# SENTENCIAS DE VBA

## *LOGICAS*:

**Sentencia If**



**If – Then**

Las sentencias If de VBA le permiten probar si las expresiones son Verdadero o Falso, ejecutando un código diferente basado en los resultados.

Veamos un ejemplo sencillo:

Sub si()

 If Range("a2").Value > 0 Then Range("b2").Value = "Positivo"

End Sub

Se comprueba si el valor del rango A2 es mayor que 0. Si es así, se establece el rango B2 igual a «Positivo»



Nota: Al probar las condiciones utilizaremos los [operadores de comparación](https://www.automateexcel.com/vba/comparison-operators-not-equal-to/) =, >, <, <>, <=, >=. Los discutiremos con más detalle más adelante en el artículo.

Esta es la sintaxis para una simple sentencia If de una línea:

If [expresión\_de\_prueba] then [acción]

Para facilitar la lectura, puede utilizar un carácter de [continuación de línea](https://www.automateexcel.com/vba/new-line-carriage-return/#Continuing_a_Statement_in_VBA) (guión bajo) para ampliar las sentencias If a dos líneas (como hicimos en la imagen anterior):

If [expresión\_de\_prueba] then \_

 [acción]

If Range("a2").Value > 0 then \_

 Range("b2").Value = "Positivo"

**End If**

La declaración «single-line» if anterior funciona bien cuando está probando una condición. Pero a medida que sus declaraciones IF se complican con múltiples condiciones, necesitará añadir un «End If» al final de la declaración if:

Sub si()

 If Range("a2").Value > 0 Then

 Range("b2").Value = "Positivo"

 End If

End Sub



Aquí la sintaxis es

If [expresión\_de\_prueba] Then

 [acción]

End If

El End If significa el final de la sentencia If.

Ahora vamos a añadir un ElseIf:

**ElseIf – Condiciones múltiples**

El ElseIf se añade a una sentencia If existente. ElseIf comprueba si una condición se cumple **SÓLO si las condiciones anteriores no se han cumplido**.

En el ejemplo anterior probamos si el valor de una celda es positivo. Ahora también probaremos si el valor de la celda es negativo con un ElseIf:

Sub If\_ElseIf()

 If Range("A2").Value > 0 Then

 Range("B2").Value = "Positivo"

 ElseIf Range("A2").Value < 0 Then

 Range("B2").Value = "Negativo"

 End If

End Sub



Puede usar múltiples ElseIfs para probar múltiples condiciones:

Sub If\_MultiplesCondiciones()

 If Range("a2").Value = "Gato" Then

 Range("b2").Value = "Miau"

 ElseIf Range("a2").Value = "Perro" Then

 Range("b2").Value = "Guau"

 ElseIf Range("a2").Value = "Pato" Then

 Range("b2").Value = "Cuack"

 End If

End Sub



Ahora añadiremos un **Else**:

**Else**

El **Else** se ejecutará si **no se cumplen otras condiciones anteriores**. Terminaremos nuestro ejemplo utilizando un Else para indicar que si el valor de la celda no es positivo o negativo, entonces debe ser cero:

Sub else\_()

 If Range("a2").Value > 0 Then

 Range("b2").Value = "Positivo"

 ElseIf Range("a2").Value < 0 Then

 Range("b2").Value = "Negativo"

 Else

 Range("b2").Value = "Cero"

 End If

End Sub



**If-Else**

El tipo más común de sentencia If es un simple If-Else:

Sub If\_Else()

 If Range("a2").Value > 0 Then

 Range("b2").Value = "Positvo"

 Else

 Range("b2").Value = "No Negativo"

 End If

End Sub



**IFs anidados**

También puede «anidar» las sentencias if unas dentro de otras.

