FUNDAMENTOS DE INFORMÁTICA

FUNDAMENTOS DE PROGRAMACIÓN VISUAL BASIC

Escuela Universitaria de Ingeniería Vitoria-Gasteiz
Ingeniaritzako Unibertsitate Eskola Vitoria-Gasteiz

Departamento LSI (Lenguajes y Sistemas Informáticos)
Escuela Universitaria de Ingeniería Vitoria-Gasteiz
Curso Académico 2.007-2.008

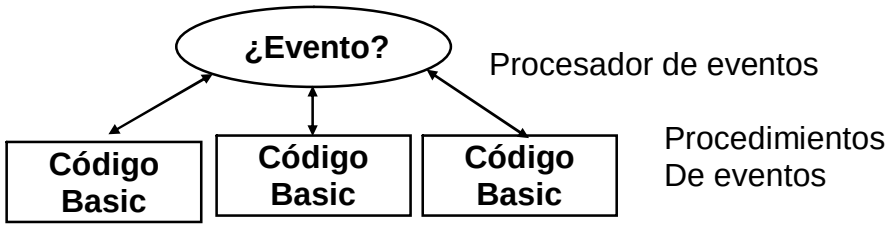
FUNDAMENTOS DE PROGRAMACIÓN EN VISUAL BASIC

1.- INTRODUCCIÓN. QUÉ ES VISUAL BASIC

Visual Basic es un **HERRAMIENTA SOFTWARE** que permite desarrollar aplicaciones Microsoft Windows. Son aplicaciones de interacción sencilla con el usuario.

Visual Basic está **ORIENTADO A EVENTOS/OBJETOS** (lo cual quiere decir que el código no se activa hasta que se llama como respuesta a un evento, por ejemplo Click de botón, Selección de un menú, …). NO SUCEDE NADA HASTA QUE SE DETECTA UN EVENTO.

Cuando se detecta un evento, el código correspondiente a dicho evento (procedimiento de evento) es ejecutado.



Algunas características de Visual Basic

- Conjunto de objetos (para "dibujar" la aplicación)
- Muchos iconos y dibujos
- Respuesta al ratón y al teclado
- Acceso a la impresora y al clipboard
- Una completa colección de funciones matemáticas, de cadena y gráficas
- Puede manejar variables fijas y dinámicas y arrays de controles
- Soporte de acceso a ficheros secuencial y random (de acceso aleatorio)
- Depurador muy útil y facilidades de manejo de errores
- Importantes herramientas de acceso a bases de datos
- Soporte ActiveX
- Auxiliar Package & Deployment Wizard para distribuir con facilidad las aplicaciones

Una breve Historia del Basic

Lenguaje desarrollado en los primeros 1960 en el Dartmouth College:

B (eginner's) (para principiantes)
A (All-Purpose) (de propósito general)
S (Symbolic) (simbólicas)
I (Instruction) (instrucciones)
C (Code) (código)

Surge como respuesta a los lenguajes de programación más complicados (FORTRAN, Algol, Cobol ...).

A mediados de 1970, dos estudiantes de instituto escribieron el primer Basic para un microcordenador (Altair) - costaba 350 dólares en una cinta de casete. Probablemente te suenen sus nombres: **Bill Gates y Paul Allen**.

Todos los Basic posteriores se han basado esencialmente en aquella versión inicial. Por ejemplo: GW-Basic, QBasic, QuickBasic.

Visual Basic apareció en 1991. Los primeros Visual Basic para DOS y Visual Basic para Windows fueron introducidos en 1991.

Visual Basic 3.0 (una gran mejora respecto a las versiones anteriores) salió en1993.

Visual Basic 4.0 salió a finales de 1995 (con soporte para aplicaciones de 32 bit).

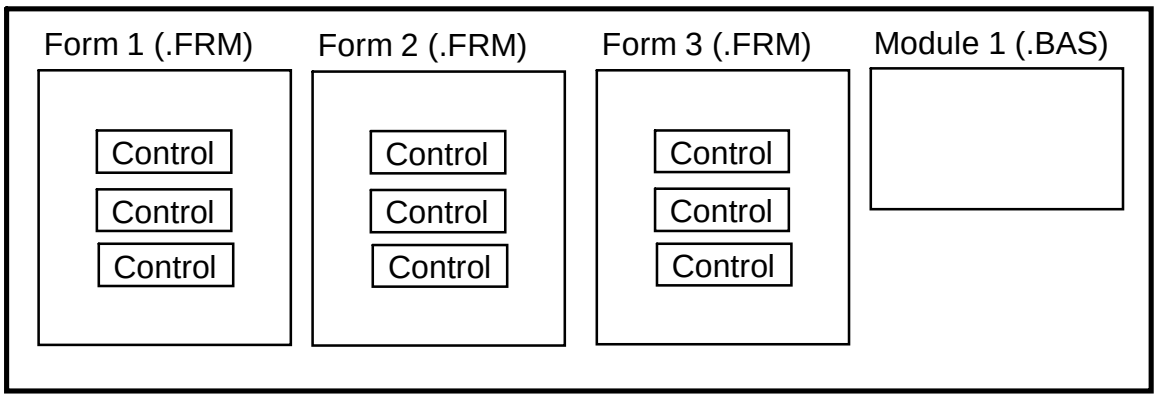
Visual Basic 5.0 salió a finales de 1996. Nuevo entorno, creación de controles ActiveX, anulado el soporte de 16 bit.

Y, **a finales de 1998 Visual Basic 6.0** , con algunas de las nuevas capacidades:

- Compilador más rápido
- Nuevo objeto de control de datos ActiveX
- Permite integración de la base de datos con un gran número de aplicaciones
- Nuevo creador de informes
- Nuevo auxiliar New Package & Deployment
- Más capacidades para Internet

2.- ESTRUCTURA DE UNA APLICACIÓN EN VISUAL BASIC

Proyecto (.VBP, .MAK)



La aplicación o proyecto está formado por:

- **Formularios** – Ventanas que creamos para relacionarnos con el usuario
- **Controles** – Gráficos colocados en los formularios para permitir la interacción del usuario (text boxes, labels, scroll bars, command buttons, etc.) (los Formularios y los Controles son **objetos**.)
- **Propiedades** – Cada característica de un formulario o de un control se especifica por medio de una propiedad. Por ejemplo name, caption, size, color, position, caption... Existen propiedades por defecto. Se pueden modificar las propiedades en tiempo de diseño o en ejecución.
- **Métodos** – Procedimiento ya creado que puede ser llamado para ejecutar una acción sobre un objeto.

  Métodos de los Objetos

  Ya hemos visto cómo cada objeto tiene asociados unas propiedades y unos eventos. Un tercer concepto que también se asocia a los objetos son los **métodos**. Un método es un procedimiento o función (un programa) que ejecuta alguna acción sobre el objeto.

  El formato para llamar a un método es el siguiente:

  NombreDelObjeto.Método {argumentos opcionales }

  Observemos que aquí también se utiliza el punto.

- **Módulos** – Serie de procedimientos genéricos, declaraciones de variables y definiciones de constantes para toda la aplicación.

