# Introducción a VBA

Lo primero de todo

|  |  |
| --- | --- |
|  | Guarda el fichero “libro de Excel habilitado para macros” Si no te acuerdas de hacerlo no podrás ejecutar el código… |
|  | Cada vez que abras un fichero de Excel habilitado para macros, el Excel te preguntará si quieras habilitar las macros, acuérdate de hacerlo. Si no contestas, el Excel entenderá que quieres que permanezcan deshabilitadas. |
|  | Para programar, ve a la pestaña de “desarrollador” y pincha en el botón “Visual Basic”  Si no tienes la pestaña de desarrollador habilítala siguiendo estos pasos:  Excel 2007 🡪 botón de Office 🡪 Opciones de Excel 🡪 más frecuentes 🡪 mostrar ficha programador 🡪 aceptar.  Excel 2010, 2013, 2016 y 2019 🡪 pestaña archivo 🡪 opciones 🡪 personalizar cinta de opciones 🡪 activar casilla Programador. |
|  | Para poder empezar a programar ve a Insertar 🡪 Módulo.  El módulo es dónde escribirás el código del programa.  Y ya podemos empezar… |

# Dar nombre a un programa

Sub nombre\_programa()

‘Todo lo que esté entre sub y end sub es un programa, y será lo que se ejecute.

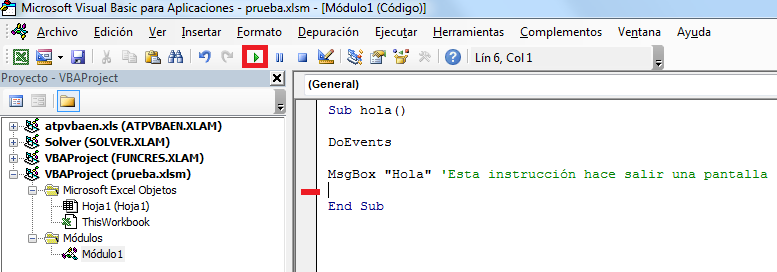
‘ Este apostrofe permite escribir comentarios en el código. Todo lo que esté a su derecha no se ejecutará, al considerarse un comentario

Doevents ’Esta instrucción le indica al ordenador que, mientras se ejecute el código esté atento a otros eventos. El poner esta instrucción nos permitirá, en caso de que se bloquee nuestro código, pararlo, sin tener que reiniciar el ordenador. Ponedla siempre al inicio de cada programa.

End sub

# Ejecutar un programa

Si estamos dentro de la pestaña de programación, ponemos el curso dentro del programa que queremos ejecutar (esto es muy importante) y pinchamos en el botón de “play”



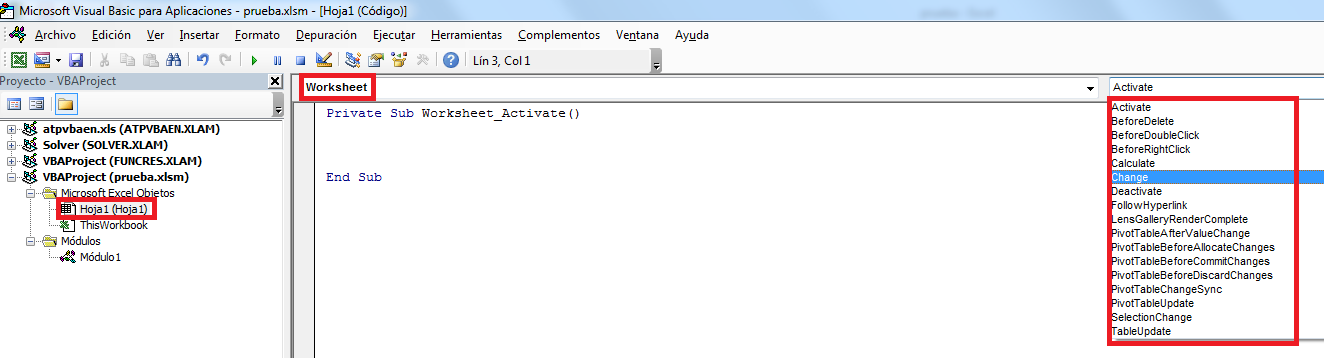
Si queremos ejecutar un programa desde Excel, podemos insertar un botón (desarrollador 🡪 Insertar 🡪 Elegimos el botón y lo pintamos) y asignar a ese botón el programa que queremos que se ejecute. En este caso, el programa hola.