Sub Ifs\_Anidados()

 If Range("a2").Value > 0 Then

 Range("b2").Value = "Positivo"

 Else

 If Range("a2").Value < 0 Then

 Range("b2").Value = "Negativo"

 Else

 Range("b2").Value = "Cero"

 End If

 End If

End Sub



**IF – Or, And, Xor, Not**

**If – Or**

El operador **Or** comprueba si **se cumple al menos una condición**. El siguiente código probará si el valor en el Rango A2 es menor a 5,000 o mayor a 10,000:

Sub If\_Or()

 If Range("a2").Value < 5000 Or Range("a2").Value > 10000 Then

 Range("b2").Value = "Fuera de rango"

 End If

End Sub



Puede incluir varios Ors en una línea:

Sub If\_Varios\_Or()

 If Range("a2").Value < 5000 Or Range("a2").Value > 10000 Or Range("a2").Value = 9999 Then

 Range("b2").Value = "Fuera de rango"

 End If

End Sub

Si va a utilizar varios Ors, se recomienda utilizar un carácter de continuación de línea para que su código sea más fácil de leer:

Sub If\_Varios\_Or()

 If Range("a2").Value < 5000 Or \_

 Range("a2").Value > 10000 Or \_

 Range("a2").Value = 9999 Then

 Range("b2").Value = "Fuera de Rango"

 End If

End Sub



**If And**

El operador And permite comprobar si **se cumplen TODAS las condiciones**.

Sub If\_And()

 If Range("a2").Value >= 5000 And Range("a2").Value <= 10000 Then

 Range("b2").Value = "En Rango"

 End If

End Sub



**If Xor**

El operador Xor permite comprobar si **se cumple exactamente una condición**. Si se cumplen cero condiciones, Xor devolverá FALSE, Si se cumplen dos o más condiciones, Xor también devolverá false. Rara vez he visto el uso de Xor en la programación VBA.

**If Not**

El operador Not se utiliza para convertir FALSE en TRUE o TRUE en FALSE:

Sub IF\_Not()

 MsgBox Not (True)

End Sub



Observe que el operador Not requiere paréntesis alrededor de la expresión a cambiar. El operador Not también puede aplicarse a las sentencias If:

Sub if\_not\_ii()

 If Not (Range("a2").Value >= 5000 And Range("a2").Value <= 10000) Then

 Range("b2").Value = "Fuera de Rango"

 End If

End Sub



**Comparaciones If**

Cuando se realizan comparaciones, normalmente se utiliza uno de los operadores de comparación:

| **Operador de Comparación** | **Explicación** |
| --- | --- |
| = | Igual a |
| <> | No es igual a |
| > | Mayor que |
| >= | Mayor o igual que |
| < | Menos que |
| <= | Menor o igual que |

Sin embargo, también puede utilizar cualquier expresión **o**[**función**](https://www.automateexcel.com/vba/function/)que dé como resultado TRUE o FALSE

**If – Función booleana**

Cuando construya expresiones para sentencias If, también puede utilizar cualquier función que genere TRUE o FALSE. VBA tiene algunas de estas funciones:

| **Función** | **Descripción** |
| --- | --- |
| [IsDate](https://www.automateexcel.com/vba/isdate-function/) | Devuelve TRUE si la expresión es una fecha válida |
| [IsEmpty](https://www.automateexcel.com/vba/isempty-function/) | Comprueba si hay celdas en blanco o variables no definidas |
| [IsError](https://www.automateexcel.com/vba/iserror-function/) | Comprueba si hay valores de error |
| [IsNull](https://www.automateexcel.com/vba/isnull-function/) | Comprueba si hay un valor NULL |
| [IsNumeric](https://www.automateexcel.com/vba/isnumeric-function/) | Comprueba si el valor es numérico |

Pueden ser llamadas así:

If IsEmpty(Range("A1").Value) Then MsgBox "Celda Vacía"

Excel también tiene muchas funciones adicionales que pueden ser llamadas usando [WorksheetFunction](https://www.automateexcel.com/vba/worksheet-functions-in-macro/). Aquí hay un ejemplo de la [función IsText de Excel](https://www.automateexcel.com/functions/istext-formula-excel/)

:

If Application.WorksheetFunction.IsText(Range("a2").Value) Then \_

 MsgBox "La celda es texto"

También puede crear sus propias Funciones Definidas por el Usuario (UDFs). A continuación crearemos una simple función booleana que devuelva TRUE. Luego llamaremos a esa función en nuestra sentencia If:

Sub If\_Funcion()

 If FuncionVerdadero Then

 MsgBox "True"

 End If

End Sub

Function FuncionVerdadero() As Boolean

 FuncionVerdadero = True

End Function



**Comparación de texto**

También puede comparar texto de forma similar a la comparación de números:

Msgbox "a" = "b"

Msgbox "a" = "a"

Al comparar texto, debe tener en cuenta si el (Case) tipo de letra  (mayúsculas o minúsculas). Por defecto, VBA considera que las letras con diferentes (Case) no coinciden. En otras palabras, «A» <> «a».

