CBS
Colegio Bautista Shalom

Computación
Quinto PCOC PFS
Tercer Bimestre

Contenidos

CONVERSIÓN DE TIPOS DE DATOS

- ✓ FUNCIONES DE CONVERSIÓN DE TIPOS.
- ✓ OBJETO CELLS(FILA, COLUMNA).
- ✓ UTILIZAR CELLS PARA REFERENCIAR UN RANGO.
- ✓ VARIABLES DE OBJETOS.
- ✓ OPTION EXPLICIT.

ESTRUCTURA IF...ELSE

- ✓ ESTRUCTURAS IF ANIDADAS.

OPERADORES LÓGICOS

- ✓ OPERADOR LÓGICO AND (Y).
- ✓ OPERADOR LÓGICO OR (O).
- ✓ OPERADOR LÓGICO NOT (NO).
- ✓ TABLAS DE LA VERDAD.

ESTRUCTURA SELECT CASE

LA FUNCIÓN MSGBOX

- ✓ SINTÁXIS DE MSGBOX.

LA INSTRUCCIÓN WITH

ESTRUCTURAS REPETITIVAS

- ✓ ESTRUCTURA REPETITIVA PARA (FOR).
- ✓ RECORRER CELDAS DE UNA HOJA DE CÁLCULO.
- ✓ PROPIEDAD CELLS.
- ✓ PROPIEDADES ROW Y COLUMN.
- ✓ ESTRUCTURA REPETITIVA DO WHILE...LOOP (HACER MIENTRAS).
- ✓ ESTRUCTURA DO...LOOP WHILE.
- ✓ ESTRUCTURA DO...LOOP UNTIL (HACER... HASTA QUE SE CUMPLA LA CONDICIÓN).

ESTRUCTURA FOR EACH

NOTA: conforme avances en tu aprendizaje tu catedrático(a) te indicará la actividad o ejercicio a realizar. Sigue sus instrucciones.

CONVERSIÓN DE TIPOS DE DATOS

FUNCIONES DE CONVERSIÓN DE TIPOS

Val(Cadena). Convierte la cadena a un valor numérico.
Str(Número). Convierte el número a una expresión cadena.

Las siguientes funciones tienen la forma **Función (Expresión)**

Función	*Tipo devuelto*	*Intervalo del argumento expresión*
Cbool.	**Boolean**	Cualquier expresión de cadena o numérica válida
Cbyte	**Byte**	0 a 255.
Ccur	**Currency**	-922.337.203.685.477,5808 a 922.337.203.685.477,5807
Cdate	**Date**	Cualquier expresión de fecha.
CDbl	**Double**	-4,94065645841247E-324 para valores negativos; 4,94065645841247E-324 a 1,79769313486232E308 para valores positivos.
Cdec	**Decimal**.	+/-7,9228162514264337593543950335. La menor posición para un número que no sea cero es 0,0000000000000000000000000001
CInt	**Integer**	-32.768 a 32.767; las fracciones se redondean.
CLng	**Long**	-2.147.483.648 a 2.147.483.647; las fracciones se redondean.
CSng	**Single**	-3,402823E38 a -1,401298E-45 para valores negativos; 1,401298E-45 a 3,402823E38 para valores positivos.
CVar	**Variant**	El mismo intervalo que Double para valores numéricos. El mismo intervalo que String para valores no numéricos.
CStr	**String**	El valor de retorno de CStr depende del argumento expresión.

OBJETO CELLS(FILA, COLUMNA)

Sirve, como el objeto range, para referenciar una celda o rango de celdas, pero en lugar de utilizar la referencia de la forma A1, B1, X320, ... utiliza la fila y la columna que ocupa la celda dentro de la hoja (objeto WorkSheet).

Por ejemplo, para poner hola en la celda A1 de la hoja activa seria:

```
ActiveSheet.Cells(1,1).Value="Hola"
```

UTILIZAR CELLS PARA REFERENCIAR UN RANGO

Esto sería el equivalente a Range("Celda_Inicial:Celda_Final"). La forma que se obtiene utilizando Cells es un poco más larga, pero se verá que a veces resulta mucho más funcional que utilizando únicamente range. Para referirnos al rango A1:B8, pondremos,

```
Range(Cells(1, 1), Cells(8, 2)).Value = "Hola"
```

Otra forma interesante de Cells es la siguiente,

```
Range("A5:B10").Cells(2, 1).Value = "Hola"
```

Pondrá en la celda A6 el valor "Hola", observe que en este ejemplo Cells comienza a contar filas y columnas a partir del rango especificado en el objeto Range.

VARIABLES DE OBJETOS

Una variable objeto sirve para hacer referencia a un objeto, esto significa que podremos acceder a las propiedades de un objeto e invocar a sus métodos a través de la variable en lugar de hacerlo directamente a través del objeto. Posiblemente no se utilice demasiado esta clase de, pero hay casos en los que no hay más remedio que utilizarlas, por ejemplo, en estructuras For Each ... Next como veremos, o cuando sea necesario construir funciones que devuelvan rangos, referencias a hojas, etc.

Para declarar una variable objeto se utiliza también la palabra Dim de la forma siguiente,

```
Dim Var_Objeto As Objeto
```

Por Ejemplo:

```
Dim R As Range

Dim Hoja As WorkSheet
```

Para asignar un objeto a una variable debe utilizar la instrucción **Set.**

```
Set Variable_Objeto = Objeto
```

Por Ejemplo:

```
Set R= ActiveSheet.Range("A1:B10")
Set Hoja = ActiveSheet
Set Hoja = WorkSheets(1)
```

Ejemplo:

Llenar el rango de A1 a B10 con la palabra "Hola" y después poner negrita, observe como se asigna una variable objeto al objeto y luego como se trabaja con esa variable de la misma forma que trabajaría directamente sobre el objeto

```
Sub obj()
    Dim R As Range
    Set R = ActiveSheet.Range("A10:B15")
    R.Value = "Hola"
    R.Font.Bold = True
End Sub
```

Estructuras condicionales.