Si queremos cambiar el programa que ejecuta el botón, seleccionamos el botón y, con el botón izquierdo del ratón, elegimos la opción “Asignar Macro”

|  |  |
| --- | --- |
|  |  |

Una manera más avanzada de ejecutar un programa sería utilizar eventos. Estos son “sucesos” que, al ocurrir, ejecutan el programa. Para ello, seleccionamos la hoja en la que queremos que suceda el evento y elegimos, de la lista de eventos, el que queremos que active el código.

El evento Activate, ejecutaría automáticamente el código cuando el usuario seleccionase la hoja de Excel.



Ejemplo: Queremos que el programa se ejecute únicamente cuando cambie algún valor del rango “A1:C10”

Private Sub Worksheet\_Change(ByVal Target As Range)

Dim KeyCells As Range

'La variable KeyCells contiene las celdas que provocan que el código se ejecute.

Set KeyCells = Range("A1:C10")

If Not Application.Intersect(KeyCells, Target) Is Nothing Then

MsgBox "La celda " & Target.Address & " ha cambiado."

End If

End Sub

Si quisiera escribir mucha información en la pantalla, el modo más rápido de hacerlo es mediante un rango.

Dim matriz\_invisible(1 to 1000, 1 to 1000) as string

Dim matriz\_visible as range

Sub escribir\_rapido()

Set matriz\_visible = worksheets(“Hoja1”).range(“matriz”)

For fila = 1 to 1000

For columna = 1 to 1000

matriz\_invisible(fila, columna) = “Hola”

Next

Next

matriz\_visible = matriz\_invisible

End sub

¿Tengo que recordar esto? 🡪 El código NO, el que existe esta posibilidad, sí.

# Declarar variables

Las variables son “cajas” en donde podemos almacenar información con la que trabajar.

Las variables hay que declararlas (darlas de alta), se pueden declarar en cualquier punto del programa, pero lo mejor es declararlas antes de que empiece el programa, antes de “Sub nombre\_programa()”

Una variable que se declarase dentro de un programa serviría únicamente para ese programa, una variable que se declara fuera de los programas se puede utilizar (consultar), por todos los programas que escribas dentro de un mismo módulo.

¿Cómo se declara una variable? 🡪 Dim nombre\_variable as variant

Dim es como decir “declaro que existe”

Nombre\_variable es como quieres llamarla, puedes llamarla como quieras…

As variant indica al programa qué tipo de contenido vas a meter en la variable.

Para empezar a programar el concepto variant es muy útil ya que no le indicamos qué tipo de información vamos a meter, trasladando el problema al ordenador. El ordenador reservará un espacio de memoria para cada variable, cuanto más precisos seamos menos espacio consumiremos y más eficiente será el programa.

Un programador amateur usará la clase variant con frecuencia ya que no tiene que pensar qué tipo de variable va a ser (pero ralentiza al ordenador). Un programador un poco más avanzado querrá especificar qué tipo de variable es.

As variant ‘Cualquier cosa

As integer ‘Números enteros

As double ‘Números con decimales

As Boolean ‘True o False

As string ‘Una cadena de texto.

**Trabajar con variables dentro de un programa:**

Variable = sheets(“Hoja1”).cells(1,1).value

Aquí estamos diciendo que queremos meter el contenido de la celda (fila 1, columna 1) en la variable. El orden de (fila, columna) es vital recordarlo.