Si desea que VBA ignore las mayúsculas y minúsculas, debe añadir la declaración de [Opción de Comparación de Texto](https://www.automateexcel.com/vba/case-sensitive/)

en la parte superior de su módulo:

Option Compare Text

Después de hacer esa declaración «A» = «a»:

Option Compare Text

Sub If\_Text()

 MsgBox "a" = "A"

End Sub

**VBA If Like**

El [Operador Like](https://www.automateexcel.com/vba/like-operator/)  de VBA le permite hacer comparaciones inexactas de texto. Haga clic en el enlace «Operador Like» para aprender más, pero mostraremos un ejemplo básico a continuación:

Dim nombre String

nombre = "Mr. Charles"

If nombre Like "Mr\*" Then

 MsgBox "True"

Else

 MsgBox "False"

End If

Aquí estamos usando un [comodín](https://www.automateexcel.com/formulas/wildcard-characters/) de asterisco «\*». El \* representa cualquier número de caracteres. Así que la declaración If anterior devolverá TRUE. El operador Like es una herramienta extremadamente poderosa, pero a menudo subutilizada, para tratar con texto.

**Bucles If**

[Los bucles de VBA](https://www.automateexcel.com/vba/loops/) le permiten repetir acciones. La combinación de IF-ELSEs con bucles es una gran manera de procesar rápidamente muchos cálculos. Continuando con nuestro ejemplo de Positivo / Negativo, agregaremos un Bucle [For Each](https://www.automateexcel.com/vba/for-each/) para [recorrer un rango de celdas](https://www.automateexcel.com/vba/loop-through-range/):

Sub If\_Bucle()

Dim Celda As Range

 For Each Celda In Range("A2:A6")

 If Celda.Value > 0 Then

 Celda.Offset(0, 1).Value = "Positivo"

 ElseIf Celda.Value < 0 Then

 Celda.Offset(0, 1).Value = "Negativo"

 Else

 Celda.Offset(0, 1).Value = "Cero"

 End If

 Next Celda

End Sub



**Ejemplos de If Else**

**Comprobar si la celda está vacía**

Este código comprobará si una celda está vacía. Si está vacía ignorará la celda. Si no está vacía, enviará el valor de la celda a la celda de la derecha:

Sub If\_Celda\_Vacia()

 If Range("a2").Value <> "" Then

 Range("b2").Value = Range("a2").Value

 End If

End Sub



**Comprobar si la celda contiene un texto específico**

La [función Instr](https://www.automateexcel.com/vba/instr-find-text/)

comprueba si una cadena de texto se encuentra en otra cadena. Utilícela con una sentencia If para comprobar si una celda contiene un texto específico:

If Instr(Range("A2").value,"texto") > 0 Then

 Msgbox "Texto Encontrado"

End If

**Comprobar si la celda contiene texto**

Este código comprobará si una celda es texto:

Sub If\_Celda\_Es\_Texto()

 If Application.WorksheetFunction.IsText(Range("a2").Value) Then

 MsgBox "Celda contiene texto"

 End If

End Sub

**If Goto**

Puede utilizar el resultado de una sentencia If para [«Ir a» otra sección de código](https://www.automateexcel.com/vba/goto-line-label/)

.

Sub IfGoTo()

 If IsError(Cell.Value) Then

 GoTo Skip

 End If

 'Algo de código

Skip:

End Sub

**Borrar fila si la celda está en blanco**

Usando Ifs y bucles puede [probar si una celda está en blanco y si es así borrar toda la fila](https://www.automateexcel.com/vba/remove-blank-rows/)

Sub borrarFilaSiCeldaEsBlanco()

 Dim Celda As Range

 For Each Celda In Range("A2:A10")

 If Celda.Value = "" Then Celda.EntireRow.Delete

 Next Celda

End Sub

**If MessageBox Sí / No**

Con los cuadros de mensaje [de VBA](https://www.automateexcel.com/vba/yes-no-message-box/)

puede pedir al usuario que seleccione entre varias opciones. La Caja de Mensajes Si/No le pide al usuario que seleccione Si o No. Usted puede agregar una Caja de Mensajes Si/No a un procedimiento para preguntarle al usuario si le gustaría continuar ejecutando el procedimiento o no. Usted maneja la entrada del usuario usando una sentencia If.

