Ver el elemento nº:"**, números del 1 al 6.

Nos interesa que al escribir un número (del 1 al 49), en el cuadro de texto **"Text1"** y pulsar la tecla [Return], aparezca en el cuadro de texto **"Text2"** un mensaje indicando si lo hemos adivinado o no….



- Accede a la ventana de código con las opciones:
  Objeto: **Text1**
  Procedimiento: **KeyDown**

  Y escribe el siguiente procedimiento:

```
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
   Dim i, Adivinado As Integer
   If KeyCode = 13 Then
      Adivinado = False
      For i = 1 To 6
         If Lotería(i - 1) = Val(Text1) Then Adivinado = True
```

```
      Next i
      If Adivinado Then
         Text2 = "Muy Bien"
         Else
         Text2 = "Lo siento, pruébalo otra vez"
      End If
   End If
End Sub
```

Es decir:

Si pulsamos la tecla [Return] en el **Text1** (KeyCode = 13), entonces el número que tenemos escrito en el **Text1**, se busca en el vector **Lotería()**

- Si lo encuentra entonces **Adivinado = True** y por lo tanto aparece en el **Text2** el mensaje "Muy Bien"
- Si no lo encuentra (Adivinado = False), aparece en el **Text2** el mensaje: "Lo siento, pruébalo otra vez"

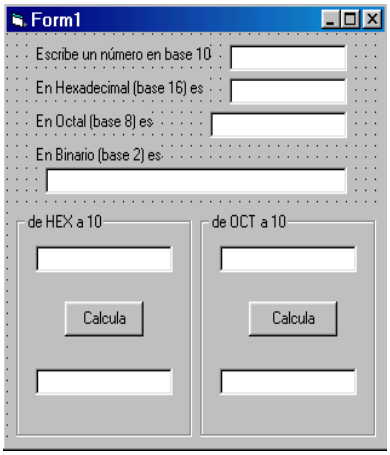
- Ejecuta el programa y pruébalo exhaustivamente (espero que te funcione correctamente).

- Elimina los últimos **6 TextBox** (selecciónalos y pulsa [Supr])

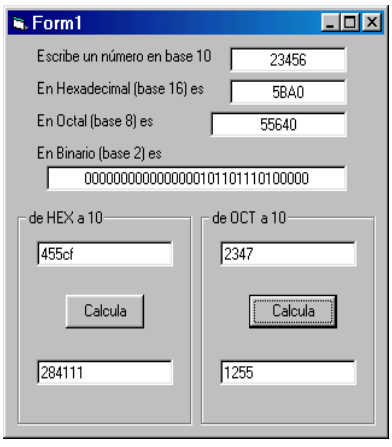
- Graba el formulario con el nombre **Prog42.frm** y el proyecto con el nombre **Prog42.vbp** en *TuCarpeta*

o) Vamos a hacer el último proyecto de este capítulo, se trata de trabajar con las funciones **Oct()** y **Hex()**, que nos permiten trabajar en base 8 (octal) y 16 (hexadecimal)

Haz un nuevo proyecto con el siguiente contenido (observa que tenemos dos **frames**):



En funcionamiento:



Escribe el siguiente código:

```
Dim HexNum
Dim num As Integer
Dim num1 As Integer

Private Sub Command1_Click()
   Text6.Text = CLng("&H" & Text5.Text)
End Sub

Private Sub Command2_Click()
   Text8.Text = CLng("&O" & Text7.Text)

End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
   HexNum = Text1.Text
'    num = Hex(HexNum)
'    Text2.Text = num
   HexNum = Text1.Text
'    num1 = Oct(HexNum)
'    Text3.Text = num1
   Dim I As Long, J As Long, K As Long, A As Long, H As Long
   Dim Cadena
   On Error Resume Next
   A = Text1.Text
      For J = 30 To 0 Step -1
         If A And 2 ^ J Then
            H = 1
         Else
            H = 0
         End If
      Cadena = Cadena & H
      Next
   Text4.Text = Cadena
   '
'    HexNum = Text2.Text
'    num = CLng("&H" & Text1.Text)
   Text2.Text = Hex(Text1.Text)
   '
  ' HexNum = Text3.Text
  ' num = CLng("&o" & Text1.Text)
   Text3.Text = Oct(Text1.Text)
End If
End Sub
```

Graba el programa como **Prog43.frm, Prog43.vbp** y pruébalo.

Para saber más IV

Control "Label"

La finalidad de un control "label" es situar un texto estático en el interior de un formulario. Que sea un texto estático, quiere decir que el usuario del programa no podrá interactuar con él.

Principales Propiedades de un Label

Name:	nombre del control
Caption:	texto estático que aparece
Left:	posición horizontal
Top:	posición vertical
Width:	anchura
Height:	altura
Alignment:	alineación del texto
AutoSize:	modificación automática del control, al tamaño del texto
WordWrap	distribución del texto en varias líneas
Font	tipo, aspecto y tamaño de letra.
BackColor	color de fondo
ForeColor	color de la letra
BackStyle	estilo del fondo (transparente u opaco.
BorderStyle	estilo del borde alrededor del control.
Enabled	activa o desactiva el control.
Visible	hace visible o no el control.

Control "TextBox"

La finalidad de un **TextBox** es solicitar la entrada de datos en un formulario.
Muchas de las propiedades ya conocidas para un **Label** son comunes en el **TextBox**, veamos algunas própias del cuadro de texto:

- Propiedad **Text**
  El TextBox no tiene propiedad "Caption", su equivalente es **Text**, que nos permite dar un valor inicial durante el diseño. Inicialmente el tamaño máximo del texto que podemos introducir es de 2 Kb.

- Propiedad **MaxLength**
  Permite asignar el número límite de caracteres. El valor 0 indica que no hay límite

- Propiedad **ReadOnly**
  Por defecto es "false". Si lo ponemos a "True", el texto que contiene el TextBox no podrá modificarse en tiempo de ejecución

- Propiedad **MultiLine**
  Permite activar la característica de múltiples líneas en una caja de texto.
  Si activamos la propiedad MultiLine (True), automáticamente el tamaño máximo del texto aumenta hasta 32 Kb.

- Propiedad **ScrollBars**
  Por defecto tiene el valor 0. Puede tomar los valores vbHorizontal, vbVertical o vbBoth, que determina una barra de desplazamiento horizontal, vertical o ambas.

- Propiedad **TabIndex**

Cuando en un formulario existen varios controles no estáticos, por ejemplo varios TextBox, para movernos de uno a otro determinado podemos utilizar la propiedad **TabIndex**.
Por defecto el número de orden que se otorga a cada control se va asignando a medida que son creados en la propiedad **TabIndex**, empezando por el valor 0.
Si deseamos cambiar el orden en que se moverá el cursor de un control a otro, bastará cambiar la propiedad **TabIndex**

Control CommandButton

Es un control que aparece como un rectángulo o cuadrado con un título en su interior, al pulsarlo el botón parece hundirse, y se ejecuta el código asociado a su evento **Click**

Propiedad **TabStop**
Normalmente un control no estático (CommandButton o TextBox), puede estar activo en un momento determinado; debido a que por defecto la propiedad **TabStop** tiene el valor "True". Si asignamos "False" a la propiedad "TabStop", la única forma de activar ese control será mediante la selección con el ratón o mediante su tecla rápida.

Propiedades **Default** y **Cancel**
Podemos activar un botón de otra forma:

- Al pulsar [Return] si el botón es por **defecto**
- Al pulsar [Esc] si es el botón de **cancelación**

Sólo pude existir un botón por defecto y un botón de cancelación en un formulario.
Para que un botón sea el botón por defecto, daremos el valor **True** a la **propiedad Default**
Para que un botón sea el de cancelación asignaremos el valor **True** a la **propiedad Cancel**.
Por defecto ambas propiedades, tienen el valor **False**.

Ejercicios IV

1) Repite el ejercico **Ejer14**, que servia para simplificar fracciones, pero de forma “visual”, es decir con controles (label, textbox, commandbutton) y colores.
Por ejemplo:

Grábalo como **Ejer24**

2) Repite el **Ejer15**, programa que servia para sumar dos fracciones, pero hazlo de “forma visual”.
Grábalo como **Ejer25**

3) Repite el **Ejer17**, programa que servia para calcular el área de un triángulo en el espacio, pero hazlo de “forma visual”.
Grábalo como **Ejer26**

4) Repite el **Ejer19**, programa que nos preguntaba productos aleatorios, pero hazlo de “forma visual”.
Grábalo como **Ejer27**

5) Repite el **Ejer20**, programa que servia para calcular un determinante de 3r. orden, pero hazlo de “forma visual”.
Grábalo como **Ejer28**

6) Repite el **Ejer21**, programa que servia para discutir y resolver un sistema lineal de 3 ecuaciones con 3 incógnitas, pero hazlo de “forma visual”.
Grábalo como **Ejer29**

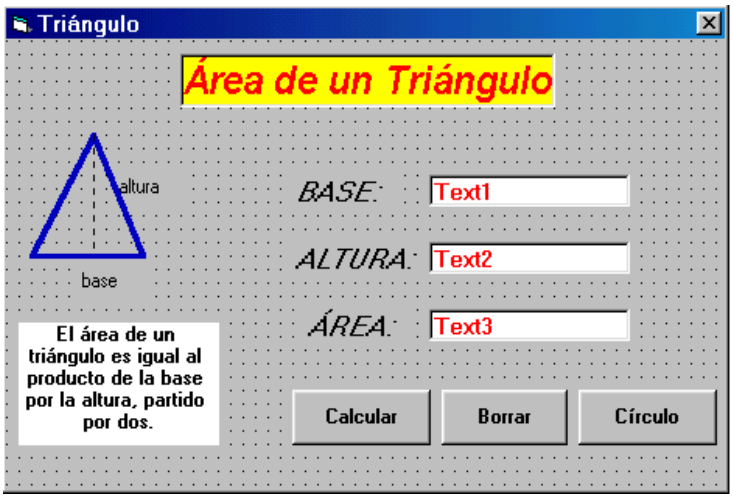
7) Haz un programa de nombre **Ejer30** que resuelva una ecuación de grado 2, pero de forma “visual”.

V.- Proyectos con más de un formulario. Eventos

a) Haz un nuevo proyecto

- Accede a la "Ventana de propiedades del Form1" y cambia las siguientes propiedades:
  - Caption: **TRIÁNGULO**
  - Nama: **AREA1**

Queremos conseguir:



El proceso a seguir es el siguiente:

- Observa que el label superior (label1) contiene un título "guapo", haz lo siguiente:
  - Accede a la ventana de propiedades del **Label1** y cambia las siguientes propiedades:
  - Caption: **ÁREA DE UN TRIÁNGULO**
  - Font: haz CLIC en el botón **[...]** y selecciona:
    Fuente: **Arial**
    Estilo de fuente: **Negrita Cursiva**
    Tamaño: **18**
  - ForeColor: selecciona el color **rojo**
  - BorderStyle: **1-Fixed Single**
  - BackColor: selecciona el color **amarillo**
  - AutoSize: **True**

- Utilizando el icono **"Line"**: 
del "Cuadro de Controles"

  Dibuja el triángulo con las siguientes características:

  - Una línea: **Line1** con las propiedades:
    BorderColor: **azul**
    BorderWidth: **3**
  - Otra línea: **Line2**, de propiedades:
    BorderColor: **azul**
    BorderWidth: **3**
  - Otra línea: **Line3** de propiedades:
    BorderColor: **azul**
    BorderWidth: **3**
  - Otra línea: **Line4** (representará la altura del triángulo), de propiedades:
    BorderStyle: **3-Dot**

- Coloca en el form: **TRIÁNGULO**, tres “labels” con las siguientes características:
  - Label2:
    Caption: base
  - Label3:
    Caption: altura
  - Label4:
    Caption: El Área de un Triángulo es igual al producto de la base por la altura, partido por dos.
    Alignment: 2 – Center
    Appearance: 0 – Flat
    Font: MS Sans Serif
    Negrita
    Tamaño: 10

- Coloca en el “form” 3 labels y 3 textboxs, con las siguientes características:
  - Label5:
    Caption: BASE:
    Font: MS Sans Serif
    Negrita cursiva
    Tamaño: 12
  - Label6:
    Caption: ALTURA:
    Font: MS Sans Serif
    Negrita
    Tamaño: 12
  - Label7:
    Caption: ÁREA:
    Font: MS Sans Serif
    Negrita cursiva
    Tamaño: 12
  - TextBox: Text1
    Font: MS Sans Serif – Negrita – Tamaño: 10
    ForeColor: rojo
    Locked: True
    La propiedad anterior, evitará poder escribir en el cuadro de texto, en tiempo de ejecución.
  - TextBox: Text2
    Font: MS Sans Serif
    Negrita
    Tamaño: 10
    ForeColor: rojo
    Locked: True
  - TextBox: Text3
    Font: MS Sans Serif
    Negrita
    Tamaño: 10
    ForeColor: rojo
    Locked: True

- Coloca en el form, tres botones de comando con las siguientes características:
  - CommandButton: Command1
    Caption: Calcular
    Font: Negrita
  - CommandButton: Command2

Caption: Borrar
Font: Negrita

- CommandButton: Command3
  Caption: Círculo
  Font: Negrita

- Accede a las propiedades del form: **AREA1** y cambia la siguiente propiedad:
  BorderStyle: **1 – Fixed Single**
  El valor **1 – Fixed Single** para el "BorderStyle", evitará que podamos redimensionar el **form**, en tiempo de ejecución.

- Graba el form **AREA1**, en *TuCarpeta* con el nombre **Prog44a.frm**:

b) Vamos a escribir el "código" para el form **AREA1**
Escribe:

```
Private Sub Command1_Click
    Dim BAS, ALT, ARE As Double
    BAS = InputBox ("", "Base del Triángulo")
    Text1.Text = BAS
    ALT = InputBox ("", "Altura del Triángulo")
    Text2.Text = ALT
    ARE = BAS * ALT / 2
    Text3.Text = ARE
End Sub
```

```
Private Sub Command2_Click
    Text1.Text = ""
    Text2.Text = ""
    Text3.Text = ""
End Sub
```

- Cierra la ventana de código

- Graba de nuevo el form: **Prog44a**, con el mismo nombre

- Ejecuta el form: **Prog44a**:
  - CLIC en el icono **"Iniciar"**
  - CLIC en [Calcular]
  - Escribe: 57,8 y [Aceptar]
  - Escribe: 22,6 y [Aceptar]
  - Si todo funciona correctamente en el form, tendremos:
    BASE: 57,8
    ALTURA: 22,6
    ÁREA: 653,14
  - CLIC en [Borrar]
  - CLIC en el icono **"Terminar"**

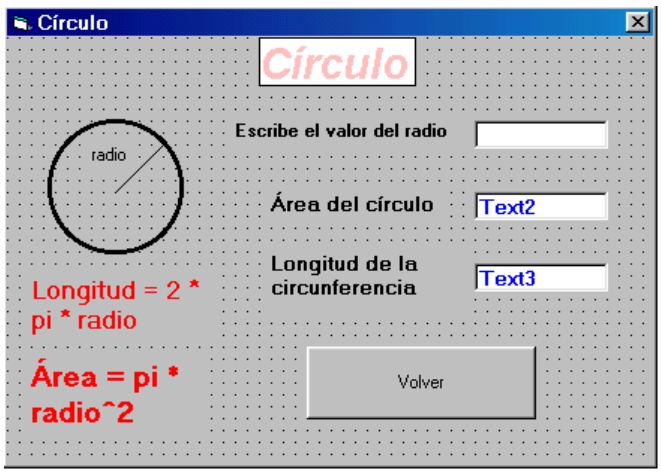
c) Vamos a hacer otro form, en el mismo proyecto anterior...

- Con el form**: Prog44a** a la vista, haz lo siguiente:
  - Menú Project
    Add Form
    [Open]

  - Cambia las siguientes características del nuevo **form**:
    Caption: CÍRCULO
    Name: AREA2
    BorderStyle: 1 – Fixed Single

- Graba el form **AREA2** en *TuCarpeta*, con el nombre **Prog44b.frm**, y el proyecto con el nombre **Prog44.vbp**

Queremos conseguir en este segundo formulario:



El proceso que debes seguir es el siguiente:

- Coloca en el **form: Prog44b** un label, será el **Label1**, con las siguientes propiedades:
  - Caption: CÍRCULO
  - Font:
    Fuente: Arial
    Estilo de fuente: Negrita Cursiva
    Tamaño: 22
  - Appearance: 0-Flat
  - BorderStyle: 1 Fixed Single
  - ForeColor: selecciona el color rosa.
  - BackColor: azul claro

- Dibuja en el **form: AREA2**, un círculo de la siguiente forma:

  - CLIC en el icono **"Shape"**: del "Cuadro de Controles" 

  - "Dibuja" un rectángulo
  - Establece las siguientes propiedades para el objeto **"Shape1"**:
    Shape: 3-Circle
    BorderWidth: 3

- Dibuja en el form: AREA2 una línea (CLIC en el icono “Line”) representará el radio del círculo, con las siguientes propiedades:
  BorderStyle: 3-Dot

- Coloca en el form: AREA2 los siguientes LABELS:
  - Label2
    Caption: radio
  - Label3:
    Caption: Longitud = 2 * pi * radio
    Font: Negrita
    Tamaño: 12
    Forecolor: rojo
  - Label4:
    Caption: Área = pi * radio^2
    Font: Negrita
    Tamaño: 14
    Forecolor: rojo
  - Label5:
    Caption: Escribe el valor del radio:
    Font: Negrita
  - Label6:
    Caption: Área del Círculo:
    Font: Negrita
    Tamaño: 10
  - Label7:
    Caption: Longitud de la Circunferéncia:
    Font: Negrita
    Tamaño: 10

- Coloca en el form: AREA2 los siguientes TextBoxs:
  - Text1:
    Font: Negrita
    Text: (botta el texto que aparece por defecto)
  - Text2:
    Locked: True
    TabStop: False
    Las dos propiedades anteriores nos permitirán: no modificar el valor del **Text2** y al pulsar [Return] o [Tab] en el control anterior, el cursor no se situará en el **Text2**.
    Font: Negrita
    Tamaño: 10
    ForeColor: azul
  - Text3:
    Locked: True
    TabStop: False
    Font: Negrita
    Tamaño: 10
    ForeColor: azul

- Coloca en el form: AREA2 un CommandButton, con las siguientes características:
  Command1:
  Caption: VOLVER

d) Vamos a escribir el código del form: AREA2...

- Escribe el siguiente procedimiento:

```
Private Sub Text1_Change()
      If Text1.Text <> "" Then
            Text2.Text = 3.141592 * Text1.Text ^2
            Text3.Text = 2 * 3.141592 * Text1.Text
      Else
            Text2.Text= ""
            Text3.Text = ""
      End If
End Sub
```

Es decir:

"Al cambiar el valor que hay escrito en el **Text1**, en el **Text2** aparecerá el valor del área del círculo y en el **Text3** aparecerá el valor de la longitud de la circunferéncia".
Hemos de excluir la posibilidad: **Text1.Text = ""**, ya que en caso contrario nos daría un error.

- Graba el form: **Prog44b** con el mismo nombre

e) Vamos a agrupar los dos formularios **AREA1** y **AREA2** en un único programa:
El proceso a seguir es el siguiente:

- En primer lugar hemos de establecer cuál debe ser el **formulario principal**, es decir el formulario que se ejecutará al ejecutarse el programa, de la siguiente forma:
  Menú Project
  Project1 Properties
  - Selecciona sino lo está ya, la pestaña "General"
  - En el campo **"Startup Object"**, selecciona sino lo está ya, **AREA1**
  - [Aceptar]
- Acabamos de establecer que al ejecutarse el programa, se abrirá automáticamente el form: **AREA1**.

Para poder trabajar en un momento determinado con el otro form, es necesario que esté en memória, por esta razón necesitamos hacer lo siguiente:

- Escribe el procedimiento siguiente, en el **módulo del AREA1**:

```
Private Sub Form_Load()
   Load AREA2
End Sub
```

Es decir, al ejecutarse el formulario principal (AREA1), se leerá el formulario AREA2, es decir estará en memoria para cuando lo necesitemos.

- Con el form: AREA1 a la vista
  - Accede al módulo de AREA1, y
  - Escribe:

```
Private Sub Command3_Click()
    AREA2.Show
End Sub
```

Observa de qué forma "visualizaremos" el otro formulario.

- Graba el form: **AREA1** con el mismo nombre **Prog44a**

- Selecciona el form: **AREA2** y

- Accede a su ventana de código
- Escribe:

```
Private Sub Command1_Click()
    AREA1.Show
End Sub
```

- Graba el form: **AREA2** con el mismo nombre **Prog44b**

f) Bien, el programa en "principio" está acabado. Vamos a probarlo:
- CLIC en el icono **"Iniciar"**
- "Juega" con todos los botones
- Espero que te funcione correctamente.

Para acabar grábalocon el mismo nombre: **Prog44a.frm**
Prog44b.frm
Prog44.vbp

Un **evento** no es más que una determinada "situación" que se produce al ejecutarse un programa. Por ejemplo:

Form_Load: al ejecutarse un formulario
Command1_Click: al hacer CLIC en un botón
Timer1_Timer: al transcurrir un cierto tiempo
Text1_Change: al cambiar el contenido de un TextBox.

Un "evento" está siempre asociado a un objeto de **VB**, por ejemplo el evento **Click**, puede estar asociado a un botón (Command_Click) o a un formulario (Form_Click).

Al acceder a la **ventana de código** de un programa, seleccionamos en el campo **Objeto**, el objeto: Form, Command1, Text2, etc y en el campo **Procedimiento**, el evento: Click, Change, etc. Automáticamente se genera en la ventana de código, el esqueleto del **procedimiento de evento** correspondiente:

```
Private Sub Form_Load()

End Sub
```

```
Private Sub Command1_Click()

End Sub
```

Etc, etc.

Vamos a trabajar, con diferentes **eventos**, unos ya conocidos y otros desconocidos.

g) Crea un nuevo proyecto
- Coloca en el form un control **"Label"** con las propiedades:
  Caption: **Eventos Click y DblClick**
  BorderStyle: **1 – Fixed Single**

- Escribe en su módulo los siguientes procedimientos:

```
Private Sub Form_Click()
        Print "Has hecho CLIC en el formulario"
End Sub
```

```
Private Sub Form_DblClick()
        Print "Has hecho CLIC-CLIC en el formulario"
End Sub
```

```
Private Sub Label1_Click()
        Print "Has hecho CLIC en la etiqueta"
End Sub
```

```
Private Sub Label1_DblClick()
        Print "Has hecho CLIC-CLIC en la etiqueta"
End Sub
```

- Prueba el programa, es decir:
  - CLIC en el icono **"Iniciar"**
  - Haz CLIC's y Doble CLIC's en la etiqueta y en el formulario.

- Graba el programa como **Prog45.frm, Prog45.vbp**

h) Agrega al formulario anterior 3 botones de comando y un control de etiqueta con las siguientes características:

- Form1:
  Caption: **GotFocus = Al recibir el FOCO**
- Label1:
  Alignment: **2 – Center**

- Activa la **ventana de código** y escribe los siguientes procedimientos de evento:

```
Private Sub Command1_GotFocus()
 Label1.Caption = "El Command1 tiene el foco"
End Sub
```

```
Private Sub Command2_GotFocus()
 Label1.Caption = "El Command2 tiene el foco"
End Sub
```

```
Private Sub Command3_GotFocus()
 Label1.Caption = "El Command3 tiene el foco"
End Sub
```

- Ejecuta el programa y prueba el funcionamiento del evento **GotFocus**, pulsando repetidamente la tecla **[Tab]** o haciendo CLIC con el ratón en los diferentes botones de comando.

- Graba el formulario con el nombre **Prog46.frm** el proyecto con el nombre **Prog46.vbp** en *TuCarpeta*.

i) Con el formulario anterior a la vista, vamos a estudiar el evento **LostFocus**: Al perder el foco...

- Añade un nuevo "label" al formulario (será el **Label2**)

- Accede a la ventana de código y escribe los siguientes **procedimientos de evento**:

```
Private Sub Command1_LostFocus()
        Label2.Caption= "El Command1 acaba de perder el foco"
End Sub
```

```
Private Sub Command2_LostFocus()
        Label2.Caption= "El Command2 acaba de perder el foco"
End Sub
```

```
Private Sub Command3_LostFocus()
        Label2.Caption= "El Command3 acaba de perder el foco"
End Sub
```

- Ejecuta el programa. Recuerda que has de ir pulsando repetidamente la tecla [Tab] para obligar a cambiar el "foco"

- Graba el formulario con el nombre **Prog47.frm** y el proyecto con el nombre **Prog48.vbp** en *TuCarpeta.*

j) Haz un nuevo proyecto con las siguientes características:

- Form1:
  Caption: **El evento UNLOAD**

- Escribe el siguiente **procedimiento de evento**:

```
Private Sub Form_Unload (Calcel As Integer)
If MsgBox ("¿Desea salir?", vbOKCancel, "SALIDA") = vbCancel Then
        Cancel = True
End If
End Sub
```

- Disponemos de un formulario vacío y un procedimiento de evento.

  El **evento Unload** se produce al cerrar el formulario (al hacer CLIC en la **X** del extremo superior derecho del **form**)

  El **Procedimiento de Evento Unload** dispone de una variable intrínseca de nombre **Cancel**. Si el valor de la variable **Cancel** es **False**, el formulario se cerrará y en el caso **Cancel** = **True**, no se cerrará el formulario.

- Recuerda la sintaxis de la función **MsgBox**:

```
MsgBox("¿Desea salir?", vbOKCancel, "SALIDA") = vbCancel
```

  - El primer argumento: **¿Desea salir?**, es el mensaje que aparece escrito en el interior de la ventana MsgBox.
  - El segundo argumento: **vbOKCancel**, indica que en la ventana del MsgBox, aparezca el botón [Aceptar] (OK) y también el botón [Cancelar] (Cancel).
  - El tercer argumento: **"SALIDA"**, es el título que aparece en la ventana.

- El **MsgBox** tiene dos posibles valores: **vbCancel**, si hacemos CLIC en el botón [Cancelar] y **vbOK**, si hacemos CLIC en el [Aceptar].

- Ejecuta y prueba el programa para ver si es verdad que funciona.

- Graba el programa como **Prog48.frm, Prog48.vbp**

g) Vamos a estudiar en este apartado, los eventos **Activate** y **Deactivate**. Para ello necesitamos un proyecto con dos formularios (Menú Project – Add Form – [Open]).

- Tenemos pues un proyecto que consta de dos ventanas vacías: **Form1** y **Form2**. Sitúa los dos "forms" en la pantalla de forma que se visualizen los 2 a la vez.

- Selecciona el **Form1**. Accede a su "ventana de código" y escribe los siguientes procedimientos:

```
Private Sub Form_Activate()
 Print "Evento Activate del Form1"
End Sub


Private Sub Form_Deactivate()
 Print "Evento Deactivate del Form1"
End Sub

Private Sub Form_Load()
 Form2.Show
End Sub
```

- Selecciona el **Form2**. Accede a su "ventana de código" y escribe los siguientes procedimientos:

```
Private Sub Form_Activate()
 Print "Evento Activate del Form2"
End Sub

Private Sub Form_Deactivate()
 Print "Evento Deactivate del Form2"
End Sub
```

- Ejecuta el programa. Para observar su funcionamiento deberás activar alternativamente el **Form1** y el **Form2**.

- Graba el programa como **Prog49a.frm, Prog49b.frm, Prog49.vbp**

h) Vamos a trabajar en este apartado con un último evento: **Resize**...

- Crea un nuevo proyecto

- En la propiedad **Caption** del **Form1** escribe: **Ejemplo del evento RESIZE**

- Coloca en el form un **"TextBox"** con la propiedad **Multiline = True.**
  La propiedad anterior nos permitirá introducir más de una línea de texto en el **TextBox**.

- Escribe en la ventana correspondiente, el siguiente procedimiento de evento:

```
Private Sub Form_Resize()
 Text1.Left = 0
 Text1.Top = 0
 Text1.Width = Form1.ScaleWidth
 Text1.Height = Form1.ScaleHeight
End Sub
```

- Ejecuta el programa de la siguiente forma:
  - CLIC en el icono **"Iniciar"**
  - Escribe un texto que ocupe varias líneas.
  - Redimensiona la ventana del **form**, es decir hazla más pequeña o más grande y observa que sucede con el texto que acabas de escribir.

- Veamos si entendemos lo que sucede:
  - Las propiedades **ScaleWidth** y **ScaleHeight** determinan el tamaño del control: la anchura y altura en nuestro caso del **Form1**.
  - Las líneas:

    Text1.Left = 0
    Text1.Top = 0

    Determinan la "posición" del cuadro de texto en el formulario, en nuestro caso en el origen (extremo superior izquierdo) del **Form1**.
  - **Text1.Width** y **Text1.Height** determinan el "tamaño" del control, en nuestro caso del cuadro de texto.
  - Las líneas:

    Text1.Width = Form1.ScaleWidth
    Text1.Height=Form1.ScaleHeight

    Otorgan al **TextBox** un tamaño igual al tamaño del formulario.
  - Y todo lo anterior se produce al "redimensionar el formulario": **Form_Resize**

- Graba el formulario con el nombre **Prog50.frm** y el proyecto con el nombre **Prog50.vbp** en *TuCarpeta*.

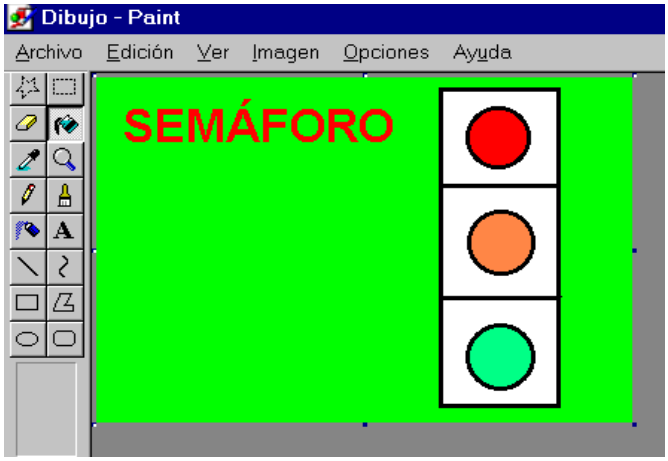
VI.- Timer. PictureBox. Image

Vamos a incluir dibujitos a nuestros proyectos. Comenzaremos con dibujitos nuestros, es decir que hemos de hacer...

- Ejecuta el **"Paint"**, es decir:
  CLIC en [Inicio]
  Programas
  Accesorios
  CLIC en **Paint**

Menú Imagen
Atributos...
Pixeles
Ancho: 360 Alto: 240

Haz aproximadamente el siguiente dibujo:



Ten en cuenta que el fondo es verde, la palabra **SEMÁFORO** debe estar en rojo. El fondo del semáforo es blanco. La "luz" superior está en rojo, la media en ámbar y la inferior en verde.

Graba el dibujo en *TuCarpeta* con el nombre **SEMATODO**

Selecciona la luz roja. Es decir:
- CLIC en el icono **"Selección"** del "**Paint"** (2º icono de la 1ª fila: rectángulo punteado)
- "Marca" un cuadrado que abarque "la luz roja" del semáforo.

Menú Edición
Copiar

Menú Archivo
Nuevo

Menú Edición
Pegar

Redimensiona el área de dibujo de forma que abarque sólo un cuadrado que contenga el círculo rojo.

Graba el dibujo que tienes en pantalla (luz roja), en *TuCarpeta* con el nombre **ROJO**

Recupera el fichero **SEMATODO**:

Menú Archivo
Abrir...

Selecciona la "luz ámbar" y grábala con el nombre **AMBAR**, en *TuCarpeta*. Es decir:

- Con el dibujo **SEMATODO** a la vista
- CLIC en el icono **"Selección"**
- "Marca" un cuadrado que abarque la "luz ámbar" del semáforo.
- Menú Edición – Copiar
- Menú Archivo – Nuevo
- Menú Edición – Pegar
- Redimensiona el área de dibujo de forma que abarque sólo un cuadrado blanco que contiene el círculo ámbar.
- Graba el dibujo que tienes en pantalla (luz ámbar), en tu carpeta con el nombre **AMBAR**
- Recupera el dibujo **SEMATODO** (Menú Archivo – Abrir...)

Repite el proceso anterior para grabar la "luz verde" en tu carpeta, con el nombre **VERDE**

Se trataría de grabar el "semáforo sin luces". Veamos:

- Con el **SEMATODO** a la vista
- "Selecciona" el color rojo y pulsa el botón [Supr]
- Haz lo mismo para borrar el color ámbar y el verde.
- Graba el dibujo resultante con el nombre **SEMAVACI**.

Sal del programa **"Paint"** y ejecuta el **"Visual Basic"**

- Pretendemos en este ejercicio, hacer un programa que simule un semáforo...

Accede a la ventana de propiedades del **Form1** y cambia las siguientes propiedades:

- Caption: SEMÁFORO
- Name: LUCES
- Picture:
  Haz CLIC en el botón [...] del campo **"Picture"** y selecciona el fichero **SEMAVACI** de tu carpeta. Por último CLIC en [Abrir]
- Redimensiona el tamaño de la ventana al tamaño del dibujo.
- BorderStyle: 1 – Fixed Single
  Con el valor de esta propiedad evitamos poder cambiar las dimensiones del "form" en tiempo de ejecución.

Graba el programa como **Prog51.frm, Prog51.vbp**
Vamos a colocar las "3 luces" en el semáforo...

CLIC en el icono **"Image"**:
del "Cuadro de Controles"

"Marca" un cuadrado en el lugar del semáforo correspondiente a la luz roja.

Accede a las propiedades del control **Image1** y cambia las propiedades:

- **Picture**: selecciona el fichero **ROJO.BMP** de *TuCarpeta*.
  Probablemente deberás redimensionar el dibujo para que ocupe el lugar del rojo en el semáforo.
- **Visible:** False

Coloca la "luz ámbar" en su sitio (fichero **AMBAR.BMP**) y en la propiedad **"Visible"** selecciona **"False"**.

Haz lo mismo para la "luz verde"

Ya tenemos el semáforo con sus tres luces. Si ejecutamos el "form" no veremos ninguna "luz", ya que los tres controles **"Image"** que simulan las 3 luces tienen la propiedad **Visible** igual a **False**.

Probablemente no conseguirás colocar las 3 luces alineadas, debido a la **rejilla** que tenemos por defecto en el **form**, para ir colocando los controles. De todas formas, siempre puedes hacer la rejilla más pequeña para "alinear" mejor los controles en el formulario.

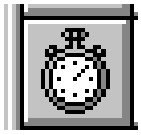
Pruébalo haciendo:

Menú Tools
Options ...
Selecciona la pestaña "General"

No es necesario que hagas la rejilla más pequeña, basta que desactives la opción: **Forzar controles a cuadrícula** (Align Controls to Grid)

Vamos a colocar en el form un nuevo control...

CLIC en el icono **"Timer"**:
del "Cuadro de Controles"



"Marca" en el form: **LUCES** un cuadrado donde tú quieras.

Observa el valor de las siguientes propiedades del control **Timer1**:
Enabled: True
Interval: 0

Ejecuta el programa y observarás:
- No sucede nada
- El control **Timer1** no es visible

El control **Timer** es un control no visual, es decir que en "tiempo de ejecución" no es visible. Tiene como finalidad generar un evento a intervalos regulares según el valor de la propiedad **Interval**. La propiedad "Interval" se mide en milisegundos, es decir, si "Interval = 1000", cada segundo se generará el evento **timer**.

Veamos que utilidad podemos dar al control **Timer** en nuestro "semáforo"...
- Escribe **1000** en la propiedad **"Interval"** del control **Timer1**.
- Accede a la ventana de código y selecciona los campos:
  Objeto: **Timer1**
  Procedimiento: **Timer**
- Escribe:

```
Private Sub Timer1_Timer()
    Image2.Visible = Not Image2.Visible
End Sub
```

Recuerda que el control **Image2** corresponde al color ÁMBAR del semáforo.

