MACROS DE EXCEL, PARA PERSONAS OCUPADAS

ING. LUIS CRUZ

Automatiza tu trabajo en Excel y déjalo hacer el trabajo duro por ti.

Macros de Excel: Para personas ocupadas

Aprende en poco tiempo a crear macros que hagan el trabajo duro por ti.

Luis Alberto Cruz Orellana

Este libro está a la venta en http://leanpub.com/libro-macros-excel

Esta versión se publicó en 2019-11-21

Este libro esta dedicado a Dios, a mi esposa Patty y mis hijos Luis y Gabriela. Ellos son la razón por la que me esfuerzo cada dia en ser mejor.

Índice general

Tus primeras macros

En este capítulo voy a introducirte al mundo de las macros, aprenderás que es una macro, por que necesitas usarlas, como puedes usarlas y como crear macros sencillas.

Que es una macro

Una macro es una secuencia de comandos o pasos para realizar una tarea. En nuestro caso una macro de Excel no es más que una serie de acciones que puedes realizar en Excel.

Por ejemplo, si te envían una hoja de cálculo y te piden preparar un reporte con esos datos, entonces tú sigues una serie de paso o acciones en Excel para preparar ese reporte.

Si los pasos que sigues son siempre los mismos o muy similares, entonces puedes crear una macro de Excel para que haga ese trabajo por ti.

Como pueden ayudarte las macros en tu trabajo

La mayor limitante de todos nosotros es nuestro tiempo, ese es tu recurso más valioso. Puedes usar tu tiempo para ver una película, leer un libro, pasar tiempo con tus seres queridos, trabajando o en lo que tú desees.

Por su puesto, todos necesitamos y debemos trabajar para poder sobrevivir, pero con la ayuda de las computadoras y las macros de Excel, puedes hacer que la computadora haga gran parte del trabajo en tu lugar, y así poder ganar un poco de tiempo extra para hacer algo que te guste más.

Por otra parte, también puede ayudarte a ser mucho más productivo en tu trabajo y sobresalir de entre todos tus compañeros.

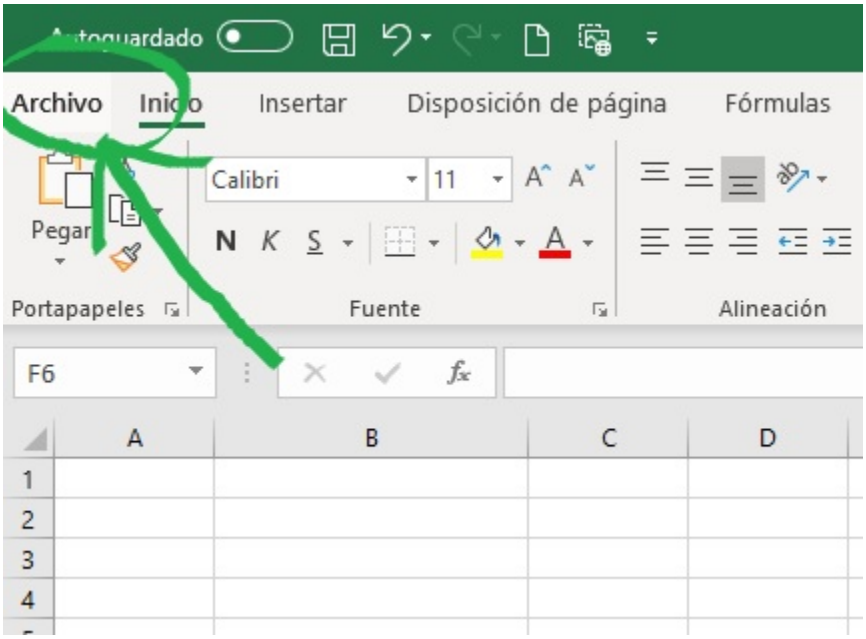
Como se usan las macros

Ahora vas a comenzar a trabajar con macros, para el resto del capítulo vamos a tomar un caso hipotético: Imagina que tienes una hoja de Excel que frecuentemente debes compartir con otras personas, no te afecta que las personas vean los datos, pero no quieres que conozcan las fórmulas que usas para calcularlas (es algo secreto, como la fórmula de la Coca-Cola o algo así).

Entonces cada vez que debes enviar la hoja de cálculo a alguien, pasas todas las fórmulas a valores absolutos. Pero para ahorrarte ese trabajo, has creado una macro.

Como aún no hemos visto como crear macros, puedes descargar la macro de la que hablo haciendo clic aquí[1]

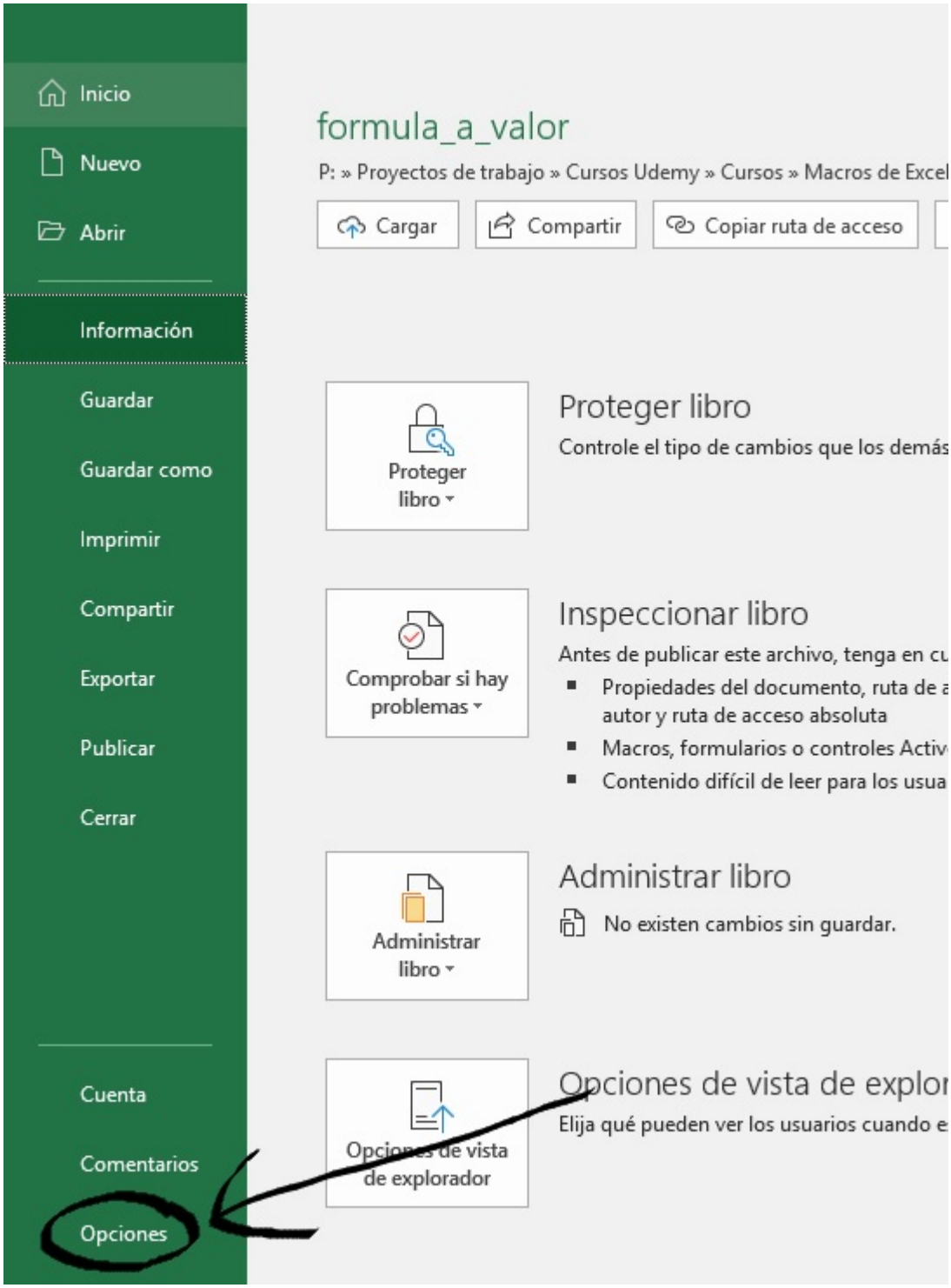
Ahora vamos a activar un menú secreto de Excel, primero haces clic en el menú *Archivo*



