

CBS

Colegio Bautista Shalom



Ofimática II

Quinto PMP

Cuarto Bimestre

Contenidos

PROCEDIMIENTOS Y FUNCIONES

- ✓ DEFINIR UN PROCEDIMIENTO.
- ✓ LLAMAR A UN PROCEDIMIENTO.
- ✓ GENERALIZAR UNA FUNCIÓN.
- ✓ PARÁMETROS.
- ✓ VARIABLES LOCALES Y VARIABLES GLOBALES.
- ✓ PASO POR REFERENCIA Y PASO POR VALOR.
- ✓ FUNCIONES.
- ✓ MOSTRAR LA BARRA DE HERRAMIENTAS PARA CONTROLES ActiveX.
- ✓ CAMBIAR EL TEXTO DEL CONTROL LABEL.
- ✓ LOS EVENTOS.
- ✓ CUADROS COMBINADOS (ComboBox).
- ✓ CONTROL NUMÉRICO.
- ✓ CELDAS DE VERIFICACIÓN (CheckBox).
- ✓ BOTONES DE OPCIÓN (Option Button).

NOTA: conforme avances en tu aprendizaje de este contenido, tu catedrático(a) indicará la actividad o ejercicio a realizar.

PROCEDIMIENTOS Y FUNCIONES

Se define como procedimiento i/o función a un bloque de código que realiza alguna tarea. Cada tarea la realizará un procedimiento, si esta tarea implica la ejecución de otras tareas, cada una se implementará y solucionará en su correspondiente procedimiento de manera que cada uno haga una cosa concreta. Así, los diferentes pasos que se deben ejecutar para que un programa haga algo, quedaran bien definidos cada uno en su correspondiente procedimiento, si el programa falla, fallará a partir de un procedimiento y de esta forma podremos localizar el error más rápidamente. Los procedimientos son también un eficaz mecanismo para evitar la repetición de código en un mismo programa e incluso en diferentes programas. Suponemos que habrá intuido que hay muchas tareas que se repiten en casi todos los programas veremos cómo los procedimientos que ejecutan estas tareas se pueden incluir en un módulo de forma que este sea exportable a otros programas y de esta manera ganar tiempo.

DEFINIR UN PROCEDIMIENTO

Sub Nombre_Procedimiento

Sentencias.

End Sub.

LLAMAR A UN PROCEDIMIENTO

Para llamar un procedimiento desde otro se utiliza la instrucción **Call** *Nombre_Procedimiento*.

Sub P_Uno

Sentencias.

.

Call P_Dos

.

Sentencias

.

End Sub

Las secuencias del procedimiento *P_Uno* se ejecutan hasta llegar a la línea **Call** *P_Dos*, entonces se salta al procedimiento *P_Dos*, se ejecutan todas las sentencias de este procedimiento y el programa continua ejecutándose en el procedimiento *P_Uno* a partir de la sentencia que sigue a **Call** *P_Dos*.

Ejemplo

Es el mismo programa que el visto en el **ejemplo 29** pero el código que salta celda hasta que se encuentra una vacía se implementa en un procedimiento llamado, *Saltar_Celdas_Llenas*. Observe que para entrar valores se ha sustituido Do While..Loop por Do.. Loop While.

Sub Ejemplo_32()

Dim Nombre **As** String

Dim Ciudad **As** String

Dim Edad **As** Integer

Dim fecha **As** Date

' Llamada a la función Saltar_Celdas_Llenas, el programa salta aquí a ejecutar las

'instrucciones de este procedimiento y luego vuelve para continuar la ejecución a partir de la

'instrucción Do

Call Saltar_Celdas_Llenas

Do

Nombre = InputBox("Entre el Nombre (Return para Terminar) : ", "Nombre")

Ciudad = InputBox("Entre la Ciudad : ", "Ciudad")

Edad = Val(InputBox("Entre la Edad : ", "Edad"))

Fecha=Cdate(InputBox("Entra la Fecha : ", "Fecha"))

With ActiveCell

.Value = Nombre

.Offset(0,1).Value = Ciudad

.Offset(0,2).Value = Edad

.Offset(0,3).value = fecha

End With

ActiveCell.Offset(1,0).Activate

Mas_datos = MsgBox("Otro registro ?", vbYesNo+vbQuestion,"Entrada de datos")

Loop While Mas_Datos = vbYes

End Sub

' Función que salta celdas de una misma columna. Si rve para encontrar la primera celda vacía de la

' columna

Sub Saltar_Celdad_Llenas()

```
Worksheets("Hoja1").Activate
```

```
ActiveSheet.Range("A1").Activate
```

```
Do While not IsEmpty(ActiveCell)
```

```
ActiveCell.Offset(1,0).Activate
```

```
Loop
```

```
End Sub
```

GENERALIZAR UNA FUNCIÓN

Observe que para saltar un rango de celdas llenas sólo necesitará llamar a la función Saltar_Celdas_Llenas, pero, siempre y cuando este rango esté en una hoja llamada "Hoja1" y empiece en la celda A1, el procedimiento es poco práctico ya que su ámbito de funcionamiento es limitado. En la siguiente sección modificaremos el procedimiento de manera que sirva para recorrer un rango que empiece en cualquier celda de cualquier hoja.

PARÁMETROS

Los parámetros son el mecanismo por el cual un procedimiento puede pasarle valores a otro y de esta forma condicionar, moldear, etc. las acciones que ejecuta. El procedimiento llamado gana entonces en flexibilidad. La sintaxis de llamada de un procedimiento es la siguiente,

```
Call Procedimiento(Parámetro1, Parámetro2,..., ParámetroN)
```

Los parámetros pueden ser valores o variables.

La sintaxis para el procedimiento llamado es la siguiente,

```
Sub Procedimiento(Parámetro1 as Tipo, Parámetro2 As Tipo,..., Parámetro3 As Tipo)
```

Observe que aquí los parámetros son variables que recibirán los valores, evidentemente debe haber coincidencia de tipo. Por ejemplo, si el primer parámetro es una variable tipo Integer, el primer valor que se le debe pasar al procedimiento cuando se llama también ha de ser de tipo Integer (recuerde que puedeser un valor directamente o una variable).

Ejemplo 33

El mismo programa que en el ejemplo 32 pero ahora la función Saltar_Celdas_Llenas tiene dos parámetros Hoja y Celda_Inicial que reciben respectivamente la hoja donde está el rango a recorrer y la celda inicial del rango.

```
Sub Ejemplo_33()
```

```
Dim Nombre As String
```

```
Dim Ciudad As String
```

```
Dim Edad As Integer
```

```
Dim fecha As Date
```

```
' Llamada a la función Saltar_Celdas_Llenas, observar que mediante dos  
parámetros se
```

' Al procedimiento en que hoja está el rango a saltar y en la celda donde debe empezar.

