

Visual Basic para Aplicaciones del Access 2007

TuCarpeta significa una carpeta de tu ordenador donde es conveniente que grabes todos los ejercicios de este manual.

1

Fundamentos de VBA

a) Ejecuta el **Access 2007**:

- En la pantalla inicial:

Más en Office Online [Descargas]

Buscar: **extensiones para programadores Access 2007**

Descargamos y/o ejecutamos el fichero: **AccessDeveloperExtensions.exe**

A partir de este momento en el **Menú del Office** tendremos una nueva opción:

Programador

- Botón del Office

Opciones de Access

Centro de confianza

[Configuración del Centro de confianza]

Configuración de Macros

Habilitar todas las macros (no recomendado ...)

[Aceptar]

[Aceptar]

- En la pantalla inicial, clic en la opción: **Base de datos en blanco**

- Sitúate en *TuCarpeta*, es decir en el campo **“Guardar en:”**, debe aparecer *“TuCarpeta”*

- En el campo **“Nombre de archivo”**, escribe: **PROGRAMAS**

y CLIC en [Aceptar]

[Crear]

Acabamos de crear en “nuestra carpeta” una **base de datos** de nombre **PROGRAMAS**

- Haz lo siguiente:

Cinta de Opciones: Herramientas de base de datos

Grupo: Macro

Visual Basic

Acabamos de acceder al **Editor de Visual Basic**

b) Vamos a escribir nuestro primer programa en **“Visual Basic”**.

En la ventana **“PROGRAMAS-Módulo1(Código)”** y debajo de la línea:

Option Compare Database

Escribe el siguiente programa:

```

Sub Programa1()
  MSGBOX "Hola Mundo"
End Sub

```

Habrás observado varias cosas:

- Las "sentencias" **Sub** y **End Sub** aparecen de color azul
- Aunque escribas **MSGBOX** (todo en mayúsculas), aparece **MsgBox**

Las sentencias **Sub**, **End Sub** y **MsgBox** son "palabras reservadas". Dicho de otra forma: forman parte del lenguaje "Visual Basic"

Acabamos de escribir nuestro primer "**procedimiento**", ya veremos que hay diferentes tipos de **programas**. De momento tenemos un programa **PROCEDIMIENTO**.

Observa la *estructura de un procedimiento*:

```

Sub nombrePrograma()
  -----
  -----
  -----
End Sub

```

c) Vamos a "ejecutar" el procedimiento "**Programa1**"...

Haz lo siguiente:

- Menú Ver
Ventana Inmediato
- Escribe:
Programa1 y pulsa [Return]

Si todo va bien, aparece una ventana de nombre "**Microsoft Access**" con el mensaje: **Hola Mundo** y el botón [Aceptar]

- Haz CLIC en el botón [Aceptar]
- "Cierra" la ventana "**Inmediato**" (es decir: CLIC en la **X** del vértice superior derecho de la ventana "**Inmediato**").
- "Cierra" la ventana **Módulo1**. Es decir, CLIC en la **X** del extremo superior derecho de la ventana "**Programas-Módulo1(Código)**".
- CLIC en el icono "**Ver Microsoft Access**":



- "Cierra" el "Access". Es decir, CLIC en la **X** del extremo superior derecho de la ventana "**Microsoft Access**".

A la pregunta: ¿Desea guardar los cambios en el diseño de módulo "Módulo1"?, haz CLIC en el botón [Sí]. Cómo el nombre **Módulo1** ya nos va bien, haz CLIC en [Aceptar]

Aunque la mayor parte del desarrollo de una aplicación en **VBA** se realiza de forma “visual”: controles en formularios y estableciendo propiedades, también es cierto que una parte muy importante de todo programa, es el “código” que se encargará de responder a los **eventos** (situaciones), que se producirán en la aplicación.

En este ejercicio se trata de estudiar el lenguaje de programación que necesitamos para **escribir el código**.

d) Ejecuta el **Access**:

- **Más ...**
- Sitúate en “**TuCarpeta**”. Es decir, en el campo “**Buscar en:**” debe aparecer *TuCarpeta*.
- CLIC en **PROGRAMAS**, para seleccionar el fichero.
- CLIC en [Abrir]
- Como sólo tenemos un módulo (Módulo1) ya está seleccionado, haz un doble clic en **Módulo1**
- Observa la “**Barra de Tareas del Windows**” (última línea de la pantalla): Tenemos activado el “**Microsoft Visual Basic**” y al lado tenemos el “**Microsoft Access**” desactivado. Es decir, por el sólo hecho de acceder a un **módulo**, automáticamente nos situamos en el “**Editor de VB**”
- Haz CLIC en “**Microsoft Access**” de la “barra de tareas”: está claro lo que sucede ¿no?. Volvemos al **Access**. Vuelve a hacer CLIC, pero ahora en “**Microsoft Visual Basic**” de la barra de tareas y volveremos al “**Editor de VB**”
- Sitúa el cursor de escritura al final de la ventana, después de la línea **End Sub** del procedimiento **Programa1**.
- Escribe lo siguiente:

```
Sub Programa2()  
    MsgBox "Esto es el primer mensaje"  
    'Esto es un comentario, porque al principio _  
      de la línea he escrito un apóstrofe  
    MsgBox "Esto es el segundo mensaje"  
    'Esto es otro comentario que ocupa una línea  
    MsgBox "Esto es el tercer mensaje"  
End Sub
```

- Antes de ejecutar el programa, asegúrate de que está bien escrito, concretamente:
 - Para introducir un “comentario” en el código, basta comenzar la línea con el “apóstrofe” (tecla del interrogante ?). El comentario aparece automáticamente en color verde.
 - Podemos escribir líneas de programa distribuyéndolas en varias líneas, sin más que escribir el **símbolo de subrayado** (tecla del “menos”) **precedido de un espacio**.
- Graba lo que hemos hecho, es decir:
 - Menú Archivo
 - Guardar PROGRAMAS
 - o
 - CLIC en el icono “**Guardar**”

- Ejecuta el programa, es decir:
Menú Ver
Ventana Inmediato

Escribe:

Programa2 y pulsa [Return]

Espero que te funcione. En el siguiente apartado haremos un programa con algún error, para observar cómo nos avisa el **Access**.

- “Cierra” la ventana “**Inmediato**”, es decir CLIC en la **X** del extremo superior derecho de la ventana correspondiente.

e) Escribe el siguiente procedimiento:

```
Sub Programa3()  
  MSSGBOX "A ver que pasa"  
  ' Está claro que hemos escrito un error  
End Sub
```

- Ejecuta el programa anterior...

No es necesario que hagas “**Menú Ver – Ventana Inmediato**”, basta que pulses las teclas [CTRL][G]

Escribe: **Programa3** y [Return]

- Si todo funciona correctamente, el programa “protesta”. Tenemos siempre dos posibilidades:
 - [Ayuda]
 - [Aceptar]

La primera vez que ejecutamos un programa, es lógico pensar que nos hemos equivocado al escribir y por ésta razón es mejor hacer CLIC en [Aceptar] (si no sabemos de donde viene el error y al ejecutar el programa ya corregido, nos vuelve a decir lo mismo, es más lógico hacer CLIC en [Ayuda])

- Haz CLIC en [Aceptar]
- Observa que el **Access** nos señala la línea que no entiende...
Corrige el error, es decir en lugar de **MSSGBOX** escribe **MSGBOX**.
- Para continuar, haz:
Menú Ejecutar
Continuar
o si quieres ir más deprisa, pulsa la tecla **[F5]**
- Acaba de ejecutar el programa, es decir: CLIC en el botón [Aceptar] del mensaje “**A ver que pasa**”
- “Cierra” la ventana de “**Inmediato**”.
- Graba lo que hemos hecho hasta ahora (CLIC en el icono “**Guardar**”)

Recapitulemos lo que hemos hecho hasta este momento:

Estructura de un procedimiento:

```
Sub NombrePrograma()
    .....
    .....
    .....
End Sub
```

MsgBox “mensaje”

Aparece una ventana que contiene el “mensaje” y un botón [Aceptar]. Al hacer CLIC en el [Aceptar] anterior, se continúa la ejecución del programa.

Ya veremos más adelante que el “**MsgBox**” es otro tipo de programa, ya incorporado al **VBA**, llamado **función**.

Si queremos **añadir comentarios** a un programa, basta comenzar la línea de comentarios con un **apóstrofe**.

Si queremos que una “instrucción” ocupe **más de una línea**, basta “romper” la línea de programa con el **símbolo de subrayado precedido de un espacio**.

En los siguientes apartados nos iremos introduciendo poco a poco en el **VBA**...

- f) Con el **Módulo1** a la vista. Escribe el siguiente procedimiento:

```
Sub Programa4()
    Dim n1 As Integer, n2 As Integer
    n1 = InputBox("Escribe un número")
    n2 = InputBox("Escribe otro número")
    MsgBox "La Suma es = " & n1 + n2
End Sub
```

Antes de ejecutar el programa anterior observa:

- El **Programa 4** sirve para sumar dos números: el programa nos pedirá los dos números (InputBox) y nos dará (MsgBox) el resultado de sumarlos.
- Los dos números a sumar son las **variables** n1 y n2
- En un programa VBA es conveniente declarar previamente las **variables** que hemos de utilizar en el procedimiento.
- La forma de declarar las variables es:


```
Dim variable1 As Integer, variable2 As Integer
```

 A cada variable hemos de especificar su “tipo”, aunque sea el mismo.
- **Integer** quiere decir que el valor que tomarán las variables son números enteros entre – 32.768 y 32.767

- El símbolo **&** sirve para concatenar datos. En nuestro caso:
Aparecerá el mensaje **“La suma es =”** (porque está entre comillas) y a continuación (porque hay el símbolo **&**) el resultado de $n1+n2$ (porque no está entre comillas).

Veamos pues lo que hace el **Programa 4**:

- Definimos dos variables $n1$ y $n2$ tipo entero
- El programa nos pedirá un número (InputBox), una vez escrito el número, el programa lo “guardará” en la **variable $n1$** .
- El programa nos pedirá otro número (segundo InputBox), una vez escrito, el programa lo “asignará” a la **variable $n2$** .
- El programa nos mostrará (MsgBox) el mensaje **“La suma es =”** y a continuación el resultado de la suma de los dos números introducidos ($n1 + n2$).

Ejecuta el programa de la siguiente forma:

- Pulsa **[CTRL][G]**
- Escribe: **Programa4** y [Return]
- Al mensaje: **“Escribe un número”**. Introduce el número **527** y haz CLIC en [Aceptar] o pulsa la tecla [Return].
- Al mensaje **“Escribe otro número”**, escribe **100** y [Return]
- Si todo va bien, aparece un **“MsgBox”** con el mensaje: **“La suma es = 627”**
- Haz CLIC en [Aceptar]
- Si no te ha funcionado, debes observar detenidamente lo que has escrito y corregir los errores que has hecho.
- Vuelve a ejecutar el **Programa 4...**
 - Con la “Ventana Inmediato” (llamada también ventana de “depuración”), a la vista.
 - Sitúa el cursor de escritura *detrás* de la palabra **Programa4** y pulsa [Return]
 - A la primera “pregunta” escribe **-5799**
 - A la segunda “pregunta”, escribe un número que no sea “Integer”, por ejemplo **3,7**
 - Si todo va bien, aparece:
La suma es = -5795
 - Es decir, el programa “funciona” pero incorrectamente, ya que **-5799 + 3,7 = -5795,3**
- El **error** que hace el programa es el siguiente: al declarar las variables como números enteros, si introducimos un número no entero (por ejemplo **3,7**), lo transforma en número entero (en nuestro caso **4**).

g) Vamos a solucionar el problema del **Programa4...**

- Escribe el siguiente programa:

```

Sub Programa5()
  Dim n1 As Double, n2 As Double
  n1 = InputBox("Escribe un número")
  n2 = InputBox("Escribe otro número")
  MsgBox "La Suma es = " & n1 + n2
End Sub

```

Como el **Programa5** es muy parecido al **Programa4**, en lugar de escribir de nuevo el **Programa5**, sería más rápido:

- Selecciona el **Programa4**
 - CLIC en el icono **Copiar** o **Menú Edición – Copiar**
 - Sitúa el cursor al final del **Programa4**, en una línea nueva.
 - CLIC en el icono **Pegar**
 - Corrige el **4** de la copia por un **5**
 - Corrige los **Integer** de la copia por **Double**
- Ejecuta el **Programa5**, introduciendo los números:

-5,79
+2,61

Si todo va bien aparece **-3,18**

La instrucción:

```
Dim n1 As Double, n2 As Double
```

Significa que declaramos las variables n1 y n2 como números decimales.

- Acuérdate de grabar todo lo que vas haciendo (CLIC en el icono **Guardar**)

h) Vamos a hacer a partir de ahora “programas autoexplicativos”, es decir: entre las líneas de programa aparecen en comentarios (apóstrofe), las explicaciones. Por supuesto, si no quieres escribir los **comentarios** en tus programas, éstos funcionarán exactamente igual.

- Escribe en el **Módulo1** el siguiente procedimiento:

```

Sub Programa6()
  'Cálculo del área de un TRIÁNGULO
  Dim bas As Double
  Dim alt As Double
  Dim are As Double
  'Observa que defino el área como variable _
  a diferencia del programa anterior
  bas = InputBox("¿Cuál es la base del triángulo?")
  alt = InputBox("¿Cuál es la altura del triángulo?")
  are = bas * alt / 2
  'Observa como asigno el valor de la 3ª variable _
  a partir de las otras dos
  MsgBox "El área del triángulo es " & are
End Sub

```


- Ejecuta el **Programa6**
- Grábalo (CLIC en el icono **Guardar**)

i) Escribe en el **Módulo1** de la base de datos **PROGRAMAS** el siguiente procedimiento:

```
Sub Programa7()
    'Programa que nos pide nuestro nombre
    Dim nom As String
    'El tipo "String" significa texto
    nom = InputBox("Escribe tu nombre y apellidos")
    MsgBox "Hola " & nom
End Sub
```

- Ejecuta el programa
- Grábalo.

j) Escribe en el **Módulo1** el siguiente procedimiento:

```
Sub Programa8()
    'Estructura de programación If-Then-Else-End If
    Dim num1 As Double, num2 As Double
    'Defino dos variables tipo Double
    num1 = InputBox("Escribe el primer número")
    num2 = InputBox("Escribe el segundo número")
    'El programa nos solicita dos números
    'Atención con la estructura de programación _
    que aparece a continuación
    If num1 < num2 Then
        MsgBox "El primer número " & num1 & " es menor que " _
            & "el segundo " & num2
    Else
        MsgBox "El primer número " & num1 & " no es menor que " _
            & "el segundo " & num2
    End If
End Sub
```

- Prueba el programa y grábalo
- La estructura de programación: **If – Then – Else – End If** es la estructura de programación más sencilla, observa su funcionamiento:

```
If condición Then
    Instrucción1
    Instrucción2
Else
    Instrucción3
    Instrucción4
End If
```

Traducido al castellano diría:

Si se cumple la condición **entonces**
ejecuta las instrucciones 1 y 2
en caso contrario (es decir sino se cumple la condición)
ejecuta las instrucciones 3 y 4
Fin de la estructura.

k) Escribe en el **Módulo1** el siguiente procedimiento:

```
Sub Programa9()  
  Dim A As String  
  A = InputBox("¿Quieres continuar (S/N)?")  
  If A = "S" Or A = "s" Then  
    MsgBox "Pepe"  
  End If  
End Sub
```

- Prueba el programa y grábalo.
- Veamos el funcionamiento del **Programa9**:
 - En primer lugar definimos una variable tipo **String**, que guardará **S** o **N**
 - Si a la pregunta **¿Quieres continuar?**, contestamos **S**. En la variable **"a"** se guardará el valor **"S"**
 - Gracias a la estructura **If – Then – End If** (observa que no es necesaria la cláusula **Else**). El programa escribirá **Pepe** o no.

l) El programa anterior tiene un problema, en efecto:

- Ejecuta el **Programa9**
- A la pregunta **¿Quieres continuar?**, contesta: **Sí** y [Return]
- Observa que no funciona, es decir, no aparece la ventana con el mensaje "Pepe". Es lógico que así sea, ya que en la condición del **If – Then** tenemos: **a="S" Or a="s"**

Vamos a solucionar este problema:

- Escribe en el **Módulo1**, el siguiente programa:

```
Sub Programa10()  
  Dim A As String * 1  
  A = InputBox("¿Quieres continuar (S/N)?")  
  If A = "S" Or A = "s" Then  
    MsgBox "Pepe"  
  End If  
End Sub
```

- Ejecuta el programa, contestando **"Sí"** a la pregunta **¿Quieres continuar?**.

- La diferencia entre el programa 10 y el 9 está en que:
 - En el programa 9 definimos una variable de texto de longitud variable: **Dim a As String**
 - En el programa 10 definimos una variable de texto de longitud fija (exactamente de longitud = 1 carácter): **Dim a As String*1**
- Si necesitáramos una variable de texto de longitud 5, escribiríamos: **Dim a As String*5**
- Acuérdate de grabar el programa (CLIC en el icono **Guardar**).

m) Escribe en el **Módulo1** el siguiente procedimiento:

```
Sub Programa11()  
    Dim nom As String * 7  
    nom = InputBox("Escribe tu nombre y apellidos")  
    MsgBox "Hola " & nom  
End Sub
```

- Ejecuta el programa y si todo va bien observarás que “trunca” tu nombre+apellidos a 7 caracteres, debido a la definición de la variable **nom = String*7 = 7 caracteres**.

n) Vamos a estudiar otra estructura de programación, escribe en el **Módulo1** el siguiente procedimiento:

```
Sub Programa12()  
    Dim contador As Integer  
    contador = 1  
    Do While contador <= 5  
        MsgBox "Pepe"  
        contador = contador + 1  
    Loop  
End Sub
```

- Ejecuta el programa
- Si todo va bien debe aparecer 5 veces el mensaje “**Pepe**”
- Vamos a ver si entendemos el “**Programa12**”...
 - **Dim contador As Integer**
Definimos una variable de nombre **contador** y tipo **Integer**
 - **contador = 1**
Inicializamos (asignamos) la variable **contador** a un valor igual a la unidad.
 - **Do While** condición
Instrucción 1
Instrucción 2
Loop

Se trata de la estructura de programación **Do – While – Loop**, que funciona de la siguiente forma:

“Mientras se vaya cumpliendo la condición, se ejecutará la instrucción 1 y 2”

Es decir, en nuestro caso:

Mientras la variable **contador** sea inferior o igual a **5**:

1º) Aparece el mensaje **“Pepe”**

2º) El valor del contador se incrementa en una unidad.

Veamos:

- Inicialmente el **contador** = 1
- Como se cumple la condición (contador <= 5), aparece **“Pepe”** y **contador** = 2
- Como se cumple la condición (contador <= 5), aparece **“Pepe”** y **contador** = 3
- Se repetirá el proceso anterior, **mientras** el contador <= 5

Conclusión: al ejecutar el **Programa12**, aparece **5 veces** el mensaje **“Pepe”**.

o) Se trata de hacer un programa que nos escriba tantas veces como queramos **PEPE...**

La solución podría ser la siguiente:

```
Sub Programa13()
  Dim pregunta As String * 1
  pregunta = "S"
  Do While pregunta = "S" Or pregunta = "s"
    MsgBox "PEPE"
    pregunta = InputBox("¿Quieres continuar?")
  Loop
End Sub
```

- Escribe el programa anterior en el **Módulo1**, pruébalo y grábalo.

p) Se trata de hacer un programa que vaya sumando los números (enteros) que queramos.

La solución podría ser la siguiente:

```
Sub Programa14()
  Dim num As Integer, total As Integer
  total = 0
  num = InputBox("Escribe un número")
  Do While num <> 0
    total = total + num
    num = InputBox("Escribe un nuevo valor")
  Loop
  MsgBox "La Suma total es " & total
End Sub
```

- Escribe el programa anterior en el **Módulo1**. Pruébalo y grábalo.

- Está claro, que debido a la condición de nuestro While, para acabar hemos de escribir **0**.
- Observa de qué forma conseguimos “acumular la suma”:

total = total + num

Es decir: nuevo valor de **total** = anterior valor de **total** + valor actual de **num**.

q) Vamos a estudiar una nueva estructura de programación...

Escribe en el **Módulo1** el siguiente programa:

```

Sub Programa15()
  Dim indice As Integer
  For indice = 1 To 10 Step 2
    MsgBox "El valor del índice es = " & indice
  Next
  MsgBox "Lo siento se ha acabado. " & _
    "El último valor del índice ha sido " & _
    indice
End Sub

```

- Ejecuta el programa anterior y grábalo.
- Observa el funcionamiento del ciclo **FOR – TO – NEXT**:

```

For indice=1 To 10 Step 2
  Instrucción1
  Instrucción2
Next

```

Las instrucciones “encerradas” entre **For** y **Next**, en nuestro caso un mensaje que nos da el valor del índice, se van repitiendo: *For indice =1 To 10 Step 2*, es decir, desde el valor inicial de “índice” que es 1 hasta 10 de 2 en 2.

Veamos:

Al iniciarse el ciclo:

Indice = 1

Se muestra el mensaje “**El valor del índice es = 1**”

Al encontrarse la sentencia **Next**, se vuelve a iniciar el ciclo.

Al volverse a iniciar el ciclo:

Indice = 3 (ya que **Step = 2**)

Se muestra el mensaje “**El valor del índice es = 3**”

Al encontrarse la sentencia **Next**, se vuelve a iniciar el ciclo.

Al volverse a iniciar el ciclo:

Indice = 5

Aparece el mensaje: “**El valor del índice es = 5**”

Al volverse a iniciar el ciclo:

Indice = 7

Aparece el mensaje: “**El valor del índice es = 7**”

Al volverse a iniciar el ciclo:

Índice = 9

Aparece el mensaje: **“El valor del índice es = 9”**

Al volverse a iniciar el ciclo:

Índice = 11

Cómo teníamos de empezar por 1 y acabar en **10** (de 2 en 2). Salimos del ciclo **For – To – Next**

Y aparece el mensaje (que hay fuera del **ciclo**): **“Lo siento se ha acabado. El último valor del índice ha sido 11”**

r) Al trabajar con la sentencia **MsgBox**, sólo podemos visualizar un mensaje o valor, ya que para visualizar el siguiente **MsgBox** es necesario antes “cerrar” el anterior. Nos gustaría “ver” todos los mensajes o valores que genera el programa...

Escribe en el **Módulo1** el siguiente procedimiento:

```

Sub Programa16()
    Dim num As Integer, i As Integer
    Dim nom As String
    num = InputBox("¿Cuántas veces quieres que te salude?")
    nom = InputBox("¿Cuál es tu nombre?")
    For i = 1 To num
        Debug.Print "Hola " & nom
    Next
End Sub

```

- Ejecuta y graba el **Programa16**

- Observa:

- Definimos dos variables “Integer”, la primera: **“num”**, que indicará el número de “mensajes”, en nuestro caso el número de veces que se ha de repetir el ciclo **For – To – Next**. La segunda variable integer “i”, corresponde al **“índice”** del ciclo **For – To – Next**
- La variable **“nom”**, tipo texto (String) almacenará “nuestro nombre”.
- El programa nos pregunta: **“cuántas veces queremos el saludo”**, que no es más que el valor de la variable **“num”** o número de veces que queremos se repita el ciclo **For – To – Next**.
- Observa la sentencia que está en el ciclo **For – To – Next**:

```

Debug.Print "Hola " & nom

```

Indica que la frase “Hola + el nombre introducido”, se escriba en la **Ventana Inmediato**. La ventaja que tiene la sentencia **Debug.Print** a diferencia del **MsgBox**, es que todo lo que se escribe en la ventana “Inmediato”, permanece.

Ten en cuenta que el **panel inferior** de la “Ventana Inmediato”, donde escribimos el nombre del procedimiento para que se ejecute y también donde aparece el contenido de los “Debug.Print”, se puede hacer más grande.

s) Se trata de hacer un programa que “resuelva cualquier ecuación de 2º grado”.

Recordemos:

Dada la ecuación: $ax^2 + bx + c = 0$, con “a” diferente de 0

Definimos “discriminante” de la ecuación:

$$\text{Discriminante} = b^2 - 4ac$$

Si **Discriminante > 0**

Las soluciones de la ecuación son:

$$x_1 = \frac{-b + \sqrt{\text{Discriminante}}}{2a}$$

$$x_2 = \frac{-b - \sqrt{\text{Discriminante}}}{2a}$$

Si **Discriminante = 0**

La única solución de la ecuación es:

$$x = \frac{-b}{2a}$$

Si **Discriminante < 0**

La ecuación no tiene soluciones “reales”

- Escribe en el **Módulo1** el siguiente programa:

```

Sub Programa17()
    Dim A As Double
    Dim b As Double
    Dim c As Double
    Dim dis As Double
    Dim x1 As Double
    Dim x2 As Double
    Dim x As Double
    A = InputBox("Coeficiente de x^2 = ")
    Debug.Print "Coeficiente de x^2 = " & A
    If A = 0 Then
        Debug.Print "No es una ecuación de 2º grado"
    Else
        b = InputBox("Coeficiente de x = ")
        Debug.Print "Coeficiente de x = " & b
        c = InputBox("Termino independiente = ")
        Debug.Print "Termino independiente = " & c
        dis = b ^ 2 - 4 * A * c
        If dis < 0 Then
            Debug.Print "Las soluciones son imaginarias"
        End If
        If dis = 0 Then
            x = (-b) / (2 * A)
            Debug.Print "La ecuación tiene una única solución " _
                & "que es = " & x
        End If
        If dis > 0 Then

```

```

        x1 = (-b + Sqr(dis)) / (2 * A)
        x2 = (-b - Sqr(dis)) / (2 * A)
        Debug.Print "x1 = " & x1
        Debug.Print "x2 = " & x2
    End If
End If
End Sub

```

- Ejecuta el programa anterior 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
- Observa detenidamente cómo están escritos los **IF – END IF**
- **Sqr** es una función incorporada que calcula la raíz cuadrada.
- Recuerda que es muy importante **grabar** el “trabajo” que vas haciendo (CLIC en el icono **Guardar**)

t) Vamos a “definir” otro tipo de variable...

Escribe en el **Módulo1** el siguiente procedimiento:

```

Sub Programa18()
    Dim A(1 To 3) As Double
    Dim i As Integer
    For i = 1 To 3
        A(i) = InputBox("Introduce un número")
    Next
    Debug.Print "Los números que hay en la matriz son: "
    For i = 1 To 3
        Debug.Print "A(" & i & ")= " & A(i)
    Next
End Sub

```

- Ejecuta el procedimiento **Programa18** varias veces, observando detenidamente su funcionamiento.
- Veamos:

En nuestro programa, necesitamos “introducir 3 números”. Podríamos utilizar 3 variables (como hicimos en el programa 17) o una variable especial llamada **matriz o array** de una dimensión que “guarde” tres valores.

La variable **A** del programa 18 es una “matriz de una dimensión” (llamada también **vector**) de 3 valores...

- Observa la declaración de nuestra matriz:
Dim A(1 To 3) As Double

A= nombre de la variable

1 To 3 = valores distintos que puede tomar, llamado también **índice** de la matriz.

Si en la declaración hubiéramos escrito: **Dim A(3) As Double**

sería equivalente a escribir: **Dim A(0 To 3) As Double**

es decir una matriz unidimensional de 4 valores o índice 4, ya que por omisión el 0 es el primero y el último el número que se indica).

- Observa de qué forma “indicamos” un valor concreto de la matriz:
A(1) es el primer valor
A(2) es el segundo
A(3) es el tercero

u) Vamos a hacer un programa más complicado...

Observa la siguiente tabla:

	Lunes	Martes	Miércoles	Jueves	Viernes
Inicio Jornada Laboral	8	10,5	6	9	7
Hora final de la Jornada	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 8h y terminamos a las 2h de la tarde, el martes empezamos a trabajar a las 10 y media y terminamos a las 17h etc.

Necesitamos de entrada un programa que nos permita introducir los 10 números que representan la hora de inicio y la hora de finalización de la jornada laboral, para cada uno de los días laborables de la semana.

La solución a nuestro problema, podría ser la utilización de una matriz de 2 dimensiones. Una dimensión indicaría el inicio y el final de la jornada diaria (2 valores) y la otra dimensión indicaría cada uno de los días laborables (5 valores).

Matriz de dos dimensiones:

Una dimensión = 2 valores

Otra dimensión = 5 valores

Total de valores de la matriz = 2 x 5 = **10 valores**

La matriz correspondiente la declararemos: **Dim A(1 To 2, 1 To 5) As Double**, de forma que:

A(1,3) indicará la hora de **inicio** del **miércoles**

A(2,4) indicará la hora de **finalización** del **jueves**

Etc.

- Escribe en el **Módulo1** el siguiente programa:

Sub Programa19()

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("hora inicio y después hora finalización, para cada día de la semana,
    empezando por el lunes y acabando en el jueves")
  Next
Next
For j = 1 To 5
  For i = 1 To 2
    Debug.Print A(i, j)
  Next
Next
End Sub

```

- Ejecuta el programa, introduciendo los valores de la tabla, siguiendo el orden que nos indica el **inputbox** del programa, es decir, deberás escribir los números de la tabla en el orden: 8; 14; 10,5; 17; etc.

Si todo funciona correctamente, en la “ventana de depuración” deberían aparecer los números que has introducido, siguiendo el orden de introducción.

- Vamos a intentar entender el programa 19, ya que aparecen unos cuantos “elementos nuevos”:
 - Dim A(1 To 2, 1 To 5) As Double
Definimos la matriz de 2 dimensiones: 2 x 5 = 10 valores tipo “Double”.
 - Dim i, j As Byte
Definimos 2 variables tipo **Byte**. Byte no es más que un número de 0 a 255. Ya que éstas variables (i, j) corresponderán a los índices de dos ciclos **FOR – TO – NEXT**, no hay necesidad de declararlas **Integer**. La única diferencia es que las variables **Byte** ocupan menos memoria que las **Integer**
 - For j=1 To 5
 For i=1 To 2
 A(i,j) = InputBox(“.....”)
 Next
Next

Se trata de dos ciclos for – to – next **anidados**, veamos como funcionan:

```

Al inicio j=1
  For i=1 To 2
    A(i,1)= InputBox(“.....”)
  Next

```

Es decir:

```

J=1
  I=1      A(1,1)=
  I=2
J=2
  I=1      A(1,2)
  I=2      A(2,2)
J=3
  I=1

```

```

                A(1,3)
            I=2
                A(2,3)
        Etc.
        Etc.

```

- For j=1 To 5
 - For i=1 To 2
 - Debug.Print A(i,j)
 - Next
- Next

Se trata de dos ciclos for – to – next **anidados**, igual que los anteriores, con la única diferencia que en este caso nos escribe en la ventana de depuración los valores que hemos “guardado” en los anteriores ciclos **for – to – next** anidados.

v) El programa anterior, programa 19, es muy bueno para utilizar matrices bidimensionales y ciclos anidados, pero es completamente inútil.

Se trataría de “modificar” el programa anterior para conseguir que el programa nos calculara el **número total de horas trabajadas a la semana...**

- Escribe en el **Módulo1** el siguiente programa:

```

Sub Programa20()
    Dim A(1 To 2, 1 To 5) As Double
    Dim i As Byte, j As Byte
    Dim suma As Double, 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
        Debug.Print "Una jornada= " & diaria
    Next
    Debug.Print "Toda la Semana= " & suma
End Sub

```

- Ejecuta el programa, introduciendo los mismos valores de la tabla. Para ejecutar el programa no es necesario que te sitúes en la “Ventana Inmediato” y escribas **Programa20** y [Return]. Basta que sitúes el cursor de escritura en cualquier punto del interior del **Programa20()** y pulses la tecla **[F5]**.

Si todo funciona correctamente en la “ventana de depuración” aparecerá:

```

Una jornada = 6
Una jornada = 6,5
Una jornada = 7,5
Una jornada = 4

```

Una jornada = 11
 Toda la semana = 35

- Observemos de qué forma lo conseguimos:

```
For j=1 To 5
    Diaria = A(2,j) - A(1,j)
    suma = suma + diaria
    Debug.Print "Una jornada =" & diaria
Next
Debug.Print "Toda la semana=" & suma
```

- Hemos inicializado antes de todo, la variable **suma** a **0**
- Cuando empieza el ciclo **for – to – next**:
 J=1
 diaria = A(2,1) – A(1,1)
 suma = suma + diaria
 La variable **diaria** guarda la jornada laboral del 1r. día (hora final – hora inicial)
 La variable **suma** = anterior **suma** (0) + contenido **diaria** (jornada del 1r. día).
- Cuando vuelve a iniciarse el ciclo:
 J=2 (2º día)
 diaria = A(2,2) – A(1,2)
 suma = suma + diaria
 La variable **diaria** guardará la jornada del 2º día
 La variable **suma** = **suma** anterior (jornada del 1r. día) + contenido de la actual **diaria** (jornada del 2º día)
- Está claro que la variable **suma** va guardando el total de “**jornadas**”
- Al acabarse el ciclo **for – to – next**, la variable **suma** contendrá el total de horas de trabajo de la semana. Por esta razón, la línea: **Debug.Print “Toda la semana= “ & suma**, nos da lo que queríamos conseguir: Toda la Semana = 35 horas de trabajo.

- w) Escribe el siguiente programa en el **Módulo1**:

```
Sub Programa21()
    Dim A As String * 1
    Dim i As Byte
    For i = 1 To 100
        A = InputBox("¿Quieres continuar?")
        If A = "N" Or A = "n" Then
            Exit For
        End If
    Next
    MsgBox "Se acabó"
End Sub
```

- Ejecuta el programa anterior varias veces, hasta que descubras para qué sirve la instrucción **Exit For**

x) Escribe en el **Módulo1** el siguiente programa:

```

Sub Programa22()
    Dim contador As Integer
    Dim fahrenheit As Integer
    Dim celsius As Integer
    Debug.Print "Temperaturas Farenheit y Celsius"
    Debug.Print "===== "
    For contador = -2 To 13
        celsius = 10 * contador
        fahrenheit = 32 + (celsius * 9) / 5
        'La fórmula anterior transforma la temperatura de
        'grados centígrados a grados farenheit
        Debug.Print " " & celsius & " " & fahrenheit
        If celsius = 0 Then
            Debug.Print "Atención: Punto de congelación del Agua"
        End If
        If celsius = 100 Then
            Debug.Print "Atención: Punto de ebullición del agua"
        End If
    Next
End Sub

```

- Prueba el programa anterior. Es "guapo" ¿verdad?. Suponiendo que te funcione, ¡claro!

y) Queremos hacer un programa que nos dé la suma y el producto de todos los números pares hasta 30...

Escribe en el **Módulo1** el siguiente programa:

```

Sub Programa23()
    Dim par As Integer
    Dim sum As Integer
    Dim pro As Double
    sum = 0: pro = 1
    Debug.Print "Par - Suma parcial - Producto parcial"
    For par = 2 To 30 Step 2
        sum = sum + par
        pro = pro * par
        Debug.Print par & " - " & sum & " - " & pro
    Next
    Debug.Print "Suma total = " & sum
    Debug.Print "Producto total = " & pro
End Sub

```

- Ejecuta el programa anterior, para comprobar que funciona.
 Observa de qué manera podemos escribir varias sentencias en una misma línea de programa: basta **separarlas con dos puntos**.

z) En **VBA** hay diferentes tipos de programas. Todos los "programas" que hemos hecho hasta ahora se llaman **procedimientos** y tienen la estructura:

```

Sub NombrePrograma()
    .....
    .....
End Sub

```

Veamos otro tipo de programa en **VBA**, que se llama **función**.

- Escribe en el **Módulo1** la siguiente **función**:

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

- Antes de “ejecutar el programa anterior” observemos:
 - El programa anterior “pretende” calcular el promedio de 2 números enteros: n1 y n2, los dos declarados como **Integer**.
 - El “resultado” del programa anterior “pretende” ser el valor de “**Media**”, variable declarada como **Double**, cuyo valor debe ser el resultado de la fórmula: **Media = (n1 + n2)/2**, es decir el promedio de los números **n1** y **n2**.
- Vamos a “ejecutar” el programa “Media”, es decir:
 - [CTRL][G] para abrir la “ventana inmediato” (si no está ya abierta).
 - Escribe el nombre del programa, es decir: **Media** y pulsa [Return]
 - Está claro que alguna cosa no funciona, ya que aparece un mensaje de error.
 - “Acepta” el mensaje de error.
 - Escribe en la “ventana inmediato”:
`?Media(2,3)` y [Return]
 Si todo va bien, aparece **2,5** (promedio de 2 y 3)
- Vuelve a ejecutar el “programa Media”, para calcular el promedio de los números 153 y 352. Es decir:
 - En la “Ventana de Depuración”, escribe:
`?Media(153,352)` y [Return]
 Si todo va bien, debe aparecer el número **252,5**
- Está claro con lo hecho hasta ahora que una **función** es un programa, pero de un tipo distinto a un **procedimiento**.
- Observa la estructura de una función:

```
Function Nom(.....) As .....
    .....
    .....
    .....
End Function
```

- En el paréntesis que hay al lado del nombre de la función, hemos de colocar los argumentos de la función y declararlos. En nuestra función **Media**, los argumentos son n1 y n2, los dos tipo **Integer**.
- Los argumentos de la función son los valores que necesitará la función para ejecutarse. En nuestro caso al escribir: **Media(2,3)** estamos llamando a la función **Media**, con n1=2 y n2=3.

- Una función siempre devuelve algún valor, que debe declararse. En nuestro caso es **Double** y la forma de declarar es:
Function nombre (..... ,) As Double
- El "interior" de la función debe contener la relación entre los **argumentos** de entrada (n1 y n2) con el valor de salida (Media), en nuestro caso es simplemente la fórmula:
Media = (n1 + n2) / 2

a1) Escribe en el **Módulo1** la siguiente función:

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

- Ejecuta la función anterior de la siguiente forma:
 - Sitúate en la "ventana inmediato"
 - Escribe: **?Raiz4(625)** y pulsa [Return]
 - Si todo va bien aparecerá 5, que no es más que la raíz cuadrada de la raíz cuadrada, es decir la raíz cuarta de 625.
 - Escribe: **?Raiz4(72.81)** y [Return]
Si todo va bien, aparecerá: **2,92110896612857** que no es más que la raíz cuarta de 72.81
 - Escribe: **?Sqr(72.81)** y [Return]
 - Escribe: **?Sqr(8.53287759199674)** y [Return]

Está claro, ¿verdad?. La instrucción **Sqr** no es más que una función, pero diferente a la función **Raiz4**: la **Sqr** es una función incorporada en el **VBA**, en cambio **Raiz4** es una función definida por el usuario.

- Escribe: **?MsgBox("Hola Pepe")** y pulsa [Return]

Así pues, **MsgBox** también es una **función** incorporada en el VBA (siempre y cuando escribamos el **argumento** entre paréntesis).

- Escribe: **?Raiz4(-7)**

Está claro el problema: No existe la raíz cuadrada de un número negativo. Haz CLIC en [Finalizar] para aceptar el mensaje de error.

- Vamos a mejorar la función **Raiz4**...
 - Modifica la función **Raiz4** de forma que nos quede de la siguiente forma:

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

- Prueba la función anterior de la siguiente forma:
 - En la “ventana inmediato” escribe: **?Raiz4(17)** y [Return]
Si todo va bien debe aparecer 2,03054318486893
 - Escribe: **?Raiz4(-17)** y [Return]
Si todo va bien debe aparecer el mensaje de error. “Acepta” el mensaje.

b1) Sabemos como llamar una función desde la **Ventana de Depuración (inmediato)**, pero también podemos ejecutar una función desde un procedimiento. En efecto:

Escribe en el **Módulo1** el siguiente procedimiento:

```
Sub Programa24()  
    Dim num As Double  
    num = InputBox("Introduce un número")  
    Debug.Print "La raíz cuarta de " & num & " es " _  
        & Raiz4(num)  
End Sub
```

- Prueba el programa 24
 - Si “pruebas” el programa para un número negativo, observarás que también funciona
- Vamos a cambiar un poco el procedimiento anterior
- Escribe en el **Módulo1** el siguiente procedimiento:

```
Sub Programa25()  
    Dim num As Double  
    num = InputBox("Introduce un número")  
    If num < 0 Then  
        Debug.Print "Lo siento, vuelve a probarlo"  
    Else  
        Debug.Print "La raíz cuarta de " & num & " es " _  
            & Raiz4(num)  
    End If  
End Sub
```

- Prueba el programa anterior utilizando números positivos y también negativos.

c1) Vamos a estudiar en este apartado una nueva **estructura de programación...**

- Escribe en el **Módulo1** el siguiente **procedimiento**:

```
Sub Programa26()  
    Dim sexo As String  
    sexo = InputBox("Escribe VARÓN o HEMBRA, según tu sexo")  
    Select Case sexo  
        Case "VARÓN"  
            Debug.Print "¿Qué tal, guapo?"  
        Case "HEMBRA"
```



```

        Debug.Print "¿Qué tal, guapa?"
    Case Else
        Debug.Print "¿Qué tal, sexo ambiguo?"
    End Select
End Sub

```

- Ejecuta el programa anterior (espero que te funcione). Recuerda que la forma más rápida de ejecutar el programa, es colocar el cursor de escritura en su interior y pulsar la tecla [F5].

- Observa la “nueva” estructura de programación **Select Case**:

```

Select Case variable
    Case un valor determinado de la variable
        Sentencia 1
        Sentencia 2
    Case otro valor de la variable
        Sentencia 3
        Sentencia 4
    Case Else
        Sentencia 5
        Sentencia 6
End Select

```

Según el valor de la “variable” se ejecutarán unas líneas de programa u otras. Si la variable no toma ninguno de los valores que nos interesan, podemos agrupar las líneas de programa en “Case Else”. La opción “Case Else” es opcional.

- d1) Escribe en el **Módulo1** los siguientes programas (3 funciones y un procedimiento):

```

Function AYUDA()
    Debug.Print "Escribe R para calcular el área de un rectángulo"
    Debug.Print "Escribe T para un triángulo"
End Function

```

```

Function Rectángulo(bas As Double, alt As Double) As Double
    Rectángulo = bas * alt
End Function

```

```

Function Triángulo(bas As Double, alt As Double) As Double
    Triángulo = bas * alt / 2
End Function

```

```

Sub Programa27()
    Dim opción As String * 1
    Dim al As Double, ba As Double
    opción = InputBox("¿Qué opción?")
    Select Case opción
        Case "R"
            ba = InputBox("Base del rectángulo")
            al = InputBox("Altura del rectángulo")
            Debug.Print "El área del rectángulo es =" & Rectángulo((ba), (al))

```

```
Case "T"  
    ba = InputBox("Base del triángulo")  
    al = InputBox("Altura del triángulo")  
    Debug.Print "El área del triángulo es =" & Triángulo((ba), (al))  
Case Else  
    AYUDA  
End Select  
End Sub
```

- Ejecuta el programa 27, es decir:
 - Abre la “ventana de depuración”
 - Escribe: **Programa27** y [Return]
 - Juega con las diferentes opciones (deberás ejecutar el programa varias veces).

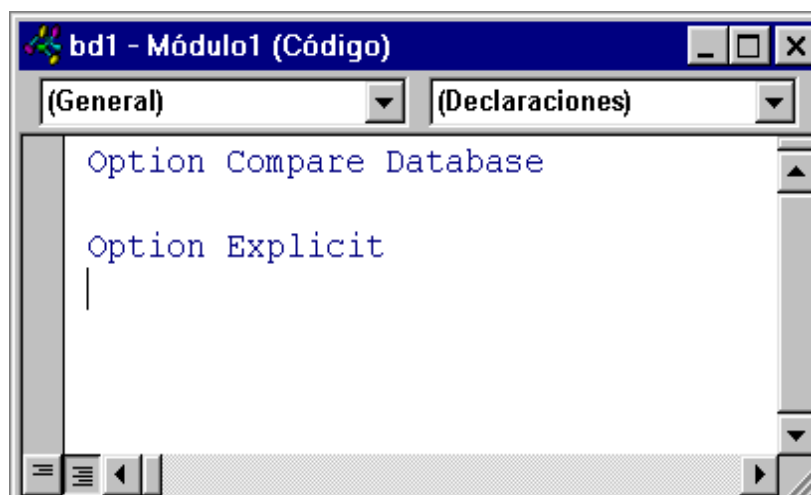
Para saber más

Declaración de Variables

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 previamente las variables que utilizamos en un programa, a diferencia de otros lenguajes de programación como el C/C++. Sería interesante **obligar** al Visual Basic 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” de un módulo la línea: **Option Explicit**



De todas formas, podemos conseguir que el **VB** lo haga por nosotros, basta que hagas: Desde el “Editor de Visual Basic”:

Menú Herramientas
 Opciones...
 Solapa: Editor
 Activa la casilla:
Requerir declaración de variables

Tipos de Variables más usuales

Tipo	Valor
Byte	0 a 255
Boolean	True o False
Integer	-32768 a 32767
Long	-2147483648 a 2147483647
Double	(números decimales positivos y/o negativos)
Date	1/1/100 a 31/12/9999
String	(cadena alfanumérica)
Variant	(tipo de datos por defecto)

Para más detalles consulta el tema: “Resumen de tipos de datos” de “Grupos” de “Referencia del lenguaje de Visual Basic”, de la **ayuda del Visual Basic del Access 2000** (basta que te sitúes en el “Editor de Visual Basic” del Access y Menú Ayuda – Ayuda de Microsoft Visual Basic)

“**Variant**” es el tipo de datos, para todas las variables si no se declaran explícitamente (Dim). No es más que un tipo de datos especial que puede contener cualquier clase de datos.

Observa el siguiente programa:

```

Sub ProgramaX()
  Dim num1, num2 As Integer
  num1 = InputBox (“Escribe un número”)
  num2 = InputBox (“Escribe otro número”)
  MsgBox “La Suma es = “ & num1 + num2
End Sub

```

En principio, el programa anterior funciona sin problemas, es decir sumará los números **num1** y **num2**, pero...

En la línea **Dim num1, num2 As Integer**

Declaramos la variable **num2** como **Integer**, y la variable **num1** como no tiene “tipo definido”, es **Variant**

El **Access** detecta en la operación **num1 + num2**, que deseamos operar dos números y convierte la variable **num1** a numérica. El problema está en que la variable “**Variant**” ocupa más espacio en memoria que una “Integer” y además el programa se ralentiza, porque debe **convertir** la variable.

En definitiva: es conveniente declarar el tipo de cada una de las variables (aunque sea el mismo). En nuestro caso: **Dim num1 As Integer, num2 As Integer**

Función MsgBox

El “**MsgBox**” es una función incorporada en el **VB** que tiene muchas posibilidades, veámoslo:

- Recupera la base de datos **PROGRAMAS.mdb**
- Selecciona el **Objeto: Módulos** y click en [Nuevo]

- Escribe el siguiente procedimiento:

```

Sub Programa28()
  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", , "Título"
  MsgBox "Mira el icono de " & vbCrLf & "pregunta", _
    vbQuestion, "Icono Interrogación"
  MsgBox "Otro icono", vbCritical, "Icono Crítico"
  MsgBox "otro", vbExclamation, "Icono Exclamación"
  MsgBox "otro más", 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")
  Debug.Print Respuesta
  Respuesta = MsgBox("tres botones", vbYesNoCancel + _
    vbInformation, "Con icono Información")
  Debug.Print Respuesta
  Respuesta = MsgBox("tres botones pero" & vbCrLf & _
    "el activo es el segundo", vbAbortRetryIgnore _
    + vbCritical + vbDefaultButton2, "Icono Critico")
  Debug.Print Respuesta
End Sub

```

- Click en el icono “Guardar”
Graba el nuevo módulo con el nombre **Módulo2**
- Pulsa las teclas [CTRL][G] para abrir la ventana “Inmediato”
- Escribe **Programa28** y pulsa [Return]

Observa detenidamente lo que sucede y ejecútalo varias veces para verlo mejor.

Explicación del Programa28:

- El primer “MsgBox”:
MsgBox(“Hola” & nom)
es el tipo de cuadro que ya habíamos utilizado
- 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

Función **InputBox**

Igual que sucede con el **MsgBox**, el **InputBox** tiene más posibilidades que las vistas hasta ahora.

- Escribe en el **Módulo2** de **PROGRAMAS** el siguiente procedimiento:

```
Sub Programa29()
  Dim Respuesta As String
  Respuesta = InputBox("Primera línea" & vbCrLf _
    & "segunda línea", "Título del InputBox")
  Respuesta = InputBox("Haz CLIC en [Cancelar]", _
    "A ver que pasa si cancelo")
End Sub
```

```

Debug.Print "Al pulsar cancelar resulta= " & Respuesta
Respuesta = InputBox("Aparece un valor por defecto", _
    "Título", "Esto aparece por defecto")
Respuesta = InputBox("Sitúo 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 observando detenidamente lo que sucede.
- En definitiva, la sintaxis completa de la función "inputbox" es:
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.

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 como funciona esta ayuda inteligente (IntelliSense), 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 sintaxis 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 **Debug** y un punto.

Basta seleccionar la propiedad o método del cuadro y pulsar la tecla [Tab].

Ya veremos más adelante, qué es exactamente un objeto, propiedad y método.

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 constantes y pulsar [Tab].

Si te molesta la "ayuda inteligente" basta que pulses la tecla [Esc] cuando aparece.

Comencemos a programar visualmente

En este primer capítulo nos hemos introducido en el estudio del **Visual Basic**, sin utilizar ninguna de las posibilidades “visuales” de que dispone el **Access**. En el siguiente capítulo nos dedicaremos precisamente a este “estudio”, pero nos gustaría ejecutar los programas que hemos escrito en el editor desde el **Access** no desde el **editor de Visual Basic**.

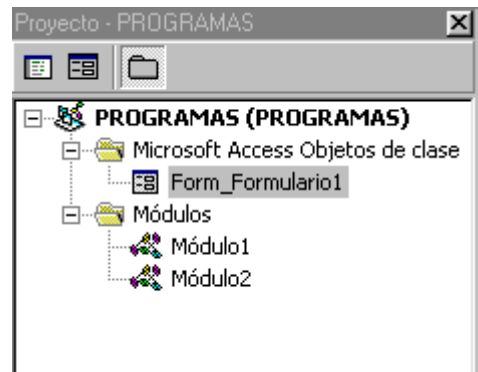
- Crea un formulario en la base de datos **Programas**. Es decir:
 - Desde la pantalla inicial de la base de datos **Programas**
 - Cinta de Opciones: Crear
 - Grupo: Formularios
 - Formulario en blanco
 - Cinta de Opciones: Herramientas de presentación de formulario
 - Ficha: Formato
 - Grupo: Vistas
 - Clic en la flechita de **Ver**
 - Vista Diseño
 - Observa la nueva **Cinta de Opciones: Herramientas de diseño de formulario**, con sus fichas **Diseño** y **Organizar**.
 - Observa los **controles** que tenemos en
 - Cinta de Opciones: Herramientas de diseño de formulario
 - Ficha: Diseño
 - Grupo: Controles
 - Observa también que el control **Utilizar Asistentes para controles** se encuentra activado por defecto
 - Desactiva el control **Utilizar Asistentes para Controles**
 - **Haz clic en el icono “Nbotón” del Grupo de Controles**
 - “Marca” un pequeño recuadro en el formulario
 - Con el **Comando0** seleccionado, haz clic en su interior.
 - Borra el texto que aparece por defecto y escribe en su lugar **“Saludo”**

- Con el botón [Saludo] seleccionado, haz clic en el icono **“Ver Código”**:
del **Grupo: Herramientas**



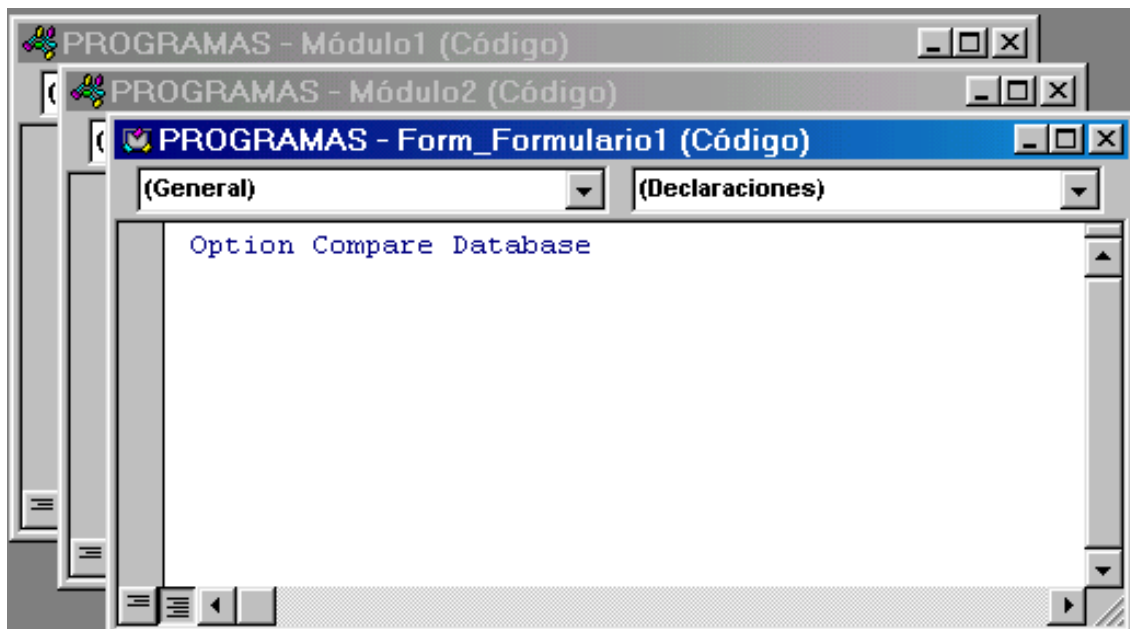
- Observa que hemos ido a parar al “Editor de Visual Basic”
Concretamente en una ventana llamada **“PROGRAMAS - Form_Formulario1 (Código)”**, que denominaremos **módulo del formulario** (a diferencia de los otros módulos)

- Observa la ventana **Proyecto - PROGRAMAS**:



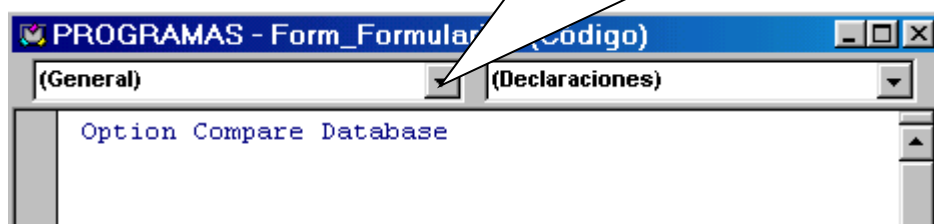
Si no la tienes a la vista, deberás hacer: **Menú Ver - Explorador de proyectos**. Donde podemos visualizar los diferentes objetos de nuestra base de datos, entre ellos los módulos 1 y 2.

También podemos visualizar los diferentes módulos, detrás del que hemos llamado “módulo del formulario”:



- Con el cursor de escritura en el interior del módulo de formulario, haz clic en la flecha del cuadro combinado de la izquierda:

flecha del cuadro combinado de la izquierda



- Selecciona la opción **Comando0**. Observa que en el cuadro combinado de la derecha aparece automáticamente **Click**.

Entre las líneas:


```
Private Sub Comando0_Click()
y
End Sub
```

Escribe: **Programa7**

El programa que tenemos en el “módulo del formulario”:

```
Private Sub Comando0_Click()
Programa7
End Sub
```

Es lo que se llama **procedimiento de evento** (ya los estudiaremos en detalle más adelante), que se ejecutará al hacer clic en el botón [Saludo] (el nombre del botón en realidad es **Comando0**), por esta razón el nombre del procedimiento es **Comando0_Click**

- Nuestro procedimiento de evento contiene una única instrucción que es **Programa7**. Es decir, se ejecutará el procedimiento **Programa7** que tenemos en el Módulo1

Vamos a ver si es verdad ...

- Sitúate en el **Microsoft Access**, es decir, clic en el icono **Ver Microsoft Access**:
- Graba el formulario con el nombre **Formulario1** (clic en el icono **Guardar**)
- Ejecuta el **Formulario1**, es decir,
 - Cinta de Opciones: Herramientas de diseño de formulario
 - Ficha: Diseño
 - Grupo: Vistas
 - Clic en la flechita de **Ver**
 - Vista Formulario
- Haz clic en el botón [Saludo]



Espero que te funcione correctamente, es decir, espero que se te ejecute el **Programa7** que habíamos escrito en el **Módulo1**.

Vamos a hacer otro botón ...

- Sitúate en la pantalla de diseño del formulario, es decir:
 - Cinta de Opciones: Inicio
 - Grupo: Vistas
 - Clic en la flechita de **Ver**
 - Vista Diseño
- Inserta en el formulario un nuevo botón de comando.
- Vamos a cambiar el texto que aparece en su interior de otra forma:
 - Con el botón **Comando1** seleccionado, y el cursor del ratón en su interior
 - Pulsa el botón derecho del ratón; de esta forma accedemos a lo que se llama su “menú contextual”.
 - Selecciona la opción “**Propiedades**”. En la propiedad **Título**, escribe: **Suma de números (parar acabar introduce el cero)**

- Vuelve a acceder al **menú contextual** del segundo botón del formulario y clic en la opción **Generar evento ...**
- Selecciona la opción **Generador de código** y [Aceptar]

Si todo ha funcionado correctamente nos hemos situado en el **Editor de Visual Basic**, exactamente en el **módulo del formulario** y aparece el “esqueleto” del procedimiento de evento correspondiente

- Escribe:


```
Private Sub Comando1_Click()  
    Programa14  
End Sub
```
- Graba los cambios que hemos hecho (clic en el icono **Guardar**)
- Vuelve al **Microsoft Access**. Ejecuta el **Formulario1** y prueba el funcionamiento del nuevo botón.

Nos gustaría hacer un nuevo botón, pero para ejecutar el **Programa16**.

Recordemos:

```
Sub Programa16()  
    Dim num As Integer, i As Integer  
    Dim nom As String  
    num = InputBox("Cuántas veces quieres que te salude ")  
    nom = InputBox("cuál es tu nombre")  
    For i = 1 To num  
        Debug.Print "Hola " & nom  
    Next  
End Sub
```

Observa el problema: La “salida” del programa es en la **Ventana de Inmediato** del editor de VB.

Vamos a ver como lo solucionamos:

- Desde la pantalla de diseño del **Formulario1**, inserta un nuevo botón de comando con el texto **Repite Saludo**
- Accede al **módulo del formulario** y escribe el siguiente procedimiento de evento:


```
Private Sub Comando2_Click()  
    Programa16  
    DoCmd.OpenModule "Módulo1", ""  
End Sub
```

La instrucción: **DoCmd.OpenModule "Módulo1", ""**, no es más que una instrucción “Visual Basic” pero propia del **Access** (que ya iremos estudiando más adelante), que simplemente abrirá el **Módulo1**

En definitiva nuestro programa hará lo siguiente:

- Ejecutará el **Programa16**
- Nos situaremos en el **Editor Visual Basic**, para visualizar la ejecución en la Ventana Inmediato (por supuesto que hemos de dejar abierta esta ventana).

- Recuerda de grabar los cambios que hemos hecho y sitúate en el **Formulario1**
- Haz clic en [Repite Saludo]

Espero que te funcione correctamente.

Para acabar este capítulo, vamos a hacer una última “mejora” ...

Ya que hemos empezado a ver las posibilidades “visuales” del VBA del Access, supongo que estarás de acuerdo conmigo en que es muy triste ejecutar un programa utilizando la ventana **Inmediato**.

En realidad, la utilidad de la ventana **Inmediato** está en depurar un programa, es decir básicamente para probarlo.

Vamos a modificar el **Programa16** para que sea más vistoso ...

- Sitúate en el “Editor de Visual Basic”
- Accede al **Módulo2**, es decir haz un doble clic en **Módulo2** en la ventana de **Proyectos**.
- Sitúate al final del **Módulo2** y escribe el siguiente programa:

```

Sub Programa30()
  Dim num, i As Integer
  Dim nom As String, salida As String
  salida = ""
  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
  MsgBox salida
End Sub

```

- Vuelve al **Microsoft Access**
- Inserta en el **Formulario1** un nuevo botón (Comando3) con el texto: **Repite Saludo Mejorado**
- Sitúate en el Editor VB
- Sitúate en el módulo del formulario. Basta hacer un doble clic en “**Form_Formulario1**” de la **ventana de proyectos**
- Escribe el siguiente procedimiento de evento:


```

Private Sub Comando3_Click()
  Programa30
End Sub

```
- Ejecuta el **Formulario1** y haz clic en [Repite Saludo Mejorado]
Espero que te funcione.
- Sitúate en el **Editor Visual Basic - Módulo2** para visualizar el **Programa30**:

```

Sub Programa30()

```

```

Dim num, i As Integer
Dim nom As String, salida As String
salida = ""
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
MsgBox salida
End Sub

```

- Definimos una nueva variable **String**, de nombre "salida" que inicializamos a vacío ("").
- **salida=salida & " Hola " & nom & vbCrLf**
En la variable **salida** se irá acumulando (salida = salida &), la palabra "Hola", el valor de la variable **nom** y un **vbCrLf** (es decir un [Return]).
- Por último mostramos con un cuadro de mensajes el valor final de "salida": **MsgBox salida**

Autoevaluación 1

En la **base de datos Programas**, crea un nuevo módulo de nombre **Autoevaluación1**, donde has de escribir los siguientes procedimientos:

- 1º) Haz un programa de nombre **Prog1**, 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 y salimos del programa.
 - El programa nos pregunta al final si queremos volver a empezar.

- 2º) Escribe un procedimiento de nombre **Prog2**, 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.

- 3º) Escribe un procedimiento de nombre **Prog3**, que haga lo mismo que el anterior, pero además nos dé la suma de todos los números introducidos, exceptuando el 9999

- 4º) Escribe un procedimiento de nombre **Prog4**, que haga lo mismo que el anterior, pero además nos dé el producto de los números introducidos, exceptuando el 9999.

- 5º) Escribe un procedimiento de nombre **Prog5**, que escriba todos los múltiplos de 23 inferiores a 1000 y por último nos dé la suma de todos ellos.

- 6º) Escribe un procedimiento de nombre **Prog6**, que sirva para hacer una tabla de valores de la función **$y=\text{sen}(7x-5)$**
 - El programa nos pide los dos valores de "x" (valores máximo y mínimo)
 - El programa nos pide el incremento (variación) de la "x".

- 7º) Escribe un procedimiento de nombre **Prog7**, 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.

8º) Escribe un procedimiento de nombre **Prog8**, que escriba los 15 primeros múltiplos de 7, su suma y su producto. El programa ha de tener la posibilidad de volver a empezar.

9º) Escribe un procedimiento de nombre **Prog9**, 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. Utiliza la estructura "Case".

10º) Escribe un procedimiento de nombre **Prog10**, que "dibuje" un rectángulo de asteriscos a partir de la base y la altura.

11º) Escribe un procedimiento de nombre **Prog11**, que "dibuje" un cuadrado con el carácter que introducimos por teclado, a partir del lado.

12º) Escribe un procedimiento de nombre **Prog12**, que nos pida un número y dé como resultado la tabla de multiplicar del número introducido.

13º) Escribe un procedimiento de nombre **Prog13**, que nos dé las opciones:

- Calcular una suma (crea una "function" de nombre suma)
- Calcular una raíz cuadrada (crea una "function" de nombre raizcua)
- Calcular un logarimo neperiano (crea una "function" de nombre neperiano).

14º) Escribe un procedimiento de nombre **Prog14**, que nos pida dos sumandos y nos dé la suma de sus cuadrados y su diferencia (utiliza las funciones: SumaCuadrado y RestaCuadrado)

15º) Escribe un procedimiento de nombre **Prog15** que funcione de la siguiente forma:

- El programa nos pide 10 valores.
- El programa calcula la media aritmética (function).
- El programa calcula las desviaciones respecto a la media.
- El programa calcula la desviación media (llamada a la función anterior).
- El programa calcula la varianza (llamada a la función anterior).
- Por último el programa calcula la desviación típica.

16º) Escribe un procedimiento de nombre **Prog16**, que calcule los 25 primeros términos de la sucesión de término general: $(3n+1) / (2n-1)$

17º) Crea un formulario de nombre **Formulario2** en la B. D. Programas con 5 botones, que permitan ejecutar los programas:

- Prog8 Múltiplos de 7
- Prog11 Dibuja cuadrado
- Prog14 Suma y Resta de cuadrados
- Prog15 Estadística
- Prog16 Sucesión.

2

Introducción a la Programación Visual

- a) Haz una **nueva** base de datos de nombre **VISUAL**, en *TuCarpeta*

En la base de datos **VISUAL** crea un formulario de la siguiente forma:

Cinta de Opciones: Crear
Formulario en blanco

Ver

Vista Diseño

- Si el icono “**Utilizar Asistente para controles**” está activado:



desactívalo.

- b) Se trata de colocar en el formulario un **Control: Cuadro de Texto**, de la siguiente forma:

- CLIC en el icono “**Cuadro de texto**” del **Grupo: Controles**



- Haz CLIC en cualquier punto del centro del formulario.
- Sitúate en la “etiqueta” del “cuadro de texto” que acabamos de crear. Es decir, CLIC en **Texto0** para seleccionar la etiqueta y otro CLIC para situarnos en su interior.
- Borra el texto que aparece por defecto y escribe en su lugar: **Mensaje**
- Haz CLIC en cualquier punto del formulario, fuera de la etiqueta del cuadro de texto.
- Vuelve a seleccionar el “cuadro de texto”, es decir: CLIC en el interior del “cuadro”, donde aparece el texto “Independiente”
- Puedes mover **la etiqueta** del “cuadro de texto”, sin más que situar el cursor del ratón en el extremo superior - izquierdo, y “arrastrar” el cursor hasta situar la etiqueta donde quieras.
- Puedes mover el **cuadro de texto** (recuadro donde aparece el texto “Independiente”), de la misma forma:
- Cursor en su extremo superior izquierdo
 - “Arrastra” el cursor del ratón.

- Mueve la “etiqueta” y el “cuadro de texto” de forma que nos quede aproximadamente de la siguiente forma:



- c) Vamos a colocar en nuestro formulario un botón
- CLIC en el icono “**Botón**” del **Grupo: Controles**
 - Haz CLIC en cualquier punto del formulario, que no sea en el “cuadro de texto”.
 - Con el botón que acabamos de crear, seleccionado. Haz CLIC en su interior. Borra el texto que aparece por defecto y escribe en su lugar: **VALE**
 - CLIC en cualquier punto fuera del botón.
 - Graba el formulario con el nombre **PROG1**, sin salir de la pantalla de diseño. Es decir:
 - Clic en el icono **Guardar** del extremo superior izquierdo de la pantalla.
 - En el campo “**Nombre del formulario**”
 - Escribe: **PROG1**
 - CLIC en [Aceptar]
- d) Vamos a crear un “**Procedimiento**” asociado al botón [**VALE**], pero antes...
- Selecciona el “cuadro de texto” (CLIC en su interior, donde hay el texto “Independiente”)
 - CLIC en el icono “**Hoja de Propiedades**”, del **Grupo: Herramientas**
 - Aparece la ventana de propiedades del control seleccionado, en nuestro caso del “cuadro de texto”. Asegúrate que la **Pestaña:Todas** está activada.
 - En la propiedad “**Nombre**”, aparece siempre un texto por defecto, normalmente Texto0, Texto1, etc. Cambia el texto que aparece en la propiedad “Nombre” por **CUADRO**
- Lo que hemos hecho es “bautizar” nuestro “cuadro de texto”, con el nombre **CUADRO**.
- Bautiza “nuestro botón” con el nombre **BOTÓN1**. Es decir:
 - Selecciona el botón [VALE]. Es decir, CLIC en su interior.

- Cambia el contenido de la **Propiedad: Nombre**, por **BOTÓN1** (no te dejes el acento).
- “Desmarca” el botón (CLIC en cualquier punto fuera del botón).

Tenemos de momento un formulario de nombre **PROG1** (por cierto, haz CLIC en el icono del “disquete” para guardar los últimos cambios). Formulario que contiene 2 controles:

- Un “cuadro de texto” de nombre **CUADRO** (aunque no se vea este nombre)
- Un “botón” de nombre **BOTÓN1** (aunque no se vea este nombre).

Antes de continuar: No se debe confundir el nombre que se ve en un control (en el caso del botón: VALE) con el nombre interno del control (en el caso del botón: BOTÓN1) que no se vé sino abrimos su ventana de propiedades.

En la “programación de los controles”, lo que cuenta es su “nombre interno”.

e) Vamos a crear un “procedimiento” para el botón VALE (nombre: BOTÓN1)...

- Selecciona el botón [VALE]
- Botón Derecho: Generar evento ...
- Si aparece una ventana que nos permite elegir entre tres generadores, haz CLIC en **Generador de código** y [Aceptar]

- Entre las líneas:
Private Sub BOTÓN1_Click()
y
End Sub

Escribe: **[CUADRO]=”¡Hola, qué tal!”**

- Observa nuestro **procedimiento**:

```
Private Sub BOTÓN1_Click  
    [CUADRO]=”¡Hola, qué tal!”  
End Sub
```

Que quiere decir: “Cuándo hagas CLIC en el control **BOTÓN1**, aparecerá en el control **CUADRO**, la frase **¡Hola, qué tal!**”

- Debes salir del “Editor de Visual Basic”, para volver al Access. Es decir, CLIC en el icono “**Ver Microsoft Access**”:



O si tú quieres: CLIC en el botón [Microsoft Access] de la “Barra de Tareas”

- Vuelve a grabar el formulario con el mismo nombre **PROG1** (CLIC en el icono del disquete).

f) Vamos a ver lo que hemos conseguido:

En estos momentos tenemos el formulario **PROG1** en “**Modo Diseño**”. Haz CLIC en la flechita del **icono Ver**, para pasar a **Vista Formulario**

- Haz CLIC en el botón [VALE]
- Si todo funciona correctamente, en el “cuadro de texto” acaba de aparecer la frase **¡Hola, qué tal!**
- Vuelve a hacer CLIC en la flechita del **icono Ver** para volver a **Vista Diseño**

Vamos a crear en el formulario **PROG1** otro botón de nombre **BORRAR** (con nombre interno **FUERA**), que sirva para borrar el contenido del cuadro de texto (el código correspondiente será: **[CUADRO]=”**)...

- CLIC en el icono “**Botón**”
- CLIC en el lugar del formulario donde quieras colocar el nuevo botón.
- Accede a sus “**Propiedades**”:
- En la propiedad “**Nombre**” escribe: **FUERA**
- Con el nuevo botón seleccionado. Haz CLIC en su interior para acceder a su “nombre externo”.
- Borra el texto que aparece por defecto y escribe en su lugar **BORRAR**
- CLIC en cualquier punto fuera del botón.
- Selecciona el botón [BORRAR]
- Accede a su menú contextual (botón derecho del ratón) y “Generar evento...”
- CLIC en “**Generador de código**” y [Aceptar]
- Escribe entre las líneas:
Private Sub FUERA_Click()
y
End Sub
La línea:
 [CUADRO]=”
- Vuelve al **Microsoft Access**
- Graba de nuevo el formulario con el mismo nombre **PROG1**
- CLIC en [BORRAR]
- CLIC en [VALE]
- CLIC en [BORRAR]

g) Antes de continuar con la “**programación de controles**”, vamos a mejorar el “**aspecto**” del formulario...

- Tenemos en estos momentos el formulario **PROG1** a la vista, en “**Modo Ejecución**”. Se trataría de eliminar los elementos que aparecen en el formulario, que sólo tienen sentido en un **formulario de base de datos** (Observa los iconos de navegación de la última línea de la pantalla) . Veamos:

- Sitúate en “**Vista Diseño**”
- “Selecciona el formulario”, es decir: CLIC en el botón:



Sabremos que el formulario está seleccionado, si aparece un cuadrado negro en el centro del botón anterior.