Terminología básica de Programación Orientada a Objetos

Las Aplicaciones Manejadas por Eventos

En las aplicaciones manejadas por eventos, la ejecución no sigue una ruta predefinida. En vez de esto, se ejecutan diferentes secciones de código en respuesta a eventos. Los eventos se desencadenan por acciones del usuario, por mensajes del sistema o de otras aplicaciones. La secuencia de eventos determina la secuencia en que el código se ejecuta. Es por esto que la ruta que sigue el código de la aplicación es diferente cada vez que se ejecuta el programa.
Una parte esencial de la programación manejada por eventos es el escribir código que responda a los posibles eventos que pueden ocurrir en una aplicación. Visual Basic facilita la implementación del modelo de programación manejada por eventos.

¿Qué es un objeto?

Cada formulario (ventana), menú o control que se crea con Visual Basic es un módulo autocontenido llamado **objeto.** Los bloques básicos de construcción de una aplicación con Visual Basic son los objetos. Cada objeto tiene un conjunto de características y un comportamiento definido (**propiedades, métodos y eventos)** que lo diferencian de otros tipos de objeto. En otras palabras, un objeto formulario ha sido diseñado para cumplir determinada función en una aplicación, y no es lo mismo que un objeto menú.

Propiedades

El conjunto de datos que describen las características de un objeto se le conoce como sus **propiedades.** Para un formulario tenemos por ejemplo, las propiedades **BackColor** (color de fondo), **Height** (altura).
Algunas propiedades no solo determinan el aspecto que tiene el objeto, sino que además pueden determinar su comportamiento; por ejemplo, la propiedad **MaxButton** establece si el formulario tendrá o no el botón *Maximizar*. La presencia o ausencia de este botón determinará si el formulario se puede o no maximizar.

Métodos

Los métodos son un conjunto de procedimientos que permiten que un objeto ejecute una acción o tarea sobre sí mismo. Por ejemplo, para un formulario tenemos el método **Hide** que hará que el formulario se oculte; o el método **Show** que hará que el formulario se vuelva a mostrar.

Eventos

Un **evento** es una acción que es reconocida por el objeto. Un evento ocurre (se dispara) como resultado de la interacción del usuario con el objeto. También puede dispararse debido a la

ejecución de código (sentencias) o como resultado de la interacción de otro objeto con el objeto de poseedor del evento. Para un formulario tenemos por ejemplo; el evento **Load** que se dispara cuando se carga el formulario; o el evento **Click** para un botón de comando, se dispara cuando se hace clic sobre él.

*¿Qué papel cumplen las **propiedades, métodos y eventos**?*

Toda aplicación necesita una interfaz de usuario, la parte visual a través de la cual el usuario interactúa con la aplicación. Los bloques básicos de construcción de una interfaz de usuario son los formularios y los controles. Visual Basic utiliza técnicas de **programación visual** para diseñar las aplicaciones.

3.- PASOS PARA EL DESARROLLO DE UNA APLICACIÓN

Son 3:

1. *Dibujar la interfaz de usuario*
2. Asignar propiedades a los controles
3. Escribir códigos para los controles

4.- PREPARANDO LA INTERFAZ DE USUARIO

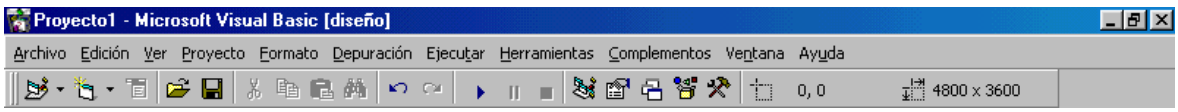
Visual Basic trabaja en 3 modos diferentes:

1. Modo DISEÑO
2. Modo EJECUCIÓN
3. Modo INTERRUPCIÓN (la aplicación se detiene para que podamos depurarlo)

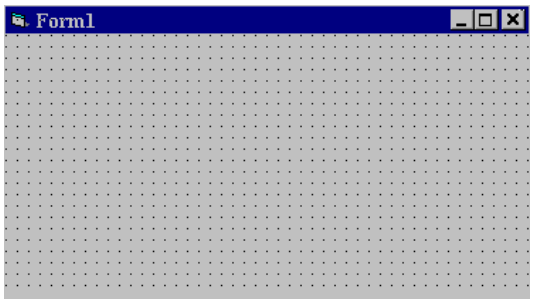
Veamos el modo **Diseño:**

Al arrancar VB aparecen **seis ventanas:**

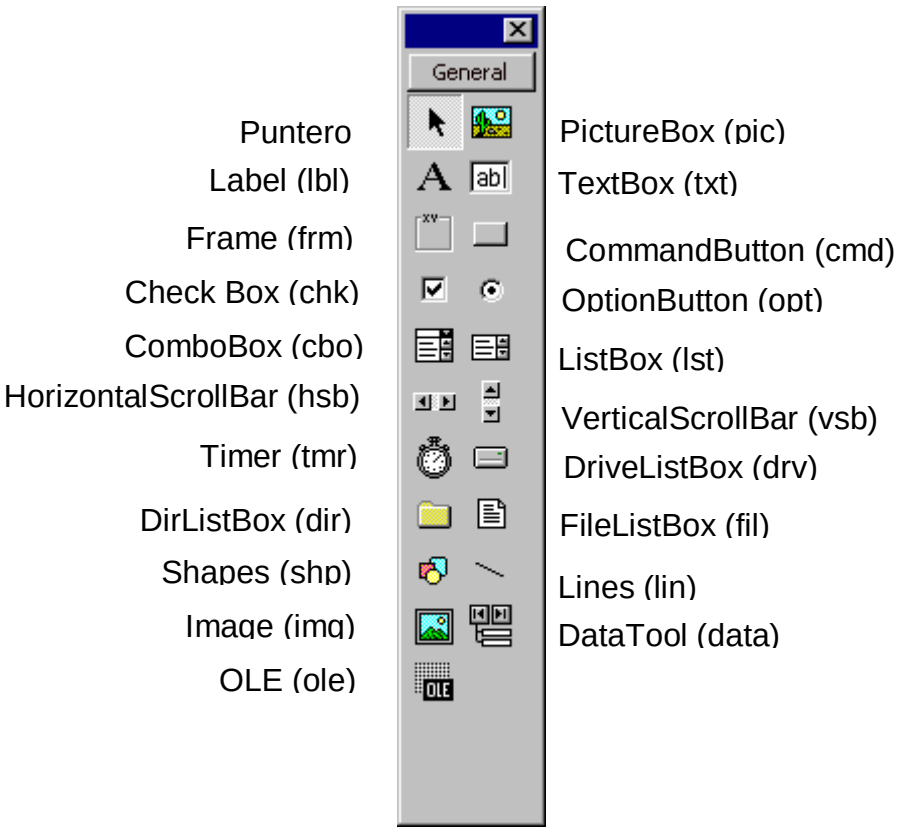
- La **Ventana Principal** consiste en la barra de título, la barra de menús y la barra de herramientas. La barra de título indica el nombre del proyecto, el modo actual de operación de VB y el formulario actual. La barra de menú contiene menús desde los cuales se controlan las operaciones del entorno de VB. La barra de herramientas tiene botones para activar algunas opciones del menú. La ventana principal también muestra la localización del formulario activo en relación con la esquina superior izquierda de la pantalla (medido en twips) y la