Las estructuras condicionales son instrucciones de programación que permiten controlar la ejecución de un fragmento de código en función de si se cumple o no una condición.

Estudiaremos en primer lugar la instrucción **if** Condición **then..End if** (**Si** Condición **Entonces...Fin Si**).

La estructura condicional que se construye con la instrucción **Si** Condición **Entonces... Fin Si** tiene la forma siguiente.

```
Si Condición Entonces
    Senténcia1
    Senténcia2
    .
    .
    SenténciaN
Fin Si
```

Cuando el programa llega a la instrucción Si Condición Entonces, se evalúa la condición, si esta se cumple (es cierta), se ejecutan todas las sentencias que están encerradas en el bloque, si no se cumple la condición, se saltan estas sentencias. Esta estructura en Visual Basic tiene la sintaxis siguiente,

```
If Condición Then
    Senténcia1
    Senténcia2
    .
    .
    SenténciaN
End If
```

Ejemplo:

Ingresar una cantidad que representa el precio de algo utilizando el teclado con la instrucción ***InputBox*** y guardarlo en la celda A1 de la hoja activa. Si el valor entrado desde el teclado (y guardado en A1) es superior a 1000, pedir descuento con otro **InputBox** y guardarlo en la celda A2 de la hoja activa. Calcular en A3, el precio de A1 menos el descuento de A2.

```
Sub Condicional()

    ActiveSheet.Range("A1").Value = 0 ' Poner las celdas donde se guardan los
    valores 0.

    ActiveSheet.Range("A2").Value = 0

    ActiveSheet.Range("A3").Value = 0

    ActiveSheet.Range("A1").Value = Val(InputBox("Entrar el precio", "Entrar"))

    ' Si el valor de la celda A1 es mayor que 1000, entonces, pedir descuento

    If ActiveSheet.Range("A1").Value > 1000 Then

    ActiveSheet.Range("A2").Value = Val(InputBox("Entrar Descuento", "Entrar"))

    End If

    ActiveSheet.Range("A3").Value = ActiveSheet.Range("A1").Value - _

    ActiveSheet.Range("A2").Value

End Sub
```

Ejemplo:

El mismo que el anterior, pero utilizando variables.

OPTION EXPLICIT

```
Sub Condicional()


    Dim Precio As Integer

    Dim Descuento As Integer
```

```
    Precio = 0

    Descuento = 0

    Precio = Val(InputBox("Entrar el precio", "Entrar"))

    ' Si el valor de la variable precio es mayor que 1000, entonces, pedir descuento

    If Precio > 1000 Then

    Descuento = Val(InputBox("Entrar Descuento", "Entrar"))

    End If

    ActiveSheet.Range("A1").Value = Precio

    ActiveSheet.Range("A2").Value = Descuento

    ActiveSheet.Range("A3").Value = Precio – Descuento


End Sub
```

Las variables, aunque muchas veces "innecesarias", quizás dejan los programas más legibles y claros. Y la legibilidad de un programa es lo más valioso del mundo para un programador, sobre todo si se da el caso (inevitable el 99,999...% de las ocasiones) que se tenga que modificar un programa para dotarle de más funcionalidades, facilitar su manejo, etc.

En la mayoría de ejemplos, que encontrará en este manual verá que se utilizan variables preferentemente. Aunque muchas veces su función sea simplemente recoger datos de las celdas para operarlas y dejarlas en otras celdas y, consecuentemente, aumente el número de operaciones, creemos que con ello se gana en legibilidad y flexibilidad.

Ejemplo:

Macro que compara los valores de las celdas A1 y A2 de la hoja activa. Si son iguales pone el color de la fuente de ambas en azul.

```
Sub Condicional2()

    If ActiveSheet.Range("A1").Value = ActiveSheet.Range("A2").Value Then

    ActiveSheet.Range("A1").Font.Color = RGB(0, 0, 255)

    ActiveSheet.Range("A2").Font.Color = RGB(0, 0, 255)

    End If

End Sub
```

ESTRUCTURA IF...ELSE

Esta estructura se utiliza cuando se requiere una respuesta alternativa a una condición.

Su estructura es la siguiente… (en la siguiente página)

Si Condición **Entonces**

Senténcia1

Senténcia2

.

.

SenténciaN

Sino

Senténcia1

Senténcia2

.

.

SenténciaN

Fin Si

Observe que, si se cumple la condición, se ejecuta el bloque de sentencias delimitado por **Si** Condición **Entonces** y Si no se cumple la condición se ejecuta el bloque delimitado por **Sino y Fin Si.** En Visual Basic la instrucción **Si** Condición **Entonces ... Sino ... Fin Si** se expresa con las instrucciones siguientes.

If Condición **Then**

Senténcia1

Senténcia2

.

.

SenténciaN

Else

Senténcia1

Senténcia2

.

.

SenténciaN

End If

Ejemplo:

Ingresar por teclado una cantidad que representa el precio de algo con la instrucción **InputBox** y guardarlo en la celda A1 de la hoja activa. Si el valor entrado desde el teclado (y guardado en A1) es superior a 1000, se aplica un descuento del 10% sino se aplica un descuento del 5%, el descuento se guarda en la celda A2 de la hoja activa. Colocar en A3, el total descuento y en A4 el total menos el descuento.

```
Sub Condicional_Else()

    Dim Precio As Single

    Dim Descuento As Single

    Precio = 0

    Precio = Val(InputBox("Entrar el precio", "Entrar"))

    ' Si el valor de la variable precio es mayor que 1000, entonces, aplicar
    descuento del 10%

    If Precio > 1000 Then

        Descuento = Precio * (10 / 100)

        ActiveSheet.Range("A2").Value = 0,1

    Else ' Sinọ  Aplicar descuento del 5%

        Descuento = Precio * (5 / 100)

        ActiveSheet.Range("A2").Value = 0,05
End If

    ActiveSheet.Range("A1").Value = Precio

    ActiveSheet.Range("A3").Value = Descuento

    ActiveSheet.Range("A4").Value = Precio - Descuento

End Sub
```

Ejemplo:

Restar los valores de las: celda A1 y A2. Guardar el resultado en A3. Si el resultado es positivo o 0, poner la fuente de A3 en azul, sino ponerla en rojo.

```
Sub Condicional_Else2()

    ActiveSheet.Range("A3").Value = AvtiveSheet.Range("A1").Value - _

    ActiveSheet.Range("A2").Value

    If ActiveSheet("A3").Value < 0 Then

        ActiveSheet.Range("A3").Font.Color = RGB(255,0,0)
```

```
    Else
        ActiveSheet.Range("A3").Font.Color = RGB(0,0,255)
    End If
End Sub
```

ESTRUCTURAS IF ANIDADAS

No tiene que sorprenderle, dentro de una estructura if puede ir otra, y dentro de esta otra, y otra... Vea el ejemplo siguiente.