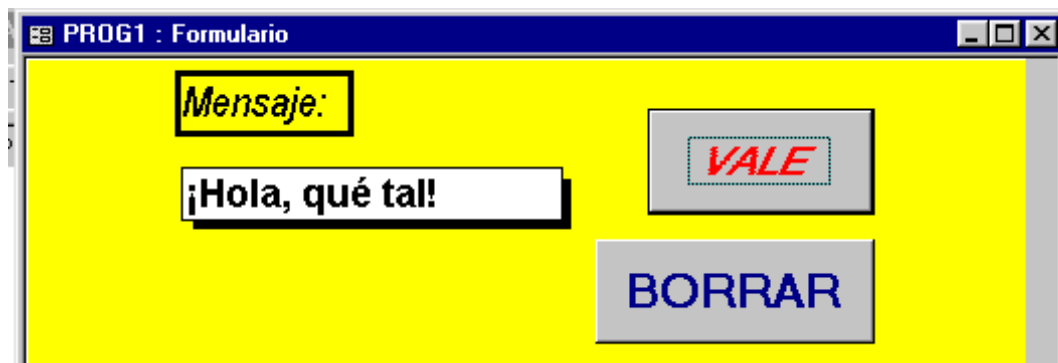
- CLIC en el icono “**Hoja de Propiedades**”
- Modifica las siguientes propiedades:
 - Selectores de registro: No
 - Botones de desplazamiento: No
 - Separadores de registros: No
- Ejecuta el formulario, es decir: **Ver: Vista Formulario**

Supongo que estarás de acuerdo conmigo en que el “formulario” tiene una presencia más elegante (no aparece el cuadro de controles de navegación en la última línea de la pantalla):

Vamos a mejorar aún más, el aspecto del formulario:

- Sitúate en **Vista Diseño** de nuevo
- Selecciona el “cuadro de texto”.

Se trata de que consigas el siguiente aspecto (aproximado) para el formulario **PROG1** (utilizando las opciones de la **Cinta de Opciones**):



- Graba el formulario **PROG1** con todos sus cambios con el mismo nombre.
- Ejecuta el formulario **PROG1**, y juega con “sus botones”.

h) Vamos a hacer otro programa...

- Crea un nuevo formulario de nombre **PROG2**, en la base de datos **VISUAL**, es decir:
 - Cinta de Opciones: Crear
 - Formulario en blanco
- Graba el formulario con el nombre **PROG2** y sitúate en **Vista Diseño**

- Inserta en el formulario **PROG2** un “cuadro de texto” con las siguientes características:
 - Etiqueta: **Pedido de fecha**
 - Propiedades:
 - Nombre: **PEDIDO**
 - Origen del Control: **=Fecha()**
- Graba el formulario con el mismo nombre **PROG2** (CLIC en el icono del disquete).
- Vamos a ver lo que hemos conseguido:
 - Ejecuta el formulario
 - Si todo va bien, aparece en el campo “Pedido de fecha”, la fecha actual (del reloj del ordenador).

El programa que tenemos hecho de momento (PROG2) consiste en un “cuadro de texto” que tiene asociado una función. Una función, que en nuestro caso es “incorporada por el Access” (podríamos definir una función propia, paciencia, ya lo veremos).

La “función” que podemos **asociar** al cuadro de texto, puede ser mas o menos complicada, puede ser incorporada o no incorporada por el Access, puede consistir también en una “operación de funciones”. Pero lo que es más importante por ahora es que observes “de qué manera asociamos una función a un control”.

Para asociar una función a un control, basta acceder a las “propiedades del control” y en la propiedad “**Origen del Control**”, escribir: **igual** y a continuación el **nombre** de la función. En nuestro caso: **=Fecha()**

- i) Vamos a “complicar” el programa **PROG2**...
- Se trata de incluir otro “cuadro de texto” en el programa **PROG2** con las siguientes características:
 - Etiqueta: **Fecha de Facturación**
 - Propiedades:
 - Nombre: **FACTURA**
 - Origen del Control: **=PrimerMes()**
 - Tendremos de “programar” una función de nombre “**PrimerMes()**”, que nos dé la fecha del primero de mes siguiente a la fecha actual.
 - Vamos a crear la función **PrimerMes()**...
 - Clic en **Ver Código** del **Grupo: Herramientas**
 - Con el cursor en la ventana: **VISUAL-Form_PROG2 (Código)**, es decir el módulo de nuestro formulario PROG2
 - Para escribir la función, haz lo siguiente:
 - Menú Insertar
 - Procedimiento...
 - Asegúrate que estén activadas las opciones: **Función** y **Ambito Público**.
En el campo **Nombre**, escribe: **PrimerMes** y CLIC en [Aceptar]
 - Escribe entre las líneas:

Public Function PrimerMes()

y

End Function

La siguiente línea:

PrimerMes= DateSerial(Year(Now), Month(Now) +1, 1)

(no te preocupes de momento en “entender” la línea de programa anterior)

- CLIC en el icono “**Guardar**”.

- Vuelve al “**Microsoft Access**”
- “Ejecuta” el **PROG2** (CLIC en el icono “**Vista**”, para pasar a “Modo Ejecución”)

Si todo funciona correctamente en el campo “**Fecha de Facturación**” aparece el día 1 del mes siguiente a la fecha que tienes en el campo “**Pedido de fecha**”.

Antes de continuar deberíamos tener claro lo que significa:

DateSerial (Year(Now), Month(Now)+1, 1)

En lugar de explicarte lo que quiere decir la fórmula anterior; prefiero enseñarte cómo puedes descubrirlo tú mismo...

Haz lo siguiente:

En primer lugar debes situarte en el **Editor de Visual Basic**

Menú Ayuda

Ayuda de Microsoft Visual Basic

Referencia del lenguaje de Visual Basic

Funciones

M-P

Now

etc

- Nos quedaría “investigar” lo que hace “**DateSerial**” y “**Month**”. Es importante que te acostumbres a “navegar” por la ayuda, ya que hay mas información en la **ayuda del Access** que en el libro más grueso que puedas encontrar en cualquier librería (y además es más barato). También es cierto, que muchas veces el “lenguaje técnico” que utiliza la ayuda no es demasiado afortunado. Piensa de todas formas, que cuánto más sepas sobre el “Visual Basic para Aplicaciones”, más provecho podrás sacar de la ayuda.


j) Vamos a “completar” el **PROG2**...

- Se trata de hacer una nueva función parecida a **PrimeroMes()**, pero que sirva para cualquier fecha...
 - Accede al “**Editor de Visual Basic**”
 - La nueva función que haremos, la colocaremos en un nuevo módulo. Haz lo siguiente:
 - Menú Insertar
 - Módulo y
 - Menú Insertar
 - Procedimiento...
 - Tipo: Función
 - Ámbito: Público
 - En el campo “**Nombre**” escribe: **PrimeroMesCualquiera** y CLIC en [Aceptar]
 - Escribe:

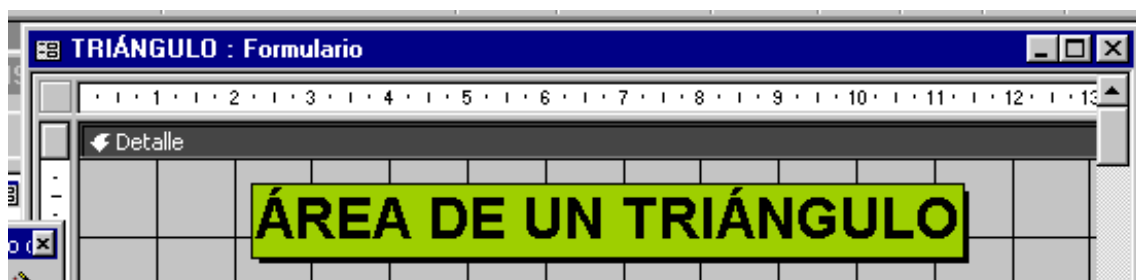

```
Public Function PrimeroMesCualquiera(Cual As Date) As Date
PrimeroMesCualquiera=DateSerial(Year(Cual), Month(Cual)+1, 1)
End Function
```
- “Vuelve” la ventana “**Microsoft Access**”.

- Vamos a colocar la función **PrimeroMesCualquiera** en el formulario **PROG2** de la siguiente forma:
 - “Abre” en **modo diseño** el formulario **PROG2**
 - Inserta un nuevo “cuadro de texto” con las características:
 - Etiqueta: **Escribe una fecha**
 - Propiedades:
 - Nombre: **FECHA**
 - Formato: **Fecha corta**
 - Valor predeterminado: **=Fecha()**
 - Inserta otro “cuadro de texto” con las siguientes características:
 - Etiqueta: elimínala (selecciona la etiqueta y pulsa [Supr])
 - Propiedades:
 - Origen del control: **=PrimeroMesCualquiera([FECHA])**
 - Formato: **Fecha larga**
- Graba el formulario **PROG2** con el mismo nombre y ejecútalo:
 - Escribe en el campo **“Escribe una fecha”**, la fecha que quieras.
 - Deseo fervientemente que te funcione correctamente, ya que en caso contrario deberás repasarlo todo y si no encuentras los posibles errores, deberás repetirlo.

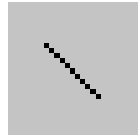
k) “Cierra” la base de datos **VISUAL** y crea una nueva base de datos (en tu carpeta) de nombre **VISUAL1**.

- Vamos a crear un formulario (en **VISUAL1**) de nombre **TRIÁNGULO** que sirva para calcular el área de un triángulo...
 - Crea un Formulario en blanco y sitúate en **Vista Diseño**
 - Desactiva (si está activado) el icono **“Utilizar Asistentes para controles”** del **Grupo: Controles**.
 - Graba el formulario con el nombre **TRIÁNGULO**
 - Vamos a colocar en primer lugar un “título guapo” a nuestro formulario...
 - CLIC en el icono **“Etiqueta”** del **“Grupo: Controles”**:
 
 - Traza un rectángulo en la parte superior del formulario (contendrá la frase: **ÁREA DE UN TRIÁNGULO**)
 - Escribe: **ÁREA DE UN TRIÁNGULO** y CLIC en cualquier punto fuera de la “etiqueta”.
 - Haz CLIC en la “etiqueta” anterior para seleccionarla.
 - Utilizando las opciones del **Grupo: Fuente** y **Grupo: Controles**

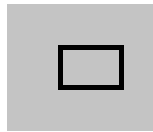
Consigue aproximadamente lo siguiente:



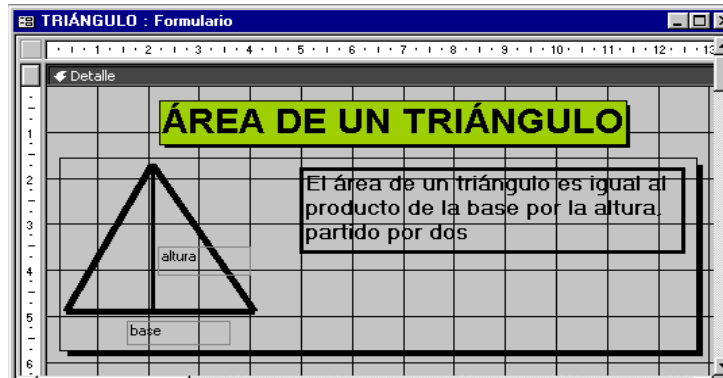
- Utilizando el icono “Línea”:



- el icono “Rectángulo”:



Además del icono “Etiqueta” que hemos utilizado anteriormente del “Cuadro de Herramientas”. Consigue, aproximadamente lo siguiente:



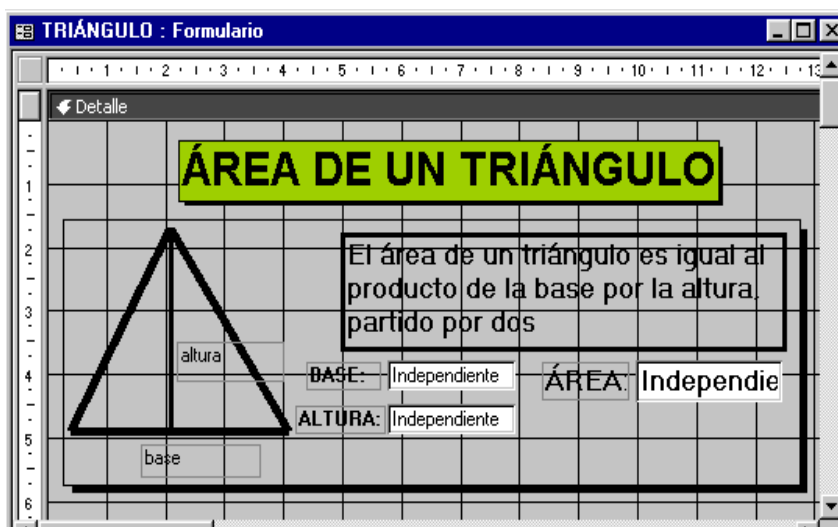
- Inserta en el formulario, 3 cuadros de texto, con las siguientes características:

Cuadro de texto 1: Etiqueta = **BASE** (en **Negrita**)
Propiedad “Nombre” = **base**

Cuadro de texto 2: Etiqueta = **ALTURA** (en **Negrita**)
Propiedad “Nombre” = **altura**

Cuadro de texto 3: Etiqueta = **ÁREA** (Negrita y Tamaño de Fuente: 12)
Propiedad “Nombre” = **triángulo** (negr. y fuen:12)

- Coloca los 3 cuadros de texto aproximadamente con la siguiente distribución:

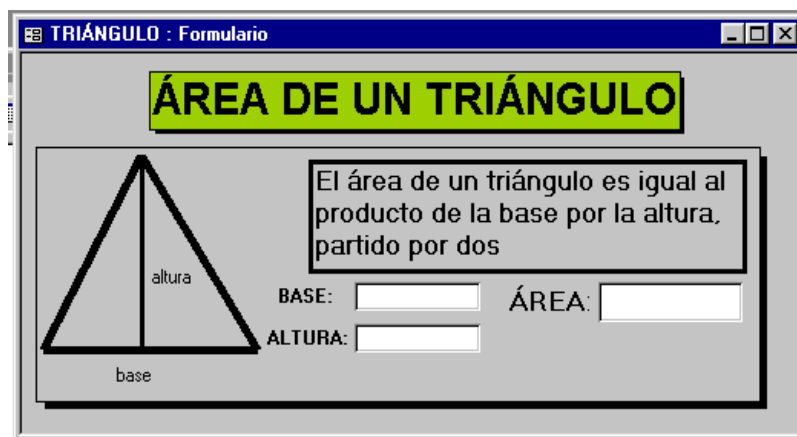


- Selecciona el formulario, es decir: CLIC en el botón



- CLIC en el icono “**Propiedades**”:
- Cambia las propiedades:
Barras de desplazamiento: **Ninguna**
Selectores de registro: **No**
Botones de desplazamiento: **No**
Separadores de registro: **No**
- Graba el formulario con el mismo nombre **TRIÁNGULO**
- Pasa a “Modo Ejecución” (**Vista Formulario**)

Deberás tener aproximadamente lo siguiente:



- Vuelve a “Vista Diseño”
- Inserta en nuestro formulario 2 botones de comando con las siguientes características:

Botón 1: Nombre Externo: **CALCULAR**
Propiedad “Nombre”: **calcular**

Botón 2: Nombre Externo: **BORRAR**
Propiedad “Nombre”: **borrar**

- Graba de nuevo el formulario con el mismo nombre **TRIÁNGULO**

Bien, la parte “visual” del programa ya está hecha, vamos a por la parte de “creación de código”...

Se trata de hacer el siguiente programa:

“Al hacer CLIC en [CALCULAR], el programa nos pregunta cuál es la base y la altura del triángulo y a continuación nos escribe en el formulario la base y la altura que hemos introducido y también nos escribe en el campo correspondiente, el área del triángulo”.

Veamos:

- Selecciona el botón [CALCULAR]
- Menú Contextual
Generar evento...
- Selecciona la opción “**Generador de código**” y [Aceptar]
- Escribe el siguiente programa:

```
Private Sub calcular_Click()
    Dim bas As Double
    Dim alt As Double
    Dim are As Double
    bas = InputBox("¿Cuál es la BASE del triángulo?")
    alt = InputBox("¿Cuál es la ALTURA del triángulo?")
    are = bas*alt / 2
    [base] = bas
    [altura] = alt
    [triángulo] = are
End Sub
```

- Graba lo que hemos hecho (CLIC en el icono “Guardar”)
- Vuelve al “Microsoft Access”
- Pasa a “Modo Ejecución” y CLIC en [CALCULAR]
Sería conveniente que probaras varias veces el “programa”
- Vuelve a “Modo Diseño”
- Vamos a “programar” el botón [BORRAR]...
 - . Selecciona el botón [BORRAR]
 - . Selecciona la opción “**Ver código**”
 - . Escribe el siguiente programa:

```
Private Sub borrar_Click()
    [base] = ""
    [altura] = ""
    [triángulo] = ""
End Sub
```

- . Vuelve al “Microsoft Access”
- . Pasa a “Modo Ejecución” y CLIC en [BORRAR]

Sería conveniente que probaras varias veces el “programa”, es decir, el funcionamiento de los botones [CALCULAR] y [BORRAR]

l) Se trata de hacer en este apartado lo mismo que en el anterior, pero para el “trapecio”...

- Desde la pantalla inicial de la base de datos **VISUAL1**
- Crear **Formulario en blanco**
- “Vista Diseño”

- Desactiva, si está activado, el icono “Utilizar Asistentes para controles” del Grupo: Controles.
- Graba el formulario con el nombre **TRAPECIO**
- Consigue aproximadamente el formulario que aparece en la siguiente ilustración:



Y con las siguientes características:

- Un control “Etiqueta” que contiene el texto: **ÁREA DE UN TRAPECIO**
- El dibujo de un trapezio, utilizando el icono “Línea”
- Tres controles “etiqueta” que contienen el texto:
 - base mayor
 - base menor
 - altura
 del dibujo.
- Un control “etiqueta” que contiene la “explicación” del área de un trapezio.
- Cuatro “cuadros de texto” con las siguientes características:
 - Cuadro de Texto 1: Etiqueta = **BASE MAYOR**
Propiedad “Nombre” = **base1**
 - Cuadro de Texto 2: Etiqueta = **BASE MENOR**
Propiedad “Nombre” = **base2**
 - Cuadro de Texto 3: Etiqueta = **ALTURA**
Propiedad “Nombre” = **altura**
 - Cuadro de Texto 4: Etiqueta = **ÁREA**
Propiedad “Nombre” = **trapezio**
- Dos “botones de comando” con las características:
 - Botón 1: Nombre externo = **CALCULAR**
Propiedad “Nombre” = **calcular**
 - Botón 2: Nombre externo = **BORRAR**
Propiedad “Nombre” = **borrar**

- Un recuadro que rodea todo el formulario, utilizando el icono **Rectángulo** del “**Grupo: Controles**”.
 - Cambia las siguientes “**propiedades del formulario**”:
Barras de desplazamiento = **Ninguna**
Selectores de registro = **No**
Botones de desplazamiento = **No**
Separadores de registro = **No**
- Asocia el siguiente programa al botón **[CALCULAR]**:

```
Private Sub calcular_Click()
Dim bmayor As Double
Dim bmenor As Double
Dim alt As Double
Dim trape As Double
bmayor = InputBox (“¿Cuál es la base mayor?”)
bmenor = InputBox (“¿Cuál es la base menor?”)
alt = InputBox (“¿Cuál es la altura?”)
trape = bmayor + bmenor * alt / 2
[base1] = bmayor
[base2] = bmenor
[altura] = alt
[trapecio] = trape
End Sub
```

- Asocia el siguiente procedimiento al botón **[BORRAR]**:

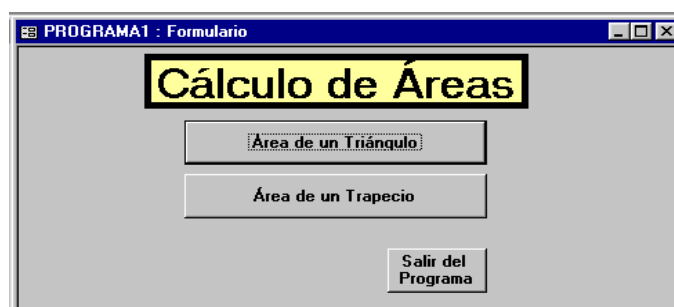
```
Private Sub borrar_Click()
[base1] = “”
[base2] = “”
[altura] = “”
[trapecio] = “”
End Sub
```

- Prueba el funcionamiento del formulario **TRAPECIO**

m) Vamos a hacer en la B. de D. **VISUAL1** un nuevo formulario de nombre **PROGRAMA1** que haga las veces de “**MENÚ**” para los dos formularios anteriores: **TRIÁNGULO** y **TRAPECIO**...

- Haz un **nuevo** formulario en blanco de nombre **PROGRAMA1** con el siguiente aspecto y características:

Aspecto:



Características:

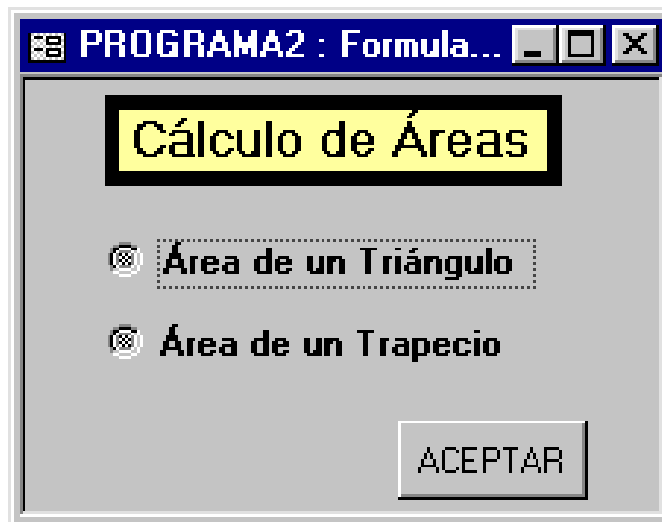
- Un control “etiqueta” que contiene el texto: **CÁLCULO DE ÁREAS**
 - Tres “botones de comando”:
 - Botón 1: Propiedad “Nombre”: **triángulo**
Propiedad “Título”: **Área de un Triángulo**
 - Botón 2: Propiedad “Nombre”: **trapecio**
Propiedad “Título”: **Área de un Trapecio**
 - Botón 3: Propiedad “Nombre”: **fuera**
Propiedad “Título”: **Salir del Programa**
 - Cambia las propiedades de siempre en el formulario **PROGRAMA1**:
Barras de desplazamiento = **Ninguna**
Selectores de registro = **No**
Botones de desplazamiento = **No**
Separadores de registro = **No**
- El “**código**” correspondiente deberá ser el siguiente:
- Para el botón 1:


```
Private Sub triángulo_Click()
    DoCmd.OpenForm "TRIÁNGULO"
End Sub
```
 - Para el botón 2:


```
Private Sub trapecio_Click()
    DoCmd.OpenForm "TRAPECIO"
End Sub
```
 - Para el botón 3:


```
Private Sub fuera_Click()
    DoCmd.Quit acExit
End Sub
```
- Observa la instrucción que sirve para “abrir un formulario” y la instrucción que sirve para salir del Access.
- Ejecuta y prueba los botones del formulario **PROGRAMA1**, pero antes graba el formulario.
- n) Se trata de hacer en la B. de D. **VISUAL1**, un nuevo formulario de nombre **PROGRAMA2** que haga lo mismo que el formulario **PROGRAMA1**, pero utilizando “**Botones de opción**” en lugar de “**Botones de Comando**”...
- Haz un nuevo formulario en blanco, de nombre **PROGRAMA2** con el siguiente aspecto y características:

Aspecto:



Características:

- Un control “etiqueta” que contiene el texto: “**Cálculo de Áreas**”
- Un control “**Botón de opción**”, deberás hacer CLIC en el icono “**Botón de opción**” del “Grupo: Controles”:



Con las siguientes características:

Propiedad “**Nombre**”: **triángulo**

En la etiqueta del botón, escribe: **Área de un Triángulo**

- Un segundo “botón de opción” con las siguientes características:
Propiedad “**Nombre**” = **trapecio**
Etiqueta = **Área de un Trapecio**
 - Cambia las “propiedades de siempre” en el formulario **PROGRAMA2**
 - Un “botón de comando” con las siguientes propiedades:
Propiedad “**Nombre**” = **continuar**
Propiedad “**Título**” = **ACEPTAR**
- El “**código**” que debes asociar al botón [**ACEPTAR**] es el siguiente:

```
Private Sub continuar_Click()
    If [triángulo] = True Then DoCmd.OpenForm "TRIÁNGULO"
    If [trapecio] = True Then DoCmd.OpenForm "TRAPECIO"
End Sub
```

Es decir:

Si el botón de opción 1º ([triángulo]) está activado (True), se abrirá el formulario **TRIÁNGULO**. Exactamente lo mismo para el formulario **TRAPECIO**.

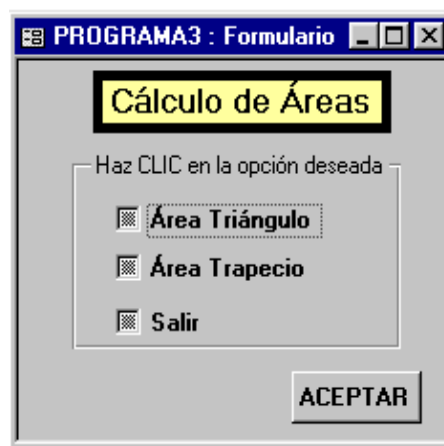
- Ejecuta y prueba el funcionamiento del formulario **PROGRAMA2**

Observa lo que sucede si tenemos “activados” los dos botones de opción y hacemos CLIC en [Aceptar].

o) Se trata de hacer otro formulario para la B. de D. **VISUAL1**, de nombre **PROGRAMA3** que haga lo mismo que el **PROGRAMA2**, pero solucionando el problema que hemos observado en el anterior formulario y además trabajaremos con un nuevo “control”: “**Casilla de verificación**” que es muy parecido al “botón de opción”...

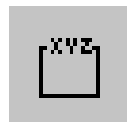
- Haz un nuevo formulario en blanco de nombre **PROGRAMA3** con el siguiente aspecto y características:

Aspecto:



Características:

- Un control “etiqueta” con el texto: **Cálculo de Áreas**
- Un control “**Grupo de opciones**”, deberás hacer CLIC en el icono “**Grupo de Opciones**” del “Grupo: Controles”:



Con las siguientes características:

Propiedad “**Nombre**” = **todo**

En la etiqueta del “grupo de opciones” escribe: **Haz CLIC en la opción deseada.**

- Tres controles “**Casilla de verificación**” (búscalo en el “Grupo: Controles”), que debes colocar en el interior del “Grupo de Opciones”, es decir, con el “Grupo de Opciones” seleccionado, haz CLIC en el icono “**Casilla de verificación**” y CLIC en el interior del **grupo de opciones**. Y escribe como “etiqueta” de las tres casillas de verificación:

Área Triángulo
Área Trapecio
Salir

- Un “botón de comando” con las siguientes propiedades:
Propiedad “Nombre” = **continuar**
Propiedad “Título”: **ACEPTAR**
- Cambia las propiedades de siempre, en el formulario **PROGRAMA3**.

- El “código” que debes asociar al botón [ACEPTAR] es el siguiente:

```
Private Sub continuar_Click()  
    Select Case [todo]  
        Case 1  
            DoCmd.OpenForm “TRIÁNGULO”  
        Case 2  
            DoCmd.OpenForm “TRAPECIO”  
        Case 3  
            DoCmd.Quit acExit  
    End Select  
End Sub
```

Antes de continuar, observa:

- Utilizamos la estructura “**Select Case – End Select**”, para evitar escribir tres estructuras “**If – Then**”, una para cada una de las 3 posibilidades de nuestro “grupo de opciones”.
 - Nuestro “grupo de opciones” tiene 3 posibilidades, ya que contiene 3 casillas de verificación (sería igual si fueran tres botones de opción). El valor del grupo de opciones: **[todo]**, puede ser **1** (si está activada la 1ª casilla), **2** (si lo está la 2ª) y **3** (si lo está la 3ª).
- Ejecuta y prueba el formulario **PROGRAMA3**. Observarás que sólo puedes activar una de las tres opciones, esto es debido a que se encuentran en un control “grupo de opciones”.

p) Como el formulario **PROGRAMA3** nos ha quedado muy bien, nos gustaría que se autoejecutara al abrir la B. de D. **VISUAL1**. Haz lo siguiente:

- Desde la pantalla inicial de la B. de D. **VISUAL1**, haz:
 - Botón del Office
 - Opciones de Access
 - Base de datos actual
 - Título de la aplicación, escribe: **ÁREAS**
 - Mostrar formulario: **PROGRAMA3**
 - [Aceptar]
- “Cierra” y vuelve a abrir la B. de D. **VISUAL1**. Si todo funciona correctamente debe aparecer el formulario **PROGRAMA3** y en la primera línea de la pantalla al lado del icono del “Access” debe aparecer el texto **ÁREAS**.

q) Vamos a hacer en este apartado, un programa que sirva para calcular el I.V.A. Intentaremos que sea un programa “cerrado” es decir que no se pueda modificar (una vez acabado) y que tenga un “aspecto profesional”...

- Crea una nueva “base de datos” de nombre **IVA** (en tu carpeta).

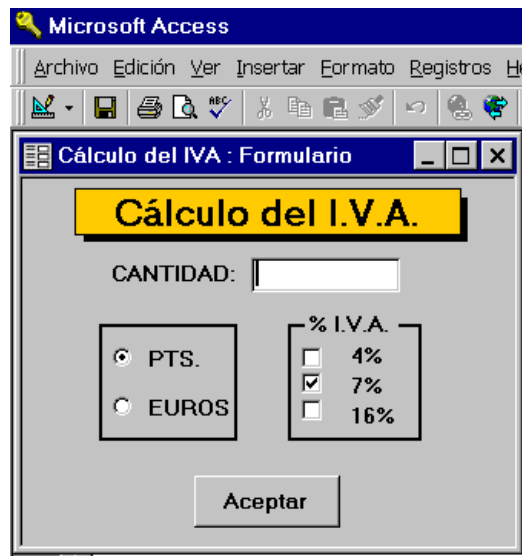
- Crea un formulario en blanco de nombre **“Cálculo del IVA”** con el siguiente aspecto y características:

Aspecto:

Características:

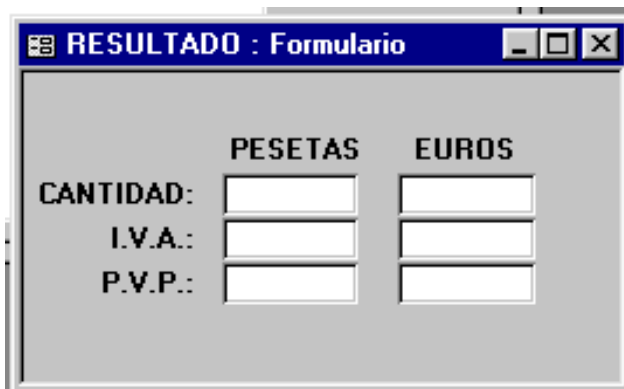
- Control **“etiqueta”** con el texto: **Cálculo del I.V.A.**
- Control **“cuadro de texto”** de características:
Propiedad **“Nombre”**: **cantidad**
Propiedad **“Formato”**: **Fijo**
Propiedad **“Lugares decimales”**: **2**
En su **“etiqueta”** escribe: **“CANTIDAD:”**
- Control **“grupo de opciones”** con dos **botones de opción**:
Grupo de Opciones:
Propiedad **“Nombre”**: **moneda**
Propiedad **“Valor predeterminado”**: **1**
Elimina su **“etiqueta”**
Botón de opción 1:
Etiqueta: **PTS.**
Botón de opción 2:
Etiqueta: **EUROS**
- Control **“grupo de opciones”** con tres **casillas de verificación**:
Grupo de Opciones:
Propiedad **“Nombre”**: **iva**
Propiedad **“Valor predeterminado”**: **2**
Etiqueta: **“% I.V.A.”**
Casilla de verificación 1:
Etiqueta: **4%**
Casilla de verificación 2:
Etiqueta: **7%**
Casilla de verificación 3:
Etiqueta: **16%**
- Control **“botón de comando”**:
Propiedad **“Nombre”**: **OK**
Propiedad **“Título”**: **Aceptar**
- Propiedades del formulario:
Barras de desplazamiento = **Ninguna**
Selectores de registro = **No**

Botones de desplazamiento = **No**
 Separadores de registro = **No**



- “Cierra” de momento el formulario “**Cálculo del IVA**” y crea un nuevo formulario en blanco de nombre **RESULTADO** con el siguiente aspecto y características:

Aspecto:



Características:

- Cinco controles “**etiqueta**” con el texto:
 PESETAS
 EUROS
 CANTIDAD:
 I.V.A.:
 P.V.P.:
- Seis “**cuadros de texto**”, todos ellos sin etiqueta y propiedad “**Nombre**”:

	PESETAS	EUROS
CANTIDAD	canpts	caneur
I.V.A.	ivapts	ivaeur
P.V.P.	pvppts	pvpeur

Y todos ellos con la propiedad “**Formato**” = **Fijo** y la propiedad **Lugares decimales = 2**

- Propiedades del formulario:
 - Barras de desplazamiento = **Ninguna**
 - Selectores de registro = **No**
 - Botones de desplazamiento = **No**
 - Separadores de registro = **No**
 - Ajuste de tamaño Automático = **No**
- Supongo que te habrás dado cuenta de lo que pretendemos, se trata de conseguir el siguiente programa:

<< Escribimos en el cuadro de texto **[cantidad]** del formulario **“Cálculo del IVA”** un número, seleccionamos **PTS** o **EUROS**, seleccionamos uno de los tres **“% de IVA”**. Al pulsar el botón **[Aceptar]** pretendemos que se abra el formulario **“RESULTADO”** con sus 6 campos que nos muestran el resultado de los cálculos >>.

Veamos:

- Selecciona en **“Modo Diseño”** el botón [Aceptar] del formulario **“Cálculo del IVA”** y asóciate el siguiente **“código”**:

```
Private Sub OK_Click()
  DoCmd.OpenForm "RESULTADO"
  If [moneda] = 1 Then
    Forms![RESULTADO].[caneur] = ""
    Forms![RESULTADO].[canpts] = [cantidad]
  End If
  If [moneda] = 2 Then
    Forms![RESULTADO].[canpts] = ""
    Forms![RESULTADO].[caneur] = [cantidad]
  End If
  If Forms![RESULTADO].[canpts] = "" Then Forms![RESULTADO].[canpts] = _
    Forms![RESULTADO].[caneur] * 166.386
  If Forms![RESULTADO].[caneur] = "" Then Forms![RESULTADO].[caneur] = _
    Forms![RESULTADO].[canpts] / 166.386
  If [IVA] = 1 Then
    Forms![RESULTADO].[ivapts] = Forms![RESULTADO].[canpts] * 0.04
    Forms![RESULTADO].[ivaeur] = Forms![RESULTADO].[caneur] * 0.04
  End If
  If [IVA] = 2 Then
    Forms![RESULTADO].[ivapts] = Forms![RESULTADO].[canpts] * 0.07
    Forms![RESULTADO].[ivaeur] = Forms![RESULTADO].[caneur] * 0.07
  End If
  If [IVA] = 3 Then
    Forms![RESULTADO].[ivapts] = Forms![RESULTADO].[canpts] * 0.16
    Forms![RESULTADO].[ivaeur] = Forms![RESULTADO].[caneur] * 0.16
  End If
  Forms![RESULTADO].[pvpppts] = CDbf(Forms![RESULTADO].[canpts]) _
    + CDbf(Forms![RESULTADO].[ivapts])
  Forms![RESULTADO].[pvpeur] = CDbf(Forms![RESULTADO].[caneur]) _
    + CDbf(Forms![RESULTADO].[ivaeur])
End Sub
```

Bien, antes de continuar observemos:

- Debido a que el procedimiento lo llamaremos desde el formulario **“Cálculo del IVA”**, al referirnos a los campos del formulario **“RESULTADO”**: [canpts], [caneur], etc.