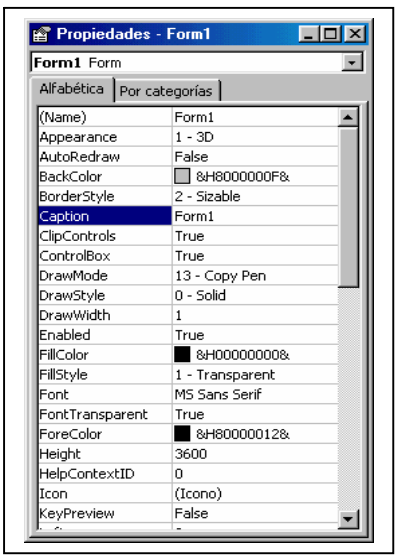
- La **Ventana** del **Formulario** es fundamental para desarrollar las aplicaciones de VB. Es donde se dibuja la aplicación



- La **Caja de Herramientas** permite seleccionar los controles utilizados en la aplicación.

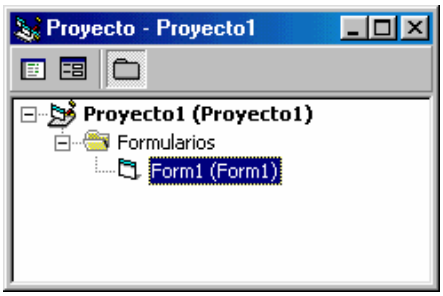


- La **Ventana** de **Propiedades** se utiliza para establecer los valores iniciales de las propiedades de los objetos. La caja que aparece en lo alto de la ventana contiene todos los objetos del formulario activo. Se puede ver de dos maneras: en orden Alfabético y por Categorías. Dentro de esta ventana nos encontramos con las propiedades que podemos utilizar, en tiempo de diseño, del objeto seleccionado.



- La **Ventana Posición del Formulario** muestra donde se verá el formulario dentro de la pantalla, en tiempo de ejecución:

- La **Ventana** del **Explorador de Proyectos** muestra una lista de todos los formularios y módulos que componen la aplicación. Desde esta ventana también se puede elegir entre ver el **Código** o el **Formulario.**



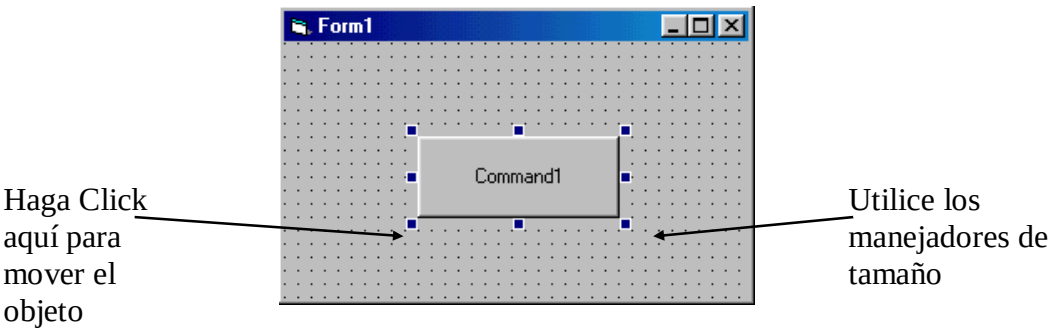
Como ya hemos comentado, la interfaz de usuario se "dibuja" en la ventana del formulario. Existen dos maneras de colocar controles en un formulario:

1. Doble-click en la herramienta elegida dentro de la caja de herramientas: el objeto se crea con un tamaño fijo en la form. Posteriormente se puede mover y modificar el tamaño.

2. Click en la herramienta elegida dentro de la caja de herramientas, a continuación mover el puntero al formulario. El puntero se convierte en una cruz. Situar la cruz en la esquina superior izquierda de donde se quiera dibujar, presionar el botón izquierdo y mantenerlo presionado mientras se dibuja un rectángulo hasta el borde inferior derecho. Soltar el ratón y el objeto queda en el sitio.

Para **mover** un control ya dibujado, hacer click en el objeto y arrastrarlo.

Para **cambiar de tamaño** un control, hacer click en el objeto para que aparezcan los punteros manejadores de tamaño. Utilizarlos.



5.- QUÉ ES UN PROGRAMA

Un programa es una colección de instrucciones. En Visual Basic las instrucciones se integran en PROCEDIMIENTOS (aquellos que empiezan son *Sub nombreProcedimento* y terminan con *End Sub*).

6.- SANGRÍAS EN LAS LÍNEAS DE COMANDOS

Para facilitar la modificación de los programas en caso de errores y conferir a los programas la mayor claridad posible.

7.- LÍNEAS DE COMENTARIO

Coma alta (‘) ó con la palabra reservada Rem. Aparecen en color verde. Sirven para aclarar el funcionamiento de un programa. Lo que viene a continuación NO ES INTERPRETADO por Visual Basic.

8.- CONTINUACIÓN DE LÍNEA

Si la sentencia es muy larga, se puede continuar en la siguiente línea utilizando el carácter de subrayado (_) precedido de un espacio en blanco. Ejemplo:

```
Meses = Log(Final * TipoInteres / Deposito + 1) _
/ Log(1 + TipoInteres)
```

9.- DECLARACIÓN DE VARIABLES

Un programa realiza determinadas tareas: procesa eventos, realiza cálculos, ordena grupos de datos … Para ello necesita disponer de una memoria intermedia donde poder guardar valores, cadenas de caracteres y otros elementos, así estarán disponibles cuando el programa deba recurrir a ellos. CON ESTE FIN SE UTILIZAN LAS VARIABLES.

Las variables pueden contener números, datos de fecha y hora, textos o imágenes.

Generalmente la vida útil de una variable es limitada, es decir, sólo conserva su valor dentro de un procedimiento (Private Sub …. End Sub).

9.1.- MODOS DE DECLARACIÓN DE VARIABLES

Existen 2 modos:

a) Declaración EXPLÍCITA: en este caso todos los módulos del código de un proyecto nuevo contienen la línea 'Option Explicit'. Las variables se declaran con **Dim nombreVariable**
b) Declaración IMPLÍCITA: con este modo no es necesario declarar variables al comienzo del programa mediante Dim nombreVariable. No se declaran las varibles al comienzo de un programa sino que se emplean directamente durante el programa.

9.2.-NOMBRES DE VARIABLES

Deben ser lo más cortos y significativos posibles.

- Deben comenzar por letras.
- Deben tener menos de 256 caracteres..
- No pueden incluir puntos ni caracteres especiales.