Ejemplo:

Comparar los valores de las celdas A1 y A2 de la hoja activa. Si son iguales, escribir en A3 *"Los valores de A1 y A2 son iguales"*, si el valor de A1 es mayor que A2, escribir *"A1 mayor que A2"*, si no escribir *"A2 mayor que A1"*.

```
Sub Condicional()
    If ActiveSheet.Range("A1").Value = ActiveSheet.Range("A2").Value Then
        ActiveSheet.Range("A3").Value = "Los Valores de A1 y A2 son iguales"
    Else
        If ActiveSheet.Range("A1").Value > ActiveSheet.Range("A2").Value Then
            ActiveSheet.Range("A3").Value = "A1 mayor que A2"
        Else
        ActiveSheet.Range("A3").Value = "A2 mayor que A1"
        End If
    End If
End Sub
```

Observe que la segunda estructura **If...Else...End If** queda dentro del **Else** de la primera estructura. Esta es una regla general, cuando pone un **End If**, este cierra siempre el último **If** (o **Else**) abierto.

OPERADORES LÓGICOS

Estos operadores se utilizan cuando se necesitan evaluar dos o más condiciones para decidir si se ejecutan o no determinadas acciones.

OPERADOR LÓGICO AND (Y)

Utilizaremos este operador cuando sea preciso que para ejecutar un bloque de instrucciones se cumpla más de una condición. Observe que deberán cumplirse todas las condiciones.

Ejemplo:

Ingresar el Nombre, la cantidad y el precio de un producto desde el teclado y guardarlos respectivamente en A1, A2 y A3. Calcular el total y guardarlo en A4. Si el total es superior a 10.000 **y** el nombre del producto es "Patatas",

pedir un descuento, calcularlo el total descuento y guardarlo en A5, luego restar el descuento del total y guardarlo en A6.

```
Sub Ejemplo_12()

    Dim Producto As String

    Dim Cantidad As Integer

    Dim Precio As Single

    Dim Total As Single

    Dim Descuento As Single

    Dim Total_Descuento As Single

    Precio = 0

     Producto = InputBox("Entrar Nombre del Producto","Entrar")

     Precio = Val(InputBox("Entrar el precio", "Entrar"))

     Precio = Val(InputBox("Entrar la cantidad", "Entrar"))

     Total = Precio * Cantidad

     ActiveSheet.Range("A1").Value = Producto

     ActiveSheet.Range("A2").Value = Precio

     ActiveSheet.Range("A3").Value = Cantidad

     ActiveSheet.Range("A4").Value = Total

     ' Si total mayor que 10.000 y el producto es Patatas, aplicar descuento.

     If Total > 10000 And Producto = "Patatas" Then

          Descuento = Val(InputBox("Entrar Descuento", "Entrar"))

          Total_Descuento = Total * (Descuento / 100)

          Total = Total - Total_Descuento
        ActiveSheet.Range("A5").Value = Total_Descuento

        ActiveSheet.Range("A6").Value = Total

    End If

End Sub
```

Observe que para que se ejecute el bloque de instrucciones entre If.. End If deben cumplirse las dos condiciones que se evalúan, si falla cualquiera de las dos (o las dos a la vez), no se ejecuta dicho bloque.

OPERADOR LÓGICO OR (O)

Utilizaremos este operador cuando sea preciso que para ejecutar un bloque de instrucciones se cumpla alguna de una serie de condiciones. Observe que sólo es necesario que se cumpla alguna de las condiciones que se evalúan.

Ejemplo:

Ingresar el Nombre, la cantidad y e l precio de un producto desde el teclado y guardarlos respectivamente en A1, A2 y A3. Calcular el total y guardarlo en A4. Si el total es superior a 10.000 **o** el nombre del producto el "Patatas", pedir un descuento, calcularlo el total descuento y guardarlo en A5, luego restar el descuento del total y guardarlo en A6.

```
Sub Ejemplo_13()

    Dim Producto As String

    Dim Cantidad As Integer

    Dim Precio As Single

    Dim Total As Single

    Dim Descuento As Single

    Dim Total_Descuento As Single

    Precio = 0

    Producto = InputBox("Entrar Nombre del Producto","Entrar")

    Precio = Val(InputBox("Entrar el precio", "Entrar"))

    Precio = Val(InputBox("Entrar la cantidad", "Entrar"))

    Total = Precio * Cantidad

    ActiveSheet.Range("A1").Value = Producto

    ActiveSheet.Range("A2").Value = Precio

    ActiveSheet.Range("A3").Value = Cantidad

    ActiveSheet.Range("A4").Value = Total

    ' Si total mayor que 10.000 o el producto es Patatas, aplicar descuento.

    If Total > 10000 Or Producto = "Patatas" Then

        Descuento = Val(InputBox("Entrar Descuento", "Ent rar"))

        Total_Descuento = Total * (Descuento / 100)

        Total = Total - Total_Descuento

        ActiveSheet.Range("A5").Value = Total_Descuento
```

```
        ActiveSheet.Range("A6").Value = Total

    End If

End Sub
```

Observe que para que se ejecute el bloque de instrucciones entre If.. End If sólo es necesario que se cumpla alguna de las dos condiciones que se evalúan (o las dos a la vez). Sólo cuando no se cumple ninguna de las dos no se ejecutan las instrucciones del bloque.