Aquí está el Cuadro de Mensajes Sí/No en la práctica:



Sub MsgBoxVariable()

 Dim respuesta As Integer

 respuesta = MsgBox("Quiere continuar?", vbQuestion + vbYesNo)

 If respuesta = vbYes Then

 MsgBox "Si"

 Else

 MsgBox "No"

 End If

End Sub

**VBA If, ElseIf, Else en Access VBA**

Las funciones If, ElseIf y Else funcionan exactamente igual en Access VBA que en Excel VBA. Puede utilizar una sentencia If para comprobar si hay registros en un [Recordset](https://www.automateexcel.com/vba/access-vba-recordsets-open-count-loop-and-more/).

## Repetitivas o Bucles



En VBA los bucles te permiten **recorrer un conjunto de objetos o valores y analizarlos uno por uno** realizando tareas específicas en cada elemento. De forma sencilla: ejecutas la macro una vez y una serie de instrucciones se ejecutan un número determinado de veces.

Imagina que en una hoja tienes un rango de celdas y te gustaría resaltar con otro color las celdas de las filas pares. Para esta tarea puedes utilizar un bucle para recorrer cada celda del rango, averiguar el número de fila y si es par, colorear el fondo.

Es un pequeñísimo ejemplo de uso de bucles para automatizar tareas. Algunos ejemplos más que me vienen a la cabeza...

* Recorrer un rango de celdas y darle un formato determinado a las que contengan un texto específico.
* Proteger o desproteger todas las hojas a la vez.
* Cerrar todos los libros excepto el que contiene el código.
* Recorrer y almacenar los valores de una matriz.
* Vaciar todos los controles de un formulario.
* ... y así puedo seguir con muchas más tareas...

Ahora que más o menos sabes para qué se usan los bucles, te mostraré qué tipos existen y las sintaxis de cada uno. Por supuesto, complementaré cada uno con ejemplos.

## El bucle For – Next

Estoy seguro de que este bucle es el más utilizado en cualquier lenguaje de programación.

Sirve para **ejecutar un conjunto de instrucciones un número de veces determinado**.

Por ejemplo, si quisieras sumar los números del 1 al 10 de forma manual supongo que lo harías sumando los dos primeros, luego sumarías el tercero al resultado, luego sumarías el cuarto al resultado... y así hasta completar los 10. ¿Me equivoco?

Pues en VBA se aplica la misma lógica en el bucle For-Next.

Al crear la macro se le indica al código cuántas veces quieres ejecutar el bucle y qué quieres que haga el código cada vez que se ejecute. Observa la sintaxis del bucle:

For contador = comienzo To final [Step paso]

 [bloque de código con instrucciones

Next [contador]

La variable contador cuenta el número de veces que pasa el flujo de código por ella desde el valor comienzo hasta el final.

Siguiendo el ejemplo anterior, si quieres contar los números enteros del 1 al 10, la variable comienzo sería 1 y la variable final sería 10.

## Cómo sumar los 10 primeros números

Este es el código que suma los 10 primeros números positivos usando el bucle For-Next. Después de calcular la suma muestra un cuadro de mensaje con el resultado:

Sub Suma10()

 Dim Total As Integer

 Dim contador As Integer

 Total = 0

 For contador = 1 To 10

 Total = Total + contador

 Next contador

 MsgBox Total

End Sub

Las [variables](https://ayudaexcel.com/variables-constantes-tipos-datos/) Total y contador se declaran a nivel de procedimiento, por tanto, al comienzo de la ejecución sus valores son 0.

Una vez que el flujo entra en el bucle se mantienen los valores de las variables después de cada pasada. Así que en la primera vuelta el valor de contador es 1 y el valor de Total también es 1. Después de la segunda vuelta el valor de contador es 2 y por tanto el de Total será 3 (2+1). Así sucesivamente hasta completar las 10 vueltas establecidas en el bucle.

Al finalizar, la variable Total tiene la suma de los 10 primeros números positivos. En la última línea de la macro, se muestra un mensaje (con MsgBox) con el resultado del cálculo.