Call Saltar_Celdas_Llenas("Hoja1", "A1")

Do

"Nombre")

Ciudad = InputBox("Entre la Ciudad : ", "Ciudad")

Edad = Val(InputBox("Entre la Edad : ", "Edad"))

Fecha=Cdate(InputBox("Entre la Fecha : ", "Fecha"))

With ActiveCell

.Value = Nombre

.Offset(0,1).Value = Ciudad

.Offset(0,2).Value = Edad

.Offset(0,3).value = fecha

End With

ActiveCell.Offset(1,0).Activate

Mas_datos = MsgBox("Otro registro ?", vbYesNo+vbQuestion,"Entrada de datos")

Loop While Mas_Datos = vbYes

,

' Procedimiento Saltar_Celdas_Llenas.

' Sirve para Saltar celdas llenas de una columna hasta encontrar una vacía que se convierte en activa ' Parámetros :

' Hoja : Hoja donde está el rango a saltar.

' Celda_Inicial : Celda Inicial de la columna

Sub Saltar_Celdas_Llenas(Hoja **As String**, Celda_Inicial **As String**)

WorkSheets(Hoja).Activate

ActiveSheet.Range(Celda_Inicial).Activate

```
Do While not IsEmpty(ActiveCell)
```

```
    ActiveCell.Offset(1,0).Activate
```

```
Loop
```

```
End Sub
```

Observe que ahora el procedimiento Salto_Celdas_Llenas sirve para recorrer cualquier rango en cualquier hoja. Observe que al procedimiento se le pasan dos valores directamente, recuerde, y esto es quizás lo más habitual, que también pueden pasarse variables, por ejemplo.

```
Sub Ejemplo_33
```

```
.
```

```
.
```

```
Dim Hoja As String
```

```
Dim Celda_Inicial As String
```

```
Hoja = InputBox("En que hoja está la base de datos : ", "Entrar Nombre de Hoja")
```

```
Celda_Inicial = InputBox("En que celda comienza la base de datos", "Celda Inicial")
```

```
' Observe que los parámetros son dos variables cuyo valor se ha entrado desde teclado en
```

```
' las dos instrucciones InputBox anteriores.
```

```
Call Salto_Celdas_Llenas(Hoja, Celda_Inicial)
```

```
.
```

```
.
```

```
End Sub
```

VARIABLES LOCALES Y VARIABLES GLOBALES

El ámbito de una variable declarada dentro de una función es la propia función, es decir no podrá utilizarse fuera de dicha función. Así, el siguiente programa que debería sumar las cinco filas siguientes a partir de la celda activa y guardar el resultado en la sexta es incorrecto.

```
Sub Alguna_Cosa()
```

```
.
```

```
..
```

```
Call Sumar_Cinco_Siguientes
```

```
ActiveCell.Offset(6,0).Value = Suma
```

```
.
```

Es incorrecto porque tanto la variable *i* como la variable Suma están declaradas dentro del procedimiento Sumar_Cinco_Siguientes consecuentemente, su ámbito de acción es este procedimiento. Por tanto, la instrucción *ActiveCell.Offset(6,0).Value = Suma* del procedimiento *Alguna_Cosa*, generaría un error (con Option Explicit activado) ya que la variable Suma no está declarado dentro de él. Si piensa en declarar la variable Suma dentro del procedimiento *Hacer_Algo*, no solucionará nada porque esta será local a dicho procedimiento, en este caso tendría dos variables llamadas Suma, pero cada una de ellas local a su propio procedimiento y consecuentemente con el ámbito de acción restringido a ellos. Una solución, que a nosotros no nos gusta, sería declarar suma como variable global.

Una variable global se declara fuera de todos los procedimientos y es reconocida por todos los procedimientos del módulo,

Option Explicit

' Suma es una variable global reconocida por todos los procedimientos del módulo.

```
Dim Suma As Single
```

```
Sub Alguna_Cosa()
```

```
.
```

```
Call Sumar_Cinco_Siguientes
```

```
ActiveCell.Offset(6,0).Value = Suma
```

```
.
```

```
End Sub
```

```
Sub Sumar_Cinco_Siguientes()
```

```
Dim i As Integer
```

```
Suma=0
```

```
For i=1 To 5
```

```
Suma = Suma+ActiveCell.Offset(i,0).Value
```

```
Next i
```

```
End Sub
```

Las variables globales son perfectas en ciertas ocasiones, para este caso sería mejor declarar Suma en la función *Hacer_Algo* y pasarla como parámetro al procedimiento *Sumar_Cinco_Siguientes*.

.

End Sub

Sub Sumar_Cinco_Siguientes()

Dim i As Integer

Dim Suma **As Single**

Suma=0

For i=1 **To** 5

Suma = Suma+ActiveCell.Offset(i,0).Value

Next i

End Sub

Sub Alguna_Cosa()

Dim Suma **As Single**

.

.

' Llamada a la función Sumar_Cinco_Siguientes pasándole la variable Suma

Call Sumar_Cinco_Siguientes(Suma)

ActiveCell.Offset(6,0).Value = Suma

.

.

End Sub

Sub Sumar_Cinco_Siguientes(S **As Single**)

Dim i **As Integer**

Suma=0

For i=1 **To** 5

S = S+ActiveCell.Offset(i,0).Value

Next i

End Sub

Esto le funcionaria porque la variable parámetro S (y se le ha cambiado el nombre adrede) de *Sumar_Cinco_Siguientes* es la variable Suma declarada en *Hacer_Algo*. Funcionará porque en visual basic, a menos que se indique lo contrario, el paso de parámetros es por referencia, vea la siguiente sección.

PASO POR REFERENCIA Y PASO POR VALOR

El paso de parámetros por valor y el paso de parámetros por referencia, sólo indican que el paso por valor significa que la variable parámetro del procedimiento recibe el valor de la variable (o directamente el valor) de su parámetro correspondiente e de la instrucción de llamada y en el paso por referencia, la variable parámetro del procedimiento es la misma que su parámetro correspondiente de la instrucción de llamada, es decir, la declarada en el procedimiento desde el que se hace la llamada. Por defecto, y siempre que en la instrucción de llamada se utilicen variables, las llamadas son por referencia. Si desea que el paso de parámetros sea por valor, debe anteponer a la variable parámetro la palabra reservada **ByVal**, por ejemplo,

Sub Saltar_Celdas_Llenas(**ByVal** Hoja As String, **ByVal** Celda_Inicial As String)

Ejemplo Efecto_Lateral.