OPERADOR LÓGICO NOT (NO)

Este operador se utiliza para ver si NO se cumple una condición. El siguiente ejemplo hace lo mismo que el ejemplo 7 pero utilizando el operador Not.

Ejemplo:

Ingresar una cantidad que representa el precio de algo por el teclado con la instrucción InputBox y guardarlo en la celda A1 de la hoja activa. Si el valor entrado desde el teclado (y guardado en A1) es superior a 1000, pedir descuento con otro InputBox y guardarlo en la celda A2 de la hoja activa. Calcular en A3, el precio de A1 menos el descuento de A2.

```
Sub Ejemplo_14()

    Dim Precio As Integer

    Dim Descuento As Integer

    Precio = 0

    Descuento = 0

    Precio = Val(InputBox("Entrar el precio", "Entrar"))

    ' Si el valor de la variable precio NO es menor igual 1000, entonces, pedir
    descuento

    If Not (Precio <= 1000) Then

        Descuento = Val(InputBox("Entrar Descuento", "Entrar"))

    End If

    ActiveSheet.Range("A1").Value = Precio

    ActiveSheet.Range("A2").Value = Descuento

    ActiveSheet.Range("A3").Value = Precio - Descuento

End Sub
```

TABLAS DE LA VERDAD

Vea las tablas siguientes para ver los resultados de evaluar dos condiciones con el operador And y con el operador Or.

And.

Condición1	Condición2	Resultado
Falsa	Falsa	Falso
Falsa	Cierta	Falso
Cierta	Falsa	Falso
Cierta	Cierta	Cierto

Or.

Condición1	Condición2	Resultado
Falsa	Falsa	Falso
Falsa	Cierta	Cierto
Cierta	Falsa	Cierto
Cierta	Cierta	Cierto

Observe que con el operador AND deben de cumplirse todas las condiciones (dos o veinticinco) para que el resultado sea cierto. Con el operador OR sólo es necesario que se cumpla una (de las dos o de las veinticinco) para que el resultado sea cierto.

ESTRUCTURA SELECT CASE

En ocasiones se dará el caso que en función del valor o rango de valores que pueda tener **una** variable, **una** celda, **una** expresión, etc. deberán llevarse a cabo diferentes acciones o grupos de acciones.

Ejemplo

Macro que suma, resta, multiplica o divide los valores de las celdas A1 y A2 dependiendo de si B1 contiene el signo +, -, x, :. El resultado lo deja en A3. Si en B1 no hay ninguno de los signos anteriores en A3 debe dejarse un 0.

```
Sub Ejemplo_15()

    Dim Signo As String
    Dim Valor1 As Integer, Valor2 As Integer, Total As Integer

    Valor1 = ActiveSheet.Range("A1").Value

    Valor2 = ActiveSheet.Range("A2").Value

    Signo = ActiveSheet.Range("B1").Value

    Total=0

    If  Signo = "+" Then

        Total = Valor1 + Valor2

    End if

     If  Signo = "-" Then

        Total = Valor1 - Valor2
```

```
        End if


        If  Signo = "x" Then

            Total = Valor1 * Valor2
        End if
        If Signo = ":" Then

            Total = Valor1 / Valor2
        End if
        ActiveCell.Range("A3").Value = Total
    End Sub
```

Observe que en el ejemplo anterior todas las instrucciones if evalúan la misma variable.

El programa funciona correctamente, pero para estos casos es mejor utilizar la instrucción Select Case, el motivo principal es por legibilidad y elegancia.

Select Case tiene la sintaxis siguiente,

```
Select Case Expresión
    Case valores :
    Instrucciones.
    Case valores :
    Instrucciones.
    .
    .
    Case valores:
    Instrucciones.
    Case Else
    Instrucciones en caso que no sean ninguno de los valores anteriores.
End Select
```

Vea el ejemplo anterior solucionado con esta estructura.

Ejemplo:

```
Sub Ejemplo_16()

    Dim Signo As String

    Dim Valor1 As Integer, Valor2 As Integer, Total As Integer

    Valor1 = ActiveSheet.Range("A1").Value

    Valor2 = ActiveSheet.Range("A2").Value

    Signo = ActiveSheet.Range("A3").Value

    Select Case signo

        Case "+"

        Total = Valor1 + Valor2

        Case "-"

        Total = Valor1 - Valor2

        Case "x"

        Total = Valor1 * Valor2

       Case Else

       Total = 0

    End Select

    ActiveCell.Range("A3").Value = Total

End Sub
```

Vea el ejemplo siguiente donde cada sentencia Case evalúa un rango de valores. Ejemplo:

Programa que pide tres notas de un alumno mediante la función InputBox. Las notas van a parar respectivamente a las celdas A1, A2 y A3 de la hoja activa. El programa calcula la media y la deja en A4. Si la media está entre 0 y 2 deja en A5 el mensaje "Muy deficiente", si la nota es 3 deja en A5 el mensaje "Deficiente", si la nota es 4 deja "Insuficiente", si es 5 "Suficiente", si es 6 "Bien", si está entre 7 y 8 deja "Notable", si es mayor que 8 deja "Sobresaliente".

```
    Sub Ejemplo_17()

        Dim Nota1 As Integer, Nota2 As Integer, Nota3 As Integer
```

```
Dim Media As Single

Nota1 = Val(InputBox("Entrar Nota primera evaluación", "Nota"))

Nota2 = Val(InputBox("Entrar Nota Segunda evaluación", "Nota"))

Nota3 = Val(InputBox("Entrar Nota Tercera evaluación", "Nota"))

Media = (Nota1 + Nota2 + Nota3) / 3

 ActiveSheet.Range("A1").Value = Nota1

 ActiveSheet.Range("A2").Value = Nota2

 ActiveSheet.Range("A3").Value = Nota3

 ActiveSheet.Range("A4").Value = Media

        Case 0 To 2

        ActiveSheet.Range("A5").Value = "Muy deficiente"

        Case 3

        ActiveSheet.Range("A5").Value = "Deficiente"

        Case 4

        ActiveSheet.Range("A5").Value = "Insuficiente"

        Case 5

        ActiveSheet.Range("A5").Value = "Suficiente"

        Case 6

        ActiveSheet.Range("A5").Value = "Bien"

        Case 7 To 8

        ActiveSheet.Range("A5").Value = "Notable"

        Case >8

        ActiveSheet.Range("A5").Value = "Sobresaliente"

    End Select

End Sub
```

LA FUNCIÓN MSGBOX

Esta función muestra un mensaje en un cuadro de diálogo hasta que el usuario pulse un botón. La función devuelve un dato tipo Integer en función del botón pulsado por el usuario. A la hora de invocar está función, s e permiten diferentes tipos de botones.

