Material Imprimible

Curso de Excel Macros

Módulo 3: Herramientas para la programación I

Programación de una macro

Los cuadros de información

En Visual Basic existen dos tipos de cuadros de información que permiten mostrar o solicitar algún dato del usuario.

El cuadro que permite mostrar información al usuario se llama MESSAGE BOX (el comando en VBA es MSGBOX) y el que permite solicitar información del usuario se llama INPUT BOX (el comando en VBA es INPUTBOX).

Sintaxis:

La sintaxis de un **MSGBOX** o cuadro de **salida de información** es la siguiente:

MSGBOX "Mensaje", Tipo, "Título"

En "Mensaje" se escribirá el mensaje que aparecerá en el cuadro.

En Tipo se indicará el tipo de alerta que aparecerá en el cuadro: vbExclamation, vbInformation, vbCritical, etcétera.

En "Título" se escribirá el título que aparecerá en la parte superior del cuadro.

La sintaxis de un **INPUTBOX** o cuadro de **entrada de información** es la siguiente:

INPUTBOX "Mensaje", "Título"

En "Mensaje" se escribirá el mensaje que aparecerá en el cuadro.

En "Título" se escribirá el título que aparecerá en la parte superior del cuadro.

En ambos casos, si el MSGBOX o el INPUTBOX están escritos en una línea de código autónoma, se lo escribirá SIN PARÉNTESIS. Pero si comparten la línea de código con otra parte de un comendo, se lo escribirá CON PARÉNTESIS.

Ejemplos SIN PARÉNTESIS:

MSGBOX("La operación se ha realizado exitosamente", vbInformation, "Exito")

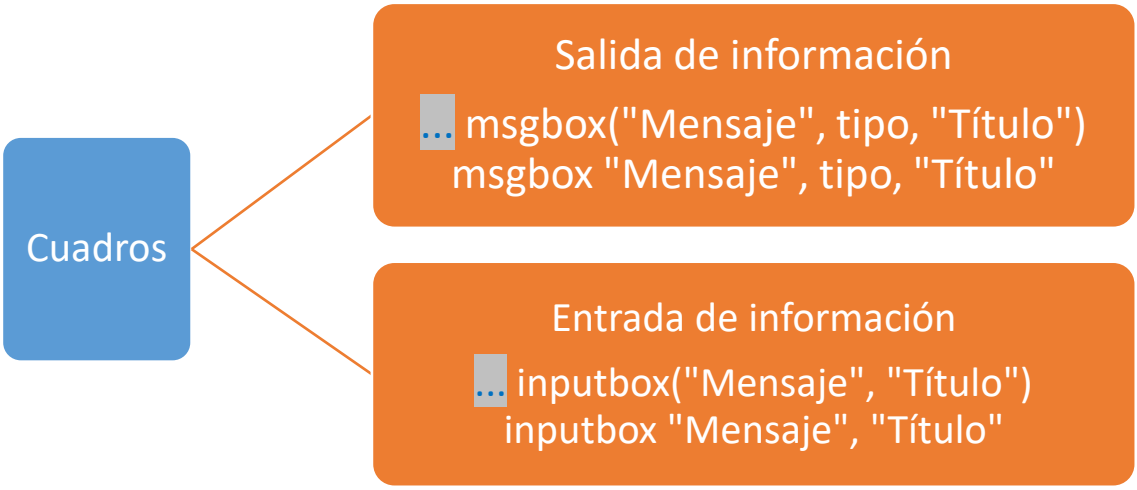
INPUTBOX("Escribir el país a filtrar:", "Criterio de filtro")

Ejemplos CON PARÉNTESIS:

Confirmación = MSGBOX("¿Confirma que desea salir de la aplicación?", vbYesNo, "Salir")

VariableEdad = INPUTBOX("Escribir edad del paciente:", "Ingresar dato")

Resumen:



La estructura condicional 'IF-THEN-ELSE'

La estructura IF-THEN-ELSE nos permite condicionar la ejecución de un grupo de comandos de nuestra macro al cumplimiento de una condición.