La línea de código: **Image2.Visible = Not Image2.Visible**, lo que hace es cambiar la propiedad "Visible" del color ÁMBAR.

Cuando la línea anterior está en un procedimiento de evento **Timer**, de un control **Timer** con frecuencia: 1000. Significa que cada segundo cambiará la "visibilidad" del color ámbar. Dicho de otra forma: habremos conseguido la "señal de precaución".

Veamos si funciona...
- "Cierra" la ventana de código
- Ejecuta el programa (CLIC en el icono **"Iniciar"**)
- Vuelve a ejecutar el programa, pero antes escribe en la propiedad **"Interval"** del **Timer**, el número **250**.

El problema que tenemos en nuestro semáforo es que sólo funciona la luz de precaución.

Vamos a solucionar el problema:
Accede a la **ventana de código** con las opciones:
Objeto: **Form**
Procedimiento: **Load**
Escribe:

```
Private Sub Form_Load()
    Timer1.Enabled = False
End Sub
```

Es decir: al ejecutarse el programa, desactivamos el control **Timer**

Coloca en el formulario 2 **CommandButton** con las siguientes características:
- Command1:
  Caption: Precaución
  Name: Intermitente
- Command2:
  Caption: Ninguno
  Name: Borra

CLIC en [Ver código] y selecciona: Objeto= **Intermitente** y Procedimiento = **Click**

Escribe:

```
Private Sub Intermitente_Click()
    Timer1.Enabled = True
End Sub
```

Desde la ventana de código, selecciona las opciones:
Objeto: Borra
Procedimiento: Click

Escribe:

```
Private Sub Borra_Click()
    Timer1.Enabled = False
End Sub
```

"Cierra" la ventana de código y graba de nuevo el programa.

Ejecuta el "form" y prueba los 2 botones.

Vamos a completar nuestro "semáforo"...

Coloca en el form, tres nuevos **CommandButtons** con las siguientes características:
- Command1:
  Caption: **ROJO**
- Command2:
  Caption: **ÁMBAR**
- Command3:

Caption: **VERDE**

Accede a la **ventana de código** y escribe los siguientes procedimientos:

```
Private Sub Command1_Click()
        Image1.Visible = True
End Sub

Private Sub Command2_Click()
        Image2.Visible = True
End Sub

Private Sub Command3_Click()
        Image3.Visible = True
End Sub
```

Edita el procedimiento **Borra_Click()** y modifícalo de forma que quede:

```
Private Sub Borra_Click()
        Timer1.Enabled = False
        Image1.Visible = False
        Image2.Visible = False
        Image3.Visible = False
End Sub
```

Ejecuta el programa y juega con sus botones.

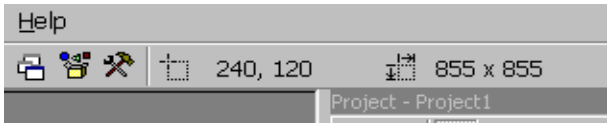
Graba el formulario con el mismo nombre **Prog51.frm** y el proyecto con el nombre **Prog51.vbp** en *TuCarpeta*

Vamos a trabajar con **Matrices de controles** ...

Haz un nuevo proyecto con los siguientes objetos y características:

- Form1:
  Caption: DADO
- Picture1:
  AutoRedraw: True
  BackColor: rojo
  Posición: 240, 120
  Tamaño: 855 x 855

Para conseguir la **Posición** y el **Tamaño** indicados, basta situar el control con el ratón sin perder de vista la zona derecha de la barra de iconos, donde aparece la posición y tamaño del control que estemos situando en un momento dado:



De todas formas podemos conseguir una posición y tamaño exactos cambiando las siguientes propiedades del control:

Height: 855 (segundo número de "tamaño")
Left: 240
Top: 120
Width: 855 (primer número de "tamaño")

- Coloca en el **interior** del **Picture1**, un control **Shape** con las siguientes características:
  Shape1:

Shape: 3 – Circle
BackColor: negro
BackStyle: 1-Opaque
Posición: 360, 360
Tamaño: 132 x 132

Pretendemos simular en este proyecto la tirada de un dado. Acabamos de **dibujar una cara** del dado. Deberíamos dibujar las otras 5 caras y definir una matriz de 6 objetos que contenga las 6 caras del dado.

En lugar de dibujar las 6 caras del dado, vamos a hacerlo más rápido…

Antes de nada observa que hemos dado a la propiedad **AutoRedraw** del **PictureBox** el valor **True**. Esto es así para que se **redibuje** el **PictureBox** cada vez que cambie la cara del dado.

Haz lo siguiente:

- Selecciona el **PictureBox**

- Menú Edición
  Copiar

- Menú Edición
  Pegar

- Observa la pregunta: ¿Desea crear una matriz de controles?. Haz CLIC en [Sí]

- A la siguiente pregunta, vuelve a contestar con un [Sí]

- Coloca la "copia" al lado del original y con la copia seleccionada observa la ventana de propiedades: tenemos el **Picture1(1)**. Si seleccionas el dado original y miras su ventana de propiedades verás que se trata del **Picture1(0)**.

- Selecciona el círculo negro del 2º dado y Menú Edición – Copiar

- Selecciona el 2º dado y Menú Edición – Pegar

- Mueve los dos círculos en el interior del 2º dado hasta que queden estéticos.

- Sigue el mismo proceso hasta conseguir las seis caras del dado.

- Tenemos pues una matriz de **PictureBox** de nombre **Picture1** y valores: Picture1(0), Picture1(1), …, Picture1(5), que representan cada una de las caras de un dado.

- Coloca la propiedad **Visible** de todos los Picture1 **menos el** primero a **False**.

- Coloca todos los Picture1, encima del primero (posición 240, 120 y tamaño 852 x 852)

- Añade los siguientes controles al formulario **DADO**:
  - Timer1:
    Interval: 10
    Enabled: False
  - Command1:
    Caption: Tirada
    Posición: 1320, 360
    Tamaño: 972 x 372

Accede a la ventana de código y…
Y escribe:

```
Dim NúmeroDado As Integer
```

Es decir: definimos una variable global de nombre **NúmeroDado**, que representará el número del dado.

- Objeto: Form – Procedimiento: Load
  Y escribe:

```
Private Sub Form_Load()
 Randomize
 NúmeroDado = 1
End Sub
```

Es decir: al ejecutarse el programa inicializamos el dado a **NúmeroDado = 1**, dicho de otro modo: la cara correspondiente al dígito 1

- Objeto: Command1 – Procedimiento: Click
  Y escribe:

```
Private Sub Command1_Click()
    Timer1.Enabled = True
End Sub
```

Es decir, cada vez que hagamos CLIC en el botón [Tirada], activaremos el **"Timer"**.

- Objeto: Timer1 – Procedimiento: Timer
  Y escribe:

```
Private Sub Timer1_Timer()
   Static Veces As Integer
   Picture1(NúmeroDado - 1).Visible = False
   NúmeroDado = Int(6 * Rnd) + 1
   Beep
   Picture1(NúmeroDado - 1).Visible = True
   Veces = Veces + 1
   If Veces = 10 Then
      Timer1.Enabled = False
      Veces = 0
   End If
End Sub
```

Veamos:

- Definimos una nueva variable: **Veces** que declaramos tipo **Static**, ya que cada vez que se ejecute el **Timer1** (Interval = 10), es decir cada 10 milisegundos (cada centésima de segundo), el valor de la variable **Veces** ha de conservarse. La variable **Veces**, no es mas que un contador, que contará hasta 10.
- La función **Beep** determina un pitido en el altavoz del ordenador.

- Ejecuta el programa y pruébalo

- Graba el formulario con el nombre **Prog52.frm** y el proyecto con el nombre **Prog52.vbp** en *TuCarpeta*.

m) Haz un nuevo proyecto con los siguientes controles y propiedades:

Form1:
Caption: MOVIMIENTO
BorderStyle: 1-Fixed Single
Posición: 864, 1080
Tamaño: 4200 x 4740

Timer1:
Interval: 100

Text1:
Text: (nada)
BorderStyle: 0-None
Enabled: False
Font: Negrita Cursiva. Tamaño: 12
Posición: 840, 240
Tamaño: 1212 x 300

Accede a la ventana de código y escribe el siguiente procedimiento de evento:

```
Private Sub Timer1_Timer()
 Text1.Text = Format(Now, "hh:mm:ss")
End Sub
```

Es decir:
Cada décima de segundo (Interval = 100) se escribirá en el cuadro de texto, la hora actual (Now), en formato **hh:mm:ss** (Format).

Ejecuta el programa para ver si es verdad.
Observa lo fácil que es conseguir un "reloj digital".

Con el formulario **MOVIMIENTO** a la vista.

CLIC en el icono **"PictureBox":**
del "Cuadro de Controles".

"Marca" un cuadrado de posición: 1440, 1560 y tamaño: 852 x 852

La característica fundamental de un **PictureBox** es que puede contener dibujos y otras herramientas, en cambio un control **Image**, sólo puede contener imágenes ya hechas (por un programa de dibujo).

Cambia el valor de la propiedad **"BorderStyle"** del **"Picture1"** anterior, a **0 – None**.

Coloca en el **PictureBox** anterior un control **Shape** con las siguientes características:

Shape1:
Shape: 3-Circle
BackColor: rojo
BackStyle: 1-Opaque
BorderStyle: 0-Transparent
Left: 0
Top: 0
Height: 732
Width: 732

Coloca en el **form**, un nuevo control **"Timer"** con su propiedad **"Interval=10"**

Queremos conseguir que el círculo rojo se mueva por el form…

Consideramos 4 movimientos:

Parado
Movimiento Vertical
Movimiento Horizontal
Movimiento Diagonal

Hemos de definir en primer lugar una variable (EstadoCírculo), que determinará el estado del círculo y 4 constantes que determinarán el estado de movimiento: MovCírculoParado, MovCírculoVertical, MovCírculoHorizontal y MovCírculoDiagonal.

Accede a la ventana de código, selecciona:
Objeto: **(General)**
Procedimiento: **(Declaraciones)**
Y escribe:

```
Dim EstadoCírculo As Integer
Const MovCírculoParado = 0
Const MovCírculoVertical = 1
Const MovCírculoHorizontal = 2
Const MovCírculoDiagonal = 3
```

Desde la ventana de código, selecciona:
Objeto: **Form**
Procedimiento: **Load**

Y escribe:

```
Private Sub Form_Load()
    EstadoCírculo = MovCírculoParado
End Sub
```

Es decir: al leerse el formulario, inicializamos la variable **EstadoCírculo** a 0.

Desde la ventana de código, selecciona:
Objeto: **Picture1**
Procedimiento: **Click**
Y escribe:

```
Private Sub Picture1_Click()
  EstadoCírculo = (EstadoCírculo + 1) Mod 4
End Sub
```

Es decir:
Cada vez que hagamos CLIC en el círculo, la variable **EstadoCírculo**, se incrementará en una unidad.
Al escribir **"Mod 4"** conseguimos que el valor de **EstadoCírculo** sea 0, 1, 2, 3, 0, 1, 2, 3, etc.

El movimiento de nuestro círculo dependerá del **Timer2**…
Desde la ventana de código, selecciona:

Objeto: **Timer2**
Procedimiento: **Timer**

Y escribe:

```
Private Sub Timer2_Timer()
   Dim X, Y As Integer
   Select Case EstadoCírculo
      Case MovCírculoParado
         X = Picture1.Left
         Y = Picture1.Top
      Case MovCírculoHorizontal
         X = (Picture1.Left + 50) Mod Form1.Width
         Y = Picture1.Top
      Case MovCírculoVertical
         X = Picture1.Left
         Y = (Picture1.Top + 50) Mod Form1.Height
      Case MovCírculoDiagonal
         X = (Picture1.Left + 50) Mod Form1.Width
         Y = (Picture1.Top + 50) Mod Form1.Height
   End Select
   Picture1.Move X, Y
End Sub
```

Veamos:

- La orden responsable del movimiento es la última línea: **Picture1.Move X, Y**. Es decir, el círculo se situará en las coordenadas X, Y de pantalla y esto lo hará cada 10 milisegundos (el **Timer2** tiene un Interval = 10)
- El valor que toman las coordenadas X, Y dependen del **EstadoCírculo** (Select Case). Y el estado del círculo dependía de hacer CLIC en el **Picture1**.
- Si el **EstadoCírculo** = 0 (parado), el valor de las coordenadas es

  X = Picture1.Left
  Y = Picture1.Top

es decir la posición actual del círculo. Dicho de otra forma: la orden **Picture1.Move X, Y** no determinará ningún movimiento en el círculo.

- Si es **EstadoCírculo = 1** (movimiento vertical), el valor de las coordenadas será:

  X = Picture1.Left
  Y = (Picture1.Top + 50) Mod Form1.Height

Es decir: la coordenada horizontal no varía, pero la vertical aumenta en 50 unidades. Esto se traducirá en un movimiento hacia debajo de 50 unidades. El uso del operador : **Mod Form1.Height** es para evitar que el valor de la coordenada Y, sea más grande que la altura del **Form1**. Dicho de otra forma: el círculo al llegar al borde inferior del **form**, desaparecerá y volverá a aparecer en el borde superior.

- Básicamente es lo mismo que sucede si **EstadoCírculo = 2** (horizontal): la coordenada X, aumenta en 50 unidades, es decir el círculo se mueve hacia la derecha. Si llega al borde derecho del **form**, gracias al operador: **"Mod Form1.Width"**, vuelve a aparecer por el borde izquierdo.
- Si **EstadoCírculo = 3** (diagonal): la coordenada "X" aumenta en 50 unidades y la coordenada "Y" también. Esto se traducirá en un movimiento hacia abajo y a la derecha.

Graba el formulario con el nombre **Prog53.frm** y el proyecto con el nombre **Prog53.vbp** en *TuCarpeta*.

Ejecuta el proyecto y pruébalo.

Nos gustaría completar el proyecto anterior, pero antes vamos a recordar dos funciones incorporadas al **V.B.…**.

Crea un **nuevo proyecto** y coloca en el formulario un **CommandButton**.

Accede a la "ventana de código" y escribe el siguiente procedimiento:

```
Private Sub Command1_Click()
        Dim X As Double
        Dim I As Integer
        For I = 1 To 5
                X = Rnd
                Print X
        Next I
End Sub
```

Ejecuta el programa, es decir:

- CLIC en el icono **"Iniciar"**
- CLIC en [Command1]
- Observa que aparecen 5 números menores de 1
- Vuelve a hacer CLIC en [Command1]
- Sal del programa: CLIC en el icono **"Terminar"**

Recuerda que la función incorporada **"Rnd"**, servia para generar números "aleatorios" entre 0 y 1.
Pero hay un problema:

- Ejecuta varias veces el programa y observarás que el listado de números aleatorios es cada vez el mismo.

Vamos a solucionar el problema anterior:

- "Edita" el procedimiento de evento **"Command1_Click"** y escribe cómo primera línea del procedimiento la orden:

Randomize

- Vuelve a ejecutar el programa tantas veces cómo quieras y observarás que cada vez el listado de números aleatorios es distinto.

Con la sentencia **"Randomize"**, escrita antes de la función **Rnd**, conseguimos que la generación de números aleatorios sea realmente **"aleatoria"**.

La función **"Rnd"**, nos permite conseguir números aleatorios de3 cualquier tipo, no únicamente entre 0 y 1.
En efecto:

- Edita el procedimiento de evento **"Command1_Click"** y en lugar de la línea:

  X = Rnd

  escribe:

  X = Int(Rnd * 100) + 1

- Vuelve a ejecutar (tantas veces cómo quieras) el programa y observa el tipo de números que se generan.
  Está claro que se generan números aleatorios entre 1 y 100.
- Si quisiéramos simular las tiradas de un dado de parchís, deneríamos escribir:

  X = Int(Rnd * 6) +1

- Pruébalo.

Graba el formulario con el nombre **Prog54.frm** y el proyecto con el nombre **Prog54.vbp** en *TuCarpeta*.

Recupera el proyecto **Prog53.vbp** de *TuCarpeta*.

Recuerda que en el área **"General"** de la ventana de código, teníamos definida la variable **EstadoCírculo** y 4 constantes:

```
MovCírculoParado = 0
MovCírculoVertical = 1
MovCírculoHorizontal = 2
MovCírculoDiagonal = 3
```

En el procedimiento **Form_Load()**, teníamos inicializada la variable **EstadoCírculo** a 0 (parado).

Se trata de cambiar el procedimiento **Timer2_Timer**, para que el movimiento del círculo sea aleatorio…

- Accede al procedimiento **"Timer2_Timer"**
- Añade entre las líneas:

```
Dim X, Y As Integer
Y
Select Case EstadoCírculo
```

Las siguientes líneas de programa:

```
Randomize
EstadoCírculo = Int(Rnd * 3)
```

Ejecuta el programa y no hagas ningún CLIC en el círculo rojo.

Graba el formulario con el nombre **Prog55.frm** y el proyecto con el nombre **Prog55.vbp**.

Vamos a utilizar dos nuevos controles: las barras de desplazamiento vertical (VscrollBar) y horizontal (HscrollBar)

n) Haz un nuevo proyecto, con los siguientes objetos y propiedades:

Label1:
- Name: Etiqueta
- Caption: Coordenadas

HScroll1 (has de buscarlo en el icono **HscrollBar** de la barra de herramientas):
- Name: BarraHorizontal
- LargeChange: 25

VSroll1 (has de buscarlo en el icono **VscrollBar** de la barra de herramientas):
- Name: BarraVertical
- LargeChange: 25

Escribe los siguientes procedimientos:

```
Private Sub BarraHorizontal_Change()
        Etiqueta.Left = BarraHorizontal.Value
        Etiqueta.Caption = Etiqueta.Left & ", " & Etiqueta.Top
End Sub


Private Sub BarraHorizontal_Scroll()
        BarraHorizontal_Change
End Sub


Private Sub BarraVertical_Change()
        Etiqueta.Top = BarraVertical.Value
        Etiqueta.Caption = Etiqueta.Left & ", " & Etiqueta.Top
End Sub

Private Sub BarraVertical_Scroll()
        BarraVertical_Change
End Sub
```

```
Private Sub Form_Resize()
        With BarraHorizontal
                .Move 0, ScaleHeight  - .Height, ScaleWidth,  .Height
                .Max = ScaleWidth – BarraVertical.Width
        End With
        With BarraVertical
                .Move ScaleWidth  -  .Width, 0,  .Width , ScaleHeight  -
                                                        BarraHorizontal.Height
                .Max = ScaleHeight  - BarraHorizontal.Height
        End With
        BarraHorizontal_Change
        BarraVertical_Change
End Sub
```

Graba el programa como **Prog56.frm, Prog56.vbp** y prueba el programa.

o) Haz un nuevo proyecto e inserta
tres HscrollBar de nombre: **HSColorR, HSColorG, HSColorB**
un label de nombre **lblColor**
un commandbutton

Y código:

```
Private Sub Command1_Click()
   Unload Me
End Sub

Private Sub hsColorR_Scroll()
lblColor.Caption = "Color RGB(" & HSColorR.Value & "," & HSColorG.Value & "," &
HSColorB.Value & ")"
End Sub

Private Sub hsColorG_Scroll()
lblColor.Caption = "Color RGB(" & HSColorR.Value & "," & HSColorG.Value & "," &
HSColorB.Value & ")"
End Sub

Private Sub hsColorB_Scroll()
lblColor.Caption = "Color RGB(" & HSColorR.Value & "," & HSColorG.Value & "," &
HSColorB.Value & ")"
End Sub
```

```
Private Sub hsColorR_Change()
lblColor.Caption = "Color RGB(" & HSColorR.Value & "," & HSColorG.Value & "," &
HSColorB.Value & ")"
End Sub

Private Sub hsColorG_Change()
lblColor.Caption = "Color RGB(" & HSColorR.Value & "," & HSColorG.Value & "," &
HSColorB.Value & ")"
End Sub

Private Sub hsColorB_Change()
lblColor.Caption = "Color RGB(" & HSColorR.Value & "," & HSColorG.Value & "," &
HSColorB.Value & ")"
End Sub

Private Sub lblColor_Change()
lblColor.ForeColor = RGB(HSColorR.Value, HSColorG.Value, HSColorB.Value)
End Sub
```

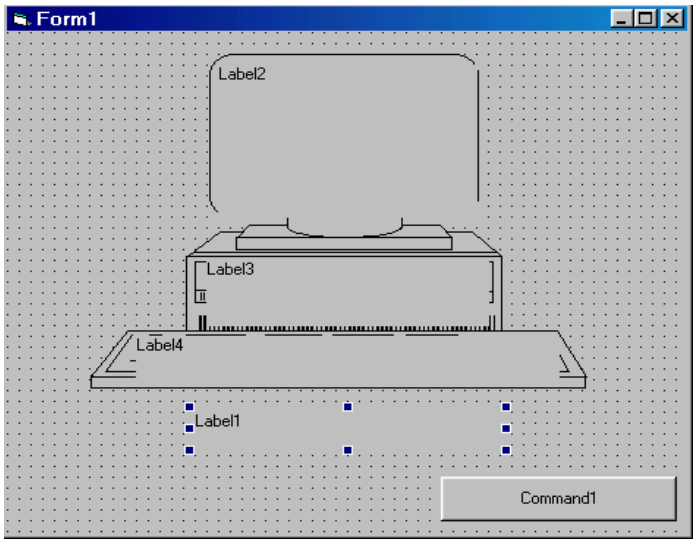
Graba el programa como **Prog57.frm, Prog57.vbp** y pruébalo.

p) Supongamos que en TuCarpeta tenemos un gráfico de nombre **PCOMPUTR.WMF,** que simula un ordenador, con sus tres partes fundamentales: monitor, teclado y CPU.

Crea un nuevo proyecto

- Inserta un “Image” que ocupe la mayor parte del form.
Propiedades del Image1:
Picture = PCOMPUTR.WMF
Stretch = True

- Inserta 4 labels y 1 commandbutton de la siguiente manera:



Observa que:

a) El “label2” tapa todo el monitor del ordenador
b) El “label3” tapa la CPU
c) El “label4” tapa el teclado.

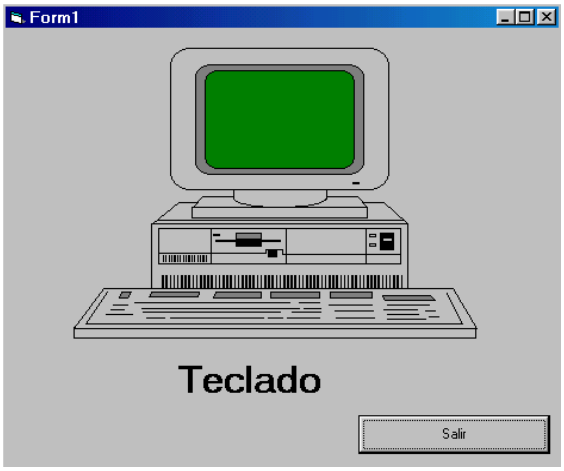
- Propiedades del Label2, Label3 y Label4:
Caption = borra su contenido
BackStyle = 0 – Transparent

- Borra el contenido de la propiedad Caption del “Label1”. Y en la propiedad “Font”: Bold - 24

- Código:

```
Private Sub Command1_Click()
   End
End Sub

Private Sub Form_Load()
   Command1.Caption = "Salir"
End Sub

Private Sub Label2_Click()
   Label1.Caption = "Monitor"
End Sub

Private Sub Label3_Click()
   Label1.Caption = "C.P.U."
End Sub

Private Sub Label4_Click()
   Label1.Caption = "Teclado"
End Sub
```

- Grábalo como **Prog58.frm, Prog58.vbp** y pruébalo:



q) Vamos a ver un proyecto que demuestra como pueden detectarse las pulsaciones de teclas mediante los eventos de teclado.

Crea un nuevo proyecto

Hemos de conseguir un formulario que muestra el estado de pulsación de las teclas de modificación Shift, Alt y Ctrl. Al pulsar una tecla aparece un icono de pulsación en la casilla correspondiente.

Proceso a seguir:

Añadir al formulario por defecto tres imágenes y tres etiquetas.

Necesitamos como puedes observar en la ilustración una imagen o icono, supongamos que es: Mano.ico

Especificar las siguientes propiedades en tiempo de diseño:

Objeto	**Propiedad**	**Valor**
Form1	Caption	Teclas de modificación
Image1	Name	imgShift
	Picture	Mano.ico
	Visible	False
Image2	Name	imgAlt
	Picture	Mano.ico
	Visible	False
Image3	Name	imgCtrl
	Picture	Mano.ico
	Visible	False
Label1	Name	lblShift
	BorderStyle	1 (Fixed Single)
	Caption	Shift
	FontBold	True
Label2	Name	lblAlt
	BorderStyle	1 (Fixed Single)
	Caption	Alt
	FontBold	True
Label3	Name	lblCtrl
	BorderStyle	1 (Fixed Single)
	Caption	Ctrl
	FontBold	True

3. Escribir los siguientes procedimientos de evento:

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
  If Shift And vbShiftMask Then imgShift.Visible = True
  If Shift And vbAltMask Then imgAlt.Visible = True
  If Shift And vbCtrlMask Then imgCtrl.Visible = True
End Sub

Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
  If Not (Shift And vbShiftMask) Then imgShift.Visible = False
  If Not (Shift And vbAltMask) Then imgAlt.Visible = False
  If Not (Shift And vbCtrlMask) Then imgCtrl.Visible = False
End Sub
```

4. Guardar el formulario y el proyecto en los archivos **Prog59.frm y Prog59.vbp**, respectivamente.

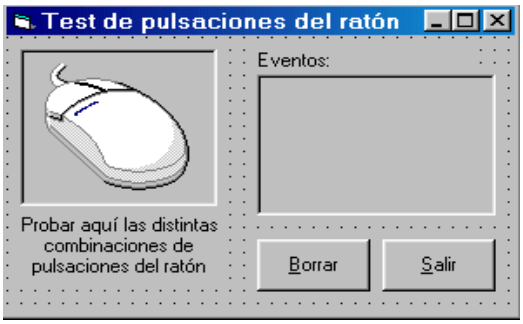
r) Vamos a hacer un proyecto que ilustra el orden en que se producen los eventos relacionados con las pulsaciones del ratón.

Crea un nuevo proyecto

Se trata de conseguir una superficie sobre la que pueden realzarse distintas pruebas de clics. Los eventos producidos se irán mostrando en el panel situado a su derecha.

El proceso que debes seguir es el siguiente:

1. Añadir al formulario por defecto un cuadro de imagen, un cuadro de dibujo, dos etiquetas y dos botones de comando.



2. Especificar las siguientes propiedades en tiempo de diseño:

Objeto	Propiedad	Valor
Form1	Caption	Test de pulsaciones del ratón
Image1	BorderStyle	1 (Fixed Single)
	Picture	MOUSE.BMP
Picture1	FontBold	True
Label1	Caption	Probar aquí las distintas combinaciones de pulsaciones del ratón
	Alignment	2 (Center)
Label2	Caption	Eventos:
Command1	Caption	&Borrar
Command2	Caption	&Salir

3. Escribir los siguientes procedimientos de evento:

```
Private Sub Command1_Click()
  'Borrar el cuadro de dibujo
  Picture1.Cls
End Sub

Private Sub Command2_Click()
  Unload Me
End Sub

Private Sub Image1_Click()
  Picture1.Print "Click"
End Sub

Private Sub Image1_DblClick()
  Picture1.Print "DblClick"
End Sub
```

```
Private Sub Image1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Picture1.Print "MouseDown"
End Sub

Private Sub Image1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Picture1.Print "MouseUp"
End Sub
```

4. Guardar el formulario y el proyecto en los archivos **Prog60.frm y Prog60.vbp**, respectivamente.

s) Vamos a ver como pueden utilizarse los controles y métodos de Visual basic para crear animaciones.

Crea un nuevo proyecto

Hemos de conseguir que un pequeño balón rebote indefinidamente contra los bordes del área de cliente del formulario.

El proceso a seguir es:

1. Añadir al formulario por defecto un cuadro de imagen y un temporizador.

2. Especificar las siguinetes propiedades en tiempo de diseño:

Objeto	Propiedad	Valor
Form1	name	frmDemo
	Caption	Ejemplo de animación
Timer1	Name	tmrDemo
	Interval	10
Image1	name	imgPelota
	Picture	BALON.ICO

3. Escribir las siguientes declaraciones en la sección general:

```
'Para guardar los incrementos actuales
Dim Horiz As Integer, Vertic As Integer
```

3. Escribir los siguientes procedimientos de evento:

```
Private Sub Form_Load()
  'Valores iniciales de los incrementos
  Horiz = 20
  Vertic = 20
End Sub
```

```
Private Sub tmrDemo_Timer()
  'Mover la pelota
  imgPelota.Move imgPelota.Left + Horiz, imgPelota.Top + Vertic
  'Si se ha llegado al borde izquierdo
  If imgPelota.Left < 0 Then
    Horiz = 20
  End If
  'Si se ha llegado al borde derecho
  If imgPelota.Left >= (frmDemo.ScaleWidth - imgPelota.Width) Then
    Horiz = -20
  End If
  'Si se ha llegado al borde superior
  If imgPelota.Top < 0 Then
    Vertic = 20
  End If
  'Si se ha llegado al borde inferior
  If imgPelota.Top >= (frmDemo.ScaleHeight - imgPelota.Height) Then
    Vertic = -20
  End If
End Sub
```

4. Guardar el formulario y el el proyecto en los archivos **Prog61.frm y Prog61.vbp**, respectivamente.

OBSERVA
Las variables Horiz y Vertic contien los incrementos en la posición del balón cada vez que produce un tic del temporizador. Su valor puede ser positivo o negativo, dependiendo de la dirección del movimiento. Además, es necesario controlar cuando se llega a un borde, con el fin de cambiar la dirección del movimiento. Nótese que la posición del balón viene referida a la esquina superior izquierda de la imagen que lo contiene.

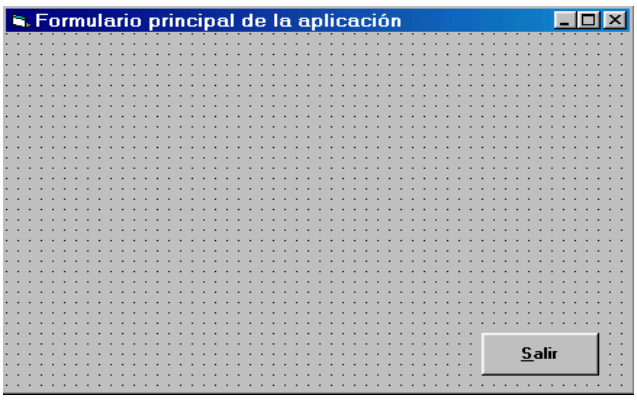
t) Vamos a ver como puede utilizarse un formulario para realizar una presentación temporal de una aplicación.

Crea un nuevo proyecto

Hemos de conseguir una ventana de presentación que permita realizar operaciones de carga, descargándose pasado un cierto tiempo y dando paso a la ventana principal de la aplicación.

El proceso que debes seguir es el siguiente:

1. Añadir al formulario por defecto un botón de comando.



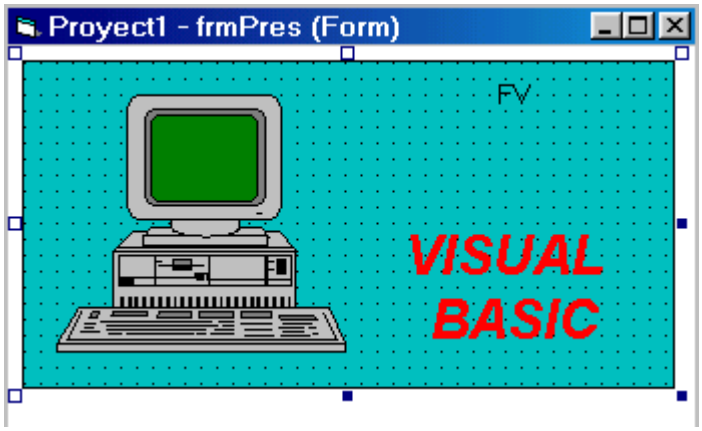
2. Especificar las siguientes propiedades en tiempo de diseño:

Objeto	Propiedad	Valor
Form1	Name	frmPrin
	Caption	Formulario principal de la aplicación
Command1	Name	cmdSalir
	Caption	&Salir
	FontBold	True

3. Escribir los siguientes procedimientos de evento:

```
Private Sub Form_Load()
  'Cargar el formulario de presentación
  Load frmPres
  'Sitúe aquí el código de inicialización
  For i = 1 To 1500000
  Next
  'Descargar el formulario de presentación
  Unload frmPres
End Sub

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

4. Añadir un segundo formulario al proyecto, con una imagen y dos etiquetas.



5. Especificar para éste las siguientes propiedades en tiempo de diseño:

Objeto	Propiedad	Valor
Form2	Name	frmPres
	BackColor	&H00C0C000& (Azúl claro)
	BorderStyle	1 (Fixed Single)
	ControlBox	False
	MaxButton	False
	MinButton	False
Image1	Stretch	True
	Picture	PCOMPUTR.WMF
Label1	Caption	VISUAL BASIC
	Alignment	2 (Center)
	BackStyle	0 (Transparent)
	FontName	Arial
	FontBold	True
	FontSize	20
	FontItalic	True

	ForeColor	&H000000FF& (Azúl)
Label2	Caption FV	
	Alignment	2 (Center)
	BackStyle	0 (Transparent)

5. Escribir los siguientes procedimientos de evento:

```
Private Sub Form_Load()
  frmPres.Visible = True
  frmPres.Refresh
End Sub
```

6. Guardar los formularios y el proyecto en los archivos **Prog62a.frm, Prog62b.frm y Prog62.vbp**, respectivamente.

OBSERVA
Antes de visualizarse el formulario principal, se carga el secundario y se visualiza. En este punto se ha incluido un bucle de espera que representa las operaciones de inicialización de la aplicación. Una vez finalizado el bucle, el formulario secundario se descarga, y se da paso al formulario principal.

u) Vamos a ver como puede utilizarse el control temporizador para generar eventos a intervalos establecidos por el programador, permitiendo realizar acciones periódicas.

Crea un nuevo proyecto

Se trata de conseguir un pequeño reloj digital que indica la hora y la fecha del sistema, tanto en estado normal como cuando está minimizado.

El proceso que debes seguir es el siguiente:

1. Añadir al formulario por defecto cuatro etiquetas y un control temporizador.

2. Especificar las siguientes propiedades en tiempo de diseño:

Objeto	**Propiedad**	**Valor**
Form1	Name	frmReloj
	Caption	Reloj digital
	BorderStyle	1 (Fixed Single)
	Icon	CLOCK.ICO
Timer1	Interval	1000
	Enabled	True
Label1	Caption	Hora:
Label2	Caption	Fecha:
Label3	Name	lblHora
	Alignment	2 (Center)
	BackColor	&H00000000& (Negro)
	BorderStyle	1 (Fixed Single)
	FontName	Arial
	FontBold	True

	FontSize	14
	ForeColor	&H0000FFFF& (Amarillo)
Label4	Name	lblfecha
	Alignment	2 (Center)
	BackColor	&H00000000& (Negro)
	BorderStyle	1 (Fixed Single)
	FontName	Arial
	FontBold	True
	FontSize	14
	ForeColor	&H0000FF00& (Verde)

3. Escribir el siguiente procedimiento de evento:

```
Private Sub Timer1_Timer()
  'Si la ventana se encuentra minimizada
  If WindowState = vbMinimized Then
    frmReloj.Caption = "Reloj: " & Now
  Else
    frmReloj.Caption = "Reloj"
    lblFecha = Date
    lblHora = Time
  End If
End Sub
```

OBSERVA
Para obtener la fecha y la hora del sistema se ha hecho uso de las funciones Date y Time, que proporcionan dichos valores. Además, la función Now devuelve una cadena con ambas, que es la utilizada cuando el reloj se encuentra minimizado.