SINTÁXIS DE MSGBOX

MsgBox(Mensaje, Botones, Título, Archivo de ayuda, contexto).

Mensaje: Obligatorio, es el mensaje que se muestra dentro del cuadro de diálogo.

Botones: Opcional. Es un número o una suma de números o constantes (vea tabla Valores para botones e Iconos), que sirve para mostrar determinados botones e iconos dentro del cuadro de diálogo. Si se omite este argumento asume valor 0 que corresponde a un único Botón OK.

Título: Opcional. Es el texto que se mostrará en la barra del título del cuadro de diálogo.
Archivo de Ayuda: Opcional. Si ha asignado un texto de ayuda al cuadro de diálogo, aquí debe especificar el nombre del archivo de ayuda donde está el texto.

Context: Opcional. Es el número que sirve para identificar el texto al tema de ayuda correspondiente que estará contenido en el archivo especificado en el parámetro Archivo de Ayuda.

Tabla para botones e iconos del cuadro MsgBox

Constante	Valor	Descripción
VbOKOnly	0	Muestra solamente el botón Aceptar.
VbOKCancel	1	Muestra los botones Aceptar y Cancelar.
VbAbortRetryIgnore	2	Muestra los botones Anular, Reintentar e Ignorar
VbYesNoCancel	3	Muestra los botones Sí, No y Cancelar.
VbYesNo	4	Muestra los botones Sí y No.
VbRetryCancel	5	Muestra los botones Reintentar y Cancelar.
VbCritical	16	Muestra el icono de mensaje crítico.
VbQuestion	32	Muestra el icono de pregunta de advertencia.
VbExclamation	48	Muestra el icono de mensaje de advertencia.
VbInformation	64.	Muestra el icono de mensaje de información
VbDefaultButton1	0	El primer botón es el predeterminado
VbDefaultButton2	256	El segundo botón es el predeterminado.
VbDefaultButton3	512	El tercer botón es el predeterminado.
VbDefaultButton4.	768	El cuarto botón es el predeterminado

VbApplicationModal	0	Aplicación modal; el usuario debe responder al cuadro de mensajes antes de poder seguir trabajando en la aplicación actual.
VbSystemModal	4096	Sistema modal; se suspenden todas las aplicaciones hasta que el usuario responda al cuadro de mensajes.

El primer grupo de valores (0 a 5) describe el número y el tipo de los botones mostrados en el cuadro de diálogo; el segundo grupo (16, 32, 48, 64) describe el estilo del icono, el tercer grupo (0, 256, 512) determina el botón predeterminado y el cuarto grupo (0, 4096) determina la modalidad del cuadro de mensajes. Cuando se suman números para obtener el valor final del argumento buttons, se utiliza solamente un número de cada grupo.

Los valores que puede devolver la función msgbox en función del botón que pulse el usuario se muestran en la tabla siguiente.

Tabla de valores que puede devolver MsgBox

Constante	Valor	Descripción
VbOK	1	Aceptar
VbCancel	2	Cancelar
VbAbort	3	Anular
VbRetry	4	Reintentar
VbIgnore	5	Ignorar
VbYes	6	Sí
VbNo	7	No

Ejemplos de MsgBox:

```
Sub Tal()

.

.

' El cuadro Muestra los botones Si y No y un icono en forma de interrogante.
Cuando se pulsa

' un botón, el valor lo recoge la variable X. En este caso los valores devueltos
pueden ser 6 o 7
```

```
' que corresponden respectivamente a las constantes VbYes y VbNo, observe la
instrucción If de
'después.
X = MsgBox("Desea Continuar", vbYesNo + vbQuestion, "Opción",,)
' Se ha pulsado sobre botón Si
If X = vbYes Then
        .....
Else ' Se ha pulsado sobre botón No
        .....
End If
.
.
End Sub
```

Algunas veces puede que le interese simplemente desplegar un cuadro MsgBox para mostrar un mensaje al usuario sin que se requiera recoger ningún valor. En este caso puede optar por la forma siguiente,

```
MsgBox Prompt:="Hola usuaria, Ha acabado el proceso", Buttons:=VbOkOnLy _
Title:="Mensaje"
```

Lo que no puede hacer porque Visual Basic daría error es poner la primera forma sin igualarla a ninguna variable. Por ejemplo, la expresión siguiente es incorrecta,

```
MsgBox ("Hola usuario, Ha acabado el proceso", VbOkOnly, "Mensaje")
```

Sería correcto poner:

```
X= MsgBox ("Hola usuario, Ha acabado el proceso", VbOkOnly, "Mensaje")
```

En este caso, aunque X reciba un valor, luego no se utiliza para nada, es decir simplemente se pone para que Visual Basic dé error.

LA INSTRUCCIÓN WITH

Suponemos que llegado a este punto le parecerá engorroso tener que referirse a los objetos siguiendo toda o casi toda la jerarquía. Ya hemos indicado que es mejor hacerlo de esta manera porque el programa gana en claridad y elegancia y, consecuentemente, el programador gana tiempo a la hora de hacer modificaciones o actualizaciones. La sentencia With le ayudará a tener que escribir menos código sin que por esto el programa pierda en claridad. Concretamente esta función sirve para ejecutar una serie de acciones sobre un mismo Objeto.