Menú Archivo

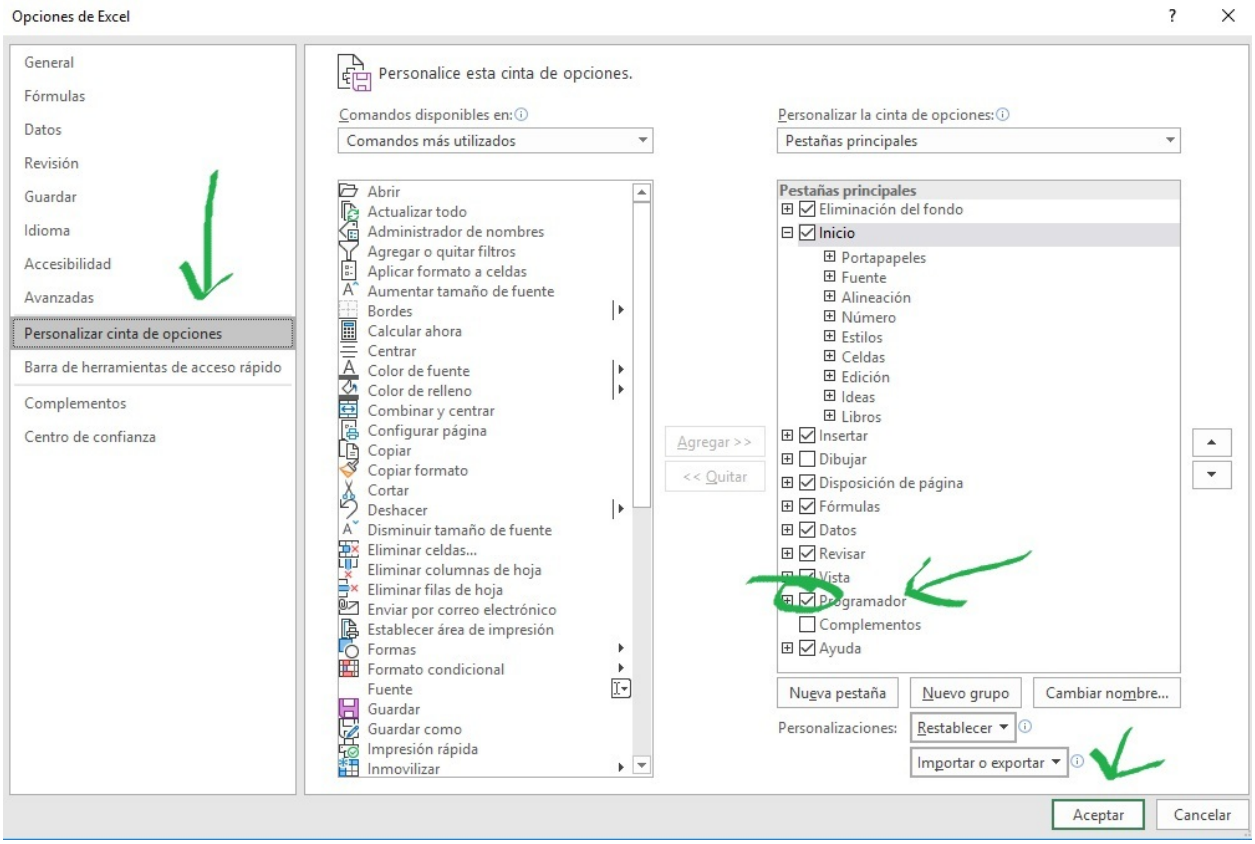
Luego seleccionas *Opciones*

[1] https://my.pcloud.com/publink/show?code=XZN9jd7Z1hwPMri9O681FVEH44EWBjO6qVAV



Opciones

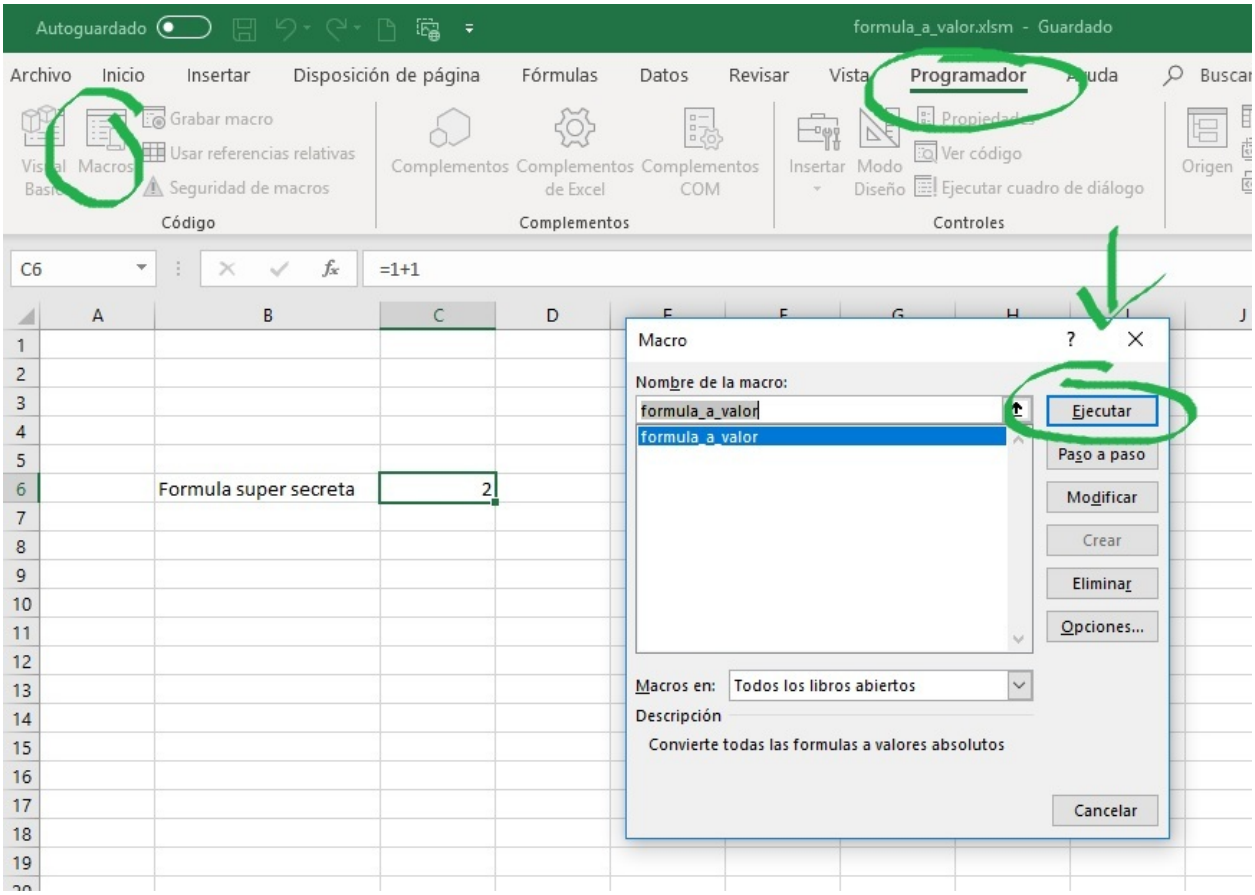
Y seleccionas *Personalizar cinta de opciones* y marcas la casilla de selección que dice *Programador*, finalmente haces clic en *Aceptar* y listo, ya has activado el menú secreto.



Activar menú programador

Abre el archivo que descargaste y verás que en la celda *C6* hay una formula (es una formula muy sencilla, pero esto no es importante por ahora)

Si vas al nuevo menú *Programador*, luego presionas el botón que dice *Macros* veras que se abre una ventana, selecciona la primera macro en la lista y presiona el botón *Ejecutar*, esto hará que todas las fórmulas en esta hoja sean reemplazadas por su valor en texto, eliminado su fórmula, pero no su resultado



Ejecutar una macro

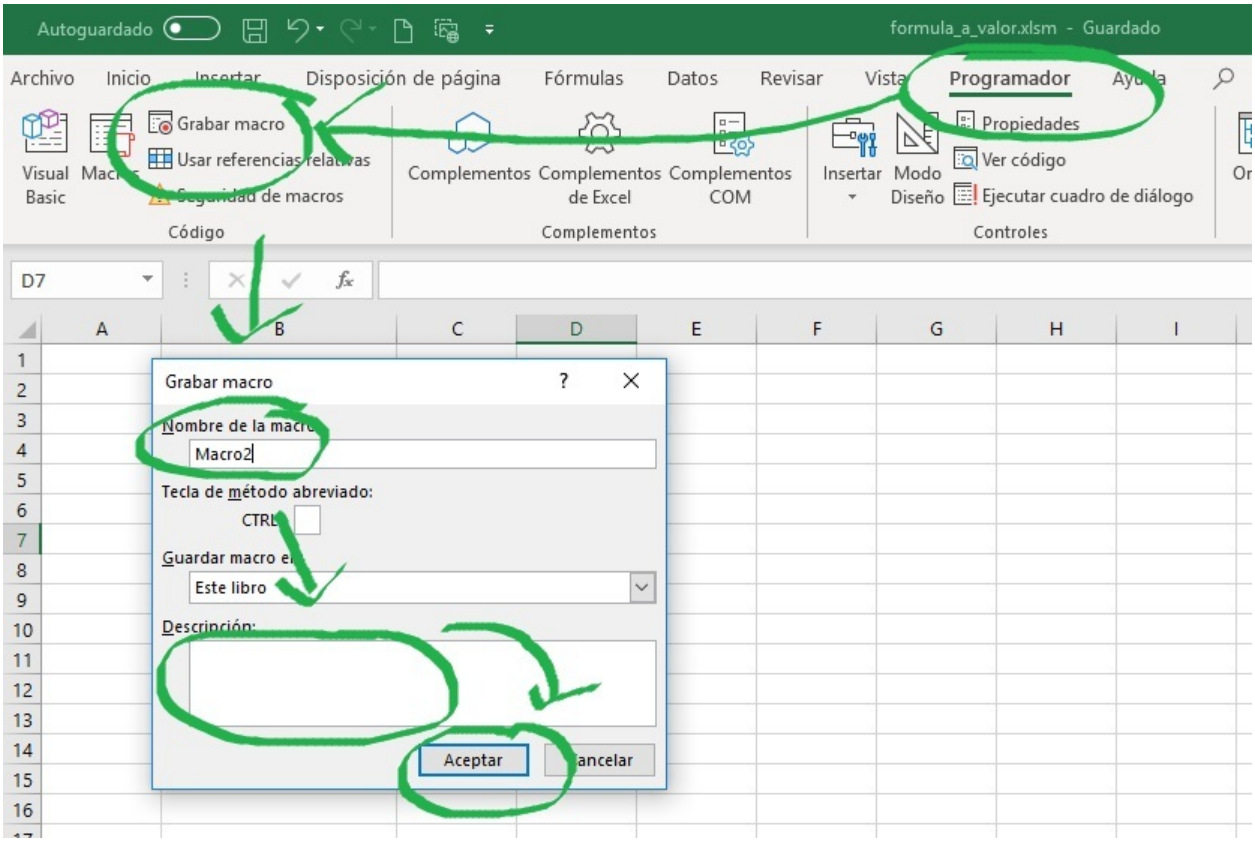
Existen otras formas para ejecutar una macro, pero por el momento vamos a utilizar esta.

Tu primer macro

Ahora que ya ejecutaste tu primera macro es hora de que crees tú primer macro. Crea un archivo nuevo de Excel y en el menú *Programador* presiona el botón *Grabar macro*

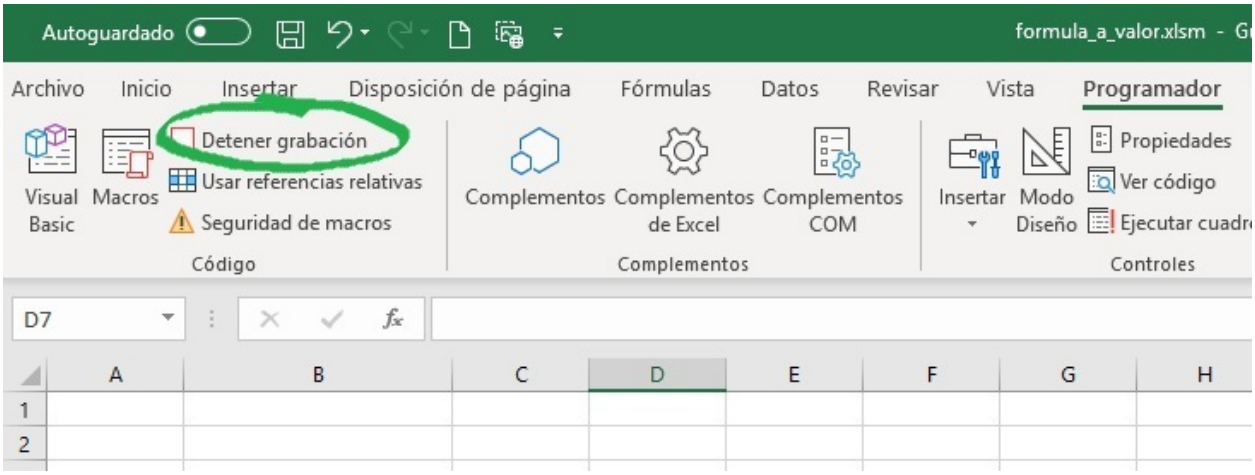
Luego veras una ventana en la que tienes que colocar un nombre para la macro, una descripción y presionar el botón *Aceptar*

> Los nombres de macros no pueden llevar espacios en blanco, pero puedes usar guiones para separar palabras. Tampoco puedes usar caracteres especiales.