Si lo hiciésemos al contrario.

sheets(“Hoja1”).cells(1,1).value = Variable

Le estaríamos diciendo al programa que nos ponga en la celda (fila 1, columna 1) lo que valga la variable en ese momento.

Lo que está a la izquierda de la igualdad es en donde metemos lo que está a la derecha de la igualdad.

**Se puede introducir valor en una variable de diversas maneras:**

a=1500 Aquí estamos indicando que la variable vale una cantidad determinada. Pero si la cantidad cambiase deberíamos recordar dónde asignamos su valor en el código. Es la manera menos eficiente de hacerlo.

a=b Aquí decimos que la variable a es igual a la variable b. Si b valiese 100, a pasaría a valer también 100.

a=inputbox(“Introduce el valor de la variable”) Aquí le pedimos al usuario que introduzca el valor de la variable. Cuando se ejecute esta línea de código, aparecerá un pop up que le pedirá al usuario que introduzca el valor de la variable.

a = sheets(“Hoja1”).cells(1,1).value Le indicamos que la variable a tome su valor de una hoja y una celda en concreto. Pero si se moviese la celda del Excel, la referencia no se actualizaría sola en el código. Cuidado con esto.

a=Range(“Dato”).value Indicamos que la variable a toma su valor de un rango de excel. Para dar nombre a un rango seleccionamos un rango de datos en Excel y, con el botón derecho del ratón, asignado un nombre al rango, en este caso lo llamamos Dato. Si alguien moviese la celda de sitio el código sí se actualizaría solo. Es la manera más eficiente de dar valor a una variable.

# Números aleatorios

Randomize

Numero = rnd()

Para sacar un número aleatorio entre 0 y 1 se utiliza la expresión rnd(). En este caso estamos metiendo un número aleatorio en la variable número.

Randomize debe utilizarse antes de rnd. Esta instrucción “garantiza” que el número que sacamos es aleatorio.

¿Cómo puedo sacar un número aleatorio entre 25 y 50?

Randomize

Numero = Int((lim\_superior – lim\_inferior +1) \* rnd + lim\_inferior)

¿Tengo que recordar esto? 🡪 NO. Lo único que tienes que recordar es que rnd sirve para sacar número aleatorios. Si tienes que usar este código en el futuro acude a internet. Escribe en google “vba aleatorio entre” y encontrarás la fórmula anterior.

# Estructuras condicionales

**Estructura condicional IF**

Ejecuta un código u otro en función de si se cumple o no una condición. Esta estructura es muy útil a la hora de programar, ya que diriges el código en una u otra dirección.

If variable > 5 then

‘Si es cierto se ejecutará lo que esté aquí.

Msgbox “Es cierto”

Else

‘Si no es cierto se ejecutará lo que esté aquí.

Msgbox “No es cierto”

End if

If Sheets("Hoja1").Cells(1, 1).Value > 5 Then

MsgBox "Mayor de 5"

ElseIf Sheets("Hoja1").Cells(1, 1).Value < 2 Then

MsgBox "Menor de 2"

Else

MsgBox "Otro"

End If

Con Elseif puedes escribir condiciones adicionales (tantas como quieras) o combinarlas con “and” u “or” 🡪 if edad > 65 and peso > 90 then

**Estructura condicional Select Case**

Hay ocasiones en que queremos evaluar muchas condiciones y ejecutar un código y otro en función del valor de una variable. En estos casos, escribir un IF sería muy largo, por lo que utilizamos un select case.

|  |  |
| --- | --- |
| Queremos que se ejecute un código u otro en función del valor de la variable índice.  Select case indice  Case “Dax”  Msgbox “Estamos trabajando con el Dax”  Case “Bovespa”  Msgbox “Estamos trabajando con Bovespa”  Case else  Msgbox “Otro índice”  End select | Queremos que se ejecute un código u otro en función del número de activos con los que trabajemos.  Select case numactivos  Case 1 to 5  ……  Case 6,7,8  ……  Case 9 to 10  ……  Case else  …….  End select |

# Estructuras reiterativas

**Estructura reiterativa FOR**

Utilizamos la estructura for para repetir un número de veces determinado (y conocido de antemano) un código en concreto.