Su sintaxis es la siguiente.

```
With Objeto
    Instrucciones
End With
```

Ingresar el Nombre, la cantidad y el precio de un producto desde el teclado y guardarlos respectivamente en A1, A2 y A3. Calcular el total y guardarlo en A4. Si el total es superior a 10.000 o el nombre del producto el "Patatas", pedir un descuento, calcularlo el total descuento y guardarlo en A5, luego restar el descuento del total y guardarlo en A6.

```
Sub Ejemplo_19()
    Dim Producto As String
    Dim Cantidad As Integer
    Dim Precio As Single
    Dim Descuento As Single
    Dim Total_Descuento As Single
    Precio = 0
    Producto = InputBox("Entrar Nombre del Producto","Entrar")
    Precio = Val(InputBox("Entrar el precio", "Entrar"))
    Precio = Val(InputBox("Entrar la cantidad", "Entrar"))
    Total = Precio * Cantidad
    With ActiveSheet
        .Range("A1").Value = Producto
        .Range("A2").Value = Precio
        .Range("A3").Value = Cantidad
        .Range("A4").Value = Total
    End With
    ' Si total mayor que 10.000 o el producto es Patatas, aplicar descuento.
    If Total > 10000 Or Producto = "Patatas" Then
        Descuento = Val(InputBox("Entrar Descuento", "Entrar"))
Total_Descuento = Total * (Descuento / 100)
```

```
        Total = Total - Total_Descuento

            End With

        End If

End Sub
```

ESTRUCTURAS REPETITIVAS

Este tipo de estructuras permiten ejecutar más de una vez un mismo bloque de sentencias.

Ejemplo:

Supongamos que tenemos que hacer un programa para entrar las notas de una clase de 5 alumnos que se guardaran respectivamente en las celdas de A1 a A5 de la hoja activa. Después hacer la media que se guardará en A6. Con las estructuras vistas hasta ahora, podríamos hacer:

```
Sub Ejemplo_20 ()

    Dim Nota As Integer

    Dim Media As Single

    Media = 0

    Nota = Val(InputBox("Entrar la 1 Nota : ","Entrar Nota"))

    ActiveSheet.Range("A1").Value = Nota

    Media = Media + Nota

    Nota = Val(InputBox("Entrar la 1 Nota : ","Entrar Nota"))

    ActiveSheet.Range("A2").Value = Nota
    Media = Media + Nota

    Nota = Val(InputBox("Entrar la 1 Nota : ","Entrar Nota"))

    ActiveSheet.Range("A3").Value = Nota

    Media = Media + Nota

    Nota = Val(InputBox("Entrar la 1 Nota : ","Entrar Nota"))

    ActiveSheet.Range("A4").Value = Nota

    Media = Media + Nota

    Nota = Val(InputBox("Entrar la 1 Nota : ","Entrar Nota"))

    ActiveSheet.Range("A5").Value = Nota
```

```
        Media = Media + Nota

        Media = Media / 5

        ActiveSheet.Range("A6").Value = Media

    End Sub
```

Observe que este programa repite el siguiente bloque de sentencias, 5 veces.

Nota = Val(InputBox("Entrar la 1 Nota : ","Entrar Nota"))

ActiveSheet.Range("A5"). Value = Nota

Media = Media + Nota Para evitar esta tipo de repeticiones de código, los lenguajes de programación incorporan instrucciones que permiten la repetición de bloques de código.

ESTRUCTURA REPETITIVA PARA (FOR)

Esta estructura sirve para repetir la ejecución de una sentencia o bloque de sentencias, un número definido de veces. La estructura es la siguiente:

```
Para var =Valor_Inicial Hasta Valor_Final Paso Incremento Hacer

Inicio

    Sentencia 1

    Sentencia 2

    .

    .

    Sentencia N

Fin
```

Var es una variable que la primera vez que se entra en el bucle se iguala a *Valor_Inicial*, las sentencias del bucle se ejecutan hasta que **Var** llega al *Valor_Final*, cada vez que se ejecutan el bloque de instrucciones **Var** se incrementa según el valor de Incremento.

En Visual Basic para Excel la estructura Para se implementa con la instrucción **For ... Next**.

```
For Varible = Valor_Inicial To Valor_Final Step Incremento

    Sentencia 1

    Sentencia 2

    .

    .

    Sentencia N
```

```
Next Variable
```

- *Si el incremento es 1, no hace falta poner Step 1.*

Ejemplo:

Ingresar 10 valores utilizando la función InputBox, sumarlos y guardar el resultado en la celda A1 de la hoja activa.

```
Sub Ejemplo_21()

    Dim i As Integer

    Dim Total As Integer

    Dim Valor As Integer

    For i=1 To 10

        Valor= Val(InputBox("Entrar un valor","Entrada"))

        Total = Total + Valor

    Next i

    ActiveCell.Range("A1").Value = Total

End Sub
```

RECORRER CELDAS DE UNA HOJA DE CÁLCULO

Una operación bastante habitual cuando se trabaja con Excel es el recorrido de rangos de celdas para llenarlas con valores, mirar su contenido, etc.

Las estructuras repetitivas son imprescindibles para recorrer grupos de celdas o rangos. Vea los siguientes ejemplos para ver ejemplos de utilización de estructuras repetitivas para recorrer rangos de celdas, observe la utilización de las propiedades Cells y Offset.

PROPIEDAD CELLS

Sirve para referenciar una celda o un rango de celdas según coordenadas de fila y columna.