Grabar macros

Ahora Excel va a grabar todo lo que hagas, por ejemplo, selecciona primera fila de la hoja de cálculo y coloca un color de fondo, puedes hacer cualquier cosa que desees. Cuando termines presiona el botón *Detener grabación*

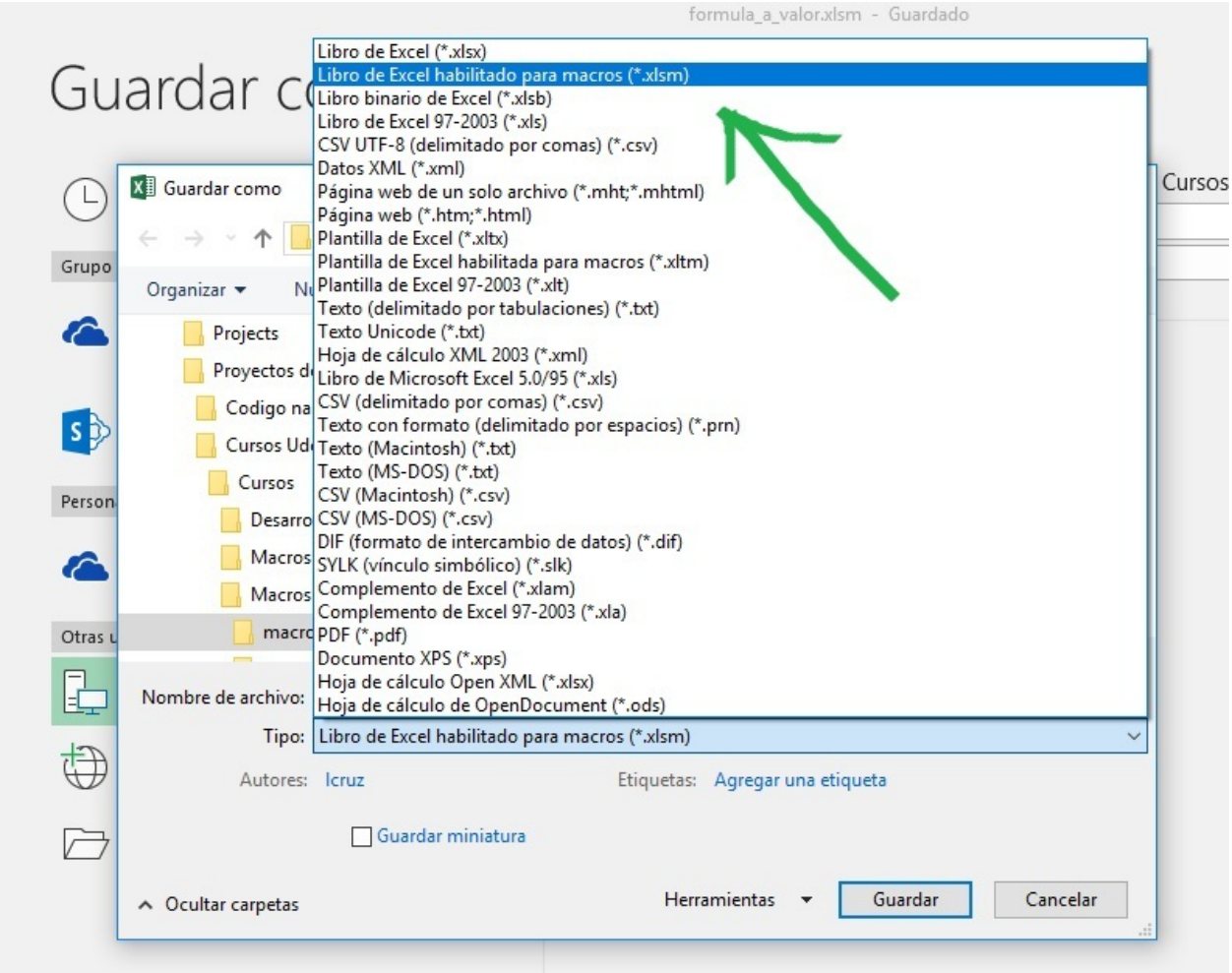


Detener grabación

Ya está, has creado tu primer macro. Puedes ejecutarla como lo hiciste con la macro que descargaste,

pero para poder ver que repite todo, puedes seleccionar una hoja nueva.

Si intentas grabar este archivo, vas a recibir una advertencia, y es que las hojas de cálculo "normales" no pueden contener macros. Para poder grabar y conservar tu macro debes de elegir el formato de archivo *Libro de Excel habilitado para macros*



Archivo con macros

> Hasta ahora las macros que crees, son parte del archivo en que estas trabajando, si el archivo no está abierto, la macro no está disponible. En otros capítulos vamos a aprender a hacer que tu macro este siempre disponible

Ejercicio

Ahora viene la parte práctica, imagina algo que has estado haciendo en Excel, y que repites una y otra vez. Intenta crear una macro que automatice el trabajo y lo haga por ti.

Tiene que ser algún muy sencillo como dar formato a algo, cambiar algún nombre, borrar columnas que no usas o algo así, recuerda que estas comenzando, luego podrás hacer cosas más difíciles.

Introducción a VBA

En este capítulo aprenderás qué es VBA, cómo usar el editor de VBA para crear macros usando código de programación y a editar macros grabadas con la grabadora de macros.

Qué es VBA

VBA significa **Visual Basic for Applications**, este es un lenguaje de programación hecho por Microsoft para automatizar cualquier cosa en su suite de oficina *Office*. Está basado en Visual Basic, por lo que sus comandos son prácticamente los mismos.

Cada opción, cada botón que presionas, cada cosa que haces en Excel Puede ser realizada mediante código de VBA, de hecho, cuando grabas una macro usando la grabadora, cada acción que haces se guarda como código de VBA.

Pero la grabadora de macros tiene sus limitaciones Cuando escribes código de VBA, no existen los límites. Si puedes hacer algo en Excel, también puedes hacerlo usando VBA.

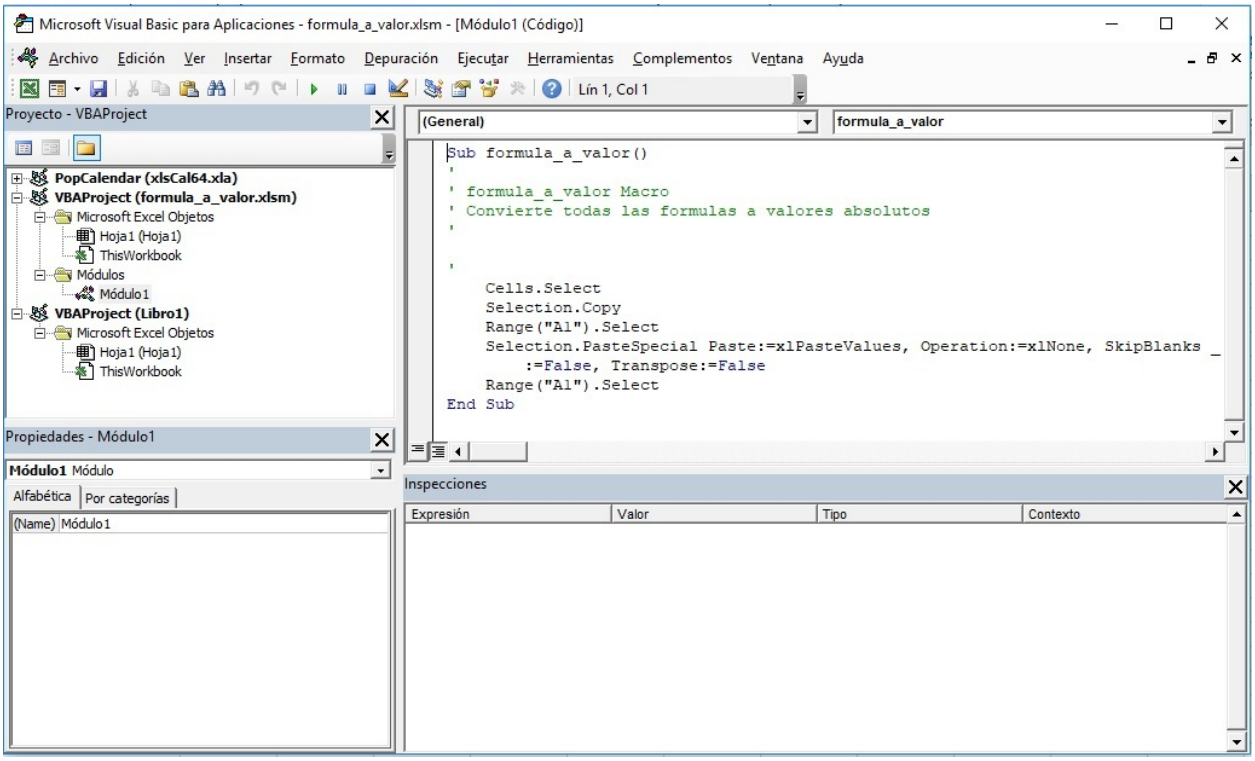
Este es un ejemplo de código de VBA:

```
Sub formula_a_valor()
' Convierte todas las formulas a valores absolutos
'
    Cells.Select
    Selection.Copy
    Range("A1").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Range("A1").Select
End Sub
```

Lo sé, se ve bastante confuso ahora, pero no te preocupes, aun si nunca has programado. Este libro está diseñado no solo para enseñarte a utilizar VBA, sino también para enseñarte a programar.

Editor de VBA

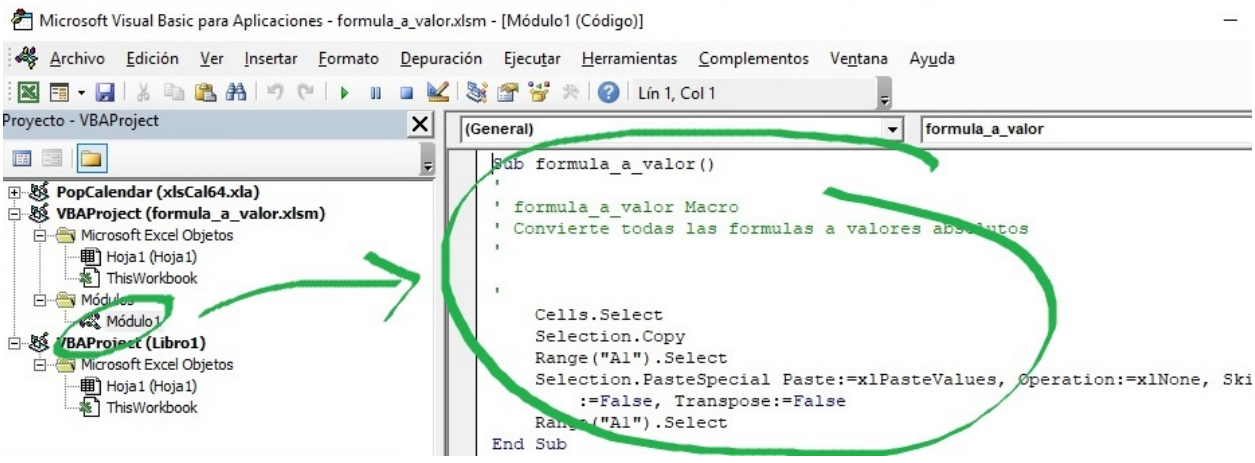
Para ingresar al editor de VBA, abre alguna hoja de Excel y presiona las teclas `ALT+F11`, al hacer esto verás una ventana como esta:



Editor de VBA

Esta está dividida en varias secciones, no te preocupes si no puedes verlas todas en este momento, esto se debe a que tu hoja de Excel aún no tiene macros.

En la parte superior izquierda verás el explorador de proyectos, cada libro de Excel o archivo se trata como un proyecto, el código de VBA ira dentro de una carpeta llamada Módulos, dentro de esta carpeta existirá uno o más archivos que a su vez contiene código de programación (macros). En la parte derecha hay una ventana, ahí es en donde se puede visualizar el código de las macros.



