Visual Basic para Aplicaciones del Access 2000

(Manual FV)

Índice

Antes de empezar 3

1 Fundamentos de VBA 5

Autoevaluación 1 47

Soluciones 1 50

2 Introducción a la Programación Visual 58

Autoevaluación 2 110

Soluciones 2 117

3 Formularios y Controles 121

Autoevaluación 3 153

Soluciones 3 157

4 Objetos de Acceso a Datos (DAO) 163

Autoevaluación 4 184

Soluciones 4 188

A Programación en SQL 191

B Tratamiento de errores y Depuración 205

C Los “otros” VBA 215a224

Antes de empezar

- **Manual F.V.**
  Significa “manual **práctico** de informática”, pero realmente realmente PRÁCTICO.

Esto que tienes en las manos, no es un “libro”, esto que tienes en las manos es un **curso práctico de informática**. Dicho de otro modo:

- No has de “leer”
- Sino que has de “hacer”

Necesitas:

- Un ordenador **PC** con el **Microsoft Access 2000** instalado.
- Este “manual”
- Un bolígrafo
- Un poco de tiempo cada día.
- Paciencia y muchas ganas de “aprender”

Mi consejo:

- Siéntate cómodamente delante del ordenador.
- Haz paso a paso (sin prisa) **todo** lo que tienes en este manual, incluyendo los ejercicios de autoevaluación, que hay en cada capítulo.
- Si un ejercicio no te sale, vuelve a empezar.
- Toma notas, subraya, tacha, corrige y amplia todo lo que consideres importante, con el bolígrafo, en este manual.
- Experimenta contínuamente por tu cuenta.
- Acostumbrate a utilizar la estupenda **ayuda** del programa, aunque su lenguaje es más técnico que didáctico, poco a poco te habituarás.

Mis deseos:

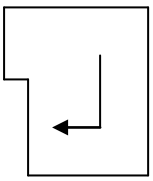
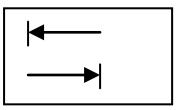
Espero que este manual (**tu** manual) dentro de un par de meses esté completamente destrozado. No porque lo hayas roto de rabia, sino porque

hayas pasado sus páginas miles de veces y hayas escrito es sus márgenes cientos de notas.

Conocimientos previos de Informática:

Es conveniente tener conocimientos, no necesariamente profundos, del entorno **Windows** y del **Microsoft Access 2000** (puedes bajarte el **Manual FV del Access 2000**)

Notación Utilizada:

clic	significa	pulsar una vez el botón izquierdo del ratón.
doble clic **o** **clic-clic**	significa	pulsar dos veces seguidas el botón izquierdo del ratón
[Aceptar]	significa	hacer clic en el botón “Aceptar”
Menú Formato **Ancho de columna...** **Ajuste perfecto**	significa	hacer clic en la opción “Formato” de la **barra de menús**. hacer clic en la opción “Ancho de columna...” de la nueva ventana que aparece y volver a hacer clic en la opción “Ajuste perfecto” de la nueva ventana.
[Return]	significa	pulsar la tecla: 
[Tab]	significa	pulsar la tecla: 
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 2000**:

CLIC en [Inicio]
Cursor en **Programas**
CLIC en **Microsoft Access**

- En la pantalla inicial, activa la opción: **Base de datos de Access en blanco**
  Y CLIC en [Aceptar]

- Sitúate en tu carpeta, es decir en el campo **"Guardar en:"**, debe aparecer "*TuCarpeta*"

- En el campo **"Nombre de archivo"**, escribe: **PROGRAMAS**
  y CLIC en [Crear]

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

- CLIC en el **Objeto: Módulos** 

- CLIC en el botón [Nuevo] 

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]

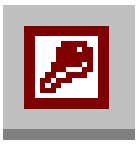
Si todo va bien, aparece una ventana de nombre **"Microsoft Access"** con el mensaje: **Hola Mundo** y el botón [Aceptar]

- 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" la base de datos **PROGRAMAS**. Es decir, CLIC en la **X** del extremo superior derecho de la ventana **"PROGRAMAS: Base de datos"**.
  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]

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

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**:

CLIC en [Inicio]
Cursor en **Programas**
CLIC en **Microsoft Access**

- "Abrir un archivo existente"
  Más archivos ...
  [Aceptar]

- Sitúate en **"tu carpeta"**. Es decir, en el campo **"Buscar en:"** debe aparecer *TuCarpeta*.

- CLIC en **PROGRAMAS.mdb**, para seleccionar el fichero.

- CLIC en [Abrir]

- Selecciona el "**Objeto: Módulos**", si no está ya seleccionado.

- Como sólo tenemos un módulo (Módulo1) ya está seleccionado, haz CLIC en el botón [Diseño]

- 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 llamado **"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-Tehn-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 "indice" 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:
Indice = 9
Aparece el mensaje: **"El valor del índice es = 9"**

Al volverse a iniciar el ciclo:
Indice = 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: $\mathbf{ax^2 + bx + c = 0}$, con **“a”** diferente de **0**

Definimos “discriminante” de la ecuación:

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

Si **Discriminante > 0**
Las soluciones de la ecuación son:

$$X1 = \frac{-b + \sqrt{Discriminante}}{2a}$$

$$X2 = \frac{-b - \sqrt{Discriminante}}{2a}$$

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
                A(2,1)=
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 farenheit As Integer
   Dim celsius As Integer
   Debug.Print "Temperaturas Farenheit y Celsius"
   Debug.Print "=============================="
   For contador = -2 To 13
      celsius = 10 * contador
      farenheit = 32 + (celsius * 9) / 5
```

```
        'La fórmula anterior transforma la temperatura de
        'grados centígrados a grados farenheit
        Debug.Print "    " & celsius & "    " & farenheit
        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 raiz 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 raiz 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).

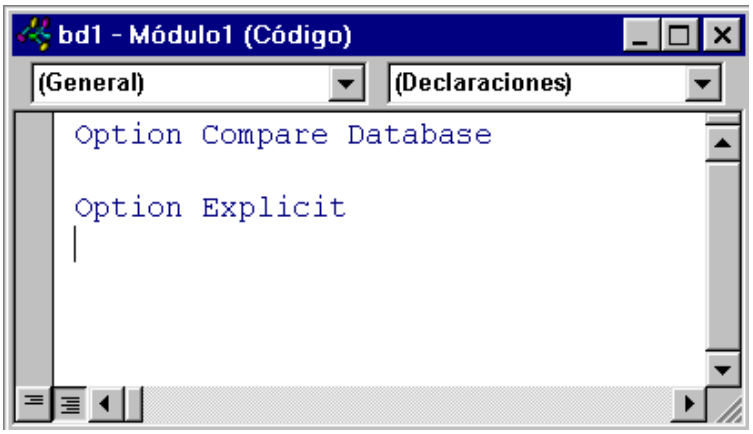
Para saber más

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++. Seria 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 explicitamente (Dim). No es mas 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")
   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].

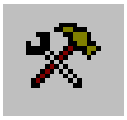
Si te molesta la “ayuda inteligente” basta que pulses la tecla [Esc] cuando aparece.

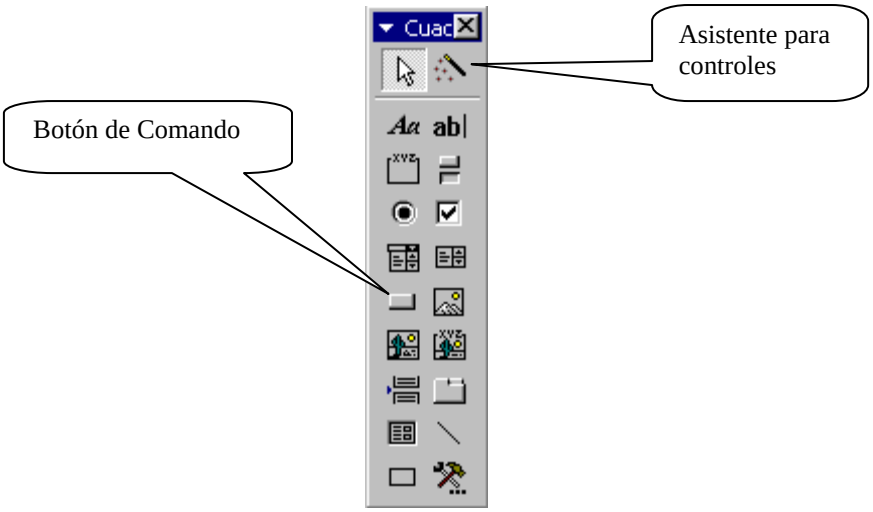
Comencemos a programar visualmente

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**
  - Clic en el **Objeto: Formularios**: 

  - Con la opción **Crear formulario en Vista Diseño** seleccionada, pulsa [Return]
  - Con el **Cuadro de herramientas** a la vista, si no lo está, haz clic en el icono **Cuadro de herramientas**: 
  - Asegúrate que el icono **Asistente para controles** del cuadro de herramientas, se encuentra desactivado:



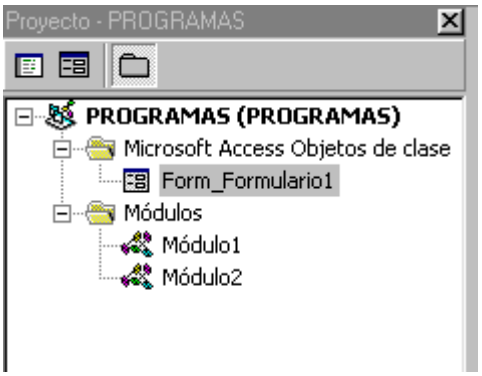
  - Haz clic en el icono **“Botón de Comando”** del cuadro de herramientas.
  - “Marca” un pequeño recuadro en el formulario
  - Con el **Comando0** seleccionado, haz clic en su interior.