Graba el programa como **Prog63.frm**
Prog63.vbp

v) Crea un nuevo proyecto
Se trata de conseguir una ventana que contiene un dibujo de una bombilla y un botón que permite apagarla y encenderla cuando se hace clic sobre él.

Proceso a seguir:

1. Añadir al formulario por defecto dos controles de imagen, situándolos uno encima del otro, y dos botones de comando.

2. Especificar las siguientes propiedades en tiempo de diseño:

Objeto	**Propiedad**	**Valor**
Form1	Caption	Ejemplo de botones de comando
imgEncendida	Picture	LIGHTON.ICO (gráfico de la luz encendida)
	Visible	False
imgApagada	Picture	LIGHTOFF.ICO (gráfico de la luz apagada)
cmdInterruptor	Caption	&Encender
cmdSalir	Caption	&Salir

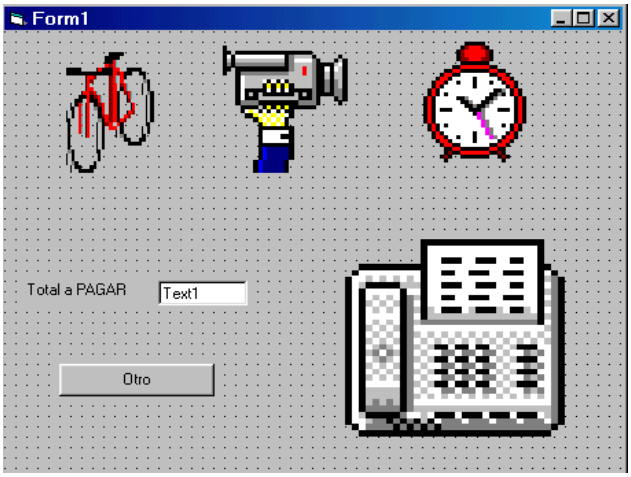
3. Escribir los siguientes procedimientos de evento:

```
Private Sub cmdInterruptor_Click()
  'Cambiar el dibujo
  imgEncendida.Visible = Not imgEncendida.Visible
  imgApagada.Visible = Not imgApagada.Visible
  'Cambiar el texto del interruptor
  If imgEncendida.Visible Then
    cmdInterruptor.Caption = "&Apagar"
  Else
    cmdInterruptor.Caption = "&Encender"
  End If
End Sub

Private Sub cmdSalir_Click()
  End
End Sub
```

Grábalo como **Prog64.frm**
Prog64.vbp

Ejercicios VI

1) Crea un programa con las siguientes características:



Representa una “tienda virtual”: nuestra tienda dispone de tres únicos articulos, cuyos gráficos aparecen en la parte superior del formulario. Al “arrastrar” uno o más artículos a la calculadora que tenemos más abajo, en el textbox aparece el “total a pagar”

Ayudas:

- Las “Images” correspondientes a los artículos deben tener la propiedad **DragMode=1**, para poder arrastrarlas.

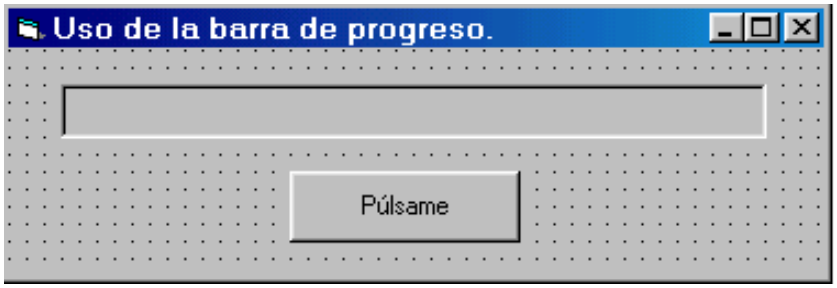
- Código:

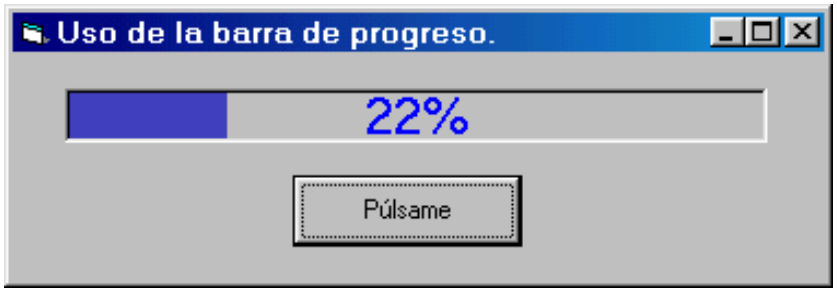
```
Dim total As Double

Private Sub Command1_Click()
   Text1.Text = ""
   total = 0
End Sub

Private Sub Image4_DragDrop(Source As Control, X As Single, Y As Single)
   If Source = Image1 Then
      total = total + 27000
   ElseIf Source = Image2 Then
      total = total + 113000
   ElseIf Source = Image3 Then
      total = total + 800
   End If
   Text1.Text = total
End Sub
```

- Grábalo como **Ejer31.frm, Ejer31.vbp**

2) Crea un programa que simule una barra de progreso.

Para ello te doy el aspecto del formulario en tiempo de diseño y de ejecución y también el código del programa:



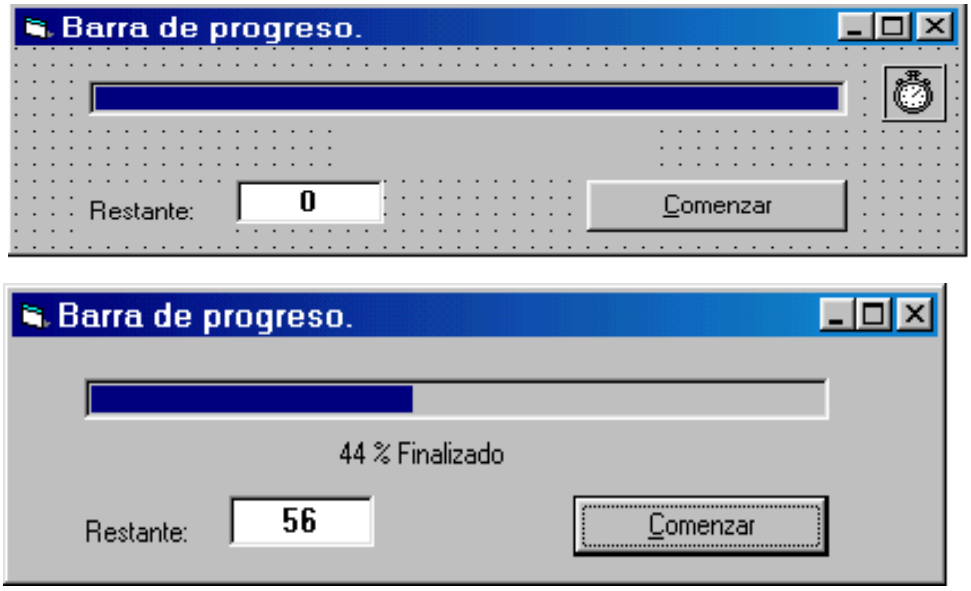



```
Dim I As Long

Private Sub Command1_Click()
   Picture1.ForeColor = RGB(0, 0, 255)

   For I = 0 To 100
      actualizaprogress Picture1, I
      Call pausa
   Next I
   Picture1.Cls ' limpia la barra al final

End Sub

Private Sub actualizaprogress(pb As Control, ByVal percent)
   Dim num$ ' porcentaje
   'el autoredraw debe estar a = true
   pb.Cls
   pb.ScaleWidth = 100
   pb.DrawMode = 10
   num$ = Format(percent, "##") + "%"
   pb.FontSize = 18
   pb.CurrentX = 50 - pb.TextWidth(num$) / 2
   pb.CurrentY = (pb.ScaleHeight - pb.TextHeight(num$)) / 2
   pb.Print num$
   pb.Line (0, 0)-(percent, pb.ScaleHeight), , BF
   pb.Refresh
End Sub

Private Sub pausa()
   Dim controlar
   Dim comenzar

   comenzar = Timer
```

```
   Do Until controlar >= comenzar + 0.2
      controlar = Timer
   DoEvents
   Loop
End Sub
```

Graba el programa como **Ejer32.frm, Ejer32.vbp**

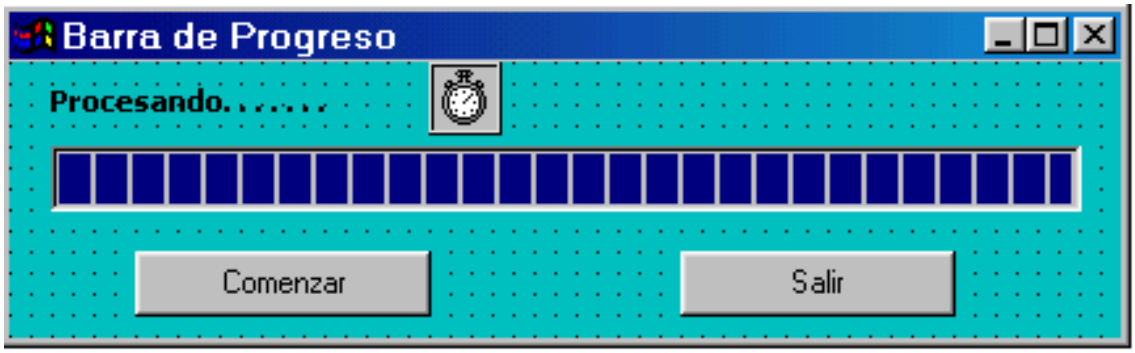
3) Crea otra “barra de progreso” pero con el siguiente aspecto:

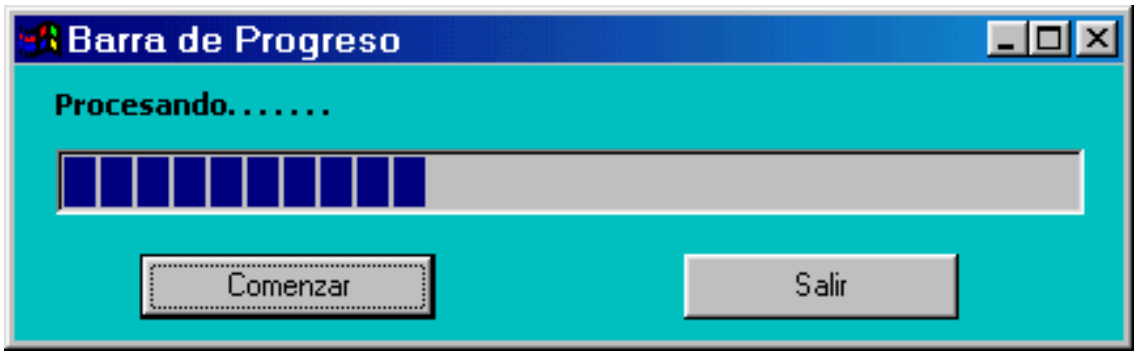



```
Option Explicit
'Ponemos el valor (100)
Private Const m_intInicio As Integer = 100
Dim i As Integer, x As Integer
Private Sub cmdComenzar_Click()

   'Establecemos el primer valor de la etiqueta.
   Me.lblCuenta.Caption = m_intInicio
   'Establece el intervalo al Timer para que
   'comience a descontar .
   Me.tmrCuenta.Interval = 100    'milisegundos
 End Sub

Private Sub tmrCuenta_Timer()
   'Cada décima de segundo, descontamos.
   Progreso.Value = Progreso.Value + 1
   Me.lblCuenta.Caption = 100 - (Progreso.Value \ 1)
   Label1 = Progreso & " % Finalizado"
   'Cuando llega a cero, paramos.
   If CInt(Me.lblCuenta.Caption) = 0 Then
      Me.tmrCuenta.Interval = 0
   End If
End Sub
```

Graba el programa como **Ejer33.frm, Ejer33.vbp**

4) Programa otra “barra de progreso”con el siguiente aspecto:

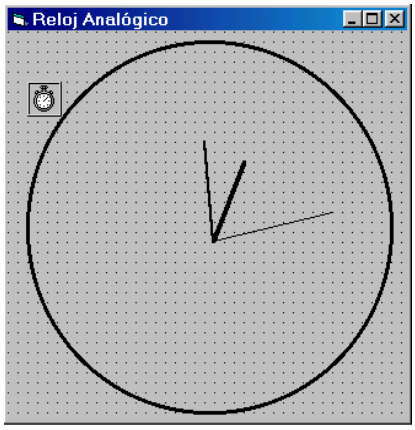


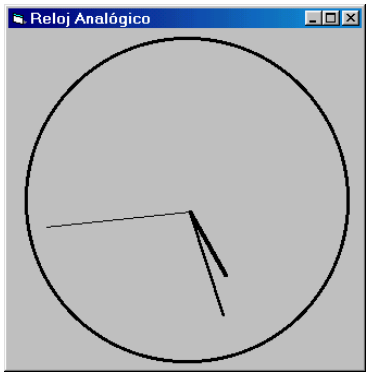



```
Private Sub cmdSalir_Click()
  End
End Sub

Private Sub Form_Load()
  'Inicializamos el timer para que este apagado
  Timer1.Enabled = False
End Sub

Private Sub cmdComenzar_Click()
  'Inicializamos la barra de progreso, y comenzamos el Timer
  Progreso.Value = 0
  Timer1.Enabled = True
End Sub

Private Sub Timer1_Timer()
  'Si el progreso todavía no termina, seguir aumentando Value hasta 100
  With Progreso
    If .Value < .Max Then
      .Value = .Value + 1
    Else
      'Termino
      MsgBox "Proceso Terminado", , "Barra de Progreso"
      'Deshabilitamos el Timer
      Timer1.Enabled = False
    End If
  End With
End Sub
```

Graba el programa como **Ejer34.frm, Ejer34.vbp**

5) Programa un reloj analógico con el siguiente aspecto:





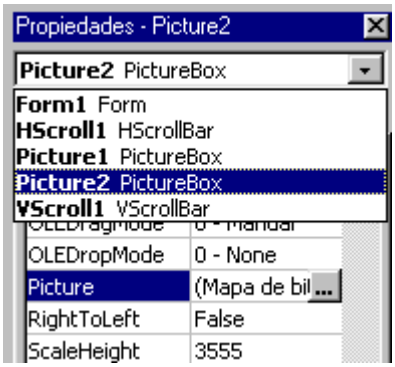

```
Private Sub Reloj_Timer()
   Const Radianes = 3.1415927 / 180

   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
```

Graba el programa como **Ejer35.frm, Ejer35.vbp**

6) Habrás observado que en los controles gráficos no se produce un efecto de “scroll” si el “picture” que insertamos es más grande que el tamaño del control.
Te propongo que hagas un programa con un Picture con Scroll.

Es decir:

Observa los controles que debes considerar:



Y el código correspondiente:

```
Private Sub Form_Load()
   picture2_change
End Sub

Private Sub HScroll1_Change()
   Picture2.Left = Picture1.Left - HScroll1.Value
End Sub

Private Sub picture2_change()
   VScroll1.Min = Picture1.Top
   HScroll1.Min = Picture1.Left

   VScroll1.Max = Picture2.Height - Picture1.Height

   If VScroll1.Max < VScroll1.Min Then
      VScroll1.Max = VScroll1.Min
   End If

   HScroll1.Max = Picture2.Width - Picture1.Width
   If HScroll1.Max < HScroll1.Min Then
      HScroll1.Max = HScroll1.Min
```

```
    End If

    VScroll1.LargeChange = Picture2.Height / 10
    HScroll1.LargeChange = Picture2.Width / 10
End Sub

Private Sub VScroll1_Change()
    Picture2.Top = Picture1.Top - VScroll1.Value
End Sub
```

Graba el programa como **Ejer36.frm, Ejer36.vbp**

VII.- OptionButton. CheckBox. List y ComboBox

Crea un nuevo proyecto

- Establece las siguientes propiedades para el **Form**:
  - Caption: IVA
  - Name: IVA1
  - BorderStyle: 1-Fixed Single
  - Height: 3444
  - Left: 864
  - Top: 1080
  - Width: 4596

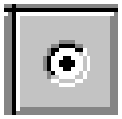
- Coloca en el "form" anterior los siguientes "labels":
  - Label1:
    Posición: 360, 0
    Tamaño: 3732 x 372
    Caption: Cálculo del I.V.A.
    Font: Negrita cursiva
    Tamaño: 18
    Alignment: 2-Center
    ForeColor: verde
  - Label2:
    Posición: 480, 600
    Tamaño: 1452 x 252
    Caption: CANTIDAD:
    Font: Negrita
    Tamaño: 12

- Coloca en el form: **IVA1**, el siguiente "TextBox":
  - Text1:
    Text: (borra el texto que aparece por defecto)
    Font: Negrita
    Tamaño: 12
    ForeColor: rojo
    Height: 420
    Left: 2040
    Top: 600
    Width: 1332

- Graba el form en *TuCarpeta* con el nombre **Prog65a**.

Vamos a trabajar con un nuevo control...

- CLIC en el icono **"OptionButton":** del "Cuadro de Controles".



- Marca un recuadro en el "form" de posición: 600, 1320 y tamaño: 1092 x 372

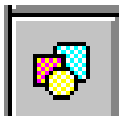
- Establece para el **OptionButton**: **Option1**, las siguientes propiedades:
  - Caption: PTAS
  - Font: Negrita

- Coloca en el "form" otro **OptionButton** con las siguientes características:

- Option2:

  Posición: 600, 1800
  Tamaño: 1092 x 372
  Caption: EUROS
  Font: Negrita

- "Dibuja" un rectángulo que abarque las **PTAS** y los **EUROS** de la siguiente forma:
  - CLIC en el icono **"Shape"**:
  del "Cuadro de Controles"

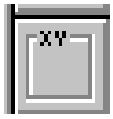


  - Marca en el "form" un recuadro de posición: 480, 1200 y tamaño: 1332 x 1092

- Ejecuta el "form" para observar el funcionamiento de los **OptionButton**. Está claro que funcionan como nos interesa; sólo podemos seleccionar uno de los 2 controles: PTAS o EUROS.

  Pero además de PTAS o EUROS, nos interesará escoger el tipo de I.V.A.: 4%, 7% o 16%. Para poder incluir 3 nuevos **OptionButton**, que sean independientes de los dos primeros, no tenemos otro remedio que trabajar con un nuevo control: **Frame**...

- CLIC en el icono **"Frame":**
  del "Cuadro de Controles".



- "Marca" en el "form", un rectángulo de posición 2640, 1080 y tamaño: 1092 x 1212

- En la propiedad **"Caption"** del control **"Frame"**, escribe: **% I.V.A.**

- Coloca en el "form" (en realidad será en el "frame"), tres OptionButton con las siguientes características:

  - Option3:

    Posición: 240, 240
    Tamaño: 612 x 252
    Caption: 4%

  - Option4:

    Posición: 240, 480
    Tamaño: 612 x 252
    Caption: 7%

  - Option5:

    Posición: 240, 720
    Tamaño: 612 x 252
    Caption: 16%

- Ejecuta el "form", para investigar si los **OptionButton**, se comportan de la forma que nos interesa:
  - Sólo podemos escoger PTAS o EUROS
  - Sólo podemos escoger un tipo de I.V.A.

- Coloca en el form: IVA1 un **CommandButton** con las siguientes características:
  - Posición: 1800, 2520
  - Tamaño: 972 x 372
  - Caption: Aceptar

- Graba el formulario **IVA1** con el mismo nombre **Prog65a.frm**.

Vamos a hacer otro "form", que contendrá el resultado de las opciones que seleccionemos en el primer form: IVA1

- Inserta un nuevo formulario a nuestro proyecto

- Establece las siguientes propiedades para el nuevo form:
  - Caption: RESULTADO
  - Name: IVA2
  - BorderStyle: 1- Fixed Single
  - Height: 2400
  - Left: 5496
  - Top: 3444
  - Width: 3780

- Graba el nuevo formulario, en *TuCarpeta* con el nombre **Prog65b.frm**

- Coloca en el "form": **IVA2** los siguientes "Labels":
  - Label1:
    Posición: 120, 480
    Tamaño: 972 x 252
    Caption: CANTIDAD:
    Font: Negrita
  - Label2:
    Posición: 120, 840
    Tamaño: 972 x 252
    Caption: I.V.A.:
    Font: Negrita
  - Label3:
    Posición: 120, 1200
    Tamaño: 972 x 252
    Caption: P.V.P.:
    Font: Negrita
  - Label4:
    Posición: 1200, 240
    Tamaño: 972 x 252
    Caption: PESETAS:
    Font: Negrita
  - Label5:
    Posición: 2520, 240
    Tamaño: 972 x 252
    Caption: EUROS:
    Font: Negrita

- Coloca en el form: **IVA2** los siguientes **TextBoxs:**
  - Text1:
    Posición: 1200, 480
    Tamaño: 1092 x 288
    Name: PESCAN

- Text2:
  Posición: 2520, 480
  Tamaño: 972 x 288
  Name: EURCAN
- Text3:
  Posición: 1200, 840
  Tamaño: 1092 x 288
  Name: PESIVA
- Text4:
  Posición: 2520, 840
  Tamaño: 972 x 288
  Name: EURIVA
- Text5:
  Posición: 1200, 1200
  Tamaño: 1092 x 288
  Name: PESPVP
- Text6:
  Posición: 2520, 1200
  Tamaño: 972 x 288
  Name: EURPVP

- Graba de nuevo el form: **IVA2** (con el mismo nombre), y el proyecto como **Prog65.vbp**

Vamos a escribir el código de nuestro programa:

- En primer lugar nos interesa el form: IVA1, como formulario principal. Por lo tanto:
  - Menú Project
    Project1 Properties
  - Activa sino lo está ya, la pestaña “General”
  - En el campo “Startup Object”, selecciona sino lo está ya, **IVA1**
  - [Aceptar]

- En segundo lugar, al abrirse el form: **IVA1** nos interesa que también se visualize el form: **IVA2**, por lo tanto:
  - Escribe en el módulo del formulario **IVA1**:

```
Private Sub Form_Load()
        Load IVA2
        IVA2.Show
                End Sub


Private Sub Command1_Click()
   If Option1.Value = True Then
      IVA2.EURCAN.Text = ""
      IVA2.PESCAN.Text = IVA1.Text1.Text
   End If
   If Option2.Value = True Then
      IVA2.PESCAN.Text = ""
      IVA2.EURCAN.Text = IVA1.Text1.Text
   End If
   If IVA2.PESCAN.Text = "" Then IVA2.PESCAN.Text = 166.386 * IVA2. EURCAN.Text
   If IVA2.EURCAN.Text = "" Then IVA2.EURCAN.Text = IVA2.PESCAN.Text /166.386
   If IVA1.Option3.Value = True Then
      IVA2.PESIVA.Text = IVA2.PESCAN.Text * 0.04
      IVA2.EURIVA.Text = IVA2.EURCAN.Text * 0.04
   End If
```

```
   If IVA1.Option4.Value = True Then
      IVA2.PESIVA.Text = IVA2.PESCAN.Text * 0.07
      IVA2.EURIVA.Text = IVA2.EURCAN.Text * 0.07
   End If
   If IVA1.Option5.Value = True Then
      IVA2.PESIVA.Text = IVA2.PESCAN.Text * 0.16
      IVA2.EURIVA.Text = IVA2.EURCAN.Text * 0.16
   End If
   IVA2.PESPVP.Text = CDbl(IVA2.PESCAN.Text) + CDbl(IVA2.PESIVA.Text)
   IVA2.EURPVP.Text  =  CDbl ( IVA2.EURCAN.Text)  +  CDbl ( IVA2 .EURIVA .Text)
End Sub
```

Bien, antes de nada, vamos a intentar entender el procedimiento anterior...

-

```
If Option1.Value = True Then
      IVA2.EURCAN.Text = ""
      IVA2.PESCAN.Text = IVA1.Text1.Text
End If
```

Quiere decir: Si el primer **OptionButton** (PTAS) está activado (Option1.Value = True), entonces en el TextBox: PESETAS/CANTIDAD deIVA2, se escribirá la cantidad que tenemos escrita en el TextBox de IVA1 (IVA2.PESCAN.Text = IVA1.Text1.Text)

- El siguiente **If-Then-End If** del procedimiento hace lo mismo que lo anterior, pero en el caso de EUROS: Escribe la cantidad que tenemos en el TextBox de IVA1 en el campo EUROS/CANTIDAD del IVA2.

-

```
If IVA2.PESCAN.Text = "" Then
         IVA2.PESCAN.Text = 168 * IVA2.EURCAN.Text
```

Si en el campo PESETAS/CANTIDAD de IVA2 no hay nada escrito, entonces escribe la cantidad en EUROS por 166.386. Estamos suponiendo que **1 Euro = 166.386 pts.**

- El siguiente **If Then- End If** hace lo mismo que lo anterior, pero en el campo EUROS/CANTIDAD. **1 Pta = 1/166.386 Euros**

-

```
If IVA1.Option3.Value = True Then
      IVA2.PESIVA.Text = IVA2.PESCAN.Text * 0.04
      IVA2.EURIVA.Text = IVA2.EURCAN.Text * 0.04
   End If
```

Si está activado el OptionButton correspondiente al IVA= 4%. En los campos PESETAS/IVA y EUROS/IVA de IVA2 se calcula el valor del **iva** correspondiente.

Exactamente lo mismo sucede para los 2 siguientes **If – Then**, pero para el 7% y el 16%.

- En las dos últimas líneas del procedimiento, sumamos los campos PESETAS/CANTIDAD y PESETAS/IVA por un lado y los campos EUROS/CANTIDAD y EUROS/IVA por otro.

Observa que utilizamos una nueva función incorporada: **CDbl**. Utilizamos esta nueva función que convierte el contenido de un cuadro de texto, en número tipo **Double**. En otros proyectos no nos habíamos preocupado de la "conversión", porque al multiplicar o dividir un cuadro de texto por un número, el VB los convierte implícitamente. En las últimas líneas del programa anterior hemos de convertir los dos cuadros de texto explícitamente con la función **CDbl**, porque sumamos. El signo de sumar en dos cuadros de texto (si no los convertimos a numérico), es equivalente a concatenar los valores de los dos cuadros; como puedes comprobar fácilmente si en el programa anterior eliminas los dos **CDbl**. El problema se ha presentado por primera vez en este programa, porque es el primero en que no

utilizamos variables que declaramos previamente (utilizamos directamente los nombres de los diferentes controles del formulario) y además necesitamos "sumar" precisamente.

Sólo nos queda ejecutar nuestro programa y probarlo exhaustivamente.
Por último vuelve a grabar el programa con el mismo nombre: **Prog65a.frm, Prog65b.frm, Prog65.vbp**.

b) Haz un nuevo proyecto con los siguientes objetos y propiedades:

- **Form1:**
  Caption: MEDIA ARITMÉTICA
  Posición: 864, 1080
  Tamaño: 4176 x 3636
- **Label1:**
  Caption: Número:
  Posición: 108, 216
  Tamaño: 744 x 252
- **Label2:**
  Caption: Lugar:
  Posición: 240, 2736
  Tamaño: 552 x 276
- **Label3:**
  Caption: Valor:
  Posición: 1980, 2748
  Tamaño: 552 x 264
- **Text1:**
  Text: (borra el texto que aparece por defecto)
  Posición: 816, 204
  Tamaño: 888 x 288
- **Text2:**
  Text: (borra el texto que aparece por defecto)
  Posición: 2760, 2160
  Tamaño: 888 x 312
- **Text3:**
  Text: (borra el texto que aparece por defecto)
  Posición: 840, 2760
  Tamaño: 732 x 288
- **Text4:**
  Text: (borra el texto que aparece por defecto)
  Posición: 2640, 2760
  Tamaño: 972 x 288
- **Command1:**
  Caption: &Introduce el número
  Posición: 132, 612
  Tamaño: 1656 x 384

  Recuerda que al escribir el símbolo ampersan (&) a la izquierda de la letra **"I"**, ésta aparece subrayada. En tiempo de ejecución será equivalente: hacer CLIC en este botón o pulsar las teclas **[ALT][I]**
- **Command2:**
  Caption: &Elimina el número seleccionado
  Posición: 132, 1068
  Tamaño: 1656 x 396
- **Command3:**
  Caption: &Borra el listado
  Posición: 120, 1572
  Tamaño: 1680 x 360
- **Command4:**

Caption: &SALIR
Posición: 108, 2160
Tamaño: 972 x 372

- **Command5:**
  Caption: &Media Aritmética
  Posición: 1308, 2136
  Tamaño: 1428 x 396

- Vamos a introducir un último control...
  - CLIC en el icono **"ListBox":** del "Cuadro de Controles" 
  - Sitúa en el **form**, un rectángulo de posición: 2280, 240 y tamaño: 1332 x 1584

- Graba el **"form"** con el nombre **EJER10A.FRM** y el proyecto con el nombre **EJER10A.VBP** en tu carpeta.

El control **ListBox** (Cuadro de Lista) sirve para guardar un listado o conjunto de valores. La gestión del **ListBox** viene determinado (igual que cualquier otro control), por sus **propiedades** y **métodos**.

Tenemos claro qué son las **propiedades** de un control, pero qué son sus **métodos**...

Los métodos son procedimientos asociados a los objetos, que permiten realizar alguna acción sobre el propio objeto al que pertenecen. Cada objeto tiene su propio repertorio de métodos, que actúan en función del tipo de objeto. Existen métodos que permiten mover los objetos, dibujar sobre ellos, pasarles el foco, etc.

Por ejemplo:
Cuando escribimos: **Form1.Show**, estamos invocando un **MÉTODO** asociado a los formularios, que consiste en visualizar el formulario correspondiente (el Form1, en este caso). En cambio, cuando escribimos: **Form1.Caption**, nos estamos refieriendo a una **PROPIEDAD** del objeto.

En el caso de nuestro **ListBox**, nos interesará trabajar con las siguientes **propiedades** y **métodos**:

- **PROPIEDADES:**
  - ListCount
    List1.ListCount: nos da el número de elementos del cuadro de lista.
  - ListIndex
    Si tenemos seleccionado un elemento del listado, la propiedad **ListIndex** nos da el lugar que ocupa en el listado. El primer lugar del cuadro de lista, corresponde a ListIndex=0
  - List
    List1.List(7) representa el valor que hay en el lugar número 8 del "cuadro de lista".

- **MÉTODOS**
  - AddItem
    List1.AddItem "3572" es un método (procedimiento), que agrega el valor **3572** al final del cuadro de lista **List1**.
  - RemoveItem
    List1.RemoveItem 7 = borra el valor que está en el lugar nº 7 de la lista
  - Clear:
    List1.Clear = borra el contenido de toda la lista

Vamos a escribir el código de nuestro programa...

Escribe en la **ventana de código** los siguientes **procedimientos de evento**:

```
Private Sub Command1_Click()
        If Text1<>"" Then
                List1.AddItem Text1
                Text1.Text=""
        End If
End Sub
```

Es decir:

Al hacer CLIC en el botón **[Introduce el número]** (o al pulsar [ALT][I]), si en el cuadro de texto **Text1** hay alguna cosa escrita (Text1<>""), entonces se añade al cuadro de lista el valor que hay escrito (List1.AddItem Text1) y a continuación se borra el valor que había en el cuadro de texto (Text1.Text="").

Ejecuta el programa y prueba el funcionamiento del botón [Introduce el número].

Continua escribiendo:

```
Private Sub Command2_Click()
        If List1.ListIndex >= 0 Then
                List1.RemoveItem List1.ListIndex
        End If
End Sub
```

Es decir:

Al hacer CLIC en el botón [Elimina el número seleccionado], si hay algún elemento seleccionado en la lista (List1.ListIndex >= 0) entonces lo elimina (List1.RemoveItem List1.ListIndex).

Ejecuta el programa y prueba el funcionamiento del botón [Elimina el número seleccionado]

```
Private Sub Command3_Click()
 Dim Respuesta As Integer
 Dim Advertencia As String
 Advertencia = "¿Seguro que desea borrar toda la lista?"
 Respuesta = MsgBox(Advertencia, vbYesNo +
          vbExclamation, "ATENCIÓN")
 If Respuesta = vbYes Then List1.Clear
End Sub
```

Veamos:

- El segundo argumento de nuestro MsgBox: **vbYesNo + vbExclamation**, significa que en el cuadro de diálogo aparecerán los botones [Sí], [No] y también el signo de exclamación.
- Si hacemos CLIC en el botón [Sí], el cuadro **MsgBox** devolverá el valor **vbYes**
- Observa la orden **List1.Clear**, que borrará definitivamente el contenido del cuadro de lista **List1**.
- Ejecuta el programa y prueba el funcionamiento del botón [Borra el listado].

```
Private Sub Command4_Click()
        End
End Sub
```

Ejecuta el proyecto y prueba el botón [SALIR].

```
Private Sub Command5_Click()
   Dim Total As Double
```

```
   Dim Actual As Integer
   Total = 0
   Actual = 0
   Do While Actual < List1.ListCount
            Total = Total + CDbl(List1.List(Actual))
            Actual = Actual + 1
   Loop
   If List1.ListCount > 0 Then
            Text2 = Total / List1.ListCount
   End If
End Sub
```

Es decir:

- Definimos dos variables: **Total** y **Actual**, la primera servirá para acumular la suma de los valores del cuadro de lista y la segunda variable hará de contador.
- Inicializamos las dos variables a 0
- La estructura **Do While – Loop** acumulará en la variable **Total**, la suma de todos los valores del cuadro de lista.
- Por último, escribimos en el cuadro de texto **Text2**, la media aritmética de los números del **List1**.
- Ejecuta el programa para probar el funcionamiento del botón [Media Aritmética].

- Nos interesa que al escribir un número en el campo **"Lugar"** (Text3), automáticamente aparezca en el campo **"Valor"** (Text4), el número correspondiente del "cuadro de lista"...

  Escribe el siguiente procedimiento de evento:

```
Private Sub Text3_Change()
 Dim indice As Integer
 indice = Val(Text3)
 If List1.ListCount > indice Then
            Text4 = List1.List(Val(Text3))
 End If
End Sub
```

- Graba el formulario y el proyecto como **Prog66.frm, Prog66.vbp**

- Ejecuta el programa y pruébalo exhaustivamente.

Al probar el programa te habrás dado cuenta de la incomodidad que representa la forma de introducir números en el "cuadro de lista".

Vamos a solucionar el problema...