Ver el código de programación

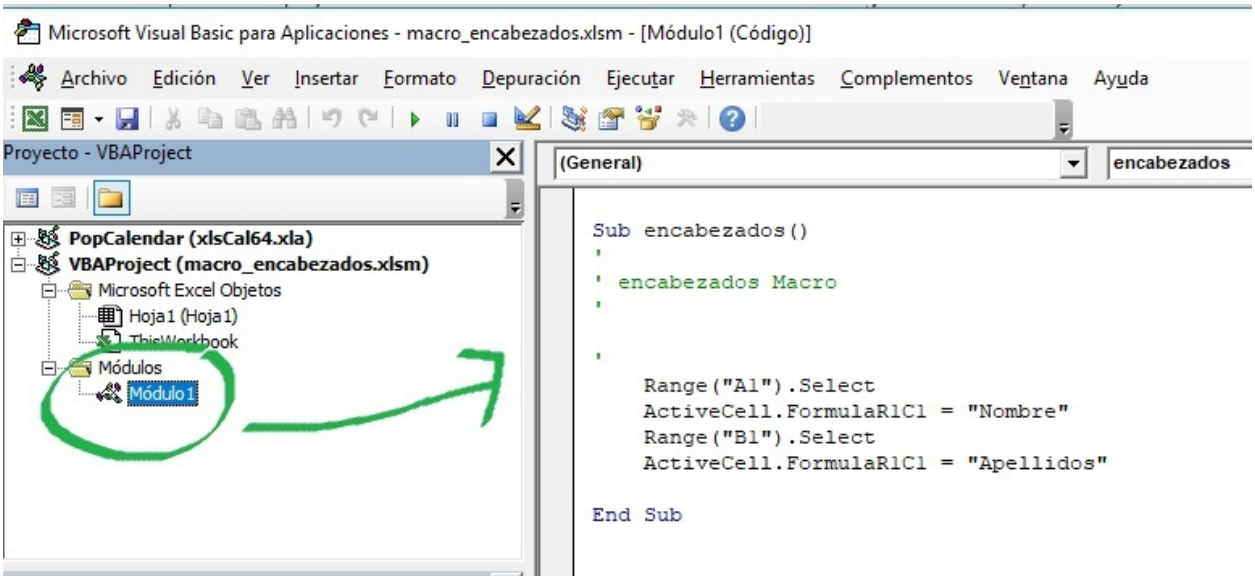
Existen varios menús y herramientas, pero los iremos viendo en detalle poco a poco para no saturarte

con información.

Editar una grabación de macros

Ahora vamos a practicar con algo sencillo, ya sabes crear macros usando la grabadora, entonces crea una macro sencilla, por ejemplo, una macro en la que cree o modifique los encabezados de una hoja de cálculo colocando en la celda `A1` la palabra *Nombre* y en la celda `B1` *Apellidos*.

Luego de grabar la macro, abres el editor de VBA (presionando las teclas Alt+F11) y posiblemente veas algo como esto:



Macro a editar

Ahora imagina que tu necesidad ha cambiado y en la columna C1 necesitas que diga *Teléfono*, entonces en lugar de grabar otra macro, solo la abres en el editor y debes hacer dos cosas

- Seleccionara la celda C1
- Escribir el texto *Teléfono*

El código para eso sería:

```
1     Range("C1").Select
2     ActiveCell.FormulaR1C1 = "Teléfono"
```

Si lo agregas a la macro, ésta deberá verse así:

```
Sub encabezados()
'
' encabezados Macro
'

'
    Range("A1").Select
    ActiveCell.FormulaR1C1 = "Nombre"
    Range("B1").Select
    ActiveCell.FormulaR1C1 = "Apellidos"
    Range("C1").Select
    ActiveCell.FormulaR1C1 = "Teléfono"

End Sub
```

Ahora solo debes de probar la macro y tendrás que ver que automáticamente rellena las celdas A1, B1 y C1 con los textos que has programado en tu macro.

Ya has modificado tu primera macro escribiendo código. Sé que este ejemplo es muy sencillo y posiblemente te preguntas cómo sería hacer algo más complicado.

No te preocupes, a medida que avances en esta lectura, iras aprendiendo más trucos y los ejemplos y ejercicios se irán complicando cada vez más, hasta que te conviertas en un experto.

Comentarios en el código

Si has puesto atención, cuando observas el código que genera la grabadora de macros de Excel, genera algunos textos, que no son comandos en las macros, estos se llama comentarios y ayudan a que el código sea más fácil de leer y entender por nosotros los humanos.

Los comentarios son parte fundamental de todo programa, pero muchos programadores no los usan porque creen que les hace perder el tiempo. Pero en realidad es todo lo contrario, los comentarios pueden ayudarte a no perder tu tiempo. Voy a explicarte por qué debes utilizar siempre comentarios en todos los programas que escribas.

Nuestra memoria no siempre es confiable y cuando escribimos código, todo está claro en nuestra mente, pero rara vez escribimos un programa y no volvemos a verlo o modificarlo nunca. Un programa debe revisarse y corregirse a lo largo de toda su vida útil.

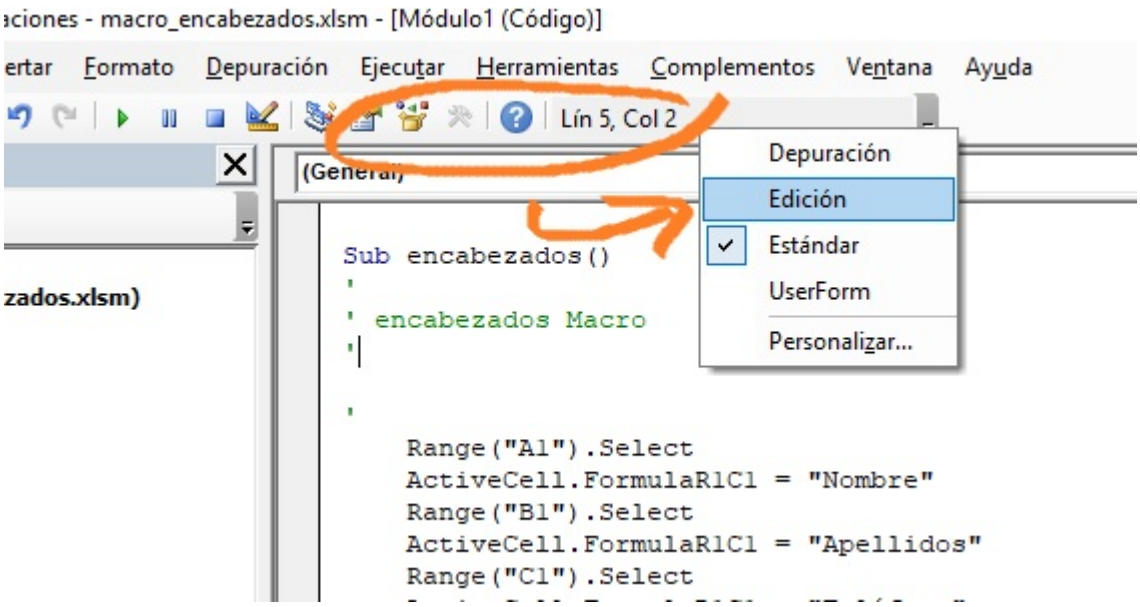
Entonces ¿qué pasa si debes agregar alguna funcionalidad a una macro 6 meses después de haberla creado?, lo más probable es que no recuerdes cómo funciona alguna fórmula que usaste, de donde tomas algún valor o por qué haces algo de una forma, en lugar de otra.

Todos estos problemas (y otros más) se resuelven al leer los comentarios que has escrito en el código. Ahora que ya sabes por qué debes usar los comentarios, veamos cómo puedes usarlos.

Si escribes una comilla simple ' , eso indica que toda la línea es un comentario y será ignorada como código. Y ya que los comentarios no se ejecutan, entonces otro uso que se le da a los comentarios es el de deshabilitar temporalmente un bloque de código.

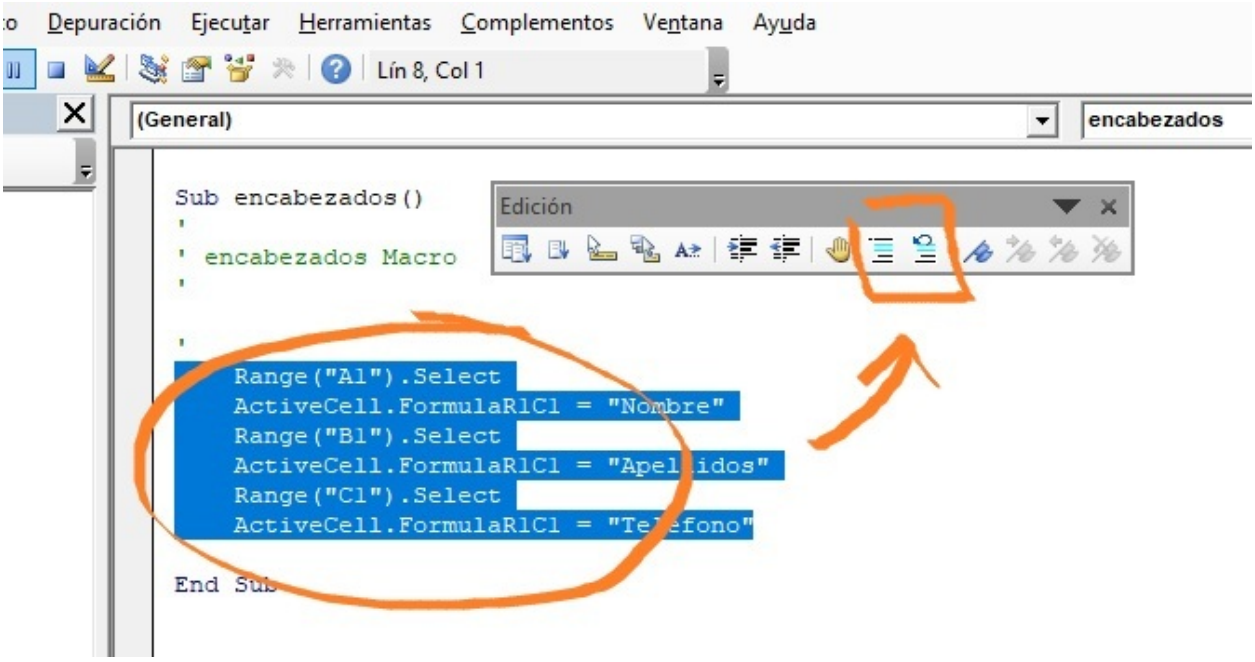
Pero si deseas convertir en comentario un bloque grande de código, no es práctico hacerlo línea por línea, entonces debes activar una barra de herramientas de Excel que te ayude en este proceso.

Haz clic derecho sobre la barra de herramientas actual y selecciona la opción *Edición* como se muestra en esta imagen:



Activar barra de herramientas Edición

Ahora para comentar un bloque, solo marcas el texto y presionas uno de los botones en la barra de herramientas para comentar o descomentar el bloque. Intenta hacerlo tú mismo.