Si la variable tiene un nombre compuesto por varias palabras, escribiremos en mayúsculas la primera letra de cada una de ellas:

Ejemplo: TiempoActual

9.3. TIPOS DE DATOS: 10 TIPOS DE DATOS EN VB

Tipo de dato	Clase	Rango de valores
BYTE	Nº enteros	0..255
INTEGER	Nº enteros	-32768 .. + 32768
LONG	Nº enteros	-2147483658 .. +2147483658
SINGLE	Nº decimales	Nº con 8 dígitos en total
DOUBLE	Nº decimales	Nº con 16 dígitos en total
CURRENCY	Decimales Monedas	15 dígitos enteros + 4 decimales
BOOLEAN	Valores lógicos	True o False
DATE	Valores de Tiempo y Hora	Desde 1 de Enero del año 100 hasta el 31 de Diciembre del año 9.999. Hora desde 0:00:00 hasta 23:59:59
STRING	Texto	Cadenas de hasta 2.000 millones de caracteres
VARIANT	- (no tiene una clase definida)	Acepta todos los tipos

Según lo visto en esta tabla podríamos definir variables de la siguiente manera:

Dim miVariable la variable se declara como Variant

Dim cadena As String es de tipo String. Posteriormente se le puede asignar un valor (p.e. cadena =”martes”)

Dim numero As Byte Declaramos un número entre 0 y 255

Dim numero As Integer Declaramos un número entre (-32.768 y +32.768)

Dim miFecha As Date intervalo de tiempo 1.1.100 al 31.12.9999

Para asignar a este último valores:
miFecha=#9/23/03# (si es con formato mm/dd/aa)

Con otros formatos:
miFecha=@23 9 03#
miFecha=#23 September 03#
miFecha=#23 Sept 03#
miFecha=#23,9,03#

10.- SENTENCIAS EN VISUAL BASIC

La sentencia más sencilla es la de **asignación**. Consiste en el nombre de una variable, seguido del operador de asignación (=), seguido de algún tipo de **expresión**.

Ejemplos:

```
HoradeInicio = Now
Caja2.Caption = "Datos Fijos"
BitCont = ByteCont * 8
Energia = Masa * VELOCIDADDELALUZ ^ 2
ValorNeto = Ganancias - Deudas
```

La sentencia de asignación almacena información

11.- OPERADORES DE VISUAL BASIC

11.1 .- OPERADORES ARITMÉTICOS

OPERADOR	OPERACIÓN
+	Suma
-	Resta
*	Multiplicación
/	División con decimales
\	División sin decimales o División Enteros (29\5 = 5)
	Potencias y raíces
Mod	Operador del Resto de una División (29 Mod 5 =4)

PRIORIDAD EN LA EVALUACIÓN DE LOS OPERADORES ARITMÉTICOS

+

|

-

OPERADOR	OPERACIÓN
* y /	Multi y División con decimales
\	División Enteros
Mod	Resto de una División
+ y -	Suma y Resta

11.2 .- OPERADORES RELACIONALES (O DE COMPARACIÓN)

OPERADOR	OPERACIÓN
=	Igual a
<, >	Menor que, Mayor que
<=, >=	Menor o igual que, Mayor o igual que
<>	Distinto que

PRIORIDAD EN LA EVALUACIÓN DE LOS OPERADORES ARITMÉTICOS

Estos operadores son iguales entre sí, no tiene prioridades distintas.

11.3.- OPERADORES LÓGICOS

OPERADOR	OPERACIÓN
And	(intNumero>10) AND (intNumero<90)
Or	(intNumero<10) OR (intNumero>20)
Not	Not (intNumero<10)

PRIORIDAD EN LA EVALUACIÓN DE LOS OPERADORES LÓGICOS

+

OPERADOR
Not
And
Or

-

TABLA DE VERDAD DE LOS OPERADORES LÓGICOS

OPERACIÓN	RESULTADO
V and V	VERDADERO
V and F	FALSO
F and V	FALSO
F and F	FALSO
V or V	VERDADERO
V or F	VERDADERO
F or V	VERDADERO
F or F	FALSO
Not V	FALSO
Not F	VERDADERO

11.4.- OPERADORES DE CONCATENACIÓN

Hay 2 operadores de concatenación, uno de ellos es el '+' y el otro '&'.

a) Con el operador '+' se emplea tanto para concatenan entre sí expresiones (cadenas de caracteres o strings) como para sumar números (y en este caso no funciona concatenando sino calculando el resultado)

Ejemplo: supongamos que intNumero="45" y que IntIncremento="50"
intNumero+intIncremento="4550"
pero de la misma forma, supongamos que inNumero=45 y que IntIncremento=50
intNumero+intIncremento=95

b) El operador '&' une cadenas de caracteres o los concatena. Solo se emplea con cadenas de caracteres:
Ejemplo: supongamos que Nombre1="pepito" y que Nombre2="Martinez"
Nombre1&Nombre2="pepitoMartinez"

12.- ESTRUCTURA DE CONTROL EN VB

Las estructuras de control le permiten controlar el flujo de ejecución del programa.
Tenemos dos tipos de estructuras de control:

a) ESTRUCTURAS DE DECISIÓN
b) ESTRUCTURAS REPETITIVAS

A) ESTRUCTURAS DE DECISIÓN

Existen 3 estructuras distintas:

1. De una rama (*If…Then)*

If condicion **Then**
sentencia(s)
End If

Ejemplo1:

```
If cualquierFecha  <  Now  Then
        CualquierFecha  =  Now
        Timer1.Enabled  =  False  ‘ Desactiva el control Timer.
End If
```

Ejemplo2:

```
If  chkAlumnoUNI.Value=1  Then
        txtCosto  =  Format (txtCosto*0.70,”Fixed”)
        txtCódigo.Enabled  =  True
End  If
```

2. De 2 ramas (*If...Then...Else)*

```
If condicion Then
        sentencia(s)
Else
        sentencia(s)
End If
```

Ejemplo1:

```
If Numero < 10 Then
        Digitos = 1
Else
        Digitos = 3
End If
```

Ejemplo2:

```
If a < b Then ' Se permutan a y b
        temp = a
        a=b
Else
        temp=b
        B=a
End If
```

3. De ramas múltiples (*Select Case o If anidades con ElseIf)*

a) If anidadas:

```
If condicion1 Then
        sentencia1
Else
        If condicion2 Then
                sentencia2
        Else
                sentencia-n
        End If
End If
```

Ejemplo1:

```
If Val (txtPromedio)  >=13 Then
                txtCondición = “Aprobado”
Else
        If Val (txtPromedio)  >=  10 Then
                txtCondición  =  “Asistente”
        Else
                txtCondición  =  “Desaprobado”
        End If
End If
```

Ejemplo2:

```
If Ventas  >  100000  Then
           strDscto  =  Format (0.10, “Fixed”)
Else
        If Ventas > 50000 Then
              strDscto  =  Format (0.05, “Fixed”)
        Else
              strDscto  =  Format (0.02, “Fixed”)
        End If
End If
```

b) Select Case:

Selec Case *expresión*
[Case *lista_expresiones1*
acción 1
[Case *lista_expresiones2*
acción 2
.
.
.
[Case Else
acción n
End Select