- Borra el texto que aparece por defecto y escribe en su lugar **“Saludo”**

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

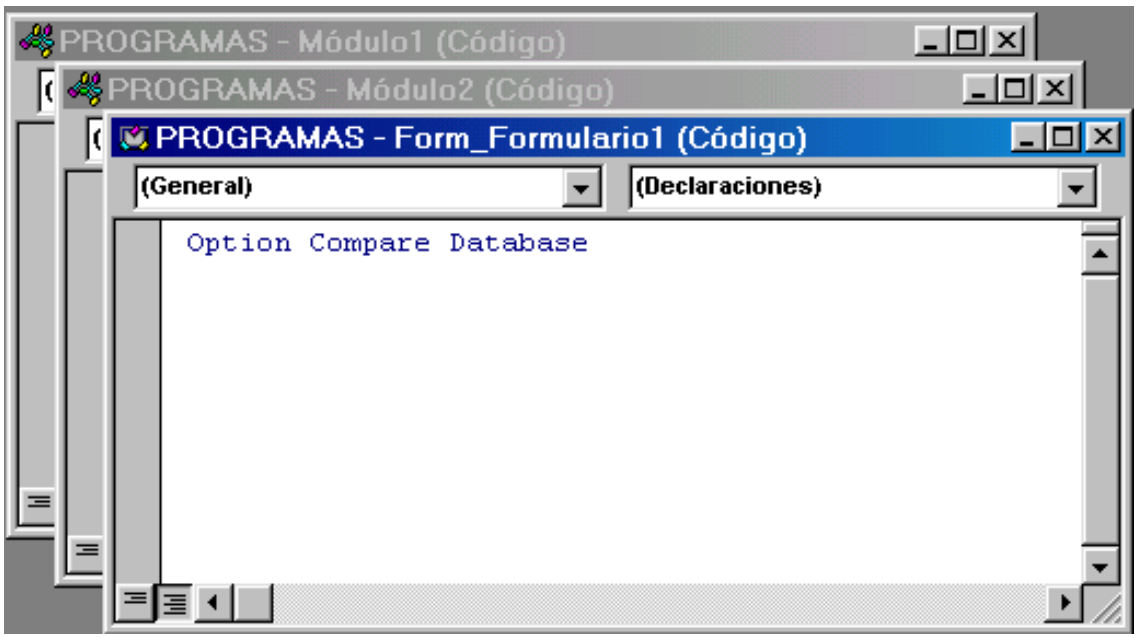
- Observa que hemos ido a parar al “Editor de Visual Basic” Concretamente en una ventana llamada **“PROGRAMAS - Form_Formulario1 (Código)”**, que denominaremos **módulo del formulario** (a diferencia de los otros módulos)

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

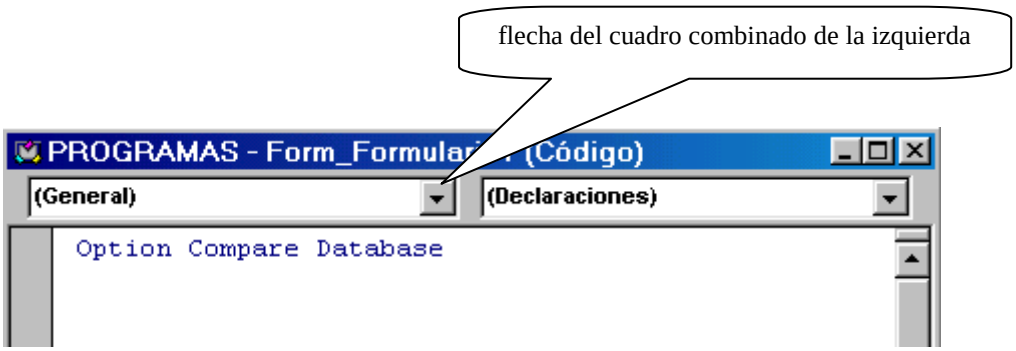


Si no la tienes a la vista, deberás hacer: **Menú Ver - Explorador de proyectos**. Donde podemos visualizar los diferentes objetos de nuestra base de datos, entre ellos los módulos 1 y 2.

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:



- 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**

- Ejecuta el **Formulario1**, es decir, desde la pantalla de diseño del formulario haz clic en el icono **Vista:**

- 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:

Clic en el icono **Vista**:

- 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)**
  - "Cierra" la ventana de **Propiedades** (clic en el icono **"X"** del extremo superior derecho de la ventana)

- 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 própia 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 formulariuo. 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 vacio (“”).
- **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

Es muy importante que antes de continuar intentes hacer los siguientes ejercicios. Si no te ves capaz de hacer alguno de ellos, consulta su solución que se encuentra más adelante. En algunos de ellos, la solución contiene alguna instrucción de VBA nueva, posiblemente encontrarás otra forma de rehacerlos, pero es interesante consultar la solución y el comentario o indicación que te doy.

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=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.

Soluciones de Autoevaluación 1

Prog1

```
Sub Prog1()
   Dim num1 As Double, num2 As Double
   Dim opc As String * 1
   opc = "S"
   Do While opc = "S" Or opc = "s"
      num1 = InputBox("Escribe un número positivo menor de 57")
      If (num1 < 0 Or num1 > 57) Then
         MsgBox ("El número es erróneo")
         Exit Sub
      End If
      num2 = InputBox("Escribe otro número positivo menor de 57")
      If (num2 < 0 Or num2 > 57) Then
         MsgBox ("El número es erróneo")
         Exit Sub
      End If
      MsgBox "El producto de los dos números es = " & (num1 * num2)
      opc = InputBox("Quieres volver a empezar (S/N)")
   Loop
End Sub
```

Podemos salir del programa, desde cualquier punto del mismo, sin más que utilizar la instrucción **Exit Sub**

Prog2

```
Sub Prog2()
   Dim num As Double, contador As Integer
   contador = 0
   num = InputBox("Escribe un número 'para acabar escribe 9999'")
   Do While num <> 9999
      contador = contador + 1
      num = InputBox("Escribe un número 'para acabar escribe 9999'")
   Loop
   MsgBox "Has escrito " & contador & " números distintos del 9999"
End Sub
```

Observa el uso de un “contador”, para **contar** las veces que se repite el bucle **Do While**

Prog3

```
Sub Prog3()
   Dim num As Double, contador As Integer
   Dim sum As Double
   contador = 0: sum = 0
   num = InputBox("Escribe un número 'para acabar escribe 9999'")
   Do While num <> 9999
      contador = contador + 1: sum = sum + num
      num = InputBox("Escribe un número 'para acabar escribe 9999'")
   Loop
   MsgBox "Has escrito " & contador & " números distintos del 9999" & vbCrLf _
      & "Y la suma de todos ellos es " & sum
End Sub
```

Prog4

```
Sub Prog4()
   Dim num As Double, contador As Integer
   Dim sum As Double, prod As Double
   contador = 0: sum = 0: prod = 1
   num = InputBox("Escribe un número 'para acabar escribe 9999'")
   Do While num <> 9999
      contador = contador + 1: sum = sum + num: prod = prod * num
      num = InputBox("Escribe un número 'para acabar escribe 9999'")
   Loop
   MsgBox "Has escrito " & contador & " números distintos del 9999" & vbCrLf _
      & "Y la suma de todos ellos es " & sum & vbCrLf _
      & "Y el producto es " & prod
End Sub
```

Prog5

```
Sub Prog5()
   Dim mul As Double, sum As Double
   Dim salida As String
   mul = 0: sum = 0: salida = ""
   Do While mul < 1000
      salida = salida & " - " & mul
      mul = mul + 23
      sum = sum + mul
   Loop
   MsgBox "Múltiplos de 23 inferiores a 1000: " & vbCrLf _
      & salida & vbCrLf _
      & "Su SUMA = " & (sum - mul)
End Sub
```

Observa la técnica de **“salida = salida & ...”**, para conseguir “acumular” todo lo que aparecerá en un cuadro **MsgBox**.

Prog6

```
Sub Prog6()
   Dim x1 As Double, x2 As Double
   Dim x As Double, y As Double
   Dim incr As Double, salida As String
   salida = "Tabla de y=sen(7x-5) " & vbCrLf
   x1 = InputBox("Escribe el valor minimo de 'x'")
   x2 = InputBox("Escribe el valor maximo de 'x'")
   incr = InputBox("Escribe el incremento de 'x'")
   For x = x1 To x2 Step incr
      y = Sin(7 * x - 5)
      salida = salida & x & " --- " & y & vbCrLf
   Next
   MsgBox salida
End Sub
```

Prueba lo que sucede para x1=1, x2=1000, incr=1. Deberás pulsar [Return] para cerrar el **MsgBox**.
Observa cómo se calcula un **seno** en Visual Basic.

Prog7

```
Sub Prog7()
   Dim hipot As Double, cat1 As Double, cat2 As Double
   Dim opc As String * 1, salida As String
   salida = "TEOREMA DE PITÁGORAS" & vbCrLf & _
        "====================" & vbCrLf & vbCrLf
   opc = "S"
   Do While opc = "S" Or opc = "s"
      hipot = InputBox("Escribe el valor de la hipotenusa")
      cat1 = InputBox("Escribe el valor del cateto")
      If cat1 > hipot Then
         MsgBox "No puede ser"
         Exit Sub
      End If
      cat2 = Sqr(hipot * hipot - cat1 * cat1)
      MsgBox salida & "Hipotenusa = " & hipot & " Cateto = " & cat1 & vbCrLf _
         & "El valor del otro CATETO es = " & cat2
      opc = InputBox("Quieres volver a empezar (S/N)")
   Loop
End Sub
```

Observa la “técnica” del “Volver a empezar”:

```
opc= “S”
Do While opc=”S” Or opc=”s”
        ...
        ...
        ...
        opc=InputBox(“Quieres volver a empezar (S/N)”)
Loop
```

Prog8

```
Sub Prog8()
   Dim mult As Double, sum As Double, prod As Double
   Dim opc As String * 1, salida As String
   opc = "S"
   Do While opc = "S" Or opc = "s"
      sum = 0: prod = 1: salida = ""
      For mult = 7 To 7 * 15 Step 7
         salida = salida & " - " & mult
         sum = sum + mult
         prod = prod * mult
      Next
      MsgBox salida & vbCrLf & "Suma = " & sum & _
         " Producto = " & prod
      opc = InputBox("Quieres Repetir (S/N)")
   Loop
End Sub
```

Compara este procedimiento con el **Prog5**

Prog9

```
Sub Prog9()
   Dim opc As String * 1, x As String * 1
   Dim bt As Double, at As Double, art As Double
   Dim br As Double, ar As Double, arr As Double
   Dim radi As Double, arcir As Double
   opc = "S"
   Do While opc = "S" Or opc = "s"
      x = InputBox("Triángulo (T), Rectángulo (R), Círculo (C)")
      Select Case x
         Case "T"
            bt = InputBox("base triángulo")
            at = InputBox("altura triángulo")
            art = bt * at / 2
            MsgBox "área triángulo = " & art
         Case "R"
            br = InputBox("base rectángulo")
            ar = InputBox("altura rectángulo")
            arr = br * ar
            MsgBox "área rectángulo = " & arr
         Case "C"
            radi = InputBox("radio círculo")
            arcir = 3.141592 * radi * radi
            MsgBox "área círculo = " & arcir
         Case Else
            MsgBox "Esta opción no existe"
      End Select
      opc = InputBox("Quieres volver a empezar(S/N)")
   Loop
End Sub
```

Prog10

```
Sub Prog10()
   Dim A As Integer, b As Integer
   Dim i As Integer, j As Integer
   Dim salida As String
   salida = ""
   A = InputBox("Altura")
   b = InputBox("Base")
   For i = 1 To A
      For j = 1 To b
         salida = salida & "*"
      Next
      salida = salida & vbCrLf
   Next
   MsgBox salida
End Sub
```

Prog11

```
Sub Prog11()
   Dim A As String * 1
   Dim i As Integer, j As Integer
   Dim salida As String
   salida = ""
   A = InputBox("Caracter")
   b = InputBox("Lado del cuadrado")
   For i = 1 To b
      For j = 1 To b
         salida = salida & A
      Next
      salida = salida & vbCrLf
   Next
   MsgBox salida
End Sub
```

Prog12

```
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
```

Prog13

```
Function sum(n1 As Double, n2 As Double) As Double
   sum = n1 + n2
End Function
Function neperiano(x As Double) As Double
   neperiano = Log(x)
End Function
Function raizcua(n As Double) As Double
   raizcua = Sqr(n)
End Function
Sub Prog13()
   Dim x As Double, y As Double, num As Double
   Dim z As Double, opc As String
   opc = InputBox("(S)uma, (R)aíz cuadrada, (L)ogaritmo neperiano")
   Select Case opc
      Case "S"
         x = InputBox("Primer sumando")
         y = InputBox("Segundo sumando")
         MsgBox "Suma = " & sum(x, y)
      Case "R"
         num = InputBox("valor a extraer la raíz cuadrada")
         MsgBox "Raíz cuadrada de " & num & " es " & raizcua(num)
      Case "L"
         z = InputBox("valor a calcular su logaritmo neperiano")
         MsgBox "El Logaritmo neperiano de " & z & " es " & neperiano(z)
      Case Else
         MsgBox "Opción no correcta"
   End Select
End Sub
```

Observa cómo se calcula en Visual Basic, un logaritmo neperiano.