Comentar bloque

Ejercicio

Tan solo hemos iniciado a conocer VBA, entonces el ejercicio de este capítulo es sencillo, debes grabar una macro (usando la grabadora), haz algo sencillo, por ejemplo una macro que escriba "Hola mundo" en la celda en donde te encuentres, luego entra al editor de VBA y explora la macro, revisa el código que se generó e intenta modificar "Hola mundo" por cualquier otro texto, luego corre la macro y comprueba que lo hiciste bien.

Introducción a la programación

Como has visto hasta ahora, puedes crear macros sin necesidad de programar, todo se ha hecho usando la grabadora de macros.

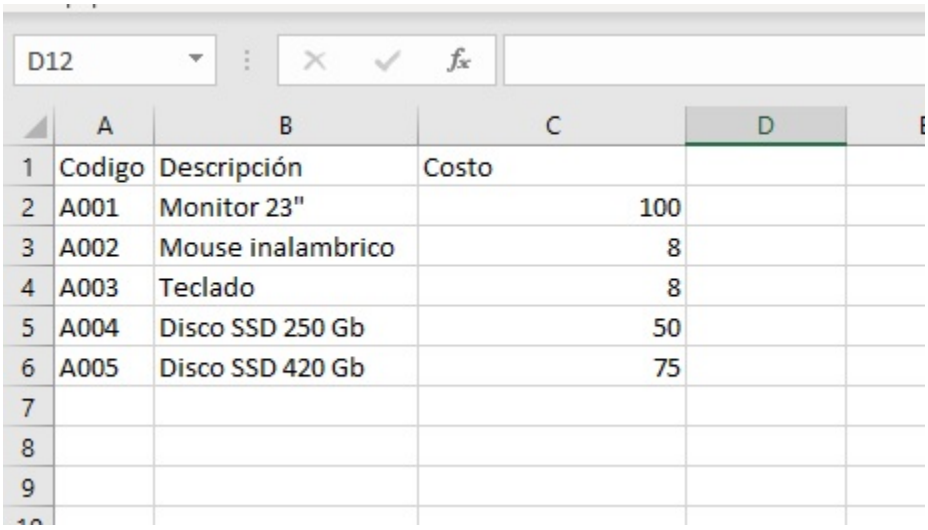
Pero para sacar el máximo provecho de las macros de Excel, necesitas crear macros usando código de VBA. Por eso ahora vas a crear tu primer macro desde cero y usando solo código de programación.

Pero no te preocupes si nunca has programado, porque este capítulo cubre todos los conceptos básicos de programación.

Tu primer macro usando VBA

Ahora imagina que trabajas en una empresa en donde te dan una hoja de Excel con una lista de productos y sus costos, tu trabajo es calcular los precios. Como política de la empresa, los precios se calculan como el costo más un 30% extra.

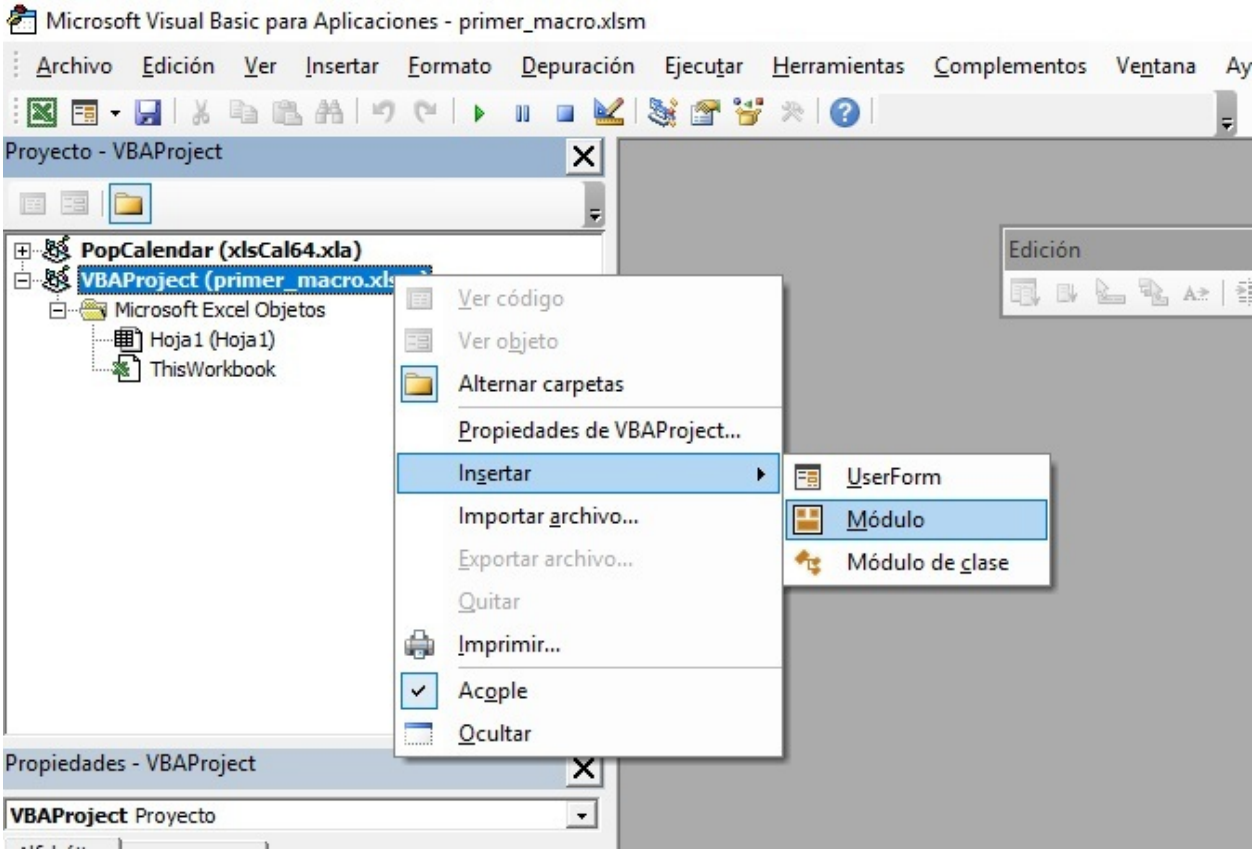
Entonces básicamente lo que debes hacer es crear una nueva columna y calcular el precio como el costo más 30%. Crea una hoja de Excel que se vea como esta:



	A	B	C	D
1	Codigo	Descripción	Costo	
2	A001	Monitor 23"	100	
3	A002	Mouse inalambrico	8	
4	A003	Teclado	8	
5	A004	Disco SSD 250 Gb	50	
6	A005	Disco SSD 420 Gb	75	
7				
8				
9				

Lista de productos

Ahora presiona las teclas `Alt+F11` para abrir el editor de VBA y haz clic derecho sobre el proyecto VBAProject con el nombre de tu hoja de cálculo y selecciona la opción de menú *Insertar* > *Módulo*



Insertar modulo

Ahora vas a crear una subrutina, todas las macros que creas en la grabadora son subrutinas. Para crearla solo debes escribir algo como esto:

```
1  Sub calcula_precios()
2
3  End Sub
```

En donde `Sub` y `End Sub` marcan el inicio y el fin de la subrutina, la palabra `calcula_precios` es el nombre de tu macro y siempre va seguido de paréntesis. Dentro de los paréntesis pueden ir algunos parámetros, pero vamos a dejarlo de esta forma por ahora.

Acabas de crear una macro vacía, para que funcione necesita algo de código (instrucciones de lo que debe hacer), ahora puedes modificarla de forma que te quede así:

```
Sub calcula_precios()

    Dim costo
    Dim fila As Integer

    'Coloca encabcezado
    Range("D1").FormulaR1C1 = "Precio"

    'Recorrer todas las filas
    fila = 2
    costo = Range("C" & fila).FormulaR1C1
    Do While IsNumeric(costo) 'Procesar mientras el costo sea un numero
        'Calcular el precio
        Range("D" & fila).FormulaR1C1 = costo * 1.3

        'Pasar a las siguiente fila y leer el costo
        fila = fila + 1
        costo = Range("C" & fila).FormulaR1C1
    Loop

End Sub
```

Hay mucho código nuevo aquí, no te preocupes por ahora, voy a ir explicando algunos conceptos que necesitas para comprender todo. Por el momento puedes descargar la hoja de cálculo y hacer unas pruebas ejecutando la macro.

> Puedes descargar la hoja de Excel junto con su macro haciendo clic aquí[2].

Funciones o subrutinas

En VBA usualmente ingresas el código en una función o en una subrutina, ambos términos se refieren a fragmentos de códigos que realizan una tarea específica y puede ser llamados las veces que necesites. Realmente no hay una limitante sobre la cantidad de tareas que pueda realizar una subrutina o una función, esto es simplemente una buena práctica, ya que de esta forma puedes reutilizar el código fácilmente, es menos propenso a errores y es más fácil de modificar o mejorar.

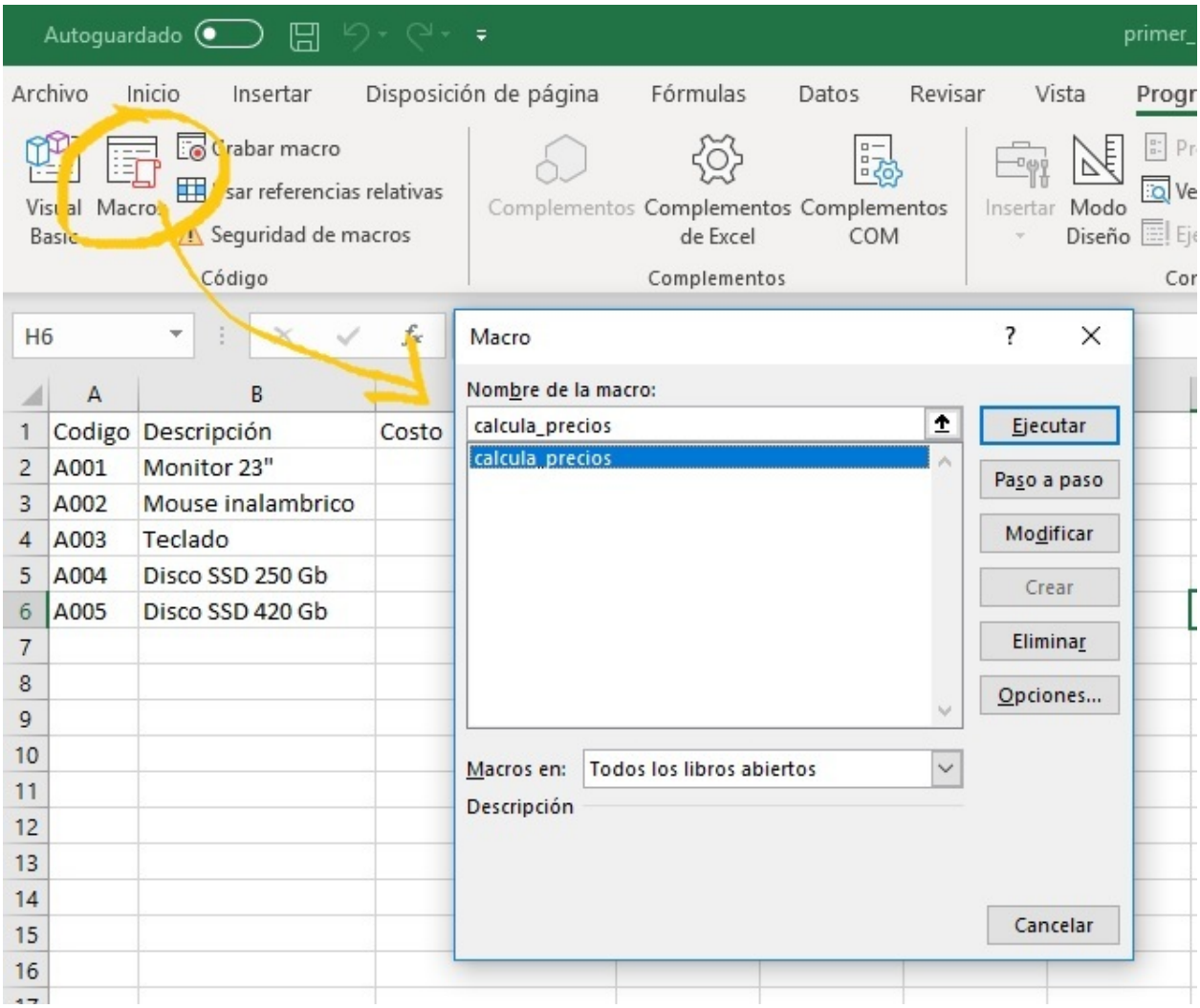
Ahora, la diferencia entre una función y una subrutina es que la función puede regresar un valor, por ejemplo, puedes crear una función que reciba como parámetro un código de cliente y regrese el nombre del cliente. En cambio, las subrutinas no pueden regresar ningún valor.