Debemos anteponer el prefijo: **Forms!**[RESULTADO]., a cada uno de los campos, ya que en caso contrario el programa no podría “encontrarlos”.

- La estructura del programa es sencilla (lo que “marea” es el prefijo **Forms!** [RESULTADO]., por todos sitios), ya que utilizamos únicamente la estructura de programación “**If – Then**”. Estúdialo con detalle, recuerda que [moneda] es el nombre del primer “grupo de opciones” de forma que si [moneda] = 1, significa que son “pesetas” y si [moneda] = 2 son “euros”. Exactamente lo mismo sucede con [iva] que puede valer 1 (4%), 2 (7%) o 3 (16%).
 - En las dos últimas líneas del programa aparece una nueva función incorporada: **Cdbl**. Utilizamos esta nueva función que convierte el contenido de un cuadro de texto, en número tipo **Double**. En los demás casos no nos hemos 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.
- Prueba exhaustivamente el funcionamiento del formulario “**Cálculo del IVA**”. Espero que te funcione correctamente.
- Vamos a intentar “mejorar” el programa:
- Observamos al ejecutar el programa (al hacer CLIC en el [Aceptar] del formulario “**Cálculo del IVA**”) que el enfoque del cursor queda situado en el campo **CANTIDAD / PESETAS**, por otro lado podemos escribir en cualquiera de los 6 campos del formulario RESULTADO. Para solucionar el problema haz lo siguiente:
 - Accede al formulario “RESULTADO” en “**Modo Diseño**”.
 - Selecciona los 6 campos (has de mantener pulsada una de las teclas de mayúsculas).
 - CLIC en el icono “**Propiedades**”
 - Cambia la propiedad:
Bloqueado = **Sí**
 - Graba de nuevo el formulario
 - Ejecuta y prueba el formulario: **Cálculo del IVA**
 - Si intentas borrar o cambiar cualquier campo del formulario “RESULTADO”, como podrás observar no podrás.
 - Pero continuamos con el problema de que el primer campo queda seleccionado y además podemos colocar el cursor en cualquiera de los campos de “RESULTADO” Ya que todo lo que aparece en el formulario “RESULTADO” es sólo “informativo”, haz lo siguiente:

- Vuelve a acceder a las “propiedades” de los 6 campos de “RESULTADO” y cambia la propiedad:
Activado = **No**
 - Si pruebas de nuevo el programa verás que hemos conseguido lo que queríamos.
- Otra “mejora” que podríamos hacer es la siguiente:
- Si abres el formulario “**Cálculo del IVA**” y en el campo “**CANTIDAD:**” no escribes nada y haces CLIC en [Aceptar], aparecerá como es lógico un mensaje de error. Para salir del “error” haz CLIC en [Terminar]
 - Para solucionar el problema haz lo siguiente:
 - “Edita” el programa que tenemos asociado al botón [Aceptar]:
Private Sub OK_Click()
.....
.....
.....
End Sub
- Y añade las siguientes líneas:
- ```
Private Sub OK_Click()
If IsNull([cantidad]) Then
 MsgBox (“Debes escribir un número en CANTIDAD”)
Else
 DoCmd.OpenForm “RESULTADO”

End If
End Sub
```
- Graba de nuevo el formulario y pruébalo. Espero que te funcione correctamente si haces CLIC en el [Aceptar] del formulario y no has escrito nada en el campo “CANTIDAD:”.
- Supongamos que estamos totalmente satisfechos con nuestro programa, ha quedado tan bonito que pretendemos venderlo, pero para ello nos encontramos con varios problemas:
- 1º) Queremos que aparezca nuestro **nombre**.
  - 2º) No nos interesa que “**modifiquen**” nuestro programa.
  - 3º) Nos gustaría que el programa se “**autoejecutara**” y no apareciera ni la Base de Datos “**IVA**”, ni cualquier referencia al **Access**.

Para solucionar estos inconvenientes, haz lo siguiente:

- Botón del Office  
Opciones de Access  
Base de datos actual

Título de la aplicación: **(escribe tu nombre y apellidos)**

Mostrar el formulario: **Cálculo del IVA**

Desactiva todas las opciones que aparecen por defecto en toda la ventana.

Activa la opción: **Ventanas superpuestas**

[Aceptar]  
[Sí]  
[Aceptar]

Cierra la base de datos **IVA** y vuélvela a abrir.

- Prueba “nuestro programa”
  - Observa que puedes hacer que se visualicen los dos formularios
  - Intenta cambiar alguna cosa .
- Resulta que ahora que ya hemos acabado y “protegido” nuestro programa, nos gustaría cambiar alguna cosa.

No hay problema, haz lo siguiente:

- Sal de nuestro programa: Menú Archivo – Salir
- Ejecuta el “Access”
- Localiza nuestra “**base de datos: IVA**”. Cuando la tengas seleccionada y antes de hacer CLIC en [Abrir], pulsa y mantén pulsada una de las dos teclas de mayúsculas. Con la tecla de mayúsculas pulsada haz CLIC en [Abrir], espera unos segundos antes de dejar de pulsar la tecla. Si todo va bien aparecerá la base de datos **IVA**, con todas sus “posibilidades”, dicho de otra forma: si mantenemos pulsada una de las dos teclas de maúsculas al abrir una base de datos, anulamos las posibles “**órdenes de Inicio**”.

r) Uno de los errores más frecuentes al programar en **VBA** es por culpa de los “tipos de datos”. Vamos a trabajar con una nueva base de datos para intentar “aclarar” el problema.

- Crea una nueva base de datos de nombre **Aclaremos**, en *TuCarpeta*.
- Crea un módulo en **Aclaremos** de nombre **Módulo1**
- Escribe en el **Módulo1** el siguiente programa:
 

```
Sub Problema1A()
 Dim x As Integer, y As Integer
 x = InputBox("Escribe un número")
 y = InputBox("Escribe otro número")
 MsgBox "La suma es = " & (x + y)
 MsgBox "El producto es = " & (x * y)
End Sub
```
- Ejecuta el programa desde la ventana **Inmediato**. En principio no hemos de observar ningún problema, siempre y cuando los dos números que introduzcamos sean **Integer**.
- Escribe en el **Módulo1** y a continuación del anterior, el siguiente programa:

```
Sub Problema1B()
 x = InputBox("Escribe un número")
 y = InputBox("Escribe otro número")
 MsgBox "La suma es = " & (x + y)
 MsgBox "El producto es = " & (x * y)
End Sub
```

- Ejecuta el programa **Problema1B** desde la ventana “Inmediato”:
  - A la primera pregunta, contesta escribiendo **2**.
  - A la segunda pregunta, contesta escribiendo **4**.
  - Si todo funciona correctamente, el primer resultado ha de ser: **La suma es = 24** y el segundo resultado **El producto es = 8**

Conclusión: multiplica bien pero no sabe sumar.

El problema se presenta porque en el **Problema1B**, no hemos declarado las variables **x** e **y**. Y el **InputBox** es una función que devuelve (por defecto) **String**. Es decir, los valores de **x** e **y** son textos.

El operador "+" en el caso de textos es equivalente al operador "&", es decir **concatena** los dos textos, por esta razón aparece **24**.

Por otro lado, el operador "\*" siempre multiplica números. El VB, al encontrarse un producto entre textos, convierte automáticamente (si tiene sentido) los dos textos a números; se dice que el **VB convierte implícitamente** los dos textos a números.

- Para verlo más claro, vuelve a ejecutar el **Problema1B** ...
  - A la primera pregunta, contesta escribiendo **Pepe**
  - A la segunda pregunta, contesta escribiendo **Paco**
  - Si todo funciona correctamente, el primer resultado será **PepePaco** y el segundo resultado dará el error nº 13: **No coinciden los tipos** de datos, es decir no puede convertir los dos textos a números y por lo tanto no puede multiplicarlos. Haz clic en [Finalizar] para salir del "error".
- Vuelve a ejecutar el **Problema1B**:
  - A la primera pregunta, contesta escribiendo **1,2**
  - A la segunda pregunta, contesta escribiendo **5,4**
  - Si todo funciona correctamente, el primer resultado será **1,25,4** y el segundo **6,48**. Es decir, VB convierte los dos textos a **Double** automáticamente y calcula correctamente su producto.

La conclusión que hemos de sacar de lo que hemos hecho es: **Es muy importante declarar las variables antes de utilizarlas.**

De todas formas es muy fácil, en un programa con muchas variables despistarse y llegar al fatídico **error nº 13: No coinciden los tipos**.

Vamos a ver un "truco" que nos permite descubrir el "tipo de datos" en cada momento ...

- Escribe en el **Módulo1** de la **B.D. Aclaremos** el siguiente programa:

```
Sub Problema1C()
 Dim x As Integer
 Dim y As Double
 x = InputBox("Escribe un número entero")
 MsgBox "El valor x = " & x & " es " & TypeName(x)
 y = InputBox("Escribe un número decimal")
 MsgBox "El valor y = " & y & " es " & TypeName(y)
 z = InputBox("Escribe lo que quieras")
 MsgBox "El valor z = " & z & " es " & TypeName(z)
End Sub
```

- Ejecuta el procedimiento **Problema1C**. Espero que esté más "claro" nuestro problema:
  - **TypeName(variable)** es una función incorporada en el VB, que nos devuelve el **tipo** de la variable.
  - Observa que en la variable **z** siempre aparecerá **String**, hayamos escrito o no un número.
- Existe una alternativa a la instrucción **Dim**, que nos permite declarar variables, y es la instrucción **Static**, aunque no es exactamente lo mismo ...
  - Escribe en el **Módulo1** el siguiente procedimiento:

```

Sub Problema1D()
 Dim n As Integer
 Dim s As Integer
 For i = 1 To 3
 n = InputBox("Escribe un entero")
 s = s + n
 Next
 MsgBox "La suma es = " & s
End Sub

```

- Ejecuta un par o tres de veces el **Problema1D**. En principio no ha de haber diferencia cada vez que ejecutes el programa: nos pregunta 3 números y nos da como resultado su **suma**. Observa de todas formas que el **contador suma: s=s+n**, no lo hemos inicializado a **cero**.
- Escribe en el **Módulo1** de la B.D. Aclaremos, el siguiente procedimiento:

```

Sub Problema1E()
 Dim n As Integer
 Static s As Integer
 For i = 1 To 3
 n = InputBox("Escribe un entero")
 s = s + n
 Next
 MsgBox "La suma es = " & s
End Sub

```

- Ejecuta un par o tres de veces el **Problema1E**. Observarás que la primera vez que ejecutas el programa, calcula correctamente la suma de los tres números introducidos; pero si vuelves a ejecutar el programa, la suma de los tres nuevos números se **acumula** a la suma anterior.

### Conclusión

Cada vez que se ejecuta una instrucción **Dim**, la variable correspondiente se inicializa (0 para numérico, vacío para String). En cambio una instrucción **Static**, no inicializa el valor de la variable: **conserva el valor anterior**. Dicho de otra forma, las dos instrucciones siguientes:

```

Static s As Integer
s = 0

```

son equivalentes a **Dim s As Integer**

Si repasas los programas del capítulo anterior, verás que en muchos casos no era necesario inicializar las variables, porque utilizamos **Dim**; pero en algún caso sí era necesario, por ejemplo en una variable que acumula un producto: es necesario inicializar la variable a **1**, ya que en caso contrario el **producto acumulado** siempre nos dará 0.

- Crea un formulario (de nombre **Formulario1**) en blanco, para la **B. D. Aclaremos**.
- Inserta en el **Formulario1**, cuatro cuadros de texto con las siguientes características:

Cuadro de Texto 1:

Etiqueta: **Número Entero**  
Propiedad Nombre: **x**

Cuadro de Texto 2:

Etiqueta: **Otro Entero**  
Propiedad Nombre: **y**

Cuadro de Texto 3:

Etiqueta: **Suma**  
Propiedad Nombre: **sum**

Cuadro de Texto 4:

Etiqueta: **Producto**

Propiedad Nombre: **prod**

- Inserta en el **Formulario1** un botón de comando con las siguientes características:

Etiqueta: **Borrar**

Propiedad Nombre: **Borrar**

- En el módulo del formulario1, escribe el siguiente procedimiento de evento:

```
Private Sub Borrar_Click()
```

```
 [x] = ""
```

```
 [y] = ""
```

```
 [sum] = ""
```

```
 [prod] = ""
```

```
End Sub
```

- Inserta en el **Formulario1** otro botón de comando con las siguientes características:

Propiedad Nombre: **Problema2A**

Propiedad Título: **Problema2A**

Asocia al botón el siguiente procedimiento de evento:

```
Private Sub Problema2A_Click()
```

```
 Dim a As Integer, b As Integer
```

```
 a = [x]
```

```
 b = [y]
```

```
 [sum] = a + b
```

```
 [prod] = a * b
```

```
End Sub
```

- Prueba el funcionamiento del [Problema2A].  
Si todo funciona correctamente no hay nada que decir.

- Inserta en el **Formulario1** otro botón de comando con las siguientes características:

Propiedad Nombre: **Problema2B**

Propiedad Título: **Problema2B**

Asocia al botón el siguiente programa:

```
Private Sub Problema2B_Click()
```

```
 [sum] = [x] + [y]
```

```
 [prod] = [x] * [y]
```

```
End Sub
```

- Prueba el funcionamiento del [Problema2B]. Si todo funciona correctamente llegarás a la conclusión que se nos presenta el mismo problema que en los **InputBox's**.

El problema es exactamente el mismo: un cuadro de texto, igual que un InputBox (por defecto) contiene un **texto**

La conclusión debería ser la misma: es muy importante utilizar variables que declaramos previamente según el "tipo" que nos interese.

De todas formas vamos a solucionar el problema de otra forma, que también podríamos utilizar para los **InputBox's** ...

- Inserta en el **Formulario1** otro botón de comando con las siguientes características:

Propiedad Nombre: **Problema2C**

Propiedad Título: **Problema2C**

Asocia al botón el siguiente código:

```
Private Sub Problema2C_Click()
```

```
 [sum] = CInt([x]) + CInt([y])
```

```
 [prod] = CInt([x]) * CInt([y])
```

### End Sub

- Prueba el programa anterior.

Observa que tenemos una alternativa al uso de variables “declaradas” y es el uso de funciones de **conversión**:

**CInt(argumento)** = convierte el “argumento” en número entero.

Si quieres consultar las diferentes funciones de conversión, accede a la ayuda de Visual Basic (no del Access), concretamente el tema: **Funciones de conversión de tipos**.

s) Hasta ahora nos hemos dedicado a programar en VBA sin utilizar “datos” de la base de datos subyacente a nuestros programas. Aunque en los últimos ejercicios hemos utilizado formularios, sólo los hemos utilizado como **soporte “visual”**.

Vamos a comenzar a trabajar con “datos” de nuestras bases de datos. Piensa, que en realidad la “programación en Visual Basic del Access”, ha de ser un instrumento para **mejorar la gestión** de nuestras bases de datos Access.

En algunos de los ejercicios anteriores hemos utilizado órdenes propias del Access, como **DoCmd.OpenForm “Triángulo”**; veremos en este apartado una forma muy fácil de estudiar este tipo de instrucciones.

- Crea una nueva base de datos de nombre **Access1** en *TuCarpeta*.
- Define en **Access1** una tabla con la siguiente estructura (basta que accedas a **Vista Diseño** haciendo clic con el botón derecho en la pestaña [Tabla1], grábala como **Tabla1**):

| Nombre del campo | Tipo de datos |
|------------------|---------------|
| NumSocio         | Texto         |
| NombreApell      | Texto         |
| NIF              | Texto         |
| Dirección        | Texto         |
| Teléfono         | Texto         |

- Observa que el primer campo (NumSocio) es la **Clave Principal**, por defecto (aparece una pequeña llave a la izquierda del nombre)

La idea es disponer de una tabla que controle los “socios” de un club.

- “Abre” la tabla **Tabla1** (es decir, **Vista Hoja de Datos**) e introduce los siguientes registros:
 

|                                  |                                           |
|----------------------------------|-------------------------------------------|
| NumSocio: <b>001A</b>            | NumSocio: <b>002A</b>                     |
| NombreApell: <b>Pepto Grillo</b> | NombreApell: <b>Francisca Sánchez</b>     |
| NIF: <b>4.352.793V</b>           | NIF: <b>7.932.001H</b>                    |
| Dirección:                       | Dirección: <b>Las Palmeras 51, 1º, 3ª</b> |
| Teléfono: <b>527 31 42</b>       | Teléfono: <b>397 52 71</b>                |
- Crea un **Formulario** (Cinta de Opciones: Crear – Formularios - Formulario) para la tabla **Tabla1**, y grábalo con el nombre **Socios**.
- Utilizando el formulario anterior introduce el siguiente registro:
 

|                                   |
|-----------------------------------|
| NumSocio: <b>003A</b>             |
| NombreApell: <b>Paquito Pérez</b> |
| NIF:                              |
| Dirección:                        |
| Teléfono: <b>493 33 55</b>        |



El problema que nos planteamos es el siguiente:

1º) En el formulario nos interesa visualizar el número de socio y el nombre y apellidos del último registro introducido.

2º) En el formulario nos interesa un control que llamaremos “Nivel Confianza”, y que según el valor que escribamos en dicho control nos interesa que suceda una cosa u otra, de la siguiente forma:

- Si el “Nivel de Confianza” es 1, después de introducir el nombre y apellidos del socio, el cursor se coloca automáticamente en el campo **Teléfono**  
Dicho de otra forma: para un nivel igual a 1 no hemos de “rellenar” el **Nif** ni la **Dirección**.  
El socio **003A** sería un ejemplo de nivel 1.
- Si el “Nivel de Confianza” es 2, no es necesario introducir la **Dirección**. Es decir, después de introducir el **NIF**, el cursor debería situarse automáticamente en el **Teléfono**.  
El socio **001A** es un socio de nivel 2.
- Si el “Nivel de Confianza” no es ni 1 ni 2 no sucede nada especial, es decir, en principio hemos de introducir todos los datos del socio.  
El socio **002A** sería un ejemplo de nivel “normal”.

Vamos a ver si lo conseguimos:

- Sitúate en la pantalla de **diseño** del formulario **Socios** e inserta tres cuadros de texto (en la parte Detalle del formulario) con las siguientes propiedades:

- Cuadro 1:  
Etiqueta: **Socio Anterior**  
Propiedades:  
Nombre: **SocioA**  
Activado: **No**
- Cuadro 2:  
Etiqueta: (ninguna)  
Propiedades:  
Nombre: **Nom**  
Activado: **No**
- Cuadro 3:  
Etiqueta: **Nivel de Confianza**  
Propiedades:  
Nombre: **Nivel**  
Índice de tabulación: **1**

- Cambia la distribución del formulario hasta que te quede aproximadamente de la siguiente forma (todo ha de estar en la parte **Detalle** del formulario, el encabezado que aparecía por defecto bórralo):

- Nos interesa que en los controles **SocioA** y **Nom** aparezcan el **NumSocio** y el **NombreApell** del último registro, para ello, entre otras cosas, necesitamos “codificar” unas cuantas órdenes propias del Access:
  - “Último registro” para poder **recoger** los valores últimos de **NumSocio** y **NombreApell**
  - “Nuevo registro” para poder situarnos en un nuevo socio.
- Sitúate en la pantalla de diseño del formulario **Socios** y selecciona el formulario, es decir:



Sabremos que hemos seleccionado el formulario, si aparece un punto negro:



- Con el cursor del ratón en el “punto negro” pulsa el botón derecho del ratón y accede a sus **propiedades**.
- Sitúa el cursor de escritura en la propiedad **Al Abrir** y clic en [...], que aparece a la derecha.
- Escoge la opción “**Generador de código**” y [Aceptar]
- Acabamos de situarnos en el **módulo del formulario Socios**, para escribir el procedimiento de evento **Form\_Open**, es decir, programa que se ejecutará al abrir el formulario (Form\_Open).
- Escribe el siguiente programa:

```
Private Sub Form_Open(Cancel As Integer)
 Dim x As String, y As String
 DoCmd.GoToRecord , "", acLast
 x = [NumSocio]: y = [NombreApell]
 DoCmd.GoToRecord , "", acNewRec
 [SocioA] = x
 [Nom] = y
End Sub
```

Observemos el programa:

- Al abrir el formulario (Form\_Open)
- Declaramos dos variables x, y de tipo String
- Nos situamos en el último registro (acLast)
- Asignamos a x e y los valores de los campos **NumSocio** y **NombreApell**, que serán los últimos, porque estamos situados en el último registro.
- Nos situamos en un nuevo registro (acNewRec)
- Asignamos a los controles **SocioA** y **Nom**, los valores de las variables x e y.

Vamos a ver si funciona:

- Ejecuta el formulario "Socios"  
Si todo funciona correctamente, en los dos controles superiores aparecerá: **003A Paquito Pérez**
- Escribe el nuevo registro:  
NumSocio: **004A**  
Nivel de Confianza: **1** (como es lógico no sucederá nada especial)  
NombreApell: **Herminia López**  
NIF: no escribas nada  
Dirección: no escribas nada  
Teléfono: **413 55 32** Pulsa [Tab] o [Return] para pasar al siguiente campo, que corresponderá a un nuevo registro.
- Supongo que te das cuenta del problema: estamos en un nuevo registro pero los datos que aparecen en los dos primeros controles del formulario, no corresponden al último registro.  
Vamos a intentar solucionar el problema:
  - Sitúate en la pantalla de diseño del formulario **Socios**
  - Inserta un botón de comando con las siguientes propiedades:  
Nombre: **NRegistro**  
Título: **Otro**
  - Accede al módulo del formulario y escribe el siguiente procedimiento de evento:
 

```
Private Sub NRegistro_Click()
 Dim x As String, y As String
 DoCmd.GoToRecord , "", acLast
 x = [NumSocio]: y = [NombreApell]
 DoCmd.GoToRecord , "", acNewRec
 [SocioA] = x
 [Nom] = y
End Sub
```

Como puedes observar se trata de exactamente el mismo programa **Form\_Open**
- Vuelve a la pantalla de diseño del formulario **Socios**
  - Selecciona el formulario
  - Accede a las propiedades del formulario
  - Cambia las propiedades siguientes:  
Selectores de registro: No  
Botones de desplazamiento: No  
Separadores de registro: No
- Abre el formulario **Socios** e inventa un par o tres de registros para observar que los dos primeros controles del formulario nos muestran los datos del "último registro".  
Espero que te funcione.

Seguramente habrás observado una incomodidad: al hacer clic en [Otro], el cursor no queda situado (se llama **foco**), en el campo **NumSocio**.

Corrige el procedimiento **NRegistro\_Click** de forma que nos quede:

```

Private Sub NRegistro_Click()
 Dim x As String, y As String
 DoCmd.GoToRecord , "", acLast
 x = [NumSocio]: y = [NombreApell]
 DoCmd.GoToRecord , "", acNewRec
 [SocioA] = x
 [Nom] = y
 NumSocio.SetFocus
End Sub

```

Es decir, añadimos una nueva instrucción: **NumSocio.SetFocus**, que forzará al cursor a colocarse (SetFocus), en el campo **NumSocio**.

Recuerda que es importante ir grabando todos los cambios que vamos haciendo.

- Vuelve a “ejecutar” el formulario **Socios** e inventa unos cuantos “socios”.
- Vamos a resolver la segunda parte del problema que nos habíamos propuesto (el del Nivel de Confianza), que será muy fácil de solucionar gracias a la nueva instrucción: **NombreCampo.SetFocus**: coloca el foco en “NombreCampo”

- Sitúate en la pantalla de diseño del formulario
- Selecciona el campo **NombreApell**
- Coloca el cursor en su propiedad **Al salir**
- Accede al procedimiento de evento correspondiente (botón [...])
- Escribe el siguiente procedimiento:

```

Private Sub NombreApell_Exit(Cancel As Integer)
 If [Nivel] = "1" Then
 Teléfono.SetFocus
 End If
End Sub

```

Observa:

“Al salir del campo **NombreApell** (NombreApell\_Exit)”

“Si en el control **Nivel** hay un **1** (If [Nivel]=“1”)”

“Entonces sitúate en el campo **Teléfono** (Then Teléfono.SetFocus)”

Es decir, que en la práctica nos saltaremos los campos **NIF** y **Dirección**.

- Sitúate al final del módulo del formulario y escribe:

```

Private Sub NIF_Exit(Cancel As Integer)
 If [Nivel] = "2" Then
 Teléfono.SetFocus
 End If
End Sub

```

- Graba todos los cambios que hemos hecho.

- Sólo falta probar lo que hemos “programado”. Ejecuta el formulario e inventa unos cuantos registros con “niveles de confianza” distintos.

La conclusión que deberíamos sacar de nuestro trabajo con la **B. D. Access1**, es que podemos manipular “datos” (hasta cierto punto, ya veremos en su momento, cuando estudiemos la programación **DAO**, la verdadera “manipulación” de los datos Access), de la misma forma que hacíamos con los controles independientes de un formulario: “**Nombre del campo**” siempre y cuando lo escribamos entre corchetes.

## Para saber más

### Evento

Es una acción del usuario o del sistema que afecta a un objeto (control) y provoca la ejecución del código correspondiente.

**BOTÓN1\_Click** : al hacer click en el botón de nombre "BOTÓN1"

**Form\_Open**: al abrir el formulario

**NIF\_Exit**: al salir del control NIF

Ya veremos en el siguiente ejercicio otros eventos.

### Módulos y Procedimientos

El **módulo** es la estructura del Access que nos permite introducir código VBA. Hay de dos tipos:

#### - Módulos estándar:

- Accedemos a ellos a través de la pantalla inicial de la BD, **objeto: Módulos** y click en [Nuevo] o [Diseño]
- En dichos módulos escribimos los procedimientos "Sub" y las funciones.

#### - Módulos de formulario y informe:

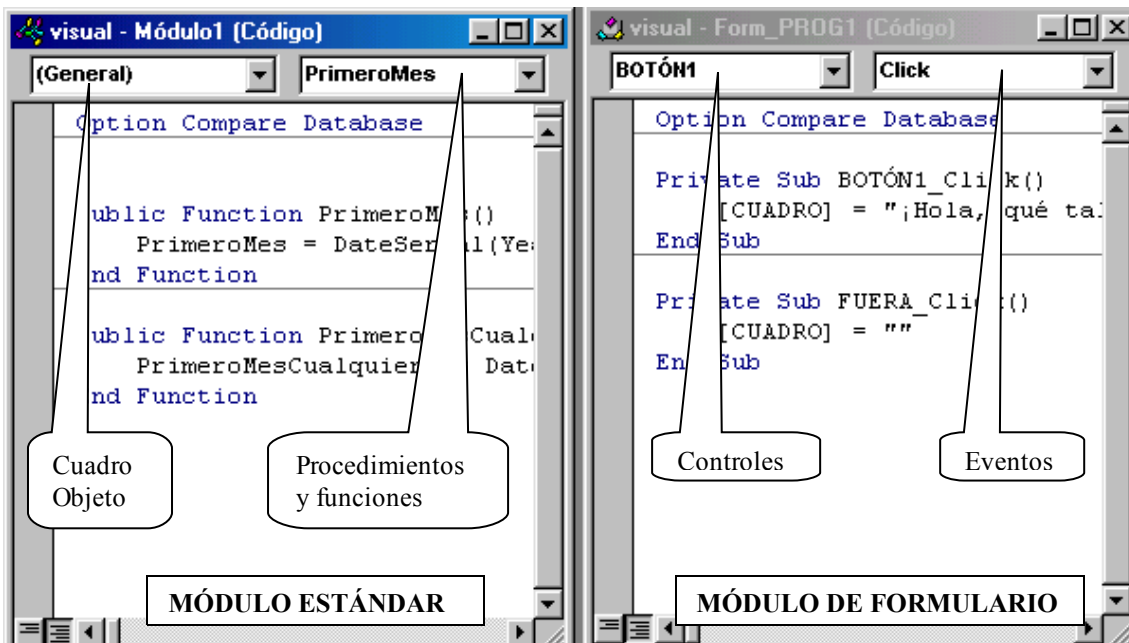
- Accedemos a ellos desde la pantalla de diseño del formulario o informe, haciendo click en el icono **Ver Código**:  
(Cinta de Opciones: Herramientas de diseño de formulario)



O con un control seleccionado, haciendo click en el icono **"Generar"** y seleccionando la opción **"Generador de Código"**

- En estos módulos escribimos los **"procedimientos de evento"**, es decir programas que se ejecutan según el evento del control correspondiente.

En todos los casos un módulo presenta dos partes bien diferenciadas:



### Ámbito de un procedimiento:

Un procedimiento puede ser:

- **Private:**  
Indica que el procedimiento sólo es accesible para los procedimientos restantes del módulo en el que se encuentra.
- **Public:**  
Indica que el procedimiento es accesible para todos los procedimientos de todos los módulos.
- Además de "Private" o "Public", un procedimiento puede ser **Static:**  
Indica que todas las variables locales del procedimiento mantienen su valor entre llamadas.

### Declaración de Variables

Existen dos maneras de declarar variables: implícita y explícitamente

#### Declaración Implícita

Cuando VBA encuentra una variable que no ha sido declarada explícitamente, la declara de manera implícita utilizando el tipo que corresponda al valor que le es asignado.

#### Declaración Explícita

La variable se declara antes de ser utilizada.

Las declaraciones explícitas de variables pueden realizarse en la sección de declaraciones de los módulos o en el cuerpo de un procedimiento o de una función.

La declaración se lleva a cabo mediante una de las cuatro instrucciones siguientes, utilizando la misma sintaxis:

- A nivel de procedimiento o función:  
Dim Total As Integer  
Static Cont As Integer
- A nivel de módulo:  
Private Nombre As String  
Public FechaNacimiento As Date

Cada vez que se ejecuta la instrucción **Dim**, la variable se reinicializa (0 para numérico, vacío para String). Si es necesario conservar el valor anterior, se debe utilizar **Static** en lugar de **Dim**.

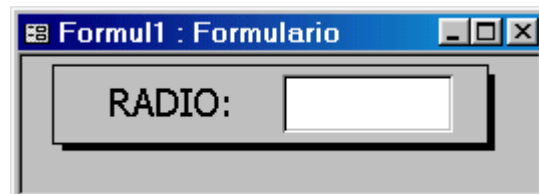
Recuerda que podemos obligar a la declaración explícita de variables utilizando la instrucción **Option Explicit**, en la sección de declaraciones de cada módulo.

## Autoevaluación 2

1) Haz un "programa" que nos permita calcular el área y la longitud de una circunferencia a partir del radio, pero de forma **visual**.

De la siguiente forma:

- Crea una base de datos de nombre **Eval2A**
- Crea en "Eval2A" un formulario de nombre **Formul1**, con los siguientes controles y aspecto aproximado:



- Crea en "Eval2A" otro formulario de nombre **Formul2**, con los siguientes controles y aspecto aproximado:



- Crea las siguientes **opciones de inicio**:
  - Mostrar Formulario: **Formul1**
  - **Desactiva el resto de opciones**

2) Repite el **Prog12** del capítulo anterior pero de forma visual

El **Prog12** nos pedía un número y nos daba como resultado la tabla de multiplicar del número introducido:

```

Sub Prog12()
 Dim num As Integer, i As Integer
 Dim salida As String
 salida = ""
 num = InputBox("Tabla del número")
 For i = 1 To 9
 salida = salida & num & " * " & i & " = " & num * i & vbCrLf
 Next
 MsgBox salida
End Sub

```

Una solución podría ser la siguiente:

Crea una base de datos de nombre **Eval2B**, con un formulario de nombre **Tabla Multiplicar**:

Parece muy complicado de hacer, pero si procedes de la siguiente forma es muy fácil:

a) Haz el formulario:

b) Selecciona la primera multiplicación, es decir los 3 cuadros de texto y las dos etiquetas (X, =) a la derecha del primer cuadro de texto (Tabla del)

- Clic en el icono **Copiar**
- Clic en el icono **Pegar**, 8 veces.

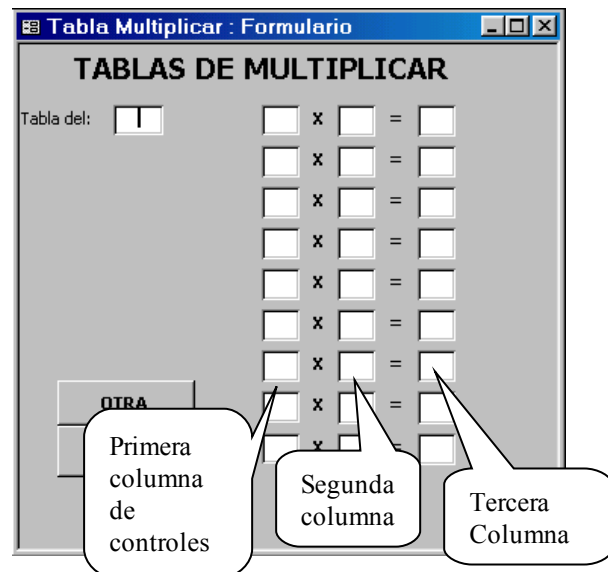


**Nombres de los controles:**

El primer cuadro: **num**

Primera columna de controles:

- a0
- a1
- a2
- a3
- a4
- a5
- a6
- a7
- a8



Segunda columna: b0, b1, b2, b3, b4, b5, b6, b7, b8

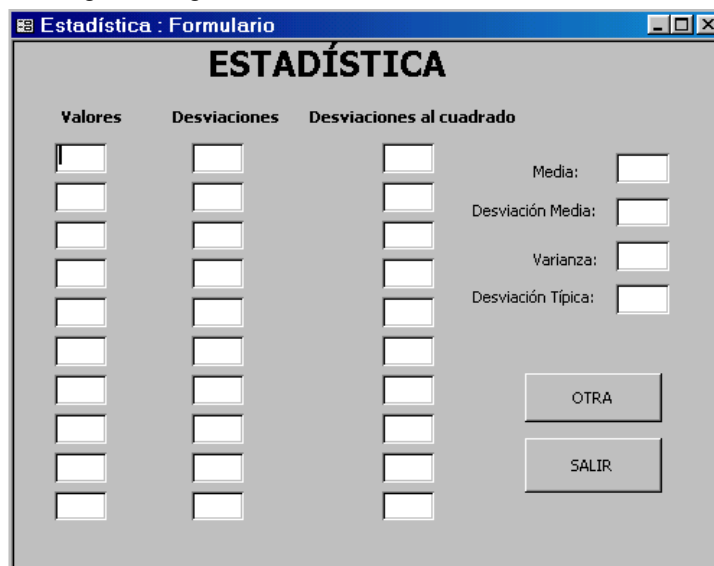
Tercera columna: c0, c1, c2, c3, c4, c5, c6, c7, c8.

- El código es muy sencillo si utilizas 3 matrices: **a(0 To 8)**, **b(0 To 8)**, **c(0 To 8)**
  - En la primera matriz asignas: 1, 2, 3, 4, 5, 6, 7, 8, 9.
  - En la segunda matriz asignas a cada elemento el número que tenemos en **[num]**
  - En la tercera matriz asignas el producto del elemento correspondiente de la primera y segunda matriz.
  - Por último asignas a nuestros campos del formulario, los valores de las matrices: **[a0] = a(0) : [a1] = a(1) : ....**
- Introduce las mismas opciones de **inicio** que en el ejercicio anterior.

3) Repite el **Prog15** del capítulo anterior, pero de forma **visual**.

Procede de la siguiente forma:

- Crea una base de datos de nombre **Eval2C** y crea un formulario de nombre **Estadística**, aproximadamente igual al siguiente:



- Crea un módulo de nombre **Módulo1**, que contenga la función **Media10**, ya hecha en el capítulo 1 (autoevaluación1)
- Crea las mismas opciones de **Inicio** que los ejercicios anteriores

4) Haz un programa **visual** que sirva para simplificar fracciones.