Antes de copiar el programa, active una hoja en blanco y ponga valores del 1 al 15 distribuidos de la forma siguiente, en el rango A1:A5 valores del 1 al 5, en el rango B1:B5 valores del 6 al 10, en el rango C1:C5 valores del 11 al 15. El siguiente programa debe recorrer cada una de tres las columnas de valores, sumarlos y poner el resultado en las filas 6 de cada columna. Entonces, según los valores que ha entrado en cada una de las columnas, cuando haya acabado la ejecución del programa debe haber los siguientes resultados, A6 = 15, B6=40, C6=65. Para llevar a cabo la suma de los valores de cada columna se llama a la función *Recorrer_Sumar* tres veces, una para cada columna, esta función recibe en el parámetro *F* el valor de la fila donde debe empezar a sumar, sobre el parámetro *C* el valor de la columna a sumar y sobre el parámetro *Q* la cantidad de filas que ha de recorrer.

El programa utiliza la propiedad **Cells** para referenciar las filas y columnas de los rangos.

Observe atentamente los valores que irá cogiendo la variable Fila ya que esta será la que sufra el efecto lateral.

Sub Efecto_Lateral()

Dim Fila As Integer

Fila = 1

Call Recorrer_Sumar(Fila, 1,5) ' Columna A

Call Recorrer_Sumar(Fila, 2,5) ' Columna B

Call Recorrer_Sumar(Fila, 3,5) ' Columna C

End Sub

Sub Recorrer_Sumar(F As Integer, C As Integer, Q As Integer)

Dim i As Integer

Dim Total As Integer

Total = 0

For i =1 To Q

Total = Total + ActiveSheet.Cells(F, C).Value

F=F+1 ' *OJO con esta asignación, recuerde que F es la variable Fila declarada en*

' el procedimiento *Efecto_Lateral*

Next i

ActiveSheet.Cells(F, C) = Total

End Sub

Cuando ejecute el programa se producirá la salida siguiente, en A6 habrá un 15, hasta aquí todo correcto, pero observe que en la segunda columna aparece un 0 en B12 y en la tercera columna aparece un 0 en C18, veamos que ha pasado. La primera vez que se llama la función, la variable F vale 1 ya que este es el valor que tiene su parámetro correspondiente (*Fila*) en la instrucción **Call**. Observe que F se va incrementando una unidad a cada paso de bucle **For**, RECUERDE que F es realmente la variable *Fila* declarada en el procedimiento *Efecto_Lateral*, por tanto cuando finaliza el procedimiento *Recorrer_Sumar* y vuelve el control al procedimiento *Efecto_Lateral* *Fila* vale 6, y este es el valor que se pasará a *Recorrer_Suma* la segunda vez que se llama, a partir de ahí todo irá mal ya que se empezará el recorrido de filas por la 6. Una de las soluciones a este problema para hacer que cada vez que se llame *Recorrer_Sumar* la variable F reciba el valor 1, es utilizar un paso por valor, es decir que F reciba el valor de *Fila*, no que sea la variable *Fila*, observe que entonces, si F no es la variable *Fila*, cuando incremente F no se incrementará *Fila*, está siempre conservará el valor 1. Para hacer que F sea un parámetro por valor, simplemente ponga la palabra **ByVal** antes de F en la declaración del procedimiento.

FUNCIONES

Una función es lo mismo que un procedimiento con la salvedad que este devuelve un valor al procedimiento o función que lo llama. Vea el siguiente ejemplo, es una función muy sencilla ya que simplemente suma dos números y devuelve el resultado.

Ejemplo 34.

Función que devuelve la suma de dos valores que se le pasan como parámetros. Observe las diferentes formas en cómo se llama la función.

Ejemplo 34.

Función que devuelve la suma de dos valores que se le pasan como parámetros.

Observe las diferentes formas en como se llama la función.

Sub Ejemplo_34()

Dim x **As Integer**

Dim n1 **As Integer**, n2 **As Integer**

X = Suma(5, 5)

n1= Val (InputBox("Entrar un número : ", "Entrada"))

n2= Val (InputBox("Entrar otro número : ", "Entrada"))

X= suma(n1,n2)

ActiveCell.Value = Suma(ActiveSheet.Range("A1").Value ,
ActiveSheet.Range("A2").Value)

X = Suma(5, 4) + Suma (n1, n2)

End Sub

```
Function Suma(V1 As Integer, V2 As Integer) As Integer
```

```
    Dim Total As Integer
```

```
    Total = V1 + V2
```

```
    Suma = Total
```

```
End Function
```

Observe la sintaxis de la cabecera de función,

```
Function Suma(V1 As Integer, V2 As Integer) As Integer
```

La estructura general sería,

```
Function Nombre_Funcion(par1 As Tipo, par2 As Tipo,..., parN As Tipo) As Tipo.
```

La sintaxis es similar a la cabecera de un procedimiento, sólo que una función tiene tipo, esto tiene su lógica, ya que una función devuelve un valor, ese valor será de un tipo determinado. Así, en nuestro ejemplo de **Function** Suma, esta función es de tipo **Integer** o, dicho de otra manera, la función ejecuta sus sentencias y devuelve un valor hacia el procedimiento o la función que la llamó, el valor devuelto se establece igualando el nombre de la función a algo,

```
Nombre_Función = ....
```

En el ejemplo de **Function** Suma,

```
Suma = Total
```

Observe también la sintaxis de la llamada a la función, en el ejemplo hemos utilizado unas cuantas formas de llamarla, lo que debe tener siempre presente es que en cualquier expresión aritmética o de cálculo, el ordenador realiza un mínimo de dos operaciones, una de cálculo y otra de asignación. Por ejemplo, A= B+C El ordenador primero calcula el resultado de sumar B+C luego asigna ese resultado a la variable A. En cualquier llamada a una función, cojamos por caso,

```
X= suma(n1,n2)
```