El ejemplo más común de funciones son las funciones que vienen en Excel, por ejemplo, la función SUMA que recibe un rango de celdas y regresa la suma de todos sus valores. Un ejemplo de una

[2] https://my.pcloud.com/publink/show?code=XZLyWE7ZsQiRtGafe8Y3Wp5zRw9FpVB9DgoX

subrutina es una macro que genera un reporte de Excel, esta subrutina no regresa ningún valor, pero si puede realizar cambios en una hoja de Excel. Las funciones también pueden hacer cambios en una hoja de Excel, pero su uso habitual es devolver un valor.

Otra diferencia importante es que solo puedes llamar macros desde una hoja de Excel a aquellas que has creado como subrutinas que no lleven ningún parámetro. Puedes hacer la prueba con la subrutina que acabas de crear.



Insertar modulo

Variables, constantes y tipos de datos

Las variables son zonas de memoria en tu computadora, a las que puedes acceder para guardar o consultar información. Para poder decirle a la computadora a que parte de la memoria te refieres, usas un nombre para tu variable.

Los nombres de variables no pueden llevar espacios y solo deben contener letras o números, pero puedes usar algunos caracteres como el guion bajo para separar palabras. También hay algunos nombres que no puedes usar, estos nombres se conocen como palabras reservadas.

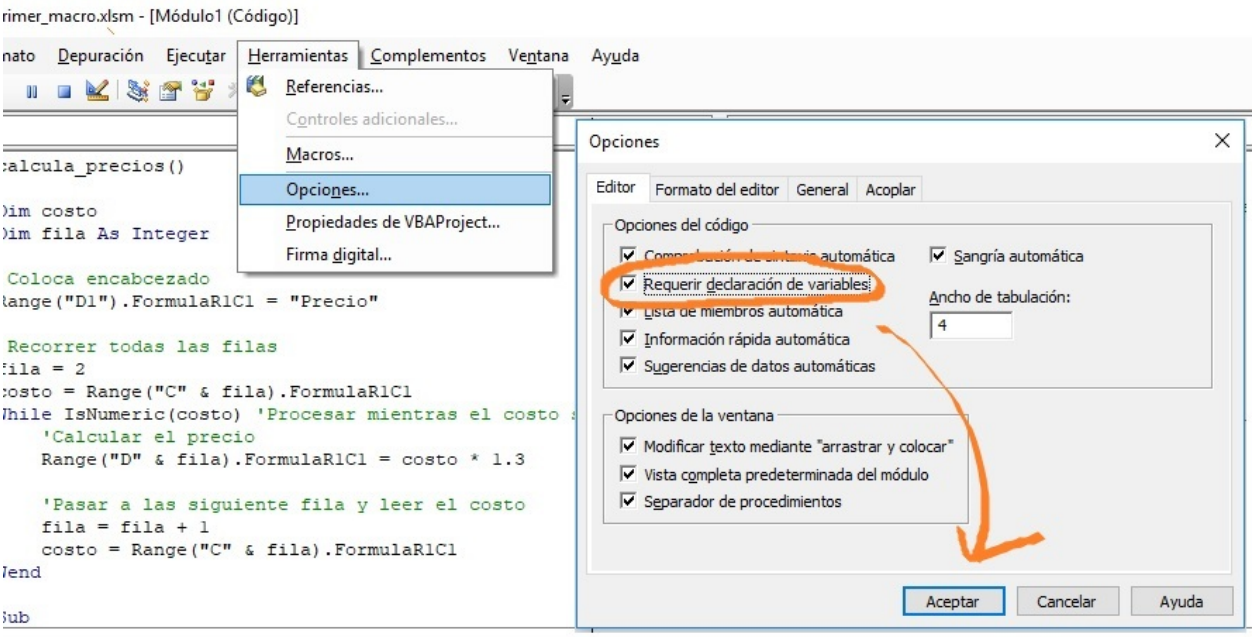
Algunas de estas palabras que no puedes usar son: Function, Sub, Dim, etc. Para una lista completa puedes consultar este artículo que detalla todas las palabras reservadas[3].

Ahora, para declarar una variable utilizas la palabra `Dim` seguida del nombre de tu variable y opcionalmente la palabra `as` seguida del tipo de datos, por ejemplo, para declarar una variable llamada fila y de tipo numérico usamos un código como este:

```
1  Dim fila As Integer
```

Si has intentado aprender VBA por tu cuenta, quizá sepas que no es necesario declarar una variable para poder usarla, pero si es una buena práctica hacerlo. Te recomiendo que siempre las declares, si no lo haces podrías crear errores lógicos, los cuales son muy difíciles de corregir. Por ejemplo, imagina que usas una variable llamada `impuestos`, luego haces algunos cálculos y tratas de consultar su valor, pero te equivocas y escribes `impuesto`, es decir, olvidas colocar la letra `s` al final. Lo que pasará es que no obtienes ningún error, pero leerás el contenido de una variable sin declarar y sin el valor que esperas.

Para evitar este tipo de error puedes forzar a VBA a que siempre te pida declarar las variables, para esto seleccionas el menú `Herramientas` > `Opciones` y en la pantalla que se muestre debes marcar la opción `Requerir declaración de variables`.



Solicitar declaración de variables

[3] https://excel.facilparami.com/2019/04/palabras-reservadas-en-vba

Ahora te verás obligado a declarar siempre cada variable que utilices, pero a cambio ahorrarás mucho tiempo tratando de encontrar errores lógicos. Para declarar una variable puedes usar cualquiera de estos tipos de datos:

Tipo de datos	Rango de valores
Boolean	True o False (Verdadero o Falso)
Integer	números del −32,768 al 32,767
Long	números del −2,147,483,648 al 2,147,483,647
Single	números del −3.402823E38 al 1.401298E45
Double	números del −1.79769313486232E308 al −4.94065645841247E-324
Double	números del 4.94065645841247E−324 al 1.79769313486232E308
Date	Fechas del 1/1/100 al 31/12/9999
String	Cualquier texto
Object	Cualquier objeto
Variant	Cualquier valor de cualquier tipo
Currency	−922,337,203,685,477.5808 a 922,337,203,685,477.5807

Nota: El tipo Double está dos veces, por que posee dos rangos, uno para valores negativos y otro para valores positivos.

Ya sabes cómo declarar una variable, ahora, para asignarle un valor solo escribes el nombre de la variable y escribes un signo = y el valor que deseas asignar, por ejemplo:

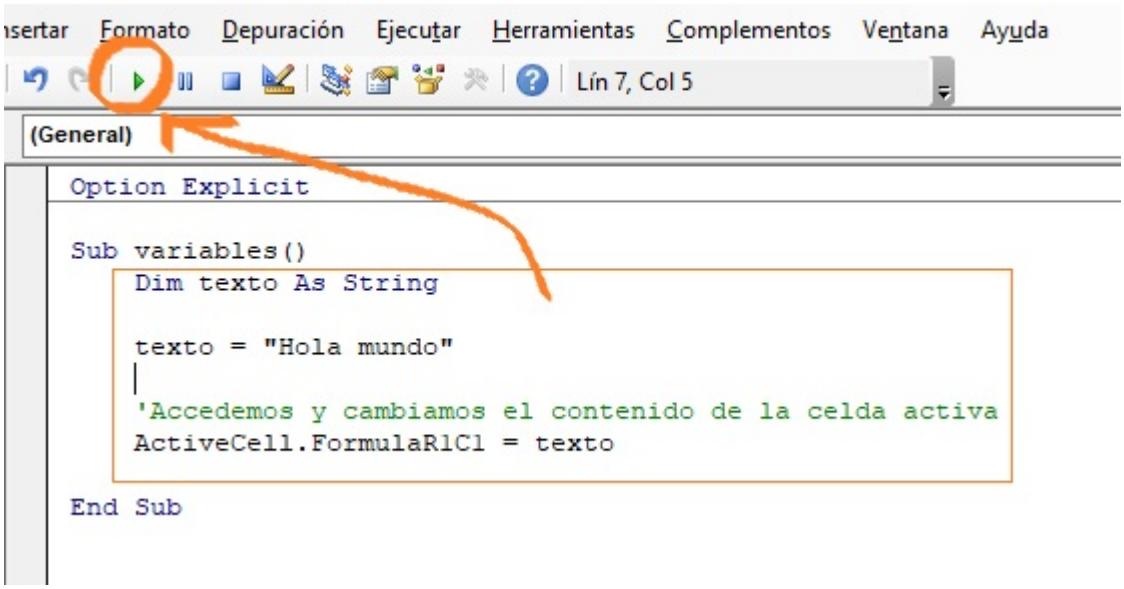
```
1  Dim fila as Integer
2  Dim texto as String
3
4  fila = 1
5  texto = "Hola mundo"
```

Como puedes notar los valores numéricos se colocan tal y como los ves, pero los valores de texto se escriben entre comillas dobles. Ahora que tienes valores definidos en tus variables, puedes utilizarlos en tu macro, veamos un ejemplo sencillo del uso de variables, donde tendremos una macro que actualiza el contenido de una celda.

```
1  Sub variables()
2  Dim texto As String
3
4  texto = "Hola mundo"
5
6  'Accedemos y cambiamos el contenido de la celda activa
7  ActiveCell.FormulaR1C1 = texto
8
9  End Sub
```

Intenta crear y ejecutar la macro anterior en una hoja de Excel, y verás que una vez que hayas asignado un valor a la variable `texto`, podrás usarla y la macro entenderá que debe usar el valor asignado en la variable. Modifica el valor de la variable texto y revisa lo que pasa cuando la ejecutas de nuevo.

Una vez que crees la subrutina anterior puedes ejecutarla posicionando el cursor sobre cualquier parte de su código y luego presionando la tecla `F5` o presionando el botón Ejecutar.