Ejemplo1:

```
Select Case TipoUsuario
        Case  "Supervisor"
              ' Proporciona al usuario privilegios de Supervisor
              ...
              ...
        Case  "Usuario"
              ' Proporciona al usuario privilegios de Usuario
              ...
              ...
       Case Else
              ' Proporciona al usuario privilegio de invitado
             ...
             ...
    End Select
```

Ejemplo2:

```
Select Case Cantidad
    Case 1
          sngDscto = 0.0
    Case 2,  3
          sngDscto = 0.05
```

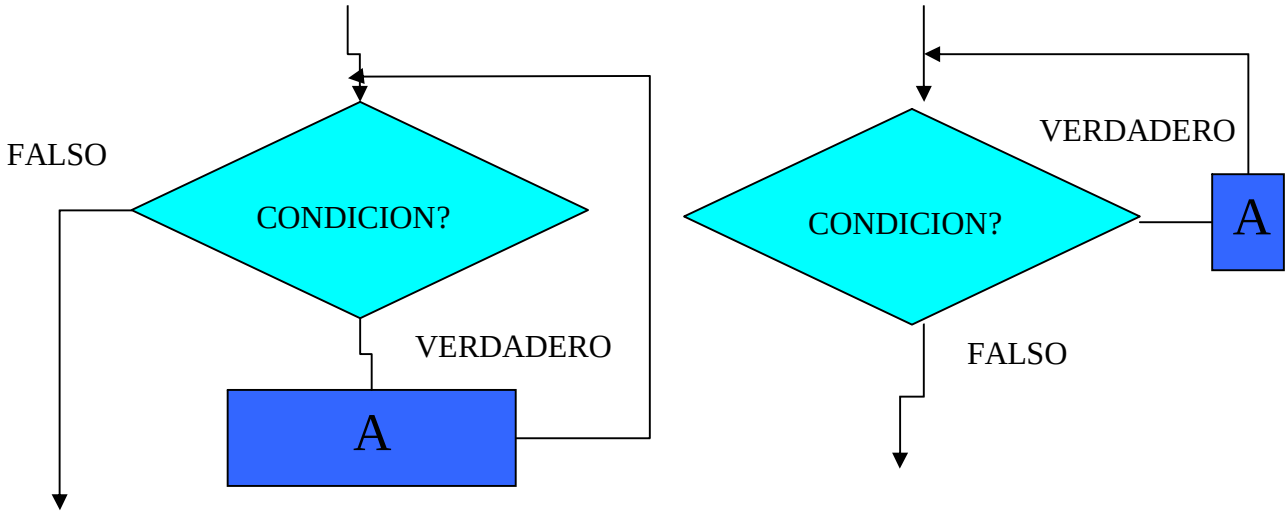
```
        Case 4 To 6
            sngDscto = 0.10
        Case Else
            sngDscto = 0.20
    End Select
```

B) ESTRUCTURAS DE REPETICIÓN

Existen 3 estructuras distintas:

1. <u>Repetitiva MIENTRAS (0-n veces)</u> While/Loop

Se representa de cualquiera de las 2 siguientes maneras:



Utilizar el bucle **Do** para ejecutar un bloque de sentencias un número indefinido de veces. Hay algunas variantes en la sentencia **Do...Loop,** pero cada una evalúa una condición numérica para determinar si continúa la ejecución. Como ocurre con **If...Then,** la ***condición*** debe ser un valor o una expresión que dé como resultado **False** (cero) o **True** (distinto de cero).
En el siguiente ejemplo de **Do...Loop,** las **sentencias** se ejecutan siempre y cuando ***condición*** sea **True:**

Do While condición

Sentencias

[exit Do]

Loop

Cuando Visual Basic ejecuta este bucle **Do,** primero evalúa ***condición.*** Si ***condición*** es **False** (cero), se salta todas las ***sentencias.*** Si es **True** (distinto de cero) Visual Basic ejecuta las ***sentencias***, vuelve a la instrucción **Do While** y prueba la condición de nuevo.
Por tanto, el bucle se puede ejecutar cualquier número de veces, siempre y cuando ***condición*** sea distinta de cero o **True.** Nunca se ejecutan las ***sentencias*** si ***condición*** es **False** inicialmente.

Ejemplo1:

```
Counter=1
Do While Counter<=1000
        Debug.Print Counter
        Counter=Counter+1
Loop
```

Si la cadena destino no está en la otra cadena, **InStr** devuelve 0 y no se ejecuta el bucle.

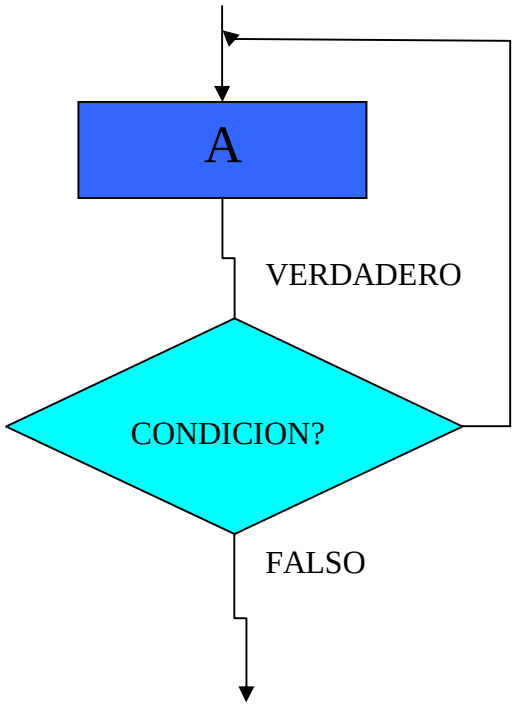
Una variable de este caso (y que hace exactamente lo mismo) es la estructura **Until/Loop:**

Ejemplo2:

```
Counter=1
Do Until Counter>1000
        Debug.Print Counter
        Counter=Counter+1
Loop
```

2. Repetitiva REPETIR (1-n veces) Do/Loop While

Se representa de la siguientes maneras:



Otra variante de la instrucción **Do...Loop** ejecuta las ***sentencias*** primero y prueba la ***condición*** después de cada ejecución. Esta variación garantiza al menos una ejecución de ***sentencias:***

Do

Sentencias

[exit Do]

Loop While *condición*

Ejemplo1:

```
Sum=1
Do
        Debug.Print Sum
        Sum=Sum+3
Loop While Sum<=50
```

Una variable de este caso (y que hace exactamente lo mismo) es la estructura **Do/Loop Until:**

Ejemplo2:

```
Sum=1
Do
        Debug.Print Sum
        Sum=Sum+3
Loop Until Sum>50
```

En cualquiera de los casos la sentencia exit Do nos permite salir de los bucles.

3. Repetitiva FOR..NEXT

Los bucles **Do** funcionan bien cuando no se sabe cuántas veces se necesitará ejecutar las ***sentencias*** del bucle. Sin embargo, cuando se sabe que se va a ejecutar las ***sentencias*** un número determinado de veces, es mejor elegir el bucle **For...Next.** A diferencia del bucle **Do,** el bucle **For** utiliza una variable llamada ***contador*** que incrementa o reduce su valor en cada repetición del bucle. La sintaxis es la siguiente:

__For__ contador = iniciar __To__ finalizar [__Step__ incremento]
Sentencias
__Next__ [contador]

Los argumentos **contador, iniciar, finalizar** e **incremento** son todos numéricos.