Aplicación de ejemplo

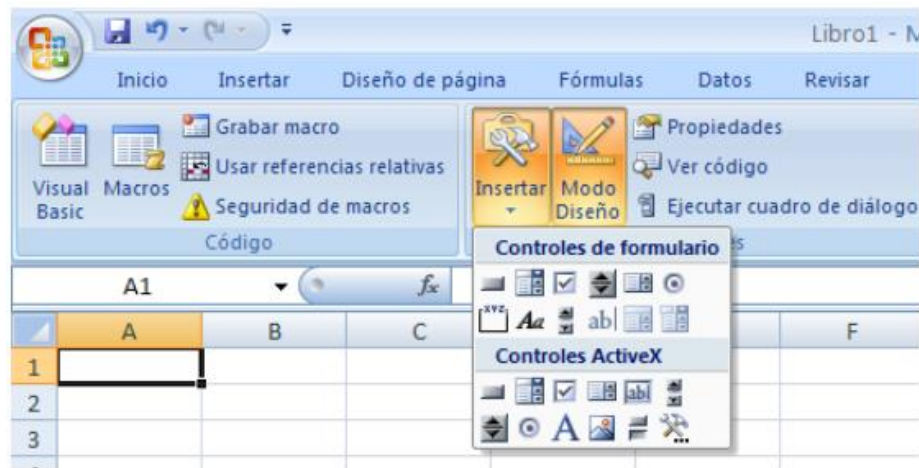
Para ver el funcionamiento de los distintos controles, construiremos una pequeña aplicación que nos sirva para gestionar una pequeña tabla de registros, básicamente extraer datos que cumplan una determinada condición. La mayoría de las funciones que aplicaremos pueden hacerse directamente desde las utilidades del menú **Datos/Filtro avanzado** que lleva incorporado el propio Excel, pero creemos que será un buen ejemplo para ver las posibilidades de los controles.

Abra su aplicación Excel y active la hoja de cálculo Lista7.xls, en la primera hoja está la lista de registros que utilizaremos en los ejemplos que veremos a continuación.

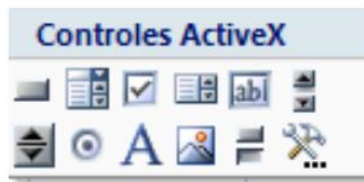
En la Hoja2 se encuentran algunos datos necesarios para los controles.

MOSTRAR LA BARRA DE HERRAMIENTAS PARA CONTROLES ActiveX.

Para insertar controles en la hoja deberá tener activada la barra de controles ActiveX que se encuentra en la pestaña Programador opción Insertar, allí se encuentran las barras de Controles de formulario y de Controles ActiveX



La barra de Controles ActiveX permite programar cada uno de los controles.



CUADRO DE TEXTO Y BOTÓN

Lo primero que haremos es algo muy sencillo, simplemente copiaremos en Hoja2, los datos correspondientes a los registros de Hoja1 cuyos datos correspondientes a la columna *Nombre* coincidan con el que teclearemos en una cuadro de texto que insertaremos en Hoja2. Los datos se copiarán a partir de la celda A16. El botón simplemente servirá para invocar la macro que copiará los datos.

INSERTAR EL CUADRO DE TEXTO

Sólo tiene que seleccionar el elemento  de la barra de controles y dibujarlo sobre la hoja (Hoja 2 en nuestro ejemplo, procure que coja más o menos el rango correspondiente a las celdas C2 y D2).

INSERTAR UNA ETIQUETA

Las etiquetas sirven básicamente para acompañar los controles con texto descriptivo. Seleccione el botón  y dibuje en la hoja el rectángulo para insertar la etiqueta, póngalo al lado del control cuadro de texto.

INSERTAR UN BOTÓN

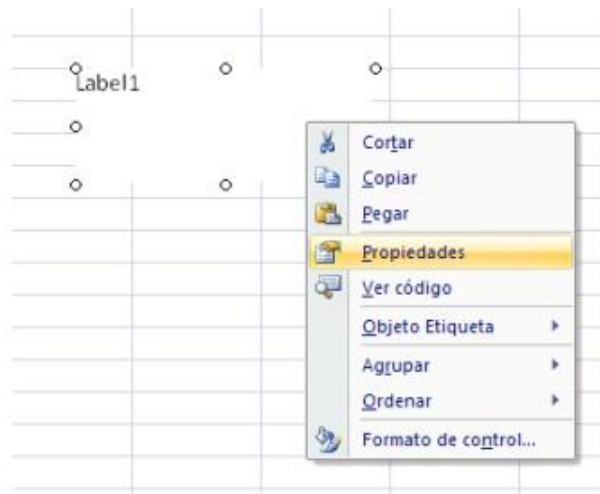
Los botones  se utilizan básicamente para invocar las macros que realizarán las acciones. No es el único control que puede invocar macros, cualquiera de los controles puede invocarla, pero es el más habitual.

Cambiar las propiedades de los objetos. A continuación, desplegaremos la ventana de propiedades para cambiar

algunas de los objetos acabados de incrustar. Debe estar en modo diseño,  el botón de estar activado.

CAMBIAR EL TEXTO DEL CONTROL LABEL

Propiedad caption



1. Seleccione el control **Etiqueta**.
2. Pulse sobre el botón  de la barra de controles, se activa la ventana de Propiedades.
3. En la propiedad **Caption**, cambien el texto **Label1** por **Datos a Buscar**.
4. Ajuste la posición y el tamaño del control.

Cambiar el nombre del control cuadro de texto

Propiedad name

No es necesario cambiar el nombre de los controles, pero si muy conveniente, tenga en cuenta que a través de los nombres de un control será como se refiera a ellos a través de las macros. Siempre es mejor llamar a un control por un nombre descriptivo que por Text1 o Command1, etc. Al control cuadro de texto le pondremos el nombre **Datos_Buscar**.

1. Seleccione el control Cuadro de Texto.
2. Si no tiene activada la ventana de propiedades, actívela.
3. En la propiedad **Name**, cambie el text1 por **Datos_Buscar**.

Cambie la propiedad **Caption** del Botón por **Copiar Datos** y su propiedad **Name** por **Copiar_Datos** (debe poner el guion bajo ya que la propiedad **Name** no permite espacios en blanco).

Establecer la acción de copiar datos cuando se pulse el botón

A continuación, crearemos la macro que será invocada cuando se pulse el botón. La macro simplemente debe buscar en la columna A de la lista de Hoja1 el nombre que coincida con el teclado en el cuadro de texto y luego copiarlo hacia Hoja2 a partir de la celda A16. La macro controlará que haya algo en el cuadro de texto. Se copiarán todas las coincidencias, es decir si hay dos nombres Ramón se copiarán los dos. Si no hay ninguna coincidencia se mostrará un mensaje avisando de ello.