Prog14

```
Function SumaCuadrado(x As Double, y As Double) As Double
   SumaCuadrado = x * x + y * y
End Function
Function RestaCuadrado(A As Double, b As Double) As Double
   RestaCuadrado = A * A - b * b
End Function
Sub Prog14()
   Dim n1 As Double, n2 As Double
   n1 = InputBox("Primer sumando")
   n2 = InputBox("Segundo sumando")
   MsgBox "La suma de los cuadrados es " & SumaCuadrado(n1, n2) & vbCrLf _
      & "La diferencia de los cuadrados es " & RestaCuadrado(n1, n2)
End Sub
```

Prog15

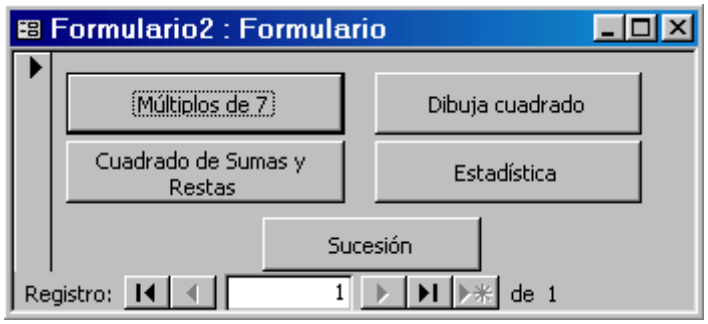
```
Function Media10(A() As Double) As Double
Dim sum As Double, i As Integer
sum = 0
For i = 1 To 10
   sum = sum + A(i)
Next
Media10 = sum / 10
End Function
Sub Prog15()
   Dim i As Integer, x(1 To 10) As Double
   Dim des(1 To 10) As Double
   Dim med As Double, d As Double
   Dim descua(1 To 10) As Double
   Dim var As Double, destip As Double
   Dim salida As String
   For i = 1 To 10
      x(i) = InputBox("Valor")
   Next
   med = Media10(x())
   For i = 1 To 10
      des(i) = Abs(x(i) - med)
   Next
   d = Media10(des())
   For i = 1 To 10
      descua(i) = des(i) * des(i)
   Next
   var = Media10(descua())
   destip = Sqr(var)
   salida = "Valores: "
   For i = 1 To 10
      salida = salida & x(i) & " -"
   Next
   salida = salida & vbCrLf
   salida = salida & "Media = " & med & vbCrLf
   salida = salida & "Desviaciones respecto a la Media = "
   For i = 1 To 10
      salida = salida & des(i) & " - "
   Next
   salida = salida & vbCrLf
   salida = salida & "Desviación Media = " & d & vbCrLf
   salida = salida & "Varianza = " & var & vbCrLf
   salida = salida & "Desviación Típica = " & destip
   MsgBox salida
End Sub
```

Observa:

- Si una función tiene por argumento una matriz, siempre hemos de declararla sin dimensión: **Function Media10(A() As Double) As Double**
- Las desviaciones respecto a la media son las diferencias de cada valor y la media, pero en valor absoluto. **Abs()** calcula el valor absoluto
- Observa la importancia que tiene la técnica **salida = salida & ...**, para conseguir mucha información en un único **MsgBox**.

Prog16

```
Sub Prog16()
   Dim num(1 To 25) As Double
   Dim den(1 To 25) As Double
   Dim i As Integer, salida As String
   For i = 1 To 25
      num(i) = 3 * i + 1
      den(i) = 2 * i - 1
   Next
   salida = ""
   For i = 1 To 25
      salida = salida & num(i) & "/" & den(i) & " , "
   Next
   salida = salida & vbCrLf & vbCrLf & vbCrLf & "Valores de la Sucesión = "
   For i = 1 To 25
      salida = salida & num(i) / den(i) & " , "
   Next
   MsgBox salida
End Sub
```




```
Private Sub Comando0_Click()
   Prog8
End Sub

Private Sub Comando1_Click()
   Prog11
End Sub

Private Sub Comando2_Click()
   Prog14
End Sub

Private Sub Comando3_Click()
   Prog15
End Sub

Private Sub Comando4_Click()
   Prog16
End Sub
```

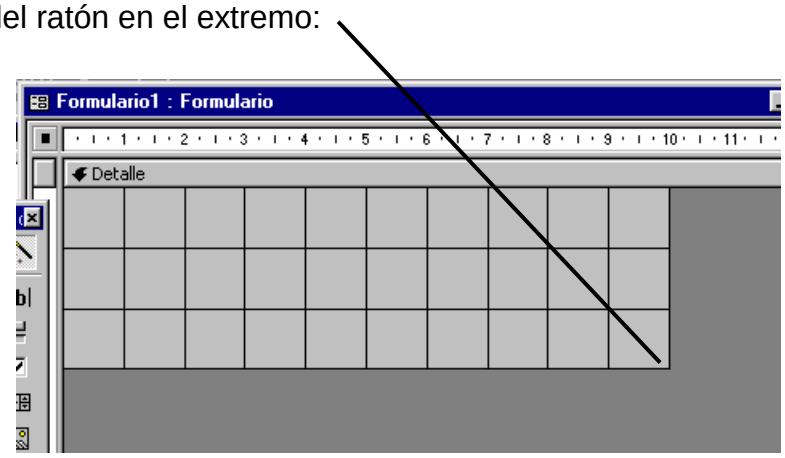
2

Introducción a la Programación Visual

a) Haz una **nueva** base de datos de nombre **VISUAL**, en tu carpeta

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

- Selecciona el **Objeto: Formularios**
- CLIC en el botón [Nuevo]
- Con la opción **Vista Diseño** seleccionada y sin "elegir ninguna tabla ni consulta", haz CLIC en [Aceptar]
- Haz el formulario un poco mas grande. Es decir:
  - Sitúa el cursor del ratón en el extremo:



  - Y "arrastra" un poco el cursor hacia la derecha y abajo

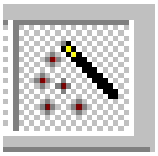
- Asegúrate que el “Cuadro de Herramientas” está presente:

Si esta barra de iconos no la tienes “en pantalla”, haz:

Menú Ver

Cuadro de Herramientas

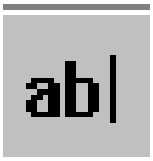
- Si el icono **“Asistente para controles”** está activado:



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”** de la barra de iconos: “Cuadro de Herramientas”:



- 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.

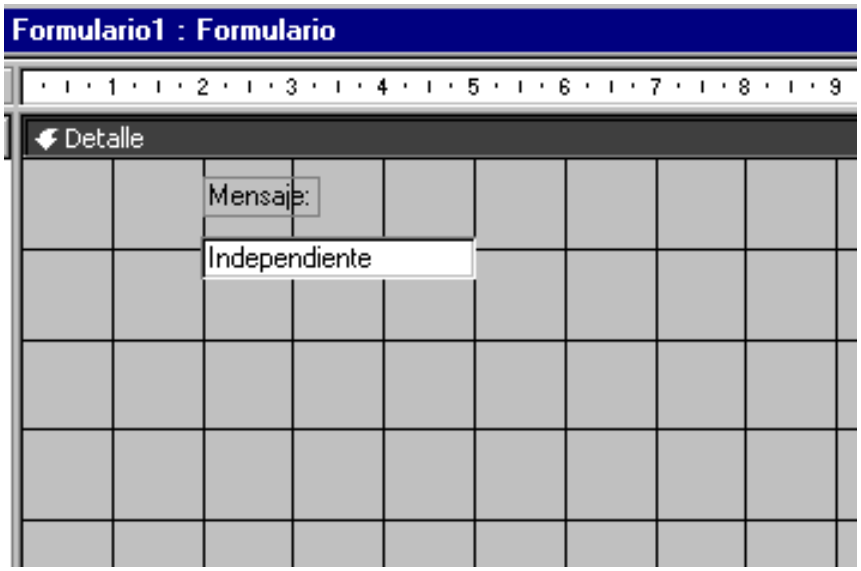
- Vuelve a seleccionar el “cuadro de texto”, es decir: CLIC en el interior del “cuadro”, donde aparece el texto “Independiente”

- Puedes mover **la etiqueta** del “cuadro de texto”, sin más que situar el cursor del ratón en el extremo superior - izquierdo, y en el momento que aparece un **dedo**, “arrastra” 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
  - En el momento en que aparece “el dedo”
  - “Arrastra” el cursor del ratón.

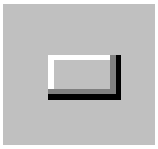
- Si quieres mover la etiqueta junto con el cuadro de texto, debes situar el cursor del ratón en cualquier **borde** y cuando el cursor del ratón cambia a una **mano extendida**, “arrastra” el cursor adonde quieras.

- 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 de comando"** de la barra de iconos: Cuadro de Herramientas:



- 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**

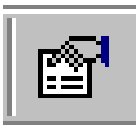
- CLIC en cualquier punto fuera del botón.

- Graba el formulario con el nombre **PROG1**, sin salir de la pantalla de diseño. Es decir:
  - Menú Archivo
    Guardar como …
  - En el campo **"Guardar Formulario 'Formulario1' en:"**
  - 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 **"Propiedades"**, de la barra de iconos superior:

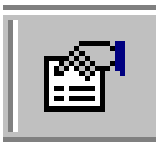


- Aparece la ventana de propiedades del control seleccionado, en nuestro caso del "cuadro de texto".
  Asegúrate que la Solapa:Todas está activada.