Ejemplo:

Llenar el rango de las celdas A1..A5 con valores pares consecutivos empezando por el 2.

```
Sub Ejemplo_22()

    Dim Fila As Integer

    Dim i As Integer

    Fila = 1
```

```
    For i=2 To 10 Step 2
        ActiveSheet.Cells(Fila,1).Value = i
        Fila = Fila+1
    Next i
End Sub
```

Ejemplo:

Llenar un rango de filas, empezando por una celda, que se debe especificar desde teclado, con una serie de 10 valores correlativos (comenzando por el 1).

```
Sub Ejemplo_23()
    Dim Celda_Inicial As String
    Dim i As Integer
    Dim Fila As Integer, Columna As Integer
    Celda_Inicial = InputBox("Introducir la celda Inicial : ", "Celda Inicial")
    ActiveSheet.Range(Celda_Inicial).Activate
    ' Coger el valor de fila de la celda activa sobre la variable Fila
    Fila = ActiveCell.Row
    ' Coger el valor de columna de la celda activa sobre la variable Fila
    Columna = ActiveCell.Column
    For i = 1 To 10
        ActiveSheet.Cells(Fila, Columna).Value = i
        Fila = Fila + 1
    Next i
```

PROPIEDADES ROW Y COLUMN

Como habrá deducido del ejemplo anterior devuelven la fila y la columna de un objeto range. En el ejemplo anterior se utilizaban concretamente para obtener la fila y la columna de la celda activa.
Otra forma de solucionar el ejemplo 23 seria.

```
Sub Ejemplo_23()
    Dim Celda_Inicial As String
    Dim i As Integer
```

```
    Dim Fila As Integer, Columna As Integer

    Celda_Inicial = InputBox("Introducir la celda Inicial : ", "Celda Inicial")

    ActiveSheet.Range(Celda_Inicial).Activate

    Fila = 1

    For i = 1 To 10

        ActiveSheet.Range(Celda_Inicial).Cells(Fila, 1).Value = i

        Fila = Fila + 1

    Next i

End Sub
```

** Recuerde que cuando utilizamos **Cells** como propiedad de un rango (Objeto Range), **Cells** empieza a contar a partir de la celda referenciada por **Range**.

Ejemplo:

El mismo con el que introducíamos el tema (ejemplo 20), pero utilizando el **for** y propiedad **Cells**

```
Sub Ejemplo_24()


    Dim Nota As Integer

    Dim Media As Single

    Dim Fila As Integer

    Media = 0

    For Fila = 1 To 5

        Nota=Val(InputBox("Entrar la " & Fila & " Nota : ", "Entrar Nota"))

        ActiveSheet.Cells(Fila, 1) = Nota

        Media = Media + Nota

    Next Fila

    Media = Media / 5

    ActiveSheet.Cells(6, 1).Value = Media


End Sub
```

ESTRUCTURA REPETITIVA DO WHILE...LOOP (HACER MIENTRAS)

La estructura repetitiva **for** se adapta perfectamente a aquellas situaciones en que se sabe previamente el número de veces que se ha de repetir un proceso, entrar veinte valores, recorrer cincuenta celdas, etc. Pero hay ocasiones o casos en los que no se sabe previamente el número de veces que se debe repetir un proceso. Por ejemplo, suponga que ha de recorrer un rango de filas en los que no se sabe cuántos valores habrá (esto es, cuantas filas llenas habrá), en ocasiones puede que haya veinte, en ocasiones treinta, en ocasiones ninguna, etc. Para estos casos la estructura **for** no es adecuada y deberemos recurrir a la sentencia **Do While..Loop** en alguna de sus formas. Esta estructura repetitiva está controlada por una o varias condiciones, la repetición del bloque de sentencias dependerá de si se va cumpliendo la condición o condiciones.

```
Hacer Mientras (se cumpla la condición)
    Sentencia1
    Sentencia2
    .
    .
    Sentencia N
Fin Hacer Mientras
```

** Los ejemplos que veremos a continuación sobre la instrucción **Do While..Loop** se harán sobre una base de datos. Una base de datos en Excel es simplemente un rango de celdas en que cada fila representa un registro y cada columna un campo de registro, la primera fila es la que da nombre a los campos. Para nuestra base de datos utilizaremos los campos siguientes, *Nombre, Ciudad, Edad, Fecha*. Ponga estos títulos en el rango A1:D1 de la Hoja1 (En A1 ponga Nombre, en B1 ponga Ciudad, en C1 ponga Edad y en D1 Fecha), observe que los datos se empezarán a entrar a partir de A2.

Ejemplo:

Programa para ingresar registros en la base de datos. Cada campo se entra con InputBox. El programa va pidiendo datos mientras se entre un valor en el InputBox correspondiente al nombre, es decir cuando al preguntar el nombre no se entre ningún valor, terminará la ejecución del bloque encerrado entre **Do While...Loop**. Observe la utilización de la propiedad **Offset** para colocar los datos en las celdas correspondientes.

```
Sub Ejemplo_27()
    Dim Nombre As String
    Dim Ciudad As String
    Dim Edad As Integer
    Dim fecha As Date
    'Activar hoja1
    WorkSheets("Hoja1").Activate
    'Activar celda A2
    ActiveSheet.Range("A2").Activate
```

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

' Mientras la variable Nombre sea diferente a cadena vacía

Do While Nombre <> ""

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

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

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

    ' Copiar los datos en las celdas correspondientes

    With ActiveCell

        .Value = Nombre

        .Offset(0,1).Value = Ciudad

        .Offset(0,2).Value = Edad

        .Offset(0,3).Value = fecha

    End With

'Hacer activa la celda de la fila siguiente a la actual

ActiveCell.Offset(1,0).Activate

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

Loop

End Sub
```

Ejemplo:

Preste especial atención a este ejemplo ya que seguro que el código que viene a continuación lo utilizará en muchas ocasiones. Antes que nada, observe el ejemplo anterior, fíjese en que siempre empezamos a llenar el rango de la hoja a partir de la celda A2, esto tiene una nefasta consecuencia, la segunda vez que ejecute la macro machacará los datos de A2:D2 y si continúa ejecutando machacará los datos de los rangos siguientes. Una solución sería observar cual es la celda vacía siguiente y cambiar en la instrucción **ActiveSheet.Range("A2").Activate** , la referencia **A2** por la que corresponde a la primera celda vacía de la columna A. El código que le mostramos a continuación hará esto por nosotros, es decir recorrerá una fila de celdas a partir de A1 hasta encontrar una vacía y dejará a esta como celda activa para que la entrada de datos comience a partir de ella.

```
Sub Ejemplo_28()

    .

    .