LOS EVENTOS

Cuando se programan controles bien sea directamente en la hoja como estamos haciendo ahora o desde un formulario, debe tener en cuenta los eventos. Un evento es cuando ocurre algo sobre un objeto, en entornos

Windows constantemente se está produciendo eventos. Clicks con el ratón sobre un control, teclear sobre un cuadro de texto, etc. provocan eventos que son recogidos por el sistema. Programar un evento significa hacer que se ejecuten determinadas instrucciones cuando ocurra dicho evento. En el caso que nos ocupa ahora, haremos que las acciones necesarias para copiar los datos se ejecuten cuando se haga un clic sobre el botón **Copiar_Datos**. En general, todos los controles son capaces de capturar diferentes eventos. El sistema de eventos es bastante más complejo de lo que estudiaremos aquí, nosotros simplemente tendremos en cuenta que evento debemos elegir para lanzar la ejecución de determinado código. Veamos en la siguiente sección como asociar el código necesario para copiar datos cuando ocurre el evento click (pulsar el botón y soltarlo) sobre el botón **Copiar_Datos**.

Escribir código para el evento Click del Botón



Deberá estar en modo **Diseño**, asegúrese que el botón está pulsado.

1. Haga doble click sobre el botón, observe que se activa automáticamente la ventana de Visual Basic y aparece un esqueleto de función

```
Sub Copiar_Datos_Click()
```

```
End Sub
```

Es lo que se llama procedimiento de evento, es decir, este procedimiento está asociado al evento Click del Botón **Copiar_Datos**, observe que el procedimiento lleva un nombre compuesto por el nombre del control "Copiar_Datos", un guión bajo y el nombre del evento "Click", en general todos los procedimientos de evento se nombran de esta forma:

NombreDeControl_NombreDeEvento

Observe la lista de la parte superior derecha, la que tiene el elemento **Click**. Es la lista de eventos, si la despliega verá que además del elemento **Click** aparecen unos cuantos más **DbClick** (Doble Click) **GotFocus** (Coger el foco), etc. todos ellos son eventos programables del control botón, es decir, podemos incluir código que se ejecutará cuando ocurren dichos eventos. Por otra parte, todos los controles tienen un evento "por defecto", dicho de otra forma, cuando se programa un evento del control casi siempre será ese. En el caso de nuestro botón (y de todos los botones), el evento por defecto es **Click**, observe que lo habitual es que queramos que el código se ejecute cuando se hace click sobre el botón, no cuando este coge el foco o cuando el puntero de ratón pasa sobre él, etc. El evento por defecto de un control es el que aparece cuando, en modo diseño, se hace doble clic sobre él, obviamente este se puede cambiar, por el que más nos convenga.

2. Teclear el código para llevar a cabo las acciones. Recuerde que lo que se desea hacer es copiar hacia hoja2 todos los nombres que coincidan con el que está en el cuadro de texto. El código será el que sigue, observe los comentarios.

Option Explicit

' Numero de columnas(campos) de las que consta cada registro

```
Const Num_Columnas = 6
```

```
Private Sub Copiar_Datos_Click()
```

```
    Dim r1 As Range, r2 As Range
```

```
    Dim encontrado As Boolean
```

' Si el cuadro de texto está vacío, no se busca nada

If Len(Datos_Buscar.Value) = 0 **Then**

MsgBox ("No hay datos que buscar")

Else

' Borrar los datos actuales

Call borrar_datos

' Activar Celda A16 de Hoja2 y referenciarla con r2, Es la celda donde se copiarán

'los datos en caso que se encuentren

Worksheets(2).Range("A16").Activate

Set r2 = ActiveCell

' Activar celda A2 de Hoja1 y referenciarla con r1

Worksheets(1).Activate

Worksheets(1).Range("A2").Activate

' Recorrer todo el rango de datos de Hoja1

encontrado = False

Do While Not IsEmpty(ActiveCell)

' Si la celda activa = Datos_Buscados

If ActiveCell.Value = Datos_Buscar.Text **Then**

encontrado = True

' Referenciar con r1 la celda donde están los datos

Set r1 = ActiveCell

' Copiar los datos

Call Copiar_Datos_Hojas(r1, r2)

' Referenciar con r2 la celda donde se copiaran los próximos datos

Set r2 = r2.Offset(1, 0)

```
        End If

        ActiveCell.Offset(1, 0).Activate

    Loop

    Worksheets(2).Activate

    If encontrado Then

        MsgBox ("Datos Copiados")

    Else

        MsgBox ("Ninguna coincidencia")

    End If

End If

End Sub

' Procedimiento para borrar los datos de Hoja2 se llama antes de proceder a la nueva copia

Private Sub borrar_datos()

    Dim i As Integer

    Worksheets(2).Range("A16").Activate

    Do While Not IsEmpty(ActiveCell)

        For i = 0 To Num_Columnas - 1

            ActiveCell.Offset(0, i).Value = ""

        Next i

        ActiveCell.Offset(1, 0).Activate

    Loop

End Sub

' Procedimiento para copiar los datos de Hoja1 a Hoja3

' Parámetros.

' r1 = Celda Origen

' r2 = Celda Destino
```

```
Private Sub Copiar_Datos_Hojas(r1 As Range, r2 As Range)
```

```
    Dim i As Integer
```

```
    Dim Datos As Variant
```

```
    ' Recorrer las columnas del registro y copiar celda a celda
```

```
    For i = 0 To Num_Columnas - 1
```

```
        Datos = r1.Offset(0, i).Value
```

```
        r2.Offset(0, i).Value = Datos
```

```
    Next i
```

```
End Sub
```

CUADROS COMBINADOS (ComboBox)



Con lo hecho hasta ahora podemos extraer de la tabla los registros cuyo nombre coincida con el tecleado en el cuadro de texto. A continuación, haremos que se pueda escoger el campo, es decir, podremos extraer coincidencias del Nombre, los Apellidos, la Ciudad, etc. Para ello incluiremos un cuadro combinado que permita escoger en que campo o columna tiene que buscarse la coincidencia. La lista, por supuesto, mostrará los nombres de las columnas.

Incluya un cuadro combinado en Hoja2 y póngale por nombre (propiedad **Name**). **Lista_Campos**

Propiedad ListFillRange.

Con esta propiedad deberemos definir los elementos que debe mostrar la lista, debe especificarse el rango que contiene los elementos a mostrar, el rango debe ser una columna (o dos, o tres, etc.). En nuestro caso el rango será J1:J6 de Hoja2 (Observe que en este rango están especificados los nombres de las columnas).

Propiedad LinKedCell.

En esta propiedad debe especificar en que celda debe copiarse el elemento seleccionado de la lista. En esta lista no utilizaremos esta propiedad. Cuidado con esta propiedad, tenga en cuenta que los elementos de la lista son tratados como datos de tipo String aunque contenga números o fechas, por lo que en estos casos, a veces será necesario aplicar funciones de conversión de datos antes que el dato se copie en la hoja. Por ejemplo, si alguna vez construye una lista con números verá que el dato seleccionado se alinea a la derecha, si son fechas, no se muestra con el formato correspondiente.