La estructura comienza con el comando *If* y finaliza con *End If* (si escribiéramos toda la sentencia en una única línea podremos omitir este último comando).

Luego del *If* escribiremos la condición a evaluar y el comando *Then*.

El bloque de comandos sólo será ejecutado si la condición se cumple y sólo cuando ello ocurra. Para el caso de que ello no suceda podremos indicar que realice otro bloque de instrucciones a través del comando *Else*.

Estructura:

```
IF condición THEN
        Comandos si la condición es VERDADERA
ELSE
        Comandos si la condición es FALSA
END IF
```

Por ejemplo:

```
IF Range("E2").Value >= 6 THEN
        msgbox "APROBADO"
ELSE
        msgbox "DESAPROBADO"
END IF
```

En este ejemplo, chequeamos la nota de un examen, que está escrita en la celda E2. Si la nota es igual o superior a 6 aparecerá un cuadro de mensaje que diga "APROBADO", caso contrario, aparecerá un cuadro de mensaje que diga "DESAPROBADO".

Nota: Es recomendable dejar sangría en cada línea dentro de una estructura para que visualmente sepamos que esos comandos están allí adentro.

Ejemplo de la sentencia IF-THEN-ELSE en una sola línea:

```
If Nota >= 6 Then Resultado = "Aprobado" Else Resultado = "No aprobado"
```

Anidar una estructura dentro de otra

Anidar implica introducir una estructura dentro de otra. VBA nos permite anidar dentro de una estructura o bucle cualquier otro (el mismo o uno distinto) y nos permite hacerlo en varios niveles.

Por ejemplo:

```
IF Range("E2").Value >= 6 THEN
        msgbox "APROBADO"
ELSE
  IF Range("E2").Value >= 4 THEN
        msgbox "DESAPROBADO"
  ELSE
        msgbox "APLAZADO"
  END IF
END IF
```

En este ejemplo, chequeamos la nota de un examen, que está escrita en la celda E2. Si la nota es igual o superior a 6 aparecerá un cuadro de mensaje que diga "APROBADO", caso contrario, se evaluará si la persona está desaprobada o aplazada. En este último supuesto, si la nota es superior o igual a 4, aparecerá un cuadro de mensaje que diga "DESAPROBADO", caso contrario, "APLAZADO".

Utilizar variables. Tipos de variables

Una variable es un espacio virtual en la memoria del ordenador que permite almacenar un valor que puede cambiar a lo largo de la ejecución de la macro pese a que su nombre se mantenga.

Las variables tienen un NOMBRE, un TIPO y un VALOR almacenado.

El NOMBRE de una variable puede ser elegido libremente, siempre que no se utilice alguna palabra reservada por VBA, que no se utilicen espacios y que no se asignen a dos variables el mismo nombre.

El TIPO de una variable es el tipo de dato que almacena, como, por ejemplo, un valor numérico, una fecha o una cadena de texto.

El VALOR de una variable es el dato que guardaremos dentro de la misma, para ser utilizado en nuestra macro.

Según el dato que almacenen las variables pueden clasificarse según su TIPO:

Tipo de variable	En VBA	Valores que puede almacenar el tipo de variable
BYTE	Byte	Almacena un número entero entre 0 y 255
ENTERO	Integer	Almacena un número entero entre -32.768 y 32.767.
ENTERO LARGO	Long	Almacena un número entero entre 2.147.483.648 y 2.147.483.647.
DECIMAL SIMPLE	Single	Almacena un número decimal entre $-3,4*10^{38}$ y $3,4*10^{38}$
DECIMAL DOBLE	Double	Almacena un número decimal entre $-1,79*10^{308}$ y $1,79*10^{308}$
CADENA DE TEXTO	String	Almacena una cadena de texto alfanumérica.
BOOLEANO	Boolean	Almacena valores lógicos. Sólo permite dos valores: VERDADERO o FALSO.
FECHA	Date	Almacena un valor de fecha desde el 01/01/0001 al 31/12/9999 y/o un horario.
OBJETO	Object	Almacena un objeto.
VARIANTE	Variant	Almacena cualquier tipo de dato.
NOTA: Las variables que admiten números decimales admiten por supuesto números enteros. Sin embargo, ello no es aconsejable porque el espacio de la memoria asignado a los decimales se desperdicia.		