For dia = 1 to 100

El código que esté aquí metido se ejecutará 100 veces.

Msgbox “Hola”

next

Me saldrían 100 pantallas de Hola (no lo hagáis)

Ojo, For tiene un límite de poco más de 42.000 repeticiones.

**Estructura reiterativa DO**

Utilizamos la estructura DO para repetir un número de veces indeterminado (NO conocido de antemano) un código en concreto. La repetición parará cuando se cumpla la condición que le indiquemos.

numero = 0

Do until numero > 50

numero = numero +1

‘El código que esté aquí metido se ejecutará 50 veces.

loop

numero es una variable que estamos utilizando como “contador”. La manera de acumular datos en una variable es indicar que el valor de esa variable es lo que ya contenía + 1 🡪 numero = numero +1

# Estructuras anidadas

Las estructuras anteriores son las básicas. Tenemos que comprenderlas y dominarlas.

Lo interesante es que podemos anidar unas estructuras dentro de otras. Por ejemplo, podemos poner un for dentro de otro for.

For fila =1 to 10

For columna = 1 to 10

Sheets(“Hoja1”).cells(fila,columna).value = “Hola”

Next

Next

En este caso el programa pondrá Hola en las 10 primeras filas y columnas.

Ahora bien, ¿qué pondrá primero, las filas o las columnas?

Se pueden anidar estructuras distintas. Por ejemplo

For fila = 1 to 100

If fila = 50 then

Sheets(“Hoja1”).cells(fila,1).value = “Es cierto”

Else

Sheets(“Hoja1”).cells(fila,1).value = “Hola”

End if

next

Puedes mezclar y combinar como quieras los conceptos que hemos visto: Números aleatorios, estructuras IF, SELECT CASE, FOR y DO. Estas estructuras debes dominarlas.

# Llamar a otro programa

Los programas pueden llamarse entre sí mediante la instrucción Call. Lo normal es tener un programa “padre” que llame a todos los demás programas. Este procedimiento es muy útil a la hora de encontrar y depurar errores.



**Llamar a un programa pasándole información**

Cuando llamamos a un programa podemos hacerlo facilitándole información para su ejecución o no. Si las variables que necesita el segundo programa están declaradas al principio del todo, lo más probable es que no necesitemos indicar nada, tan solo llamar al programa 2.

Sub programa1

Call programa 2

End sub

No obstante, habrá casos en lo que queramos especificar información de entrada al programa hijo. Veamos un par de ejemplos:

|  |  |
| --- | --- |
| Dim area\_calculada as double  Sub programa\_padre()  Doevents  Call area(20,10)  Msgbox area\_calculada  End sub  Sub area (base as double, altura as double)  Doevents  area\_calculada = base \* altura  end sub | Sub programa\_padre()  Doevents  area\_calculada = calcular\_area(10,15)  msgbox area\_calculada  end sub  Function calcular\_area(base as double, altura as double)  Doevents  calcular\_area = base \* altura  end function |

**Recursividad**

Llegado el caso, un programa puede llamarse a sí mismo.

¿Y esto para qué vale? 🡪 Si el código que se está ejecutando arroja un resultado que no queremos, podemos volver a iniciar el programa haciendo que se llame a sí mismo.

Ejemplo: Estamos haciendo un programa que saca los números de la combinación ganadora de la lotería. Imaginaros que el programa saca, aleatoriamente, un número que ya ha salido. Dado que no puede salir un mismo número dos veces, le indicamos al programa que se llame a sí mismo para reiniciarse.

Sub loteria()

Call sacar\_numeros()

End sub

Sub sacar\_numeros()

If “numero ya ha salido” then

Call sacar\_numeros

End sub

End if

End sub

# Trabajar con matrices

Declarar una matriz

Se puede declarar con el resto de variables o en mitad del código.