- Accede a la ventana de código y escribe el siguiente procedimiento de evento:

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
        If KeyAscii = 13 Then Command1_Click
End Sub
```

- **KeyPress** es el evento: "Al pulsar una tecla"
- **If KeyAscii = 13 Then Command1_Click**, es decir: Si pulsamos la tecla [Return] (Ascii = 13) entonces se ejecuta el procedimiento de evento asociado al botón [Introduce el número]

- Vuelve a grabar el formulario y el proyecto con el mismo nombre.

- Vuelve a probar “nuestro proyecto”, pero sin utilizar el botón [Introduce el número]. Basta que pulses [Return], cuando hayas acabado de escribir un número en el campo **“Número:”**.

c) Haz un nuevo proyecto con los siguientes objetos y características:

- Form1:
  Caption: COLORES
  Posición: 828, 1056
  Tamaño: 3192 x 2592
- Command1:
  Caption: Borra los Colores
  Posición: 120, 120
  Tamaño: 1812 x 372
- Frame1:
  Caption: COLORES
  Posición: 120, 600
  Tamaño: 1332 x 1332
- Option1:
  Caption: Verde
  Name: OptColor
  Index: 0
  Posición: 120, 240
  Tamaño: 1092 x 252
- Option2:
  Caption: Rojo
  Name: OptColor
  Index: 1
  Posición: 120, 600
  Tamaño: 1092 x 252
- Option3:
  Caption: Azul
  Name: OptColor
  Index: 2
  Posición: 120, 960
  Tamaño: 1092 x 252
- Text1:
  Text: (nada)
  Index: 0
  Posición: 1680, 840
  Tamaño: 1212 x 288

- Text2:
  Text: (nada)
  Name: Text1
  Index: 1
  Posición: 1680, 1200
  Tamaño: 1212 x 288

- Text3:
  Text: (nada)
  Name: Text1
  Index: 2
  Posición: 1680, 1560
  Tamaño: 1212 x 288

Observa que los “Options” son una matriz de objetos y los “textbox” otra

- Hemos creado el vector de objetos: **OptColor** con 3 valores: OptColor(0), OptColor(1) y OptColor(2) y el vector: **Text1** con 3 valores también: Text1(0), Text1(1), Text1(2).
- La forma de crearlos ha sido simplemente darles el mismo **Name** y utilizar la propiedad **Index** (0, 1 o 2).

Escribe el siguiente procedimiento:

```
Private Sub OptColor_Click(Index As Integer)
   Dim j As Integer
   Select Case Index
      Case 0
         Text1(Index).BackColor = RGB(0, 255, 0)
      Case 1
         Text1(Index).BackColor = RGB(255, 0, 0)
      Case 2
         Text1(Index).BackColor = RGB(0, 0, 255)
   End Select
   For j = 0 To 2
      If j <> Index Then Text1(j).BackColor = RGB(255, 255, 255)
   Next j
End Sub
```

- Observa:
  - La función **RGB(, ,)** es un color que en el caso de los parámetros **0, 255, 0** corresponde al verde. Si los parámetros son **255, 0, 0** es el rojo y en el caso **0, 0, 255** es el azul. Si los tres parámetros son iguales a **255**, corresponde al color blanco.

  - El funcionamiento del procedimiento es simple: según el botón de opción que seleccionemos en el cuadro correspondiente (mismo “Index”) aparecerá el color seleccionado.

    Y escribe el siguiente procedimiento:

```
Private Sub Command1_Click()
   Dim i As Integer
   For i = 0 To 2
      Text1(i).BackColor = RGB(255, 255, 255)
      OptColor(i).Value = False
   Next
End Sub
```

- **Ejecuta el programa y pruébalo (espero que te funcione).**

- Graba el formulario con el nombre **Prog67.frm** y el proyecto con el nombre **Prog67.vbp** en *TuCarpeta*.

d) Vamos a hacer un nuevo proyecto con una lista origen con los nombre de los doce meses, en la que es posible seleccionar un o varios elementos a la vez y pasarlos a una segunda lista destino.

Crea un nuevo proyecto y

1. Añadir al formulario por defecto dos cuadros de lista y tres botones de comando.

2. Especificar las siguientes propiedades en tiempo de diseño:

Objeto	Propiedad	Valor
Form1	Name	frmLista
	Caption	Ejemplo de cuadros de lista
List1	Name	lstOrigen
	MultiSelect	2 (Extended)
List2	Name	lstDestino
Command1	Name	cmdAñadir
	Caption	&Añadir
Command2	Name	cmdVaciar
	Caption	&Vaciar
Command3	Name	cmdSalir
	Caption	&Salir

3. Escribir los siguientes procedimientos de evento:

```
Private Sub Form_Load()
  lstOrigen.AddItem "Enero"
  lstOrigen.AddItem "Febrero"
  lstOrigen.AddItem "Marzo"
  lstOrigen.AddItem "Abril"
  lstOrigen.AddItem "Mayo"
  lstOrigen.AddItem "Junio"
  lstOrigen.AddItem "Julio"
  lstOrigen.AddItem "Agosto"
  lstOrigen.AddItem "Septiembre"
  lstOrigen.AddItem "Octubre"
  lstOrigen.AddItem "Noviembre"
  lstOrigen.AddItem "Diciembre"
End Sub

Private Sub cmdAñadir_Click()
  Dim N As Integer
  'Para cada elemento de la lista origen
  For N = 0 To lstOrigen.ListCount - 1
    'Si se encuentra seleccionado
    If lstOrigen.Selected(N) Then
      'Añadirlo a la lista destino
      lstDestino.AddItem lstOrigen.List(N)
    End If
```

```
  Next
End Sub

Private Sub cmdVaciar_Click()
  lstDestino.Clear
End Sub

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

OBSERVA

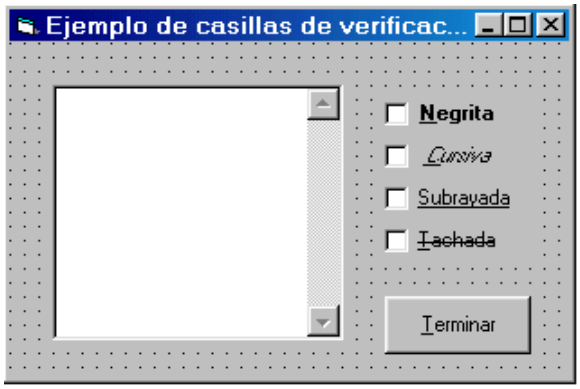
Para poder seleccionar varios elementos, ya sea individualmente o un rango, debe activarse el modo de selección extendido de la lista origen. Para pasar sólo los elementos seleccionados a la otra lista puede consultarse la propiedad Selected de cada elemento.

Graba el programa como **Prog68.frm**
Prog68.vbp

e) Vamos a hacer un programa que contenga un texto sobre el que es posible especificar distintos efectos de fuente (negrita, cursiva, etc.). Para ello, bastará con activar la casilla correspondiente.

Crea un nuevo proyecto

1. Añadir al formulario por defecto un cuadro de texto, cuatro casillas de verificación y un botón de comando.



2. Especificar las siguientes propiedades en tiempo de diseño:

Objeto	Propiedad	Valor
Form1	Name	frmCasillas
	Caption	Ejemplo de casillas de verificación
Text1	Name	txtDemo
	MultiLine	True
	ScrollBars	2 (Vertical)
Check1	Name	chkNegr
	Caption	&Negrita
	FontBold	True
Check2	Name	chkCurs
	Caption	&Cursiva
	FontItalic	True
Check3	Name	chkSubr
	Caption	&Subrayada
	FontUnderline	True

Check4	Name	chkTach
	Caption	&Tachada
	FontStrikethru	True
Command1	Name	cmdTerm
	Caption	&Terminar

3. Escribir los siguientes procedimientos de evento:

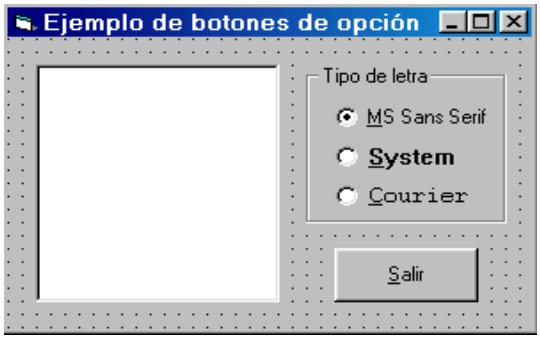
```
Private Sub chkCurs_Click()
  txtDemo.Font.Italic = Not txtDemo.Font.Italic
End Sub

Private Sub chkNegr_Click()
  txtDemo.Font.Bold = Not txtDemo.Font.Bold
End Sub

Private Sub chkSubr_Click()
  txtDemo.Font.Underline = Not txtDemo.Font.Underline
End Sub

Private Sub chkTach_Click()
  txtDemo.Font.Strikethrough = Not txtDemo.Font.Strikethrough
End Sub

Private Sub cmdTerm_Click()
  End
End Sub
```

Graba el programa como **Prog69.frm**
Prog69.vbp

f) Vamos a hacer un programa con un texto sobre el que es posible especificar una de tres fuentes posibles para su visualización. Para ello, bastará con activar la opción correspondiente.

Crea un nuevo proyecto

1. Añadir al formulario por defecto un cuadro de texto, un marco, tres botones de opción y un botón de comando.



2. Especificar las siguientes propiedades en tiempo de diseño:

Objeto	**Propiedad**	**Valor**
Form1	Caption	Ejemplo de botones de opción
Frame1	Name	fraLetra
	Caption	Tipo de letra
Option1	Name	optSerif
	Caption	&MS Sans Serif
	Value	True

Option2	Name	optCourier
	Caption	&Courier
	FontName	Courier
Option3	Name	optSystem
	Caption	&System
	FontName	System
Text1	Name	txtDemo
	MultiLine	True
Command1	Name	cmdSalir
	Caption	&Salir

3. Escribir los siguientes procedimientos de evento:

```
Private Sub optCourier_Click()
  txtDemo.Font.Name = "Courier"
End Sub

Private Sub optSerif_Click()
  txtDemo.Font.Name = "MS Sans Serif"
End Sub

Private Sub optSystem_Click()
  txtDemo.Font.Name = "System"
End Sub

Private Sub cmdSalir_Click()
  End
End Sub
```

OBSERVA
Para el correcto funcionamiento del ejemplo, los botones de opción deben estar contenidos en el "frame", y no simplemente situados encima.

Graba el programa como **Prog70.frm**
Prog70.vbp

g) Vamos a hacer un programa que muestra como puede utilizarse un formulario para crear un cuadro de diálogo personalizado (no será más que otro formulario), permitiendo al usuario ver e introducir determinada información.

Aunque el cuadro de diálogo no realize ninguna tarea concreta, ilustra el aspecto y los controles que pueden utilizarse para que el usuario introduzca información, en este caso acerca de un ordenador.

Crea un nuevo proyecto

1. Añadir al formulario por defecto un cuadro de dibujo y dos botones de comando.

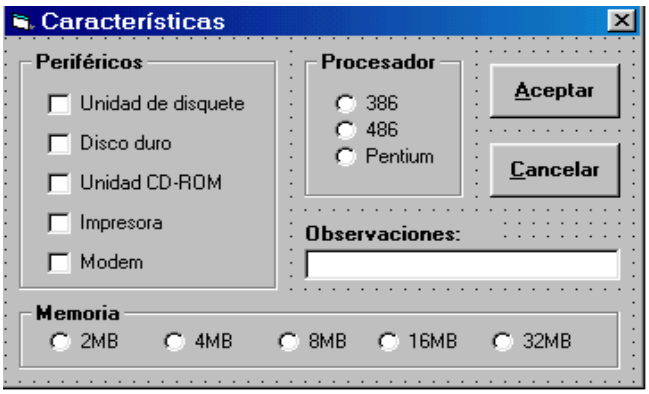
2. Especificar las siguientes propiedades en tiempo de diseño:

Objeto	Propiedad	Valor
Form1	Name	frmOrdenad
	Caption	Ordenador
Command1	Caption	&Especificar las características
	FontBold	True
Command2	Caption	&Salir
Picture1	BackColor	&H00FFFFFF&
	Picture	METAFILE\BUSINESS\PCOMPUTR.WMF

3. Escribir los siguientes procedimientos de evento:

```
Private Sub Command1_Click()
  frmCaract.Show 1
End Sub

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

4. Añadir un segundo formulario al proyecto incluyendo en él tres marcos, nueve botones de opción, cinco casillas de verificación, una etiqueta, un cuadro de texto y dos botones de comando.



5. Especificar para ellos las siguientes propiedades en tiempo de diseño:

Objeto	Propiedad	Valor
Form2	Name	frmCaract
	BorderStyle	3 (Fixed Dialog)
	Caption	Características
	MaxButton	False
	MinButton	False
	ShowInTaskbar	False
Frame1	Caption	Procesador
	FontBold	True
Option1	Caption	386
Option2	Caption	486
Option3	Caption	Pentium
Frame2	Caption	Periféricos
	FontBold	true
Check1	Caption	Unidad de disquete
Check2	Caption	Disco duro
Check3	Caption	Unidad CD-ROM
Check4	Caption	Impresora
Check5	Caption	Modem
Frame3	Caption	Memoria
	FontBold	True
Option4	Caption	2MB

Option5	Caption	4MB
Option6	Caption	8MB
Option7	Caption	16MB
Option8	Caption	32MB
Option9	Caption	64MB
Label1	Caption	Observaciones:
	FontBold	True
Text1	Text	(vacío)
Command1	Caption	&Aceptar
	FontBold	True
Command2	Caption	&Cancelar
	FontBold	True

5. Escribir los siguientes procedimientos de evento:

```
Private Sub Command1_Click()
  Unload Me
End Sub

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

6. Guardar los formularios y el proyecto en los archivos **Prog71a.frm, Prog71b.frm, Prog71.vbp**, respectivamente.

OBSERVA
El primer marco deben incluirse 3 botones de opción, para seleccionar un tipo de procesador. En el segundo se incluyen las casillas de verificación. Finalmente, en el tercero se incluyen los otros seis botones de opción, para seleccionar una configuración de memoria.

h) Vamos a ver un programa que muestra como puede utilizarse un cuadro combinado (ComboBox) para seleccionar un elemento de una lista desplegable de opciones.

Se trata de una lista desplegable con los nombres de varios países. Al seleccionar uno de ellos, se visualiza a su derecha un icono representativo del mismo, el problema es que necesitaremos 10 iconos de paises.

Crea un nuevo proyecto

1. Añadir al formulario por defecto un cuadro combinado, una etiqueta y diez imágenes.

2. Especificar las siguientes propiedades en tiempo de diseño:

Objeto	**Propiedad**	**Valor**
Form1	Caption	Paises
	Icon	EARTH.ICO

Combo1	Name	cmbPais
	Style	2 (Dropdown List)
Label1	Caption	Seleccione un país:
Image1	Name	imgPais
Image2	Name	imgMatriz
	Index	1
	Picture	CTRGERM.ICO
Image3	Name	imgMatriz
	Index	2
	Picture	CTRCAN.ICO
Image4	Name	imgMatriz
	Index	3
	Picture	CTRSPAIN.ICO
Image5	Name	imgMatriz
	Index	4
	Picture	CTRUSA.ICO
Image6	Name	imgMatriz
	Index	5
	Picture	CTRITALY.ICO
Image7	Name	imgMatriz
	Index	6
	Picture	CTRUK.ICO
Image8	Name	imgMatriz
	Index	7
	Picture	CTRJAPAN.ICO
Image9	Name	imgMatriz
	Index	8
	Picture	CTRFRAN.ICO
Image10	Name	imgMatriz
	Index	9
	Picture	CTRMEX.ICO

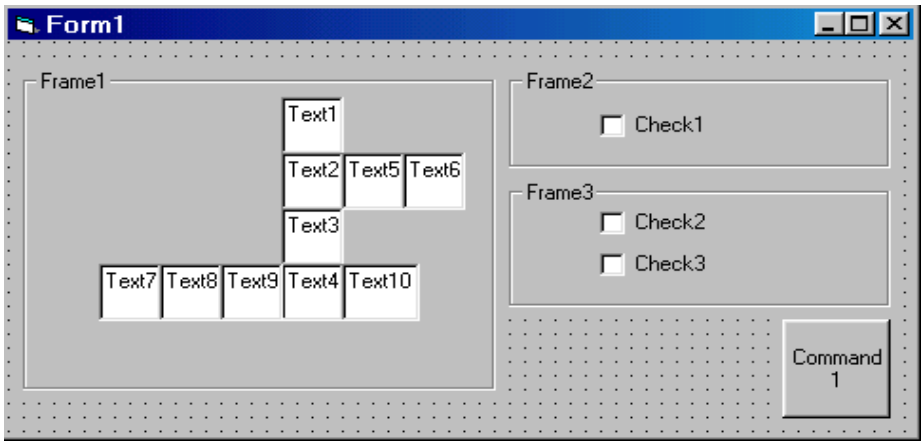
3. Escribir los siguientes procedimientos de evento:

```
Private Sub cmbPais_Click()
  imgPais.Picture = imgMatriz(cmbPais.ListIndex)
End Sub


Private Sub Form_Load()
  'cargar los nombres de los paises
  cmbPais.AddItem "Alemania"
  cmbPais.AddItem "Canadá"
  cmbPais.AddItem "España"
  cmbPais.AddItem "Estados Unidos"
  cmbPais.AddItem "Inglaterra"
  cmbPais.AddItem "Francia"
  cmbPais.AddItem "Italia"
  cmbPais.AddItem "Méjico"
  cmbPais.AddItem "Japón"
  'Seleccionar el primero
  cmbPais.ListIndex = 0
End Sub
```

Graba el programa con el nombre **Prog72.frm**
Prog72.vbp

i) Vamos a programar un crucigrama. Crea un nuevo proyecto.

– Inserta:



- Propiedades:

* Frame1
Caption =

* Frame2
Caption = Verticales
ForeColor = Rojo
Font = Bold 10

* Frame3
Caption = Horizontales
ForeColor = Verde
Font = Bold 10

* Todos los TextBoxs (del 1 al 10)
Text =
Font = Bold 18

* Check1
Caption = Animal que lo repite todo

* Check2
Caption= Metal Precioso

* Check3
Caption = Animal que come queso

* Command1
Picture = TRFFC02.ICO
Style = 1 – Graphical
Caption = SALIR

- Inserta 3 PictureBox, donde insertaremos las 3 caras:

* Picture1
Picture = Face01.ico

* Picture2
Picture = Face03.ico

* Picture3
Picture = Face04.ico

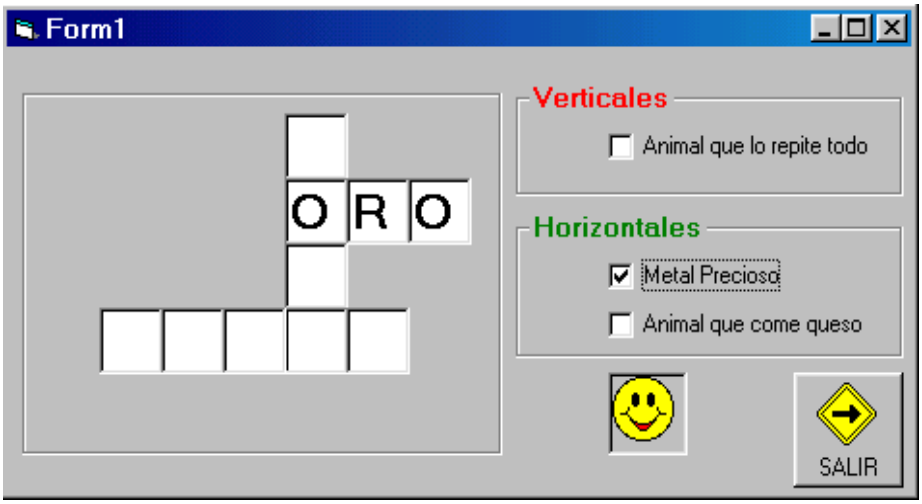
- Código:

```
Private Sub Command1_Click()
   End
End Sub

Private Sub Check1_Click()
   If UCase(Text1.Text) = "L" And UCase(Text2.Text) = "O" _
      And UCase(Text3.Text) = "R" And UCase(Text4.Text) = "O" Then
      Picture1.Visible = False
      Picture2.Visible = True
   Else
      Picture3.Visible = True
   End If
End Sub

Private Sub Check2_Click()
   If UCase(Text2.Text) = "O" And UCase(Text5.Text) = "R" _
      And UCase(Text6.Text) = "O" Then
```

```
      Picture1.Visible = False
      Picture2.Visible = True
   Else
      Picture3.Visible = True
   End If
End Sub

Private Sub Check3_Click()
   If UCase(Text7.Text) = "R" And UCase(Text8.Text) = "A" _
      And UCase(Text9.Text) = "T" And UCase(Text4.Text) = "O" _
      And UCase(Text10.Text) = "N" Then
      Picture1.Visible = False
      Picture2.Visible = True
   Else
      Picture3.Visible = True
   End If
End Sub

Private Sub Form_Load()
   Picture2.Visible = False
   Picture3.Visible = False
End Sub
```

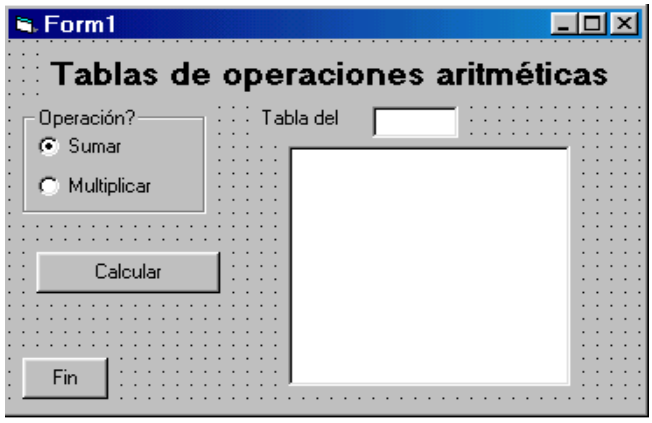
- Grábalo como **Prog73.frm, Prog73.vbp**

- El programa en funcionamiento:

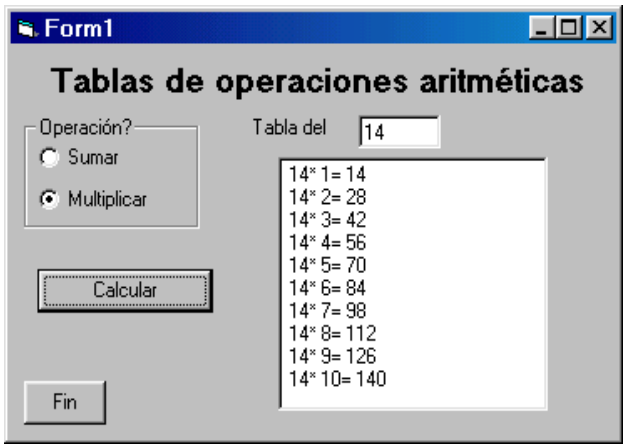


j) Crea un nuevo proyecto

- Inserta



- Queremos conseguir:



Para ello basta que escribas el código:

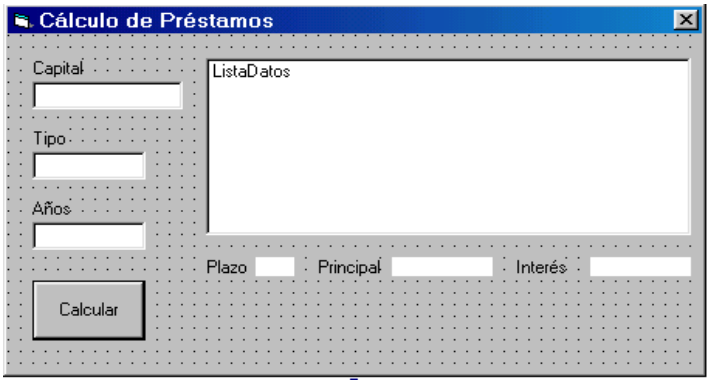
```
Sub llenartabla()
   Dim i As Integer
   Dim numtau As Integer
   Dim taula As String
   numtau = Val(Text1)
   If Option1 Then
      For i = 1 To 10
         taula = taula + Str$(numtau) + "+" + Str$(i) + "="
         taula = taula + Str$(numtau + i) + Chr$(13) + Chr$(10)
      Next i
   Else
      For i = 1 To 10
         taula = taula + Str$(numtau) + "*" + Str$(i) + "="
         taula = taula + Str$(numtau * i) + Chr$(13) + Chr$(10)
      Next i
   End If
   Text2.Text = taula
End Sub

Private Sub Command1_Click()
   llenartabla
End Sub

Private Sub Command2_Click()
End
End Sub
```

```
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
   If KeyCode = 13 Then llenartabla
End Sub
```

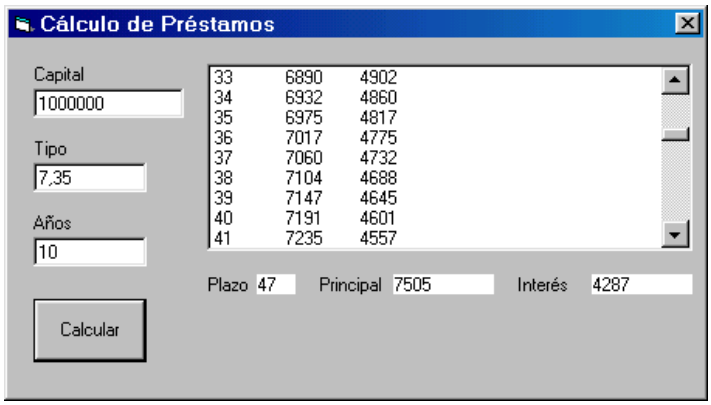
Graba el programa como **Prog74.frm, Prog74.vbp**

k) Crea un nuevo proyecto

Inserta:



Queremos conseguir:



Para ello has de escribir el siguiente código:

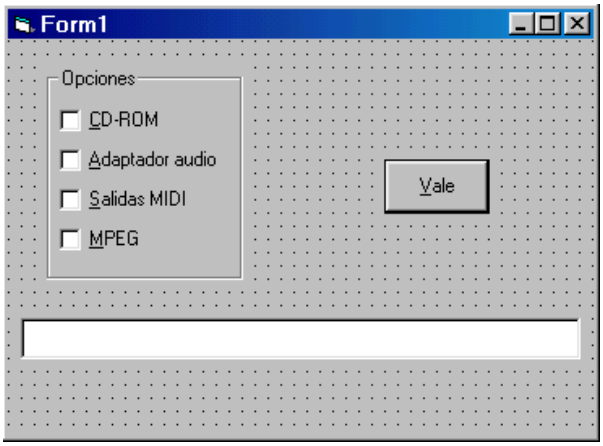
```
   ' Se ha pulsado el botón calcular
Private Sub Calcular_Click()
   Dim nPlazos As Integer, I As Integer

   nPlazos = Años * 12 ' Obtener el número de meses

   ListaDatos.Clear ' Limpiar la lista de datos
      ' Añadir una cabecera
   ListaDatos.AddItem "Plazo" & Chr(9) & "Principal" & Chr(9) & "Intereses"
   For I = 1 To nPlazos ' Recorrer todos los meses
         ' añadiendo el principal y los intereses mensuales
      ListaDatos.AddItem I & Chr(9) & Abs(CLng(PPmt(Tipo / 100 / 12, I, nPlazos, Capital))) & _
                  Chr(9) & Abs(CLng(IPmt(Tipo / 100 / 12, I, nPlazos, Capital)))
   Next
End Sub
   ' La aplicación destino  envía un comando
Private Sub Form_LinkExecute(CmdStr As String, Cancel As Integer)
   If Left(UCase(CmdStr), 6) = "ACTIVA" Then ' Si el comando es ACTIVA
      Dim I As Integer
      I = Mid(CmdStr, 8) ' Tomar el número de plazo a activar
```

```
      If I > 0 And I <= Años * 12 Then ' Si el plazo existe
         ListaDatos.ListIndex = I ' activarlo
         Cancel = False ' e indicar que el comando se ha ejecutado
      End If
   End If
   If UCase(CmdStr) = "CALCULA" Then ' Si lo que se quiere es recalcular
      Calcular_Click ' Llamar al procedimiento correspondiente
      Cancel = False ' Indicar que el comando se ha ejecutado
   End If
End Sub

   ' Al pulsar sobre un elemento de la lista
Private Sub ListaDatos_Click()
   If ListaDatos.ListIndex <> 0 Then ' Si se ha elegido una fila de datos
      Plazo = ListaDatos.ListIndex ' Mostrar los datos correspondientes a ella
      Principal = Abs(CLng(PPmt(Tipo / 100 / 12, ListaDatos.ListIndex, Años * 12, Capital)))
      Interés = Abs(CLng(IPmt(Tipo / 100 / 12, ListaDatos.ListIndex, Años * 12, Capital)))
   End If
End Sub
```

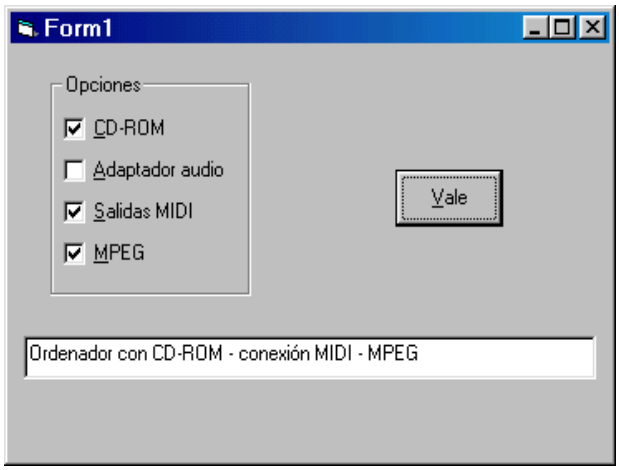
Graba el programa como **Prog75.frm, Prog75.vbp**

l) Crea un nuevo proyecto.

Inserta:



Se trata de conseguir:



Para ello considera el siguiente código:

```
Private Sub Vale_Click()
   Dim Cadena As String, Separador As String
   Cadena = "Ordenador con "
   Separador = ""
   If OpCD.Value = 1 Then Cadena = Cadena & "CD-ROM": Coma = " - "
   If OpAudio.Value = 1 Then Cadena = Cadena & Coma & "tarjeta de sonido": Coma = " - "
   If OpMIDI.Value = 1 Then Cadena = Cadena & Coma & "conexión MIDI": Coma = " - "
   If OpMPEG.Value = 1 Then Cadena = Cadena & Coma & "MPEG"
   Equipo.Text = Cadena
End Sub
```

Graba el programa como **Prog76.frm, Prog76.vbp**

m) Crea un nuevo proyecto

Inserta:

Queremos conseguir:

Escribe el siguiente código:

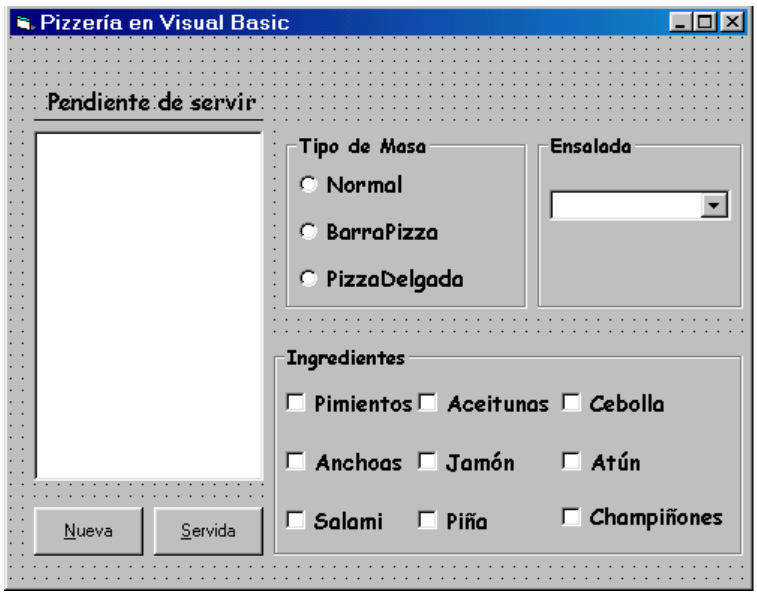
```
Const PVP = 300000
Const supMemo = 15000
Const supDVD = 20000
Const supKit = 30000
Const supMonitor = 50000
```

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

Private Sub cbOpciones_Click(Index As Integer)
   Select Case Index
      Case 0
            If cbOpciones(0).Value = 1 Then
               lbSuplemento.Caption = supMemo
               lbTotal.Caption = lbTotal.Caption + supMemo
            Else
               lbSuplemento.Caption = supMemo
               lbTotal.Caption = lbTotal.Caption - supMemo
            End If
      Case 1
            If cbOpciones(1).Value = 1 Then
               lbSuplemento.Caption = supDVD
               lbTotal.Caption = lbTotal.Caption + supDVD
            Else
               lbSuplemento.Caption = supDVD
               lbTotal.Caption = lbTotal.Caption - supDVD
            End If
      Case 2
            If cbOpciones(2).Value = 1 Then
               lbSuplemento.Caption = supKit
               lbTotal.Caption = lbTotal.Caption + supKit
            Else
               lbSuplemento.Caption = supKit
               lbTotal.Caption = lbTotal.Caption - supKit
            End If
      Case 3
            If cbOpciones(3).Value = 1 Then
               lbSuplemento.Caption = supMonitor
               lbTotal.Caption = lbTotal.Caption + supMonitor
            Else
               lbSuplemento.Caption = supMonitor
               lbTotal.Caption = lbTotal.Caption - supMonitor
            End If
   End Select
End Sub


Private Sub Form_Load()
Dim Indice As Integer
   lbTotal = 300000
   For Indice = 0 To 3
      cbOpciones(Indice).Value = vbUnchecked 'vbChecked o 1 se usa para activar, vbUnchecked y
0 para desactivar
   Next
End Sub
```

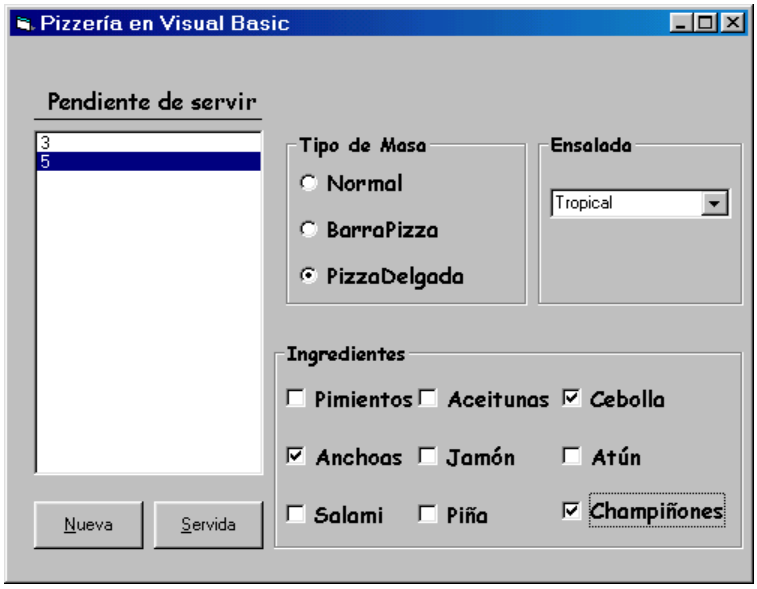
Graba el programa como **Prog77.frm, Prog77.vbp**

n) Crea un nuevo proyecto

Inserta:



Queremos conseguir:



Escribe el siguiente código:

```
Private Type tNotaPedido
   iTipoMasa As Integer
   iIngredientes(0 To 8) As Integer
   iEnsalada As Integer
   'Declaración de tipo de dato. Preparamos la plantilla de 11 casillas,
   'que luego duplicaremos tantas veces como mesas haya.
   'Será donde se almacenen las cosas que ha pedido el cliente.
End Type
Const NumeroMesas = 24
Dim npMesas(1 To NumeroMesas) As tNotaPedido
'Indicamos que queremos tantas filas como mesas haya

Private Sub btnNueva_Click()
   Dim sRespuesta As String
   Dim iIndice As Integer
```

```
   If lbMesas.ListCount = NumeroMesas Then
      MsgBox "No hay mesas disponibles en este momento"
      Exit Sub
   End If
   sRespuesta = InputBox("Número de la mesa", "Nuevo pedido")
   If CInt(sRespuesta) < 1 Or CInt(sRespuesta) > NumeroMesas Then
      MsgBox "Ese número de mesa no es válido"
      Exit Sub
   End If
   'Con estos (2) if controlamos que la mesa introducida sea correcta

   lbMesas.AddItem sRespuesta
   lbMesas.ListIndex = lbMesas.NewIndex
   '(2) Ponemos la respuesta de la mesa que se quiere y la señalamos

   obMasa(0).Value = True 'Activo el 1er valor de la masa (normal)
   cbEnsalada.ListIndex = -1 'No selecciono ninguna ensalada
   For iIndice = 0 To 8
      cbIngrediente(iIndice).Value = 0
      'Ningún ingrediente queda seleccionado
   Next
   btnServida.Enabled = True
   frMasa.Enabled = True
   frIngredientes.Enabled = True
   frEnsalada.Enabled = True
   'Activo los frm y el botón servida para empezar a seleccionar
End Sub

Private Sub btnServida_Click()
   If lbMesas.ListIndex <> -1 Then
      lbMesas.RemoveItem lbMesas.ListIndex
      'Si hay alguna mesa, borro la que está seleccionada
   End If
   If lbMesas.ListCount = 0 Then
      btnServida.Enabled = False
      frMasa.Enabled = False
      frEnsalada.Enabled = False
      frIngredientes.Enabled = False
      'Si no hay ninguna mesa más desactivo todo
   Else
      lbMesas.ListIndex = 0 'Se coloca en la 1ª posición
   End If
End Sub