Ejemplo1:

```
Private Sub Form-Click ( )
    Dim I As Integer
    For i = 0 To Screen.FontCount
    Print Screen.Fonts (i)
    Next
```

End Sub

For Each...Next

El bucle **For Each...Next** es similar al bucle **For...Next,** pero repite un grupo de sentencia por cada elemento de una colección de objetos o de una matriz en vez de repetir las sentencias un número especificado de veces. Esto resulta especialmente útil si no se sabe cuántos elementos hay en la colección. He aquí la sintaxis del bucle **For Each...Next:**

For Each *elemento* ***In*** *grupo*
    Sentencias
Next *elemento*

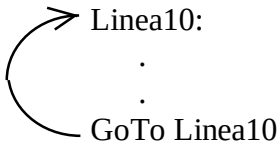
Por ejemplo, el siguiente procedimiento **Sub** abre la base de datos Biblio.mdb y agrega el nombre de cada tabla a un cuadro de lista.

```
Sub ListTableDefs ( )
    Dim objDb As Database
    Set objDb = OpenDatabase("c:/Archivos de
    programa/Devstudio/" & _
        "vb/biblio.mdb", True, False)
    For Each TableDef In objDb.TableDefs ( )
            List1.AddItem TableDef.Name
    Next TableDef
End Sub
```

12.1- SENTENCIA GOTO

Otra instrucción de bifurcación, y probablemente la instrucción más odiada en el ámbito de la programación, es la **GoTo**. La utilizaremos para capturar errores en tiempo de ejecución. El formato sería **GoTo *Label***, donde ***Label*** es un línea con una etiqueta. Las líneas con etiqueta se crean tecleando el *Label* seguido de dos puntos.

Ejemplo de **GoTo**:



Cuando el código llegue al GoTo, el control del programa pasa a la línea que tiene la etiqueta Linea10.

13.- FUNCIONES EN VISUAL BASIC

FUNCION: es una sentencia que realiza una determinada tarea (p.e., una petición de información al usuario) y después devuelve un resultado al programa. El valor devuelto por una función puede asignarse a una variable o a una propiedad o a otra sentencia o función. Normalmente las funciones utilizan argumentos para definir su actividad.

Visual Basic tiene una gran cantidad de **funciones** ya preparadas. La ayuda en línea te puede informar acerca de cualquiera de las funciones y de cómo se utilizan. Algunos ejemplos:

Función	**Valor que devuelve**
Abs	Valor absoluto de un número
Asc	Código ASCII o ANSI de un carácter
Chr	Carácter que corresponde a un código ASCII o ANSI
Cos	Coseno de un ángulo
Date	Fecha actual como cadena de texto
Format	Fecha o número convertido(s) en cadena de texto
Left	Selección izquierda de una cadena de texto
Len	Número de caracteres en una cadena de texto
Mid	Selección de una cadena de texto
Now	Hora y Fecha actuales
Right	Selección derecha de una cadena de texto
Rnd	Número aleatorio
Sin	Sene de un ángulo
Sqr	Raíz cuadrada de un número
Str	Número convertido en una cadena de texto
Time	Hora actual como una cadena de texto
Timer	Número de segundos transcurridos desde medianoche
Val	Valor numérico de una cadena de texto

Ampliación de la Función Rnd

- Escribiendo software para juegos y aprendizaje, utilizamos la función **Rnd** para crear sucesos aleatorios. Así conseguimos diferentes resultados cada vez que ejecutamos un programa. La función Rnd de VB nos devuelve un número aleatorio de precisión simple entre el 0 y el 1 (en realidad, mayor o igual a cero y menor que 1). Para crear números enteros (I) entre Imin e Imax, usaríamos la siguiente fórmula:

  I = Int((Imax - Imin + 1) * Rnd) + Imin

- Este número aleatorio generado por VB tiene que tener un punto de partida, una **semilla** que inicialice el generador. Para ello se utiliza la sentencia **Randomize** de la siguiente manera:

  Randomize semilla

  Si se usa la misma semilla cada vez que se ejecuta la aplicación, se generará la idéntica secuencia de números aleatorios. Para asegurarnos de que Rnd nos devolverá diferentes números cada vez que usemos el programa, conviene que la semilla sea la función **Timer**:

  Randomize Timer

  Esta sentencia debería ir en el procedimiento de evento **Form_Load**.

- **Ejemplos**: Para tirar un dado de seis caras (un nùmero entre 1 y 6):

  NúmeroTirada = Int(6 * Rnd) + 1

  Para conseguir un número entre 100 y 200:

  Número = Int(101 * Rnd) + 100

13.1- EJEMPLOS DE FUNCIONES: MSGBOX e INPUTBOX

13.1.1. MSGBOX

Una de las mejores funciones de VB es la caja de mensaje, **message box**. Esta caja presenta en pantalla un mensaje, un icono y unos botones de comando. El usuario responde haciendo click en un botón.

Cuando se usa como **sentencia**, no devuelve ningún valor (solamente se muestra la caja):

MsgBox Mensaje, Tipo, Título

donde

Mensaje Texto que se mostrará
Tipo Tipo del mensaje
Título Texto en la barra de título de la caja

No se puede controlar el lugar de aparición de la caja en la pantalla.

Cuando se usa como **función**, devuelve un valor entero (correspondiente al botón pulsado por el usuario). Ejemplo de uso (Respuesta es la variable que recoje el valor retornado):

```
Dim Respuesta as Integer
Respuesta = MsgBox(Mensaje, Tipo, Título)
```

El argumento **Tipo** se maneja sumando cuatro posibles componentes: los botones que queremos que se muestren, el icono, el botón por defecto y la modalidad de la caja de mensaje.

El primer componente de **Tipo** especifica los botonoes a visualizar:

Valor	Botones	Constante Simbólica
0	Aceptar	vbOKOnly
1	Aceptar /Cancelar	vbOKCancel
2	Anular/Reintentar/Ignorar	vbAbortRetryIgnore
3	Sí/No/Cancelar	vbYesNoCancel
4	Sí/No	vbYesNo
5	Reintentar/Cancelar	vbRetryCancel

El segundo componente de **Tipo** especifica el **icono** que se verá en la caja de mensaje:

Valor	Significado	Constante Simbólica
0	Sin icono	(None)
16	Icono Critical	vbCritical
32	Interrogación	vbQuestion
48	Exclamación	vbExclamation

64	Información	vbInformation

El tercer componente de **Tipo** especifica cuál es el botón por **defecto** (que se activará al pulsar Enter):

Valor	**Botón por defecto**	**Constante Simbólica**
0	El primero	vbDefaultButton1
256	El segundo	vbDefaultButton2
512	El tercero	vbDefaultButton3

El cuarto y último componente de **Tipo** especifica la **modalidad**:

Valor	**Significado**	**Constante Simbólica**
0	Modal a la Aplicación	vbApplicationModal
4096	Modal al Sistema	vbSystemModal

Si la caja es **Modal a la Aplicación**, el usuario tiene que responder a la caja antes de poder continuar con la aplicación en curso. Si la caja es **Modal al Sistema**, todas las aplicaciones esperan hasta que el usuario responda a la caja de mensaje.