- 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**

- "Cierra" la ventana de propiedades. Es decir, CLIC en la **X** que aparece en el extremo superior derecho de la ventana correspondiente

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.
  - CLIC en el icono **“Propiedades”**: 
  - Cambia el contenido de la **Propiedad: Nombre**, por **BOTÓN1** (no te dejes el acento).
  - “Cierra” la ventana de propiedades.
  - “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:

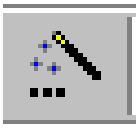
- Un “cuadro de texto” de nombre **CUADRO** (aunque no se vea este nombre)
- Un “botón” de nombre **BOTÓN1** (aunque no se vea este nombre).

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]

- CLIC en el icono **“Generar”** de la barra de iconos superior (no confundir con el icono “Asistentes para controles” del “Cuadro de Herramientas”):



- 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!”**

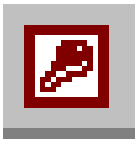
- Observa nuestro **procedimiento**:

```
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!”**

- “Cierra” la ventana **“VISUAL-Form_PROG1/Código)”**, es decir: CLIC en la **X** del extremo superior derecho de la ventana correspondiente.

- 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 el icono **Vista**:

del extremo superior izquierdo de la pantalla, para pasar a **“Modo Ejecución”**.

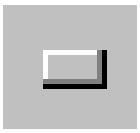
- Haz CLIC en el botón [VALE]

- Si todo funciona correctamente, en el “cuadro de texto” acaba de aparecer la frase **¡Hola, qué tal!**

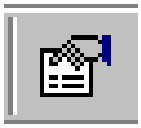
- Vuelve a hacer CLIC en el icono **Vista**, para volver a **"Modo Diseño"**:

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]=""**)…

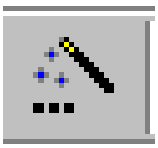
- CLIC en el icono **"Botón de Comando"**:



- CLIC en el lugar del formulario donde quieras colocar el nuevo botón.
- CLIC en el icono **"Propiedades"**:

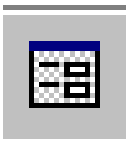


- En la propiedad **"Nombre"** escribe: **FUERA**
- "Cierra" la ventana de propiedades.
- Con el nuevo botón seleccionado. Haz CLIC en su interior para acceder a su "nombre externo".
- Borra el texto que aparece por defecto y escribe en su lugar **BORRAR**
- CLIC en cualquier punto fuera del botón.
- Selecciona el botón [BORRAR]
- CLIC en el icono **"Generar"**:



- CLIC en **"Generador de código"** y [Aceptar]
- Escribe entre las líneas:
  Private Sub FUERA_Click()
  y
  End Sub
  La línea:
  [CUADRO]=""
- "Cierra" la ventana **VISUAL-Form_PROG1 (Código)**
- Vuelve al **Microsoft Access**

- Graba de nuevo el formulario con el mismo nombre **PROG1**
- CLIC en el icono **"Vista":**

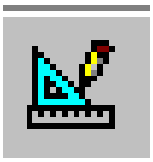
  

  para volver a "Modo Ejecución"
- CLIC en [BORRAR]
- CLIC en [VALE]
- CLIC en [BORRAR]

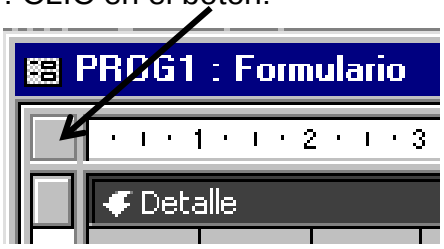
  Está claro lo que hemos conseguido, ¿verdad?

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**. Veamos:
  - CLIC en el icono **"Vista"**:

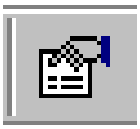
    

    para pasar a "Modo Diseño".
  - "Selecciona el formulario", es decir: CLIC en el botón:

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

- CLIC en el icono **"Propiedades"**:

  
- Modifica las siguientes propiedades:
  Selectores de registro: No

Botones de desplazamiento: No
Separadores de registros: No

- "Cierra" la ventana de propiedades.
- CLIC en el icono **"Vista"** para pasar a "Modo Ejecución".

Supongo que estarás de acuerdo conmigo en que el "formulario" tiene una presencia más elegante.

Vamos a mejorar aún más, el aspecto del formulario:

- CLIC en el icono **"Vista"** para pasar a "Modo Diseño"

- Selecciona el "cuadro de texto".

- Observa la 2ª línea de iconos, el icono de la izquierda nos indica el "nombre interno" del control seleccionado, en nuestro caso: **CUADRO**

- Selecciona el botón [VALE] y observa el contenido del primer icono de la segunda línea de iconos. Si todo va bien, debe aparecer el nombre: **BOTÓN1**.

- Vuelve a seleccionar el "cuadro de texto" y observa el resto de iconos de la **Segunda Barra de Iconos**:

Estos iconos nos permiten cambiar el **aspecto** del control seleccionado:

Podemos seleccionar:
- la **etiqueta** del cuadro de texto
- el **cuadro de texto**, propiamente dicho.
- Un **botón** u otro
- La sección **"Detalle" del formulario**.

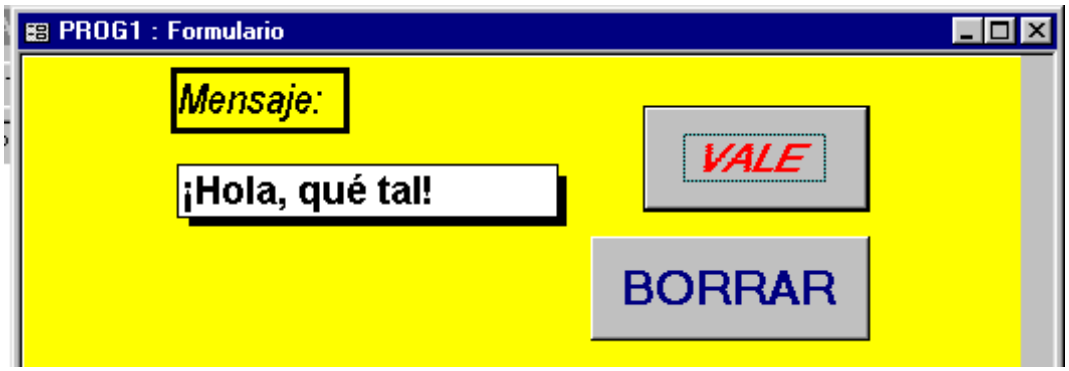
Según lo que tengamos seleccionado, podremos utilizar más o menos opciones de la "Segunda Barra de Iconos", como puedes ir observando si seleccionas cada uno de los objetos de nuestro formulario.

También puedes seleccionar varios elementos a la vez, sin más que **mantener pulsada una de las dos teclas de mayúsculas**, mientras vas haciendo CLIC en los diversos elementos que quieras seleccionar.

Teniendo en cuenta la utilidad de cada icono:
- Tipo de letra (Fuente)
- Tamaño de fuente
- Negrita

- Cursiva
- Subrayado
- Alinear a la izquierda
- Centrar
- Color de fondo o de relleno
- Color de fuente o de primer plano
- Color de borde o de línea
- Ancho de borde o de línea
- Efecto especial.

Se trata de que consigas el siguiente aspecto (aproximado) para el formulario **PROG1**:



- Graba el formulario **PROG1** con todos sus cambios con el mismo nombre.

- Ejecuta el formulario **PROG1**, y juega con "sus botones".

- "Cierra" el formulario **PROG1**.

h) Vamos a hacer otro programa…

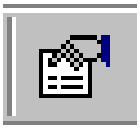
- Crea un nuevo formulario de nombre **PROG2**, en la base de datos **VISUAL**, es decir:

  - Con la "base de datos: VISUAL" a la vista y el **Objeto: Formularios** activado.
  - CLIC en el botón **[Nuevo]**
  - Con la opción "Vista Diseño" seleccionada
  - CLIC en [Aceptar]
  - Haz el formulario un poco más grande
  - Asegúrate que la barra de iconos "Cuadro de Herramientas" está a la vista.
  - Desactiva (si está activado) el icono "Asistente para controles" de la barra "Cuadro de Herramientas".

- Graba el formulario con el nombre **PROG2** (Menú Archivo – Guardar como…)

- Inserta en el formulario **PROG2** un "cuadro de texto" con las siguientes características:
  - Etiqueta: **Pedido de fecha**
  - Propiedades:
    Nombre: **PEDIDO**
    Origen del Control: **=Fecha()**

  Es decir:

  - CLIC en el icono "Cuadro de texto": 
  - CLIC en cualquier punto del formulario
  - CLIC en la "etiqueta" del cuadro de texto, para seleccionarla.
  - CLIC otra vez en la "etiqueta" del cuadro de texto, para situarse en su interior.
  - Borra el texto que aparece por defecto y escribe en su lugar **Pedido de fecha**.
  - Selecciona el "cuadro de texto", es decir: CLIC en el cuadro donde aparece el texto **"Independiente"**.
  - CLIC en el icono **"Propiedades"**: 
  - En la propiedad **"Nombre"**, escribe: **PEDIDO**
  - En la propiedad **"Origen del Control"**, escribe: **=Fecha()**
  - "Cierra" la ventana de propiedades

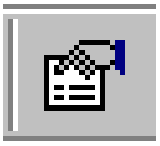
- Graba el formulario con el mismo nombre **PROG2** (CLIC en el icono del disquete).

- Vamos a ver lo que hemos conseguido:
  - CLIC en el icono **"Vista"**: 

  para pasar a "Modo Ejecución".
  - Si todo va bien, aparece en el campo "Pedido de fecha", la fecha actual (del reloj del ordenador).