Private Sub cbEnsalada_Click()
   npMesas(lbMesas.Text).iEnsalada = cbEnsalada.ListIndex
   'Apunto la ensalada seleccionada en la matriz npMesas
End Sub

Private Sub cbIngrediente_Click(Index As Integer)
   npMesas(lbMesas.Text).iIngredientes(Index) = cbIngrediente(Index) 'Esto último dice si está
activado o no
   'Apunto (con 1 ó 0), en la matriz, los ingredientes seleccionados
End Sub

Private Sub lbMesas_Click()
   Dim iIndice As Integer
   obMasa(npMesas(lbMesas.Text).iTipoMasa).Value = True 'Lo que hay entre () indica el indice
de obMasa que tiene que activar
```

```
   cbEnsalada.ListIndex = npMesas(lbMesas.Text).iEnsalada 'Muestra la ensalada que había
seleccionada
   For iIndice = 0 To 8
      cbIngrediente(iIndice).Value = npMesas(lbMesas.Text).iIngredientes(iIndice)
   Next
   'Recorre los ingredientes viendo cuales han sido elegidos y cuales no
End Sub

Private Sub obMasa_Click(Index As Integer)
   npMesas(lbMesas.Text).iTipoMasa = Index
   ' Apunta el índice del botón de radio en la matriz, así conocemos que tipo de masa ha
seleccionado
End Sub
```

Graba el programa como **Prog78.frm, Prog78.vbp**

VIII.- Dibujo

a) Haz un **nuevo proyecto** con las siguientes características:

- Form1:
  Posición: 816, 1032
  Tamaño: 4080 x 4080

- Disponemos de un formulario que tiene su origen en el punto de coordenadas: 816, 1032.
  Veamos:
  a) Observa las siguientes propiedades del form:
     Left: 816
     Top: 1032
  b) Posición: 816, 1032
     Propiedades: Left= 816
     Top= 1032

     Coordenada o columna X = Left = 816 unidades a la derecha
     Coordenada o Línea Y = Top = 1032 unidades hacia abajo.

     Es decir:
     El origen del formulario (extremo superior izquierdo), está situado 816 unidades hacia la derecha del origen de la pantalla (extremo superior izquierdo de la pantalla) y 1032 unidades hacia abajo del origen de la pantalla.

- Pero **¿qué tipo de unidades utiliza el VB?**

  c) El V.B. utiliza una unidad de medida propia llamada **TWIPS**.
  d) La medida en **twips** tiene como finalidad que las dimensiones y posición de los controles en un formulario sean siempre los mismos, aun cuando se trabaje con distintas resoluciones.

- Continuemos observando nuestro formulario:
  e) Tamaño: 4080 x 4080
  f) La primera coordenada (X) nos indica la anchura del formulario: Propiedad **Width** = 4080 twips.
  g) La segunda coordenada (Y) nos indica la altura del formulario: Propiedad **Height** = 4080 twips.

- Coloca un control en nuestro formulario, por ejemplo el siguiente:
  Label1:
  Left: 0
  Top: 0
  Width: 972
  Height: 252
  O lo que es equivalente:
  Posición: 0, 0
  Tamaño: 972 x 252

  h) Tenemos un control (label1) situado en el punto 0, 0 del formulario. Es decir: su extremo superior izquierdo está exactamente en el extremo superior izquierdo del formulario.
  i) Las dimensiones del control son:
  Primera coordenada = X = columnas = Width = 972 twips.
  Segunda coordenada = Y = filas = Height = 252 twips.

- Todo formulario por defecto dispone de una cuadrícula de referencia (los puntitos que aparecen en el form en tiempo de diseño).

    j) La separación entre un punto y el siguiente es de 120 twips.
    k) Todo formulario, por defecto, tiene activada la opción "Forzar a cuadrícula" (Align Controls to Grid). Es decir, al colocar un control en el **form** necesariamente se coloca en uno de los puntos de la cuadrícula. Así como la anchura y altura.
    l) Podemos cambiar estas opciones por defecto si accedemos a:

    Menú Tools
        Options…
        Solapa: General

b) En principìo podemos dibujar en un formulario o en un "PictureBox".

Los **métodos** de dibujo que incorporan estos objetos son: **Circle, Cls, Line, Point, Print, Pset, TextHeight, TextWidth**.
Las **propiedades** (ligadas al hecho de "dibujar" son: **AutoRedraw, CurrentX, CurrentY, DrawMode, DrawStyle, DrawWidth, FillColor, FillStyle, FontBold, FontItalic, FontName, FontSize, FontStrykeTrue, FontTransparent, FontUnderline, ScaleHeight, ScaleLeft, ScaleMode, ScaleTop, ScaleWith, ForeColor** y **ClipControls**.

- Elimina el control **Label1** de nuestro formulario.

- Disponemos de un tablero de dibujo (Form1) de 4080 x 4080 twips en el que nos dedicaremos a dibujar…

- Accede a la ventana de código, selecciona las opciones: Objeto: **Form**, Procedimiento: **Click** y escribe el siguiente procedimiento:

```
Private Sub Form_Click()
   Dim Pi As Double
   Dim i As Integer
   Pi = 4 * Atn(1)
   CurrentX = 0
   CurrentY = 0
    Print "Hola tio/a"
   Line (1000, 2000)-(4000, 1000)
   FillStyle = 1
   Line (400, 300)-(700, 800), QBColor(5), B
   Line (1700, 300)-(2000, 800), RGB(Rnd * 255, Rnd * 255, Rnd * 255), BF
   Line -Step(500, 500)
   Circle (600, 1400), 300, RGB(Rnd * 255, Rnd * 255, Rnd * 255)
   FillColor = QBColor(Rnd * 15)
   FillStyle = 0
   Circle (2000, 1400), 300, QBColor(6), , , 0.8
   Circle (2000, 550), 300, QBColor(6), 0, Pi / 2, 0.8
   For i = 1 To 4500 Step 100
      PSet (i, 2500), QBColor(4)
   Next i
End Sub
```

- Ejecuta el programa y pruébalo. Deberás hacer varios CLICS en el formulario.

c) Para entender las diferentes sentencias que aparecen en el procedimiento de evento **Form_Click** anterior, crearemos un módulo donde podremos estudiar una por una dichas sentencias…

Haz lo siguiente:

- Menú Project
  Add Module
  [Open]

- Escribe en la ventana **Módulo1** el siguiente **procedimiento general**:

```
Public Sub Dibujo1()
 Dim Pi As Double
 Pi = 4*Atn(1)
 Form1.Print Pi
End Sub
```

- Accede a las propiedades del **form** y cambia su propiedad **AutoRedraw** a True.

- Accede a la ventana de código del **form** y escribe el siguiente procedimiento de evento:

```
Private Sub Form_Load()
 Dibujo1
End Sub
```

- Ejecuta el programa

- Veamos:
  m) Definimos la variable **Pi** tipo **Double**
  n) **Atn** es la función matemática "arcotangente". Es decir: **Atn(1) = 45º** (el ángulo cuya tangente es 1 vale 45º) o en radianes: **Atn(1) = Pi/4**. Por lo tanto **4*Atn(1)** nos da el valor matemático del número "pi".

d) Accede a la ventana de código **Módulo1** y escribe el siguiente procedimiento general:

```
Public Sub Dibujo2()
        Form1.CurrentX = 1000
        Form1.CurrentY = 500
        Form1.Print "Hola tio/a"
End Sub
```

- Edita el procedimiento de evento **Form_Load()** y cambia la línea **Dibujo1** por **Dibujo2**

- Ejecuta el programa

- Veamos:
  o) Las propiedades **CurrentX** y **CurrentY** definen el punto en el cual nos queremos colocar.
  p) En nuestro caso en el punto **1000, 500**. Cómo a continuación hay un **Print**, la frase correspondiente se escribe en el punto de coordenadas **X= 1000, Y= 500**.

e) Accede a la ventana de código **Módulo1** y escribe el siguiente procedimiento general:

```
Public Sub Dibujo3()
    Dim X, Y As Integer
    For X= 0 To 3000 Step 500
        For Y= 0 To 3000 Step 200
            Form1.CurrentX = X
            Form1.CurrentY = Y
            Form1.Print "Hola"
        Next Y
    Next X
End Sub
```

- Edita el procedimiento de evento **Form_Load** y cambia **Dibujo2** por **Dibujo3**

- Ejecuta el proyecto.

f) Escribe en el **Módulo1** el siguiente procedimiento:

```
Public Sub Dibujo4()
    Form1.Line (1000, 2000) – (3000, 1000)
    Form1.Print "Hola"
End Sub
```

- Edita el procedimiento de evento **Form_Load** y cambia **Dibujo3** por **Dibujo4**

- Ejecuta el proyecto

- Veamos:
  - **q) Line (1000, 2000) – (3000, 1000)**
    Dibuja una línea desde el punto de coordenadas (1000, 2000) al punto (3000, 1000)
  - r) El método anterior cambia el valor de **CurrentX** y **CurrentY** al extremo de la línea, por esta razón el mensaje **"Hola"** aparece en el punto de coordenadas 3000, 1000.

g) Escribe en el **Módulo1** el siguiente procedimiento:

```
Public Sub Dibujo5()
    Form1.Line (500,250) – (1000, 2000)
    Form1.Line – (0, 0)
End Sub
```

- Edita el procedimiento de evento **Form_Load** y cambia **Dibujo4** por **Dibujo5**

- Ejecuta el proyecto

- Veamos:
  - s) Line (500, 250) – (1000, 2000)
    Dibuja una línea desde el punto (500, 250) hasta el punto 1000, 2000
  - **t) Line – (0, 0)**
    Dibuja una línea desde el estado actual del **CurrentX, CurrentY** (será el extremo de la línea anterior) hasta el punto 0, 0

- Graba el formulario con el nombre **Prog79.frm,** el módulo como **Prog79.bas** y el proyecto como **Prog79.vbp**.

h) Haz un nuevo proyecto con las siguientes características:

- Form1:
  Posición: 1035, 1200
  Tamaño: 4200 x 4980
  AutoRedraw: True
- Command1:
  Caption: Origen
  Posición: 0, 4200
  Tamaño: 732 x 372
- Command2:
  Caption: Mov.Horiz.
  Posición: 840, 4200
  Tamaño: 972 x 372
- Command3:
  Caption: Mov.Vert.
  Posición: 1920, 4200
  Tamaño: 972 x 372
- Command4:
  Caption: Mov.Diag.
  Posición: 3000, 4200
  Tamaño: 972 x 372
- Accede a la "ventana de código" con las opciones:
  Objeto: **Form**
  Procedimiento: **Load**

  Y escribe el siguiente procedimiento de evento:

```
Private Sub Form_Load()
   Dim i, j As Integer
   For i = 0 To 4000 Step 500
      For j = 0 To 4000 Step 500
         PSet (i, j), QBColor(4)
      Next j
   Next i
   CurrentX = 0
   CurrentY = 0
End Sub
```

- Ejecuta el proyecto y observemos el procedimiento:

  u) **Pset(i, j)** dibuja un punto en el punto de coordenadas (i, j). Como i, j=0 To 4000 step 500, en el formulario se dibuja una cuadrícula de puntos cada **500 twips** de distancia.
  v) **QBColor (número)** es el color del punto, en nuestro caso QBColor(4) = rojo. El valor de número va de 0 (negro) hasta 15 (blanco brillante).

- Accede a la ventana de código...

  Y escribe el siguiente procedimiento de evento:

```
Private Sub Command1_Click()
   Dim x, y As Integer
   x = InputBox("Coordenada X")
   y = InputBox("Coordenada Y")
   If (x < 0 Or x > 4000) Or (y < 0 Or y > 4000) Then
      MsgBox ("Lo siento no tiene sentido")
      Else
      CurrentX = x
      CurrentY = y
```

```
   End If
End Sub
```

Está claro que con el procedimiento anterior establecemos el origen que queremos (CurrentX, CurrentY)

- Escribe el siguiente procedimiento:

```
Private Sub Command2_Click()
   Dim ho As Integer
   ho = InputBox("Pasos Horizontales")
   If CurrentX + ho > 0 And CurrentX + ho < 4000 Then
      Line -(CurrentX + ho, CurrentY)
      Else
      MsgBox ("Lo siento, no hay espacio")
   End If
End Sub
```

- Ejecuta el proyecto de la siguiente forma:
  w) CLIC en el icono **"Iniciar"**
  x) CLIC en [Origen]
  y) Coordenada X: 500 [Aceptar]
  z) Coordenada Y: 1000 [Aceptar]
  aa) CLIC en [Mov.Horiz.]
  bb) Pasos Horizontales: 500 [Aceptar]
  cc) [Mov.Horiz.]
  dd) Pasos horizontales: -900 [Aceptar]
  ee) [Mov.Horiz.]
  ff) Pasos Horizontales: 5000 [Aceptar]
  gg) "Acepta" el mensaje de error
  hh) CLIC en el icono **"Terminar"**

- Escribe en la ventana de código el siguiente procedimiento de evento:

```
Private Sub Command3_Click()
   Dim ve As Integer
   ve = InputBox("Pasos Verticales")
   If CurrentY + ve > 0 And CurrentY + ve < 4000 Then
      Line -(CurrentX, CurrentY + ve)
      Else
      MsgBox ("Lo siento, no puedo")
   End If
End Sub
```

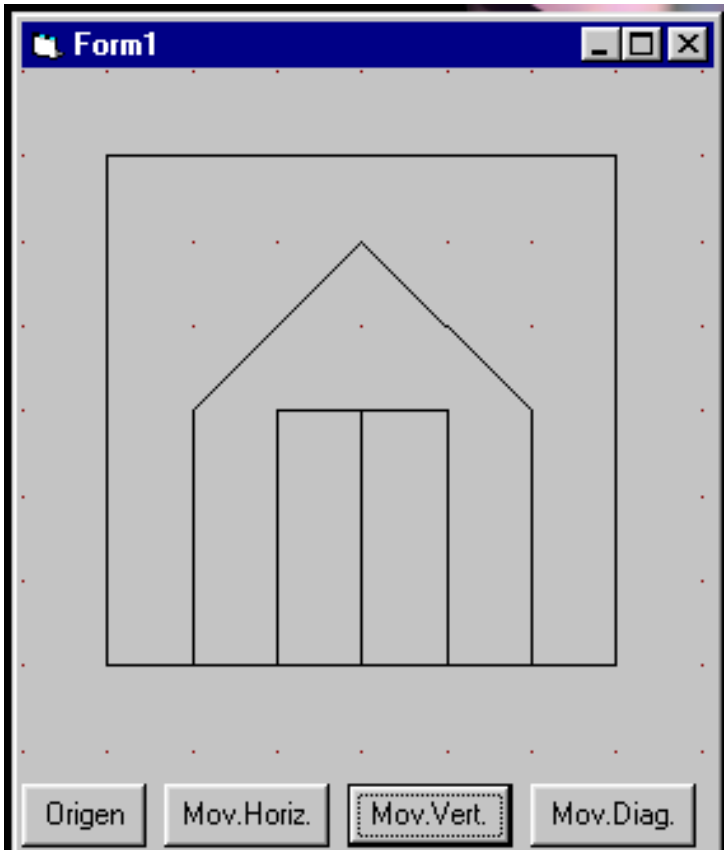
- **Ejecuta el proyecto para probar varias veces los botones [Origen], [Mov.Horiz.] y [Mov.Vert.]**

- Escribe en la ventana de código, nuestro último procedimiento:

```
Private Sub Command4_Click()
   Dim ho, ve As Integer
   ho = InputBox("Pasos derecha")
   ve = InputBox("Pasos abajo")
   If CurrentX + ho < 4000 And CurrentY + ve < 4000 Then
      Line -(CurrentX + ho, CurrentY + ve)
```

```
      Else
      MsgBox ("Lo siento, no hay espacio")
   End If
End Sub
```

- **Ejecuta el proyecto y a ver si consigues hacer el siguiente dibujo:**



- Graba el formulario con el nombre **Prog80.frm** y el proyecto con el nombre **Prog80.vbp**

i) Haz un nuevo proyecto con las siguientes características:

- Form1:
  Posición: 1140, 1200
  Tamaño: 4215 x 5220
- Command1:
  Caption: Colores 1
  Posición: 120, 4320
  Tamaño: 972 x 492
- Command2:
  Caption: Colores 2
  Posición: 1320, 4320
  Tamaño: 1092 x 492
- Accede a la ventana de código y escribe el siguiente procedimiento de evento:

```
Private Sub Command1_Click()
   Dim y As Integer
   Cls
   Randomize
```

```
For y = 100 To 4000 Step 100
                Line (100, y)-(4000, y), QBColor(Rnd * 15)
            Next y
        End Sub
```

- Escribe el siguiente procedimiento de evento:

```
Private Sub Command2_Click()
   Dim aum As Integer
   Cls
   Randomize
   For aum = 0 To 4000 Step 100
      Line (aum, 100)-(4000 - aum, 4000), RGB(255 * Rnd, 255 * Rnd, 255 *
Rnd)
   Next
        End Sub
```

- Ejecuta el proyecto y prueba el funcionamiento de los botones [Colores 1] y [Colores 2], tantas veces como quieras.
  - ii) La función **Cls** sirve para borrar la pantalla, mejor dicho el contenido del formulario.
  - jj) Recuerda que **QBColor** daba el color, y podía ir de 0 a 15. Por lo tanto **QBColor(Rnd*15)** nos dará aleatoriamente todos los posibles colores.
  - kk) Si en lugar de **QBColor(Rnd*15)** escribes **RGB(Rnd*255, Rnd*255, Rnd*255)** funcionará exactamente igual. Con una diferencia fundamental: utilizando **RGB** disponemos de **16.777.216** colores, en cambio con **QBColor** sólo disponemos de **16**.

- Graba el formulario con el nombre **Prog81.frm** y el proyecto con el nombre **Prog81.vbp** en *TuCarpeta*.

j) Haz un nuevo proyecto con las siguientes características:

- Form1:
  AutoRedraw: True
  Posición: 1080, 1170
  Tamaño: 4575 x 5250
- Command1:
  Caption: Rectángulo
  Posición: 120, 4440
  Tamaño: 1215 x 375
- Command2:
  Caption: Rectángulo lleno
  Posición: 1440, 4440
  Tamaño: 1575 x 375
- Command3:
  Caption: Círculos
  Posición: 3120, 4440
  Tamaño: 1215 x 375

- Escribe el procedimiento de evento:

```
Private Sub Form_Load()
   Dim i, j As Integer
   For i = 0 To 4000 Step 500
      For j = 0 To 4000 Step 500
```

```
            PSet (i, j), QBColor(4)
        Next j
    Next i
    CurrentX = 0
    CurrentY = 0
End Sub
```

- **Escribe en la ventana de código el procedimiento:**

```
Private Sub Command1_Click()
    Dim x1, x2, y1, y2 As Integer
    x1 = InputBox("Coordenada X del vértice 1")
    y1 = InputBox("Coordenada Y del vértice 1")
    x2 = InputBox("Coordenada X del vértice 2")
    y2 = InputBox("Coordenada y del vértice 2")
    Line (x1, y1)-(x2, y2), QBColor(Rnd * 15), B
End Sub
```

Ejecuta el proyecto y prueba el botón [Rectángulo] varias veces.

ll) El parámetro **B** al final de una sentencia **LINE** dibuja el rectángulo de vértices opuestos, los 2 puntos que hay a continuación de **LINE**.

- Escribe en la ventana de código el procedimiento:

```
Private Sub Command2_Click()
    Dim x1, x2, y1, y2 As Integer
    x1 = InputBox("Coordenada X del vértice 1")
    y1 = InputBox("Coordenada Y del vértice 1")
    x2 = InputBox("Coordenada X del vértice 2")
    y2 = InputBox("Coordenada y del vértice 2")
    Line (x1, y1)-(x2, y2), QBColor(Rnd * 15), BF
End Sub
```

mm) Ejecuta el proyecto y prueba el botón [Rectángulo Lleno] varias veces.

nn) El parámetro **BF** al final de la sentencia **LINE** dibuja el rectángulo **lleno** de vértices opuestos, los 2 puntos que hay a continuación de **LINE**.

- Escribe en la ventana de código el siguiente procedimiento:

```
Private Sub Command3_Click()
    Dim R As Integer
    Randomize
    For R = 0 To 2000 Step 50
        Circle (2000, 2000), R, RGB(255 * Rnd, 255 * Rnd, Rnd
* 255)
    Next R
End Sub
```

oo) La orden **Circle(2000, 2000), R, RGB(, ,)** dibuja un círculo de centro (2000, 2000) y radio R.

- Ejecuta el proyecto y pruébalo

- Graba el formulario con el nombre **Prog82.frm** y el proyecto con el nombre **Prog82.vbp** en *TuCarpeta*

k) Haz un nuevo proyecto con las siguientes características:

- Form1:
  Posición: 1080, 1170
  Tamaño: 4845 x 4755
  AutoRedraw= True
- Command1:
  Caption: Elipse
  Posición: 360, 3960
  Tamaño: 1095 x 375
- Command2:
  Caption: Elipses
  Posición: 1920, 3960
  Tamaño: 1095 x 375

- En la ventana de código escribe los siguientes procedimientos de evento:

```
Private Sub Form_Load()
   Dim i, j As Integer
   For i = 0 To 4000 Step 500
      For j = 0 To 4000 Step 500
         PSet (i, j), QBColor(4)
      Next j
   Next i
   CurrentX = 0
   CurrentY = 0
End Sub
```

```
Private Sub Command1_Click()
    Cls
    Dim CentroX, CentroY, Radio As Integer
    Dim Aspecto As Double
    Randomize
    CentroX = InputBox("Coordenada X del centro")
    CentroY = InputBox("Coordenada Y del centro")
    Radio = InputBox("Radio")
    Aspecto = InputBox("Aspecto de la elipse")
    FillStyle = 0
    FillColor = QBColor(Rnd * 15)
    Circle (CentroX, CentroY), Radio, QBColor(Rnd * 15), , , Aspecto
End Sub
```

pp) La orden **Circle** del anterior procedimiento dibujará una **elipse**, cuya excentricidad quedará determinada por el valor que demos al **Aspecto**.

qq) Las ordenes: **FillStyle= 0, FillColor= QBColor(Rnd*15)**, sirven para rellenar la elipse de color (aleatorio en nuestro caso).

```
Private Sub Command2_Click()
   Cls
   Randomize
   Dim Aspecto As Double
   For Aspecto = 0 To 10 Step 0.1
```

```
    Circle (2000, 2000), 1000, QBColor(Rnd * 15), , , Aspecto
  Next Aspecto
End Sub
```

- Ejecuta el proyecto y juega con los botones [Elipse] y [Elipses]. En el aspecto debes introducir un número del 0 al 10, que puede ser decimal. Es interesante investigar cómo es la elipse si **aspecto= 1**

- Graba el formulario con el nombre **Prog83.frm** y el proyecto con el nombre **Prog83.vbp** en *TuCarpeta*.

l) Recupera el proyecto **Prog79.vbp**,

Accede a la ventana de código, procedimiento **Form – Click**

Con todo lo que hemos hecho en los últimos apartados, ya podemos entender perfectamente todas las líneas de código, excepto una:

```
Circle (2000, 550), 300, QBColor(6), 0, Pi/2, 0.8
```

La orden anterior dibuja un **arco de elipse**:
rr) El centro del arco es el punto (2000, 550)
ss) El radio es 300 twips
tt) El color del arco es rojo (RGColor(4))
uu) El inicio del arco está en el ángulo = rad.
vv) El punto final del arco está en el ángulo pi/2 rad (90º)
ww) La curvatura del arco es 0.8 (1 = circular)

m) Crea un nuevo proyecto con los siguientes objetos y características:

- Form1:
  Posición: 990, 1110
  Tamaño: 5220 x 3900
  Caption: Ejes de Coordenadas
- Label1:
  Posición: 3480, 120
  Tamaño: 615 x 255
  Caption: Ancho:
- Label2:
  Posición: 3600, 480
  Tamaño: 375 x 255
  Caption: Alto:
- Label3:
  Posición: 3240, 1320
  Tamaño: 735 x 255
  Caption: Vértice A:
- Label4:
  Posición: 3120, 1680
  Tamaño: 255 x 255
  Caption: X=
- Label5:
  Posición: 4080, 1680
  Tamaño: 255 x 255
  Caption: Y=

- Label6:
  Posición: 3240, 2040
  Tamaño: 735 x 255
  Caption: Vértice B:
- Label7:
  Posición: 3120, 2400
  Tamaño: 255 x 255
  Caption: X=
- Label8:
  Posición: 4080, 2400
  Tamaño: 255 x 255
  Caption: Y=
- Label9:
  Posición: 4200, 120
  Tamaño: 855 x 255
  Caption: (nada)
- Label10:
  Posición: 4200, 480
  Tamaño: 855 x 255
  Caption: (nada)
- Picture1:
  Posición: 120, 240
  Tamaño: 2895 x 2895
- Command1:
  Posición: 3120, 840
  Tamaño: 975 x 375
  Caption: Dibuja
- Command2:
  Posición: 4200, 840
  Tamaño: 855 x 375
  Caption: Borra
- Command3:
  Posición: 3240, 2760
  Tamaño: 1695 x 375
  Caption: Aplica Escala
- Command4:
  Posición: 3240, 3120
  Tamaño: 1695 x 375
  Caption: Escala en mm.
- Text1:
  Posición: 3360, 1680
  Tamaño: 615 x 285
  Text: (nada)
- Text2:
  Posición: 4440, 1680
  Tamaño: 615 x 285
  Text: (nada)
- Text3:
  Posición: 3360, 2400
  Tamaño: 615 x 285
  Text: (nada)
- Text1:
  Posición: 4440, 2400
  Tamaño: 615 x 285
  Text: (nada)

Accede a la ventana de código y escribe el siguiente procedimiento general:

```
Public Sub Escala(XA, YA, XB, YB As Single)
   Picture1.Scale (XA, YA)-(XB, YB)
   Label9.Caption = XB - XA
   Label10.Caption = YA - YB
   Text1 = Str(Int(XA))
   Text2 = Str(Int(YA))
   Text3 = Str(Int(XB))
   Text4 = Str(Int(YB))
End Sub
```

Estamos de acuerdo que de momento no entendemos nada.

Escribe el siguiente procedimiento de evento:

```
Private Sub Form_Load()
   Dim VérticeAX, VérticeAY As Single
   Dim VérticeBX, VérticeBY As Single
   Dim MedidaX, MedidaY As Single
   Dim TwpsMm As Single
   TwpsMm = 10 / 567
   MedidaX = Picture1.Width * TwpsMm
   MedidaY = Picture1.Height * TwpsMm
   VérticeAX = -MedidaX / 2
   VérticeAY = MedidaY / 2
   VérticeBX = MedidaX / 2
   VérticeBY = -MedidaY / 2
   Escala VérticeAX, VérticeAY, VérticeBX, VérticeBY
End Sub
```

- **Ejecuta el programa, si todo va bien, aparece:**

Ancho: 51,0582
Alto: 51,0582
Vértice A: X = -26 Y = 25
Vértice B: X = 25 Y = -26

Veamos lo que acabamos de conseguir; accede a la ventana de código para observar los dos procedimientos que hemos escrito.

Private1.Scale (XA, YA) – (XB, YB)

Sirve para definir un sistema de coordenadas con el extremo superior izquierdo el punto (XA, YA) y el extremo inferior derecho el punto (XB, YB). Veamos de qué forma…

XA = Vértice AX = - MedidaX / 2 = - Picture1.Width * TwpsMm / 2

Dicho en palabras:
La coordenada X del vértice superior izquierdo es: Menos la anchura del **PictureBox** pasada a **mm**.
La anchura de nuestro **Picture1** era 2895 twips, en mm = 2895 x 10/567 = 51,0582 mm.

Conclusión:
La coordenada X del vértice superior izquierdo es: XA = Vértice AX = -51,0582/2 = -25,5291. Su parte entera es igual a **–26**

Exactamente lo mismo podríamos hacer para YA, XB, YB y obtendríamos:

YA = 25XB = 25YB = -26

Hemos conseguido en definitiva, gracias a nuestros dos procedimientos, definir en el **PictureBox** un sistema de coordenadas de centro (0, 0), en el centro del "PictureBox", extremo superior izquierdo el punto (-26, 25) y extremo inferior derecho el punto (25, -26); y todo esto en **mm**

De todas formas en pantalla, las medidas en mm no son reales porque podemos tener configurada la pantalla con más o menos resolución, pero si imprimimos el formulario (Archivo – Imprimir… - Proyecto – Imagen de formulario), veríamos que las medidas (en el papel impreso), del cuadro de dibujo son exactamente **51 x 51 milímetros**.

Accede a la ventana de código. Y escribe el siguiente procedimiento de evento:

```
Private Sub Command1_Click()
   Dim i As Integer
   Picture1.Line (-10, -10)-(10, 10), QBColor(5), BF
   For i = -30 To 30 Step 2
      Picture1.PSet (i, 0), QBColor(5)
      Picture1.PSet (0, i), QBColor(5)
   Next
End Sub
```

- Picture1.Line (-10, -10) – (10, 10), QBColor(5), BF
  Dibujará en nuestro "PictureBox" un cuadrado (BF) de color fucsia (color = 5) de vértices (-10, -10) y (10, 10) según "nuestro sistema de coordenadas".

- For i = -30 To 30 Step 2
  Picture1.Pset (i, 0), QBColor(5)
  Picture1.Pset (0, i). QBColor(5)
  Next

Simulará los dos ejes de coordenadas (de nuestro sistema de coordenadas), con los puntos de color fucsia separados 2 unidades (mm en papel impreso).

- Ejecuta el programa para ver si es verdad.

Escribe los siguientes procedimientos de evento:

```
Private Sub Command2_Click()
      Picture1.Cls
End Sub
```

```
Private Sub Command3_Click()
   Escala Val(Text1), Val(Text2), Val(Text3), Val(Text4)
End Sub
```

El procedimiento anterior establecerá un nuevo sistema de coordenadas en el PictureBox de extremos los números que tengamos escritos en los 4 TextBox.

```
Private Sub Command4_Click()
```

```
    Form_Load
End Sub
```

El procedimiento anterior restablecerá el sistema de coordenadas primitivo (en mm).

- Ejecuta el programa de la siguiente forma:
  - CLIC en el icono **Iniciar**
  - CLIC en [Dibuja]

  Tenemos un cuadrado de 20 mm de lado (en papel impreso)
  - Escribe como vértice A: -15, 15 y vértice B: 15, -15
  - CLIC en [Borra]
  - CLIC en [Aplica Escala]
  - CLIC en [Dibuja]

  Resulta un cuadrado más grande. Aunque el lado del cuadrado contínua siendo 20 unidades, pero ahora no son milímetros, porque las medidas del **PictureBox** ahora son **30 x 30**.

  - Sería interesante investigar cómo queda el dibujo para las escalas:
    A= (-100, 100) B= (100, -100)
    A= (-50, 150) B= (150, -50)
    A= (-40, 20) B= (40, -20)
    A= (-20, 30) B= (20, -30)

- Graba el formulario con el nombre **Prog84.frm** y el proyecto con el nombre **Prog84.vbp** en *TuCarpeta*.

n) Vamos a hacer un nuevo proyecto que sirva para representar gráficamente la función: $\mathbf{y = x^2 - 1}$

- Haz un nuevo proyecto con los siguientes objetos y características:
  - Form1:
    Posición: 990, 1125
    Tamaño: 3720 x 4920
  - Picture1:
    Posición: 120, 120
    Tamaño: 3375 x 4335

- En la ventana de código escribe el siguiente procedimiento general:

```
Public Sub EscalaCm()
   Dim MedidaX, MedidaY As Single
   Dim TwipsCm As Single
   Dim AX, AY, BX, BY As Single
   TwipsCm = 1 / 567
   MedidaX = Picture1.Width * TwipsCm
   MedidaY = Picture1.Height * TwipsCm
   AX = -MedidaX / 2
   AY = MedidaY / 2
   BX = MedidaX / 2
   BY = -MedidaY / 2
   Picture1.Scale (AX, AY)-(BX, BY)
End Sub
```

Está claro que este procedimiento define un sistema de coordenadas en el **PictureBox**, de centro el centro del **PictureBox** y unidad **un centímetro**.

- Escribe el siguiente procedimiento de evento:

```
Private Sub Form_Load()
   EscalaCm
   Dim I As Integer
   For I = -5 To 5 Step 1
      Picture1.PSet (I, 0), QBColor(5)
      Picture1.PSet (0, I), QBColor(5)
   Next
End Sub
```

Es decir: al leerse el formulario, se establecerá en el **PictureBox**, el sistema de coordenadas en cm y se dibujarán en fucsia puntos separados 1 cm. Que simularan los dos ejes de coordenadas.

- Escribe el siguiente procedimiento de evento:

```
Private Sub Picture1_Click()
   Dim x As Single
   x = -3
  Do While x < 3
      Picture1.Pset (x, x * x - 2)
      x = x + 0.1
   Loop
End Sub
```

El procedimiento anterior dibujará la función $\mathbf{y = x^2 - 2}$
Es decir los puntos **(x, x*x –2)** entre **–3 y 3**

- Ejecuta el proyecto para ver si es verdad (debes hacer CLIC en el **PictureBox**, ya que el procedimiento que dibuja la parábola es el **Picture1.Click**).

El programa anterior es muy bonito pero nos gustaría que la parábola se viera mejor. Es decir, en lugar de puntos nos gustaría que la gráfica de la función se viera de forma contínua…

Corrige el procedimiento **Picture1_Click** de forma que nos quede:

```
Private Sub Picture1_Click()
   Dim x As Single
   x = -3
   Picture1.CurrentX = x
   Picture1.CurrentY = x * x - 2
   Do While x < 3
      Picture1.Line -(x, x * x - 2)
      x = x + 0.1
   Loop
End Sub
```

- Ejecuta el proyecto, para ver si funciona.

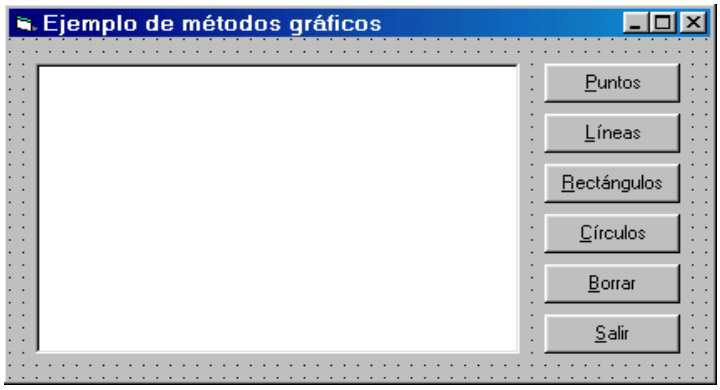
- Graba el formulario con el nombre **Prog85.frm** y el proyecto con el nombre **Prog85.vbp** en *TuCarpeta*.

o) Vamos a hacer un proyecto que muestra una serie de dibujos realizados con los distintos métodos gráficos de que disponen los controles contenedores (form y picturebox).

Se trata de crear un panel de dibujo sobre el que se visualizen diversas figuras geométicas de tamaños, posiciones y colores aleatorias. cada botón muestra los dibujos realizados con uno de los métodos (circulos, puntos rectángulos, etc).

Crea un nuevo proyecto...

1. Añadir al formulario por defecto un cuadro de dibujo y cinco botones de comando.



2. Especificar las siguinetes propiedades en tiempo de diseño:

Objeto	**Propiedad**	**Valor**
Form1	Name	frmGraficos
	Caption	Ejemplo de métodos gráficos
Picture1	Name	picDemo
	BackColor	&H00FFFFFF& (Blanco)
Command1	Name	cmdSalir
	Caption	&Salir
Command2	Name	cmdBorrar
	Caption	&Borrar
Command3	Name	cmdCirculos
	Caption	&Círculos
Command4	Name	cmdRectang
	Caption	&Rectángulos
Command5	Name	cmdLineas
	Caption	&Líneas
Command6	Name	cmdPuntos
	Caption	&Puntos

3. Escribir los siguientes procedimientos de evento:

```
Private Sub cmdBorrar_Click()
  'Borra todo el cuadro de dibujo
  picDemo.Cls
End Sub

Private Sub cmdCirculos_Click()
  Dim X As Integer, Y As Integer
  Dim Color As Long, Radio As Integer
  Dim N As Integer
  'Para cada circunferencia
  For N = 1 To 50
    'Obtener coordenadas del centro aleatorias
    X = Rnd * picDemo.ScaleWidth
    Y = Rnd * picDemo.ScaleHeight
```

```
    'Obtener radio aleatorio
    Radio = Rnd * picDemo.ScaleHeight
    'Obtener color aleatorio
    Color = QBColor(Int(Rnd * 16))
    'Dibujar la circunferencia
    picDemo.Circle (X, Y), Radio, Color
  Next N
End Sub

Private Sub cmdLineas_Click()
  Dim X1 As Integer, Y1 As Integer
  Dim X2 As Integer, Y2 As Integer
  Dim Color As Long, N As Integer
  'Para cada línea
  For N = 1 To 50
    'Obtener coordenadas iniciales y finales aleatorias
    X1 = Rnd * picDemo.ScaleWidth
    Y1 = Rnd * picDemo.ScaleHeight
    X2 = Rnd * picDemo.ScaleWidth
    Y2 = Rnd * picDemo.ScaleHeight
    'Obtener color aleatorio
    Color = QBColor(Int(Rnd * 16))
    'Dibujar la línea
    picDemo.Line (X1, Y1)-(X2, Y2), Color
  Next N
End Sub

Private Sub cmdPuntos_Click()
  Dim X As Integer, Y As Integer
  Dim Color As Long, N As Integer
  'Poner grosor del punto a 5 pixels
  picDemo.DrawWidth = 5
  'Para cada punto
  For N = 1 To 50
    'Obtener coordenadas aleatorias
    X = Rnd * picDemo.ScaleWidth
    Y = Rnd * picDemo.ScaleHeight
    'Obtener color aleatorio
    Color = QBColor(Int(Rnd * 16))
    'Dibujar el punto
    picDemo.PSet (X, Y), Color
  Next N
  'Restaurar el grosor de dibujo
  picDemo.DrawWidth = 1
End Sub

Private Sub cmdRectang_Click()
  Dim X1 As Integer, Y1 As Integer
  Dim X2 As Integer, Y2 As Integer
  Dim Color As Long, N As Integer
  'Para cada rectángulo
  For N = 1 To 50
    'Obtener coordenadas iniciales y finales aleatorias
    X1 = Rnd * picDemo.ScaleWidth
    Y1 = Rnd * picDemo.ScaleHeight
    X2 = Rnd * picDemo.ScaleWidth
    Y2 = Rnd * picDemo.ScaleHeight
    'Obtener color aleatorio
    Color = QBColor(Int(Rnd * 16))
    'Dibujar el rectángulo
```

```
    picDemo.Line (X1, Y1)-(X2, Y2), Color, B
  Next N
End Sub

Private Sub cmdSalir_Click()
  Unload Me
End Sub

Private Sub Form_Load()
  'Iniciar las secuencias aleatorias
  Randomize
End Sub
```

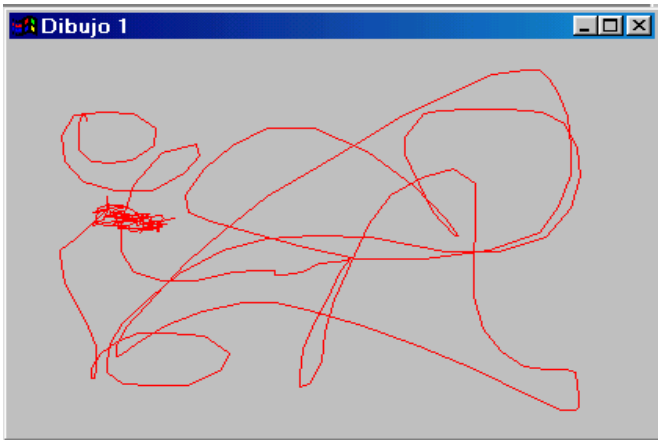
4. Guardar el formulario y el el proyecto en los archivos **Prog86.frm y Prog86.vbp**, respectivamente.

p) Vamos a simular el dibujar a “mano alzada” en un formulario....

Crea un nuevo proyecto

Sin insertar ningún control accede a la ventana de código y escribe:

```
Private r As Integer, g As Integer, b As Integer
Private Dibujo As Integer

Private Sub Form_Load()
   Move (Screen.Width - Width) \ 2, (Screen.Height - Height) \ 2
End Sub

Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Dibujo = Dibujo - 1
   Select Case Button
   Case vbLeftButton
      r = 255
      g = 0
      b = 0
   Case vbMiddleButton
      r = 0
      g = 255
      b = 0
   Case vbRightButton
      r = 0
      g = 0
      b = 255
   End Select
   CurrentX = X
   CurrentY = Y
End Sub

Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   If Dibujo Then
      Line -(X, Y), RGB(r, g, b)
   End If
End Sub

Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Dibujo = 0
End Sub
```

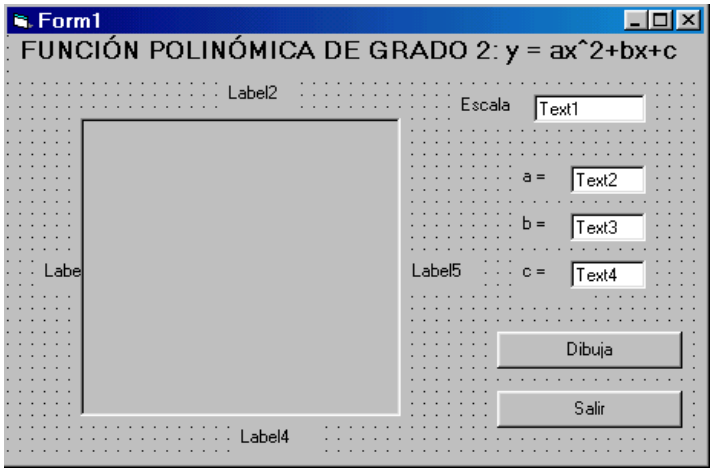


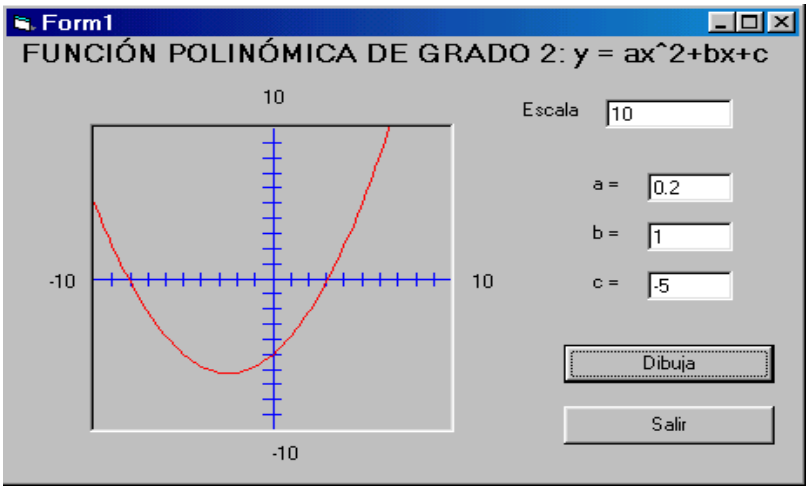
Graba el programa como **Prog87.frm, Prog87.vbp**

q) Vamos a representar gráficamente una función polinómica de segundo grado, cualquiera:

Crea un nuevo proyecto

Inserta:



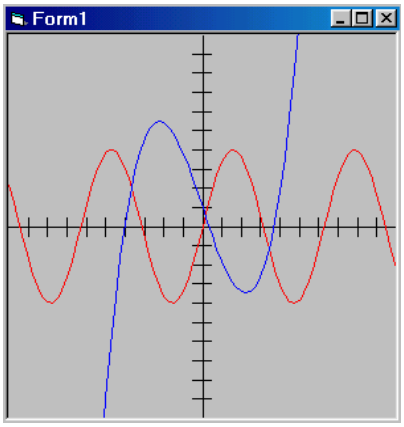


El código que debes escribir es el siguiente:

```
Private Sub Command1_Click()
   xmin = -Val(Text1.Text)
   Picture1.ForeColor = vbRed
   x = xmin
   a = Val(Text2.Text)
   b = Val(Text3.Text)
   c = Val(Text4.Text)
   y = a * x * x + b * x + c
   Picture1.CurrentX = x
   Picture1.CurrentY = y
   For x = xmin To -xmin Step 0.25
      y = a * x * x + b * x + c
      Picture1.Line -(x, y)
   Next x
End Sub

Private Sub Command2_Click()
   End
End Sub

Private Sub Form_Load()
   Text1.Text = ""
   Text2.Text = ""
   Text3.Text = ""
   Text4.Text = ""

End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then
      escala = Val(Text1.Text)
      Picture1.ForeColor = vbBlue
      Picture1.Scale (-escala, escala)-(escala, -escala)
      Picture1.Line (-escala, 0)-(escala, 0)
      For i = -escala + 1 To escala - 1
         Picture1.Line (i, -0.5)-(i, 0.5)
      Next i
      Picture1.Line (0, -escala)-(0, escala)
      For i = -escala + 1 To escala - 1
         Picture1.Line (-0.5, i)-(0.5, i)
      Next i
      Label2.Caption = Str(escala)
      Label3.Caption = Str(-escala)
      Label4.Caption = Str(-escala)
      Label5.Caption = Str(escala)
   End If
End Sub
```

Graba el programa como: **Prog88.frm, Prog88.vbp**

r) Vamos a dibujar dos funciones más...
Crea un nuevo proyecto e inserta en el formulario un **PictureBox** que ocupe todo el form.



Escribe el siguiente código:

```
Private Sub Form_Load()
Dim i As Integer
Dim x As Single
Dim y As Single

   Picture1.Scale (-10, 10)-(10, -10)

   ' Draw X axis.
   Picture1.Line (-10, 0)-(10, 0)
   For i = -9 To 9
      Picture1.Line (i, -0.5)-(i, 0.5)
   Next i

   ' Draw Y axis.
   Picture1.Line (0, -10)-(0, 10)
   For i = -9 To 9
      Picture1.Line (-0.5, i)-(0.5, i)
   Next i

   ' Draw y = 4 * sin(x).
   Picture1.ForeColor = vbRed
   x = -10
   y = 4 * Sin(x)
   Picture1.CurrentX = x
   Picture1.CurrentY = y
   For x = -10 To 10 Step 0.25
      y = 4 * Sin(x)
      Picture1.Line -(x, y)
   Next x

   ' Draw y = x ^ 3 / 5 - 3 * x + 1.
   Picture1.ForeColor = vbBlue
   x = -10
   y = x ^ 3 / 5 - 3 * x + 1
   Picture1.CurrentX = x
   Picture1.CurrentY = y
   For x = -10 To 10 Step 0.25
      y = x ^ 3 / 5 - 3 * x + 1
      Picture1.Line -(x, y)
   Next x
End Sub
```

Graba el programa como **Prog89.frm, Prog89.vbp**

IX.- Controles ActiveX

Un control Activex es una extensión del cuadro de herramientas que incorpora Visual Basic.
El funcionamiento de un control ActiveX es el mismo que el de los controles estándar que lleva incorporados VB. En el momento en que añadimos un control **ActiveX** a una aplicación, este control pasa a ser parte del entorno de desarrollo y de tiempo de ejecución.

Cómo cargar los controles ActiveX
Los controles ActiveX tienen la extensión .OCX
Hay de dos tipos:
- Los suministrados por Visual Basic
    8) Los controles ActiveX estándar (los que aparecen en la barra de controles)
    9) Los controles ActiveX no estándar
- Los suministrados por otros programadores

Para **cargar** un control ActiveX (que no aparece en la barra de herramientas):
- New Project
- Menú Project
        Components...
            Selecciona el que quieras. En el campo "Location" aparece el archivo donde se encuentra
        [Apply]
- En la barra de controles aparece el nuevo o nuevos controles.

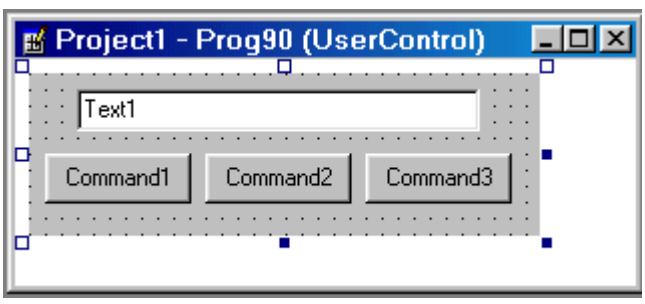
La versión "5.0 CCE" es una versión del Visual Basic especial para la creación de controles ActiveX.
No nos dedicaremos en este capítulo a la creación profesional de controles ActiveX...
Puedes encontrar todos los que quieras en:
- Versiones más modernas del Visual Basic
- www.activex.com
- terra.canalsw.com
        ActiveX download
- Galería de componentes ActiveX de Microsoft.

Lo que sí estudiaremos será: la creación de controles ActiveX como una forma de incluir los programas que hemos hecho hasta ahora en otros programas como el Excel o en una página Web por ejemplo.
Es decir, ya que nuestra versión de Visual Basic (5.0 CCE) no nos permite compilar nuestros programas para poderlos ejecutar sin el Visual Basic, nos ingeniamos para poder hacerlo de todas formas. La idea es ejecutar y "compilar" nuestros programas sin necesidad del VB, utilizando los "controles ActiveX".

a) Menú File
        New Project
            Con el icono **"ActiveX Control"** activado
            [Aceptar]
En principio la única diferencia que apreciamos en el "form" de un ActiveX, es que no tiene borde.

- Propiedad **Name** del "UserControl" = **Prog90**

- Inserta en el form:



- Propiedades:

Text1
Text =

Command1	**Command2**	**Command3**
Caption = Hola	Caption = Adiós	Caption = Borrar

- Código:

```
Private Sub Command1_Click()
  Text1.Text = "HOOOOOOOOLA"
End Sub

Private Sub Command2_Click()
  Text1.Text = "ADIOOOOOOOOS"
End Sub

Private Sub Command3_Click()
  Text1.Text = ""
End Sub
```

- Graba nuestro "programa":
Menú File
Save Project As ...
Observa la extensión **CTL**, de nuestro "UserControl"
Sitúate en *TuCarpeta* y como **name** del fichero escribe: **Prog90**
Graba el proyecto como **Project1.vbp**

- El siguiente paso es compilar el control. Haz lo siguiente:
Menú File
Make Prog90.ocx ...
Grábalo en *TuCarpeta*
- Intenta ejecutar nuestro programa. Observarás que no podemos. Ya veremos en otro ejercicio la solución a este problema.

Cierra el proyecto (Menú File - Remove Project)

b) Vamos a utilizar nuestro control **Prog90**

- Menú File
New Project
Standard EXE
[Aceptar]

- Nos gustaría insertar nuestro control **Prog90**, pero no lo vemos en el "Cuadro de Controles". Haz lo siguiente:
Menú Project
Components ...

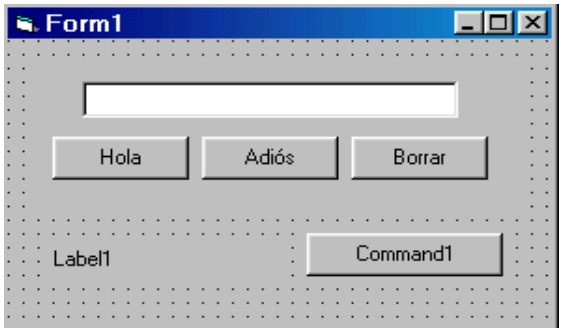
10) En el listado correspondiente aparece **Project1**
11) Si seleccionas **Project1** en el campo "Location" aparece: c:*TuCarpeta*\Prog90.ocx
12) En caso de que no aparezca, haz clic en [Browse] y localízalo en *TuCarpeta*

Clic en [Apply]
Por último clic en [Ok]

- Si todo ha funcionado correctamente aparece en el "Cuadro de Controles" un nuevo icono. Si inmovilizas el cursor del ratón en el nuevo icono, después de un par de segundos, aparecerá la leyenda **Prog90**

- Inserta el nuevo control en el formulario

- E inserta algo más, por ejemplo un **Label** y un **CommandButton**:



- Accede a la ventana de código y escribe:

```
Private Sub Command1_Click()
    Label1.Caption = "Pasaaaaaaaa aquí"
End Sub
```

- Graba el formulario como **Prog91.frm** y el proyecto como **Prog91.vbp**

- Ejecuta el programa para probarlo

c) Vamos a ver como incluir nuestro control en otro programa, es decir se trata de ejecutar el programa **Prog90** en un entorno que no sea el Visual Basic.

- Cierra el Visual Basic y ejecuta el **Microsoft Excel**

- Menú Ver
  Barras de herramientas
  Cuadro de controles

- Clic en el icono **"Más controles"** de la barra anterior:

- Localiza y selecciona (clic encima) el **Project1.Prog90**

- Observa que aparece un nuevo icono (diseño), que se encuentra pulsado.

- Marca un recuadro en la hoja

- Clic en el nuevo icono (dejará de estar pulsado), que desaparecerá

Juega con "nuestro programa"

- Graba la hoja de cálculo con el nombre **Prog92.xls** en *TuCarpeta*

d) El problema que nos queda resolver es la “transportabilidad” de nuestro programa **Prog90**

Es decir, si intentamos ejecutar el **Prog92.xls** en otro ordenador que no sea el nuestro, aunque tengamos el Excel, última versión, no funcionará porque el control **Prog90.ocx** sólo se encuentra en nuestro ordenador.

Vamos a solucionar el problema:

- Sal del Visual Basic

- Desde la pantalla inicial del Windows haz lo siguiente:
    [Inicio]
        Programas
            Visual Basic 5.0 CCE
                Application Setup Wizard

- Ante la pantalla de introducción, haz clic en [Next>]

- Utilizando [Browse...] localiza el control **Prog90.ocx**, recuerda que el fichero de proyecto correspondiente era **Project1.vbp** de *TuCarpeta*, es decir en el campo correspondiente debe aparecer: **c:\ *TuCarpeta*\Project1.vbp**
Asegúrate que la opción **Create a Setup Program** está activada y clic en [Next>]

- En la siguiente pantalla, activa la opción: **Disk Directories (\Disk1, \Disk2, etc)**
y [Next>]

- En el campo correspondiente, escribe:
    c:*TuCarpeta*\ActiveX
    En **Disk Size** debe aparecer **1.44 Mb**
Clic en [Next>]
Como la carpeta “ActiveX” no existía, el asistente nos pide permiso para crearla, haz clic en [Yes]

- Clic en [Next>]

- Clic en [Next>]

- Clic en [Finish]

Si todo ha funcionado correctamente tendremos en la carpeta **ActiveX** de *TuCarpeta*, las nuevas carpetas **Disk1** y **Disk2**, que contienen una serie de ficheros con una capacidad total inferior a 1.44 Mb

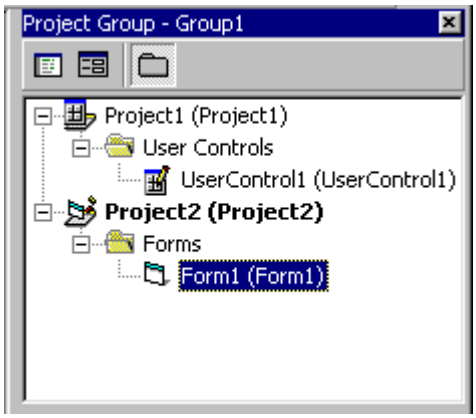
Basta copiar el contenido de **Disk1** y **Disk2** en dos disquetes para poder instalar nuestro control **Prog90.ocx** en cualquier ordenador (doble clic en el fichero **Setup.exe** del primer disquete **Disk1**).

e) Vamos a hacer otro control **ActiveX**, que sirva para simplificar una fracción (ya lo habíamos hecho en un ejercicio).

Como el control que queremos hacer es más complejo que el “Prog90”, convendría probarlo antes de salir del entorno de VB, por esta razón procederemos de otra forma:

- Menú File
    New Project
        Selecciona el icono **CtlGroup**
    [Aceptar]

– Observa la **Ventana de propiedades:**



Disponemos de:

El UserControl
El Project1 (proyecto del control ActiveX)
El Form1, con su proyecto (Project2), que no es más que un programa que nos permitirá probar el control "UserControl", antes de grabarlo definitivamente.
El "Project Group" (observa el marco de la ventana de proyecto), que es la "estructura" que contiene todos los elementos anteriores.

- Antes de continuar vamos a poner nombre y a grabar todos los elementos de "nuestro grupo" ...

Selecciona (doble clic) el **UserControl1 (UserControl1)** en la ventana de proyecto y en la propiedad **Name** de la ventana de propiedades escribe: **Prog93**
Selecciona (doble clic) el **Project1(Project1)** en la ventana de proyecto y en la propiedad **Name** de la ventana de propiedades escribe: **Prog93p**
Selecciona (doble clic) el **Form1(Form1)** en la ventana de proyecto y en la propiedad **Name** de la ventana de propiedades escribe: **Prog94**
Selecciona (doble clic) el **Project2(Project2)** en la ventana de proyecto y en la propiedad **Name** de la ventana de propiedades escribe: **Prog94p**

Vamos a grabar ...

Menú File
Save Project Group As ...

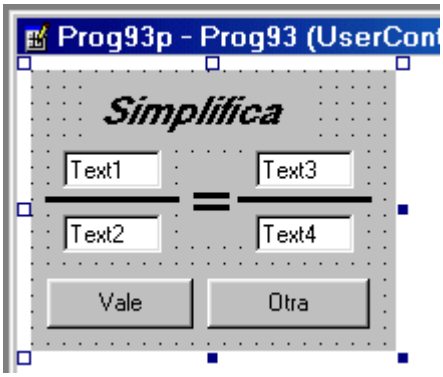
1) Nombre del control: **Prog93.ctl**
2) Nombre del proyecto de control: **Prog93p.vbp**
3) Nombre del formulario de prueba: **Prog94.frm**
4) Nombre del proyecto de prueba: **Prog94p.vbp**
5) Nombre del grupo: **Prog95.vbg**

Observa el marco de la "ventana de proyecto": **Project Group - Prog95**

La finalidad de nuestro grupo **Prog95.vbg** es crear y probar el control **ActiveX** que sirva para simplificar una fracción y poder modificar (y probar) en el futuro tantas veces como queramos el "control".

f) Activa el **User Control: Prog93** (doble clic en la "ventana de proyecto").

- Inserta:



- En la propiedad **Text** de los 4 TextBoxs, borra el contenido que aparece por defecto y no escribas nada.

- Código:

```
Public Function MCD(a As Long, b As Long) As Long
   Dim resto As Long, aux As Long
   If a < b Then
      aux = a
      a = b
      b = aux
   End If
   If a Mod b = 0 Then
      resto = b
   End If
   Do While a Mod b <> 0
      resto = a Mod b
      a = b
      b = resto
   Loop
   MCD = resto
End Function

Private Sub Command1_Click()
        ' simplificar
   Dim x As Long, y As Long
   Dim n As Long
   x = Text1.Text
   y = Text2.Text
   n = MCD((x), (y))
   If n <> 0 Then
      Text3.Text = x / n
      Text4.Text = y / n
   End If
End Sub

Private Sub Command2_Click()
        'Otra
   Text1.Text = ""
   Text2.Text = ""
   Text3.Text = ""
   Text4.Text = ""
Text1.SetFocus
End Sub
```

g) Vamos a **probar** nuestro control ...

- Cierra la ventana **Prog93p - Prog93 (UserControl)**
Observa el nuevo icono en la “Barra de Controles”

- Activa el formulario: **Prog94 (Prog94.frm)**

- Inserta el control **Prog93** en el formulario

- Pruébalo, es decir clic en **Start** (Iniciar)

- Supongamos que nuestro control está acabado
En este caso:

13) Con la ventana del control **Prog93** abierta
14) Menú File

Make **Prog93p.ocx**

Graba todo lo que hemos hecho con el mismo nombre

A partir de este momento ya podemos utilizar el control **Prog93p** en nuestros programas VB u otros programas (pruébalo).

h) Supongamos que queremos hacer una página Web que contenga nuestro control **Prog93p**

Haz lo siguiente:
- Ejecuta el "Application Setup Wizard"

- Ante la pantalla de "Introduction", haz clic en [Next>]

- Selecciona el "Prog93p.vbp" de *TuCarpeta* y activa la opción **Create Internet Download Setup**
[Next>]

- c:*TuCarpeta*\ActiveX\Internet
[Yes], para crear el directorio
[Next>]

- Activa la opción **Use alternate location** y no escribas nada en el campo "URL, UNC, or ..."
[Next>]

- [Next>]

- [Next>]

- [Finish]

Investiga, utilizando el "Explorador de Windows" el contenido de la carpeta **Internet** que acabas de crear.

Basta que ejecutes (doble clic) el fichero **Prog93p.htm** y verás la página web con nuestro control ActiveX.

X.- Ficheros

Visual Basic nos proporciona 3 controles estándar (es decir, que siempre aparecen en la barra de controles) especializados en el acceso al sistema de ficheros del entorno Windows.

Estos controles son:

- **DriveListBox** (Cuadro de lista de unidades) 

No es mas que un cuadro de lista que muestra las diferentes **unidades** del sistema.

- **DirListBox** (Cuadro de lista de directorios) 

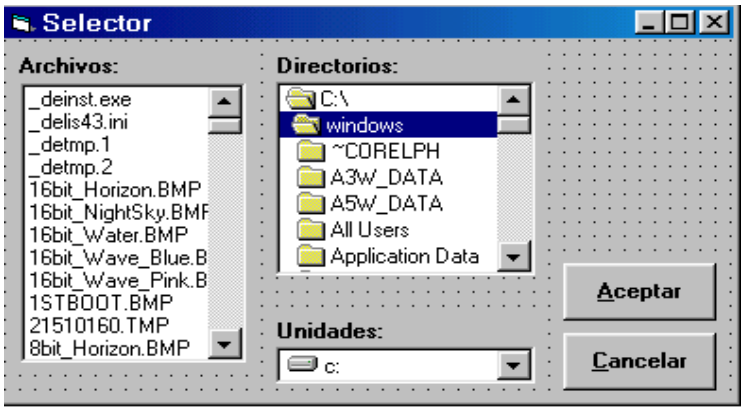
Muestra los **directorios** del sistema de archivos del ordenador

- **FileListBox** (Cuadro de lista de archivos) 

Muestra los **archivos** de una determinada carpeta

a) Crea un nuevo proyecto

Hemos de conseguir una ventana que permita seleccionar un archivo en cualquier directorio de cualquier unidad disponible, permitiendo al usuario navegar por éstas.



El proceso que debes seguir es el siguiente:

1. Añadir al formulario por defecto tres botones de comando, dos etiquetas, y los tres controles comentados.

2. Establecer las siguientes propiedades en tiempo de diseño:

Objeto	**Propiedad**	**Valor**
Form1	Caption	Selector
Command1	Name	cmdAceptar
	Caption	&Aceptar
	FontBold	true
Command2	Name	cmdCancelar
	Caption	&Cancelar
	FontBold	True

File1	Name	filArchivos
Dir1	Name	dirDirectorios
Drive1	Name	drvUnidades
Label1	name	lblDirectorios
	Caption	Directorios:
	FontBold	True
Label2	Name	lblUnidades
	Caption	Unidades:
	FontBold	True
Label3	Name	lblArchivos
	Caption	Archivos:
	FontBold	True

3. Escribir los siguientes procedimientos de evento:

```
Private Sub cmdCancelar_Click()
  MsgBox "No se ha seleccionado ningún archivo", , "AVISO"
  Unload Me
End Sub

Private Sub cmdAceptar_Click()
  If filArchivos.ListIndex <> -1 Then
    MsgBox "Se ha seleccionado el archivo: " & _
         filArchivos.List(filArchivos.ListIndex), , "AVISO"
    Unload Me
  End If
End Sub

Private Sub dirDirectorios_Change()
  filArchivos.Path = dirDirectorios.Path
End Sub

Private Sub drvUnidades_Change()
  dirDirectorios.Path = drvUnidades.Drive
End Sub

Private Sub filArchivos_DblClick()
  cmdAceptar_Click
End Sub
```

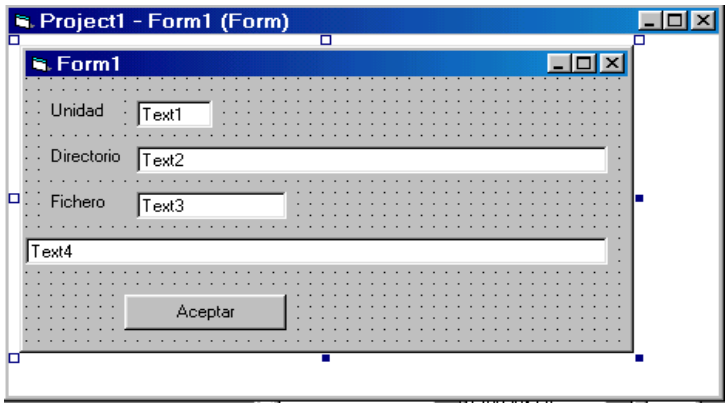
4. Guardar el formulario y el proyecto en los archivos **Prog96.frm** y **Prog96.vbp**, respectivamente.

OBSERVA

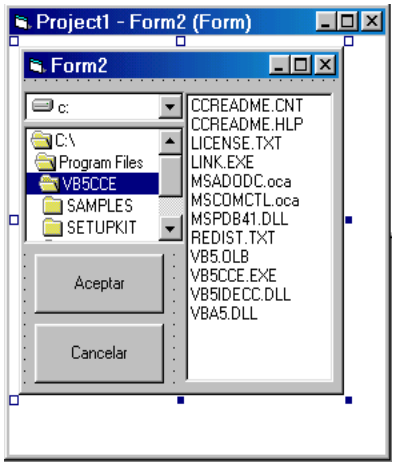
Para determinar si un archivo ha sido seleccionado, se comprueba que la propiedad ListIndex de la lista de archivos no esté a -1. La selección del archivo puede realizarse seleccionado un elemento en la lista de archivos y haciendo clic en el botón "Aceptar", o bien haciendo doble clic sobre el elemento. Este ejemplo puede utilizarse como parte de otras aplicaciones que necesiten realizar la operación de seleccionar un archivo, que es lo que haremos en el siguiente proyecto.

b) Vamos a completar el programa anterior para solucionar lo que realmente nos interesa que es: en un momento determinado, localizar un fichero de nuestro ordenador:
Crea un nuevo proyecto

- Inserta:



- Inserta otro formulario con el siguiente contenido:



- Código del form1:

```
Private Sub Command1_Click()
   Form2.Show
End Sub

Private Sub Form_Load()
   Text1.Text = ""
   Text2.Text = ""
   Text3.Text = ""
   Text4.Text = ""
End Sub
```

- Código del form2:

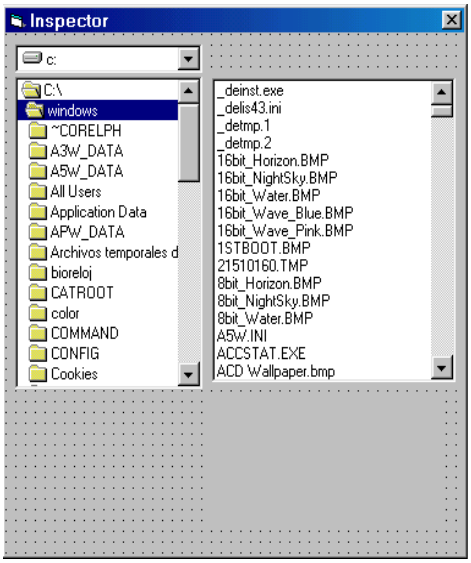
```
Private Sub Command1_Click()
   If File1.ListIndex <> -1 Then
      Form1.Text3.Text = File1.List(File1.ListIndex)
      Form1.Text1.Text = Drive1.Drive
      Form1.Text2.Text = Dir1.Path
      Form1.Text4.Text = Form1.Text2.Text & "\" & Form1.Text3.Text
      Unload Me
   End If
End Sub

Private Sub Command2_Click()
   Form1.Text1.Text = ""
   Form1.Text2.Text = ""
   Form1.Text3.Text = ""
   Form1.Text4.Text = ""
   Unload Me
```

```
End Sub

Private Sub Dir1_Change()
   File1.Path = Dir1.Path
End Sub

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

Private Sub File1_Click()
   Command1_Click
End Sub
```

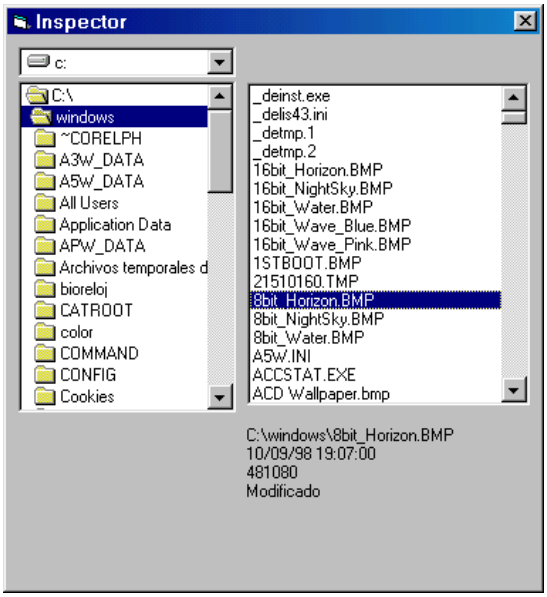
- Graba el programa como **Prog97a.frm, Prog97b.frm y Prog97.vbp** y pruébalo.

c) Crea un nuevo proyecto

Inserta:



Se trata de conseguir, que al seleccionar un archivo, aparezca en el “label”, las caracteristicas del fichero:



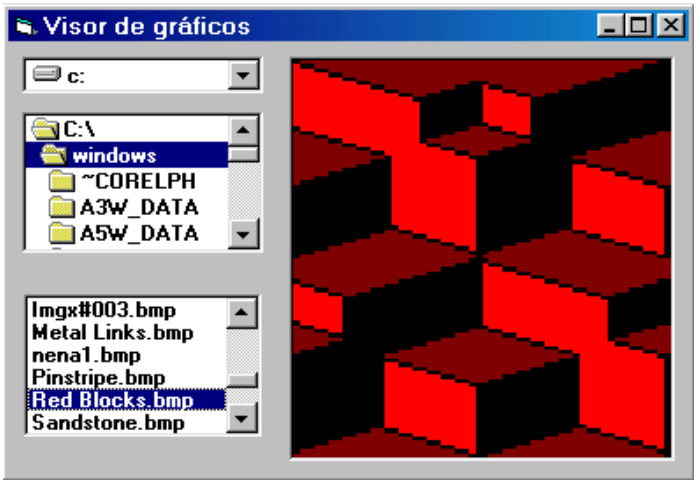
- Código:

```
   ' Al seleccionar un elemento en la lista de archivos
Private Sub Archivos_Click()
   Dim NombreArchivo As String, NuevaLinea As String
   NuevaLinea = Chr(13) + Chr(10) ' Carácter de separación de línea
      ' Obtener el nombre completo del archivo
   NombreArchivo = Archivos.Path
   If Right(NombreArchivo, 1) <> "\" Then NombreArchivo = NombreArchivo & "\"
   NombreArchivo = NombreArchivo & Archivos.List(Archivos.ListIndex)
      ' Obtener los atributos
   Dim Atributos As Integer, ListaAtributos As String
   Atributos = GetAttr(NombreArchivo)
      ' Según los atributos activos ir formando la cadena
   If Atributos And vbReadOnly Then ListaAtributos = "Sólo lectura  "
   If Atributos And vbHidden Then ListaAtributos = ListaAtributos & "Oculto "
   If Atributos And vbSystem Then ListaAtributos = ListaAtributos & "Sistema "
   If Atributos And vbArchive Then ListaAtributos = ListaAtributos & "Modificado"
      ' Mostrar toda la información en la etiqueta de texto
   Información.Caption = NombreArchivo & NuevaLinea & _
                         FileDateTime(NombreArchivo) & NuevaLinea & _
                         FileLen(NombreArchivo) & NuevaLinea & ListaAtributos
End Sub