Ejecutar una macro desde el editor de VBA

> Nota: Con ActiveCell.FormulaR1C1 puedes acceder al valor de la celda actual, esto es algo muy útil en muchas macros y a lo largo de este libro vas a aprender muchas otras cosas que te serán de utilidad al crear macros.

El contenido de una variable puede ser modificado en cualquier momento, esta es la única diferencia entre una variable y las constantes. Para declarar una constante utilizas la palabra `Const` seguida del nombre de la constante, el tipo de datos y su valor. Aquí hay un ejemplo en que se define una constante llamada FactorIVA de tipo `single` y con valor 0.13

```
Const FactorIVA As Single = 0.13
```

Quizá te preguntes por que declarar la constante anterior, si puedes usar su valor 0.13 y sabes que no va a cambiar, bueno hay dos razones muy importantes por las que debes usar constantes en lugar de los valores:

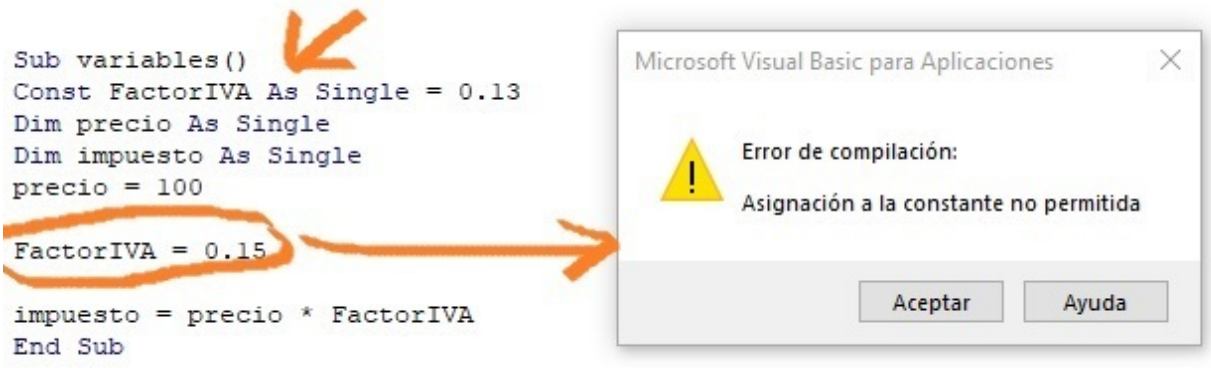
Para el caso anterior 0.13 es el factor para el IVA (un impuesto) y cuando estas leyendo el código de tu macro es más fácil de entender cuando usas constantes mira estos dos ejemplos idénticos y piensa cual es más legible:

```
impuesto = precio * 0.13

impuesto = precio * FactorIVA
```

La segunda razón es porque, si bien es cierto que el valor de la constante no va a cambiar durante toda la ejecución de tu macro, si puede darse el caso en que debes actualizar su valor, por ejemplo, ahora el valor del impuesto (IVA) es del 13%, pero que pasaría si el gobierno decide subirlo a 15%, en ese caso tendrías que leer todo tu código y evaluar si ese 0.13 se refiere al impuesto y si es así, entonces cambiarlo en todas las líneas de código en los que lo has usado, pero si decidiste utilizar una constante, solo debes modificar su valor y ya tienes todo arreglado.

Posiblemente también te preguntas, ¿por qué necesito la constante, si puedo hacer todo lo anterior usando una variable?, la respuesta es sencilla, las constantes existen para evitar que por error modifiques su valor en algún momento del programa. Como vimos puedes asignar un valor a una constate en tu código, pero si intentas asignar un valor a una constante ya definida, veras un error de compilación al intentar ejecutar tu macro.



Error al modificar una constante

> Compilación es un proceso en el cual el código que escribes se traduce a código que la computadora pueda entender y ejecutar.

Que son los arreglos

Un arreglo o array es un tipo especial de variable, que puede contener múltiples valores. Puedes acceder a todo el conjunto de valores o a un valor individual al especificar su índice, también puedes agregar o eliminar más valores al arreglo.

Puedes ver los arreglos como una lista de cosas, por ejemplo, una lista del supermercado, en la que puedes agregar más cosas que deseas o borrar algo que ya no quieres, también puedes leer la lista de forma secuencial, es decir, del primer ítem, hasta el último, o puedes consultar: “Cual es el primer producto en mi lista” y leer solo ese, por que has dado su índice (el primero en la lista).

Como declarar un arreglo

A diferencia de una variable normal, los arreglos siempre deben de ser declarados, otra regla que debes de seguir es que debes declarar la cantidad de elementos que piensas almacenar. Para tener todo más claro veamos un ejemplo de cómo declarar un arreglo con 25 ítems.

```
1  Dim ListaSupermercado(1 To 25) As String
```

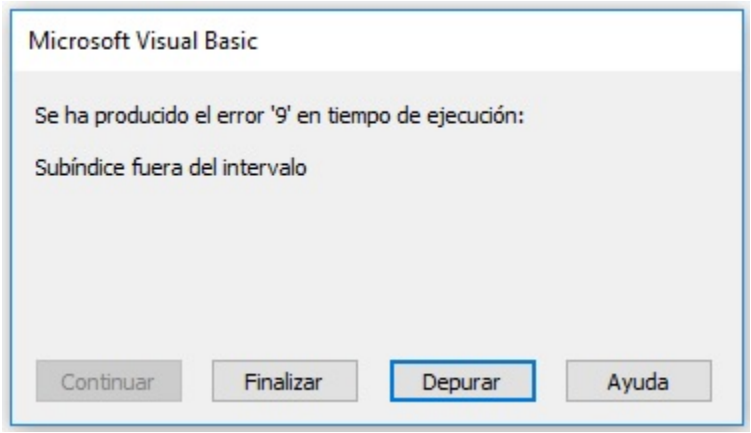
Como vemos, el tamaño se declara `1 to 25` entre paréntesis, con eso indicamos que el índice va desde `1` hasta 25.

Trabajar con los valores del arreglo

Ahora para asignar un valor a un arreglo, debemos hacer referencia a él, por su índice y luego asignar su valor como se hace con cualquier otra variable.

```
1  Dim ListaSupermercado(1 To 25) As String
2  'Asignar un valor al indice #1
3  ListaSupermercado(1) = "Pollo"
```

Si declaraste un arreglo de 25 posiciones y tratas de utilizar un índice que no existe, por ejemplo, el 26, obtendrás un error como este:



Error en índice del arreglo

Si no conoces la cantidad de posiciones que vas a necesitar, puedes declarar un arreglo dinámico sin detallar la cantidad de posiciones y luego redimensionarlo con la sentencia `ReDim`, aquí hay un ejemplo:

```
Sub Arreglos()

    'Arreglo dinamico, no detallamos la cantidad de posiciones
    Dim ListaSupermercado() As String

    'Asignamos una cantidad inicial
    ReDim ListaSupermercado(1 To 1)

    ListaSupermercado(1) = "Pollo"

    'Ahora lo cambiamos a 2 posiciones, pero usamos
    'la palabra Preserve para no perder los datos
    'que ya teniamos
    ReDim Preserve ListaSupermercado(1 To 2)
    ListaSupermercado(2) = "Queso"

    MsgBox ListaSupermercado(1) & " y " & ListaSupermercado(2)

End Sub
```

Ahora puedes cambiar en cualquier momento la cantidad de elementos, pero ahora es más difícil saber si estás haciendo referencia a un elemento que no existe, pero esto se puede solucionar con las funciones `LBound` y `UBound`, las cuales regresan el índice inferior y superior, respectivamente. Veamos de nuevo unos ejemplos:

```
Sub Arreglos()

    'Arreglo dinamico, no detallamos la cantidad de posiciones
    Dim ListaSupermercado() As String
    Dim indice As Integer

    'Asignamos una cantidad inicial
    ReDim ListaSupermercado(1 To 2)

    ListaSupermercado(1) = "Pollo"
    ListaSupermercado(2) = "Queso"

    'Obtener el indice inferior y superior para recorrer el arreglo
    For indice = LBound(ListaSupermercado) To UBound(ListaSupermercado)
        MsgBox ListaSupermercado(indice)
    Next indice

End Sub
```

También puedes usar estas funciones para obtener el primer y el último elemento de un arreglo o para validar si el arreglo tiene un índice, antes de intentar leerlo.

Decisiones y operadores lógicos

Los programas deben de tomar decisiones y ejecutar un bloque de código u otro dependiendo de la información que estén procesando. Las personas hacemos eso todo el tiempo, por ejemplo, si vas a tomar un café, lo pruebas y tomas una decisión, si está muy caliente, esperas a que se enfríe, si no entonces sigues tomando.

La forma de tomar decisiones es con bloques `if`, la estructura más general es la siguiente:

```
If evaluación Then
  'Bloque de codigo si la condicion es cierta
Else
  'Bloque de codigo si la condicion es falsa
End If
```

La `evaluación` debe ser una expresión que regrese falso o verdadero, la palabra `else` separa los bloques de código que se ejecuta cuando la evaluación es verdadera o falsa y las palabras `End If` marcan el fin de la decisión. Esto se verá mejor con un ejemplo, abre un archivo de Excel para crear y ejecutar esta macro:

```
Sub decisiones()

Dim edad As Integer

edad = 15
If edad >= 18 Then
  MsgBox "Ya eres mayor de Edad"
Else
  MsgBox "Eres menor de edad"
End If

End Sub
```

En la macro anterior la decisión consiste en evaluar si la edad es mayor o igual a 18, luego usamos la función `MsgBox` de VBA para mostrar un mensaje en la pantalla. Intenta cambiar el valor de la variable `edad` para ver cómo se comporta.

En el código anterior usamos el signo `>=`, esta es una comparación entre dos valores. Las comparaciones que puedes hacer son:

Operador	Resultado
=	Verdadero, si ambos valores son iguales, sino regresa Falso
<>	Verdadero, si ambos valores son diferentes, sino regresa Falso
>	Verdadero, si el primer valor es mayor que el segundo, sino regresa Falso
<	Verdadero, si el primer valor es menor que el segundo, sino regresa Falso
>=	Verdadero, si el primer valor es mayor o igual que el segundo, sino regresa Falso
<=	Verdadero, si el primer valor es menor o igual que el segundo, sino regresa Falso

Tanto en la vida como en las macros, las decisiones no siempre son sencillas, puedes necesitar hacer más de una evaluación y para esto necesitas operadores lógicos para unir todas las evaluaciones que necesites.

Por ejemplo, si vas a aplicar un descuento especial y necesitas que la cantidad que compren sea mayor a 10 unidades y que además el cliente sea de categoría "vip" entonces necesitas un operador lógico para unir ambas condiciones. Veamos un ejemplo:

```
1  Sub OperadoresLogicos()
2      Dim categoria As String
3      Dim cantidad As Single
4      Dim precio As Currency
5
6      cantidad = 10
7      categoria = "vip"
8      precio = 100
9
10     'Evaluar descuento
11     If cantidad >= 10 And categoria = "vip" Then
12         'Aplicar 30% de descuento
13         precio = precio * 0.7
14         MsgBox "El nuevo precio es " & precio
15     End If
16 End Sub
```

Si buscas la sentencia (bloque de código) `if` veras que hay dos comparaciones unidas por una palabra And, esta es un operador lógico. Intenta cambiar los valores de las variables `cantidad` y `categoria` para ver si aplica el descuento.

En total existen 3 operadores lógicos los cuales son: `And`, `Or` y `Not`.