- Vamos a “eliminar” los elementos del formulario que no tienen sentido en un formulario que no está asociado a ninguna “Tabla” de la base de datos…
  - CLIC en el icono **“Vista”:** 

  para pasar a “Modo Diseño”.
  - 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 registros: **No**
  - “Cierra” la ventana de propiedades del formulario.
  - Graba de nuevo el formulario con el mismo nombre **PROG2**
  - Pasa a “Modo Ejecución” (CLIC en el icono **“Vista”**)
  - Vuelve a “Modo Diseño” (CLIC en el icono **“Vista”**).

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: **=PrimeroMes()**
  - Tendremos de "programar" una función de nombre **"PrimeroMes()"**, que nos dé la fecha del primero de mes siguiente a la fecha actual.

  Veamos:

- Inserta en el formulario **PROG2** un nuevo cuadro de texto (CLIC en el icono correspondiente y CLIC en el lugar del formulario, donde quieres que aparezca).

- Cambia el texto que aparece por defecto en su "etiqueta" por: **Fecha de Facturación**

- Accede a las "propiedades" del cuadro de texto y en la **"Propiedad: Nombre"**, escribe: **FACTURA**

- Vamos a crear la función **PrimeroMes()**…
  - "Cierra" el formulario **PROG2** (graba los cambios).
  - Activa el **Objeto: Módulos** y CLIC en [Nuevo]
  - 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: **PrimeroMes** y CLIC en [Aceptar]
  - Escribe entre las líneas:
    Public Function PrimeroMes()
    y
    End Function

    La siguiente línea:
    PrimeroMes= DateSerial(Year(Now), Month(Now) +1, 1)
    (no te preocupes de momento en "entender" la línea de programa anterior)
  - CLIC en el icono **"Guardar"**. Contesta a la pregunta que aparece, haciendo CLIC en [Sí]
  - Nombre del módulo: **Módulo1**
  - Vuelve al **"Microsoft Access"**

- Vamos a asociar al cuadro de texto **"Fecha de Facturación"**, la función **PrimeroMes()…**

  - "Abre" el formulario **PROG2** en "Modo Diseño"
  - Accede a las propiedades del cuadro de texto correspondiente a **"Fecha de Facturación"**
  - En la propiedad **"Origen del Control"** escribe: **=PrimeroMes()**
  - "Cierra" la ventana de propiedades
  - Graba el formulario con el mismo nombre **PROG2**

- "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:

- Menú ?
  Ayuda de Microsoft Access

  - Asegúrate que la **Solapa: Contenido** está activada
  - CLIC en el "+" de la opción "**Referencia del lenguaje Visual Basic**
  - CLIC en el "+" de **Funciones**
  - CLIC en el "+" de **M-P** y CLIC en [Abrir]
  - CLIC en **Now (Función)**
  - Lee la explicación que aparece a la derecha sobre la función **Now** y "juega" con las opciones **"Vea también"** y **"Ejemplo"**.
  - Cuando esté claro para qué sirve la función incorporada **Now**, "cierra" la ventana de ayuda.

- Podemos localizar en la ayuda una "explicación concreta" de una forma más rápida. Haz lo siguiente:

  - Menú ?
    Ayuda de Microsoft Access

- Es conveniente “maximizar” la ventana de ayuda
- Activa la **Solapa: índice**
- Escribe en el campo “Escriba palabras clave”: **Year** y CLIC en [Buscar]
- Observa que la línea: **Ejemplo de las funciones DateSerial, Day Month y Year** ha quedado seleccionada, y a la derecha de la pantalla aparece la explicación correspondiente.
- Cuando esté claro para qué sirve **Year**, “cierra” la ventana de la ayuda.

- 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 a la “Pantalla de Diseño” del **Módulo1**

  - 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...

  - Con la "base de datos: **VISUAL1**" a la vista y el **"Objeto: Formularios"** activado.
  - CLIC en [Nuevo]
  - Con la opción **"Vista Diseño"** seleccionada
  - CLIC en [Aceptar]
  - Haz el formulario un poco más grande.
  - Asegúrate que la barra de iconos **"Cuadro de Herramientas"** está a la vista.
  - Desactiva (si está activado) el icono **"Asistentes para controles"** de la barra "Cuadro de Herramientas".
  - Graba el formulario con el nombre **TRIÁNGULO** (Menú Archivo – Guardar como...)
  - Vamos a colocar en primer lugar un "título guapo" a nuestro formulario...

  - CLIC en el icono **"Etiqueta"** del "Cuadro de Herramientas": 

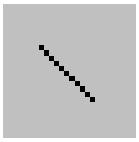
  - Traza un rectángulo en la parte superior del formulario (contendrá la frase: ÁREA DE UN TRIÁNGULO)

  - Escribe: **ÁREA DE UN TRIÁNGULO** y CLIC en cualquier punto fuera de la "etiqueta".

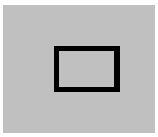
- Haz CLIC en la “etiqueta” anterior para seleccionarla.
- Utilizando las opciones de:

Consigue aproximadamente lo siguiente:

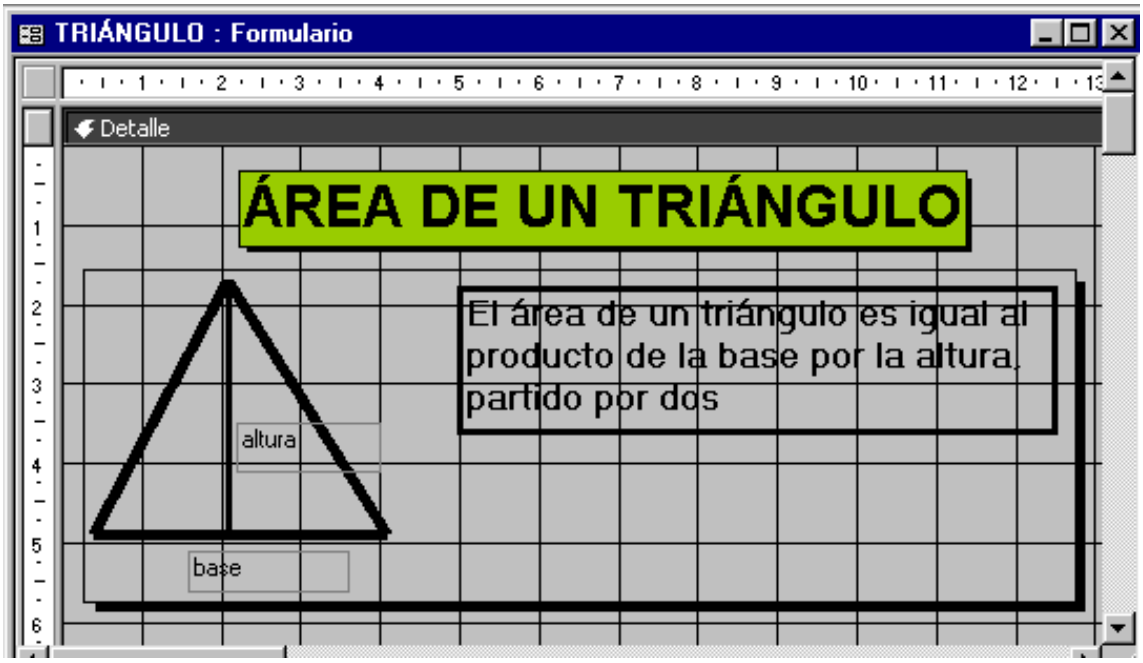
- Utilizando el icono **“Línea”**:



I el icono **“Rectángulo”**:



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



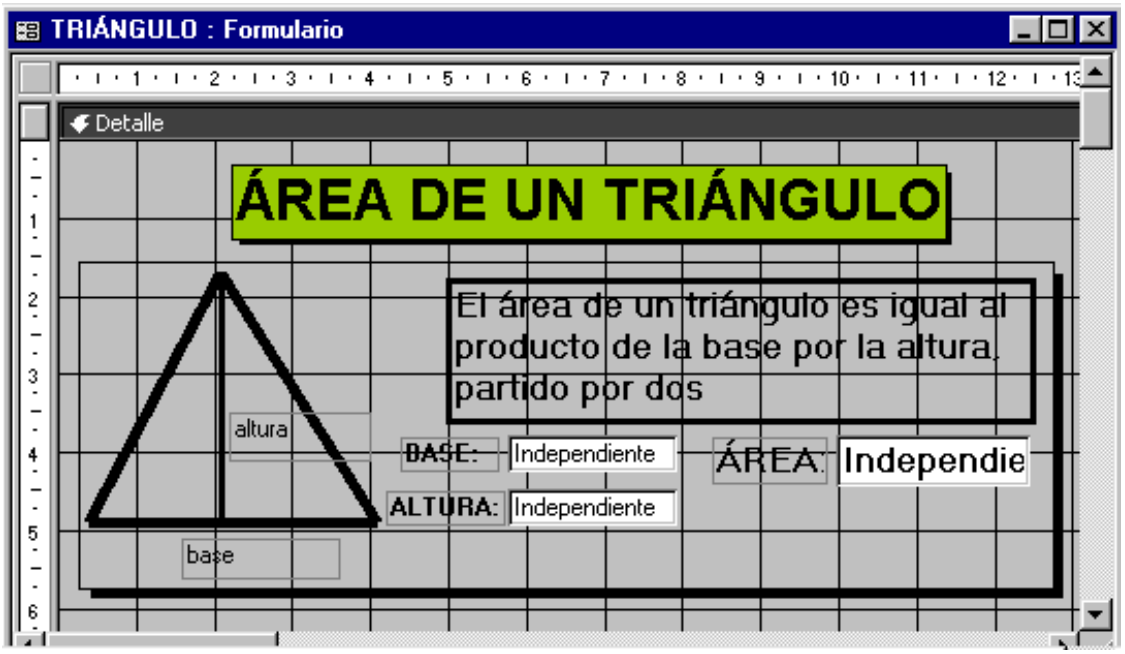
- Inserta en el formulario, 3 cuadros de texto, con las siguientes características:

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

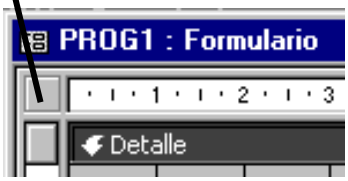
  Cuadro de texto 2: Etiqueta = **ALTURA** (en **Negrita**)
  Propiedad **"Nombre"** = **altura**

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

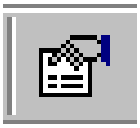
- Coloca los 3 cuadros de texto aproximadamente con la siguiente distribución:



- 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**

- "Cierra" la ventana de propiedades del formulario

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

- Pasa a "Modo Ejecución" (CLIC en el icono **"Vista"**)

  Deberás tener aproximadamente lo siguiente:

- Vuelve a "Modo Diseño" (CLIC en el icono **"Vista"**)

- 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**

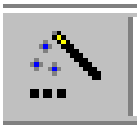
- Graba de nuevo el formulario con el mismo nombre **TRIÁNGULO**

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

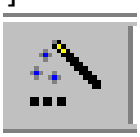
Se trata de hacer el siguiente programa:

"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]
- CLIC en el icono **"Generar"**: 
- 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]
  . CLIC en el icono **"Generar"**: 
  . Selecciona la opción **"Generador de código"** y [Aceptar]
  . 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]

Cuando estés cansado, “cierra” el formulario **TRIÁNGULO**.

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** y la Solapa: Formularios activada.

- CLIC en [Nuevo]

- “Vista Diseño” y CLIC en [Aceptar]

- Haz el formulario un poco más grande.

- Asegúrate que la barra de iconos “Cuadro de Herramientas” está a la vista.

- Desactiva, si está activado, el icono “Asistentes para controles” de la barra “Cuadro de Herramientas”.

- 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 trapecio, 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 trapecio.

- 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"** = **trapecio**

- 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 **"Cuadro de Herramientas"**.

- 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]**:

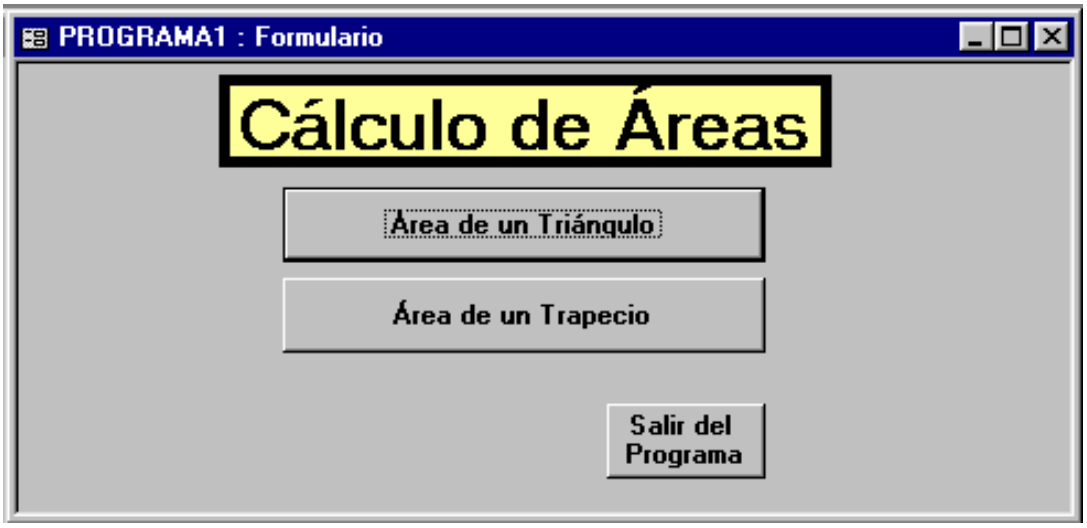
```
Private Sub borrar_Click()
   [base1] = ""
   [base2] = ""
   [altura] = ""
   [trapecio] = ""
End Sub
```

- 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
```