    ' Activar hoja1
```

```
    WorkSheets("Hoja1").Activate
    'Activar celda A2
    ActiveSheet.Range("A1").Activate
    'Mientras la celda activa no esté vacía
    Do While Not IsEmpty(ActiveCell)
        'Hacer activa la celda situada una fila por debajo de la actual
        ActiveCell.Offset(1,0).Activate
    Loop
    .
    .
End Sub
```

Ejemplo:

Es la unión de los dos programas anteriores. Es decir habrá un bucle Do While que buscará la primera celda vacía de la base de datos y otro para pedir los valores de los campos hasta que se pulse Enter en Nombre.

```
Sub Ejemplo_28()
    Dim Nombre As String
    Dim Ciudad As String
    Dim Edad As Integer
    Dim fecha As Date
    WorkSheets("Hoja1").Activate
    ActiveSheet.Range("A1").Activate
    'Buscar la primera celda vacía de la columna A y convertirla en activa
    Do While Not IsEmpty(ActiveCell)
        ActiveCell.Offset(1,0).Activate
    Loop
    Nombre = InputBox("Entre el Nombre (Return para Terminar) : ", "Nombre")
    'Mientras la variable Nombre sea diferente a cadena vacía
```

```
Do While 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
    Nombre = InputBox("Entre el Nombre (Return para Terminar) : ",
    "Nombre")
Loop
End Sub
```

Cuando se tienen que ingresar desde el teclado conjuntos de valores, algunos programadores y usuarios prefieren la fórmula de que el programa pregunte si se desean entrar más datos, la típica pregunta ¿Desea Introducir más datos?, si el usuario contesta Sí, el programa vuelve a ejecutar las instrucciones correspondientes a la entrada de datos, si contesta que no se finaliza el proceso, observe como quedaría nuestro bucle de entrada de datos con este sistema. Mas_datos = vbYes

```
Do While Mas_Datos = vbYes
    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
    'Preguntar al usuario si desea entrar otro registro.
     Mas_datos = MsgBox("Otro registro ?", vbYesNo+vbQuestion,"Entrada de
     datos")
Loop
```

** Observe que es necesaria la línea anterior al bucle **Mas_datos = vbYes**, para que cuando se evalúe la condición por vez primera esta se cumpla y se ejecuten las sentencias de dentro del bucle, Mas_datos es una variable de tipo **Integer**. Vea la sección siguiente donde se estudia una variante de la estructura **Do While** que es más adecuada para este tipo de situaciones.

ESTRUCTURA DO...LOOP WHILE

El funcionamiento de esta estructura repetitiva es similar a la anterior salvo que la condición se evalúa al final, la inmediata consecuencia de esto es que las instrucciones del cuerpo del bucle se ejecutaran al menos una vez. Observe que para nuestra estructura de entrada de datos vista en el último apartado de la sección anterior esta estructura es más conveniente, al menos más elegante, si vamos a entrar datos, al menos uno entraremos, por tanto, las instrucciones del cuerpo del bucle se deben ejecutar al menos una vez, luego ya decidiremos si se repiten o no.

```
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
```

Observe que en este caso no es necesario la línea Mas_Datos = vbYes antes de **Do** para forzar la entrada en el bucle ya que la condición va al final.

ESTRUCTURA DO...LOOP UNTIL (HACER... HASTA QUE SE CUMPLA LA CONDICIÓN)

Es otra estructura que evalúa la condición al final observe que la interpretación es distinta ya que el bucle se va repitiendo **HASTA que se cumple la condición**, no MIENTRAS se cumple la condición. Cuál de los dos utilizar, pues, no se sorprenda, la que entienda mejor o le guste más. La entrada de datos con este bucle quedaría.

```
Do

    Nombre = InputBox("Entre el Nombre (Return para Terminar) : ", "Nombre")
    Ciudad = InputBox("Entre la Ciudad : ", "Ciudad")
    Edad = Val(InputBox("Ent re 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")
    'Hasta que Mas_Datos sea igual a vbNo
```

ESTRUCTURA FOR EACH

Este bucle se utiliza básicamente para ejecutar un grupo de sentencias con los elementos de una colección o una matriz (pronto veremos los que es). Recuerde que una colección es un conjunto de objetos, hojas, rangos, etc. Vea el ejemplo siguiente que se utiliza para cambiar los nombres de las hojas de un libro de trabajo.

Ejemplo:

Programa que pregunta el nombre para cada hoja de un libro de trabajo, si no se pone nombre a la hoja, queda el que tiene.

```
Sub Ejemplo_29()
    Dim Nuevo_Nombre As String
    Dim Hoja As WorkSheet
    ' Para cada hoja del conjunto WorkSheets
    For Each Hoja In WorkSheets
        Nuevo_Nombre=InputBox("Nombre de la Hoja : " &
        Hoja.Name,"Nombrar Hojas")
        If Nueva_Nombre <> "" Then
```

```
            Hoja.Name=Nuevo_nombre
        End if
    Next
End Sub
```

** Hoja va referenciando cada una de las hojas del conjunto WorkSheets a cada paso de bucle.

Ejemplo:

Ingresar valores para las celdas del rango A1:B10 de la hoja Activa.

```
Sub Ejemplo_30()
    Dim R As Range
    'Para cada celda del rango A1:B10 de la hoja activa
    For Each R in ActiveSheet.Range("A1:B10")
        R.Value = InputBox("Entrar valor para la celda " & R.Address, "Entrada
        de valores")
    Next
End Sub
```

** Observe que se ha declarado una variable tipo Range, este tipo de datos, como puede imaginar y ha visto en el ejemplo sirve para guardar Rangos de una o más celdas, estas variables pueden luego utilizar todas las propiedades y métodos propios de los Objetos Range. Tenga en cuenta que la asignación de las variables que sirven para guardar o referenciar objetos (Range, WorkSheet, etc.) deben inicializarse muchas veces a través de la instrucción SET.

INFORMACIÓN (INCLUÍDA EN ESTE DOCUMENTO EDUCATIVO) TOMADA DE:

Sitios web:

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