Propiedad ListIndex.

Mediante esta propiedad podremos saber que elemento de la lista es el seleccionado por su número de orden. Es decir, si está seleccionado el primero, ListIndex valdrá 0, si está seleccionado el segundo valdrá 1, etc. Si no hay ningún elemento seleccionado valdrá -1. Tenga en cuenta que esta propiedad sólo está disponible en tiempo de ejecución, es decir la podremos leer mientras esté funcionando el programa, no se puede establecer en modo diseño, observe que no aparece en la ventana propiedades del cuadro combinado.

En primer lugar, cree un procedimiento llamado **Proceder** donde deberá copiar todo el código que ahora está en **Copiar_Datos**. Debemos hacer esto porque antes de proceder se deben hacer ciertas comprobaciones que ya iremos viendo conforme avanzamos, por el momento la comprobación a hacer es la de ver sobre qué campo o columna se deben buscar las coincidencias con los datos tecleados en el cuadro de texto. La función **Copiar_Datos** quedará de la forma siguiente.

Private Sub Copiar_Datos_Click()**Dim i As Integer***' Recoger el elemento seleccionado de la lista**i = Lista_Campos.ListIndex**' Si i < 0 es que no hay ningún elemento seleccionado.***If i < 0 Then****MsgBox** ("Debe Seleccionar un campo de la lista")**Else***' Llamar a proceder para iniciar la copia.***Call** Proceder(i)**End If****End Sub**

La cabecera de la función proceder quedará de la forma siguiente.

*' Procedimiento Proceder**' Inicia la copia de los datos coincidentes**' Parámetros:**' Columna = Elementos seleccionado de la lista que coincidirá con la columna sobre la que se**' debe buscar***Private Sub** Proceder(Columna As Integer)

.....

Ahora, dentro del procedimiento Proceder cambie la línea

If ActiveCell.Value = Datos_Buscar.Text **Then**

Por

If ActiveCell.Offset(0, Columna).Value = Datos_Buscar.Text **Then**

Explicación del proceso. Cuando se selecciona un elemento de la lista, su propiedad **ListIndex** es igual al orden que ocupa dicho elemento en la lista, supongamos que escogemos Ciudad, **ListIndex** valdrá 2.

Este valor se pasa a **Proceder** y es recogido por el parámetro **Columna**. Ahora observe la línea

If ActiveCell.Offset(0, Columna).Value = Datos_Buscar.Text **Then**

Es decir la coincidencia con el valor del cuadro de texto Datos_Buscar se busca respecto a la celda que está a la derecha (offset) de la activa, tantas columnas como las expresadas por el valor de la variable **Columna**. Observe

que en este caso la celda activa siempre corresponde a una de la columna A, si la variable **Columna** vale 2 la coincidencia se buscará respecto al valor de la columna C (2 más hacia la derecha) y que coincide con la columna correspondiente a la **Ciudad**.

Segunda Lista

Ahora crearemos una lista donde sea posible escoger la relación de comparación. Hasta ahora la extracción se realizaba con aquellos elementos iguales al valor entrado en el cuadro de texto **Datos_Buscar**, esta segunda lista permitirá escoger si los elementos a extraer deben ser Iguales, Menores, Mayores, Menores Iguales o Mayores Iguales que el valor de **Datos_Buscar**. Para ello debe construir una segunda lista con las propiedades siguientes.

ListFillRange = L1:L5 Observe que en este rango están los valores correspondientes a la operación relacional que se desea realizar (Igual, Menor, etc.)

Obviamente deberemos modificar las funciones para realizar las operaciones con respecto al elemento seleccionado en el cuadro de lista **Lista_Comparación**. Dejaremos el procedimiento **Proceder** de la forma siguiente y crearemos una función que haga las comparaciones, esta función a la que hemos llamado **Comparar** devuelva el valor True si el resultado de la comparación es Cierto y False si es falso.

' Procedimiento Proceder

' Inicia la copia de los datos coincidentes

' Parámetros:

' Columna = Elementos seleccionado de la lista que coincidirá con la columna sobre la que se debe

' buscar

Private Sub Proceder(Columna As Integer)

Dim r1 As Range, r2 As Range

Dim encontrado As Boolean

Dim Valor_Comparacion As Boolean

' Si el cuadro de texto está vacío, no se busca nada

If Len(Datos_Buscar.Value) = 0 Then

MsgBox ("No hay datos que buscar")

Else

' Borrar los datos actuales

Call borrar_datos

' Activar Celda A16 de Hoja2 y referenciarla con r2' Es la celda donde se copiarán

```
' los datos en caso que' se encuentren
Worksheets(2).Range("A16").Activate

Set r2 = ActiveCell

' Activar celda A2 de Hoja1 y referenciarla con r1
Worksheets(1).Activate
Worksheets(1).Range("A2").Activate
encontrado = False

' Recorrer todo el rango de datos de Hoja1

Do While Not IsEmpty(ActiveCell)

    Valor_Comparacion = Comparar(ActiveCell.Offset(0, Columna).Value, _
    Datos_Buscar.Value, Lista_Comparacion.ListIndex)

    If Valor_Comparacion = True Then

        encontrado = True

        ' Referenciar con r1 la celda donde están os datos

        Set r1 = ActiveCell

        ' Copiar los datos

        Call Copiar_Datos_Hojas(r1, r2)

        ' Referenciar con r2 la celda donde se copiaran los próximos datos

        ' Copiar los datos

        Call Copiar_Datos_Hojas(r1, r2)

        ' Referenciar con r2 la celda donde se copiaran los próximos datos

        Set r2 = r2.Offset(1, 0)

    End If

    ActiveCell.Offset(1, 0).Activate

Loop

Worksheets(2).Activate

If encontrado Then
```

MsgBox ("Datos Copiados")

Else

MsgBox ("Ninguna coincidencia")

End If

End If

End Sub

' Función que compara dos valores con un operador relacional =, >, <, etc.

' La función devuelve True o False en función de la comparación.

' Parámetros.

' Valor1 y Valor2 = Valores que se comparan

' Signo = variable que sirve para escoger el operador relacional

' en función de su valor, ver estructura Select Case

Private Function Comparar(Valor1 **As Variant**, Valor2 **As Variant**, Operador **As Integer**) **As Boolean**

Dim q **As Boolean**

Select Case Operador

Case 0:

q = Valor1 = Valor2

Case 1:

q = Valor1 > Valor2

Case 2:

q = Valor1 < Valor2

Case 3:

q = Valor1 >= Valor2

Case 4:

q = Valor1 <= Valor2

End Select

Comparar = q

End Function

Observe la línea que llama a la función **Comparar**, Valor_Comparacion = Comparar(ActiveCell.Offset(0, Columna).Value, _ Datos_Buscar.Value, Lista_Comparacion.ListIndex).