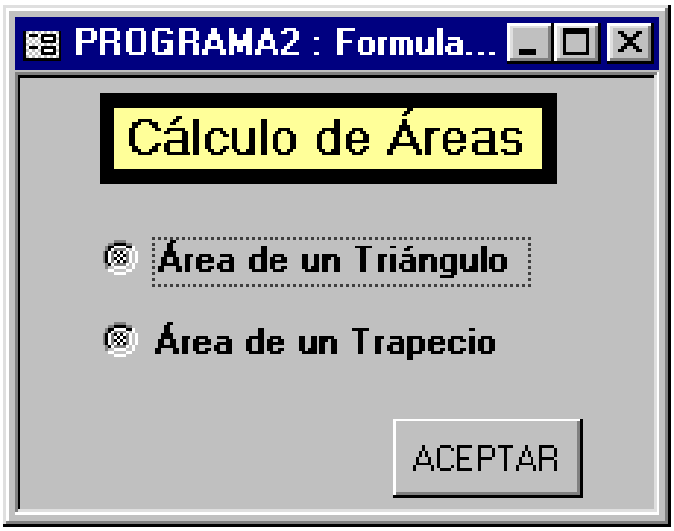
Observa la instrucción que sirve para "abrir un formulario" y la instrucción que sirve para salir del Access.

- 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:

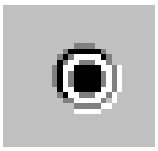
Aspecto:



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 "Cuadro de Herramientas":



  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].

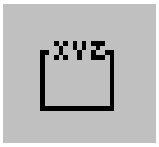
o) Se trata de hacer otro formulario para la B. de D. **VISUAL1**, de nombre **PROGRAMA3** que haga lo mismo que el **PROGRAMA2**, pero solucionando el problema que hemos observado en el anterior formulario y además trabajaremos con un nuevo "control": **"Casilla de verificación"** que es muy parecido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



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 "Cuadro de Herramientas"), 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:

  Menú Herramientas
  Inicio…

- En el campo **“Título de la aplicación”**, escribe: **ÁREAS**

- En el campo **“Mostrar”** selecciona **PROGRAMA3** y CLIC en [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%**

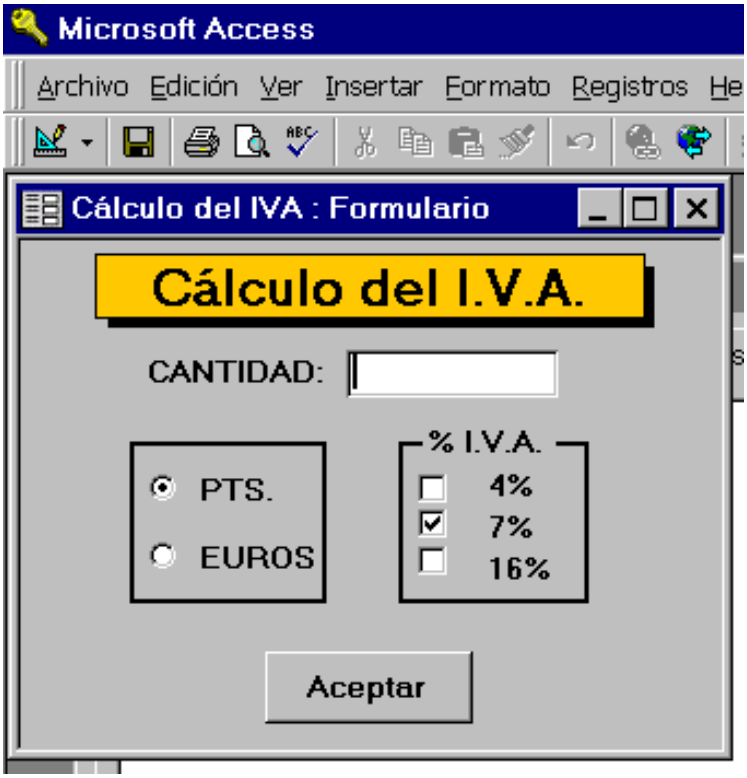
- Control **"botón de comando"**:
  Propiedad **"Nombre"**: **OK**
  Propiedad **"Título"**: **Aceptar**

- Propiedades del formulario:
  Barras de desplazamiento = **Ninguna**
  Selectores de registro = **No**
  Botones de desplazamiento = **No**
  Separadores de registro = **No**

Petendemos que el formulario aparezca siempre en un lugar determinado de la pantalla y con un tamaño también predefinido. Para conseguirlo debes cambiar la propiedad:

Ajuste de tamaño automático = **No**

Una vez hecho el formulario anterior, ejecútalo y colócalo en el ángulo superior izquierdo de la pantalla, tal como aparece en la siguiente ilustración:



Una vez conseguido el tamaño y situación del formulario, tal y cómo aparece en la ilustración anterior, **grábalo**.

Si ahora cierras y abres el formulario **"Cálculo del I.V.A."** observarás que se coloca en la posición y con el tamaño que tenía en el momento de grabarlo.

- "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**

Una vez hecho el formulario anterior, ejecútalo y colócalo en el ángulo inferior derecho de la pantalla, cuando esté situado en el lugar deseado, **grábalo** de nuevo.