## Cómo sumar los 5 primeros números pares

En el ejemplo anterior la variable contador va tomando números consecutivos de 1 a 10. Para sumar los 5 primeros números pares el código anterior no sirve porque es necesario que la variable contador vaya tomando los números pares (un número de cada dos). Observa el siguiente procedimiento:

Sub SumaPares()

 Dim Total As Integer

 Dim contador As Integer

 Total = 0

 For contador = 2 To 10 Step 2

 Total = Total + contador

 Next contador

 MsgBox Total

End Sub

Observa que en este caso el bucle se inicia en el valor 2 y se ha añadido la cláusula Step 2.

Cuando usas la cláusula Step le estás diciendo al código que incremente el valor de contador en ese valor cada vez que el flujo de código completa una vuelta por el bucle.

Así que el valor de contador comienza en 2, luego se convierte en 4, 6, 8 y 10 a medida que se pasa por el inicio.

**ATENCIÓN**: este ejemplo es solo una forma de hacerlo. También podrías ejecutar el bucle desde el 1 hasta el 10 y dentro del bucle comprobar si se trata de un número par o impar.

El valor de Step también puede ser negativo. En este caso el contador comienza en un número más alto y en cada vuelta irá disminuyendo hasta llegar al valor final.

## Cómo introducir números consecutivos en las celdas seleccionadas

También puedes usar el bucle For-Next para recorrer un conjunto de celdas. Observa el siguiente procedimiento:

Sub NumerosConsecutivos()

 Dim Rng As Range

 Dim contador As Integer

 Dim CuentaFila As Integer

 Set Rng = Selection

 CuentaFila = Rng.Rows.Count

 For contador = 1 To CuentaFila

 ActiveCell.Offset(contador - 1, 0).Value = contador

 Next contador

End Sub

Esta macro cuenta el número de filas seleccionadas y luego asigna este valor a la variable CuentaFila. A continuación se ejecuta el bucle desde 1 hasta CuentaFila.

Como antes de ejecutar la macro se pueden seleccionar cualquier número de celdas, he establecido la variable Rng en Selection. De esta forma ya es posible usar la variable Rng para hacer referencia a las celdas seleccionadas.

## Proteger todas las hojas en el libro activo

También es posible usar el bucle For-Next para recorrer todas las hojas del libro activo y protegerlas (o desprotegerlas) todas a la vez.

Este es el código que lo hace posible:

Sub ProtegerHojas()

 Dim i As Integer

 For i = 1 To ActiveWorkbook.Worksheets.Count

 Worksheets(i).Protect

 Next i

End Sub

En el ejemplo se cuenta el número de hojas usando la instrucción ActiveWorkbook.Worksheets.Count. Esto le dice a VBA cuántas veces debe ejecutarse el bucle.

En cada vuelta se hace referencia a la hoja con su número de índice (con Worksheets(i)) y la protege.

Por supuesto puedes utilizar este mismo código para desproteger todas las hojas del libro. Solo tienes que cambiar el método Protect por Unprotect.

## Los bucles For-Next anidados

Para conseguir automatizaciones más complejas puedes utilizar bucles For-Next anidados (uno dentro de otro).

Imagina que tienes cinco libros abiertos y quieres proteger todas las hojas de todos los libros. Este es el código que lo hace posible:

Sub ProtegerHojasLibros()

 Dim i As Integer

 Dim j As Integer

 For i = 1 To Workbooks.Count

 For j = 1 To Workbooks(i).Worksheets.Count

 Workbooks(i).Worksheets(j).Protect

 Next j

 Next i

End Sub

El flujo de código cuenta el número de libros abiertos y entra en el primero. A continuación, cuenta el número de hojas del primer libro. Luego las protege. Una vez protegidas entra en el segundo libro, cuenta el número de hojas, las protege... y así sucesivamente hasta que finaliza con el último libro abierto.

## Cómo salir de un bucle For-Next

Cuando quieres que el flujo de ejecución de la macro salga de un bucle For-Next antes de finalizar completamente su recorrido puedes usar la instrucción Exit For.

Habitualmente se usa cuando se cumple una determinada condición y no quieres continuar con el recorrido (por ejemplo, cuando haces una búsqueda y encuentras el resultado, no es necesario seguir buscando).