ActiveCell.Offset(0, Columna) serán los valores que se compararán con el valor del cuadro de texto.

Datos_Buscar.value es el valor del cuadro de texto.

Lista_Comparación.ListIndex devuelve el índice del elemento seleccionado de la lista, observe como se utiliza este valor en la estructura **Select Case** de **Comparar** para determinar que operador utilizar.

Pero todo esto no funcionará.

Pruebe lo siguiente.

En el cuadro de texto escriba Madrid (o simplemente M).

Seleccione de la lista de Campos Ciudad.

Seleccione de la lista de operadores Mayor.

Pulse sobre el botón y observe que se copian todos los registros cuyo campo Ciudad sea superior a Madrid (o a la M).

Hasta aquí todo correcto.

Ahora pruebe lo siguiente. En el cuadro de texto escriba 100000 Seleccione de la lista de Campos Cantidad.

Seleccione de la lista de operadores Mayor.

Pulse sobre el botón y observe que no se copia nada a pesar de que en cantidad hay registros con el valor superior a 100000.

Recuerde que los valores de un cuadro de texto son siempre datos tipo String, entonces en este caso estarán comparándose valores String (los del cuadro de texto) con valores numéricos, (los recogidos de la columna Cantidad). Tenga en cuenta siempre esta circunstancia cuando trabaje con elementos de formulario. Vea la sección siguiente donde se solucionará este problema y de paso se verá cómo construir Lista con más de una columna.

Para solucionar el problema del apartado anterior utilizaremos una segunda columna cuyos elementos indicarán el tipo de datos del campo. Observe el rango K1:K6 de Hoja2, las letras significan lo siguiente *T campo tipo texto o string, N campo Numérico, F campo fecha*. Para incluir esta segunda columna en la lista deberá cambiar las propiedades siguientes.

ListFillRange, J1:K6

ColumnCount, 2 (Indicamos el número de columnas de la lista).

Además, especificaremos el ancho de cada columna mediante la propiedad, **ColumnWidths**, 4pt; 0pt Debe separar el ancho de cada columna mediante un punto y coma.

ColumnBound, 1 significa que el dato que recogerá la propiedad Value corresponde al elemento seleccionado de la primera columna.

Si desea recoger datos de la segunda columna deberá utilizar la propiedad **Column**(Numero de Columna, Indice del elemento seleccionado) Las columnas empiezan a numerarse a partir de la 0.

La función **Comparar** y su correspondiente llamada quedarán de la forma siguiente. Observe que se han incluido variables que recogen los valores de **Lista_Comparación y Lista_Campos**, simplemente lo hemos hecho para que quede más claro.

```
Do While Not IsEmpty(ActiveCell)
```

```
' Recoger el Signo de comparación
```

```
Signo = Lista_Comparacion.ListIndex
```

```
' Recoger el tipo de datos
```

```
Tipo_Datos = Lista_Campos.Column(1, Columna)
```

```
Valor_Comparacion = Comparar(ActiveCell.Offset(0, Columna).Value, _
```

```
Datos_Buscar.Value, Signo, Tipo_Datos)
```

La función Comparar.

```
Private Function Comparar(Valor1 As Variant, Valor2 As Variant, Operador As
```

```
Integer, Tipo As
```

```
String) As Boolean
```

```
Dim q As Boolean
```

```
' Conversión del tipo de datos de las variables
```

```
Select Case Tipo
```

```
Case "N": ' Convertir a número
```

```
Valor2 = Val(Valor2)
```

```
Case "F": ' Convertir a Fecha
```

```
Valor2 = CDate(Valor2)
```

```
End Select
```

```
Select Case Operador
```

```
Case 0:
```

```
q = Valor1 = Valor2
```

```
Case 1:
```

```
q = Valor1 > Valor2
```

Case 2:

q = Valor1 < Valor2

Case 3:

q = Valor1 >= Valor2

Case 4:

q = Valor1 <= Valor2

End Select

Comparar = q

End Function

CONTROL NUMÉRICO



Inserte un control de número y póngale por nombre (propiedad **Name**) **Numero**. Establezca su propiedad **Orientation** a **fmOrientationVertical** para que se alinee verticalmente. Este control se utiliza normalmente para aumentar y disminuir valores numéricos de un cuadro de texto, aunque por supuesto puede utilizarse para otras funciones. Utilizaremos un control de este tipo para aumentar los valores del Cuadro de Texto **Datos_Buscar** pero sólo si el campo seleccionado de **Lista_Campos** es de tipo numérico. Para ello activaremos este control únicamente cuando el campo seleccionado sea de este tipo. Para activar o desactivar un control se utiliza la propiedad **Enabled**, si está a true el control estará activado y sino estará desactivado. Observe que la acción de activar o desactivar el control de número deberemos hacerlo cuando se seleccione un elemento de **Lista_Campos**. Es decir el código deberemos insertarlo en el evento **Change** (cambio) de **Lista_Campos**. Haga doble clic sobre el elemento **Lista_Campos** para desplegar su procedimiento de evento. El código para controlar la activación del control es el que sigue,

Private Sub Lista_Campos_Change()

Dim i **As Integer**

Dim Tipo_Datos **As String**

i = Lista_Campos.ListIndex

If i >= 0 **Then**

Tipo_Datos = Lista_Campos.Column(1, i)

If Tipo_Datos = "N" **Then**

Numero.Enabled = True

Else

Numero.Enabled = False

End If

End If

End Sub

Establecer los valores del control de número

Para establecer los valores que puede devolver un control de número se deben modificar las propiedades siguientes.

Max, establece el valor máximo que puede tener el control.

Min, establece el valor mínimo que puede tener el control. **Smallchange**, establece el incremento o decremento para la propiedad value cada vez que se pulse sobre alguno de los dos botones. Estos valores se pueden establecer en tiempo de diseño, pero lo que haremos a continuación será establecerlos en tiempo de ejecución dependiendo del campo que se seleccione en **Lista_Campos**.

Estableceremos los valores siguientes.

Para campo **Edad**.

Max = 99

Min = 18

SmallChange = 1

Para campo cantidad.

Max = 500.000

Min = 10.000

SmalChange = 1.000

Deberemos modificar el código del procedimiento de evento Lista_Campos_Change de la forma siguiente.