Dim matriz(5) as double

Dim matriz (1 to 5) as double

Dim matriz() as double

La primera matriz tiene 6 “huecos”, de 0 a 5, en donde podremos guardar valores.

La segunda matriz tiene 5 “huecos”, de 1 a 5.

La tercera matriz la declaramos, indicando tan solo el tipo de datos que va a contener, pero no el tamaño de la misma. Esto puede ser porque no lo conozcamos, o porque el tamaño vaya a cambiar durante la ejecución del programa.

Ojo, solo se puede tener 1 tipo de variable dentro de la matriz (string, integer, double, boolean…). Es decir, en una matriz de dos dimensiones, (lo que sería una tabla en Excel), no puede contener una “fila” de datos numéricos y que la siguiente “fila” tenga datos alfabéticos. No puede tener una fila de datos que no acepten decimales y que la siguiente sí los admita.

Añadir valores a una matriz

Dim matriz(1 to 10, 1 to 5) as string

For fila=1 to 10

For a=columna to 5

matriz(fila,columna) = Sheets(“Hoja1”).cells(fila,columna).value

Next

Next

Recuperar valores de una matriz

For fila=1 to 10

For a=columna to 5

Sheets(“Hoja1”).cells(fila,columna).value = matriz(fila,columna)

Next

Next

Redimensionar una matriz (matrices de tamaños variables)

Para poder redimensionar una matriz, debemos haber declarado esta sin indicar previamente su tamaño. Es decir 🡪 Dim matriz() as double

1. Sin conservar los datos

Basta con redimensionar la matriz. Cada vez que una matriz es redimensionada, esta pierde todos los datos que tuviera guardados. Redimensionarla es, por lo tanto, una buena manera de borrar los datos de una matriz antes de proceder a introducir datos en ella. Si no lo hiciéramos y usásemos la matriz completa para realizar algún cálculo, es posible que, de no haberla llenado al completo, permaneciesen datos antiguos que alterasen el valor final del cálculo. Este es un fallo muy típico.

Redim matriz(1 to 5)

Fijaros que no es necesario indicar qué tipo de valores contiene la matriz porque ya lo hemos especificado a la hora de declararla por primera vez. Esta naturaleza no es cambiable.

1. Conservando los datos (añadir un valor)

Imaginemos que declaramos una matriz al principio del todo:

dim matriz() as double

En un momento dado, dimensionamos la matriz por primera vez:

Redim matriz(1 to 5)

Un poco más adelante, deseamos redimensionar la matriz de nuevo para añadir un nuevo valor, conservando los anteriores:

ReDim Preserve matriz(1 To 6)

Si queremos automatizar el incremento de la matriz, otra manera de indicar al código que queremos añadir un valor más es utilizando el término UBound.

ReDim Preserve matriz(1 To UBound(matriz) + 1)

UBound(matriz) es como decir “tamaño actual de la matriz”. Si el tamaño actual es 5, decir UBound(matriz) es como decir “5”.

Ojo, solo se puede redimensionar la última dimensión de la matriz (en el caso de que tenga más de una). Imaginemos que tenemos una matriz de 10 activos en donde queremos ir guardando los valores de cierre diarios.

Declaro la matriz

dim cartera() as double

Redimensiono por primera vez la cartera (para guardar el valor de cierre de hoy)

Redim cartera (1 to 10, 1 to 1)

Pasa un día. Vuelvo a redimensionar la cartera para guardar el cierre de hoy.

ReDim Preserve cartera(1 to 10, 1 To 2)

Ojo, el siguiente código NO funciona:

ReDim Preserve cartera(1 to 10, 1 To UBound(cartera) + 1)

Ubound(cartera) indica el tamaño de la primera dimensión de la matriz y no de las siguientes.

Para poder redimensionar a partir de la segunda dimensión, lo mejor es indicar el tamaño usando una variable.