Momentos de una variable

La vida de una variable en una macro atraviesa 3 momentos:

- Declaración: se las debe declarar o crear con un nombre y un tipo específico.
- Asignación de valor: se les debe almacenar un determinado dato (que puede variar posteriormente).
- Uso o llamado: a lo largo de la macro se podrá hacer referencia al valor de dicha variable mencionándola.

1º Declaración de variables

Para crear una variable en nuestra macro y luego poder utilizarla tendremos que primero declararla. Las variables se declaran con el comando DIM seguido del nombre, del comando AS y del tipo de variable.

```
Dim Nombre As String
Dim Edad As Byte
```

En el primer ejemplo estamos declarando la variable "Nombre" que es de tipo "cadena de texto", es decir, que puede almacenar una cadena de texto alfanumérica. En el segundo, "Edad" que es de tipo "número entero" y puede almacenar valores numéricos sin decimales entre 0 y 255 (podríamos haber usado el tipo de variable "Integer" pero como sabemos que nuestra variable no utilizará números anteriores a 0 ni posteriores a 255 puesto que se refiere a la edad de una persona conviene usar Byte para ahorrar espacio en la memoria del ordenador).
Podemos asignar varias variables en una sola línea obviando el comando Dim a partir de la segunda y separándolas con una "," (coma):

```
Dim Nombre As String, Edad As Byte
```

Nota: el nombre de las variables no se escribe entre comillas como se debe hacer con cualquier cadena de texto. Ello se debe a que, una vez declarada la variable, funcionará a la par de cualquier otro objeto en Excel y podremos referirnos a ellas en cualquier momento.

2º Asignación de un valor a una variable

Para asignar un valor a la variable simplemente la nombraremos y, mediante el operador "=" le escribiremos el valor a asignar:

```
Nombre = "José"
Edad = 24
```

Nota: al asignar un valor a la variable cuando se trata de una cadena de texto sí es necesario incluir las comillas al contenido que es texto.
Nota 2: Para las variables que aceptan números decimales los mismos deben escribirse utilizando el signo "." (punto) en VBA. No se puede utilizar la coma para la separación de los decimales, aunque Excel luego convierta el punto en una coma al ejecutar la macro. Por otro lado, el Editor no admite la separación en unidades de mil, de millón, etcétera.

Para las variables de tipo *Boolean*, los valores posibles son True y False (Verdadero y falso en inglés):

```
Dim AceptarOperacion As Boolean
AceptarOperacion = True
```

Para las variables de tipo Date, los valores a escribir se deben escribir entre comillas y los datos separarse con barras (no guiones). El horario no es necesario incluirlo (en ese caso se considerará la hora 00:00:00); pero si se lo incluye irá luego de la fecha separado con un espacio. Los segundos tampoco son necesarios si se incluye el horario (se considerarán 00):

```
Dim FechaDePago As Date
FechaDePago = "01/11/2018 10:30:00"
```

También es posible utilizar el carácter # alrededor de las fechas, sin embargo, esto tendrá en cuenta que la fecha se escribe en el siguiente formato (mes/día/año): #01/11/2018# se mostrará como 11/01/2018.
A su vez, podemos usar la variable Date sólo para trabajar con horarios sin escribir ninguna fecha.

3º Uso o llamado de una variable

Una vez declarada la variable y con un valor asignado, se la podrá utilizar en cualquier momento haciendo referencia al nombre de la misma.