- Dada una fracción **x/y** el verdadero problema para poder simplificarla es calcular el máximo común divisor de x e y, ya que:

$$\frac{x / \text{MCD}(x,y)}{y / \text{MCD}(x,y)} \quad \text{nos dará la fracción simplificada}$$

- Deberá crear una función que calcule el máximo común divisor de dos números (MCD)

La mejor forma de “programar” el MCD de dos números es utilizar el **método de Euclides**. Recordemos el método de “Euclides” con un ejemplo:

1º) Dividimos el mayor entre el menor:

48 entre 36 resulta: Cociente =1 y Resto=12

2º) Dividimos el divisor entre el resto:

36 entre 12 resulta: Cociente = 3 y Resto =0

3º) Continuamos dividiendo de esta última forma: **divisor entre resto**, hasta que la división sea exacta.

4º) El **MCD** es el último resto distinto de cero (el divisor que nos da la división exacta. Es decir, en nuestro ejemplo: **MCD(48, 36) = 12**

La “function” correspondiente podría ser la siguiente:

```

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

```

Observa:

- **x Mod y** es un operador de Visual Basic que nos da el resto de la división entera entre x e y
- Observa la forma de conseguir que la variable “a” corresponda al número mayor:  
**If a < b Then**

```

aux = a
a = b
b = aux
End If

```

Utilizamos lo que se llama una variable auxiliar (**aux**)

- Crea una base de datos de nombre **Eval2D**, con un formulario de nombre **Simplificación**, con el siguiente aspecto y contenido:

- Crea en la B.D. Eval2D, un módulo de nombre **Módulo1** que contenga la función **MCD** anterior.
- Considera las opciones de **Inicio**, que hemos hecho en los ejercicios anteriores.

El resto del programa deberías hacerlo tú.

#### 5) Crea una base de datos de nombre **Eval2E**

- Crea en **Eval2E** una tabla de nombre **Tabla1** con la siguiente estructura:

|        |                                |
|--------|--------------------------------|
| Campo1 | Autonumérico y Clave Principal |
| Campo2 | Texto                          |
| Campo3 | Numérico                       |
| Campo4 | Texto                          |
| Campo5 | Texto                          |
| Campo6 | Texto                          |

- Crea un autoformulario para la **Tabla1** de nombre **Formul1**
- Inserta en el **Formul1**, un cuadro de texto independiente con las siguientes características:
  - Etiqueta: ninguna
  - Propiedades: Nombre= Campo7
  - Activado: No
- Crea un procedimiento de evento "Al salir del Campo3", que funcione de la siguiente forma:
  - Si el número que escribimos en **Campo3** es mayor que 1000:  
En el Campo7 aparece "Demasiado Grande"

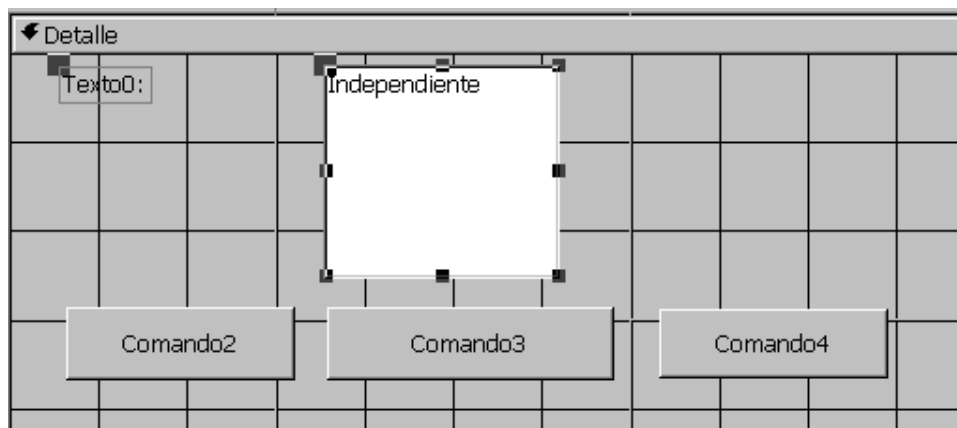
- En el Campo4 aparece "Mayor que 1000"  
El cursor queda situado en el Campo6 (es decir, saltamos el Campo5)
- Si el número que escribimos en Campo3, no es mayor que 1000:
  - En el Campo7 aparece "Vale"
  - En el Campo4 aparece "No es mayor que 1000"
  - El cursor se sitúa en Campo5
- Crea otro procedimiento de evento para el **Formul1** (Al cerrar el Formul1), que genere una ventana con el mensaje: "**Atención, se va a cerrar el formulario**".

## 3

## Formularios y Controles

- a) Crea una nueva base de datos Access en *Tu Carpeta* de nombre **CONTROLES**.

Crea en la B.D **CONTROLES** un formulario en blanco, con el siguiente contenido:



- Borra la etiqueta del cuadro de texto.
- b) Tenemos un formulario con un cuadro de texto y tres botones.
- Graba el formulario con el nombre **controles1** y accede al “módulo del formulario”, haciendo click en el icono “Ver Código”:

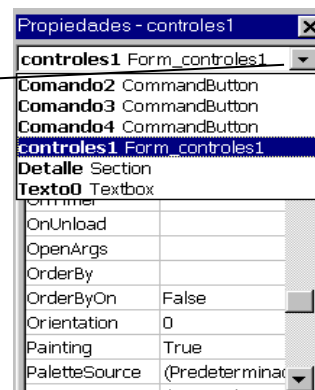


- Acabamos de situarnos en el **Editor de Visual Basic**, deberíamos tener a la vista:
  - \* La ventana de código: **CONTROLES – Form\_controles1 (Código)**
  - \* **La ventana de propiedades**. Si no la tienes a la vista deberás hacer: Menú Ver – Ventana Propiedades o pulsar la tecla [F4]
  - \* La ventana **Explorador de proyectos**. Si no la tienes a la vista deberás hacer: Menú Ver – Explorador de proyectos.

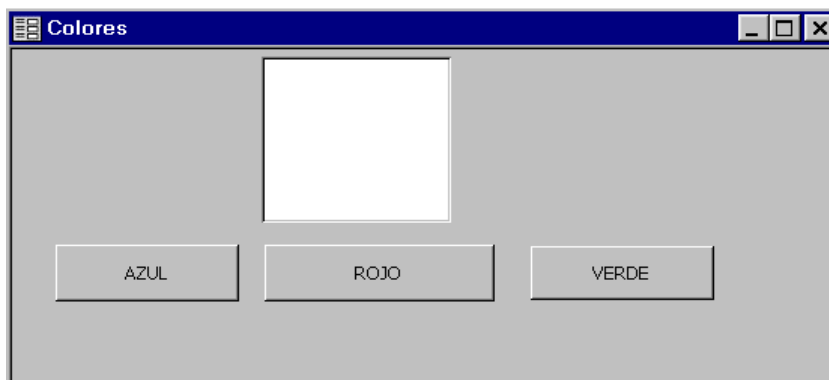
- Sitúate en la **Ventana de Propiedades** y selecciona del cuadro:

el formulario, es decir:  
click en **controles1 Form\_controles1**

Automáticamente visualizarás en dicha ventana las propiedades del objeto seleccionado, en nuestro caso del formulario **controles1**



- c) Selecciona en la **Ventana de Propiedades** el control **TextBox** (cuadro de texto)
- Localiza la propiedad **Name**, borra el texto que aparece por defecto y escribe en su lugar **txtCaja**
  - Selecciona en la **Ventana de Propiedades** el control **CommandButton**, correspondiente al primer botón que has colocado en el formulario. Cambia las siguientes propiedades:  
**Name:** cmdAzul  
**Caption:** AZUL
  - Selecciona el segundo CommandButton y cambia las siguientes propiedades:  
**Name:** cmdRojo  
**Caption:** ROJO
  - Cambia las siguientes propiedades del último botón:  
**Name:** cmdVerde  
**Caption:** VERDE
- d) Selecciona en la “Ventana de Propiedades” el formulario: **controles1 Form\_controles1**, y cambia las siguientes propiedades:  
**Caption:** Colores  
**Dividing Lines:** False  
**NavigationButtons:** False  
**RecordSelectors:** False
- e) Vuelve al **Microsoft Access**, graba de nuevo el formulario (con el mismo nombre **controles1**) y ejecuta el formulario. Si todo funciona correctamente aparecerá:



- f) Vuelve a la pantalla de diseño del formulario y al “Editor de Visual Basic”.
- Disponemos de un formulario de nombre (Name) **controles1** con las propiedades:  
**Caption:** Colores (texto que aparece en la barra de títulos del **form**)  
**DividingLines:** False (no aparece la raya vertical ni horizontal que delimita el form)  
**NavigationButtons:** False (no aparecen los controles de navegación por los registros)  
**RecordSelectors:** False (no aparece el selector de registro)

Y los siguientes controles:

- Un **TextBox** (cuadro de texto) de Name: **txtCaja**
- Tres **CommandButtons** (botones de comando) con:
  - \* Name= cmdAzul Caption= AZUL
  - \* Name= cmdRojo Caption= ROJO
  - \* Name= cmdVerde Caption= VERDE
- Pretendemos conseguir lo siguiente:
  - \* Al hacer click en [AZUL], es decir **cmdAzul\_Click()**, queremos que el cuadro de texto **txtCaja**, tome el color AZUL, es decir: **txtCaja.BackColor = vbBlue**  
Observa: **NombreControl.Propiedad = Valor**  
La propiedad responsable del color de un objeto es **BackColor**, **vbBlue** es una constante ya definida en **VBA** que representa el color azul.
  - \* Sitúate en la ventana **CONTROLES – Form\_controles1 (Código)** y selecciona en el cuadro combinado de controles (Objetos) el **cmdAzul**, automáticamente nos aparece:

```
Private Sub cmdAzul_Click()
```

```
End Sub
```

Es decir, el procedimiento de evento **Click** del CommandButton (aparece por defecto el evento **Click**, porque es el más usual en un botón).

- \* Escribe entre las líneas:

```
Private Sub cmdAzul_Click()
```

```
Y
```

```
End Sub
```

La siguiente línea de código: **txtCaja.BackColor = vbBlue**

- Accede al procedimiento de evento **Click** del CommandButton **cmdRojo**, es decir, selecciona el objeto **cmdRojo** y escribe:

```
Private Sub cmdRojo_Click()
```

```
txtCaja.BackColor = vbRed
```

```
End Sub
```

- Haz lo mismo para el **cmdVerde**, para conseguir el siguiente **procedimiento de Evento**:

```
Private Sub cmdVerde_Click()
```

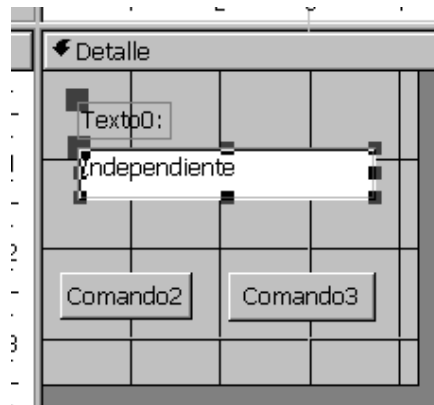
```
txtCaja.BackColor = vbGreen
```

```
End Sub
```

- g) Vuelve al “**Microsoft Access**”, graba de nuevo el formulario (con el mismo nombre) y ejecútalo.

Espero que te funcione. Está claro que **vbRed** y **vbGreen** son constantes de VB, que representan los colores rojo y verde respectivamente.

- h) Haz un nuevo formulario para la base de datos **CONTROLES**, de nombre **controles2** con el siguiente contenido:



- Vamos a cambiar una serie de propiedades, pero a diferencia del formulario anterior, se trata de hacerlo desde la pantalla de diseño del formulario...

- \* Selecciona el formulario (click en el botón correspondiente)
- \* Click en el icono “**Hoja de Propiedades**”
- \* Cambia las siguientes propiedades:
  - Título (Caption) = Contraseña Secreta
  - Selectores de Registro (RecordSelectors) = No
  - Botones de Desplazamiento (NavigationButtons) = No
  - Separadores de registro (DividingLines) = No
- \* Selecciona la **Sección: Detalle** del formulario, accede a sus propiedades y cambia el valor de:
  - Color del fondo (BackColor) = verde oscuro (32768 o #008000)
- \* Selecciona el **cuadro de texto (TextBox)** y cambia las propiedades:
  - Nombre (Name) = txtContraseña
  - Máscara de entrada (InputMask) = Accede al asistente, haciendo click en [...] y selecciona **Contraseña**
  - Cambia el contenido de su etiqueta por: **Escribe la Contraseña**
- \* Selecciona el botón de la izquierda (CommandButton) y cambia las propiedades:
  - Nombre (Name) = cmdAceptar
  - Título (Caption) = &Aceptar
  - Observa que en el botón aparece **Aceptar**. La utilidad de lo que acabamos de hacer está, en que al ejecutar el formulario, en lugar de activar el botón haciendo click con el ratón, bastará pulsar las teclas [ALT][A]
- \* Selecciona el botón de la derecha y cambia las propiedades:
  - Nombre (Name) = cmdCancelar
  - Título (Caption) = &Cancelar

- Graba el formulario con el mismo nombre **controles2**
- Accede al módulo del formulario **controles2** y contrasta las propiedades en la “ventana de propiedades”, que tenemos en el Editor de Visual Basic, con las propiedades que hemos cambiado desde la pantalla de diseño del formulario. Está claro que son las mismas, pero en el **Editor de Visual Basic** tenemos la versión en inglés.

Aunque no nos guste, es más importante ésta última, ya que en nuestros procedimientos, si deseamos cambiar una propiedad, como sucedía en **controles1**, necesariamente hemos de utilizar la “versión inglesa” de la propiedad correspondiente.

- Volvamos a “nuestro programa”...  
Resulta que la contraseña correcta es **PEPITO** o **pepito**



Accede al módulo del formulario **controles2** y escribe los siguientes procedimientos de evento:

```
Private Sub cmdAceptar_Click()
 If UCase([txtContraseña]) <> "PEPITO" Then
 MsgBox "No has acertado", vbCritical
 cmdCancelar_Click
 Else
 MsgBox "Muy bien, es correcto", vbExclamation
 cmdCancelar_Click
 End If
End Sub
```

```
Private Sub cmdCancelar_Click()
 txtContraseña = ""
 txtContraseña.SetFocus
End Sub
```

- Vuelve al “**Microsoft Access**”. Graba de nuevo el formulario y ejecútalo. Pruébalo, escribiendo **PEPITO**, **pepito** y cualquier otra cosa.

Veamos:

- \* **Ucase** es una función incorporada en el VB que convierte a mayúsculas el texto que escribimos entre paréntesis (en nuestro caso el contenido del cuadro de texto **txtContraseña**)
- \* Observa también que en el ejercicio anterior escribíamos el cuadro de texto entre corchetes, está claro que no son necesarios.
- \* Es conveniente que pruebes el uso de las teclas [ALT][A y [ALT][C]

i) Vamos a ver en este apartado un “evento” que no habíamos visto ...

- Crea en la base de datos **CONTROLES** otro formulario (controles3), con un único cuadro de texto.
- Elimina la etiqueta del cuadro de texto, y escribe como propiedad **Name**: txtColorines
- Graba el formulario con el nombre **controles3**
- Accede al módulo del formulario **controles3** y
  - \* Selecciona el **Objeto: Detalle** y el **Evento: MouseMove**

\* Escribe el siguiente procedimiento de evento:

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

- Vuelve al “Microsoft Access” y ejecuta el formulario.  
Si todo funciona correctamente, al mover el ratón por la sección detalle del formulario (Detalle\_MouseMove), el color del cuadro de texto cambia aleatoriamente.
  - \* **RGB** (argumento1, argumento2, argumento3) es una función incorporada en el VBA, que nos da como resultado un color:
    - Argumento1: componente rojo (número del 0 al 255)
    - Argumento2: componente verde (número del 0 al 255)

Argumento3: componente azul (número del 0 al 255)

- \* **Rnd**, es otra función incorporada en VBA, que devuelve un número aleatorio entre 0 y 1
- \* **Rnd\*256**: devolverá un número aleatorio entre 0 y 255

j) Vamos a trabajar con otro **evento**, que no depende del usuario sino del sistema...

- Crea en la base de datos **CONTROLES**, otro formulario con una etiqueta que contenga la frase **“Hola que tal”** y un botón de comando.  
Graba el formulario con el nombre **controles4**

- Accede al módulo del formulario anterior y cambia las siguientes propiedades:

```
* Form_controles4
Caption= El Evento Timer
RecordSelectors= False
NavigationButtons= False
DividingLines= False
TimerInterval= 1000
```

Con la propiedad **TimerInterval** del formulario a **1000**, acabamos de crear un cronómetro, de forma que cada 1000 milisegundos, es decir cada segundo, se ejecute el **evento Timer** del formulario (que aún no hemos programado)

```
* Etiqueta (Label)
Name= lblSaludo
* CommandButton
Name= cmdCambia
Caption= Otro Color
```

- Vamos a escribir los procedimientos de evento:

Al hacer click en el botón, nos interesa que cambie el color del formulario (en realidad, de la Sección Detalle del formulario).

Haz lo siguiente:

Selecciona en la ventana **CONTROLES – Form\_controles4 (Código)**, el objeto **cmdCambia**

Escribe:

```
Private Sub cmdCambia_Click()
Detalle.BackColor = RGB(Rnd * 256, Rnd * 256, Rnd * 256)
End Sub
```

\* Nos gustaría que la frase **“Hola que tal”** parpadee cada segundo: ya hemos definido un cronómetro a 1 seg, nos falta “programar” el evento correspondiente.

Haz lo siguiente:

Selecciona en la ventana de código el **objeto: Form** y el **evento: Timer**

Escribe:

```
Private Sub Form_Timer()
lblSaludo.Visible = Not (lblSaludo.Visible)
End Sub
```

**“Visible”** es una propiedad común a muchos controles y su funcionamiento es:

NombreControl.Visible = False (el control desaparece)

NombreControl.Visible = True (el control aparece)

**“Not”** es una función incorporada en el VB, que funciona de la siguiente forma:

Not(True) = False  
Not(False) = True

- Vuelve al “Microsoft Access”. Graba de nuevo el form **controles4** y ejecútalo. Espero que te guste.
- Habrás observado que al cambiar de color, aparecen dos barras, una horizontal y otra vertical de color gris que no cambian de color; dichas barras corresponden a las “barras de desplazamiento vertical y horizontal” del formulario. Si no te gustan, cambia la siguiente propiedad del form: **ScrollBars = 0**
- Si quieres un “parpadeo” diferente del mensaje **“Hola que tal”**, cambia el valor de la propiedad **TimerInterval** a 100, por ejemplo.

k) Vamos a trabajar con otro **evento**, ahora de un **TextBox**

- Crea en la base de datos **CONTROLES** otro formulario con dos cuadros de texto y un botón

Coloca como etiqueta del primer cuadro de texto: **“Escribe un número”** y como etiqueta del segundo: **“El factorial es:”**

Graba el formulario con el nombre **controles5**

- Accede al módulo del formulario anterior y cambia las siguientes propiedades:

**\* Form\_controles5**

DividingLines: False  
NavigationButton: False  
RecordSelectors: False  
ScrollBars: 0

**\* TextBox1**

Name: txtFactorial

**\* TextBox0**

Name: txtNumero  
EnterKeyBehavior: True

**\* CommandButton**

Name: cmdSalir  
Caption: Salir

- Vamos a escribir el código.

Como habrás adivinado se trata de calcular el factorial de un número. Recordemos que el factorial de un número, por ejemplo 5 es:  $1*2*3*4*5 = 120$ , es decir es el producto del número por los sucesivos números anteriores hasta llegar a la unidad.

Por lo tanto, el código para calcular el factorial podría ser:

```
factorial =1
For i=1 To número
 Factorial = factorial*i
Next
```

La variable “factorial” nos daría lo que buscamos.

Vamos a utilizar **una función** para calcular el factorial, es decir:

Desde el Editor de Visual Basic

- Menú Insertar – Módulo
- Menú Insertar – Procedimiento  
Nombre: Factorial  
Función  
Público
- Escribe:

```
Public Function Factorial(num As Long) As Long
 Dim i As Long
 Factorial = 1
 For i = 1 To num
 Factorial = Factorial * i
 Next
End Function
```

- Vamos a probar si funciona:

En la **ventana Inmediato**, escribe:

?Factorial(5) [Return]

?Factorial(12) [Return]

Si escribes ?Factorial(13), aparece un mensaje de error, ya que el máximo número **Long** era 2.147.483.647 y Factorial(13)= 6.227.020.800 que es superior

- Graba el módulo estándar con el nombre **Utilidades**
- Como ya estamos situados en el “Editor de Visual Basic”, accede al **módulo del formulario controles5**, basta que hagas un doble click en “**Form\_controles5**” de la **Ventana de Proyecto**.

Bien, el problema que nos planteamos es el siguiente:

Al escribir un número (inferior a 13) en el TextBox **txtNumero** y pulsar [Return], nos interesa que en el **txtFactorial** aparezca el factorial del número introducido.

Es decir: al pulsar [Return] (la tecla “Return” tiene por código **Ascii** el número 13), en el **txtNumero**: Evento **KeyPress**

Haz lo siguiente:

- Selecciona el **control: txtNumero** (en la ventana de código de **controles5**) y el **Evento: KeyPress**
- Escribe:

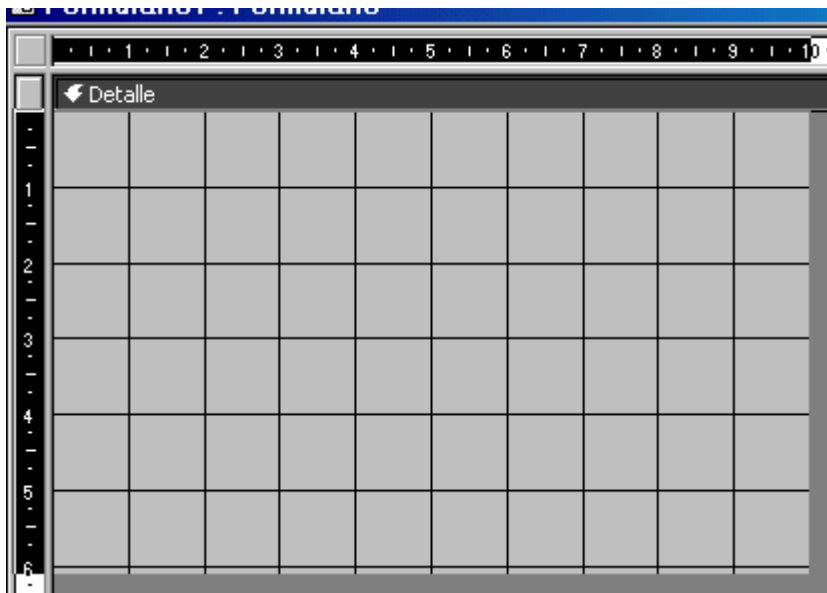
```
Private Sub txtNumero_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
 txtFactorial = Factorial((txtNumero.Text))
 txtFactorial.SetFocus
 End If
End Sub
```

- Selecciona el **control: cmdSalir** y escribe:

```
Private Sub cmdSalir_Click()
 DoCmd.Close
End Sub
```

- Vuelve a [Microsoft Access], graba de nuevo el formulario y pruébalo.

- I) Vamos a investigar en este apartado, cómo funcionan las medidas en VB...
- Crea en la base de datos **CONTROLES** otro formulario, con la sección detalle de un tamaño aproximado de 10cm x 6cm:



- Nos interesa un tamaño **exacto** de 10cm x 6cm. Haz lo siguiente:
  83. Selecciona el formulario y accede a sus propiedades
  84. En la propiedad **Ancho** escribe **10cm**
  85. Selecciona la sección **Detalle** y accede a sus propiedades. En la propiedad **Alto** escribe **6cm**
- Accede a las propiedades del formulario y cambia las cuatro propiedades famosas:
  - Barras de desplazamiento: Ninguna
  - Selectores de registro: No
  - Botones de desplazamiento: No
  - Separadores de registros: No
- Nos gustaría trabajar con un **pie** de formulario. Haz lo siguiente:
  - Cinta de Opciones: Herramientas de diseño de formulario
  - Ficha: Organizar
  - Grupo: Mostrar u Ocultar
  - Encabezado o Pie de formulario
- Selecciona el "**Encabezado del formulario**". Accede a sus propiedades y en la propiedad **Alto**, escribe **0cm**.  
Es decir, acabamos de esconder el "encabezado"
- Selecciona el "Pie del formulario", y en su propiedad "**Alto**", escribe **2,5cm**.
- Graba el formulario con el nombre **controles6**
- Accede al **módulo del formulario controles6**
- Selecciona en la ventana de propiedades: **Detalle Section**, y localiza el valor de la propiedad **Height**. Si todo funciona correctamente aparece **Height: 3402**

Más exactamente: el alto (height) de la "sección Detalle" es de **3.402 twips**

**Twips** es la unidad de medida que utiliza el Visual Basic: 1cm = 567 twips  
Teníamos: alto de detalle= 6cm, es decir **Height de Detalle= 6x567 = 3402 twips**

- Localiza el valor de la propiedad **Height** (alto) de **PieDelFormulario Section**, tenemos Height = 1418 twips = 2,5cm x 567 = 1417,5
- Localiza el valor de la propiedad **Width** del formulario (controles6 Form\_controles6), tenemos:  
Width (ancho) = 5670 twips = 10cm x 567

En definitiva:

Ancho de nuestro formulario: Width = **5670 twips** = 10 cm  
 Alto de nuestro formulario: Height de Detalle = 3402 twips = 6cm  
 Height del Pie = 1418 twips = 2,5cm

-----  
 Height = **4820 twips**

- Si quieres que al ejecutar el formulario, sus medidas correspondan a: Height = 4820 twips y Width = 5670 twips, debes acceder a las propiedades del **Form\_controles6** y cambiar las propiedades:

**InsideHeight = 4820**  
**InsideWidth = 5670**

Siempre y cuando antes:

Botón del Office: Opciones de Access  
 Base de datos actual  
 Ventanas superpuestas  
 Desactiva: Mostrar fichas de documento

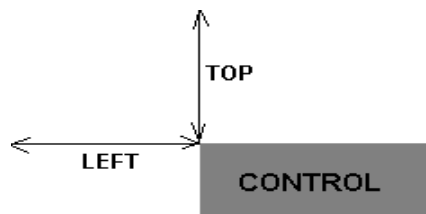
- Vuelve a la pantalla de diseño del formulario **controles6**. Inserta cuatro botones de comando en el **pie** del formulario, no te preocupes de su tamaño ni de su posición.
- Accede al módulo del formulario **controles6** y cambia las siguientes propiedades:

|                 | Name         | Caption | Height | Width |
|-----------------|--------------|---------|--------|-------|
| <b>Comando0</b> | cmdArriba    | ^       | 567    | 567   |
| <b>Comando1</b> | cmdAbajo     | v       | 567    | 567   |
| <b>Comando2</b> | cmdDerecha   | >       | 567    | 567   |
| <b>Comando3</b> | cmdIzquierda | <       | 567    | 567   |

- Vuelve a la pantalla de diseño del formulario y observa lo que acabamos de conseguir: 4 botones de exactamente 1cm x 1cm = 567 twips x 567 twips

Vamos a preocuparnos de su **posición**...

- La posición de un control queda determinada por dos propiedades: "**Top**" y "**Left**"

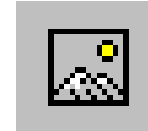


- Vuelve al módulo del formulario **controles6** y cambia las siguientes propiedades:

|                     | <b>Left</b> | <b>Top</b> |
|---------------------|-------------|------------|
| <b>cmdIzquierda</b> | 3628        | 396        |
| <b>cmdDerecha</b>   | 4762        | 396        |
| <b>cmdArriba</b>    | 4195        | 113        |
| <b>cmdAbajo</b>     | 4195        | 680        |

- Vuelve a la pantalla de diseño y verás lo que hemos conseguido.

Desde la pantalla de diseño, haz click en el control **Imagen**:  
y "dibuja" un pequeño recuadro en la sección **detalla** del formulario



Automáticamente aparece el cuadro "**Insertar imagen**", selecciona un archivo gráfico de tu ordenador.

- Accede a las propiedades del control anterior y cambia la propiedad: **Modo de cambiar el tamaño= Zoom**
- Accede al módulo del formulario **controles6**. En la ventana de propiedades, selecciona el control "**Image**" y cambia las siguientes propiedades:
  - Name: imgA
  - Height: 1000
  - Left: 2335
  - Top: 1201
  - Width: 1000
- Vamos a escribir el **código** de nuestro formulario (pretendemos "mover" la imagen):

Escribe los siguientes procedimientos de evento:

```
Private Sub cmdAbajo_Click()
 If imgA.Top + 35 < 2402 Then
 imgA.Top = imgA.Top + 35
 End If
End Sub
```

```
Private Sub cmdArriba_Click()
 If imgA.Top - 35 > 0 Then
 imgA.Top = imgA.Top - 35
 End If
End Sub
```

```
Private Sub cmdDerecha_Click()
 If imgA.Left + 35 < 4670 Then
 imgA.Left = imgA.Left + 35
 End If
End Sub
```

```
Private Sub cmdIzquierda_Click()
 If imgA.Left - 35 > 0 Then
 imgA.Left = imgA.Left - 35
 End If
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
 If MsgBox("¿Deseas Salir?", vbOKCancel, "Salida") = vbCancel Then
 Cancel = True
 End If
End Sub
```

- Vuelve al "Microsoft Access", graba de nuevo el formulario y pruébalo. Cuando estés cansado de "mover" la imagen por el formulario, "cierra" el formulario (click en la X del extremo superior derecho del formulario).

m) Vamos a trabajar en este apartado con un nuevo control

- Crea en la base de datos **CONTROLES**, otro formulario y coloca los siguientes controles:

Para el control **Lista**, debes hacer click en el icono "Cuadro de Lista":



- Cambia la etiqueta del cuadro de texto superior por: **Introduce un Número**
- Cambia la etiqueta de la Lista por: **Listado de Números**
- Cambia la etiqueta del cuadro de texto inferior por: **La MEDIA es:**
- Graba el formulario con el nombre **controles7**
- Accede al módulo del formulario
- Cambia las siguientes propiedades:
  - \* **Form\_controles7**
    - Caption: Cuadro de Lista
    - DividingLines: False
    - NavigationButton: False
    - RecordSelectors: False
    - ScrollBars: 0
  - \* **Botón de Comando superior**
    - Name: cmdAñadir
    - Caption: Añadir
  - \* **Botón de comando inferior**
    - Name: cmdCalcular
    - Caption: Calcular
  - \* **TextBox: "Introduce un número"**
    - Name: txtNumero
  - \* **TextBox: La MEDIA es**
    - Name: txtMedia



**\* Cuadro de Lista**

Name: IstListado

RowSourceType: Value List

El cuadro de lista nos permite guardar o mostrar un listado de valores. Por defecto, dicho listado de valores corresponde a un campo de una tabla o consulta, por esta razón hemos cambiado la propiedad **RowSourceType**, que era **Table/Query** (tabla/Consulta) por "lista de valores" (Value List).

- La idea del programa que pretendemos hacer es:

- \* Escribir números en el primer **TextBox**

- \* El botón [Añadir] nos permite introducir dichos números en el cuadro de lista.

- \* En un momento determinado, si hacemos click en [Calcular] en el **TextBox txtMedia**, aparece la media aritmética de los números que estamos viendo en el **ListBox**

Vamos a ver si lo conseguimos:

Escribe los siguientes procedimientos de evento:

```
Private Sub cmdAñadir_Click()
```

```
 If IstListado.RowSource = "" Then
```

```
 IstListado.RowSource = txtNumero
```

```
 Else
```

```
 IstListado.RowSource = IstListado.RowSource & ";" & txtNumero
```

```
 End If
```

```
 txtNumero = ""
```

```
 txtNumero.SetFocus
```

```
End Sub
```

```
Private Sub cmdCalcular_Click()
```

```
 Dim n As Integer, sum As Integer
```

```
 Dim val As Integer
```

```
 sum = 0
```

```
 For n = 0 To IstListado.ListCount - 1
```

```
 IstListado.Selected(n) = True
```

```
 val = IstListado.Column(ListIndex)
```

```
 sum = sum + val
```

```
 Next
```

```
 txtMedia = sum / IstListado.ListCount
```

```
End Sub
```

- Los dos procedimientos anteriores son muy sencillos, siempre y cuando tengamos en cuenta:

**\* RowSource**

Es la propiedad de un **ListBox** que "guarda" los valores de la lista, separados por un punto y coma, siempre y cuando la propiedad **RowSourceType** sea "Value List", como es nuestro caso.

**\* ListCount**

Es otra propiedad de un **ListBox**, que no es más que el número de valores que contiene la lista.

**\* Selected(3)**

Es otra propiedad de un **ListBox**, que indica si está seleccionado el elemento cuarto de la lista (True) o no lo está (False)

\* **Column(3)**

Determina el valor del número cuatro de la lista.

\* **ListIndex**

Devuelve el índice del valor seleccionado en la "lista" (n valores: índice 0, índice 1, índice 2, .... índice n-1).

- Vuelve al "Microsoft Access", graba de nuevo el formulario **controles7** y pruébalo.
- n) Vamos a trabajar con otro control que a diferencia del ListBox, nos permitirá escribir directamente nuevos valores en el listado.
- Crea en la base de datos **CONTROLES** otro formulario y coloca los siguientes controles:

\* Para colocar el "cuadro combinado" debes hacer clic en el icono "**Cuadro Combinado**":



- \* Elimina la etiqueta del cuadro combinado
- \* En la etiqueta del CommandButton superior, escribe **Añadir**
- \* En la etiqueta del segundo botón, escribe: **Escribe**
- \* Elimina las etiquetas de los dos TextBox
- \* En el último botón escribe: **Añadir al ComboBox**

- Graba el formulario con el nombre **controles8**
- Accede al **módulo del formulario** y cambia las siguientes propiedades:
  - \* **Form\_controles8**
    - Caption: Cuadro Combinado
    - DividingLines: False
    - NavigationButton: False
    - RecordSelectors: False
    - ScrollBars: 0
  - \* **ComboBox**
    - Name: cboCuadro
    - RowSourceType: Value List
    - RowSource: Lunes ; Martes ; Miercoles

- \* **Primer CommandButton**  
Name: cmdAñadir
- \* **Segundo CommandButton**  
Name: cmdEscribe
- \* **Tercer CommandButton**  
Name: cmdAñadirAlCombo
- \* **Primer TextBox**  
Name: txtValorDelCombo
- \* **Segundo TextBox**  
Name: txtValorAlCombo

Observa las dos propiedades del **Cuadro Combinado**, que son comunes con el ListBox:

**RowSourceType**= Value List

Que por defecto tiene el valor: Tabla/Consulta

**RowSource**= Lunes ; Martes ; Miercoles

Que son los valores que aparecerán en el **ComboBox**, observa que hemos de separar los valores utilizando un punto y coma.

- Escribe los siguientes **procedimientos de evento**:

```
Private Sub cmdAñadir_Click()
 cboCuadro.RowSource = cboCuadro.RowSource & ";" & _
 cboCuadro.Value
End Sub
```

Es decir, el valor que hay escrito en el **ComboBox** se añadirá al listado de valores del Cuadro Combinado.

```
Private Sub cmdEscribe_Click()
 txtValorDelCombo = cboCuadro.Value
End Sub
```

Es decir, en el TextBox aparecerá el valor que hay escrito en el ComboBox

```
Private Sub cmdAñadirAlCombo_Click()
 cboCuadro.RowSource = cboCuadro.RowSource & ";" & _
 txtValorAlCombo
End Sub
```

Es decir, el valor escrito en el TextBox se añadirá al listado del Cuadro Combinado.

- Vuelve al "Microsoft Access", graba de nuevo el formulario y pruébalo.
- n) Hasta ahora hemos trabajado con los **módulos asociados a formularios**, vamos a ver como funcionan los **módulos asociados a informes**.