El operador `And` trabaja sobre dos expresiones a su izquierda y derecha y si ambas expresiones son verdaderas entonces regresa verdadero, pero si cualquiera de las dos es falso, entonces regresa falso.

El operador `Or` también trabaja sobre dos expresiones a su izquierda y derecha y si cualquiera de las expresiones es verdadero entonces regresa verdadero y regresa falso únicamente si ambas expresiones son falsas.

El operador `Not` trabaja solamente sobre la expresión a su derecha, lo que hace es invertirla o negarla, por ejemplo, si la expresión es falsa, entonces regresa verdadero y si la expresión es verdadera entonces regresa falso.

Veamos una tabla con algunos ejemplos:

Expresión	Resultado
1 = 1 And 1=2	Falso (el primero bloque es cierto, pero el segundo es falso entonces todo es falso)
1 = 1 And 2=2	Verdadero (ambas expresiones son verdaderas)
1 = 1 Or 1 = 2	Verdadero (la primera expresión es verdadera, entonces no importa el resultado de la segunda)
1 = 2 Or 2 = 2	Verdadero (la segunda expresión es verdadera, entonces no importa el resultado de la primera)
1 = 2 Or 2 = 1	Falso (Ambas expresiones son falsas)
Not 1 = 1	Falso (La expresión es verdadera, pero esta negada)
Not 1 = 2	Verdadero (La expresión es falsa, pero esta negada)

Cuando usamos una expresión `if`, la parte del `else` es opcional y también debes de saber que puedes anidar sentencias `if`, para comprender mejor esto, veamos un ejemplo:

```
1  Sub OperadoresLogicos()
2      Dim categoria As String
3      Dim cantidad As Single
4      Dim precio As Currency
5
6      cantidad = 10
7      categoria = "vip"
8      precio = 100
9
10     'Evaluar descuento
11     If cantidad >= 10 Then
12         'If anidado: es un if dentro de otro if
13         If categoria = "vip" Then
14             'Aplicar 30% de descuento
15             precio = precio * 0.7
16             MsgBox "El nuevo precio es " & precio
17         End If
18     End If
19 End Sub
```

El *if* anidado es un bloque *if*, dentro de otro bloque *if*, puedes utilizarlo para tomar decisiones más complicadas, sin embargo, anidar muchos *if* puede generar código difícil de leer. Para evitar este problema existe otra sentencia llamada `Select Case`, en ella se evalúan varias condiciones y se puede ejecutar un bloque de código por cada evaluación.

Para comprender mejor cuando usar una sentencia `if` o `Select Case` veamos una misma macro en las dos versiones. Imagina que te piden una macro para determinar el descuento que aplica a un producto en base a la cantidad que compran aplicando estos criterios:

- Si la cantidad esta entre 0 y 20 no hay descuento
- Si la cantidad es mayor que 20 el descuento es de 10%
- Si la cantidad es mayor que 30 el descuento es de 15%
- Si la cantidad es mayor que 40 el descuento es de 20%
- Si la cantidad es mayor que 50 el descuento es de 25%

La macro usando if que resuelve este problema es la siguiente (como siempre debes crear y probarla en tu computadora)

```
Sub DescuentoIf()
    Dim cantidad As Single
    Dim descuento As Single
    cantidad = InputBox("Ingrese cantidad")


    If cantidad > 50 Then
        descuento = 25
    Else
        If cantidad > 40 Then
            descuento = 20
        Else
            If cantidad > 30 Then
                descuento = 15
            Else
                If cantidad > 20 Then
                    descuento = 10
                End If
            End If
        End If
    End If

    MsgBox "El descuento es de " & descuento & "%"
End Sub
```

Como puedes observar, tantos `if` anidados puede ser algo confuso. También debes notar que he introducido la función `InputBox` que permite preguntar al usuario un valor y guardarlo en una variable, esto hará que sea más fácil probar esta macro. Después de probar esta macro intenta probar la versión `Select Case` y verás que el resultado es el mismo, pero el código es más limpio y fácil de entender.

```
Sub DescuentoSelectCase()
    Dim cantidad As Single
    Dim descuento As Single
    cantidad = InputBox("Ingrese cantidad")

    Select Case cantidad
        Case 0 To 20
            descuento = 0
        Case 21 To 30
            descuento = 10
        Case 31 To 40
            descuento = 15
        Case 41 To 50
            descuento = 20
        Case Else
            descuento = 25
    End Select

    MsgBox "El descuento es de " & descuento & "%"
End Sub
```

Posiblemente ya descifraste la forma de usar la sentencia `Select Case`, pero si no lo has hecho, voy a explicar con mayor detalle cómo se usa. Todo el bloque se define entre las palabras `Select Case Variable` y finalizan con `End Select`, luego hay una o más sentencias `Case` con una expresión y su respectivo bloque de código. Existe un caso `Else` que es el que se ejecuta si ninguno de los casos anteriores fue ejecutado.

Como ya habrás notado, la sentencia `Select Case` es un poco más compleja que la sentencia `if`, pero más fácil de leer cuando hay varias posibles opciones a una decisión. Pero la mejor forma de dominar esta sentencia es por medio de algunos ejemplos prácticos, afortunadamente puedes leer algunos en este artículo[4].

Operaciones con variables

Trabajar con Excel es (la mayoría de las veces) realizar cálculos y para ello necesitas realizar sumas, restas, etc. En esta tabla tenemos una lista de los operadores disponibles:

[4] https://excel.facilparami.com/2019/04/ejemplos-practicos-de-la-sentencia-select-case

Operador	Descripción
+	Suma
*	Multiplicación
/	División
-	Resta
^	Exponenciación
\	División entera (regresa solo la parte entera, sin decimales)
Mod	Regresa el residuo de una división, por ejemplo `5 mod 2 = 1`
&	Concatenar dos cadenas de texto (unir dos textos)

Ya hemos utilizado algunos de estos operadores de los ejemplos anteriores, pero tener esta lista te servirá como referencia.

Qué son los ciclos

Los ciclos son repeticiones de un bloque de código, regresando al ejemplo de las decisiones, cuando te sirven una taza de café, primero lo pruebas si está muy caliente esperas un rato y lo vuelves a probar, repites este paso de probar y decidir hasta que la temperatura sea de tu agrado para tomar tu café.

Lo mismo sucede con los programas, un bloque de código puede necesitar repetirse un numero conocido o desconocido de veces.

El ciclo `for` se usa cuando conoces la cantidad de veces que necesitas repetir el bloque de código, la sintaxis (estructura) del bloque `for` es:

```
for variable = valorInicial to valorFinal Step 1
        'Ejecutar codigo
Next variable
```

Debes declarar una variable que se usará como contador, luego defines un valor inicial y un valor final, opcionalmente puedes definir el valor del incremento con la instrucción Step que de forma predeterminada es 1, si piensas hacer incrementos de 1 puedes no usarla.

El ciclo funcionará así: si la variable tiene un valor inicial de 1, un valor final de 5 y un incremento (step) de 1, entonces la primera vez que se ejecuta la variable contiene un 1, la segunda vez la variable se incrementa a 2, luego a 3, luego a 4 y finalmente se incrementa a 5 y es la última vez que se ejecuta el bloque de código.

Por ejemplo, imagina que debes generar una hoja de Excel que contiene el número de filas en la columna A, y que inicia desde la celda A2, entonces la celda A2 tendrá 1, la celda A3 tendrá 2 y así sucesivamente hasta llegar a 10. Entonces el código para hacer eso sería este:

```
Sub CicloFor()
    Dim fila As Integer

    For fila = 1 To 10
        Range("A" & fila + 1) = fila
    Next fila

End Sub
```

La función `Range` se puede usar para hacer referencia a una celda de la hoja actual, recibe como parámetro el nombre de la celda por ejemplo A1 pero como sabemos que es la columna A, pero desconocemos el número de la fila, entonces usamos una concatenación de la `A` y el contenido de la variable `fila + 1` y le asignamos el contenido de la variable `fila` que en cada ciclo contendrá 1, 2, 3...10

El ciclo `Do While` ejecuta un bloque de código durante un número desconocido de veces, se hace mientras se cumpla la condición que definimos, por ejemplo, mientras el café este muy caliente. La sintaxis de este ciclo es:

```
Do While Condicion
   ' Bloque de codigo
Loop
```

Como vemos, se colocan las palabras `Do While` seguidas de una condición y se cierra el bloque de código con una palabra `Loop`. El bloque de código va a ejecutarse mientras la condición sea verdadera.

Usamos este ciclo en la macro que creamos al inicio del capítulo, aquí está de nuevo para que la recuerdes:

```
Sub calcula_precios()

    Dim costo
    Dim fila As Integer

    'Coloca encabcezado
    Range("D1").FormulaR1C1 = "Precio"

    'Recorrer todas las filas
    fila = 2
    costo = Range("C" & fila).FormulaR1C1
    Do While IsNumeric(costo) 'Procesar mientras el costo sea un numero
        'Calcular el precio
        Range("D" & fila).FormulaR1C1 = costo * 1.3

```

```
        'Pasar a las siguiente fila y leer el costo
        fila = fila + 1
        costo = Range("C" & fila).FormulaR1C1
    Loop
  End Sub
```

Creamos una variable llamada `costo` con el contenido de la celda que corresponde la fila que estamos procesando, luego usamos la función `IsNumeric` para preguntar si su valor es un número, así sabremos si ya hemos terminado, porque no encontrará un número, sino una celda vacía cuando termine de procesar todas las filas con datos.

También debes notar que ahora usamos `Range("C" & fila).FormulaR1C1` en lugar de solo `Range("C" & fila)`, hacemos esto para poder usar la función `IsNumeric` ya que `Range("C" & fila)` convertirá el valor vacío a un cero y entonces `IsNumeric` siempre va a regresar Verdadero. `FormulaR1C1` regresa el contenido de la celda sin hacer ninguna conversión.

Hay algo muy importante que debes conocer, como ves este ciclo se repite un valor indefinido de veces y al final de este ciclo, he actualizado el valor de la variable `costo`, con el contenido de la siguiente fila, si no hago esto entonces la comparación que hace el ciclo no va a cambiar nunca y el resultado será siempre verdadero, no habrá forma de terminar el ciclo y esto se conoce como ciclo infinito, tu computadora quedara atrapada en un ciclo sin fin hasta que reinicies tu computadora o termines el proceso con el administrador de tareas.

También existen algunas variantes de este ciclo y son las siguientes:

Repetir mientras la condición sea falsa, se termina si la condición es verdadera.

```
Do Until Condicion
      'Bloque de Codigo
Loop
```

Repetir al menos una vez sin importar la condición, y luego repetir mientras la condición sea verdadera.

```
Do
      'Bloque de Codigo
Loop While Condicion
```

Repetir al menos una vez sin importar la condición, y luego repetir mientras la condición sea falsa.

```
Do
      'Bloque de Codigo
Loop Until Condicion
```

Ejercicio

El ejercicio de este capítulo es sencillo, con la macro que creamos al inicio y con lo que has aprendido trata de hacer algunas modificaciones, por ejemplo:

1. Cambia el factor de 1.3 a 1.25
2. Cambia el titulo *Precio*, por *Precio Unitario*
3. Agrega más filas de productos y mira que pasa
4. Cambia el ciclo `Do While` por un ciclo `For` y revisa si todo sigue funcionando.