Msgbox “La fecha de pago es “ & FechaDePago Cells(“E3”).Value = Edad

Resumen:

1. Crear / Declarar	**Dim *Nombre* As *Tipo*** **Ejemplos**: Dim Edad As Integer Dim Nombre As String Dim FechaDeNacimiento As Date
2. Asignar algún valor	***Nombre* = *Valor*** **Ejemplos**: Edad = 40 Nombre = Valor FechaDeNacimiento = "01-01-1985"
3. Usar / Llamar	**Ejemplos varios**: msgbox Nombre msgbox "Bienvenido/a " & Nombre Range("A1").Value = Nombre

Variables locales, privadas y públicas

Las variables declaradas dentro de nuestra macro, es decir, entre los comandos “Sub” y “End Sub” mediante el comando “Dim” son variables locales. Esto indica que sólo existen y pueden ser usadas o llamadas dentro de la macro en la cual son declaradas.

Las variables privadas son aquellas que se declaran para ser usadas por todas las macros contenidas en el mismo archivo (módulo del Editor de Visual Basic) y las públicas, por todas las macros independientemente del módulo en donde se encuentren.

Para declarar una variable privada cambiaremos el comando “Dim” por “Private”, y en el caso de las públicas por “Public”. Es importante remarcar también que la misma debe ser declarada fuera de la sección “Sub” y “End Sub” de una macro.

```
Private Apellido As String

Sub MacroFormulario()
        Apellido = "Dominguez"
End Sub

Sub MacroClientes()
        Apellido = "Gutierrez"
End Sub
```

Funciones en VBA

Una función en VBA es un comando que realiza una determinada operación con los datos (argumentos) que le brindemos. Muchas de estas funciones son las que usamos en el mismo Excel, aunque su nombre y su sintaxis varían. La sintaxis indica la forma en que debe ser escrita la función.

La sintaxis de una función en VBA es la siguiente:

Worksheetfunction. Nombre (argumento1, argumento2, argumento3)

- Comando WORSKHEETFUNCTION.
- Luego, el nombre de la función (en inglés).
- Luego, paréntesis (de apertura y de cierre).
- Dentro de los paréntesis, se incluirán los datos / argumentos que la función necesite.
    - Habrá que conocer cada función en particular para saber si son necesarios uno o más argumentos y si los mismos son obligatorios u optativos.
    - Si la función llevara más de un argumento cada uno va separado del otro con una coma.
    - Si la función no llevara ningún argumento los paréntesis deben ser escritos de todas formas.

Ejemplos de funciones en VBA con WORKSHEETFUNCTION:

AND	Devuelve VERDADERO si todos sus argumentos son VERDADERO.
AVERAGE	Devuelve el promedio de sus argumentos.
COUNT	Cuenta cuántos números hay en la lista de argumentos.
COUNTA	Cuenta cuántos valores hay en la lista de argumentos.
COUNTBLANK	Cuenta el número de celdas en blanco de un rango.
COUNTIF	Cuenta el número de celdas, dentro del rango, que cumplen el criterio especificado.
COUNTIFS	Cuenta el número de celdas, dentro del rango, que cumplen varios criterios.
HLOOKUP	Busca en la fila superior de una matriz y devuelve el valor de la celda indicada.
IFERROR	Devuelve un valor que se especifica si una fórmula lo evalúa como un error; de lo contrario, devuelve el resultado de la fórmula.
LOOKUP	Busca valores de un vector o una matriz.
MAX	Devuelve el valor máximo de una lista de argumentos.
MIN	Devuelve el valor mínimo de una lista de argumentos.
OR	Devuelve VERDADERO si cualquier argumento es VERDADERO.
ROUND	Redondea un número al número de decimales especificado.
SUM	Suma sus argumentos.
SUMIF	Suma las celdas especificadas que cumplen unos criterios determinados.
SUMIFS	Suma las celdas de un rango que cumplen varios criterios.
VLOOKUP	Busca en la primera columna de una matriz y se mueve en horizontal por la fila para devolver el valor de una celda.