For tamaño = 1 to 50

ReDim Preserve cartera(1 to 10, 1 To tamaño)

Next

Lo que NO puedo hacer es redimensionar la primera dimensión de la matriz, o las dos a la vez.

ReDim Preserve cartera(1 to 5, 1 To 2)

ReDim Preserve cartera(1 to 5, 1 To 5)

Quitar un valor

Si el valor que queremos eliminar es el último de la matriz, basta con redimensionarla con un valor menos (ojo, solo se puede redimensionar la última dimensión).

ReDim Preserve matriz(1 To UBound(matriz) - 1)

En el caso de que no sea el último valor es más complejo. No podemos extraer una posición concreta de la matriz y eliminarla.

Al no ser posible eliminar una posición concreta, debemos copiar y pegar los valores posteriores al valor que queremos quitar, con la finalidad de dejar el “valor inservible” al final de la matriz.

Imaginemos que tenemos una matriz con nombres de ciudades y queremos eliminar “Almería”

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Albacete |  | Albacete |  | Albacete |  | Albacete |  | Albacete |
| Alicante |  | Alicante |  | Alicante |  | Avila |  | Avila |
| Almería |  | Avila |  | Avila |  | Badajoz |  | Badajoz |
| Avila |  | Badajoz |  | Badajoz |  | Barcelona |  | Barcelona |
| Badajoz |  | Barcelona |  | Barcelona |  | Barcelona |  |  |
| Barcelona |  | Barcelona |  |  |  |  |  |  |

En la tabla anterior podemos ver el proceso de eliminación de las ciudades de Almería y Alicante (extracción, copia de valores y redimensionamiento de la matriz).

Así conseguimos que el valor que ya no vale se quede siempre el último. Hecho esto, podemos redimensionar la matriz eliminando el último.

For ciudad\_a\_eliminar = siguiente\_ciudad To UBound(nombre\_ciudades)

If ciudad\_a\_eliminar = UBound(nombre\_ciudades) Then

'Si estamos en la última posición de la matriz no hacemos nada

Else

nombre\_ciudades(ciudad\_a\_eliminar) = nombre\_ciudades(ciudad\_a\_eliminar + 1)

End If

Next

ReDim Preserve nombre\_ciudades(1 To UBound(nombre\_ciudades) - 1) 'Redimensionamos la matriz quitando el último valor

Usar una matriz completa en un cálculo.

Podemos usar toda la información recogida en una matriz para realizar un cálculo, sin necesidad de recorrerla usando un for. Por ejemplo, tenemos una matriz que contiene los flujos de efectivo de un proyecto y queremos calcular la TIR.

TIR = WorksheetFunction.IRR(Flujos)

O queremos calcular la covarianza existente entre dos activos.

Covarianza = WorksheetFunction.Covar(rent\_merc, rent\_act\_esp)

# Diseñamos un nuevo tipo de variable, (una clase).

Public Type activo 'definimos un nuevo tipo de datos para que refleje los datos de las empresas

porcentaje\_invertido As Double

nombreEmpresa As String

dineroasignado As Double

fecha\_compra As Date

divisa\_compra As Double

dineroasignado\_divisa As Double

pc As Double 'precio de compra

numeroacciones As Double

fecha\_venta As Date

pv As Double ' precio de venta

resultado\_en\_divisa As Double

divisa\_venta As Double

resultado\_en\_euros As Double

comision As Double

resultado\_final As Double

End Type

Dim cartera() As activo

Lo que estamos haciendo es generar un nuevo tipo de dato. El equivalente en Excel sería escribir, en la misma celda lo siguiente:

|  |
| --- |
| Invertir el 23% del dinero disponible  Ebay  23.000 €  1/1/2003  1 € equivale a 1,35 $  31.050 $  Precio de compra 15$  Compramos 2070 acciones… etc |

Si escribiésemos todo lo anterior en una misma celda no podríamos trabajar con ella. Sin embargo, en programación sí que podemos trabajar con este tipo de datos.