Imagina que tienes un rango con números en la columna A y quieres resaltar los negativos aplicando una fuente roja. En este caso es necesario analizar el valor de cada celda y luego cambiar el color de la fuente en consecuencia.

Para hacer el código más eficiente, se comprueba previamente si hay valores negativos en el rango. Si no hay números negativos puedes utilizar la instrucción Exit For para salir del procedimiento y así evitar que el flujo de código entre en el bucle.

Sub ResaltarNegativos()

 Dim Rng As Range

 Set Rng = Range("A1", Range("A1").End(xlDown))

 contador = Rng.Count

 For i = 1 To contador

 If WorksheetFunction.Min(Rng) >= 0 Then Exit For

 If Rng(i).Value < 0 Then Rng(i).Font.Color = vbRed

 Next i

End Sub

**ATENCIÓN**: Cuando utilizas la instrucción Exit For en un bucle For-Next anidado, el flujo de código saldrá del bucle interior en el que se ejecuta y pasará a ejecutar la siguiente línea del código del bucle exterior.

Sub EjemploCodigo()

 For i = 1 To 10

 For j = 1 To 10

 Exit For

 Next j

 Next i

End Sub

## El bucle Do While Loop

El bucle Do While permite comprobar una condición y ejecutar el bucle mientras se cumple esa condición. Existen dos tipos de sintaxis:

Do [While condición]

[Instrucciones a ejecutar]

Loop

y

Do

[Instrucciones a ejecutar]

Loop [While condición]

La diferencia entre las dos sintaxis está en que en la primera la condición se comprueba antes de ejecutar por primera vez el código. En el segundo ejemplo primero se ejecuta el código una vez y luego se comprueba la condición.

Esto significa que si la condición es falsa en ambos bucles, el código se ejecuta al menos una vez en el segundo caso (porque**la condición se comprueba después de que el código se haya ejecutado una vez**).

Te pongo algunos ejemplos:

## Cómo sumar los 10 primeros números

Es el mismo ejemplo que el primero. Antes utilicé el bucle For-Next, ahora te muestro la forma de hacerlo con Do While.

En este caso puedes usar el bucle Do While hasta que el siguiente número sea menor o igual a 10. Cuando el número sea mayor que 10 el bucle se detendrá.

Este es el código:

Sub Suma10()

 Dim i As Integer

 Dim Resultado As Integer

 i = 1

 Do While i <= 10

 Resultado = Resultado + i

 i = i + 1

 Loop

 MsgBox Resultado

End Sub

Al ejecutar la macro seguirá funcionando hasta que el valor de i se convierta en 11. Cuando esto sucede, el bucle termina, pues la condición se convierte en False.

Una vez finalizado el bucle se muestra el valor de la variable Resultado, que ha ido recogiendo los datos en cada vuelta del bucle.

## Introducir las fechas del mes actual

Imagina que quieres introducir todos los días del mes actual en una columna de la hoja.

Puedes hacerlo utilizando el siguiente código con el bucle Do While:

Sub DiasMesActual()

 Dim DiaMes As Date

 Dim i As Integer

 i = 0

 DiaMes = DateSerial(Year(Date), Month(Date), 1)

 Do While Month(DiaMes) = Month(Date)

 Range("A1").Offset(i, 0) = DiaMes

 i = i + 1

 DiaMes = DiaMes + 1

 Loop

End Sub

El procedimiento comienza introduciendo los días del mes actual en la columna A. El bucle continúa mientras el valor del mes de la variable DiaMes coincide con el mes actual. En caso contrario, el bucle finaliza.

## Cómo salir de un bucle Do

Al igual que en bucle For-Next también es posible salir de un bucle Do While usando la instrucción Exit Do. Cuando el flujo de código llega a esta línea, sale del bucle y pasa el control a la siguiente línea después de Loop.

En este tipo de bucles, muchas veces no sabes de antemano cuántas veces se repetirá. En caso de que existan fallos o condiciones que siempre se cumplen, el bucle se ejecutaría infinitas veces bloqueando el ordenador.

Por este motivo es útil tener presente la instrucción Exit Do y utilizarla mientras estás haciendo pruebas con el código. Una vez has comprobado que el procedimiento se ejecuta el número de veces adecuado, puedes eliminar o comentar la línea Exit Do.