- Crea en la base de datos **CONTROLES** un **informe en Vista Diseño**  
Es decir: Cinta de Opciones: Crear  
Informe en blanco  
Sítuate en Vista Diseño
- Elimina el encabezado y pie de página (Herramientas de diseño de informe: Organizar)
- Vamos a definir un área del informe de 10cm x 10cm = 5670 twips x 5670 twips:  
124. Accede a las propiedades del Informe y cambia la propiedad **Ancho: 10cm**  
125. Selecciona la sección Detalle y accediendo a sus propiedades, cambia la propiedad **Alto: 10cm**
- Graba el informe con el nombre **Informe1**
- Accede al **módulo del informe** (como siempre: click en el icono “Ver Código”)



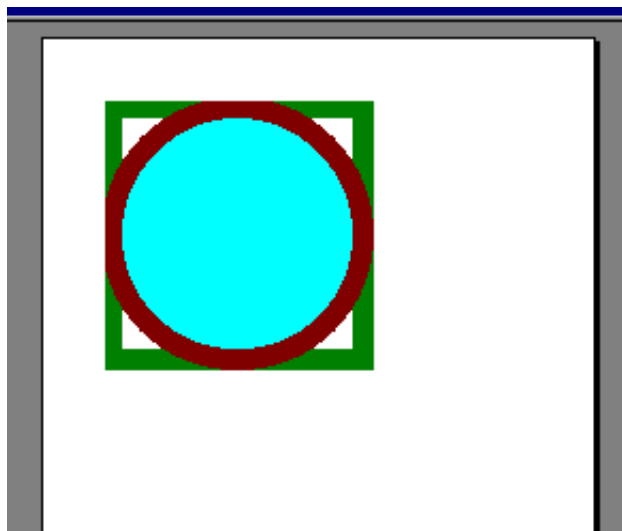
- En la ventana **CONTROLES – Report\_Informe1 (Código)**, selecciona el objeto **Detalle** y el evento **Print**

Escribe el siguiente procedimiento de evento:

```
Private Sub Detalle_Print(Cancel As Integer, PrintCount As Integer)
 DrawWidth = 30
 Line (100, 100)-(5500, 5500), QBColor(2), B
 FillStyle = 0
 FillColor = QBColor(Rnd * 15)
 Circle (2800, 2800), 2700, QBColor(4)
End Sub
```

- Vuelve al “Microsoft Access”, graba de nuevo el informe y accede a la “**Vista Preliminar**” del Informe1

Si todo funciona correctamente aparecerá:



- Bien, vamos a ver lo que hemos hecho:

\* **Circle** y **Line** son métodos de dibujo, que nos permiten “dibujar” en un **Informe (Report)**, cuando se produce el evento **Print**

- \* **DrawWidth = 30**  
Establece el ancho de línea a 30 twips.
- \* **QBColor(argumento)**  
Es una función incorporada al Visual Basic, que determina un color. Argumento = de 0 a 15  
Por ejemplo: QBColor(0): negro  
QBColor(2): verde  
QBColor(6): amarillo
- \* **Line (100,100) – (5500,5500), QBColor(2), B**  
Dibuja un rectángulo de color verde, cuyos vértices opuestos tienen por coordenadas (100,100) y (5500,5500)
- \* **FillStyle= 0**  
A partir de este momento los elementos gráficos tendrán un relleno no transparente
- \* **FillColor = QBColor(Rnd\*15)**  
A partir de este momento los elementos gráficos tendrán por relleno un color aleatorio (Rnd).
- \* **Circle (2800,2800), 2700, QBColor(4)**  
Dibuja un círculo de centro (2800,2800) y radio 2700 twips y color rojo (QBColor(4)).

Sería conveniente que consultaras la ayuda de Visual Basic, sobre los **métodos de dibujo** y sus propiedades, ya que hay infinidad de posibilidades que no tocamos en este manual.

- Crea un **nuevo** informe en vista diseño, cuya sección detalle tenga las medidas: 10cm x 10cm. Grábalo con el nombre **Informe2**

\* Accede a su módulo y escribe el siguiente procedimiento de evento:

```
Private Sub Detalle_Print(Cancel As Integer, PrintCount As Integer)
```

```
 Dim n As Integer
```

```
 DrawWidth = 10
```

```
 For n = 100 To 2700 Step 100
```

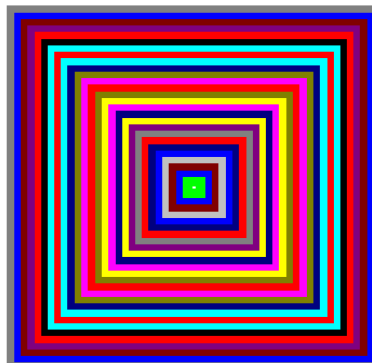
```
 Line (n, n)-(5600 - n, 5600 - n), QBColor(Rnd * 15), B
```

```
 Next
```

```
End Sub
```

\* Vuelve a la pantalla de diseño del Informe2, grba de nuevo el informe y ejecútalo (Vista Preliminar).

Si todo funciona correctamente tendremos:



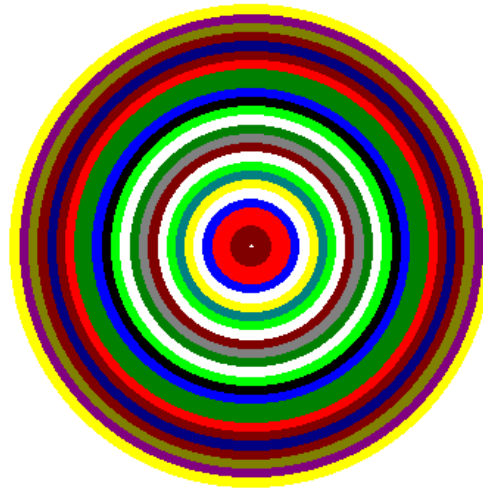
- Haz un nuevo informe de nombre **Informe3** de medidas 10cm x 10cm con el siguiente procedimiento de evento:

```

Private Sub Detalle_Print(Cancel As Integer, PrintCount As Integer)
 DrawWidth = 10
 Dim n As Integer
 For n = 100 To 2700 Step 100
 Circle (2800, 2800), n, QBColor(Rnd * 15)
 Next
End Sub

```

Si todo funciona correctamente tendremos:



- o) Vamos a trabajar con otro tipo de control que es muy importante para trabajar con fechas ...
- Crea en la base de datos **CONTROLES**, un nuevo formulario en blanco de nombre **controles10**.
  - \* Haz clic en el icono **Insertar control ActiveX** del cuadro de Controles
  - \* Selecciona la opción "**Control de Calendario 12.0**"
  - Observa que por defecto, la fecha del calendario corresponde a la fecha actual del sistema.
  - \* Accede a las propiedades del calendario y en la propiedad **Nombre** escribe: **Calendar**
- Inserta en el formulario **controles10** un botón de comando, relativamente grande con las **propiedades**:
  - Nombre: **cmdEstableceFecha**
  - Título: **Haz clic en una fecha del calendario y clic aquí para establecer la fecha INICIAL**
- Inserta en el formulario **controles10**, dos cuadros de texto con las características:
  - Nombre: **txtInicial**
  - Etiqueta: **Fecha Inicial**
  - Nombre: **txtFinal**
  - Etiqueta: **Fecha Final**
- Accede al módulo del formulario **controles10** y escribe el siguiente procedimiento de evento:

```

Private Sub cmdEstableceFecha_Click()
 Dim mensaje As String
 Dim titulo1 As String, titulo2 As String
 mensaje = "FECHAS INCORRECTAS :" _
 & "La fecha inicial debe ser " _
 & "anterior a la final"
 titulo1 = "Haz clic en una fecha del calendario" _
 & " y clic aquí para establecer la fecha" _
 & " INICIAL"
 titulo2 = "Haz clic en una fecha del calendario" _
 & " y clic aquí para establecer la fecha" _
 & " FINAL"
 If cmdEstableceFecha.Caption = titulo1 Then
 cmdEstableceFecha.Caption = titulo2
 txtInicial = Calendar.Value
 cmdEstableceFecha.ForeColor = 255
 Else
 txtFinal = Calendar.Value
 cmdEstableceFecha.Caption = titulo1
 cmdEstableceFecha.ForeColor = 0
 If [txtInicial] > [txtFinal] Then
 MsgBox mensaje
 DoCmd.CancelEvent
 End If
 End If
End Sub

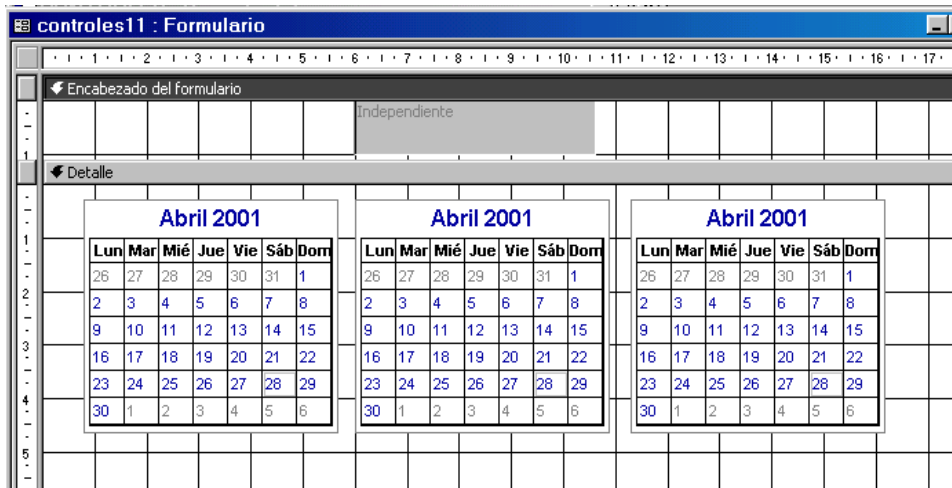
```

- Ejecuta el formulario **controles10** y prueba nuestro programa: está claro que a partir de ahora no será necesario introducir fechas “a mano” ¿verdad?.
  
- p) Utilizando el control “Calendar” vamos a “programar” nuestros calendarios ...
  
- En la base de datos CONTROLES, crea un nuevo formulario en blanco de nombre **controles11**, con las siguientes propiedades:
  - \* Selectores de registro = No
  - \* Botones de desplazamiento = No
  - \* Separadores de registro = No
  
- Selecciona la sección “**Detalle**” de **controles11**, accede a sus “Propiedades”:
  - \* Sitúa el cursor de escritura en la propiedad **Color del fondo**
  - \* Clic en el icono [...], a la derecha de la propiedad.
  - \* Selecciona el color blanco y [Aceptar]
  
- Para hacer aparecer el “encabezado” del formulario, haz:
  - Cinta de Opciones: Herramientas de diseño de formulario
  - Ficha: Organizar
  - Encabezado o pie de formulario
  
- Haz el “encabezado” más pequeño e inserta en el encabezado un cuadro de texto con las siguientes características:
  - \* Elimina su etiqueta
  - \* Nombre: **txtEncab**
  - \* Activado: **No**

- Inserta en la sección “Detalle” de **controles11** un “Control de Calendario 12.0” con las siguientes propiedades:

- \* Nombre: **Cal1**
- \* Ancho: **4,894cm**  
Alto: **4,365cm**
- \* Day: 0
- \* ShowDateSelectors: No

- Copia el control anterior dos veces y colócalos a la derecha del primero:



- Establece como propiedad **Nombre** del segundo calendario: **Cal2** y **Cal3** el tercero.
- Selecciona los tres “calendar” y cópialos 4 veces para tener los 12 meses de un año.
- Sitúate en “**Vista preliminar**”

**Botón del Office  
Imprimir**

**Vista Preliminar**

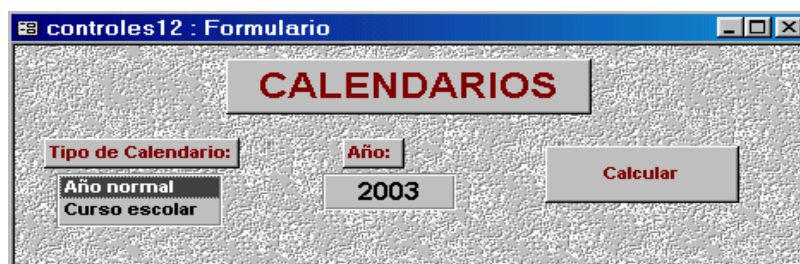
Para “ver” el calendario de todo un año en una hoja DIN A4. Si lo consideras conveniente cambia el tamaño o la situación de los meses, para que quede el calendario más estético en la hoja DIN A4.

- Cambia el nombre de los nueve meses: **Cal4, Cal5, ..., Cal12**.
- Recuerda que has de grabar los cambios hechos en **controles11**.

Vamos a “programar” nuestro calendario ...

- Crea en la B. D. CONTROLES un nuevo formulario en blanco de nombre **controles12**.

- Inserta los siguientes controles:





- \* El primer control: cuadro de lista con las opciones **Año normal** y **Curso escolar**, tiene por nombre **IstTipo**
- \* El cuadro de texto "**Año**" tiene por nombre **txtAn**
- \* El botón de comando [Calcular], tiene por nombre **cmdCalcular**

Supongo que habrás adivinado lo que pretendemos conseguir:

- En el formulario **controles12**, escogemos el tipo de calendario y el año.
- Al hacer clic en [Calcular] se abre el formulario **controles11** con el calendario correspondiente.

Vamos a ver si lo conseguimos:

- Sitúate en el módulo del formulario **controles12** y escribe el siguiente programa:

```
Private Sub cmdCalcular_Click()
 Dim opc As String, an As Integer
 opc = IstTipo.Value
 an = [txtAn]
 DoCmd.OpenForm "controles11"
 If opc = "Año normal" Then
 Forms!controles11.txtEncab = an
 Forms!controles11.cal1.Month = 1
 Forms!controles11.cal1.Year = an
 Forms!controles11.cal1.Day = False
 Forms!controles11.cal2.Month = 2
 Forms!controles11.cal2.Year = an
 Forms!controles11.cal2.Day = False
 Forms!controles11.cal3.Month = 3
 Forms!controles11.cal3.Year = an
 Forms!controles11.cal3.Day = False
 Forms!controles11.cal4.Month = 4
 Forms!controles11.cal4.Year = an
 Forms!controles11.cal4.Day = False
 Forms!controles11.cal5.Month = 5
 Forms!controles11.cal5.Year = an
 Forms!controles11.cal5.Day = False
 Forms!controles11.cal6.Month = 6
 Forms!controles11.cal6.Year = an
 Forms!controles11.cal6.Day = False
 Forms!controles11.cal7.Month = 7
 Forms!controles11.cal7.Year = an
 Forms!controles11.cal7.Day = False
 Forms!controles11.cal8.Month = 8
 Forms!controles11.cal8.Year = an
 Forms!controles11.cal8.Day = False
 Forms!controles11.cal9.Month = 9
 Forms!controles11.cal9.Year = an
 Forms!controles11.cal9.Day = False
 Forms!controles11.cal10.Month = 10
 Forms!controles11.cal10.Year = an
 Forms!controles11.cal10.Day = False
 Forms!controles11.cal11.Month = 11
 Forms!controles11.cal11.Year = an
 Forms!controles11.cal11.Day = False
 End If
End Sub
```

```
Forms!controles11.cal12.Month = 12
Forms!controles11.cal12.Year = an
Forms!controles11.cal12.Day = False
Else
Forms!controles11.txtEncab = an & "/" & (an + 1)
Forms!controles11.cal1.Month = 9
Forms!controles11.cal1.Year = an
Forms!controles11.cal1.Day = False
Forms!controles11.cal2.Month = 10
Forms!controles11.cal2.Year = an
Forms!controles11.cal2.Day = False
Forms!controles11.cal3.Month = 11
Forms!controles11.cal3.Year = an
Forms!controles11.cal3.Day = False
Forms!controles11.cal4.Month = 12
Forms!controles11.cal4.Year = an
Forms!controles11.cal4.Day = False
Forms!controles11.cal5.Month = 1
Forms!controles11.cal5.Year = an + 1
Forms!controles11.cal5.Day = False
Forms!controles11.cal6.Month = 2
Forms!controles11.cal6.Year = an + 1
Forms!controles11.cal6.Day = False
Forms!controles11.cal7.Month = 3
Forms!controles11.cal7.Year = an + 1
Forms!controles11.cal7.Day = False
Forms!controles11.cal8.Month = 4
Forms!controles11.cal8.Year = an + 1
Forms!controles11.cal8.Day = False
Forms!controles11.cal9.Month = 5
Forms!controles11.cal9.Year = an + 1
Forms!controles11.cal9.Day = False
Forms!controles11.cal10.Month = 6
Forms!controles11.cal10.Year = an + 1
Forms!controles11.cal10.Day = False
Forms!controles11.cal11.Month = 7
Forms!controles11.cal11.Year = an + 1
Forms!controles11.cal11.Day = False
Forms!controles11.cal12.Month = 8
Forms!controles11.cal12.Year = an + 1
Forms!controles11.cal12.Day = False
End If
End Sub
```

- Graba los cambios hechos en **controles12** y sólo queda “jugar” a hacer calendarios. Espero que te funcionen.

## Para Saber más

### Notación Húngara

Habrás notado que hemos “nombrado” los diferentes controles de una forma un tanto especial. Dentro del mundo de la programación en Visual Basic, existe una notación ampliamente usada que consiste en nombrar un control con tres letras, abreviatura del control en inglés, y a continuación una palabra que indica de alguna forma lo que hace el control.

Ejemplo:

**cmdSalir**, será el nombre de un CommandButton (cmd) que tiene alguna relación con “salir”.

A continuación tienes las abreviaturas de los controles más usuales:

|     |               |
|-----|---------------|
| cmd | CommandButton |
| txt | TextBox       |
| lbl | Label         |
| lst | ListBox       |
| cbo | ComboBox      |
| img | Image         |

### El lenguaje de programación “Visual Basic”

Se dice que el Visual Basic es un lenguaje de programación orientado a los **objetos** y conducido por **eventos**.

Concretemos más:

Todos los controles, formularios, informes, etc son **objetos**. Un objeto consta de propiedades, métodos y eventos:

- **Propiedades:** características del objeto

Ejemplo:

**txtCaja.BackColor = vbBlue**

“BackColor” es una propiedad del TextBox **txtCaja**, que le asignamos el color azul.

- **Métodos:** lo que puede hacerse en el objeto. No son más que procedimientos ya escritos y asociados a los objetos.

Ejemplo:

**txtNumero.SetFocus**

El “foco” se sitúa en el objeto txtNumero

- **Eventos:** situaciones que se producen por acción del usuario o del sistema

Ejemplo:

**cmdSalir\_Click**

Al hacer click en el botón “cmdSalir”

## Autoevaluación 3

1) Haz un programa que sirva para repasar las tablas de multiplicar:

- Crea una base de datos de nombre **Eval3A** con un formulario de nombre **Multiplicar**, con el siguiente contenido y aspecto:

- Nombres:

|                |                |               |                  |               |
|----------------|----------------|---------------|------------------|---------------|
|                |                | <b>txtNum</b> |                  | <b>txtPen</b> |
|                | <b>txtUno</b>  | <b>txtDos</b> | <b>txtResul</b>  |               |
| <b>txtBien</b> |                |               | <b>txtCorrec</b> |               |
| <b>txtMal</b>  |                |               |                  |               |
|                |                |               | <b>txtCorrec</b> |               |
|                |                |               | <b>txtNota</b>   |               |
|                | <b>cmdOtra</b> |               |                  |               |

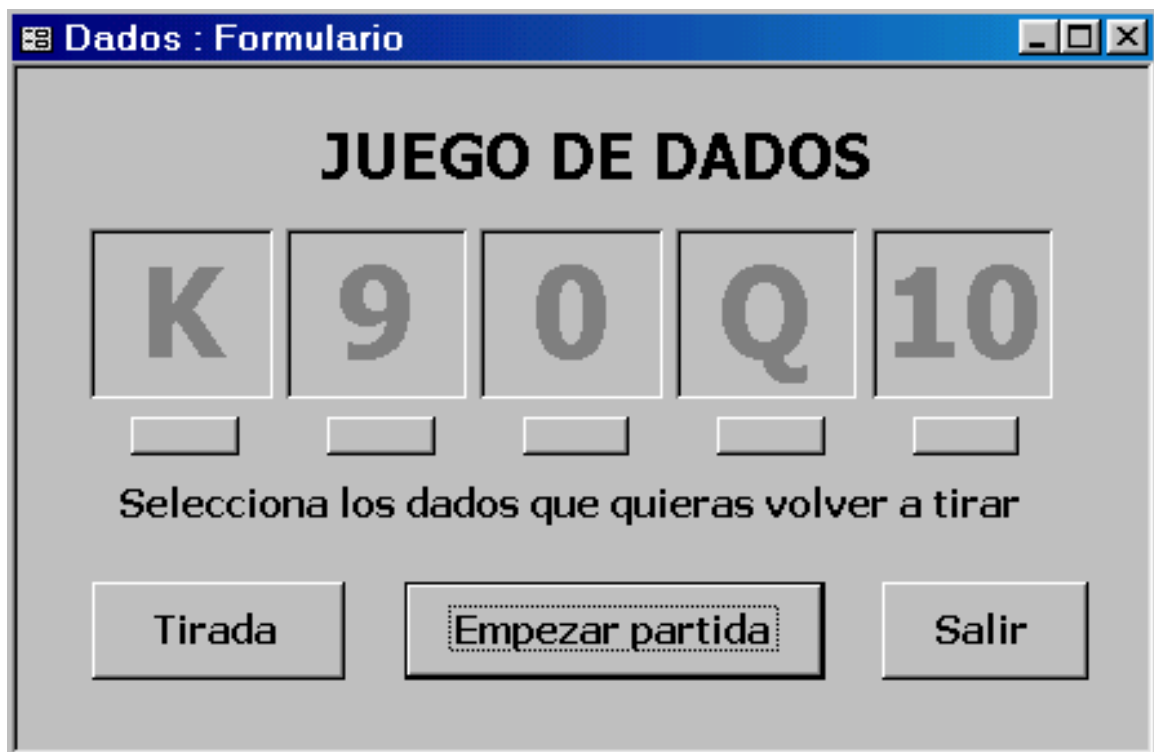
Además hay otro botón con la propiedad **Visible = False**, de nombre **cmdSalir**

- Crea un módulo de nombre **Módulo1**, que contenga una función que nos dé la **nota final**.
- El problema que nos planteamos creo que está claro:
  - \* Al abrir el formulario, escribimos en **txtNum** el número de multiplicaciones que queremos hacer.
  - \* En el **txtPen** aparece en cada momento el número de multiplicaciones que tenemos pendientes.
  - \* En los cuadros **txtUno** y **txtDos** aparecen dos números aleatorios entre 1 y 9
  - \* En el cuadro **txtResul** escribimos el resultado de la multiplicación, que aparece en **txtUno** y **txtDos**.
  - \* Si la respuesta es correcta, aparece en el cuadro **txtCorrec** la frase "Muy Bien"
  - \* Si la respuesta es incorrecta, aparece en el cuadro **txtCorrec** la contestación correcta.
  - \* Los cuadros **txtBien** y **txtMal**, cuentan el número de respuestas correctas e incorrectas respectivamente.
  - \* El botón **cmdOtra** sirve para una nueva multiplicación.

\* Al acabarse todas las multiplicaciones: en el cuadro **txtNota**, aparece la nota cualitativa (Excelente, Notable, Bien, etc). El botón [Otra] se hace invisible y aparece el botón [Salir]

2) Haz un programa que nos permita jugar a los dados:

- Crea una base de datos de nombre **Eval3B**, con un formulario de nombre **Dados** con el siguiente contenido y aspecto:



- Nombres:

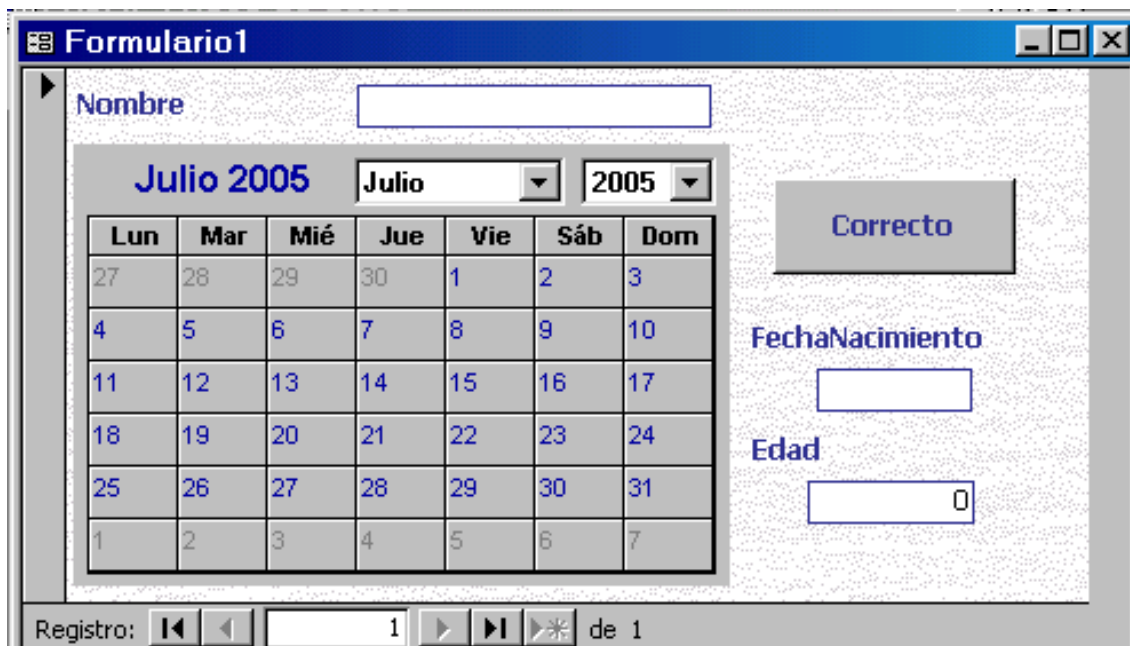
|                  |               |                   |                 |             |
|------------------|---------------|-------------------|-----------------|-------------|
| <b>txt1</b>      | <b>txt2</b>   | <b>txt3</b>       | <b>txt4</b>     | <b>txt5</b> |
| <b>b1</b>        | <b>b2</b>     | <b>b3</b>         | <b>b4</b>       | <b>b5</b>   |
|                  | <b>lblIns</b> |                   |                 |             |
| <b>cmdTirada</b> |               | <b>cmdEmpezar</b> | <b>cmdSalir</b> |             |

- Crea un módulo de nombre **Módulo1**, que contenga una función de nombre **Dado**, que sirva para asignar a cada número (del 1 al 6) cada una de las caras del dado (0, K, Q, J, 9, 10).
- El programa debería hacer lo siguiente:
  - \* Al abrir el formulario **Dados**, se inicializan los controles txt1, txt2, ..., txt5 a "vacío", el botón [Tirada] desaparece y el mensaje **lblIns** también.
  - \* Al hacer clic en [Empezar partida]: aparecen en txt1, txt2, ..., txt5 los valores "aleatorios" de los dados, el botón [Tirada] y el mensaje **lblIns** se hacen visibles.
  - \* Al hacer lo que nos dice el mensaje (clic en b1, b2, b3, ..., b5), desaparecen los dados correspondientes.
  - \* Al hacer clic en [Tirada], los dados que habían desaparecido se hacen visibles con otros valores "aleatorios".

3) Haz un programa que nos permita rellenar una tabla Access con los campos: **Nombre**, **FechaNacimiento**, **Edad** utilizando un formulario donde únicamente hemos de escribir el **nombre** y seleccionar la fecha de nacimiento en un control **Calendar**:

- Crea una base de datos de nombre **Eval3C**, con una tabla de nombre **Tabla1** de estructura:

|                 |                         |
|-----------------|-------------------------|
| Nombre          | Texto y Clave Principal |
| FechaNacimiento | Fecha/Hora              |
| Edad            | Númérico                |
- Crea un autoformulario para la tabla anterior y grábalo con el nombre **Formulario1**
- Inserta un control **Calendar** y un **botón de comando** en el **formulario1**:



Control Calendar: Nombre= **Cal**  
Indice de tabulación= **1**  
Botón de comando: Nombre= **cmdCorrecto**  
Indice de tabulación= **2**

- El programa ha de funcionar de la siguiente forma:
  - \* Escribimos el "Nombre"
  - \* Seleccionamos en el calendario la fecha de nacimiento.
  - \* Al hacer clic en [Correcto], se rellenan automáticamente los campos **FechaNacimiento** y **Edad**.

## 4

## Objetos de Acceso a Datos (DAO)

Después de la introducción al Visual Basic que hemos visto en los ejercicios anteriores, vamos a aplicar los conocimientos adquiridos de VB en nuestras bases de datos, que de hecho es lo que nos interesa.

Para trabajar con DAO, es conveniente trabajar con versiones anteriores al Access 2007, de esta forma conservamos la compatibilidad de "referencias".

Vamos a crear un programa que nos permita controlar las llamadas telefónicas ...

\* Crea una nueva base de datos de nombre **LLAMADAS** en *TuCarpeta*, en formato **Access 2002/2003** (la extensión será MDB), cuando grabes la base de datos en **Tipo** debería aparecer: **Base de datos de Microsoft Office Access (formato 2002-2003)**

\* En la B.D. LLAMADAS crea una tabla de nombre **Datos** con la siguiente estructura:

| <b>Campos</b> | <b>Tipo</b>  |
|---------------|--------------|
| Id            | Autonumérico |
| DiaHora       | Fecha/Hora   |
| Empleado      | Texto        |
| Elquellama    | Texto        |
| Presente      | Sí/No        |
| Notas         | Memo         |

Clave principal: **Id**

- Vamos a crear en la B.D. LLAMADAS un formulario que "recoja" los datos de una llamada telefónica...

Crea un formulario en blanco con los siguientes controles:

\* Un título bonito (control Etiqueta) de contenido: **Llamadas Telefónicas**

\* Un "Cuadro de Texto" sin etiqueta. Supondré que su propiedad **Nombre** es **Texto1**

\* Un "Cuadro de Lista" con las siguientes propiedades:

|                    |                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------|
| Nombre:            | Lista3                                                                                         |
| Tipo de origen:    | Lista de Valores                                                                               |
| Origen de la fila: | "Gerente";"Secretaria Administración"<br>;"Contable";"Secretaria Dirección" ; "Jefe de Ventas" |
| Etiqueta:          | Llamada para:                                                                                  |

\* Un "Cuadro de texto" de nombre **Texto5** y etiqueta: **"de parte"**

\* Un "Grupo de Opciones" de nombre **Marco7** y etiqueta: **"Está"**, que contenga dos "botones de opción":

|                         |                     |
|-------------------------|---------------------|
| Nombre: <b>Opción9</b>  | Etiqueta: <b>Sí</b> |
| Nombre: <b>Opción11</b> | Etiqueta: <b>No</b> |

\* Un "Cuadro de Texto" de nombre **Texto13** y etiqueta: **Notas**

\* Un "Botón de Comando" de nombre **Comando15** y etiqueta: **Siguiente Llamada**

\* Graba el formulario con el nombre **Portada**

- Pasemos al **Editor de Visual Basic** (módulo del formulario), es decir click en el icono **“Ver Código”**.

Cambia las siguientes **propiedades**:

**Form**

Caption: Control de Llamadas Telefónicas  
 DividingLines: False  
 NavigationButtons: False  
 RecordSelectors: False

**Textbox: Texto1**

Enabled: False

Vamos a introducir el **código**, escribe:

```
Private Sub Form_Load()
 [Texto1] = Now
End Sub

Private Sub Comando15_Click()
 Dim dbf As Database, registros As Recordset
 Set dbf = CurrentDb
 Set registros = dbf.OpenRecordset("Datos", dbOpenDynaset)
 registros.AddNew
 registros("DiaHora") = Forms![Portada]!Texto1
 registros("Empleado") = Forms![Portada]!Lista3
 registros("Elquellama") = Forms![Portada]!Texto5
 If Forms![Portada]!Marco7 = 1 Then
 registros("Presente") = 1
 Else
 registros("Presente") = 0
 End If
 registros("Notas") = Forms![Portada]![Texto13]
 registros.Update
End Sub
```

Para que funcione el programa anterior (primer programa DAO que hacemos. DAO= Objetos de Acceso a Datos), debes hacer lo siguiente:

- \* Desde la pantalla del editor de Visual Basic...
- \* Menú Herramientas  
Referencias...
- \* Debes tener activas las siguientes referencias:  
 Microsoft Access 12.0 Object Library  
 Microsoft DAO 3.6 Object Library  
 OLE Automation  
 Visual Basic For Applications  
 Microsoft Visual Basic for Applications Extensibility 5.3  
 Probablemente deberás activar la "Microsoft DAO 3.6 Object Library", en este caso ...



\* Utilizando las flechas de “**Prioridad**”, debes colocar la referencia “Microsoft DAO 3.6 Object Library” por debajo de “Microsoft Access 12.0 Object Library”.

\* Por último haz clic en [Aceptar] de la ventana “**Referencias-LLAMADAS**”

Antes de estudiar lo que acabamos de hacer, vamos a ver si funciona:

- Ejecuta el formulario “**Portada**”

La tabla “Datos” es conveniente que esté cerrada

\* Si todo funciona correctamente, aparece en el primer campo el día y hora “actuales”.

\* Selecciona en el segundo campo (cuadro de lista), el valor “Contable”.

\* “de parte de:”: **Pepito Valdemoro**

\* “Está:”: **Sí**

\* “Notas”: inventa lo que quieras.

\* Clic en [Siguiente Llamada]

\* “Cierra” el formulario y accede al contenido de la tabla **Datos**

Espero que te haya funcionado: aparece un primer registro que contiene los datos de la “llamada” que nos acaban de hacer.

- Investiguemos el nuevo programa **DAO**: accede al módulo del formulario, y observa el procedimiento **Comando15\_Click()**

\* **Dim dbf As Database, registros As Recordset**

Declaramos una variable de nombre **dbf** y tipo **base de datos** (DataBase). Declaramos otra variable de nombre **registros** y tipo **grupo de registros** (RecordSet)

\* **Set dbf= CurrentDb**

Asignamos a la variable **dbf** la base de datos activa (CurrentDb), en nuestro caso la B.D. LLAMADAS

\* **Set registros=dbf.OpenRecordset(“Datos”,dbOpenDynaset)**

Asignamos a la variable **registros**, los registros de la tabla **Datos**

\* **registros.AddNew**

Añadimos un nuevo registro en blanco (en la tabla **Datos**).

\* **registros(“DiaHora”)=Forms![Portada]!Texto1**

En el campo “**DiaHora**” de la tabla “**Datos**” del registro nuevo, que se acaba de crear, se **coloca** lo que tenemos en el **Texto1** del formulario **Portada**

\* Lo mismo para el resto de los campos. En el caso concreto del campo “**Presente**”, hemos de asignar el valor 1 (Sí) o 0 (No) según el botón de opción del **Marco7** que hay activado.

\* **registros.Update**

“Actualizamos” el registro, es decir **grabamos** el registro de la tabla “**Datos**”.

Observa de qué forma tan sencilla enlazamos los datos “independientes” de un formulario con una tabla Access.

- Vamos a mejorar el programa:

\* Inserta en el formulario **Portada**, dos nuevos cuadros de texto con las siguientes características:

\* Cuadro de texto:

Etiqueta: **Total de llamadas**  
 Nombre: **x**  
 Activado: **No**

\* Cuadro de texto:

Etiqueta: **Número de llamada**  
 Nombre: **y**  
 Activado: **No**

\* Crea el siguiente procedimiento de evento (hazlo a partir del anterior **Private Sub Form\_Load**):

```
Private Sub Form_Load()
 [Texto1] = Now
 Dim dbf As Database, reg As Recordset
 Set dbf = CurrentDb
 Set reg = dbf.OpenRecordset("Datos", dbOpenDynaset)
 reg.MoveLast
 [x] = reg.RecordCount
 reg.Close
 dbf.Close
 [y] = [x] + 1
 [Lista3] = ""
 [Texto5] = ""
 [Marco7] = ""
 [Texto13] = ""
 Lista3.SetFocus
End Sub
```

Observa:

\* Al abrir el formulario (Form\_Load)

\* **reg.MoveLast**

Nos situamos en el último registro del “recordset”

\* **reg.RecordCount**

“Cuenta” todos los registros del “recordset” (en nuestro caso de “Datos”). Es conveniente que previamente nos situemos en el último registro (MoveLast).