Lo siguiente que hacemos es generar una matriz de datos llamada “cartera”. Ahí podremos guardar cada uno de los datos compuestos, operar con esos datos y recuperarlos más adelante.

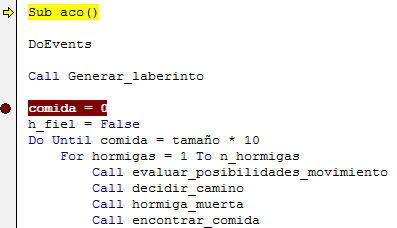
**Dimensionar la cartera**: Dim cartera (1 to 5) as activo

**Asignar valor**: cartera(3).divisa\_compra = Sheets("Hoja1"). Cells(1,1).Value

**Recuperar valor**: Sheets("Hoja1").Cells(1,1).Value = cartera(3).porcentaje\_invertido

# El programa me da un error ¿Qué hago?

Cuando el programa da un error nos pregunta si queremos depurarlo, señalando la línea en la que se ha producido el error en amarillo.



Para encontrar el error nos ponemos encima de la línea amarilla. Al hacerlo nos saldrá un mensaje dándonos alguna pista sobre la naturaleza del error.

Si nos ponemos con el cursor encima de las variables ya ejecutadas, nos aparecerá una pequeña pantalla que nos indica su valor. Lo más probable es que una de esas variables contenga el error y lo identifiquemos fácilmente. Alguna variable estará vacía, o habremos intentado meter un número en una variable que sólo admite texto….

Si no identificamos el error con rapidez, nos quedan dos opciones:

Ejecutar el código instrucción a instrucción (línea a línea) 🡪F8. De esta manera veremos en qué línea asignamos valor a las variables, cómo salta el código de una línea a otra… en resumen, nos permite ver, paso a paso, qué es lo que está haciendo el código.

Poner puntos de pausa/control 🡪 Botón izquierdo del ratón al inicio de la línea. Al hacer esto, ponemos un punto rojo al código, el cual realizará una pausa cada vez que se ejecute esa línea. Durante la pausa podremos ponernos encima de cada variable para ver qué contienen y comprobar así el buen curso del código.

# No sé cómo programar una cosa ¿Qué hago?

* La ayuda del propio Excel está en español. Dentro de visual basic escribe tu pregunta en el buscador de ayuda.
* Si sabes hacer lo que quieres en Excel pero no programarlo graba una macro. Pestaña programador 🡪 grabar macro. Esta acción convertirá en código todo lo que hagas hasta que detengas la grabación. Esta opción es muy útil si estás empezando a programar.

Ojo, el código de la macro se grabará en un módulo distinto (módulo2). Selecciona este módulo para acceder al código, léelo, mira a ver qué código es el que te interesa y pégalo en tu módulo principal.

* Si no consigues resolver tu duda, ten presente que hay muchísimo código ya resuelto en internet. La solución es tan sencilla como escribir en Google VBA + tu duda, por ejemplo “vba aleatorio entre”. Eso sí, trata siempre de entender, en la medida de lo posible, lo que estas copiando. Más que nada, para saber hacerlo en el futuro.

**Primer programa:**

**Problema a resolver:** Queremos sacar 100 combinaciones de 5 números de lotería, los cuales pueden ser entre 1 y 50.

**Pasos a dar**

1. Sacamos un número aleatorio entre 1 y 50
2. Sacamos una combinación de 5 números aleatorios
3. Comprobamos que los números no se repiten. Si lo hicieran, tendríamos que sacar un número nuevo.
4. Sacamos 100 combinaciones de números y las ponemos en Excel



**Código paso a paso**

Dim limite\_superior As Variant

Dim limite\_inferior As Variant

Dim bola As Variant 'Contiene el número aleatorio que sale.