- 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 asóciale 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].[pvppts] = CDbl(Forms![RESULTADO].[canpts]) _
      + CDbl(Forms![RESULTADO].[ivapts])
  Forms![RESULTADO].[pvpeur] = CDbl(Forms![RESULTADO].[caneur]) _
      + CDbl(Forms![RESULTADO].[ivaeur])
End Sub
```

Bien, antes de continuar observemos:

- Debido a que el procedimiento lo llamaremos desde el formulari "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:

- Menú Herramientas
  Inicio...

Título de la aplicación: **(escribe tu nombre y apellidos)**
Mostrar el formulario: **Cálculo del IVA**

Desactiva todas las opciones que aparecen por defecto, es decir CLIC en:
"Permitir el uso de menús no restringidos"
"Permitir el uso de menús contextuales predeterminados"
"Presentar la ventana Base de datos"
"Presentar la barra de estado"
"Permitir el uso de las barras de herramientas incorporadas"
"Permitir cambios en barras de herramientas y menús"

CLIC en **[Avanzadas>>]**
Desactiva la opcion que aparece y CLIC en [Aceptar]

- "Cierra" la B. de D. **"IVA"** y vuelvela a abrir
- Prueba "nuestro programa"
- Intenta cambiar alguna cosa del programa.
- La única cosa que nos gustaría eliminar o cambiar es la **"Barra de Menús"**. No te preocupes ya lo veremos en otro ejercicio

- 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:

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

- Define el campo **NumSocio** como **Clave Principal**. Es decir, sitúa el cursor en **NumSocio** y clic en el icono **"Clave Principal"**
- Graba la tabla con el nombre **Socios**

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

- "Abre" la tabla **Socios** e introduce los siguientes registros:

NumSocio: **001A**	NumSocio: **002A**
NombreApell: **Pepito Grillo**	NombreApell: **Francisca Sánchez**
NIF: **4.352.793V**	NIF: **7.932.001H**
Dirección:	Dirección: **Las Palmeras 51, 1º, 3ª**
Teléfono: **527 31 42**	Teléfono: **397 52 71**

- Crea un **autoformulario: en columnas** para la tabla **Socios**, y grábalo con el nombre **Socios**.

- Utilizando el formulario anterior introduce el siguiente registro:

  NumSocio: **003A**
  NombreApell: **Paquito Pérez**
  NIF:
  Dirección:
  Teléfono: **493 33 55**

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** seria 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** seria un ejemplo de nivel “normal”.

Vamos a ver si lo conseguimos:

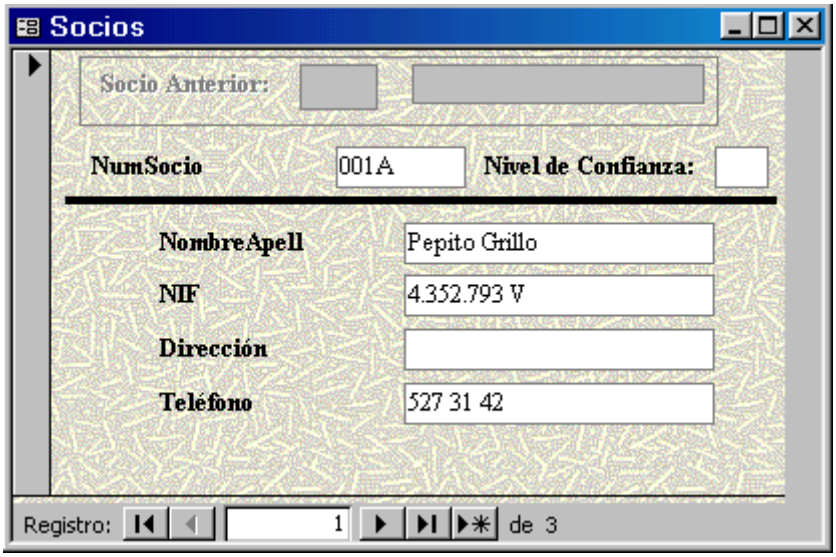
- Sitúate en la pantalla de **diseño** del formulario **Socios** e inserta tres cuadros de texto con las siguientes propiedades:

  - 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:



- 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.

- Como no tenemos ni idea de cuál deben ser estas órdenes, utilizaremos un truco muy útil:
    - Crearemos una "macro" que contenga las órdenes que nos interesan.
    - Convertiremos la macro en un programa VB, de esta forma podremos "localizar" lo que nos interesa.

Vamos a ver si funciona ...

- Crea una nueva **macro** con el siguiente contenido:

Acción	**Argumentos**
AbrirFormulario	Nombre del formulario: Socios
IrARegistro	Registro: Último
IrARegistro	Registro: Nuevo

Para grabar la macro, sigue el proceso siguiente:

Menú Archivo
Guardar como ...
Guardar Macro 'Macro1' en:
Macro1
Como: **Módulo**

Graba la macro, tal como nos advierte un mensaje de advertencia y haz clic en [Convertir] en el segundo mensaje que aparece.

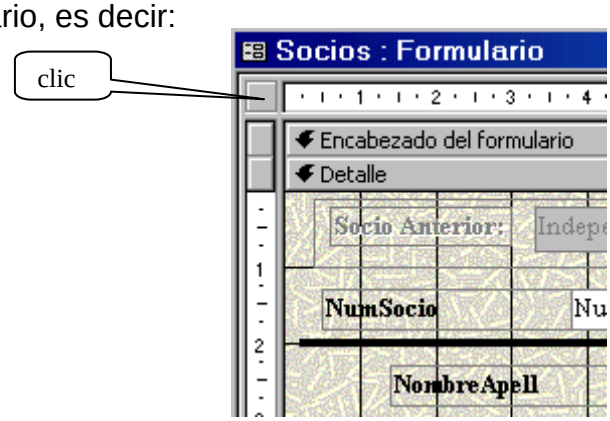
- Sitúate en el **Objeto: Módulos** y abre en modo **diseño** la **Macro convertida - Macro1**

Si todo funciona correctamente, localizarás las dos órdenes que necesitamos:

DoCmd.GotoRecord,"",acLast	último registro
DoCmd.GotoRecord,"",acNewRec	nuevo registro

Con las órdenes anteriores y lo que sabemos de Visual Basic, ya podemos solucionar el primer problema que nos habíamos propuesto. En efecto:

- 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:

- Abre 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 datol 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:

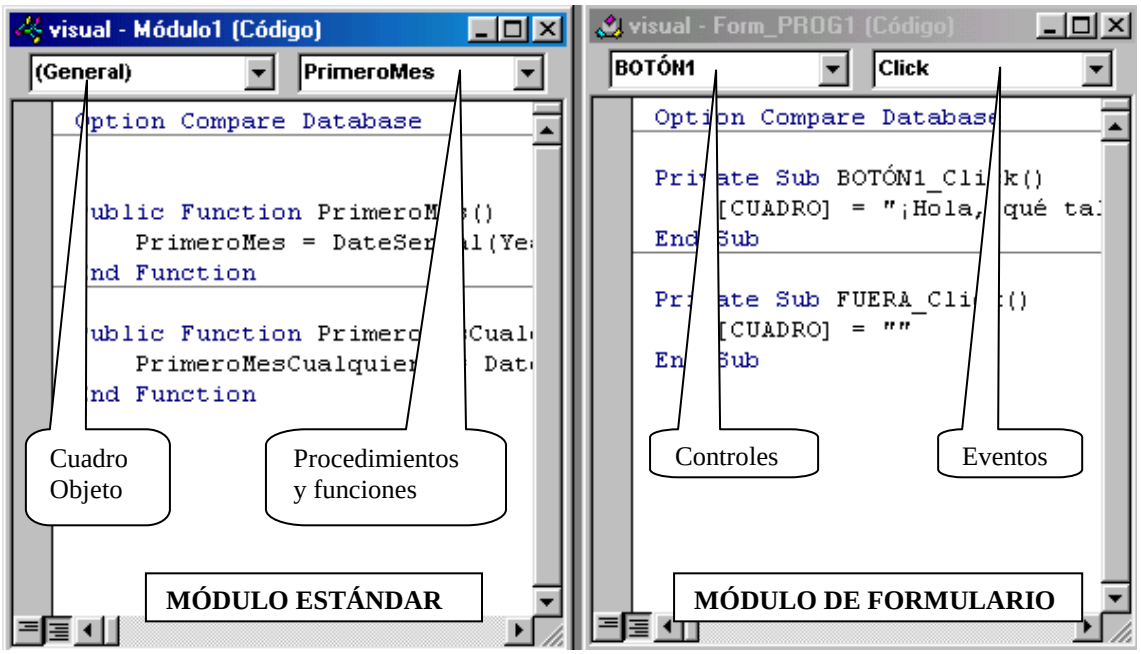
- **Módulos estándar:**
  - Accedemos a ellos a través de la pantalla inicial de la BD, **objeto: Módulos** y click en [Nuevo] o [Diseño]
  - En dichos módulos escribimos los procedimientos "Sub" y las funciones.

- **Módulos de formulario y informe**:
  - Accedemos a ellos desde la pantalla de diseño del formulario o informe, haciendo click en el icono **Código**:

    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.

Evitar la pantalla inicial del Access 2000

Un truco:
Si ejecutas el fichero **IVA.mdb** desde el explorador o de MiPC o desde un acceso directo, no evitamos que aparezca por unos instantes la pantalla inicial del Access 2000. Para solucionar el problema basta que grabes en la carpeta donde tienes el **IVA.mdb** un dibujo, imagen o logotipo en formato bmp (el del **Paint**) con el mismo nombre, es decir **IVA.bmp**. A partir de este momento lo que aparecerá por unos instantes no será la pantalla inicial del Access 2000, sino nuestro dibujo o imagen.

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.

Protección real de una base de datos Access

Habíamos visto, concretamente en la base de datos **IVA**, una forma de protegerla utilizando las opciones de **inicio**. Dicha protección no es real, ya que si al abrir la base de datos mantenemos pulsada una de las dos teclas de mayúsculas, las opciones de inicio se anulan.

Si queremos proteger realmente nuestra base de datos, hemos de grabarla en formato **MDE**: de esta forma impediremos la modificación de los formularios e informes y por otro lado el código Visual Basic se codifica.

Vamos a probarlo:

Desde la pantalla inicial de la B.D. Access1, haz lo siguiente:

```
Menú Herramientas
        Utilidades de la base de datos
                Crear archivo MDE...
```

En **Nombre de archivo**, escribe: **Access2.mde**

"Cierra" la B.D. Access1 y recupera la **Access2.mde**

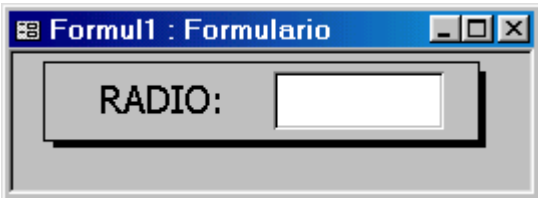
Intenta cambiar alguna cosa del formulario **Socios** o acceder al código del formulario.

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** la opción: **Presentar la ventana Base de Datos**

- Graba la base de datos **Eval2A** en formato **MDE** y con el nombre **Eval2Abis**.

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:

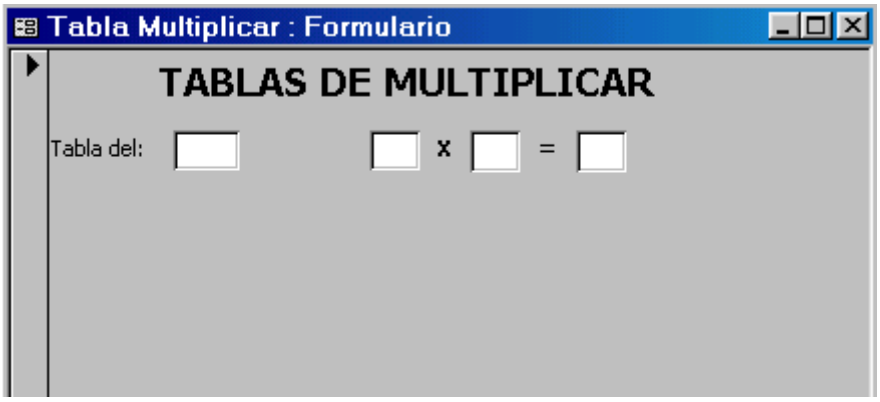
```
Sub Prog12()
   Dim num As Integer, i As Integer
   Dim salida As String
   salida = ""
   num = InputBox("Tabla del número")
   For i = 1 To 9
      salida = salida & num & " * " & i & " = " & num * i & vbCrLf
   Next
   MsgBox salida
End Sub
```

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)

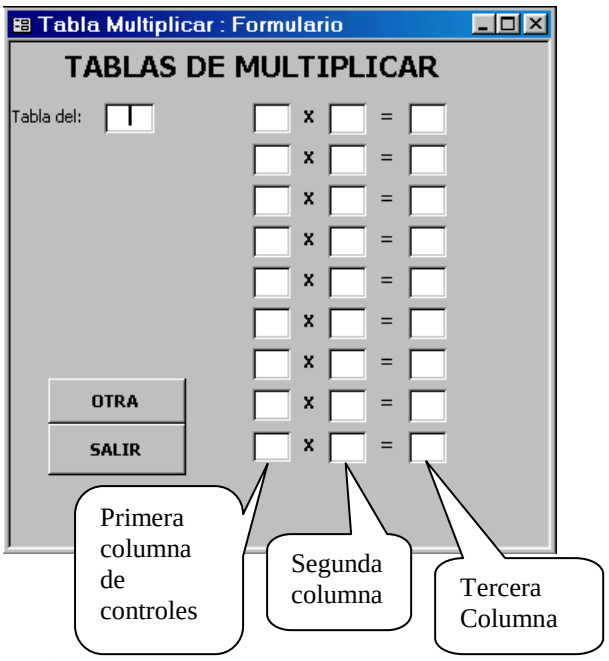
- Clic en el icono **Copiar**
- Clic en el icono **Pegar**, 8 veces.

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) : ....**

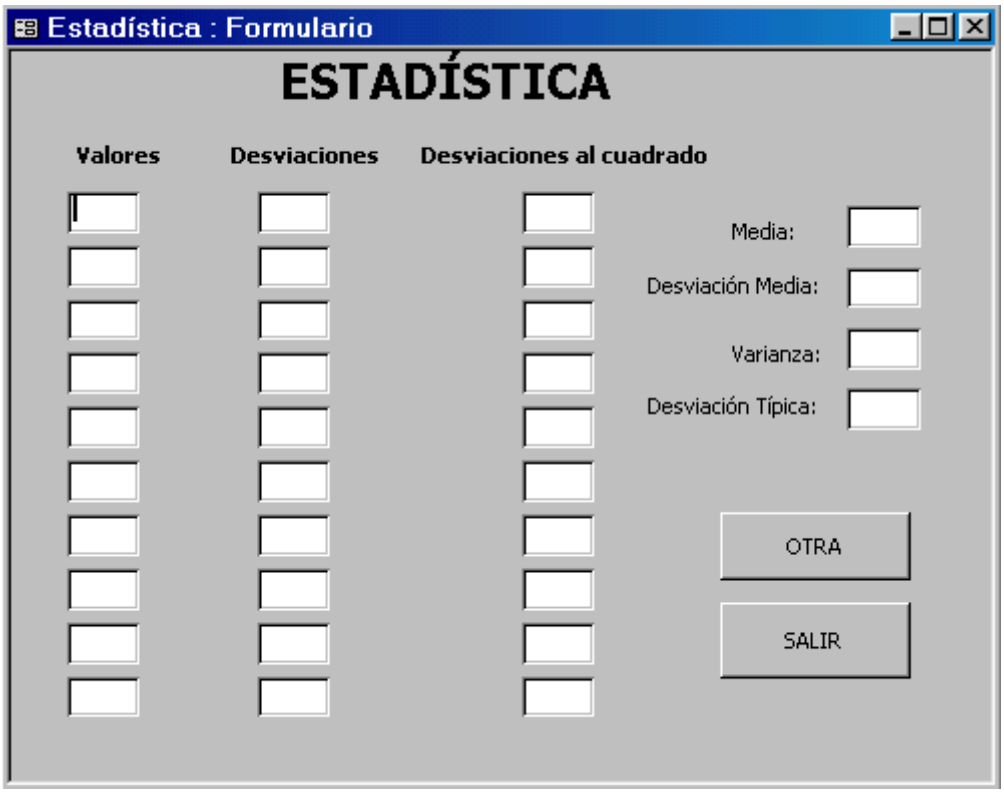
- Introduce las mismas opciones de **inicio** que en el ejercicio anterior.

- Graba la base de datos en formato **MDE**, pero con el nombre **Eval2Bbis**

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

- Graba la base de datos en formato **MDE**, pero con el nombre **Eval2Cbis**.

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

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

  x / MCD(x,y)
  ----
  y / MCD(x,y) 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
```