\* **reg.Close, dbf.Close**

Cerramos el recordset (Tabla: Datos) y la base de datos.

En definitiva: si todo funciona correctamente, al abrir el formulario **Portada** en el cuadro “x” aparecerá el número de registros de la tabla “Datos”, es decir aparecerá el **número de llamadas** y en el cuadro “y” aparecerá una unidad más, es decir, el cuadro nos mostrará la llamada “actual”.

\* No pruebes aún el programa, ya que hemos de corregir el correspondiente a "grabar las llamadas".

\* En efecto, corrige el **Comando15\_Click()**:

```

Private Sub Comando15_Click()
 Dim dbf As Database, registros As Recordset
 Set dbf = CurrentDb
 Set registros = dbf.OpenRecordset("Datos", dbOpenDynaset)
 registros.AddNew
 registros("DiaHora") = Forms![Portada]!Texto1
 registros("Empleado") = Forms![Portada]!Lista3
 registros("Elquellama") = Forms![Portada]!Texto5
 If Forms![Portada]!Marco7 = 1 Then
 registros("Presente") = 1
 Else
 registros("Presente") = 0
 End If
 registros("Notas") = Forms![Portada]!Texto13
 registros.Update
 registros.MoveLast
 [x] = registros.RecordCount
 registros.Close
 dbf.Close
 [y] = [x] + 1
 [Lista3] = ""
 [Texto5] = ""
 [Marco7] = ""
 [Texto13] = ""
 Lista3.SetFocus
End Sub

```

\* Prueba nuestro programa, introduciendo unas cuantas llamadas. Espero que te funcione.

\* Otra mejora que podríamos introducir:

Corrige:

```

Private Sub Comando15_Click()
 If MsgBox("Quieres grabar esta llamada?", vbYesNo) = vbNo Then
 [Lista3] = ""
 [Texto5] = ""
 [Marco7] = ""
 [Texto13] = ""
 Lista3.SetFocus
 Exit Sub
End If

```

```

Dim dbf As Database, registros As Recordset
Set dbf = CurrentDb
Set registros = dbf.OpenRecordset("Datos", dbOpenDynaset)
registros.AddNew
registros("DiaHora") = Forms![Portada]!Texto1
registros("Empleado") = Forms![Portada]!Lista3
registros("Elquellama") = Forms![Portada]!Texto5
If Forms![Portada]!Marco7 = 1 Then
 registros("Presente") = 1
Else

```

```

 registros("Presente") = 0
End If
registros("Notas") = Forms![Portada]![Texto13]
registros.Update
registros.MoveLast
[x] = registros.RecordCount
registros.Close
dbf.Close
[y] = [x] + 1
[Listas3] = ""
[Texto5] = ""
[Marco7] = ""
[Texto13] = ""
Lista3.SetFocus
End Sub

```

\* Ejecuta el programa para probarlo.

- Resulta que el campo correspondiente a "El que llama", lo hemos introducido, con las prisas, en letras minúsculas. Nos gustaría hacer un programa DAO, que cambiara el contenido del campo "El que llama" a letras minúsculas ...

\* En la B.D. LLAMADAS, crea un formulario en blanco de nombre **Mantenimiento**, con las siguientes propiedades:

|                    |                           |
|--------------------|---------------------------|
| Caption:           | Mantenimiento de Llamadas |
| DividingLines:     | False                     |
| NavigationButtons: | False                     |
| RecordSelectors:   | False                     |

\* Inserta en el formulario **Mantenimiento** un botón de comando con las siguientes propiedades:

|          |                            |
|----------|----------------------------|
| Name:    | cmdMayus                   |
| Caption: | El que llama en Mayúsculas |

\* Escribe el siguiente procedimiento de evento:

```

Private Sub cmdMayus_Click()
 Dim b As Database
 Dim r As Recordset
 Set b = CurrentDb
 Set r = b.OpenRecordset("Datos", dbOpenTable)
 r.MoveFirst
 Do Until r.EOF
 r.Edit
 r("Elquellama") = UCase(r("Elquellama"))
 r.Update
 r.MoveNext
 Loop
 r.Close
End Sub

```

\* Antes de continuar, vamos a ver si funciona, es decir:

\* Ejecuta el formulario "Mantenimiento"

\* Clic en [El que llama en Mayúsculas]

\* Abre la tabla "**Datos**" y comprueba que el contenido de la columna "Elquellama" está en letras mayúsculas.

Vamos a analizar el programa **cmdMayus\_Click**, porque aparecen bastantes elementos nuevos:

En primer lugar la estructura básica de un programa DAO:

```

Dim b As Database
Dim r As Recordset
Set b=CurrentDb
Set r= ...
.....
.....
r.Close

```

Es decir:

- Declaración de una base de datos (b) y un grupo de registros (r).
- Asignación de la base de datos a la **activa** (CurrentDb)
- Asignación del grupo de registros (r)
- Líneas del programa.
- Última sentencia: **r.Close**, es decir, cerramos el "recordset".

Asignación del recordset: **Set r=b.OpenRecordset("Datos",dbOpenTable)**

Es decir, "r" no es más que los registros de la tabla "Datos".

En los programas DAO anteriores, en lugar de **dbOpenTable**, habíamos utilizado el argumento **dbOpenDynaset**.

Para los programas que hemos hecho hasta ahora, es indiferente utilizar **dbOpenTable** o **dbOpenDynaset**.

**dbOpenTable** se utiliza para tablas locales, encambio **dbOpenDynaset** se utiliza para tablas locales, o vinculadas o consultas o sentencia **sql**.

#### \* **r.MoveFirst**

Nos situamos en el primer registro del recordset (primer registro de la tabla "Datos")

#### \* **Do Until r.EOF**

```

sentencia 1
sentencia 2
sentencia 3
r.MoveNext

```

#### **Loop**

La estructura anterior: recorrerá cada uno de los registros del recordset, ejecutando en cada registro las sentencias 1, 2 y 3

En nuestro caso, en cada registro:

#### 1º) **r.Edit**

Editará el registro correspondiente.

#### 2º) **r("Elquellama")=UCCase(r("Elquellama"))**

Cambiará el contenido del campo "Elquellama", por el mismo contenido, pero en mayúsculas (UCCase).

#### 3º) **r.Update**

Actualizará los cambios hechos en el registro.

- Vamos a utilizar la estructura **Do Until r.EOF - Loop** para visualizar el contenido de un campo, desde "fuera" ....

Inserta en el formulario **Mantenimiento** otro botón de comando con las siguientes propiedades:

Name: cmdVer  
Caption: Ver los que han llamado.

\* Escribe el siguiente código:

```
Private Sub cmdVer_Click()
 Dim x As Database
 Dim y As Recordset
 Set x = CurrentDb
 Set y = x.OpenRecordset("Datos")
 y.MoveFirst
 Do Until y.EOF
 MsgBox y("Elquellama")
 y.MoveNext
 Loop
 y.Close
End Sub
```

\* Prueba el funcionamiento del programa, es decir: ejecuta el formulario "Mantenimiento" y clic en [Ver los que han llamado].

Observa que en el "OpenRecordset" no especificamos el **tipo**, sea **dbOpenTable** o **dbOpenDynaset**

Vamos a hacerlo más elegante. Modifica el programa anterior de la siguiente forma:

```
Private Sub cmdVer_Click()
 Dim x As Database
 Dim y As Recordset
 Dim salida As String
 Set x = CurrentDb
 Set y = x.OpenRecordset("Datos")
 y.MoveFirst
 Do Until y.EOF
 salida = salida & y("Elquellama") & vbCrLf
 y.MoveNext
 Loop
 y.Close
 MsgBox salida
End Sub
```

\* Pruébalo.

- Vamos a continuar "manipulando" datos ...

Nos gustaría hacer un programa que hiciera lo siguiente:

1º) Crear por DAO un nuevo campo de nombre **NombreEmpleado** para la tabla "Datos".

2º) En el nuevo campo aparece el nombre del empleado correspondiente que resulta ser:

|                           |          |
|---------------------------|----------|
| Gerente                   | Paquito  |
| Secretaria Administración | Felipa   |
| Contable                  | Pepito   |
| Secretaria Dirección      | Herminia |
| Jefe de Ventas            | Ambrosio |

Vamos a ver si lo conseguimos:

\* Inserta en el formulario **Mantenimiento** un botón de comando con las propiedades:

Name: cmdNombreEmpleado  
Caption: Nombre de los Empleados

\* Vamos a hacer una función que “traduzca”: Gerente por Pepito, Secretaria Administración por Felipa, etc.

Crea un **Módulo** de nombre **Módulo1**, que contenga el siguiente programa:

```
Public Function Traduce(a As String) As String
 Select Case a
 Case "Gerente"
 Traduce = "Paquito"
 Case "Secretaria Administración"
 Traduce = "Felipa"
 Case "Contable"
 Traduce = "Pepito"
 Case "Secretaria Dirección"
 Traduce = "Herminia"
 Case "Jefe de Ventas"
 Traduce = "Ambrosio"
 End Select
End Function
```

\* Escribe el siguiente código:

```
Private Sub cmdNombreEmpleado_Click()
 Dim bd As Database
 Dim tb As TableDef
 Dim fld As Field
 Set bd = CurrentDb
 Set tb = bd.TableDefs("Datos")
 Set fld = tb.CreateField("NombreEmpleado", dbText)
 tb.Fields.Append fld
 Dim y As Recordset
 Set y = bd.OpenRecordset("Datos")
 y.MoveFirst
 Do Until y.EOF
 y.Edit
 y("NombreEmpleado") = Traduce(y("Empleado"))
 y.Update
 y.MoveNext
 Loop
 y.Close
End Sub
```

\* Antes de nada, vamos a ver si funciona, es decir:

\* “Abre” el formulario **Mantenimiento**

\* Clic en [Nombre de los Empleados]

\* “Cierra” el formulario

\* “Abre” la tabla **Datos**. Si todo ha funcionado correctamente, tenemos un nuevo campo de nombre **NombreEmpleado**, donde aparece: Pepito, Felipa, Ambrosio, etc, según el contenido del campo **Empleado** correspondiente.

Observemos el procedimiento **cmdNombreEmpleado** ya que aparecen elementos nuevos:

**Dim tb As TableDef**

Declaramos una variable de nombre **tb** y tipo "tabla"

**Dim fld As Field**

Declaramos una variable de nombre **fld** y tipo **campo**

**Set tb=bd.TableDefs("Datos")**

Asignamos la variable **tb** a nuestra tabla "Datos"

**Set fld=tb.CreateField("NombreEmpleado",dbText)**

Asignamos la variable "fld" al campo "NombreEmpleado" tipo texto (dbText) de la tabla "tb" (Datos).

**tb.Fields.Append fld**

Añadimos el campo "fld" (NombreEmpleado) a la tabla "tb" (Datos).

El resto del programa creo que está claro, ya que utilizamos instrucciones ya conocidas.

- Vamos a comenzar a trabajar con el **SQL** ...

El **SQL** (lenguaje de consulta estructurado) es el lenguaje utilizado por todos los gestores de bases de datos para consultar, actualizar y administrar bases de datos relacionales.

**SQL** se puede utilizar para recuperar, ordenar y filtrar datos específicos que se van a extraer de la base de datos.

Inserta en el formulario **Mantenimiento** un botón de comando con las siguientes propiedades:

|          |                               |
|----------|-------------------------------|
| Name:    | cmdOrdenar                    |
| Caption: | Listado ordenado por Empleado |

Escribe el siguiente código:

```
Private Sub cmdOrdenar_Click()
 Dim db As Database, r As Recordset
 Dim sql As String, salida As String
 Set db = CurrentDb
 sql = "SELECT * FROM Datos ORDER BY Empleado"
 Set r = db.OpenRecordset(sql)
 r.MoveFirst
 Do Until r.EOF
 salida = salida & r("Id") & " - " & _
 r("Empleado") & " - " & r("Elquellama") & _
 vbCrLf
 r.MoveNext
 Loop
 r.Close
 MsgBox salida
End Sub
```

Ejecuta el programa, es decir:

- "Abre" el formulario **Mantenimiento**
- Clic en [Listado ordenado por Empleados]
- Si todo funciona correctamente aparecerá el listado de nuestra tabla Datos (campos: Id, Empleado y Elquellama), ordenado según el campo **Empleado**

Estudiamos el programa **cmdOrdenar\_Click**:



- `Dim sql As String`  
Declaramos una variable de texto de nombre **sql** (podría ser cualquier otro nombre).
- `sql="SELECT * FROM Datos ORDER BY Empleado"`  
Asignamos a la variable **sql** una sentencia SQL, que indica lo siguiente: "Selecciona todos los campos (\*) de la tabla **Datos** y ordena los registros, según el campo **Empleado**"
- `Set r=db.OpenRecordset(sql)`  
Asignamos a "r" el grupo de registros correspondiente a la sentencia `sql`.  
Observa pues, que el "Recordset" no tiene porqué ser una tabla determinada.  
Supongo que intuyes la "potencia" de lo que acabamos de hacer: **una sentencia SQL como "Recordset"**. Está claro pues, la importancia de conocer el lenguaje estructurado SQL. No te preocupes de momento, dedicaremos el próximo capítulo del manual a **programar en SQL**

- Vamos a trabajar con otro objeto DAO: las consultas.

Inserta en el formulario **Mantenimiento** un nuevo botón de comando con las propiedades:

Name: cmdConsulta  
Caption: Crea Consulta

Escribe el siguiente código:

```
Private Sub cmdConsulta_Click()
 Dim bd As Database
 Dim tb As TableDef
 Dim q As QueryDef
 Set bd = CurrentDb
 Set q = bd.CreateQueryDef("Llamadas ordenadas por Empleado")
 q.sql = "SELECT * FROM Datos ORDER BY Empleado"
End Sub
```

Ejecuta el programa, es decir:

- "Abre" el formulario **Mantenimiento**
- Clic en el botón [Crea Consulta]
- "Cierra" el formulario. Entra y sal del Editor de Visual Basic, para que aparezca en el Panel de Objetos, a la izquierda de la pantalla
- Se acaba de crear una consulta en nuestra base de datos. En efecto, "abre" la consulta **Llamadas ordenadas por Empleado**
- Clic en la flechita de "Ver"
- Selecciona la opción: **SQL Vista SQL**  
Si todo va bien, aparece: **SELECT \* FROM Datos ORDER BY Empleado**

Observemos el programa `cmdConsulta_Click()`

- `Dim q As QueryDef`  
Declaramos una variable de nombre `q` tipo `consulta` (QueryDef)
- `Set q=db.CreateQueryDef("Llamadas ordenadas por Empleado")`  
Asignamos a la variable "q" la consulta "Llamadas ordenadas por Empleado"
- `q.sql = "SELECT * FROM Datos ORDER BY Empleado"`  
La consulta "q" corresponde a la sentencia SQL: toda la tabla "Datos" pero ordenada según Empleado.

- Una cosa es crear una consulta por DAO y otra diferente acceder a una consulta ya creada ...

\* Inserta un nuevo botón de comando en el formulario **Mantenimiento** con:

Name: cmdVerConsulta  
Caption: Ver la Consulta

\* Escribe el siguiente código:

```
Private Sub cmdVerConsulta_Click()
 Dim x As Database, s As String
 Dim y As Recordset
 Dim z As QueryDef
 Set x = CurrentDb
 Set q = x.QueryDefs("Llamadas ordenadas por Empleado")
 Set y = q.OpenRecordset()
 y.MoveFirst
 Do Until y.EOF
 s = s & y("Id") & " - " & y("Empleado") & vbCrLf
 y.MoveNext
 Loop
 y.Close
 MsgBox s
End Sub
```

Ejecuta el programa anterior, para comprobar que funciona.

Observemos el **cmdVerConsulta\_Click**:

- **Set q=x.QueryDefs("Llamadas ordenadas por Empleado")**  
Asignamos a "q" la consulta "Llamadas ordenadas por Empleado"
- **Set y=q.OpenRecordset()**  
Asignamos a "y" el "recordset" correspondiente a la consulta "q"

- Vamos a "ordenar" pero de una forma más sofisticada ...

Crema un **autoformulario en Tabla**, para la tabla **Datos** y grábalo con el nombre **Datos**

Es decir:

Abre la tabla **Datos**  
Cinta de Opciones: Crear – Varios Elementos

Desde la pantalla de diseño del formulario **Datos**:

Inserta en la sección "Pie del formulario" un cuadro de lista, con las siguientes propiedades:

Nombre: IstOrdenaciones  
Tipo de origen: Lista de valores  
Origen de la fila: "Id", "DiaHora", "Empleado", "Elquellama",  
"Notas", "NombreEmpleado"  
Etiqueta: Ordenar por

\* Escribe el siguiente código:

```
Private Sub IstOrdenaciones_AfterUpdate()
 Dim x As Database
 Dim sql As String
 Dim y As QueryDef
 Set x = CurrentDb
 sql = "SELECT * FROM Datos ORDER BY [" & _
```

```

Forms![Datos]![IstOrdenaciones] & "]"
Set y = x.CreateQueryDef("orden", sql)
DoCmd.ApplyFilter "orden"
x.QueryDefs.Delete "orden"
End Sub

```

Ante todo, veamos si funciona, es decir:

- Ejecuta el formulario "Datos"
- En el cuadro de lista **IstOrdenaciones**, escoge el campo que quieras.
- Si todo funciona correctamente, el listado que aparece en el formulario, se ordena según el campo que hemos seleccionado en el cuadro de lista.

Estudiamos el procedimiento **IstOrdenaciones\_AfterUpdate**

- **IstOrdenaciones\_AfterUpdate**  
Después de actualizar el control "cuadro de lista"
- **sql = "SELECT \* FROM Datos ORDER BY [" & Forms![Datos] ![IstOrdenaciones] & "]"**  
Sentencia SQL que indica: Ordenar la tabla **Datos** según el valor del control **IstOrdenaciones**
- **y=x.CreateQueryDef("orden",sql)**  
Crea una consulta de nombre "orden", según la sentencia "sql"
- **DoCmd.ApplyFilter "orden"**  
En el formulario activo se ejecuta la consulta "orden"
- **x.QueryDefs.Delete "orden"**  
Se borra la consulta "orden" de la base de datos.

- Vamos a buscar registros que cumplan un determinado criterio ...

Inserta un nuevo botón de comando en el formulario **Mantenimiento** con las características:

```

Name: cmdCriterio
Caption: Listado según Criterio

```

Escribe el siguiente código:

```

Private Sub cmdCriterio_Click()
Dim bd As Database
Dim rs As Recordset
Dim salida As String, crit As String
Set bd = CurrentDb
Set rs = bd.OpenRecordset("Datos", dbOpenDynaset)
crit = "[Id]>2"
rs.FindFirst crit
Do Until rs.NoMatch
 salida = salida & rs("Id") & " - " & rs("Empleado") & vbCrLf
 rs.FindNext crit
Loop
rs.Close
MsgBox salida
End Sub

```

Veamos si funciona, es decir:

- "Abre" el formulario **Mantenimiento**
- Clic en [Listado según criterio]

- Si todo funciona correctamente, aparece el listado de "Id" y "Empleado", correspondientes a un **Id>2**

Estudiamos el programa:

- **Set rs=bd.OpenRecordset("Datos",dbOpenDynaset)**  
El recordset debe abrirse en modo "Dynaset", para poder utilizar el método **Find**
- **crit = "[Id] > 2"**  
Asignamos a la variable **crit** el criterio que deseemos, en nuestro caso, que el campo [Id] sea superior a 2.
- **rs.FindFirst crit**  
Nos situamos en el primer registro que cumple la condición **crit**
- **Do Until rs.NoMatch**  
    **(Sentencia1)**  
    **rs.FindNext crit**

#### Loop

Mientras los registros vayan cumpliendo la condición "crit", se irá ejecutando la **sentencia1**, en cada registro que la cumpla.

- Resulta que el formulario **Mantenimiento** lo utilizan varias personas y nos gustaría tener un "registro" de todos los que abren el formulario ...

- \* Crea una nueva tabla en la B.D. LLAMADAS de nombre **Registro** y estructura:

| <b>Campos</b> | <b>Tipo</b>                    |
|---------------|--------------------------------|
| Reg           | Autonumérico y Clave Principal |
| Usuario       | Texto                          |
| DiaHora       | Fecha/Hora                     |

- \* Escribe el siguiente código, correspondiente al abrir el formulario **Mantenimiento**:

```

Private Sub Form_Load()
 Dim bd As Database
 Dim r As Recordset
 Dim nom As String
 Set bd = CurrentDb
 Set r = bd.OpenRecordset("SELECT * FROM Registro")
 nom = InputBox("Escribe tus datos")
 r.AddNew
 r("Usuario") = nom
 r("DiaHora") = Now
 r.Update
 r.Close
End Sub

```

Observa que en el "recordset" escribimos directamente una sentencia SQL, que no es más que la tabla **Registro**.

Prueba el programa, es decir abre varias veces el formulario **Mantenimiento**, y analiza lo que sucede en la tabla **Registro**

- Podemos utilizar la "filosofía" del programa anterior para simular una "Demo", es decir, nos gustaría que el formulario **Datos**, por ejemplo, sólo se pudiera utilizar 5 días ...

Crea una nueva tabla de nombre **Control**, en **LLAMADAS** con la estructura:

| <b>Campos</b> | <b>Tipo</b>                |
|---------------|----------------------------|
| numdia        | Númérico y Clave Principal |
| fecha         | Fecha/Hora                 |

\* Escribe el siguiente código, correspondiente al abrir el formulario **Datos**:

```
Private Sub Form_Load()
 Dim bd As Database
 Dim registros As Recordset
 Dim mensaje As String, tit As String
 Set bd = CurrentDb
 Set registros = bd.OpenRecordset("SELECT * FROM Control")
 If registros.RecordCount = 0 Then
 registros.AddNew
 registros("numdia") = 1
 registros("fecha") = Date
 registros.Update
 Else
 If registros("numdia") >= 5 Then
 mensaje = "Se han superado los 5 días de la demo"
 tit = "Demo"
 MsgBox mensaje, 48, tit
 DoCmd.Quit
 Else
 If registros("fecha") <> Date Then
 registros.Edit
 registros("numdia") = registros("numdia") + 1
 registros("fecha") = Date
 registros.Update
 End If
 End If
 End If
 registros.Close
End Sub
```

Prueba el funcionamiento del programa, es decir:

- Abre el formulario "Datos"
- Investiga el contenido de la tabla **Control**.
- Para comprobar el funcionamiento del programa, deberás esperar 5 días o haz lo siguiente: En el único registro de la tabla **Control**, escribe: **5** en "numdia" y cualquier fecha que no sea la de hoy en "fecha".
- Vuelve a ejecutar el formulario **Datos**.

## Para saber más

### DAO

Los objetos de acceso a datos de Microsoft, comúnmente conocidos como DAO, nos permiten trabajar con datos, únicamente. No podremos crear un formulario utilizando DAO, aunque forme parte de la base de datos, ya que los formularios no son datos, sin embargo sí podemos crear una tabla.

De manera general, cualquier cosa que podamos hacer con tablas, con consultas o con los datos propiamente dichos podrá hacerse a través de DAO.

Podríamos crear una base de datos entera utilizando DAO y algunos de los procedimientos incorporados de Microsoft Access. Por supuesto, algunas tareas son más fáciles de hacer utilizando la interfaz de usuario, como por ejemplo: crear una consulta o establecer relaciones entre tablas. Sin embargo, algunas propiedades y métodos de los objetos DAO pueden resultar útiles, porque automatizan y mejoran las rutinas que resultan tediosas o redundantes a través de la interfaz de usuario.

### Espacios de trabajo

El DAO admite dos entornos diferentes de bases de datos o “espacios de trabajo”:

#### Los espacios de trabajo Microsoft Jet

Permiten tener acceso a datos en bases de datos Microsoft Jet (Access), orígenes de datos Microsoft conectados a ODBC y orígenes de datos ISAM instalable en otros formatos, como Paradox o Lotus 1 2 3

#### Los espacios de trabajo ODBCDirect

Permiten tener acceso a servidores de bases de datos mediante ODBC, sin cargar el motor de base de datos Microsoft Jet

### Uso del espacio de trabajo Microsoft Jet

#### Abrir una base de datos

Para abrir una base de datos, simplemente abrimos un **objeto Database**. Este objeto puede representar una base de datos Microsoft Jet (archivo .mdb), una base de datos ISAM (Paradox, por ejemplo) o una base de datos ODBC conectada mediante el motor de bases de datos Microsoft Jet (también conocido como una “base de datos ODBC conectada a Microsoft Jet”).

#### Manipulación de datos

El DAO proporciona un excelente conjunto de herramientas de manipulación de datos. Podemos crear un **objeto Recordset**, para consultar convenientemente una base de datos y manipular el conjunto de registros resultante. El **método OpenRecordset** acepta una cadena SQL o un nombre de un **objeto QueryDef** (consulta almacenada), como un argumento de origen de datos, o se puede abrir desde un **objeto QueryDef** o un **objeto TableDef**, utilizando este objeto como el origen de datos. El **objeto Recordset** resultante presenta un conjunto extremadamente rico de propiedades y métodos con el que examinar y modificar datos.

El objeto Recordset está disponible en varios tipos diferentes:

**Recordset tipo Table**

Solo pueden crearse de tablas Access locales (no vinculadas)

**Recordset tipo Dynaset**

Podemos utilizar tablas locales, vinculadas al igual que consultas y sentencias sql.

Tenemos que utilizar la constante **dbOpenDynaset** sólo al abrir una tabla local, porque un "dynaset" es el recordset predeterminado para todos los otros tipos de datos.

**Recordset Snapshot**

Igual que el tipo "dynaset", pero debemos establecer la constante **dbOpenSnapshot** porque un "snapshot" no es el tipo de recordset predeterminado, las otras diferencias respecto al tipo "dynaset", dependen de sus opciones.

## Autoevaluación 4

1) Modifica la **B.D. Eval3A**, que nos permitía repasar las tablas de multiplicar de la siguiente manera:

\* Cambiáale el nombre, por **Eval4A**

\* Crea en Eval4A una tabla de nombre **Notas** con la estructura:

| <b>Campos</b>    | <b>Tipo</b>                    |
|------------------|--------------------------------|
| Id               | Autonumérico y Clave Principal |
| Usuario          | Texto                          |
| Curso            | Texto                          |
| Fecha            | Fecha/Hora                     |
| Multiplicaciones | Numérico                       |
| Bien             | Numérico                       |
| Mal              | Numérico                       |
| Nota             | Texto                          |

\* El programa debería "registrar" en la tabla **Notas**, los datos correspondientes del usuario del formulario **Multiplicar**. De la siguiente forma:

\* Crea un formulario de nombre **Datos**, que debería ser el inicial, al abrir la B.D. Eval4A, que sirva para "recoger" los datos del usuario:

Datos : Formulario

**Escuela Virtual**

Escribe tu nombre:

Cuál es tu CURSO:

Tablas de Multiplicar

**Nombres:**

txtNom  
txtCurs

cmdMultiplicar

- Al hacer clic en [Tablas de Multiplicar], se registran los datos correspondientes en la tabla **Notas** y se abre el formulario **Multiplicar**
- Al cerrar el formulario **Multiplicar**, se acaban de registrar los datos correspondientes en la tabla **Notas**.
- Cambia el botón [Salir] del formulario **Multiplicar** por "Cerrar".

- 2) Crea en la **B.D. Eval4A** un formulario de nombre **Control**, que permita analizar los resultados de la tabla **Notas**, de la siguiente forma:

Formulario **Control**:

**Nombres:**

txtCnom  
txtCbien  
txtCnota  
txtCnum  
txtCmal  
cmdCotro

Escribimos en el cuadro de texto **txtCnom**, el nombre de un alumno (Usuario de la tabla **Notas**) y al salir de **txtCnom**, se rellenan automáticamente el resto de cuadros de texto del formulario **Control**.

Al hacer clic en [Otro Alumno], volvemos a empezar.

- 3) Mejora la selección del alumno en el formulario **Control** anterior, de la siguiente forma:

Crea un formulario de nombre **Alumnos** para la tabla **Notas**, con el siguiente aspecto aproximado:



| Usuario | Curso | Fecha   | Multipl | Bien | Mal | Nota       |
|---------|-------|---------|---------|------|-----|------------|
| Pepe    | 1     | 3:05:53 | 2       | 2    | 0   | Excelente  |
| Pepe    | 1     | 7:16:47 | 3       | 1    | 2   | Deficiente |
| Paco    | 2     | 7:17:54 | 4       | 2    | 2   | Suficiente |
| Pepe    | 2     | 7:18:35 | 5       | 3    | 2   | Bien       |
| Paco    | 2     | 7:19:03 | 6       | 3    | 3   | Suficiente |
| Pepe    | 1     | 7:19:05 | 6       | 3    | 3   | Suficiente |

**Nombres:**

cmdOk

cmdCancelar

**Código:**

```
Private Sub cmdCancelar_Click()
 Forms!Control.Tag = "Cancelar"
 DoCmd.Close
End Sub
```

```
Private Sub cmdOk_Click()
 Forms!Control.Tag = "Ok"
 Me.Visible = False
End Sub
```

Inserta en el formulario **Control** un botón de comando con las siguientes propiedades:

Name: cmdAlumnos  
Caption: Alumnos

Corrige el procedimiento **Form\_Load** del formulario **Control**, para que el foco se coloque en **cmdAlumnos**

Escribe el siguiente código:

```
Private Sub cmdAlumnos_Click()
 DoCmd.OpenForm "Alumnos", , , , acFormReadOnly, acDialog
 If Me.Tag = "Ok" Then
 [txtCnom] = Forms!Alumnos![Usuario]
 DoCmd.Close acForm, "Alumnos"
 End If
End Sub
```

Investiga el funcionamiento de lo que hemos hecho y el porqué (utiliza la ayuda del Visual Basic).

## Apéndice A

### Programación en SQL

---

El lenguaje de consulta estructurado (SQL) es un lenguaje de bases de datos normalizado, utilizado por el motor de bases de datos de Microsoft Jet.

#### Consultas de selección

Las consultas de selección se utilizan para indicar al motor de datos que devuelva información de las B.D., esta información es devuelta en forma de conjunto de registros que se pueden almacenar en un objeto recordset.

- \* Crea una nueva base de datos **Access 2003** de nombre **BDsql** en *TuCarpeta*
- \* Importa las tablas **Categorías, Productos y Proveedores** de la base de datos de ejemplo **Neptuno**. Es decir:

**Cinta de Opciones: Datos externos**  
**Importar**  
**Access**

Selecciona: **Neptuno.mdb** y [Importar]

- Selecciona las tablas: **Categorías, Productos y Proveedores** [Aceptar]
- Crea un formulario en blanco con las siguientes características:



- Grábalo con el nombre **ConsultasSQL**
- Nombre del Cuadro de texto: **txtSQL**
- Botones: **cmdEjecuta** y **cmdNueva**

- Escribe el siguiente código:

```

Private Sub cmdEjecuta_Click()
 Dim x As Database
 Dim sql As String
 Dim y As QueryDef
 Set x = CurrentDb
 sql = [txtSQL]
 Set y = x.CreateQueryDef("consulta", sql)
 DoCmd.OpenQuery "consulta", acNormal, acReadOnly
 x.QueryDefs.Delete "consulta"
End Sub

```

```

Private Sub cmdNueva_Click()
 [txtSQL] = ""
 [txtSQL].SetFocus
End Sub

```

- Observemos nuestro programa:
  - Declaramos la base de datos activa como "x"
  - Declaramos la variable "sql" de texto
  - Declaramos una consulta como "y"
  - Asignamos a la variable **sql**, el valor del campo **txtSQL** (donde escribiremos las sentencias SQL).
  - Creamos en la base de datos una consulta de nombre "consulta", según la sentencia SQL, que tendremos escrita en **txtSQL**
  - Abrimos la consulta (DoCmd.OpenQuery "consulta").
  - Borramos de la base de datos la consulta anterior.
- Está claro que utilizaremos el formulario anterior, para aprender a programar en SQL ...

- b) La sintaxis básica de una consulta de selección es la siguiente:
- ```
SELECT campo1, campo2, ... FROM nombreTabla
```

Vamos a ver si funciona:

- Ejecuta el formulario **ConsultasSQL**
 - En el cuadro de texto escribe:


```
SELECT NombreCategoría, Descripción FROM Categorías
```
 - Clic en [Ejecutar]
 - Si todo funciona correctamente aparecerá el listado de todas las categorías y descripción de las mismas.
 - "Cierra" la consulta
 - Clic en [Nueva]

Si deseamos **todos** los campos de la tabla, no es necesario escribirlos uno por uno después de la cláusula SELECT, basta escribir en su lugar un asterisco.

- Pruébalo, visualizando la consulta correspondiente a:


```
SELECT * FROM Categorías
```

b) Ordenar los registros

Podemos ordenar los registros utilizando la cláusula **ORDER BY**

- Listado de los “productos” según el campo **NombreProducto**:
Escribe en el **txtSQL**:
SELECT * FROM Productos ORDER BY NombreProducto
- Se pueden ordenar los registros por más de un campo:
 - Queremos el siguiente listado:
Campos: NombreCompañía, CargoContacto, País
Tabla: Proveedores
Ordenados según el País, en primer lugar y el Nombre de la Compañía en segundo
 - Pruébalo, utilizando la siguiente sentencia SQL:

```
SELECT NombreCompañía,CargoContacto,País
FROM Proveedores
ORDER BY País,NombreCompañía
```

Si quieres escribir en el cuadro de texto **txtSQL**, la sentencia en tres líneas, basta que pulses [Ctrl][Return], después de cada línea

- Incluso se puede especificar el orden de los registros: ascendente (cláusula **ASC**, se toma este valor por defecto) o descendiente (**DESC**)
 - Consigue el listado anterior, pero la ordenación por **NombreCompañía**, que sea descendente
 - Es decir:

```
SELECT NombreCompañía,CargoContacto,País
FROM Proveedores
ORDER BY País ASC, NombreCompañía DESC
```

c) Consultas con Predicado

Los posibles predicados son:

ALL: todos los campos de la tabla (equivalente a *)

TOP: devuelve un determinado número de registros de la tabla

DISTINCT: omite los registros cuyos campos seleccionados coincidan totalmente

DISTINCTROW: omite los registros duplicados basandose en la totalidad del registro y no sólo en los campos seleccionados.

No es conveniente abusar de ALL o *, ya que obligamos al motor de la base de datos a analizar la estructura de la tabla para averiguar los campos que contiene, es mucho más rápido indicar el listado de campos deseados.

- **TOP**: Devuelve un cierto número de registros que entran entre el principio y el final de un rango especificado por una cláusula **ORDER BY**.
 - Queremos los 15 primeros Productos (según el **IdProducto**), para los campos :
IdProducto, NombreProducto y PrecioUnidad.

Deberás utilizar la sentencia:

```
SELECT TOP 15 IdProducto,NombreProducto,PrecioUnidad
FROM Productos
ORDER BY IdProducto
```

Pruébalo a ver si es verdad.

- Queremos los 10 últimos **Productos** (según el **IdProducto**), para los campos: **IdProducto**, **NombreProducto** y **PrecioUnidad**.

Prueba la siguiente sentencia:

```
SELECT TOP 10 IdProducto,NombreProducto,PrecioUnidad  
FROM Productos  
ORDER BY IdProducto DESC
```

Se puede utilizar la palabra reservada **PERCENT** para devolver un cierto porcentaje de registros que caen al principio o al final de un rango especificado por la cláusula **ORDER BY**

- Queremos el 30% de las primers **Categorías**

Prueba la siguiente sentencia:

```
SELECT TOP 30 PERCENT IdCategoría, NombreCategoría  
FROM Categorías  
ORDER BY IdCategoría
```

- Queremos el 15% de las últimas **Categorías**

Prueba la siguiente sentencia:

```
SELECT TOP 15 PERCENT IdCategoría,NombreCategoría  
FROM Categorías  
ORDER BY IdCategoría DESC
```

- **DISTINCT**: omite los registros cuyos campos seleccionados coincidan totalmente.