## El bucle Do Until Loop

El bucle Do Until es muy parecido al Do While.

Si el bucle Do While se ejecuta mientras se cumple una condición, el bucle Do Until lo hace hasta que se cumpla la condición dada.

También tiene dos sintaxis:

Do [Until condición]

[Instrucciones a ejecutar]

Loop

y

Do

[Instrucciones a ejecutar]

Loop [Until condición]

La diferencia, al igual que en los bucles Do While es que en el primero se ejecuta la condición antes de ejecutar cualquier instrucción y en el segundo, se ejecutan primero las instrucciones y luego se evalúa la condición.

Te muestro los mismos ejemplos de la sección anterior, pero usando el bucle Do Until:

## Cómo sumar los 10 primeros números

En este caso usaré el bucle Do Until para calcular la suma de los 10 primeros números:

Sub Suma10()

 Dim i As Integer

 Dim Resultado As Integer

 i = 1

 Do Until i > 10

 Resultado = Resultado + i

 i = i + 1

 Loop

 MsgBox Resultado

End Sub

El bucle sigue funcionando hasta que el valor de i se convierte en 11. Cuando esto sucede el bucle finaliza, pues la condición se convierte en verdadera.

## Introducir fechas del mes actual

En este ejemplo usaré el bucle Do Until para introducir en la columna A (a partir de la fila 1), las fechas correspondientes al mes actual:

Sub DiasMesActual()

 Dim DiaMes As Date

 Dim i As Integer

 i = 0

 DiaMes = DateSerial(Year(Date), Month(Date), 1)

 Do Until Month(DiaMes) <> Month(Date)

 Range("A1").Offset(i, 0) = DiaMes

 i = i + 1

 DiaMes = DiaMes + 1

 Loop

End Sub

El bucle anterior continúa funcionando hasta que el mes de la variable DiaMes no sea igual al de la fecha de hoy.

## El bucle For Each-Next

Los tres tipos de bucle anteriores podemos encontrarlos en todos los lenguajes de programación. VBA cuenta con un bucle especial, For Each-Next, que es capaz de recorrer un conjunto de objetos o colecciones. Algunos ejemplos de colecciones son:

* Los libros de Excel abiertos.
* Las hojas de un libro
* Las celdas seleccionadas
* Las celdas que se encuentran bajo un nombre.
* Los gráficos de un libro o una hoja.
* ... y cientos de etcéteras.

Con el bucle For Each-Next puedes recorrer cada uno de los objetos de una colección y realizar alguna acción sobre ellos.

Por ejemplo, puedes recorrer todas las hojas de un libro para protegerlas (como he hecho antes con For-Next) o recorrer todas las celdas de un rango para aplicar un formato determinado.

La ventaja de For Each-Next con respecto a For-Next es que no necesitas saber cuántos objetos contiene una colección.

El flujo de código pasa automáticamente por cada objeto y realiza las acciones especificadas. Por ejemplo, si quieres proteger todas las hojas de un libro, el código sería el mismo si tienes 2 hojas o 50.

Esta es la sintaxis:

For Each elemento In colección

 [Instrucciones a ejecutar]

Next [elemento]

Te muestro algunos ejemplos del bucle más útil de VBA

## Cómo proteger todas las hojas de un libro

Imagina que tienes un libro y quieres proteger todas las hojas.

Observa este código:

Sub ProtegerHojasLibros()

 Dim ws As Worksheet

 For Each ws In ActiveWorkbook.Worksheets

 ws.Protect

 Next ws

End Sub

En este procedimiento he definido la variable ws como un objeto de tipo hoja de trabajo.

Luego utilizo For Each-Next para recorrer cada ws (es decir cada hoja) en la colección de hojas del libro activo (definida con ActiveWorkbook.Worksheets.

Observa, que a diferencia del ejemplo en el que usaba  For-Next, aquí no hace falta contar cuántas hojas tiene el libro, pues el bucle ya se encarga de analizar todos los objetos de la colección de forma individual.

## Cómo guardar todos los libros abiertos

En muchas ocasiones se suele trabajar con varios libros a la vez. Para guardarlos todos una vez finalizado el trabajo, puede ser útil usar este procedimiento para ello:

Sub GuardarLibros()

 Dim wb As Workbook

 For Each wb In Workbooks

 wb.Save

 Next wb

End Sub

En este código no se muestra ningún aviso en caso de guardar el libro por primera vez. Se guardará en la carpeta predeterminada de Windows (suele ser Mis documentos).