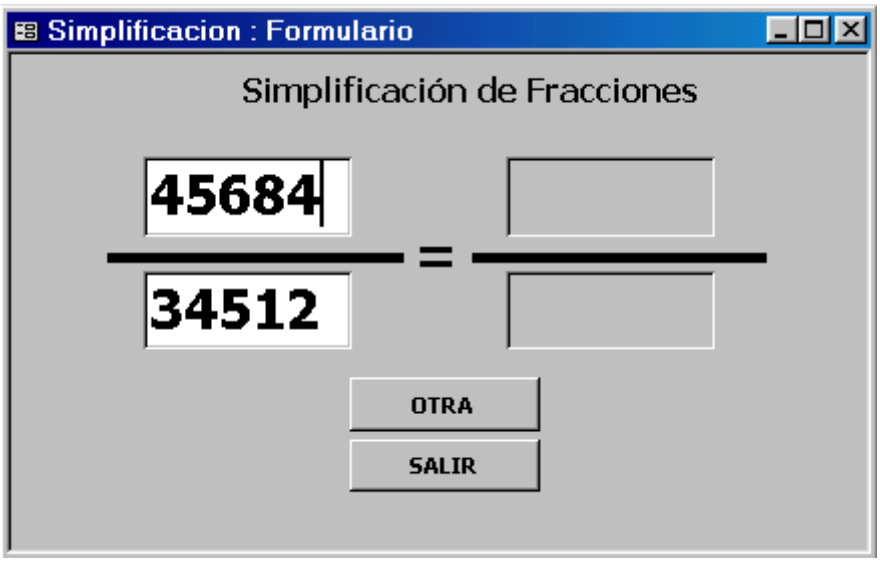
Observa:

- **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"**.

Soluciones de Autoevaluación 2

1) **Eval2A**

```
Private Sub radio_Exit(Cancel As Integer)
   DoCmd.OpenForm "Formul2"
   Dim r As Double, l As Double, a As Double
   If [radio] = "" Then
      Exit Sub
   End If
   r = [radio]
   l = 2 * 3.141592 * r
   a = 3.141592 * r * r
   Forms![Formul2].[longitud] = l
   Forms![Formul2].[area] = a
End Sub



Private Sub otro_Click()
   [longitud] = ""
   [area] = ""
   Forms![Formul1].[radio] = ""
   Forms![Formul1].SetFocus
End Sub


Private Sub salir_Click()
   [longitud] = ""
   [area] = ""
   DoCmd.Close
   Forms![Formul1].[radio] = ""
   DoCmd.Close
   DoCmd.Quit acExit
End Sub
```

2) **Eval2B**

```
Private Sub num_Exit(Cancel As Integer)
Dim a(0 To 8) As Integer
Dim b(0 To 8) As Integer
Dim c(0 To 8) As Integer
For i = 0 To 8
```

```
    a(i) = i + 1
    b(i) = [num]
    c(i) = a(i) * b(i)
Next
[a0] = a(0): [a1] = a(1): [a2] = a(2): [a3] = a(3)
[a4] = a(4): [a5] = a(5): [a6] = a(6): [a7] = a(7): [a8] = a(8)

[b0] = b(0): [b1] = b(1): [b2] = b(2): [b3] = b(3)
[b4] = b(4): [b5] = b(5): [b6] = b(6): [b7] = b(7): [b8] = b(8)

[c0] = c(0): [c1] = c(1): [c2] = c(2): [c3] = c(3)
[c4] = c(4): [c5] = c(5): [c6] = c(6): [c7] = c(7): [c8] = c(8)
End Sub



Private Sub otra_Click()
    [num] = ""
[a0] = "": [a1] = "": [a2] = "": [a3] = ""
[a4] = "": [a5] = "": [a6] = "": [a7] = "": [a8] = ""

[b0] = "": [b1] = "": [b2] = "": [b3] = ""
[b4] = "": [b5] = "": [b6] = "": [b7] = "": [b8] = ""

[c0] = "": [c1] = "": [c2] = "": [c3] = ""
[c4] = "": [c5] = "": [c6] = "": [c7] = "": [c8] = ""
[num].SetFocus
End Sub


Private Sub salir_Click()
    DoCmd.Quit acExit
End Sub
```

3) **Eval2C**

```
Private Sub otra_Click()
    [x1] = "": [x2] = "": [x3] = ""
    [x4] = "": [x5] = "": [x6] = ""
    [x7] = "": [x8] = "": [x9] = ""
    [x10] = ""
    [des1] = "": [des2] = "": [des3] = ""
    [des4] = "": [des5] = "": [des6] = ""
    [des7] = "": [des8] = "": [des9] = ""
    [des10] = ""
```

```
    [descua1] = "": [descua2] = "": [descua3] = ""
    [descua4] = "": [descua5] = "": [descua6] = ""
    [descua7] = "": [descua8] = "": [descua9] = ""
    [descua10] = ""
    [med] = "": [d] = "": [var] = "": [destip] = ""
    [x1].SetFocus
End Sub



Private Sub salir_Click()
    DoCmd.Quit acExit
End Sub



Private Sub x10_Exit(Cancel As Integer)
    Dim x(1 To 10) As Double
    Dim des(1 To 10) As Double
    Dim medv As Double, dv As Double
    Dim descua(1 To 10) As Double
    Dim varv As Double, destipv As Double
    x(1) = [x1]: x(2) = [x2]: x(3) = [x3]
    x(4) = [x4]: x(5) = [x5]: x(6) = [x6]
    x(7) = [x7]: x(8) = [x8]: x(9) = [x9]
    x(10) = [x10]
    medv = Media10(x())
    For i = 1 To 10
        des(i) = Abs(x(i) - medv)
    Next
    [des1] = des(1): [des2] = des(2): [des3] = des(3)
    [des4] = des(4): [des5] = des(5): [des6] = des(6)
    [des7] = des(7): [des8] = des(8): [des9] = des(9)
    [des10] = des(10)
    dv = Media10(des())
    For i = 1 To 10
        descua(i) = des(i) * des(i)
    Next
    [descua1] = descua(1): [descua2] = descua(2): [descua3] = descua(3)
    [descua4] = descua(4): [descua5] = descua(5): [descua6] = descua(6)
    [descua7] = descua(7): [descua8] = descua(8): [descua9] = descua(9)
    [descua10] = descua(10)
    varv = Media10(descua())
    destipv = Sqr(varv)
    [med] = medv
    [d] = dv
    [var] = varv
    [destip] = destipv
```

```
End Sub
```

4) **Eval2D**

```
Private Sub cden1_Exit(Cancel As Integer)
[cnum2] = [cnum1] / MCD([cnum1], [cden1])
[cden2] = [cden1] / MCD([cnum1], [cden1])
End Sub


Private Sub otra_Click()
   [cnum1] = "": [cden1] = ""
   [cnum2] = "": [cden2] = ""
   [cnum1].SetFocus
End Sub


Private Sub salir_Click()
   DoCmd.Quit acExit
End Sub
```

5) **Eval2E**

```
Private Sub Campo3_Exit(Cancel As Integer)
   If [Campo3] > 1000 Then
      [Campo7] = "Demasiado grande"
      [Campo4] = "Mayor que 1000"
      [Campo6].SetFocus
   Else
      [Campo7] = "Vale"
      [Campo4] = "No es mayor que 1000"
   End If
End Sub


Private Sub Form_Close()
   MsgBox "Atención, se va a cerrar el formulario"
End Sub
```

Hasta aquí, la versión no registrada del manual.

Si deseas la parte que falta, es decir:

3 Formularios y Controles 121

Autoevaluación 3 153

Soluciones 3 157

4 Objetos de Acceso a Datos (DAO) 163

Autoevaluación 4 184

Soluciones 4 188

A Programación en SQL 191

B Tratamiento de errores y Depuración 205

C Los "otros" VBA 215a224

Debes adquirir la versión registrada, es decir entera.
Es muy fácil, has de hacer lo siguiente:

1) **Rellena el siguiente formulario con tus datos:**

Nombre y Apellidos: _____

Dirección: _____

Código Postal: _____ **Población:** _____

Versión completa del "VBA Access 2000 (Manual FV)"

2) **Envíame el formulario anterior por correo ordinario junto con un "billete" (lo que consideres justo por un disquete, gastos de manipulación, envío y "estímulo por mi parte para que continue colgando en Internet, mis manuales").**

   A mi dirección que es:

   F.V.
   c) Valencia 21-25, 2º , 4ª
   08915 – Badalona (Barcelona)
   España

3) **A vuelta de correo recibirás en tu dirección, un disquete con la segunda parte del manual "VBA Access 2000 (Manual FV)".**