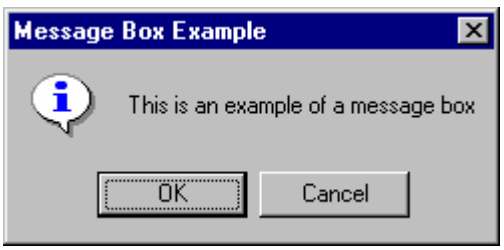
En cada opción de **Tipo** se pueden manejar igualmente los valores numéricos o las constantes simbólicas. Sin embargo es muy recomendable utilizar las constantes, por claridad en el código.

El valor que devuelve la caja de mensaje cuando se usa como función está relacionado con el botón pulsado:

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

Ejemplo de Message Box:

MsgBox “This is an example of a message box”, vbOKCancel + vbInformation, “Message Box Example”



En realidad estamos muy acostumbrados a ver cajas de mensaje al utilizar cualquier aplicación en Windows. Recuerda algún caso. Por ejemplo se suelen utilizar cajas de mensaje para preguntar si se quiere grabar antes de salir o para avisar de que la disketera no está lista.

13.1.2. INPUTBOX

Otra función de VB. Muestra en pantalla un cuadro de diálogo y pide al usuario que introduzca un valor.

Cuando se usa como **sentencia**, no devuelve ningún valor (solamente se muestra la caja):

InputBox Mensaje, Título

donde

Mensaje Texto que se mostrará
Título Texto en la barra de título de la caja

Ejemplo:

Nombre=InputBox “Introduzca su nombre”,”Mensaje al usuario”

14.- CONTROLES DE LA BARRA DE HERRAMIENTAS: PICTUREBOX E IMAGE

14.1.- PictureBoxes

Una **picture box** nos permite poner información gráfica en un formulario. Está preparada para entornos dinámicos, por ejemplo en animación.

Las picture boxes forman parte de la capa superior. Se comportan como si fueran pequeños formularios dentro de un formulario, ya que tienen casi las mismas propiedades que el formulario.

Propiedades de Picture Box:

AutoSize	Si está a True, la caja ajusta su tamaño al del gráfico.
Font	Tamaño, estilo… de la fuente.
Picture	Fichero gráfico que se visualizará en la picture box.

Eventos de Picture Box:

Click	Cuando se hace click sobre la caja.
DblClick	Cuando se hace doble-click.

Métodos de Picture Box:

Cls	Limpia la picture box.
Print	Visualiza información en la picture box.

Ejemplos

```
picEjemplo.Cls ' limpia la caja llamada picEjemplo
picExample.Print "una picture box" ' visualiza la cadena en la picture box
```

Procedimiento LoadPicture de Picture Box:

Una función muy importante cuando se utilizan las picture boxes es el procedimiento **LoadPicture**. Sirve para cargar la propiedad **Picture** en tiempo de ejecucin.

Ejemplo

```
picEjemplo.Picture = LoadPicture("c:\pix\ejem1.bmp")
```

Esta orden pone el fichero gráfico c:\pix\ejem1.bmp en la propiedad Picture de la picture box picEjemplo. El argumento de la función LoadPicture debe ser un camino y nombre de fichero válidos, porque si no e programa se cortará con un mensaje de error.

En una picture box se pueden cargar cinco tipos de ficheros gráficos:

Bitmap	Una imagen representada por pixels y almacenada como una colección de bits donde a cada pixel le correspondce un bit. Normalmente tiene la extensión**.bmp**. Aparece en su tamaño original.
Icon	Un tipo especial de fichero bitmap pero con un máximo de 32 x 32 pixels. Tiene extensión **.ico**. En la clase 5 haremos ficheros de iconos. Aparece en su tamaño original.
Metafile	Un fichero que almacena una imagen como una colección de objetos gráficos (líneas, círculos, polígonos) y no como pixels. Estos ficheros representan mejor la imagen original cuando se cambian de tamaño. Su extensión es **.wmf**. Su tamaño se adapta para acoplarse a la picture box.
JPEG	JPEG (Joint Photographic Experts Group) es un formato de bitmaps comprimidos que soporta color de 8 y de 24 bits. Se usa mucho en Internet. Su extensión es .**jpg**. Cambia de tamaño muy correctamente.
GIF	GIF (Graphic Interchange Format) es un formato de bitmaps comprimidos desarrollado por CompuServe. Soporta hasta 256 colors y también se usa mucho en Internet. Su extensión es .**gif** y también puede cambiar de tamaño sin perder mucho.

14.2.- Image Boxes

Una **image box** se parece mucho a una picture box porque también sirve para situar gráficos dentro de un formulario. Las image boxes se utilizan en casos estáticos, es decir, cuando no se va a modificar el gráfico presentado.

Las image boxes forman parte de la capa intermedia, por lo que pueden verse tapadas por picture boxes y otros objetos. Los gráficos de una image box pueden cambiar de tamaño mediante la propiedad **Stretch**.

Propiedades de Image Box:

Picture	El gráfico que se verá en la caja.
Stretch	Si es False, la image box cambia de tamaño para acoplarse al gráfico. En el caso contrario, es el gráfico el que se acopla al tamaño de la image box.

Eventos de Image Box:

Click	Cuando se hace click sobre la image box.
DblClick	Cuando se hace doble-click.

No tiene métodos, pero usa la función **LoadPicture** de la misma manera que picture box. Y puede cargar el mismo tipo de ficheros gráficos: ficheros de bitmap (.bmp), de iconos (.ico), metaficheros (.wmf), ficheros GIF (.gif), y ficheros JPEG (.jpg).

Diferenciar entre los controles PictureBox e Image

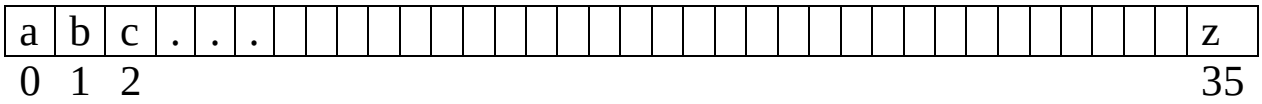
PictureBox	Image
Herramienta nº 2	Herramienta nº 19
Permite albergar: Mapas de bits (.bmp) Iconos (.ico) Archivos Fotos .jpg o .gif	Permite albergar: Mapas de bits (.bmp) Iconos (.ico) Archivos Fotos .jpg o .gif
También puede presentar texto y actuar como contenedor visual para otros controles	No actúa como un contenedor
	Puede actuar como un Botón de comando (se puede hacer Click en ella y conseguir que se ejecute un evento)
Para cambiar su tamaño se modifica la propiedad **.Autosize** a **True**	Para cambiar su tamaño de modifica la propiedad **.Strech** a **True**
Al modificar el Autosize la imagen NO cambia de su tamaño para ajustarla al tamaño del control	**Al modificar la Strech la imagen adquiere el tamaño del control en el que está contenida, es decir, se ajusta. SE VE TODA LA IMAGEN (por eso se suele emplear más este control que el anterior)**

15.- MATRICES O ARRAYS

Una matriz o array es una estructura de datos que se compone de varias posiciones contiguas, a las cuales se hace referencia mediante un índice. Ese índice comienza con 0 (0 es la primera posición de la matriz).