   ' Al cambiar de directorio
Private Sub Directorios_Change()
   Archivos.Path = Directorios.Path ' Actualizar la lista de archivos
End Sub

   ' Al cambiar de unidad
Private Sub Unidad_Change()
   On Error Resume Next
   Directorios.Path = Unidad.Drive ' Actualizar la lista de directorios
   If Err Then ' Si no es posible cambiar a la unidad seleccionada
      MsgBox "Fallo en acceso a la unidad"
   End If
   On Error GoTo 0 ' Desactivar la detección de errores
End Sub
```

- Graba el programa como **Prog98.frm y Prog98.vbp** y pruébalo

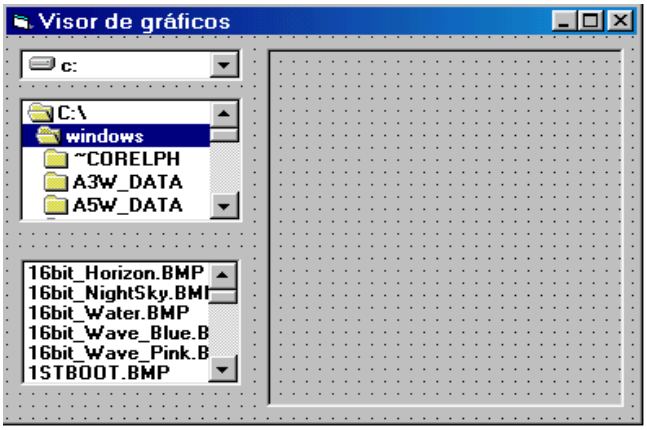
d) Vamos a ver como puede crearse un aplicación que funcione mediante operaciones arrastra y colocar (Drag & Drop).

Crea un nuevo proyecto
Hemos de conseguir una ventana que permita navegar por las unidades y directorios del equipo, buscando archivos gráficos. Una vez encontrado, lo arrastramos y colocamos sobre un cuadro de dibujo en el que se visualizará su contenido.



CREACIÓN

1. Añadir al formulario por defecto una cuadro de lista de directorios, otro de unidades, otro de archivos y una imagen.



2. Especificar las siguientes propiedades en tiempo de diseño:

Objeto	**Propiedad**	**Valor**
Form1	Name	frmVisor
	Caption	Visor de gráficos
	FontBold	True
Drive1	DragIcon	ICONS\DRAGDROP\DROP1PG
File1	DragIcon	ICONS\DRAGDROP\DROP1PG
	Pattern	*.ico;*.bmp;*.wmf
Dir1	DragIcon	ICONS\DRAGDROP\DROP1PG
Image1	BorderStyle	1 (Fixed Single)
	Stretch	True

3. Escribir los siguientes procedimientos de evento:

```
Private Sub Dir1_Change()
  'Enlazar directorios y ficheros
  file1.Path = Dir1.Path
End Sub

Private Sub Drive1_Change()
  'Enlazar unidades y directorios
  Dir1.Path = Drive1.Drive
End Sub
```

```
Private Sub File1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
  'Cambiar a icono de arrastre
  file1.DragIcon = Drive1.DragIcon
  'Iniciar modo de arrastre
  file1.Drag
End Sub

Private Sub Image1_DragDrop(Source As Control, X As Single, Y As Single)
  Dim Fichero As String
  'Añadir \ si el fichero está en el raíz
  If Mid(file1.Path, Len(file1.Path)) = "\" Then
    Fichero = file1.Path & file1.filename
  Else
    Fichero = file1.Path & "\" & file1.filename
  End If
  'Cargar la imagen
  image1.Picture = LoadPicture(Fichero)
End Sub

Private Sub Image1_DragOver(Source As Control, X As Single, Y As Single, State As Integer)
  Select Case State
    Case 0
      'Icono de prohibición se si sale de la imagen
      file1.DragIcon = Dir1.DragIcon
    Case 1
      'Icono de colocar si está dentro de la imagen
      file1.DragIcon = Drive1.DragIcon
  End Select
End Sub
```

5. Guardar el formulario y el proyecto en los archivos **Prog99.frm y Prog99.vbp**, respectivamente.

OBSERVA:
Para iniciar el arrastre de un elemento de la lista de archivos se ha utilizado el evento MouseDown de ésta, mediante la invocación del método Drag, aunque previamente debe cambiarse el icono de arrastre. El icono también se cambia cuando el elemento está siendo arrastrado en una zona en la que no es posible colocarlo.

En Visual Basic hay tres tipos de ficheros: Secuenciales, Aleatorios y Binarios

Ficheros Secuenciales
Nos permite guardar información de cualquier longitud. La información se guarda colocando un carácter tras otro. Son los más sencillos de manejar, y los utilizados para guardar texto en formato ASCII.

Abrir un fichero secuencial
* **Open** nombreF **For Output As** #númeroC
Crea un fichero de nombre **nombreF**, si este fichero ya existe, lo borra.
* **Open** nombreF **For Append As** #númeroC
Abre un fichero de nombre **nombreF** ya existente para **añadir** información
númeroC es un número entero entre 1 y 255 que representa una zona de la memoria reservada para el fichero que abrimos o creamos.

Cerrar un fichero secuencial
* Close #númeroC

Escribir en un fichero secuencial
Podemos utilizar dos métodos: Print o Write

Print
Sirve para introducir la información de forma secuencial
Ejemplo: Print #1, Text1.Text
Guardará en el fichero del canal 1, el contenido del cuadro de texto.

Para acceder al contenido de un fichero escrito por "Print":
- Abrimos el fichero:
  Open nombreF For Input As #númeroC
- Para leer los datos podemos utilizar **Input** o **Line Input**

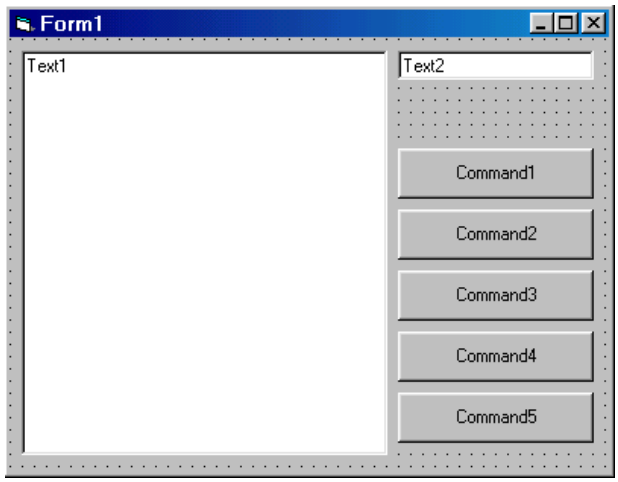
Input(1, #numeroC) = extrae un carácter
LOF(numeroC) = longitud total del fichero
Input(LOF(1),#1) = leemos todo el fichero que hay en el canal 1

"Input" devuelve todos los caracteres que lee, incluyendo puntos y coma retornos de carro, avances de línea, comillas y espacios iniciales.

La instrucción **Line Input** se utiliza para extraer una línea completa:
  Line Input #numeroC, nombreV
Extrae una línea completa y la asigna a "nombreV"

e) New Project

- Inserta:



- Propiedades:
  Text1
    Name = TBTexto
    Multiline = True
    ScrollBars = Vertical

  Text2
    Text=

  Command1
    Caption = Crear – Guardar

- Código:

```
Private Sub Command1_Click()
  Open Text2.Text For Output As #1
  Print #1, TBTexto.Text
```

```
    Close #1
End Sub
```

- Graba el formulario como **Prog100.frm** y el proyecto: **Prog100.vbp** en *TuCarpeta*

- Ejecuta el programa:
  - En el **TextBox1** escribe lo que quieras
  - En el **TextBox2** escribe **pepe**
  - Clic en [Crear – Guardar]

- Investiga utilizando el "Bloc de Notas" del Windows, el contenido del fichero **pepe**, que tienes en la carpeta del **Visual Basic**

Nos gustaría **grabar** los ficheros en *TuCarpeta*, basta que cambies la línea:

```
Open Text2.Text For Output As #1
```

Por

```
Open App.Path & "\" & Text2.Text For Output As #1
```

- Graba de nuevo nuestro programa en *TuCarpeta*

- Ejecuta de nuevo el programa:
  - En el TextBox1 escribe lo que quieras
  - En el TextBox2 escribe **pepe.txt**
  - Clic en [Crear – Guardar]

- Investiga utilizando el "Bloc de Notas", el contenido del fichero **pepe.txt** que tienes en *TuCarpeta*

- Vuelve a ejecutar el programa:
  - En el TextBox1 escribe lo que quieras, pero que sea distinto a lo que has escrito antes.
  - En el TextBox2 escribe lo mismo: **pepe.txt**

- Investiga el contenido del fichero **pepe.txt**

Espero que te haya funcionado correctamente: Al utilizar la instrucción **Open … For Output …** sólo podemos crear un nuevo fichero.

Continuemos:

- Accede al módulo del formulario y escribe:

```
Private Sub Command2_Click()
    Open App.Path & "\" & Text2.Text For Append As #1
    Print #1, TBTexto.Text
    Close #1
End Sub
```

- Cambia la propiedad **Caption** del **Command2** por **Crear o Añadir**

Investiga el funcionamiento del botón (espero que te haga lo que dice su "caption")

- Añade el siguientecódigo:

```
Private Sub Command3_Click()
    Dim vartexto As String
    Open App.Path & "\" & Text2.Text For Input As #1
    vartexto = Input(LOF(1), #1)
    Close #1
```

```
    TBTexto = vartexto
End Sub

Private Sub Command4_Click()
    Dim vartexto As String
    Open App.Path & "\" & Text2.Text For Input As #1
    Do Until EOF(1)
        vartexto = Input(1, #1)
        TBTexto = TBTexto & vartexto
    Loop
    Close #1
End Sub

Private Sub Command5_Click()
    Dim vartexto As String
    Open App.Path & "\" & Text2.Text For Input As #1
    Do Until EOF(1)
        Line Input #1, vartexto
        TBTexto = TBTexto & vartexto & vbCrLf
    Loop
    Close #1
End Sub
```

- Cambia las propiedades:

Command3
Caption = Abrir Todo

Command4
Caption = Abrir de 1 en 1

Command5
Caption = Abrir por líneas

- Graba de nuevo el programa (Prog100)

- Sólo nos queda probar exhaustivamente el programa. Debes tener en cuenta que “Leer todo el fichero de golpe” (Command3), sólo se puede hacer si el fichero es menor de 64 Kb.

f) Vamos a mejorar el programa anterior

- Añade el formulario **Prog97b.frm** ...

Menú Project
Add Form
Activa la solapa “Existing”
Selecciona el “Prog97b.frm” de *TuCarpeta*
[Open]

- Accede al módulo del form2 (Prog97b.frm) y cambia su código a:

```
Private Sub Command1_Click()
    If File1.ListIndex <> -1 Then
        Form1.Text2.Text = Dir1.Path & "\" & File1.List(File1.ListIndex)
        Unload Me
    End If
End Sub

Private Sub Command2_Click()
    Form1.Text2.Text = ""
```

```
        Unload Me
    End Sub
```

Los procedimientos Drive1_Change, Dir1_Change y File1_Click() no los modifiques.

- Graba el programa como:
  form Prog97b.frm = **Prog101b.frm**
  form Prog100.frm = **Prog101a.frm**
  Proyecto = **Prog101.vbp**

- Añade al formulario principal (Prog101a.frm) un nuevo botón de comando con:
  Name = Fichero
  Caption = Fichero?

- Añade el procedimiento:

```
Private Fichero_Click()
    Form2.Show
End Sub
```

- Y en el módulo anterior (form1), corrige la línea:
  Open App.Path & "\" & Text2.Text ...

por

  Open Text2.Text …

- Vuelve a grabar el proyecto con el mismo nombre y prueba el funcionamiento del programa.

g) Crea un fichero de texto con el "Bloc de Notas", con el siguiente contenido:

45 592
152 491
891 15
9 72
100 69

Graba el fichero con el nombre **Prog102.txt** en *TuCarpeta*

El problema que nos planteamos, es "leer" los números del fichero anterior y sumarlos por líneas.
La solución podría ser la siguiente:

- New Project

- Inserta un CommandButton en el ángulo inferior derecho del form y

```
Private Sub Command1_Click()
    Dim x(1 To 5) As Long, y(1 To 5) As Long
    Dim vartexto As String
    Open App.Path & "\Prog102.txt" For Input As #1
    j = 0
    Do Until EOF(1)
        j = j + 1
        Line Input #1, vartexto
        For i = 1 To Len(vartexto)
            If Mid(vartexto, i, 1) = " " Then
                x(j) = Left(vartexto, i - 1)
                y(j) = Right(vartexto, Len(vartexto) - i)
            End If
        Next
    Loop
    Close #1
    For j = 1 To 5
        Print x(j) & " - " & y(j)
    Next
```

```
    For j = 1 To 5
      Print x(j) + y(j)
    Next
End Sub
```

- Graba el programa como **Prog102.frm, Prog102.vbp** y pruébalo

h) **La instrucción Write**
Es la otra forma de introducir datos en un fichero secuencial. Permite introducir varias informaciones que posteriormente se podran leer de forma separada con la instrucción **Input #**
Lo que hacemos en realidad al escribir datos mediante la instrucción **Write** es escribir estos datos en un fichero secuencial utilizando una coma como separador entre los distintos datos.

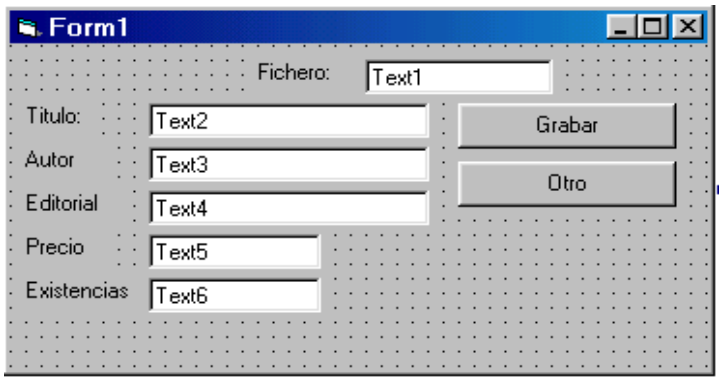
Un fichero secuencial con datos introducidos mediante la instrucción **Write** tendrá la forma:

"Pepe","Hola","153","28"
"Paco","Adios","100","7"
.....
.....

Vamos a ver como funciona en la práctica:

- New Project

- Inserta:



- Código:

```
Private Sub Command1_Click()
   Open App.Path & "\" & Text1.Text For Append As #1
   Write #1, Text2, Text3, Text4, Text5, Text6
   Close #1
End Sub

Private Sub Command2_Click()
   Text2.Text = ""
   Text3.Text = ""
   Text4.Text = ""
   Text5.Text = ""
   Text6.Text = ""
   Text2.SetFocus
End Sub
```

- Graba el programa como **Prog103.frm, Prog103.vbp** y pruébalo.

Investiga el contenido del fichero que acabas de generar, al probar el proyecto.

Si leyeramos este fichero con la instrucción **Input o Line Input**, lo leeríamos tal como está, con sus comas y comillas dobles. No sería la forma mas adecuada, ya que lo que queremos es obtener sucesivos

datos de autores, títulos, editoriales, precios y existencias. Para sacar estos datos debemos leerlos con la instrucción **Input #**

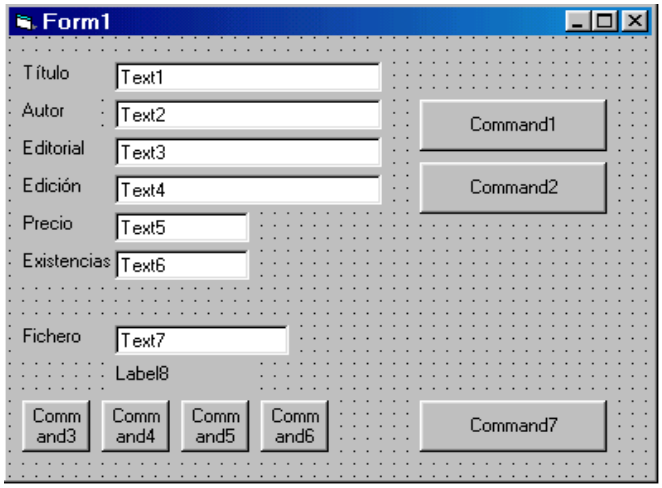
- En el formulario anterior inserta un nuevo CommandButton: **Command3** de caption “Leer Registros”. Y de código:

```
Private Sub Command3_Click()
   Open App.Path & "\" & Text1.Text For Input As #1
   Do Until EOF(1)
      parada = True
      Input #1, a, b, c, d, e
      Text2.Text = a
      Text3.Text = b
      Text4.Text = c
      Text5.Text = d
      Text6.Text = e
      Refresh
      Do While parada = True
         x = x + 1
         If x = 1000000 Then
            parada = False
            x = 0
         End If
      Loop
   Loop
   Close #1
End Sub
```

- Graba el programa con el mismo nombre (Prog103) y pruébalo.

i) Vamos a mejorar el programa anterior …
- New Project

- Inserta:



- Propiedades:

Command1
Caption = LEER

Command2
Caption = INTRODUCIR

Command3
Caption = <<

Command4
Caption = <

Command5
Caption = >

Command6
Caption = >>

Command7
Caption = SALIR

Label8
BorderStyle = 1-Fixed Single
En este “label” hemos de conseguir
El número de registros del fichero

- Inserta un módulo:
Menú Project
Add Module
Y escribe el siguiente código:

```
Type RegistroLibro
  Autor As String
  Titulo As String
  Editorial As String
  Edicion As String
  Precio As Integer
  Existencias As Integer
End Type
```

Acabamos de **definir** un nuevo **tipo de variable**

- Graba el módulo como **Prog104.bas**, el formulario como **Prog104.frm** y el proyecto **Prog104.vbp** en *TuCarpeta*

- Escribe en el módulo del formulario:

```
Dim NR As Integer 'representa el número de registros
Dim registrolibros() As RegistroLibro
Dim NRP As Integer  ' número del registro presentado
' Observa que declaramos un array del tipo RegistroLibro

Private Sub Command2_Click()
  Titulo = Text1.Text
  Autor = Text2.Text
  Editorial = Text3.Text
  Edicion = Text4.Text
  Precio = Val(Text5.Text)
  Existencias = Val(Text6.Text)
  Open App.Path & "\" & Text7.Text For Append As #1
  Write #1, Titulo, Autor, Editorial, Edicion, Precio, Existencias
  Close #1
End Sub
```

Prueba el programa para crear un nuevo fichero con varios registros.
Si investigas el contenido del fichero observarás que los campos correspondientes al precio y a las existencias, no aparecen entre comillas, ya que se trata de campos numéricos (Integer).

- Escribe:

```
Private Sub Command1_Click()
  Text1.Text = ""
  Text2.Text = ""
  Text3.Text = ""
  Text4.Text = ""
Text5.Text = ""
  Text6.Text = ""
  NR = 0
  Open App.Path & "\" & Text7.Text For Input As #1
  Do Until EOF(1)
    a = Input(1, #1)
    If a = Chr(13) Then NR = NR + 1
  Loop
```

```
    Close #1
    ' Acabo de contar todos los returns del fichero
    ' por eso en NR tengo el número de registros
    Label8.Caption = Str(NR) + " Registros"

    ' redimensionamos la matriz a NR elementos
    ReDim registrolibros(1 To NR)
    Open App.Path & "\" & Text7.Text For Input As #1

    For i = 1 To NR
        Input #1, registrolibros(i).Titulo, registrolibros(i).Autor, _
          registrolibros(i).Editorial, registrolibros(i).Edicion, _
          registrolibros(i).Precio, registrolibros(i).Existencias
    Next
    Close #1
    NRP = 1
    Text1.Text = registrolibros(NRP).Titulo
    Text2.Text = registrolibros(NRP).Autor
    Text3.Text = registrolibros(NRP).Editorial
    Text4.Text = registrolibros(NRP).Edicion
    Text5.Text = registrolibros(NRP).Precio
    Text6.Text = registrolibros(NRP).Existencias
End Sub
```

- Prueba lo que hemos hecho hasta ahora, ejecutando el programa **Prog104**

Por último, escribe los siguientes procedimientos:

```
Private Sub Command3_Click()
    NRP = 1
    Text1.Text = registrolibros(NRP).Titulo
    Text2.Text = registrolibros(NRP).Autor
    Text3.Text = registrolibros(NRP).Editorial
    Text4.Text = registrolibros(NRP).Edicion
    Text5.Text = registrolibros(NRP).Precio
    Text6.Text = registrolibros(NRP).Existencias
End Sub

Private Sub Command4_Click()
    NRP = NRP - 1
    If NRP = 0 Then
        MsgBox "No hay más registros en esta dirección"
        NRP = 1
    Else
        Text1.Text = registrolibros(NRP).Titulo
        Text2.Text = registrolibros(NRP).Autor
        Text3.Text = registrolibros(NRP).Editorial
        Text4.Text = registrolibros(NRP).Edicion
        Text5.Text = registrolibros(NRP).Precio
        Text6.Text = registrolibros(NRP).Existencias
    End If
End Sub

Private Sub Command5_Click()
    NRP = NRP + 1
    If NRP > NR Then
        MsgBox "No hay más registros en esta dirección"
        NRP = NR
    Else
        Text1.Text = registrolibros(NRP).Titulo
        Text2.Text = registrolibros(NRP).Autor
```

```
        Text3.Text = registrolibros(NRP).Editorial
        Text4.Text = registrolibros(NRP).Edicion
        Text5.Text = registrolibros(NRP).Precio
        Text6.Text = registrolibros(NRP).Existencias
    End If
End Sub

Private Sub Command6_Click()
    NRP = NR
    Text1.Text = registrolibros(NRP).Titulo
    Text2.Text = registrolibros(NRP).Autor
    Text3.Text = registrolibros(NRP).Editorial
    Text4.Text = registrolibros(NRP).Edicion
    Text5.Text = registrolibros(NRP).Precio
    Text6.Text = registrolibros(NRP).Existencias
End Sub

Private Sub Command7_Click()
    End
End Sub
```

- Graba el programa con el mismo nombre: **Prog104.bas**
Prog104.frm
Prog104.vbp

y pruébalo.

j) **Ficheros Aleatorios (Random)**

Un fichero aleatorio es un conjunto de registros, todos ellos de la misma longitud.

Los ficheros aleatorios nos permiten guardar una información similar a la del apartado anterior (libros de una biblioteca), pero para leerla no es necesario leer todo el fichero (como hacíamos en el fichero **secuencial** del apartado anterior), sino simplemente acceder a los registros que nos interesen.

También podemos realizar un cambio en un registro, sin alterar los demás.

Todo esto tiene un precio: en los archivos aleatorios cada dato tiene una longitud asignada, longitud que no se puede sobrepasar y si no la sobrepasamos perdemos esta capacidad sobrante.

- Para abrir un fichero aleatorio:
Open Nfichero For Random As #Ncanal Len=Longitud Reg.

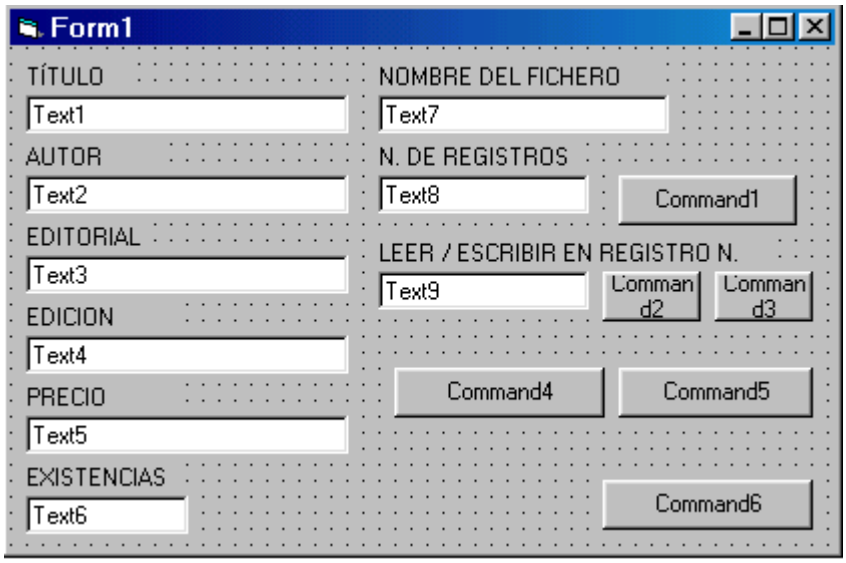
- Para escribir datos:
Put #Ncanal, Nregistro, Variable
La “Variable”, representa el contenido del registro.

- Para leer datos:
Get #Ncanal, Nregistro, Variable

Vamos a verlo en la práctica:

- New Project

- Inserta:



- Propiedades:

Text8
Text =

Command1
Caption = EXAMINAR

Command2
Caption= <----

Command3
Caption = ---->

Command4
Caption = LEER

Command5
Caption = ESCRIBIR

Command6
Cation = SALIR

- Escribe en el módulo del formulario:

```
Dim RegLibros As Registro

Private Sub Command1_Click()
   Close            'cierra todo fichero abierto
   Open App.Path & "\" & Text7.Text For Random As #1 Len = 88
   x = LOF(1)         ' longitud del fichero
   Text8.Text = x / 88 ' número de registros
End Sub

Private Sub Command5_Click()
   RegLibros.Titulo = Text1.Text
   RegLibros.Autor = Text2.Text
   RegLibros.Editorial = Text3.Text
   RegLibros.Edicion = Text4.Text
   RegLibros.Precio = Text5.Text
   RegLibros.Existencias = Text6.Text
   Put #1, Val(Text9.Text), RegLibros
End Sub
```

- Graba el programa como **Prog105.bas, Prog105.frm y Prog105.vbp**

Prueba lo que hemos hecho hasta ahora de la siguiente forma:

- Ejecuta el programa

- En “NOMBRE DEL FICHERO” escribe **Prog105**

- Clic en [EXAMINAR]
Si todo funciona correctamente, en “N de Registros”, aparecerá cero

- Inventa los datos de un libro

- En “LEER/ESCRIBIR EN REGISTRO N.”, escribe 1

- Clic en [Escribir]

- Clic en [Examinar]

- Inventa los datos de otro libro

- En “LEER/ESCRIBIR EN REGISTRO N.”, escribe 2

- Clic en [Examinar]

“Cierra” el programa
- Investiga utilizando el “Bloc de Notas”, el contenido del fichero **Prog105** que tienes en *TuCarpeta*

- Escribe los procedimientos:

```
Private Sub Command4_Click()
   Get #1, Val(Text9.Text), RegLibros
   Text1.Text = RegLibros.Titulo
   Text2.Text = RegLibros.Autor
   Text3.Text = RegLibros.Editorial
   Text4.Text = RegLibros.Edicion
   Text5.Text = RegLibros.Precio
   Text6.Text = RegLibros.Existencias
End Sub

Private Sub Command6_Click()
   End
End Sub

Private Sub Command2_Click()
   Dim x As Integer
   x = Val(Text9.Text)
   If x = 1 Then
      MsgBox "No hay nada"
   Else
      x = x - 1
      Text9.Text = x
   End If
End Sub

Private Sub Command3_Click()
   Dim x As Integer
   x = Val(Text9.Text)
   If x >= Val(Text8.Text) Then
      MsgBox "No hay nada"
   Else
      x = x + 1
      Text9.Text = x
   End If
End Sub
```

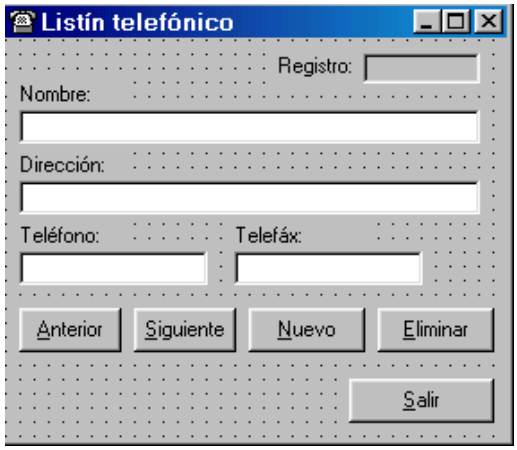
- Graba el programa con el mismo nombre **Prog105** y pruébalo

k) Vamos a ver como podemos utilizar el acceso aleatorio a archivos para manejar archivos de datos de forma similar a como se hace con una base de datos.

Se trata de crear un listín telefónico en forma de base de datos, almacenado en el fichero LISTIN.DAT. Cada registro contiene un nombre una dirección, un número de teléfono y uno de fax. es posible desplazarse hacia adelante, hacia atrás, añadir registros y eliminarlos.

Crea un nuevo proyecto

1. Añadir al formulario por defecto cinco botones de comando, cuatro cuadros de texto y seis etiquetas.



2. Establecer las siguientes propiedades en tiempo de diseño:

Objeto	**Propiedad**	**Valor**
Form1	Name	frmListin
	Caption	Listín telefónico
	Icon	ICONS\COMM\PHONE01.ICO
Command1	Name	cmdSiguiente
	Caption	&Siguiente
Command2	Name	cmdAnterior
	Caption	&Anterior
	Enabled	False
Command3	Name	cmdSalir
	Caption	&Salir
Command4	Name	cmdEliminar
	Caption	&Eliminar

Command5	Name	cmdNuevo
	Caption	&Nuevo
Text1	Name	txtNom
	FontBold	True
	MaxLength	25
Text2	Name	txtDir
	FontBold	True
	MaxLength	25
Text3	Name	txtTel
	FontBold	True
	MaxLength	15
Text4	Name	txtFax
	FontBold	True
	MaxLength	15
Label1	Caption	Registro:
Label2	Name	lblReg
	Caption	(vacío)
	Alignment	1 (Right Justify)
	BorderStyle	1 (Fixed Single)
	FontBold	True
Label3	Name	lblNom
	Caption	Nombre:
Label4	Name	lblDir
	Caption	Dirección:
Label5	name	lblTel
	Caption	Teléfono:
Label6	Name	lblFax
	Caption	Telefáx:

3. Escribir las siguientes declaraciones en la sección general:

```
Dim RegNum As Long      'Número del registro actual
Dim RegTot As Long      'Número total de registros
Dim RegBuf As Reg       'Buffer del registro actual
```

4. Escribir los siguientes procedimientos generales:

```
Public Sub Abrir()
  'Abrir el archivo de datos en modo aleatorio
  Open App.Path & "\" & "LISTIN.DAT" For Random As #1 Len = 80
  RegNum = 1
  RegTot = Int(LOF(1) / 80)
  'Si no hay registros añadir nuevo
  If RegTot = 0 Then
    Grabar
    RegTot = 1
  Else
    Leer
  End If
  'Verificar si es el primero o el último
  cmdAnterior.Enabled = Not (RegNum = 1)
  cmdSiguiente.Enabled = Not (RegNum = RegTot)
End Sub

Public Sub Leer()
  Get #1, RegNum, RegBuf
  txtNom.Text = RTrim(RegBuf.Nom)
  txtDir.Text = RTrim(RegBuf.Dir)
  txtTel.Text = RTrim(RegBuf.Tel)
  txtFax.Text = RTrim(RegBuf.Fax)
  lblReg.Caption = RegNum & "/" & RegTot
End Sub
```

```
Public Sub Grabar()
  RegBuf.Nom = txtNom.Text
  RegBuf.Dir = txtDir.Text
  RegBuf.Tel = txtTel.Text
  RegBuf.Fax = txtFax.Text
  Put #1, RegNum, RegBuf
End Sub
```

5. Escribir los siguientes procedimientos de evento:

```
Private Sub cmdEliminar_Click()
  Dim N As Long
  'Avisar si sólo hay uno
  If RegTot = 1 Then
    MsgBox "No se puede eliminar el registro" & Chr(13) & _
          "cuando sólo queda uno", vbCritical
    Exit Sub
  End If
  'Copiar a archivo temporal, excepto el registro actual
  Open App.Path & "\" & "LISTIN.TMP" For Random As #2 Len = 80
  For N = 1 To RegTot
    If N <> RegNum Then
      Get #1, N, RegBuf
      Put #2, , RegBuf
    End If
  Next
  'Recordar la posición del registro eliminado
  N = RegNum
  'Cerrar ambos archivos
  Close #1
  Close #2
  'Eliminar archivo actual
  Kill App.Path & "\" & "LISTIN.DAT"
  'Renombrar el archivo temporal
  Name App.Path & "\" & "LISTIN.TMP" As App.Path & "\" & "LISTIN.DAT"
  'Volver a abrir
  Abrir
  'Resituar el registro anterior
  RegNum = IIf(N > 1, N - 1, 1)
  Leer
  'Verificar si es el primero o el último
  cmdAnterior.Enabled = Not (RegNum = 1)
  cmdSiguiente.Enabled = Not (RegNum = RegTot)
End Sub

Private Sub cmdNuevo_Click()
  'Grabar el registro actual
  Grabar
  'Vaciar el buffer
  txtNom.Text = ""
  txtDir.Text = ""
  txtTel.Text = ""
  txtFax.Text = ""
  'Situar el registro después del último
  RegTot = RegTot + 1
  RegNum = RegTot
  'Grabar registro vacío
  Grabar
  'Verificar si es el primero o el último
  cmdAnterior.Enabled = Not (RegNum = 1)
```

```
  cmdSiguiente.Enabled = Not (RegNum = RegTot)
End Sub

Private Sub cmdAnterior_Click()
  'Grabar Actual
  Grabar
  'Leer el anterior
  RegNum = RegNum - 1
  Leer
  'Verificar si es el primero o el último
  cmdAnterior.Enabled = Not (RegNum = 1)
  cmdSiguiente.Enabled = Not (RegNum = RegTot)
End Sub

Private Sub cmdSalir_Click()
  'Grabar el registro actual
  Grabar
  'Terminar
  Close #1
  Unload Me
End Sub

Private Sub cmdSiguiente_Click()
  'Grabar Actual
  Grabar
  'leer el siguiente
  RegNum = RegNum + 1
  Leer
  'Verificar si es el primero o el último
  cmdAnterior.Enabled = Not (RegNum = 1)
  cmdSiguiente.Enabled = Not (RegNum = RegTot)
End Sub

Private Sub Form_Load()
  Abrir
  Leer
End Sub
```

6. Añadir un módulo estándar y escribir la siguiente definición en su sección general:

```
Type Reg
  Dim Nom As String * 25
  Dim Dir As String * 25
  Dim Tel As String * 15
  Dim Fax As String * 15
End Type
```

6. Guardar el formulario, el módulo estándar y el proyecto en los archivos **Prog106.frm, Prog106.bas y Prog106.vbp**, respectivamente.

OBSERVA

Para manejar los datos de cada registro se crea un el tipo definido por el usuario Reg, que debe situarse en el módulo estándar. Éste permite a las funciones Leer y Grabar intercambiar la información con el archivo. Los botones "Anterior" y "Siguiente" se deshabilitan cuando los registros son el primero o el último, respectivamente. Los registro nuevos se sitúan al final del archivo. Nótese que para borrar un registro es necesario copiar el archivo a otro temporal, LISTIN.TMP, excluyendo el registro a eliminar. Este archivo es renombrado nuevamente como LISTIN.DAT y abierto nuevamente en la posición anterior a la que se encontraba.

XI.- BASES DE DATOS

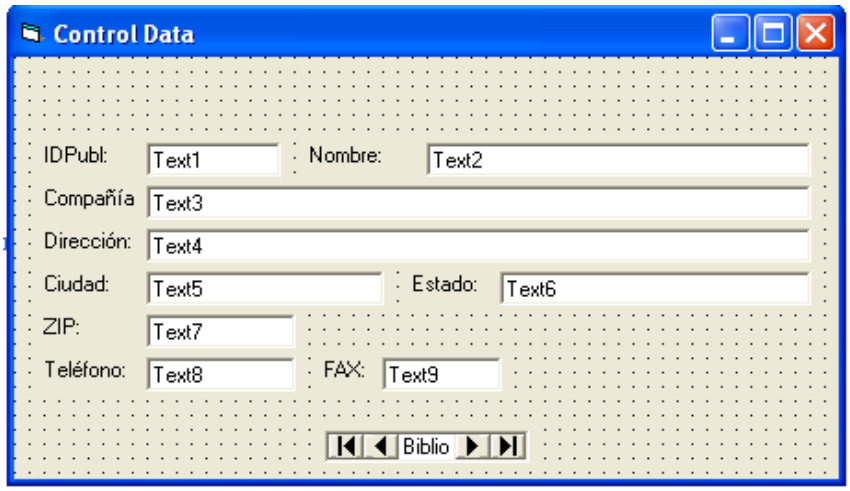
El Control DATA

- Copia el fichero **Biblio.mdb**, base de datos formato Access 97 de ejemplo que yienes en **c:\Archivos de Programas\Microsoft Visual Studio\VB98** en *TuCarpeta*
- Crea un nuevo proyecto del tipo **Controles de VB Edición Profesional**
- Clic-Clic en el **Control Data**, para insertarlo en el Form1
- Propiedades:

Form1	(Name)	BaseD
	Caption	Control DATA
Data	(Nombre)	Datos
	Caption	Biblio
	DataBaseName	c:\TuCarpeta\Biblio.mdb
	RecordSource	Publishers

El **RecordSource** = **Publishers** no es mas que una tabla de la base de datos **Biblio.mdb**

- Inserta:



- Propiedades:

Text1	(Nombre)	IDPubl
	Text	
	DataSource	Datos
	DataField	PubID
Text2	(Nombre)	Nombre
	Text	
	DataSource	Datos
	DataField	Name
Text3	(Nombre)	NomCom
	Text	
	DataSource	Datos
	DataField	Company Name
Text4	(Nombre)	Dirección
	Text	
	DataSource	Datos
	DataField	Address
Text5	(Nombre)	Ciudad
	Text	
	DataSource	Datos
	DataField	City

Text6	(Nombre)	Estado
	Text	
	DataSource	Datos
	DataField	State
Text7	(Nombre)	Zip
	Text	
	DataSource	Datos
	DataField	Zip
Text8	(Nombre)	Teléfono
	Text	
	DataSource	Datos
	DataField	Telephone
Text9	(Nombre)	Fax
	Text	
	DataSource	Datos
	DataField	Fax

- Graba el formulario como **b01.frm** y el proyecto **bd01.vbp**
- Ejecuta la aplicación y prueba los diferentes botones del control **DATA**

Gestionar la base de datos con el control DATA

Con el programa **bd01** a la vista …

- Clic en el icono **"Editor de Menús"**
- Caption = &Base de Datos
  Name = BasDat
  [Siguiente]
  [→]
- Caption = &Añadir
  Name = mnuBaAña
  [Siguiente]
- Caption = &Modificar
  Name = mnuBaMod
  [Siguiente]
- Caption = &Borrar
  Name = mnuBaBor
  [Siguiente]
- Caption = -
  Name = rallita
  [Siguiente]
- Caption = &Salir
  Name = mnuBaSalir
  [Aceptar]
- Inserta 4 botones de comando en el formularios
- Propiedades:

Command1	(Name)	Primero
	Caption	&Primero
Command2	(Name)	Último
	Caption	&Último
Command1	(Name)	Siguiente
	Caption	&Siguiente
Command1	(Name)	Anterior
	Caption	&Anterior
Datos	Visible	False

- Código:

```
Private Sub Primero_Click()
   Datos.Recordset.MoveFirst
End Sub

Private Sub Siguiente_Click()
   Datos.Recordset.MoveNext
End Sub

Private Sub Último_Click()
   Datos.Recordset.MoveLast
End Sub

Private Sub Anterior_Click()
   Datos.Recordset.MovePrevious
End Sub

Private Sub mnuBaAña_Click()
   Datos.Recordset.AddNew
End Sub
```

- Inserta un nuevo botón en la parte inferior derecha del formulario y asigna las siguientes propiedades:

Command1	(Nombre)	Aceptar
	Caption	&Aceptar
	Visible	False

La opción **Datos.Recordset.AddNew** de la opción del menú **Añadir**, nos situará en el último registro y añadirá uno nuevo. Nosotros tendremos que rellenar los diferentes campos del formulario y pulsar en el botón **Aceptar**

- Modifica el procedimiento: **mnuBaAña_Click():**

```
Private Sub mnuBaAña_Click()
   Datos.Recordset.AddNew
   Aceptar.Visible = True
   Opcion = 1
End Sub
```

Observa que además de añadir un registro en blanco, hacemos visible el botón [Aceptar] y asignamos a la variable **Opción**, el valor 1

- Hemos de declarar esta variable (Opción), para ello sitúate en la parte **General** del Módulo del formulario y escribe:

```
Dim Opcion As Integer
```

- Código:

```
Private Sub mnuBaBorrar_Click()
   Aceptar.Visible = True
   Opcion = 3
End Sub

Private Sub mnuBaMod_Click()
   Datos.Recordset.Edit
   Aceptar.Visible = True
   Opcion = 2
End Sub

Private Sub mnuBaSalir_Click()
   End
End Sub

Private Sub Aceptar_Click()
   Select Case Opcion
```

```
            Case 1
               Datos.Recordset.Update
            Case 2
               Datos.Recordset.Update
            Case 3
               Datos.Recordset.Delete
         End Select
         Aceptar.Visible = False
      End Sub
```

- Graba los cambios hechos (**bd01.frm / bd01.vbp**)
- Ejecuta la aplicación probando las diferentes opciones.

El Administrador Visual de Datos (VisData)

- Menú Complementos
  Administrador visual de datos

Se acaba de abrir el **VisData**, utilidad que incorpora el Visual Basic 6.0, que nos permite crear bases de datos.

- Menú Archivo
  Nuevo

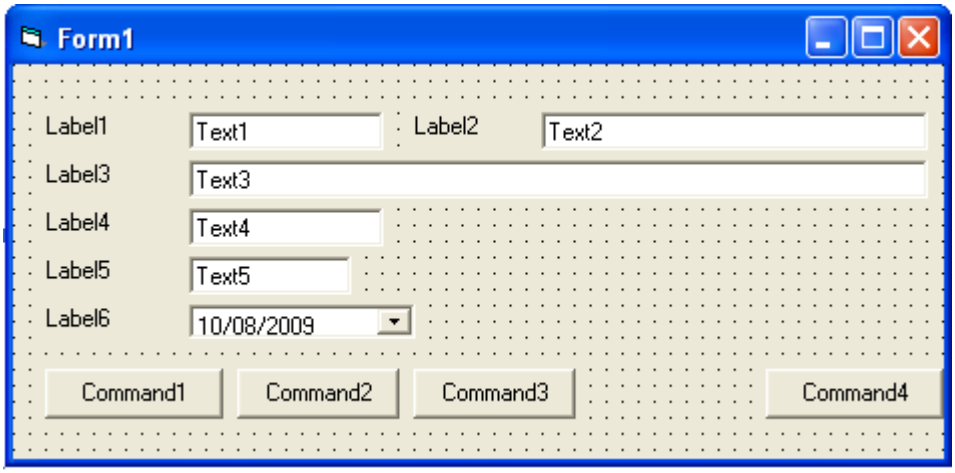
- Selecciona **Microsoft Access 7.0**
- Nombre: **Clientes** en *TuCarpeta*
- Accede al menú contextual de la **Ventana de base de datos** (botón derecho del ratón) y selecciona la opción **Nueva Tabla**
- Name de tabla: **Socios**
  [Agregar campo]
  Nombre: Nombre
  Tipo de Campo: Text
  Tamaño de campo: 15
  [Aceptar]
- "Agrega" los nuevos campos:

Nombre	**Tipo**	**Tamaño**
Apellidos	Text	30
Dirección	Text	50
Población	Text	15
Teléfono	Text	9
Cumpleaños	Date/Time	8

- [Cerrar]
- [Agregar índice]
- Nombre: Índice
- Clic-clic en el campo **Nombre** y en el campo **Apellidos**
- Observa que las opciones **Principal** y **Único** ya estan activadas.
  [Aceptar]
  [Cerrar]
  [Generar la tabla]
- Menú Archivo
  Salir

Gestionar una Base de Datos por DAO (Objetos de Acceso a Datos)

- Crea el siguiente formulario:



- Propiedades:

Form1	(Nombre) Caption	Socios Gestión de Socios
Label1	Caption	Nombre
Label2	Caption	Apellidos
Label3	Caption	Dirección
Label4	Caption	Población
Label5	Caption	Teléfono
Label6	Caption	Cumpleaños
Text1	(Nombre) MaxLength Text	Nombre 15
Text2	(Nombre) MaxLength Text	Apellidos 30
Text3	(Nombre) MaxLength Text	Dirección 50
Text4	(Nombre) MaxLength Text	Población 15
Text5	(Nombre) MaxLength Text	Teléfono 9
DTPicker1	(Nombre)	Cumpleaños

Command1	(Nombre)	Aceptar
	Caption	&Aceptar
Command2	(Nombre)	Modificar
	Caption	&Modificar
Command3	(Nombre)	Borrar
	Caption	&Borrar
Command4	(Nombre)	Salir
	Caption	&Salir

- Clic en el icono **"Editor de Menús"**

- Caption = &Gestión
  Name = Gestión
  [Siguiente]
  [→]
- Caption = &Altas
  Name = mnuAltas
  [Siguiente]

- Caption = &Modificar
  Name = mnuModificar
  [Siguiente]

- Caption = &Borrar
  Name = mnuBorrar
  [Siguiente]

- Caption = -
  Name = rallita
  [Siguiente]

- Caption = &Salir
  Name = mnuSalir
  [Aceptar]

- Para poder trabajar con DAO (Objetos de Acceso a Datos) hemos de activar la referencia correspondiente. Haz lo siguiente:

Menú Proyecto
Referencias
Seecciona la librería: **Microsoft DAO 3.6 Compatibility Library**
[Aceptar]

- Abre el **Módulo del formulario** y escribe en el apartado "General":

```
Option Explicit
Public BaseSocios As Database
Public TablaSocios As Recordset
Public Opción As Integer
Public Error As Integer
```

La variable "Opción" nos dará la opción: altas, modificar o borrar

- Código del **Form_Load**:

```
Private Sub Form_Load()
   Set BaseSocios = OpenDatabase("c:\TuCarpeta\Clientes")
   Set TablaSocios = BaseSocios.OpenRecordset("Socios", dbOpenTable)
   Opción = 0
```

```
End Sub
```

Observa:
- Abrimos la base de datos **Clientes** y consideramos como **Recordset** la tabla **Socios**.
- Asigna a los botones [Aceptar], [Modificar] y [Borrar] el valor **False** a la propiedad **Visible**

- Código para las opciones del menú:

```
Private Sub mnuAltas_Click()
   Aceptar.Visible = True
   Modificar.Visible = False
   Borrar.Visible = False
   TablaSocios.AddNew
   Opción = 1
End Sub

Private Sub mnuBorrar_Click()
   Borrar.Visible = True
   Modificar.Visible = False
   Aceptar.Visible = False
   Opción = 3
End Sub

Private Sub mnuModificar_Click()
   Modificar.Visible = True
   Aceptar.Visible = False
   Borrar.Visible = False
   Opción = 2
End Sub
```

- Código del botón [Aceptar]:

```
Private Sub Aceptar_Click()
   Validar
   If Error = 0 Then
      CopiaCampos
      TablaSocios.Update
      LimpiaCampos
      Aceptar.Visible = False
      Opción = 0
   Else
      Error = 0
   End If
End Sub
```

Observa que aparecen tres funciones:
- **Validar**: comprobará que todos los campos tengan valor, en este caso **Error = 0**, en caso contrario **Error = 1** y el registro no se grabará
- **Copia Campos:** el contenido de los campos de texto pasan a los campos correspondientes de la tabla de la base de datos.
- **LimpiaCampos:** borra el contenido de los campos de texto. Opción=0, indicará que estamos en **modo consulta**.

- Funciones:

```
Function Validar()
   If Nombre.Text = "" Then
      MsgBox "Campo NOMBRE obligatorio", vbCritical
      Nombre.SetFocus
      GoTo Salir
   End If
   If Apellidos.Text = "" Then
```

```
      MsgBox "Campo APELLIDOS obligatorio", vbCritical
      Apellidos.SetFocus
      GoTo Salir
    End If
    If Dirección.Text = "" Then
      MsgBox "Campo DIRECCIÓN obligatorio", vbCritical
      Dirección.SetFocus
      GoTo Salir
    End If
    If Población.Text = "" Then
      MsgBox "Campo POBLACIÓN obligatorio", vbCritical
      Población.SetFocus
      GoTo Salir
    End If
    If Teléfono.Text = "" Then
      MsgBox "Campo TELÉFONO obligatorio", vbCritical
      Teléfono.SetFocus
      GoTo Salir
    End If
    Exit Function
Salir:
    Error = 1
End Function


Function CopiaCampos()
    TablaSocios.Fields("Nombre") = Nombre
    TablaSocios.Fields("Apellidos") = Apellidos
    TablaSocios.Fields("Dirección") = Dirección
    TablaSocios.Fields("Población") = Población
    TablaSocios.Fields("Teléfono") = Teléfono
    TablaSocios.Fields("Cumpleaños") = Cumpleaños
End Function


Function LimpiaCampos()
    Nombre.Text = ""
    Apellidos.Text = ""
    Dirección.Text = ""
    Población.Text = ""
    Teléfono.Text = ""
End Function

Function LeeCampos()
    Nombre = TablaSocios.Fields("Nombre")
    Apellidos = TablaSocios.Fields("Apellidos")
    Dirección = TablaSocios.Fields("Dirección")
    Población = TablaSocios.Fields("Población")
    Teléfono = TablaSocios.Fields("Teléfono")
    Cumpleaños = TablaSocios.Fields("Cumpleaños")
End Function
```

- Código de los botones:

```
Private Sub Borrar_Click()
    TablaSocios.Delete
    LimpiaCampos
    Borrar.Visible = False
    Opción = 0
End Sub
```

```
Private Sub Modificar_Click()
   Validar
   If Error = 0 Then
      CopiaCampos
      TablaSocios.Update
      LimpiaCampos
      Modificar.Visible = False
      Opción = 0
   Else
      Error = 0
   End If
End Sub


Private Sub Salir_Click()
   TablaSocios.Close
   BaseSocios.Close
   End
End Sub
```

- Los botones [Modificar] y [Borrar] necesitan antes de pulsarlos que hayamos encontrado el registro, esto lo conseguiremos programando el campo de texto **Apellidos** (recuerda que nuestro índice era **nombre+apellidos**).

Escribe el siguiente código:

```
Private Sub Apellidos_LostFocus()
   If Opción = 2 Or Opción = 3 Then
      If Nombre.Text <> "" Then
         If Apellidos.Text <> "" Then
            TablaSocios.Index = "Índice"
            TablaSocios.Seek "=", Nombre.Text, Apellidos.Text
            If TablaSocios.NoMatch = True Then
               MsgBox "Este registro no existe", vbCritical
               Nombre.Text = ""
               Apellidos.Text = ""
               Nombre.SetFocus
            Else
               LeeCampos
               If Opción = 2 Then TablaSocios.Edit
            End If
         Else
            MsgBox "Teclee el apellido para realizar la búsqueda", vbInformation
            Apellidos.SetFocus
         End If
      Else
         MsgBox "Teclee un nombre para realizar la búsqueda", vbInformation
         Nombre.SetFocus
      End If
   End If
End Sub
```

- Graba el proyecto como **bd02** y pruébalo

El Control MSFlexGrid

- Con el **bd02** a la vista …

- Inserta una nueva opción al menú:
    Consultas
        Caption= &Consultas
        Name= mnuGesCon

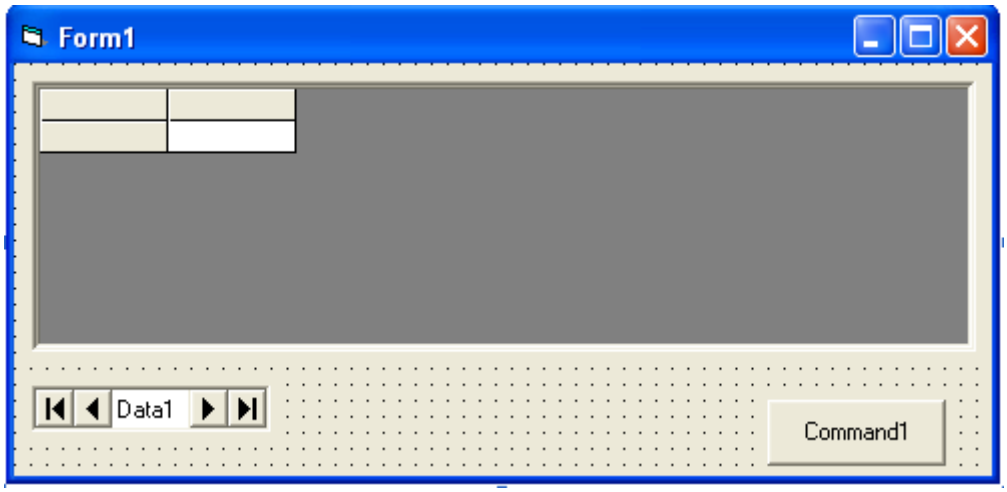
de código:

```
Private Sub mnuGesCon_Click()
   Consulta.Show 1
End Sub
```

Está claro que hemos de crear un nuevo formulario:

- Menú Proyecto
    Agregar formularios
        [Abrir]

- Inserta (el control superior es un **MSFlexGrid**):



- Propiedades:

Form1	(Nombre)	Consultarse
	Caption	Consulta de la tabla
MSFlexGrid	(Nombre)	Rejilla
	DataSource	Datos
	Cols	6
Data1	(Nombre)	Datos
	DataBaseName	c:\TuCarpeta\Clientes.mdb
	RecordSource	Socios
Command1	(Nombre)	Salir
	Caption	&Salir

- Graba el nuevo formulario como **Consul.frm**

- Código en el formulario **Consul**:

```
Private Sub Form_Load()
   Rejilla.ColWidth(0) = 0
   Rejilla.ColWidth(1) = 700
   Rejilla.ColWidth(2) = 1500
   Rejilla.ColWidth(3) = 2000
   Rejilla.ColWidth(4) = 1000
```

```
    Rejilla.ColWidth(5) = 1000
End Sub

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

- Para mejorar el aspecto del **MSFlexGrid** haz lo siguiente:
En el formulario **Consulta**, accede al Menú Contextual del **MSFlexGrid** y **Propiedades**:

[Color]
BlackColorFixed
Conjunto de colores
Colores estándar
Paleta de Colores
Azul
[Aplicar]

BackColor
Paleta de Colores
Texto de información sobre herramientas
[Aceptar]

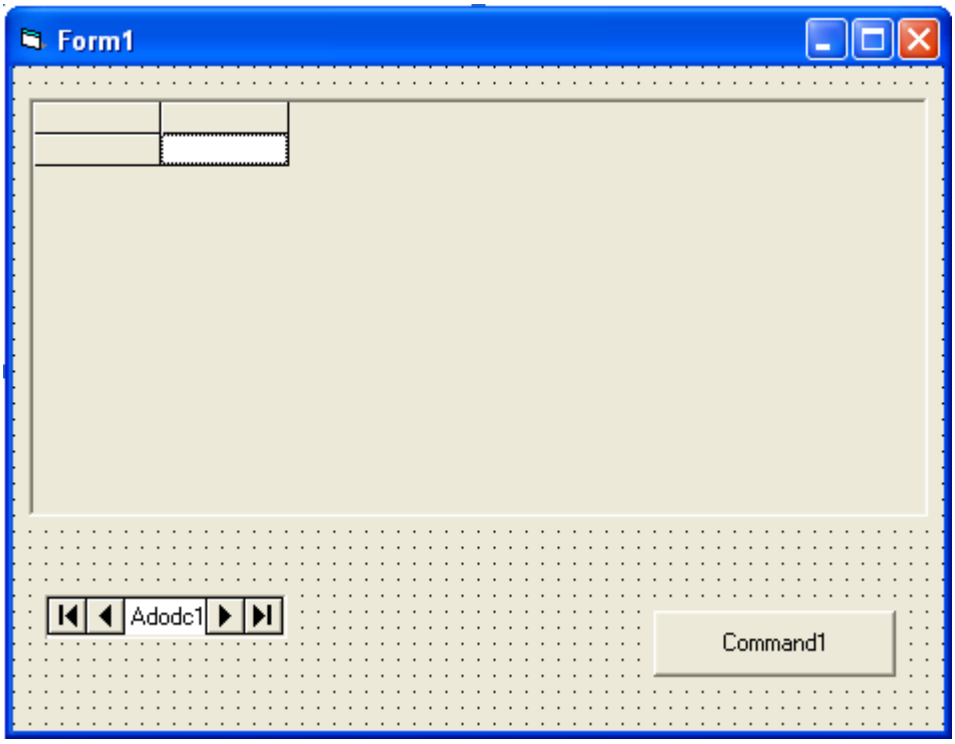
- Graba la aplicación con el mismo nombre **bd02** y pruébala.

El contro de datos ADODC y sus controles dependientes: DataList, DataCombo, MSHFlexGrid, ...

El control ADODC permite utilizar sentencias **SQL**

- Crea un nuevo proyecto con los controles profesionales de Visual Basic

- Inserta un control **MSHFlexGrid**, un ADODC y un botón de comando:



- Propiedades:

Form1	(Nombre)	Consulta
	Caption	Utilizar ADODC
ADODC1	(Nombre)	Datos
MSHFlexGrid1	(Nombre)	Rejilla
	DataSource	Datos
Command1	(Nombre)	Salir
	Caption	Salir

- Vamos a conectar el ADODC con nuestra base de datos …

Accede al Menú Contextual del **Adodc1**
Propiedades de ADODC
Usar cadena de conexión
[Generar]
Selecciona **Microsoft Jet 4.o OLE DB Provider**
[Siguiente]
Selecciona o escribe: **c:\Archivos de Programa\Microsoft Visual Studio\VB98\NWIND.MDB**
[Probar conexión]
[Aceptar]
[Aceptar]

[Origen de registros]
Texto del comando (SQL):
SELECT * FROM CLIENTES
[Aplicar]
[Aceptar]

- Graba la aplicación como **bd03.frm / bd03.vbp** y pruébala.

En lugar de un **MSHFlexGrid** podría ser un **DataGrid**, en efecto:

- Con el **bd03** a la vista, suprime el control **MSHFlexGrid** que tenemos en el formulario

- Clic – clic en el control **DataGrid** y asígnale el mismo tamaño que tenia el control anterior.

- Propiedades:

DataGrid	(Nombre)	Rejilla
	DataSource	Datos

- Accede al **Menú Contextual** de la **Rejilla** y **Recuperar Campos**
[Sí]

- Accede de nuevo al menú contextual de la rejilla y **Propiedades** …

- Activa todas las opciones de [General]

- [Diseño]
Column: NombreCompañía Width: 3000
y continua de la misma forma para cambiar la anchura de cada campo.

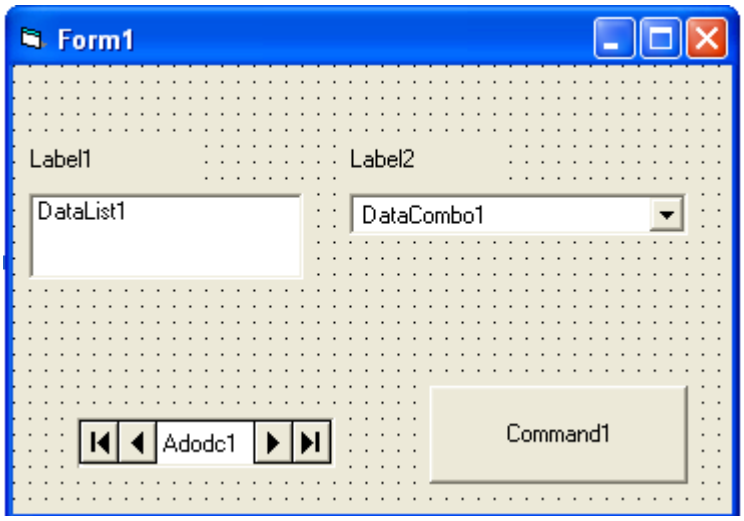
Observa que podemos cambiar la [Fuente] el [Color], el [Formato], etc.
Para acabar: [Aplicar]
[Aceptar]

- Graba de nuevo con el mismo nombre **bd03** y pruébalo.

Los Controles DataList y DataCombo

- Crea un nuevo proyecto utilizando los controles profesionales de Visual Basic.

- Inserta:



– Propiedades:

Label1	Caption	DataList
Label2	Caption	DataCombo
Command1	(Nombre)	Salir
	Caption	&Salir
DataList1	(Nombre)	Lista
DataCombo1	(Nombre)	Combo
Adodc1	(Nombre)	Datos

– Menú Contextual del ADODC:
Propiedades de ADODC
Usar cadena de conexión: [Generar]
Microsoft Jet 4.0 OLE DB Provider
[Siguiente]
Base de datos: c:\Archivos de Programa\Microsoft Visual Studio\VB98\NWIND.MDB
[Aceptar]
[Origen de regstros]
Texto del comando (SQL):SELECT * FROM CLIENTES
[Aplicar]
[Aceptar]

– Propiedades:

DataList1	DataSource	Datos
	RowSource	Datos
	ListField	ID Cliente
DataCombo1	DataSource	Datos
	RowSource	Datos
	ListField	Nombre Contacto

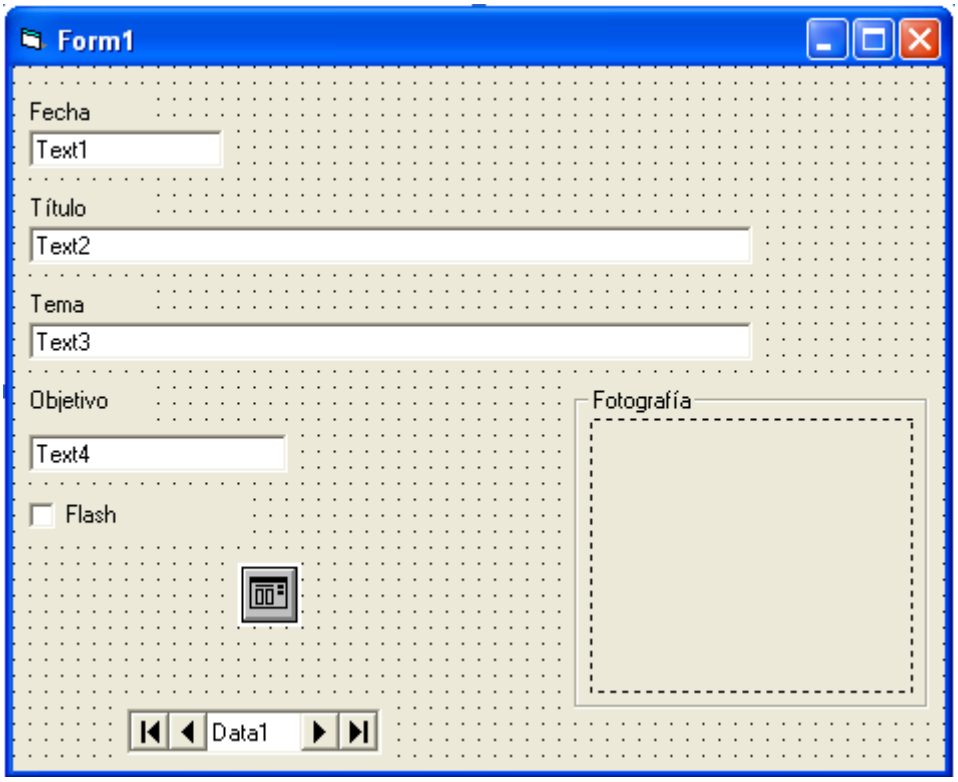
– Graba la apicación como **bd04.frm / bd04.vbp** y pruébala.

Base de Datos Gráfica

- Ejecuta el **VisData** para crear una base de datos **Microsoft Access, MDB de la versión 7.0** de nombre **FOTOGRAF.MDB** en *TuCarpeta*.
  Con una tabla de nombre **Fotografías** y estructura:

Fecha	Date/Time	-
Título	Text	50
Tema	Text	50
Objetivo	Text	50
Fotografía	Binary	-
Flash	Boolean	-

- Crea un nuevo proyecto Visual Basic con los controles de la versión profesional, e inserta en el "Form":



Observa que hemos insertado un **CommonDialog**, un frame de Caption = Fotografía, que contiene un control **Image** y un **CheckBox** aparte de las etiquetas y cuadros de texto que aparecen y un control **Data**.

- Propiedades:

Form1	(Name)	Fotografías
	Caption	Fotografías
Data1	(Name)	Fotografías
	DatabaseName	c:\TuCarpeta\FOTOGRAF.MDB
	RecordSource	Fotografías
	Align	2- AlignBottom
	Caption	Fotografías
	EOFAction	2-AddNew
Text1	(Name)	Fecha
	Text	
	DataSource	Fotografías
	DataField	Fecha

Text2	(Name)	Título
	Text	
	DataSource	Fotografías
	DataField	Título
Text3	(Name)	Tema
	Text	
	DataSource	Fotografías
	DataField	Tema
Text4	(Name)	Objetivo
	Text	
	DataSource	Fotografías
	DataField	Objetivo
CheckBox	(Name)	Flash
	Caption	Flash
	DataSource	Fotografías
	DataField	Flash
Image	(Name)	Fotografía
	DataSource	Fotografías
	DataField	Fotografía
	Strech	True
CommonDialog	(Name)	CajaComún

La propiedad **EOFAction = 2-AddNew** del Data nos permitirá añadir automáticamente registros a la tabla

- Código:

```
Private Sub Fotografía_Click()
  With CajaComún
    .DialogTitle = "Asociar fotografía"
    .Filter = "Mapas de bits (*.bmp)|*.bmp|Metaarchivos (*.wmf)|*.wmf"
    .Flags = FileMustExist
    .ShowOpen
    Fotografía = LoadPicture(.FileName)
  End With
End Sub
```

- Graba el programa como **bd05.frm / bd05.vbp** y pruébalo

INFORMES con el Control Data Report

- Crea un nuevo proyecto con los controles profesionales de Visual Basic.

- Menú Proyecto
  Agregar Data Environment

- Menú Contextual de “Connection1”
  Propiedades
    Microsoft Jet 4.0 OLE DB Provider
    [Siguiente]
    Clic en el botón [Examinar...]
      Selecciona la base de datos **Clientes**
    [Aceptar]

- Menú Contextual de “Connection1”
  Cambiar nombre
  Escribe: **Clientes** [Return]

- Menú Contextual de “Connection1” (Clientes)
  Agrega comando
  Menú Contextual de “Command1”
  Propiedades
  Nombre de comando= Socios
  Conexión= Clientes
  Objeto de base de datos = Tabla
  Nombre de Objeto= Socios
  [Aplicar]
  [Aceptar]

- Propiedades:

Proyect1	(Nombre)	InfSocios
DataEnvironment1	(Nombre)	Socios
Form1	(Nombre)	FormSocios

- Menú Proyecto
  Agregar Data Report

- Propiedades:

DataReport1	(Nombre)	Informe
	Caption	Informe para la tabla Socios
	DataSource	Socios
	DataMember	Socios

- [+] Socios
  Aparece el listado de campos

- Graba la aplicación como:
  Informe.Dsr
  Socios.Dsr
  Listado.frm
  InfSocios-vbp

- “Arrastra” **Nombre** al apartado **Detalle** y coloca:
  Encabezado
  Nombre:
  Detalle
  Nombre(Socios)

- Haz lo mismo con Apellidos, Dirección y teléfono.

- Clic en el **Encabezado** y clic en el control **RptLabel**, para colocar una etiqueta de propiedades:

Etiqueta5	Caption	Listado de la tabla Socios
	Font	Arial, con tamaño 33

- Menú Contextual de la etiqueta anterior
  Centrar en la sección Horizontalmente

- Clic en el **Pie de Informe** y inserta una etiqueta **RptLabel** de Caption= Número de Página

- Menú Contextual del pie de informe
  Insertar control
  Número actual de página

- Menú Contextual del pie de informe
    Insertar control
        Fecha actual (formato largo)

- inserta un CommandButton en el Form de Propiedades:

Form1	Caption	Listado de la tabla Socios
Command1	(Nombre)	VerInforme
	Caption	Pulsa para ver el informe

- Código

```
Private Sub VerInforme_Click()
    Informe.Show
End Sub
```

- Guarda y prueba la aplicación.

INFORMES con CRYSTAL REPORTS

Debes tener instalada la aplicación **Crystal Reports** que tienes en
...\Common\Tools\Vb\CrysRept\Crystl32.exe
Una vez instalada la aplicación, aparecerá en el Menú Complementos.

- Menú Complementos
    Diseñador de informes …

- Menú File
    New
        Clic en el asistente **Standard**

    Clic en **DataFile**: Clientes.mdb
        [Add]
        [Done]
    [Next]
    Clic en **All** para insertar todos los campos
    [Next]
    Apellidos [Add]
    [Next]
    [Next]
    [Next]
    Estilo **Standard** y clic en [Preview Report]

- [Design]
    Podemos cambiar el diseño del informe

- Menú File
    Save
        LisAge en *TuCarpeta*

- Menú File
    Exit para salir del **Crystal Reports**

    En el nuevo proyecto con controles profesionales de Visual Basic …

- Menú Proyecto
  Componentes
  Crystal Report Control 4.6
  [Aceptar]
  Aparece el nuevo **Control Crystal Reports**

- Inserta en el form un **CommandButton y**

Form1	(Nombre)	Informe
	Caption	Informe
Command1	(Nombre)	Ver
	Caption	&Ver Informe

- Inserta un control **Crystal Reports** en el formulario

- Propiedades del control:
  (Name) Crystal
  ReportFilename […]
  Selecciona el informe **LisAge.rpt**
  [Aplicar]
  [Aceptar]

- Código de [Ver Informe]:

```
Private Sub Ver_Click()
    Crystal.Action=1
End Sub
```

- Graba la aplicación omo **CrystalReports** y pruébala.

Asistente para Distribuir Aplicaciones

- [Inicio]
  Programas
  Microsoft Visual Basic 6.0
  Herramientas de Microsoft Visual Basic 6.0
  Asistente para empaquetado y distribución

- [Examinar...]
  bd04.vbp

- [Empaquetar]
  Aparece un mensaje de error …
  Haz clic en [Compilar]
  Se crea el **Ejecutable de nuestra aplicación**
  También se podía hacer con **Menú Archivo – Generar...**

- Selecciona “Paquete de instalación estándar”
  [Siguiente]

- Carpeta de Paquete:
  c:\TuCarpeta\MiPrograma
  [Siguiente]
  [Sí] para crear la carpeta

- El programa detecta que nuestra aplicación utiliza DAO
  Selecciona: msad020.tlb [Aceptar]
  comdlg32.ocx [Aceptar]

- [Siguiente]
- Un único archivo.cab
  [Siguiente]
- Título de la instalación:
  Clientes – Control de Socios
  [Siguiente]
- [Siguiente]
- [Siguiente]
- [Siguiente]
- Nombre de secuencia de comando: **Aplicación Clientes**
  [Finalizar]
- [Guardar informe]
  socios.txt
- [Cerrar]

En *TuCarpeta* tienes la carpeta **MiPrograma** que contiene nuestro programa. Ahora solo basta probarlo en otro ordenador.