- Queremos el listado de todos los proveedores de nuestros productos.

Utiliza la siguiente sentencia SQL:

```
SELECT DISTINCT IdProveedor  
FROM Productos
```

Compáralo con el resultado de la sentencia:

```
SELECT IdProveedor  
FROM Productos
```

- Queremos el listado de los países correspondientes a nuestros proveedores.

Prueba la sentencia:

```
SELECT DISTINCT País  
FROM Proveedores
```

Y compara el resultado de la misma sentencia, pero sin "DISTINCT"

d) Recuperar información de otra base de datos

Desde el formulario **ConsultasSQL** de la **B.D. BDsql**, nos gustaría visualizar los registros de los campos: **Empleado** y **Elquellama**, de la tabla **Datos** de la **B.D. LLAMADAS**

No hay ningún problema, siempre y cuando utilizemos la palabra reservada **IN**.

En efecto, prueba la siguiente sentencia SQL:

```
SELECT Empleados, Elquellama  
FROM Datos IN 'c:\TuCarpeta\LLAMADAS.mdb'
```

e) Criterios de Selección

En los apartados anteriores hemos visto la forma de recuperar **todos** los registros. Vamos a ver ahora la forma de recuperar los registros que cumplan unos criterios determinados.

Veamos en primer lugar la sintaxis que hemos de seguir en los "criterios":

1º) Al referirnos a un campo de texto, hemos de encerrarlo entre comillas simples.

2º) No es posible incluir campos de tipo **Memo**

3º) Las fechas se deben escribir siempre en formato **mm-dd-aa** y además la fecha debe ir encerrada entre almohadillas (#).

4º) Los operadores lógicos más comunes en SQL son: AND (y), OR (o) NOT (no).

- Listado de productos, cuyo campo **UnidadesEnExistencia** esté entre 20 y 60.

Utiliza la siguiente sentencia SQL:

```
SELECT * FROM Productos
WHERE UnidadesEnExistencia>20 AND UnidadesEnExistencia<60
ORDER BY UnidadesEnExistencia
```

- El mismo listado anterior, pero además deseamos los productos correspondientes a "36 cajas".

Prueba la siguiente sentencia:

```
SELECT * FROM Productos
WHERE (UnidadesEnExistencia>20 AND UnidadesEnExistencia<60)
OR CantidadPorUnidad='36 cajas'
ORDER BY UnidadesEnExistencia
```

Si todo funciona correctamente, aparecerá el mismo listado de antes, más un producto (Mezcla Gumbo del chef Anton) correspondiente a "36 cajas".

- Listado de proveedores que no son de Estados Unidos

Utiliza la siguiente sentencia:

```
SELECT NombreCompañía,Ciudad,País
FROM Proveedores
WHERE NOT País='Estados Unidos'
ORDER BY País
```

- Listado de productos de precio entre 30 y 100 y productos con unidades en existencia entre 10 y 75.

Prueba la siguiente instrucción SQL:

```
SELECT NombreProducto,PrecioUnidad,UnidadesEnExistencia
FROM Productos
WHERE (PrecioUnidad>30 AND PrecioUnidad<100)
OR (UnidadesEnExistencia>10 AND UnidadesEnExistencia<75)
ORDER BY PrecioUnidad DESC
```

Para indicar que deseamos recuperar los registros según el intervalo de valores de un campo, utilizamos el operador **Between** cuya sintaxis es:

Nombre del campo Between valor1 And valor2

La consulta correspondiente devolvería los registros que contengan en "Nombre del Campo" un valor incluido en el intervalo valor1, valor2 (ambos inclusive)

Podemos combinar los operadores **Not Between**

Repite las 3 sentencias SQL anteriores, pero utilizando el operador **Between**, es decir:

```
SELECT * FROM Productos
WHERE UnidadesEnExistencia Between 20 And 60
ORDER BY UnidadesEn Existencia
```

```
SELECT * FROM Productos  
WHERE (UnidadesEnExistencia Between 20 And 60)  
OR CantidadPorUnidad='36 cajas'  
ORDER BY UnidadesEnExistencia
```

```
SELECT NombreProducto,PrecioUnidad,UnidadesEnExistencia  
FROM Productos  
WHERE (PrecioUnidad Between 30 And 100)  
OR (UnidadesEnExistencia Between 10 And 75)  
ORDER BY PrecioUnidad DESC
```

Observa que al utilizar el operador **Between**, los dos valores del intervalo se encuentran incluidos en la condición.

- El operador **Like**

Se utiliza para comparar una expresión de cadena con un modelo en una expresión SQL. Su sintaxis es:

expresión Like modelo

Ejemplos de "modelo":

Like 'P[A-F]*'

Devolverá los datos que comienzan con la letra P seguida de cualquier letra entre A y F y de cualquier cadena a continuación (*)

Prueba la siguiente sentencia:

```
SELECT Descripción FROM Categorías  
WHERE Descripción Like 'P[A-F]*'
```

Like 'P*'

Devolverá los datos que comienzan con la letra P

Prueba la sentencia:

```
SELECT Descripción FROM Categorías  
WHERE Descripción Like 'P*'
```

Like '*dulce*'

Devolverá los datos que contienen la palabra **dulce**

Prueba la sentencia:

```
SELECT Descripción FROM Categorías  
WHERE Descripción Like '*dulce*'
```

Like '(???)*'

Devolverá los datos que empiezan con **tres** caracteres entre paréntesis.

Puébalo con la siguiente sentencia:

```
SELECT Teléfono FROM Proveedores  
WHERE Teléfono Like '(???)*'
```

El operador **In**

Este operador devuelve aquellos registros cuyo campo indicado coincide con alguno de los valores de la lista.

* Queremos todos los proveedores de Estados Unidos, Alemania y España
Considera la siguiente sentencia:

```
SELECT NombreCompañía,País  
FROM Proveedores  
WHERE País In('Estados Unidos','Alemania','España')  
ORDER BY País
```

* Queremos todos los proveedores que no sean de Estados Unidos ni de Alemania.
Prueba la siguiente sentencia SQL:

```
SELECT NombreCompañía,País  
FROM Proveedores  
WHERE País Not In('Estados Unidos','Alemania')  
ORDER BY País
```

Para acabar con este apartado prueba las siguientes sentencias SQL:

```
SELECT NombreCompañía,CargoContacto  
FROM Proveedores  
WHERE CargoContacto Like '*ventas*'
```

```
SELECT NombreProducto,UnidadesEnExistencia  
FROM Productos  
WHERE UnidadesEnExistencia Not Between 53 And 120  
ORDER BY UnidadesEnExistencia
```

f) Agrupamiento de Registros

La cláusula **GROUP BY** combina los registros con valores idénticos, en la lista de campos especificados, en un único registro. Para cada registro se crea un valor sumario si se incluye una función SQL agregada, como por ejemplo **Sum** o **Count**.

Su sintaxis es:

```
SELECT Lista de campos FROM Nombre de la tabla  
WHERE criterio  
GROUP BY campos del grupo
```

Queremos determinar el total de **Unidades en Existencia**, que tenemos por cada una de las categorías.

Prueba la siguiente sentencia:

```
SELECT IdCategoría, Sum(UnidadesEnExistencia)  
FROM Productos  
GROUP BY IdCategoría
```

El operador **HAVING**

Es similar a **WHERE**, determina qué registros se seleccionan. Una vez que los registros se han agrupado utilizando **GROUP BY**, **HAVING** determina cuales de ellos se van a mostrar.

Ejecuta la siguiente sentencia SQL:

```
SELECT IdCategoría, Sum(UnidadesEnExistencia)  
FROM Productos  
GROUP BY IdCategoría  
HAVING Sum(UnidadesEnExistencia)>350  
And Sum(UnidadesEnExistencia)<600
```


AVG

Calcula la media aritmética de un conjunto de valores contenidos en un campo determinado.

Queremos el promedio de los precios de nuestros productos, por encima de 30.

Ejecuta la sentencia:

```
SELECT Avg(PrecioUnidad) AS PromedioMas30  
FROM Productos  
WHERE PrecioUnidad>30
```

Compara el resultado con la consulta:

```
SELECT Avg(PrecioUnidad) As Promedio  
FROM Productos
```

Count

Calcula el número de registros

Ejecuta la siguiente sentencia SQL:

```
SELECT Count(*) AS Total  
FROM Productos
```

Si como argumento de la función **Count**, especificamos campos, la función **Count** cuenta un registro sólo si al menos uno de los campos no es **Null**. Si todos los campos especificados son **Null**, no se cuenta el registro. Hay que separar los nombres de los campos con el símbolo del ampersand (&).

Veamos:

Número de registros de Proveedores:

```
SELECT Count(*) AS Todos  
FROM Proveedores
```

Número de registros de Proveedores que tienen "Región":

```
SELECT Count(Región) AS Regiones  
FROM Proveedores
```

Número de registros de Proveedores que tienen "Región" o "Fax":

```
SELECT Count(Región&Fax) AS RegFax  
FROM Proveedores
```

Max, Min

Devuelven el máximo o mínimo de un conjunto de valores

Localización del máximo pedido:

```
SELECT Max(UnidadesEnPedido) AS MasPedi  
FROM Productos
```

Localización del máximo pedido de precio superior a 40:

```
SELECT Max(UnidadesEnPedido) AS Mas40  
FROM Productos  
WHERE PrecioUnidad>40
```

Sum

Devuelve la suma del conjunto de valores

Precio de todas las existencias:

```
SELECT Sum(PrecioUnidad*UnidadesEnExistencia) AS Total
FROM Productos
```

g) Consultas de Acción

Las consultas de acción son aquellas que no devuelven ningún registro, son las encargadas de acciones como añadir, borrar y modificar registros.

- DELETE

Crea una consulta de eliminación, que borra los registros que cumplen una cláusula WHERE.

Localiza los proveedores correspondientes a un “cargo de contacto” de “Gerente de ventas”. Es decir:

```
SELECT * FROM Proveedores
WHERE CargoContacto='Gerente de ventas'
```

Elimina los registros anteriores. Es decir:

```
DELETE * FROM Proveedores
WHERE CargoContacto='Gerente de ventas'
```

Antes deberás eliminar la relación que tenemos entre las tablas **Productos y Proveedores**

Comprueba que ha funcionado la sentencia SQL

INSERT INTO

Agrega un registro en una tabla. Se la conoce como una consulta de datos añadidos. Esta consulta puede ser de dos tipos: insertar un único registro o Insertar en una tabla los registros contenidos en otra tabla.

Para insertar un único registro la sintaxis es:

```
SELECT INTO Tabla(campo1,campo2,...campoN)
VALUES(valor1,valor2,...,valorN)
```

Hay que prestar especial atención a acotar entre comillas simples, los valores literales (cadenas de caracteres) y las fechas indicárlas en formato **mm-dd-aa** y entre caracteres de almohadillas (#).

Vamos a insertar un registro a la tabla Proveedores

Ejecuta la siguiente sentencia SQL:

```
INSERT INTO Proveedores(NombreCompañía,NombreContacto)
VALUES ('Pepe','Ambrosio')
```

Comprueba, investigando el contenido de **Proveedores**, que ha funcionado.

Para insertar registros de otra tabla, la sintaxis es:

```
INSERT INTO Tabla [IN base_externa](campo1,campo2,...)
SELECT TablaOrigen.campo1,TablaOrigen.campo2,...
FROM TablaOrigen
```

La condición SELECT puede incluir la cláusula WHERE para filtrar los registros a copiar.

Si “Tabla” y “TablaOrigen” tienen la misma estructura podemos simplificar la sintaxis a:

```
INSERT INTO Tabla SELECT TablaOrigen.* FROM TablaOrigen
```

Vamos a probarlo:

Ejecuta la siguiente sentencia SQL:

```
INSERT INTO Proveedores(NombreCompañía,NombreContacto)  
SELECT Categorías.NombreCategoría, Categorías.Descripción  
FROM Categorías
```

Comprueba lo que ha sucedido, investigando el contenido de la tabla **Proveedores**

UPDATE

Crea una consulta de actualización que cambia los valores de los campos de una tabla especificada basándose en una condición. Su sintaxis es:

```
UPDATE Tabla SET Campo1=valor1, Campo2=valor2,...  
WHERE Criterio
```

UPDATE no genera ningún resultado. Para saber qué registros se van a cambiar, hay que examinar primero el resultado de una consulta de selección que utilice el mismo criterio y después ejecutar la consulta de actualización.

- Cambia todos los registros de **Productos**, que tienen cero en el campo **UnidadesEnExistencia** por 2.

Deberás utilizar la siguiente sentencia:

```
UPDATE Productos SET UnidadesEnExistencia=2  
WHERE UnidadesEnExistencia=0
```

Comprueba el resultado.

- Aumenta en 3 unidades el campo **UnidadesEnExistencia** de Productos, siempre y cuando dicho campo sea superior a 2 e inferior a 5. Es decir, ejecuta la siguiente sentencia:

```
UPDATE Productos SET UnidadesEnExistencia=  
UnidadesEnExistencia + 3  
WHERE UnidadesEnExistencia Between 2 And 5
```

Comprueba el resultado.

- Aumenta el precio de todos los productos en un 10%. Es decir, deberás ejecutar:

```
UPDATE Productos SET PrecioUnidad=PrecioUnidad*1.1
```

Comprueba el resultado.

h) Consultas con Parámetros

Las consultas con parámetros son aquellas cuyas condiciones de búsqueda se definen mediante parámetros. Si se ejecutan directamente desde la base de datos donde han sido definidas aparecerá un mensaje solicitando el valor de cada uno de los parámetros. Si deseamos ejecutarlas desde una aplicación hay que asignar primero el valor de los parámetros y después ejecutarlas. Su sintaxis es la siguiente:

```
PARAMETERS nombre1, tipo1, ... Consulta SQL
```

- Queremos el listado de proveedores según el país.

Ejecuta la sentencia siguiente:

```
PARAMETERS [Escribe el país] Text;  
SELECT * FROM Proveedores  
WHERE País=[Escribe el país];
```

Observa el uso del punto y coma, que separa los **PARAMETERS** de la **consulta SQL**

Apendice B

Excel, VBA y Bases de Datos

Ejecuta el **Excel** y
Botón del Office
Opciones de Excel
Configuración de macros
Habilitar todas las macros (no recomendado...)

Si en la **Cinta de Opciones**, no aparece una ficha de nombre **Programador**, haz lo siguiente:
Botón del Office
Opciones de Excel
Más frecuentes
Activa la opción:
Mostrar ficha Programador en la Cinta de Opciones.

Activar la Referencia DAO

a) Sitúate en el **Editor de Visual Basic**, de la siguiente forma:
Cinta de Opciones: Programador
Visual Basic

Haz lo siguiente:

Menú Herramientas
Referencias
Activa la referencia "Microsoft DAO 3.6 Object Library"
[Aceptar]

Lo que acabamos de hacer es necesario si queremos trabajar con bases de datos "access" desde código de VBA

Crear una base de datos

b) Sitúate en la **Hoja1** e inserta un botón de comando de la siguiente forma:
Cinta de Opciones: Programador
Grupo: Controles
Clic en la flechita de **Insertar**
Inserta un botón de comando (control **ActiveX**)

Clic en **Propiedades** y escribe como propiedad "Caption" = **Creación de una Base de Datos**)

Cinta de Opciones: Programador
Grupo: Controles1
Ver Código

Escribe:

```

Private Sub CommandButton1_Click()
    Dim bd As Database
    Dim tau As TableDef
    Dim cam As Field
    Set bd = CreateDatabase("C:\TuCarpeta\Agenda.mdb", dbLangSpanish)
    Set tau = bd.CreateTableDef("Amigos")
    Set cam = tau.CreateField("Nombre", dbText, 40)
    tau.Fields.Append cam
    Set cam = tau.CreateField("Dirección", dbText, 60)
    tau.Fields.Append cam
    Set cam = tau.CreateField("Teléfono", dbText, 15)
    tau.Fields.Append cam
    bd.TableDefs.Append tau
    bd.Close
End Sub

```

- Graba el libro de cálculo como **Excel46** en *TuCarpeta* (Tipo: **Libro de Excel habilitado para macros**)

- Ejecuta el procedimiento anterior

Si todo funciona correctamente, acabamos de crear una base de datos de nombre **Agenda.mdb** (en *TuCarpeta*), que contiene una tabla de nombre **Amigos** con los campos: Nombre (Texto, anchura = 40), Dirección (texto, 60) y Teléfono (texto, 15).

Comprueba si es verdad ejecutando el **Microsoft Access**

- Sitúate en el Editor de Visual Basic, para estudiar nuestro procedimiento **CommandButton1_Click()**
Observa:

* **Dim bd As Database**

Declaramos una base de datos de nombre **bd**

* **Dim tau As TableDef**

Declaramos una tabla de nombre **tau**

* **Dim cam As Field**

Declaramos un campo de nombre **cam**

* **Set bd = CreateDatabase("Agenda.mdb", dbLangSpanish)**

Creamos una base de datos de nombre **bd** que contiene el fichero "access": **Agenda.mdb**.

* **Set tau = bd.CreateTableDef("Amigos")**

Creamos en la base de datos **bd** (Agenda.mdb) una tabla de nombre **tau** (**Amigos** en el fichero Agenda.mdb)

* **Set cam = tau.CreateField("Nombre", dbtEXT, 40)**

Creamos en la tabla **tau** (Amigos) un campo de nombre **cam** (Nombre) de texto con una anchura de 40 caracteres.

* Procedemos de la misma forma para definir los campos "Dirección" y "Teléfono"

* **bd.Close**

Cerramos la base de datos. Es importante no dejar ninguna base de datos "abierta", ya que en caso contrario se producirían errores indeseables.

- En definitiva, nuestro programa:

* Crea una base de datos "access": **Agenda.mdb**

* Con una tabla: **Amigos**

* Que consta de 3 campos:

Nombre	Texto	40 (anchura en caracteres)
Dirección	Texto	60
Teléfono	Texto	15

- Inserta en la **Hoja1** otro botón de comando (Control ActiveX) de propiedad **caption = RevisArti**, y código:

```

Private Sub CommandButton2_Click()
    Dim bd As Database
    Dim tau1 As TableDef
    Dim tau2 As TableDef
    Dim cam1 As Field
    Dim cam2 As Field
    Set bd = CreateDatabase("C:\TuCarpeta\RevisArti.mdb", dbLangSpanish)
    Set tau1 = bd.CreateTableDef("Revistas")
    Set cam1 = tau1.CreateField("CodRev", dbText, 5)
    tau1.Fields.Append cam1
    Set cam1 = tau1.CreateField("Revista", dbText, 25)
    tau1.Fields.Append cam1
    Set cam1 = tau1.CreateField("Tipo", dbText, 25)
    tau1.Fields.Append cam1
    Set cam1 = tau1.CreateField("Precio", dbDouble)
    tau1.Fields.Append cam1
    Set cam1 = tau1.CreateField("Arti", dbText, 5)
    tau1.Fields.Append cam1
    bd.TableDefs.Append tau1
    Set tau2 = bd.CreateTableDef("Articulos")
    Set cam2 = tau2.CreateField("CodArt", dbText, 5)
    tau2.Fields.Append cam2
    Set cam2 = tau2.CreateField("Articulo", dbText, 25)
    tau2.Fields.Append cam2
    Set cam2 = tau2.CreateField("Tematica", dbText, 25)
    tau2.Fields.Append cam2
    bd.TableDefs.Append tau2
    bd.Close
End Sub

```

- Graba de nuevo el libro de cálculo con el mismo nombre **Excel46**

- Haz clic en [RevisArti] para ejecutar el último procedimiento.

Investiga el contenido de la base de datos access: **RevisArti.mdb** que tienes en *TuCarpeta*

Introducción de nuevos registros

d) Inserta en la **Hoja1** otro botón de comando de propiedad **caption = Nuevos Amigos**, y código:

```

Private Sub CommandButton3_Click()
    UserForm1.Show
End Sub

```

- Inserta un formulario (UserForm1) de contenido:

- Código (UserForm1):

```
Private Sub CommandButton1_Click()
    Dim bd As Database
    Dim reg As Recordset
    Set bd = OpenDatabase("C:\TuCarpeta\Agenda.mdb")
    Set reg = bd.OpenRecordset("Select * From Amigos",
    dbOpenDynaset)
    reg.AddNew
    reg("Nombre") = TextBox1.Text
    reg("Dirección") = TextBox2.Text
    reg("Teléfono") = TextBox3.Text
    reg.Update
    bd.Close
    TextBox1.Text = ""
    TextBox2.Text = ""
    TextBox3.Text = ""
    TextBox1.SetFocus
End Sub
```

- Graba el libro de cálculo con el mismo nombre **Excel46**

- Prueba el “programa” anterior, es decir:

- * Clic en el botón [Nuevos Amigos] de la **Hoja1**
- * Inventa un par o tres de registros
- * Comprueba desde el “Access” el contenido de la tabla **Amigos** de **Agenda.mdb**

- Veamos:

* **Dim reg As Recordset**

Declaramos un grupo de registros de nombre **reg**.

* **reg = bd.OpenRecordset(“Select * From Amigos”, dbOpenDynaset)**

Definimos el recordset “reg”, como los registros de la tabla **Amigos**, para ello utilizamos una sentencia **SQL** (Select * From Amigos).

dbOpenDynaset es un tipo de “recordset”, que nos permite tomar datos de una o más tablas relacionadas, y además nos permite la actualización de los datos.

* Observa el proceso a seguir para introducir registros: llamamos al método **AddNew** (reg.AddNew) que añade un registro en blanco. En segundo lugar, asignamos valores a los diferentes campos y por “último” el método **Update** (reg.Update), “escribe” los nuevos datos.

e) En lugar de un formulario con sus controles, podemos utilizar **celdas** de la hoja de cálculo.

En efecto:

- Sitúate en la **Hoja2** y escribe e inserta:

	A	B	C	D	E
1	Base de Datos = RevisArti.mdb				
2					
3	Tabla = Articulos		Tabla = Revistas		
4	CodArt:	<input type="text"/>	CodRev:	<input type="text"/>	
5	Articulo:	<input type="text"/>	Revista:	<input type="text"/>	
6	Tematica:	<input type="text"/>	Tipo:	<input type="text"/>	
7			Tprecio:	<input type="text"/>	
8			Arti:	<input type="text"/>	
9					
10	Grabar Artículo		Grabar Revista		
11					
12					

- Accede al **Módulo de la Hoja2** y escribe los siguientes procedimientos:

```
Private Sub CommandButton1_Click()  
    Dim bd As Database  
    Dim r As Recordset  
    Set bd = OpenDatabase("c:\TuCarpeta\RevisArti.mdb")  
    Set r = bd.OpenRecordset("Select * from Articulos", dbOpenDynaset)  
    r.AddNew  
    r("CodArt") = Hoja2.Cells(4, 2).Value  
    r("Articulo") = Hoja2.Cells(5, 2).Value  
    r("Tematica") = Hoja2.Cells(6, 2).Value  
    r.Update  
    bd.Close  
    Range("B4:B6").Select  
    Selection.ClearContents  
    Range("B4").Select  
End Sub  
  
Private Sub CommandButton2_Click()  
    Dim bd As Database  
    Dim r As Recordset  
    Set bd = OpenDatabase("c:\TuCarpeta\RevisArti.mdb")  
    Set r = bd.OpenRecordset("Select * from Revistas", dbOpenDynaset)  
    r.AddNew  
    r("CodRev") = Hoja2.Cells(4, 4).Value  
    r("Revista") = Hoja2.Cells(5, 4).Value  
    r("Tipo") = Hoja2.Cells(6, 4).Value  
    r("Precio") = Hoja2.Cells(7, 4).Value  
    r("Arti") = Hoja2.Cells(8, 4).Value  
    r.Update  
    bd.Close  
    Range("D4:D8").Select  
    Selection.ClearContents  
    Range("D4").Select  
End Sub
```

- Prueba el funcionamiento de los dos botones de la **Hoja2** y graba el libro de cálculo como **Excel46**

Navegación por una Base de Datos

f) Inserta en la **Hoja1** otro botón de comando de propiedad **Caption = Navegación por la Agenda.mdb**, y código:

```
Private Sub CommandButton4_Click()  
    UserForm2.Show  
End Sub
```


- Inserta un nuevo formulario (UserForm2) de contenido:

- Botones de nombre:

- * Nuevo Registro = cmdNuevoRegistro
- * Grabar Registro = cmdGrabarRegistro
- * << = cmdPrimerRegistro
- * < = cmdAnteriorRegistro
- * > = cmdSiguiente Registro
- * >> = cmdUltimoRegistro

- Accede al **Módulo del UserForm2** y escribe el siguiente código:

```
Dim bd As Database
Dim reg As Recordset
Dim x As Integer
' La variable x, contendrá el número total de registros
Dim num As Integer
' La variable num nos dará el número de registro actual

Private Sub cmdAnteriorRegistro_Click()
    If num = 1 Then
        MsgBox "No hay ninguno antes de este"
        Exit Sub
    Else
        reg.MovePrevious
        num = num - 1
        TextBox1.Text = reg("Nombre")
        TextBox2.Text = reg("Dirección")
        TextBox3.Text = reg("Teléfono")
    End If
End Sub

Private Sub cmdGrabarRegistro_Click()
    reg.MoveLast
    reg.AddNew
    reg("Nombre") = TextBox1.Text
    reg("Dirección") = TextBox2.Text
    reg("Teléfono") = TextBox3.Text
```

```
reg.Update
bd.Close
UserForm_Initialize
cmdUltimoRegistro_Click
End Sub

Private Sub cmdNuevoRegistro_Click()
    TextBox1.Text = ""
    TextBox2.Text = ""
    TextBox3.Text = ""
    TextBox1.SetFocus
End Sub

Private Sub cmdPrimerRegistro_Click()
    If num = 1 Then
        MsgBox "Este es el primer registro"
        Exit Sub
    Else
        reg.MoveFirst
        num = 1
        TextBox1.Text = reg("Nombre")
        TextBox2.Text = reg("Dirección")
        TextBox3.Text = reg("Teléfono")
    End If
End Sub

Private Sub cmdSiguienteRegistro_Click()
    If num = x Then
        MsgBox "Este es el último registro"
        Exit Sub
    Else
        num = num + 1
        reg.MoveNext
        TextBox1.Text = reg("Nombre")
        TextBox2.Text = reg("Dirección")
        TextBox3.Text = reg("Teléfono")
    End If
End Sub

Private Sub cmdUltimoRegistro_Click()
    If num = x Then
        MsgBox "Este es el último registro"
        Exit Sub
    Else
        reg.MoveLast
        num = x
        TextBox1.Text = reg("Nombre")
        TextBox2.Text = reg("Dirección")
        TextBox3.Text = reg("Teléfono")
    End If
End Sub

Private Sub UserForm_Initialize()
    Set bd = OpenDatabase("c:\TuCarpeta\Agenda.mdb")
    Set reg = bd.OpenRecordset("Select * from Amigos", dbOpenDynaset)
    reg.MoveLast
    x = reg.RecordCount
    num = 1
End Sub
```

```

reg.MoveFirst
TextBox1.Text = reg("Nombre")
TextBox2.Text = reg("Dirección")
TextBox3.Text = reg("Teléfono")
End Sub

Private Sub UserForm_Terminate()
    bd.Close
End Sub

```

- Prueba el funcionamiento de lo que acabamos de hacer y graba el libro de cálculo con el mismo nombre **Excel46**. Está claro que:

MoveFirst = Nos situamos en el primer registro
MoveNext = Nos situamos en el siguiente registro
MovePrevious = Nos situamos en el anterior registro
MoveLast = Nos situamos en el último registro
RecordCount = "Cuenta" los registros del Recordset.

Búsqueda de un registro

g) Inserta en la **Hoja1** otro botón de comando de propiedad **caption = Búsqueda de un registro**, y código:

```

Private Sub
CommandButton5_Click()
    UserForm3.Show
End Sub

```

- Inserta un nuevo formulario (UserForm3) de contenido:

- Accede al **Módulo del UserForm3** y escribe el siguiente código:

```

Dim bd As Database
Dim reg As Recordset
Dim criterio As String

Private Sub CommandButton1_Click()
    criterio = "Nombre=" & TextBox1.Text & ""
    reg.FindFirst criterio

```

```

    TextBox2.Text = reg("Nombre")
    TextBox3.Text = reg("Dirección")
    TextBox4.Text = reg("Teléfono")
End Sub

Private Sub CommandButton2_Click()
    reg.FindNext criterio
    TextBox2.Text = reg("Nombre")
    TextBox3.Text = reg("Dirección")
    TextBox4.Text = reg("Teléfono")
End Sub

Private Sub UserForm_Initialize()
    Set bd = OpenDatabase("c:\TuCarpeta\Agenda.mdb")
    Set reg = bd.OpenRecordset("Select * from Amigos", dbOpenDynaset)
End Sub

Private Sub UserForm_Terminate()
    bd.Close
End Sub

```

- Prueba el funcionamiento de lo que acabamos de hacer y graba el libro de cálculo con el mismo nombre **Excel46**.

Modificar y Borrar registros

h) Inserta en la **Hoja1** otro botón de comando de propiedad **caption = Modificar y/o Borrar** y código:

```

Private Sub CommandButton6_Click()
    UserForm4.Show
End Sub

```

- Inserta un nuevo formulario (UserForm4) de contenido:

- Accede al **Módulo del UserForm4** y escribe el siguiente código:

```

Dim bd As Database
Dim reg As Recordset
Dim criterio As String

Private Sub CommandButton1_Click()

```

```

criterio = "Nombre=" & TextBox1.Text & ""
reg.FindFirst criterio
TextBox2.Text = reg("Nombre")
TextBox3.Text = reg("Dirección")
TextBox4.Text = reg("Teléfono")
End Sub

Private Sub CommandButton2_Click()
reg.FindNext criterio
TextBox2.Text = reg("Nombre")
TextBox3.Text = reg("Dirección")
TextBox4.Text = reg("Teléfono")
End Sub

Private Sub CommandButton3_Click()
reg.Edit
reg("Nombre") = TextBox2.Text
reg("Dirección") = TextBox3.Text
reg("Teléfono") = TextBox4.Text
reg.Update
End Sub

Private Sub CommandButton4_Click()
reg.Delete
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ""
End Sub

Private Sub UserForm_Initialize()
Set bd = OpenDatabase("c:\TuCarpeta\Agenda.mdb")
Set reg = bd.OpenRecordset("Select * from Amigos", dbOpenDynaset)
End Sub

Private Sub UserForm_Terminate()
bd.Close
End Sub

```

- Prueba el funcionamiento de lo que acabamos de hacer y graba el libro de cálculo con el mismo nombre **Excel46**.

Listado de registros

i) Inserta en la **Hoja1** otro botón de comando de propiedad **caption = Listado de Amigos** y código:

```

Private Sub CommandButton7_Click()
Dim i As Integer
Dim bd As Database
Dim r As Recordset
Set bd = OpenDatabase("c:\TuCarpeta\Agenda.mdb")
Set r = bd.OpenRecordset("Select * from Amigos", dbOpenDynaset)
i = 3
Do While Not r.EOF
Hoja3.Cells(i, 1).Value = r("Nombre")
Hoja3.Cells(i, 2).Value = r("Dirección")
Hoja3.Cells(i, 3).Value = r("Teléfono")
r.MoveNext

```

```

    i = i + 1
Loop
bd.Close
End Sub

```

- Prueba el funcionamiento de lo que acabamos de hacer y graba el libro de cálculo con el mismo nombre **Excel46**.

j) Debes tener la base de datos **Neptuno.mdb**, copiada en *TuCarpeta*

- Crea un nuevo libro de cálculo. Sitúate en el Editor de Visual Basic y

Menú Herramientas

Referencias

Activa la referencia: **“Microsoft DAO 3.6 Object Library”**

[Aceptar]

- Sitúate en la **Hoja1** del libro y escribe:

	A	B	C	D
1	Base de Datos Neptuno.mdb			
2	Todas las Tablas	Tablas	Campos	Listado
3				
4	Todas las Tablas	Tablas	Tabla:	
5				
6				

- Accede al **Módulo de la Hoja1** y escribe el código correspondiente al primer botón [Todas las Tablas]:

```

Private Sub CommandButton1_Click()
    Dim i As Byte
    Dim bd As Database
    Dim tau As TableDef
    Set bd = OpenDatabase("c:\TuCarpeta\Neptuno.mdb")
    i = 5
    For Each tau In bd.TableDefs
        Cells(i, 1).Value = tau.Name
        i = i + 1
    Next
End Sub

```

- Ejecuta el procedimiento anterior: Observa que las tablas correspondientes al sistema empiezan por las letras: **MSys**

- Vuelve al **Módulo de la Hoja1** y escribe el procedimiento:

```

Private Sub CommandButton2_Click()
    Dim i As Byte
    Dim bd As Database
    Dim tau As TableDef
    Set bd = OpenDatabase("c:\TuCarpeta\Neptuno.mdb")
    i = 5
    For Each tau In bd.TableDefs
        If Left(tau.Name, 4) <> "MSys" Then
            Cells(i, 2).Value = tau.Name
            i = i + 1
        End If
    Next
End Sub

```

```

    End If
  Next
End Sub

```

- Ejecuta el procedimiento anterior y graba el libro de cálculo como **Excel47**.

- Veamos:

For Each tau In bd.TableDefs

Sentencia 1

Sentencia 2

...

Next

Para cada una de las tablas de la base de datos **bd**, se repetirá la ejecución de las sentencias 1, 2,

Cells(i,2).Value = tau.Name

Escribirá en la fila "i" columna "2" el nombre de la tabla

- Sitúate en el **Módulo de la Hoja1** y escribe el procedimiento:

```

Private Sub CommandButton3_Click()
  Dim i As Byte
  Dim bd As Database
  Dim tau As TableDef
  Dim cam As Field
  Set bd = OpenDatabase("c:\TuCarpeta\Neptuno.mdb")
  Set tau = bd.TableDefs(Cells(4, 4).Value)
  i = 5
  For Each cam In tau.Fields
    Cells(i, 3).Value = cam.Name
    i = i + 1
  Next
  bd.Close
End Sub

```

- Ejecuta el procedimiento anterior para probarlo, es decir:

* Escribe en la celda D4, una de las tablas que tienes en la columna B

* Clic en [Campos]

- Graba el libro de cálculo con el mismo nombre: **Excel47**.

- Sitúate en el **Módulo de la Hoja1** y escribe:

```

Private Sub CommandButton4_Click()
  Dim i As Byte, j As Byte, k As Byte
  Dim bd As Database
  Dim r As Recordset
  Dim tau As TableDef
  Dim cam As Field
  Set bd = OpenDatabase("c:\TuCarpeta\Neptuno.mdb")
  Set r = bd.OpenRecordset("Select * from " & Hoja1.Cells(4, 4).Value, dbOpenDynaset)
  Set tau = bd.TableDefs(Hoja1.Cells(4, 4).Value)
  Hoja2.Cells(1, 1).Value = "Listado de la tabla " & Hoja1.Cells(4, 4).Value
  i = 1
  For Each cam In tau.Fields
    Hoja2.Cells(2, i).Value = cam.Name
    i = i + 1
  Next
  k = 3
  r.MoveFirst
  Do While Not r.EOF

```

```
    For j = 1 To i - 1
        Hoja2.Cells(k, j).Value = r(Hoja2.Cells(2, j).Value)
    Next
    k = k + 1
    r.MoveNext
Loop
bd.Close
End Sub
```

- Ejecuta el procedimiento anterior, para probarlo, es decir:
 - * Escribe en D4, una de las tablas que tienes en la columna B
 - * Clic en [Listado]
 - * Investiga el contenido de la **Hoja2**

- Graba el libro de cálculo con el mismo nombre **Excel47**.