Worksheetfunction.CountIf(Range("A:A"),Range("B2").Value)

El ejemplo muestra una función CONTAR.SI (COUNTIF) que cuenta dentro del rango A:A cuántas veces aparece el dato escrito en el valor de la celda B2.

Algunas funciones tienen una sintaxis especial y no requieren de WORKSHEETFUNCION:

Ejemplos de funciones en VBA sin WORKSHEETFUNCTION

Left(texto, largo)	Recorta una determinada cantidad de caracteres a contar desde la izquierda de una cadena. **Texto** = La cadena de texto sobre la cual trabajará. **Largo** = La cantidad de caracteres a recortar (núm.) Si Largo es mayor que el largo del texto, entonces devolverá el texto entero. Si es igual a 0, el resultado será una cadena de texto de longitud cero.
Right(texto, largo)	Recorta una determinada cantidad de caracteres a contar desde la derecha de una cadena. **Texto** = La cadena de texto sobre la cual trabajará. **Largo** = La cantidad de caracteres a recortar (núm.) Si Largo es mayor que el largo del texto, entonces devolverá el texto entero. Si es igual a 0, el resultado será una cadena de texto de longitud cero.
Len(texto)	Indica el número de caracteres de una cadena (se contabilizan los caracteres no imprimibles y los espacios en blanco) **Texto** = La cadena de texto sobre la cual trabajará.
LCase(Texto)	Convierte una cadena a letras minúsculas. **Texto** = La cadena de texto sobre la cual trabajará.
Ucase(Texto)	Convierte una cadena a letras mayúsculas. **Texto** = La cadena de texto sobre la cual trabajará.

Operadores en VBA

Operadores que está permitido usar en programación en VBA y cuál es su función.

OPERADORES EN VBA		
OPERADORES BÁSICOS		
OPERADOR	**SIGNIFICADO**	**EJEMPLO**
.	Mencionar un objeto de rango inferior. Mencionar una propiedad o una acción a un objeto.	**Sheets(1).Range("A1").Value** **Sheets(1).Range("A1").ClearAll**
,	Separar valores y separar parámetros.	**Range("A1").Autofilter Field:=3, Criteria1:="Italia"**
""	Indicar un valor de texto.	**Range("A1").Autofilter Field:=3, Criteria1:="Italia"**
'	Indicar un comentario en una línea vacía o en un comando.	**'Esta macro genera un recibo de sueldo**
_	Cortar un comando en más de una línea.	**Range("A1").Autofilter 3, _** **"Italia"**
:	Escribir más de un comando en una misma línea.	**Range("A1").Value = 10 :** **Range("A2").Value = 20**
=	Asignar un valor a una variable o a una propiedad.	**Range("A1").Value = 10** **VariableEdad = 10**
:=	Asignar un valor a un parámetro.	**Range("A1").Autofilter Field:=3, Criteria1:="Italia"**
&	Concatenar cadenas de texto.	**Msgbox "Bienvenido" & variableNombre**
OPERADORES LÓGICOS o DE COMPARACIÓN		
OPERADOR	**SIGNIFICADO**	**EJEMPLO**
=	Igual	**Activecell.value = ""**
<	Menor	**Activecell.value > 0**

>	Mayor	**Activecell.value < 0**
<>	Distinto	**Activecell.value <> “”**
<=	Menor o igual	**Activecell.value <= 0**
>=	Mayor o igual	**Activecell.value >= 0**
OPERADORES ARITMÉTICOS		
OPERADOR	**SIGNIFICADO**	**EJEMPLO**
+	Suma	**variableMonto = valor1 + valor2**
-	Resta	**variableMonto = valor1 - valor2**
*	Multiplicación	**variableMonto = valor1 * valor2**
/	División correcta (5/2 = 2,5)	**variableMonto = valor1 / valor2**
\	División entera (no trabaja con el resto) (5/2 = 2)	**variableMonto = valor1 \ valor2**