## Cómo resaltar las celdas negativas en un rango

Con un bucle For Each-Next también puedes recorrer todas las celdas de un rango específico o de las celdas seleccionadas.

Esto puede ser útil cuando quieras analizar cada celda por separado y realizar alguna acción sobre ella. Por ejemplo, el siguiente procedimiento (ya salió en un ejemplo anterior con otro bucle) recorre todas las celdas seleccionadas y cambia el color de fondo de las celdas con valores negativos a rojo.

Sub ResaltaNegativos()

 Dim Celda As Range

 For Each Celda In Selection

 If Celda.Value < 0 Then

 Celda.Interior.Color = vbRed

 End If

 Next Celda

End Sub

En este código el bucle recorre la colección de celdas seleccionadas (Selection). La instrucción If se utiliza para comprobar si el valor de la celda es menor que cero o no. En caso de que lo sea, la celda se colorea en rojo, de lo contrario, se pasa a la siguiente celda.

Select Case en VBA

La instrucción Select Case en VBA es uno de los métodos más fáciles y utilizados en Excel cuando se trata de asignar y clasificar en función de una única variable con varios posibles resultados. En este tutorial vamos a ver su sintaxis, sus parámetros y cómo usarla con varios ejemplos muy sencillos.

**SINTAXIS**

La sintaxis de la instrucción Select Case en VBA es muy sencilla pero hay que saber cómo usarla para sacarle verdadero partido.  En el siguiente recorte de código VBA puedes ver cómo es la sintaxis.

Select Case mi\_expresion

 Case mi\_expresion\_1

 'código para expresion\_1

 Case mi\_expresion\_2

 'código para expresion\_2

 '...

 Case mi\_expresion\_n

 'código para expresion\_n

 Case Else

 'código para cuando no hay coincidencias

End Select

Donde:

* **mi\_expresión**: es una variable que queremos evaluar. Puede ser numérica o de tipo texto y será contra la que evaluemos *mi\_expresion\_1, mi\_expresion\_2…*
* **mi\_expresion\_1,2…n**: es un valor o conjunto de valores numéricos o de texto contra los que se evaluará *mi\_expresion*. En caso de evaluarse como verdadera se ejecutará el código… sino, pasará a evaluar la siguiente *mi\_expresion\_n*.
* **Case Else**: este valor es opcional y lo usaremos para cuando no hayamos conseguido ninguna equivalencia con *mi\_expresión* y queramos ejecutar algún código.

**EJEMPLOS**

**EJEMPLO1**

En el siguiente ejemplo vamos a ver el caso más sencillo de Selec Case en VBA. Las condiciones que se evalúan tienen que ser iguales a la expresión inicial. En este ejemplo se quiere evaluar la variable Nombre. En función del Nombre se asignará un valor de color de fondo u otro.

Select Case Nombre

 Case "Enrique"

 color\_fondo = vbYellow

 Case "Juan"

 color\_fondo = vbRed

 Case Else

 color\_fondo = vbWhite

End Select

**EJEMPLO 2**

En este ejemplo se evalúa la variable Edad y en función de su valor (dado por intervalos) se da un atributo a la variable tipo\_edad.

Select Case Edad

 Case 1 To 14

 tipo\_edad = "niño"

 Case 15 To 20

 tipo\_edad = "adolescente"

 Case Else

 tipo\_edad = "adulto"

End Select

**EJEMPLO 3**

En este ejemplo, en función de la primera letra del ID (que es la variable que vamos a evaluar) asignaremos un valor a la variable tipo para clasificar el Primer\_numero\_ID.

Select Case Primer\_numero\_ID

 Case 1, 2, 3

 tipo = "grupo1"

 Case 4, 5, 6

 tipo = "grupo2"

 Case Else

 tipo = "grupo3"

End Select

**EJEMPLO 4**

Por último, en este ejemplo, para evaluar la variable Altura, las condiciones que se utilizan son expresiones aritméticas.

Select Case Altura

 Case Is < 140

 tipo = "bajito"

 Case Is < 170

 tipo = "mediano"

 Case Else

 tipo = "alto"

End Select