Sub sacar\_bola()

DoEvents

limite\_superior = 50

limite\_inferior = 1

Randomize

bola = Int((limite\_superior - limite\_inferior + 1) \* Rnd + limite\_inferior)

End Sub

**Sacamos una combinación de 5 números aleatorios**

Dim bola As Variant 'Contiene el número aleatorio que sale.

Dim combinacion\_ganadora(1 To 5) As Variant ’Matriz de 5 datos (como 5 celdas de Excel)

Dim nbola As Variant ' Indica el número de bola que estamos sacando

Dim ncombinacion As Variant ' cuantas bolas tiene la combinación

Dim limite\_superior As Variant

Dim limite\_inferior As Variant

Sub combinacion()

DoEvents

ncombinacion = 5

For nbola = 1 To ncombinacion

Call sacar\_bola

Next

End Sub

Sub sacar\_bola() Preguntar 🡪 ¿Porqué quiero hacer dos programas por separado?

DoEvents

limite\_superior = 50

limite\_inferior = 1

Randomize

bola = Int((limite\_superior - limite\_inferior + 1) \* Rnd + limite\_inferior)

combinacion\_ganadora(nbola) = bola

Sheets("Hoja1").Cells(1, nbola).Value = bola

End Sub

**Comprobamos que los números no se repiten**

Dim bola As Variant 'Contiene el número aleatorio que sale.

Dim combinacion\_ganadora(1 To 5) As Variant

Dim nbola As Variant ' Indica el número de bola que estamos sacando

Dim ncombinacion As Variant ' cuantas bolas tiene la combinación

Dim limite\_superior As Variant

Dim limite\_inferior As Variant

Dim comprobar\_bola As Variant 'para recorrer las bolas que ya han salido.

Dim ncombinaciones As Variant 'numero combinaciones

Sub combinacion()

DoEvents

ncombinacion = 5

For nbola = 1 To ncombinacion

Call sacar\_bola

Next

End Sub

Sub sacar\_bola()

DoEvents

limite\_superior = 50

limite\_inferior = 1

Randomize

bola = Int((limite\_superior - limite\_inferior + 1) \* Rnd + limite\_inferior)

For comprobar\_bola = 1 To nbola

If bola = combinacion\_ganadora(comprobar\_bola) Then

Call sacar\_bola

Exit sub

End If

Next

combinacion\_ganadora(nbola) = bola

Sheets("Hoja1").Cells(1, nbola).Value = bola

End Sub

¿Por qué el primer número cuando se compara con el primer número no dice que son iguales y entra en un bucle infinito?

Porque se asigna el número a la combinación después de hacer la comprobación, por lo que el primer número se compara con un hueco en blanco (primera posición de la matriz)

**Sacamos 100 combinaciones de números y las ponemos en excel**

Dim bola As Variant 'Contiene el número aleatorio que sale.

Dim combinacion\_ganadora(1 To 5) As Variant

Dim nbola As Variant ' Indica el número de bola que estamos sacando

Dim ncombinacion As Variant ' cuantas bolas tiene la combinación

Dim limite\_superior As Variant

Dim limite\_inferior As Variant

Dim comprobar\_bola As Variant 'para recorrer las bolas que ya han salido.

Dim ncombinaciones As Variant 'numero combinaciones

Sub combinacion()

DoEvents

ncombinacion = 5

For ncombinaciones = 1 To 100

For nbola = 1 To ncombinacion

Call sacar\_bola

Next

Next

End Sub

Sub sacar\_bola()

DoEvents

limite\_superior = 50

limite\_inferior = 1

Randomize

bola = Int((limite\_superior - limite\_inferior + 1) \* Rnd + limite\_inferior)

'Tenemos que comprobar que este número no ha salido ya

For comprobar\_bola = 1 To nbola

If bola = combinacion\_ganadora(comprobar\_bola) Then

Call sacar\_bola

Exit sub

End If

Next

combinacion\_ganadora(nbola) = bola

Sheets("Hoja1").Cells(ncombinaciones, nbola).Value = bola

End Sub