Private Sub Lista_Campos_Change()

Dim i As Integer

Dim Tipo_Datos As String

i = Lista_Campos.ListIndex

If i >= 0 Then

Tipo_Datos = Lista_Campos.Column(1, i)

If Tipo_Datos = "N" Then

Numero.Enabled = True

```
If Lista_Campos.Value = "Edad" Then
```

```
Numero.Min = 18
```

```
Numero.Max = 99
```

```
Numero.SmallChange = 1
```

```
Datos_Buscar.Value = 0
```

```
Numero.Value=0
```

```
End If
```

```
If Lista_Campos.Value = "Cantidad" Then
```

```
Numero.Min = 10000
```

```
Numero.Max = 500000
```

```
Numero.SmallChange = 1000
```

```
Datos_Buscar .Value= 0
```

```
Numero.Value=0
```

```
End If
```

```
Else
```

```
Numero.Enabled = False
```

```
End If
```

```
End If
```

```
End Sub
```

Y para terminar ya sólo debemos codificar el evento **Change** del control de número para que el Cuadro de texto vaya incrementando o decrementando su valor cada vez que se haga clic sobre el control.

```
Private Sub Numero_Change()
```

```
Datos_Buscar.Value = Numero.Value
```

```
End Sub
```

Pero además debemos hacer que cuando se cambie el valor del cuadro de texto también se cambie el del control de número de forma que cuando se pulse sobre él, incremente o decremente a partir del valor que hay en el cuadro de texto. Si no se hace así, el incremento o decremento se hará en función del valor que tenga el control de número no el que está en el cuadro de texto. Modificaremos el evento **Change** del cuadro de texto. Observe que se controla que sólo se ejecute la acción si el control de número está activado, además se debe controlar también el valor

máximo y mínimo que puede contener el cuadro de texto, si no se hiciera así se generaría un error al poner un valor mayor o menor que los definidos en las propiedades Max y Min del control de número.

Private Sub Datos_Buscar_Change()

' Si el numero de control está activado

If Numero.Enabled Then

' No permite coger valores superiores a la propiedad Max

If Val(Datos_Buscar.Value) > Numero.Max Then

MsgBox ("Valor demasiado grande")

Datos_Buscar.Value = Numero.Max

Else

' No permite coger valores inferiores a la propiedad Min

If Val(Datos_Buscar.Value) < Numero.Min Then

MsgBox ("Valor demasiado pequeño")

Datos_Buscar.Value = Numero.Min

Else

Numero.Value = Val(Datos_Buscar.Value)

End If

End If

End If

End Sub

Antes de proceder con el siguiente control déjenos remarcar que la programación de controles implica que muchas veces unos dependan de otros por lo que debe ser extremadamente cuidadoso en la elaboración del código de los eventos.

Observe las últimas operaciones que hemos realizado debido a la interdependencia de dos controles.

CELDA DE VERIFICACIÓN (CheckBox).



Estos controles se suelen utilizar para activar o desactivar la ejecución de determinadas acciones. Casi siempre implican una estructura condicional a la hora de hacer alguna cosa,

Si las celda está activada Entonces

Acciones

....

Fin Si

A continuación, utilizaremos una celda de verificación que si está activada provocará que los datos tipo texto se conviertan a mayúscula antes de copiarse, se utilizará la función **Ucase**. Simplemente se deberá comprobar que la celda esté activada y si lo está proceder a la conversión (sólo si el dato es tipo texto). Inserte un control celda de verificación. Establezca las siguientes propiedades.

Inserte un control celda de verificación. Establezca las siguientes propiedades.

Name, Mayusculas.**Captión**, Mayúsculas.

Para comprobar si la celda está activada o no simplemente debe mirarse su propiedad **Value**. Si vale true es que está activada, sino valdrá False. Vea cómo quedará el procedimiento **Copiar_Datos_Hojas**, observe que además de comprobar que la celda esté activada se utiliza la función **TypeName** para comprobar si los datos que se van a copiar son del tipo String, si no lo fueran, la función **Ucase** provocaría un error. **TypeName**(Expresión) devuelve una cadena que indica el tipo de datos de la expresión.

Private Sub Copiar_Datos_Hojas(r1 As Range, r2 As Range)**Dim i As Integer****Dim Datos As Variant***' recorrer las columnas del registro y copiar celda a celda***For i = 0 To Num_Columnas - 1***' Si la celda Mayúsculas está activada y el tipo de datos es String***If** Mayusculas.Value = True **And** TypeName(r1.Offset(0, i).Value) =**"String" Then**

Datos = UCase(r1.Offset(0, i).Value)

Else

Datos = r1.Offset(0, i).Value

End If

r2.Offset(0, i).Value = Datos

Next i

End Sub**BOTONES DE OPCIÓN (Option Button)**

Los botones de opción se utilizan para elegir una única opción entre una serie de ellas, es decir, de un grupo de opciones sólo permitirán que se escoja una. De la misma forma que las celdas de verificación casi siempre implican una estructura condicional para comprobar cuál de ellas está activada.

El botón activado tendrá su propiedad **Value** igual a true. Como ejemplo de su utilización crearemos dos botones de opción que sirvan para que, a la hora de copiar datos hacia la hoja, se copie sólo los valores de Nombre y Apellidos o todos como hasta ahora. Incluya dos botones de opción y establezca las siguientes propiedades.

Botón1.

Name, Todo.

Caption, Todo.

Botón2.

Name, Solo_Nombre.

Caption, Nombre y Apellidos.

Si está activado el primer botón deberán copiarse todos los datos mientras que si está activado el segundo solo se copiarán el Nombre y los Apellidos. El procedimiento **Copiar_Datos_Hojas** quedará de la forma siguiente.

Private Sub Copiar_Datos_Hojas(r1 As Range, r2 As Range)

Dim i **As Integer**

Dim Datos **As Variant**

Dim Final **As Integer**

' Si Botón Todo Activado, se copian todas las columnas

If Todo.Value = True **Then**

Final = Num_Columnas - 1

Else *' Sólo se copian las dos primera columnas*

Final = 1

End If

' recorrer las columnas del registro y copiar celda a celda

For i = 0 **To** Final

' Si la celda Mayúsculas está activada y el tipo de datos es String

If Mayusculas.Value = True **And** TypeName(r1.Offset(0, i).Value) =

"String" **Then**

Datos = UCase(r1.Offset(0, i).Value)

Else

Datos = r1.Offset(0, i).Value

End If

r2.Offset(0, i).Value = Datos

Next i

End Sub

INFORMACIÓN (INCLUÍDA EN ESTE DOCUMENTO EDUCATIVO) TOMADA DE:**Sitios web:**

1. <http://www3.uji.es/~berbel/Visual%20Basic/Manuales/Excelvbaplication%202010.pdf>