Cuando crea una matriz Visual Basic reserva espacio de memoria RAM para la misma, y dicha matriz permanecerá "cargada" en dicha memoria hasta el final de la ejecución del programa.

Matriz QUIMICA



Para referenciar a cada uno de los elementos de la matriz haremos:

QUIMICA(0)
QUIMICA(1)
…
QUIMICA(35)

En Visual Basic la forma de crear esta matriz sería la siguiente:

```
Dim Quimica(35) As String
```

Cada una de las posiciones de una matriz puede contener cualquier tipo de dato de los que hemos visto (integer,Byte,String …)

Vamos a estudiar 2 tipos de matrices:

a) Estáticas (que a su vez se dividen en Unidimensionales y Multidimensionales)
b) Dinámicas

a) Estáticas – Unidimensionales

Ejemplo:
```
Dim contadores(14) As integer
Dim Suma(20) as Double
```

Estáticas – Multidimensionales

Ejemplo:
```
Dim matriz(3,3) As double
```

Se crea una estructura de 2 dimensiones.

b) Dinámicas.

Ejemplo: Dim matrizDinamica() As Integer

15.2.- MATRICES DE CONTROLES

Son aquellas que se crean desde la vista Diseño de Visual Basic, insertando uno de los controles de la Barra de Controles y copiando ese mismo control sucesivas veces sobre el formulario, apareciendo una estructura en la Barra de Propiedades que nos indica que tenemos una matriz de controles.

En el código, la manera de averiguar si tenemos una matriz de controles o no es mirando si tenemos como argumento de un procedimiento Private Sub algo parecido a **Index As ….** Si es así indudablemente tendremos una matriz de controles en nuestra aplicación.

Una vez creado y nombrado un array de controles, hay que referirse a los elementos del array por su nombre y por su índice. Por ejemplo para indicar la propiedad **Caption** del elemento **6** de un array de etiquetas (label box) llamado **lblEjemplo**, haríamos:

lblEjemplo(6).Caption = “Esto es un ejemplo”

16.- OTROS CONTROLES DE VISUAL BASIC

16.1.- Common Dialog

Para que el usuario tenga un interfaz estándar en las operaciones más típicas de Windows, Visual Basic tiene un grupo de cajas de diálogo habituales (**common dialog boxes).**

El control Common Dialog no aparece por defecto en la configuración inicial de Visual Basic, tenemos que añadirlo nosotros. Esto se hace por las opciones de menú **Proyecto** – **Componentes** y eligiendo **Microsoft Common Dialog Control**.

La herramienta common dialog, aunque aparezca en el formulario como un control de tamaño fijo, no se ve en tiempo de ejecución, y no se puede controlar dónde aparecerá dentro de la pantalla. Para que se active es necesario llamar en tiempo de ejecución a uno de los cinco métodos ‘**Show’** que existen. Estos métodos son:

Metodo	Common Dialog Box
ShowOpen	Abrir Fichero
ShowSave	Grabar Fichero
ShowColor	Elegir Color
ShowFont	Elegir tipo de Letra
ShowPrinter	Imprimir

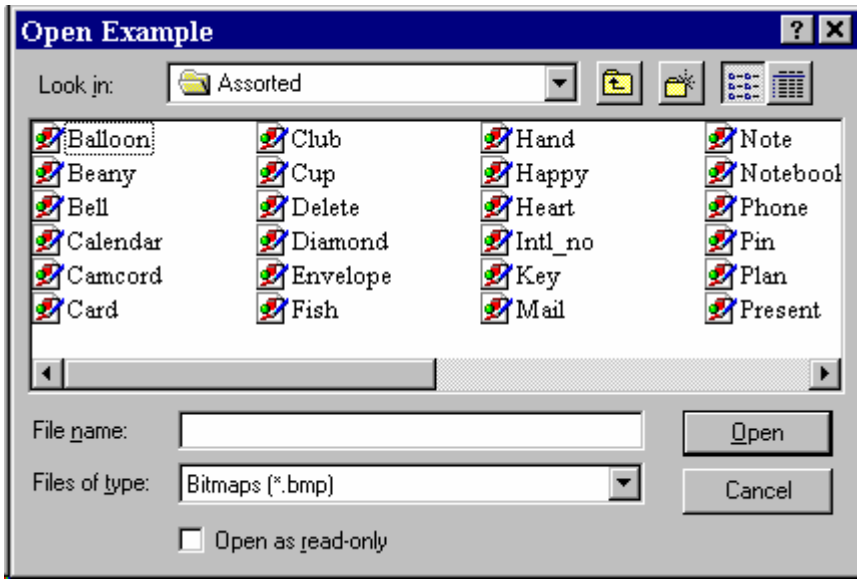
Para activar un common dialog box de nombre **cdlEjemplo** con la utilidad de Abrir un Fichero, haríamos:

cdlEjemplo.ShowOpen

Después de cerrar la caja de diálogo, el control vuelve a la línea siguiente a esta. Las common dialog boxes son modales al sistema.

Ejemplo de utilización de la CommonDialog con el método OPEN

La caja de diálogo **Open** permite que el usuario indique el nombre del fichero que quiere abrir. En la clase 6 veremos cómo abrir un fichero. Esta caja se presenta mediante el método **ShowOpen**. Veamos un ejemplo:



17.- TRABAJANDO CON FORMULARIOS

El **Formulario** es la zona donde se diseña la interfaz de usuario. Es el centro del desarrollo de las aplicaciones VB.

Eventos del Formulario:

Activate	Cuando el formulario se convierte en la ventana activa.
Click	Cuando se hace click sobre él.
DblClick	Cuando se hace doble click sobre él.
Load	Cuando se carga el formulario. Este es el lugar ideal para inicializar las variables y las propiedades.

Métodos del Formulario:

Cls	Limpia los gráficos y el texto que huebiera en el formulario. No limpia los objetos.
Print	Visualiza cadenas de texto en el formulario.

Ejemplos

```
frmExample.Cls ' limpia el formulario
frmExample.Print "Esto se verá en el formulario "
```

17.1.- CARGA Y DESCARGA DE FORMULARIOS

La sintaxis es:

Load NombreFormulario (carga)
UnLoad NombreFormulario (descarga)

17.2.- VISUALIZACIÓN / NO VISUALIZACIÓN DE FORMULARIOS

NombreFormulario.Hide (no lo muestra)
NombreFormulario.Show (lo muestra)

18.- MÓDULOS EN VISUAL BASIC

Son ficheros independientes con la extensión .bas, que contienen trozos de código en los cuales normalmente se declaran variables globales del sistema.