ACERCA DEL AUTOR

LUIS DOMINGO GIMÓN RODRÍGUEZ

Nacido en el año 1964, en la Ciudad de Las Palmas de Gran Canaria (Islas Canarias – España). Entre mis titulaciones figuran las siguientes.

- Ingeniero Técnico de Obras Públicas en la Especialidad de Construcciones Civiles por la Universidad de Las Palmas de Gran Canaria.
- Grado de Ingeniería Civil por la Universidad Católica de San Antonio de Murcia de Guadalupe (Murcia).
- Máster Universitario en Prevención de Riesgos Laborales en las Especialidades de Seguridad en el Trabajo, Higiene Industrial y Ergonomía y Psicosociología Aplicada por la Universidad Camilo José Cela de Villafranca del Castillo (Madrid).
- Coordinador de Seguridad y Salud en Obras de Construcción por el Instituto Canario de Seguridad Laboral, de la Consejería de Empleo y Asuntos Sociales del Gobierno de Canarias.
- Certificado de Aptitud Pedagógica (CAP) por el Instituto de Ciencias de la Educación de la Universidad Alfonso X El Sabio de Villanueva de la Cañada (Madrid).

Comencé a dar clases particulares a los 20 años. Entre el 1987 y 1989 estuve impartiendo clases de informática básica a niños de EGB en un Colegio de Las Palmas de Gran Canaria. Entre 1998 y 2000 fui director de un Centro de Formación que enseñaba el uso de aplicaciones informáticas de ofimática mediante un programa propio desarrollado por la empresa matriz.

En el año 2000 entré en el Servicio Municipal de Vías y Obras del Ayuntamiento de Las Palmas de Gran Canaria realizando labores de Auxiliar Técnico de Obras Públicas hasta el año 2005.

Entre los años 2005 y 2006 redacté Proyectos de obras encaminadas a la solicitud y obtención del Símbolo Internacional de Accesibilidad (SIA) (concedido por la Consejería de Empleo y Asuntos Sociales del Gobierno de Canarias) para varios Centro de Formación en Canarias.

Desde el año 2005 trabajo como Técnico Municipal de Prevención de Riesgos Laborales en el Instituto Municipal para el Empleo y la Formación (IMEF) del Ayuntamiento de Las Palmas de Gran Canaria.

Desde el año 1984 he impartido formación de MS Excel, MS Word, MS Access, MS PowerPoint, WordPerfect Suite, Windows 95, 3.11 TG, CorelDRAW, AutoCAD y Prevención de Riesgos Laborales (3 Especialidades Preventivas) para diversas empresas privadas y organismos públicos.

Actualmente (desde el 2017) soy tutor dentro de la Bolsa de Formadores Municipales del Ayuntamiento de Las Palmas de Gran Canaria, en dónde he impartido formación en AutoCAD 2D/3D, Desarrollo de Presentaciones Electrónicas (Google Drive Slide, Microsoft OneDrive PPT, Prezi) y Aplicaciones Google Drive online.

Soy especialista (he impartido formación específica) en el diseño de trazados (tracks y rutas de orientación y navegación) con los Sistema de Información Geográfica (SIG) Google Earth y Garmin BaseCamp, además del servicio de creación de mapas personalizados Google My Maps y Google Maps, entre otras herramientas informáticas, para uso en dispositivos de navegación (GPS Outdoor), principalmente orientado al uso en viajes organizados y eventos con motocicletas de carretera y de enduro (habiendo participado en varias pruebas como amateur).

Siempre me ha gustado mucho la programación. Me ha dado la sensación de ser capaz de gestionar las cosas, de controlar el proceso para conseguir el resultado que busco. De hecho, mi primer año de Carrera Universitaria fue, precisamente, en Informática. Me di cuenta entonces de lo mucho que disfrutaba con la programación y con los algoritmos.

Recuerdo lo divertido que era hacer pequeños programas en Basic (Beginners' All-purpose Symbolic Instruction Code - Código Simbólico de Instrucciones de Propósito General para Principiantes). Aquel famoso Lenguaje de Alto Nivel, diseñado originalmente en el año de mi nacimiento (1964), pensado para que cualquiera pudiera desarrollar un código, era un auténtico deleite. El primer libro de Basic que cayó en mis manos me lo leí completo en un fin de semana.

Con el paso de los años abandoné la programación, me centré en mis estudios de Ingeniería y en la vocación de enseñar el uso y manejo de diversas aplicaciones informáticas.

En los últimos 5 años he comenzado a interesarme por nuevos Lenguajes de Programación, a nivel de hobbies. Precisamente Visual Basic para Aplicaciones (VBA), Lenguaje de Programación de Microsoft Office con el que es posible crear aplicaciones nuevas y funciones personalizadas, era uno de los que quería conocer para trabajar con Microsoft Excel con mucha mayor productividad. Pero no fue hasta finales del año 2022 cuando me lo tomé en serio y decidí profundizar en él.

Y así fue como tomé la decisión de, no sólo aprender VBA Excel, sino marcarme como objetivo el poder enseñarlo. La enseñanza es mi mayor vocación y nada me satisface más que ser capaz de transmitir conocimiento y aprendizaje.

PRÓLOGO

A finales del año 2022, un buen amigo me pidió ayuda con unos archivos de Microsoft Excel. Se trataba de mejorar la productividad de las Hojas de Cálculo con las que trabajaba y lo que hice fue enseñarle algunas funciones que él desconocía.

En ese momento, le comenté que había ciertas cosas que se podría automatizar con Macros (sin entrar en la Programación VBA Excel). Pero llegó un punto en que las Macros no me dejaban ir más allá. Uno de los principales inconvenientes era no poder tomar decisiones usando condicionales.

Entonces entendí que la única manera de mejorar las Hojas con las que estábamos trabajando era implementar soluciones basadas Programación VBA Excel. Pero, aunque, hacía muchos años, había estudiado versiones iniciales de Basic y, más recientemente, me había interesado por Python, nunca me había sentado a estudiar el **Lenguaje de Macros de Microsoft Excel**.

Le propuse a mi amigo que, si tenía paciencia, yo aprendería VBA Excel y le ayudaría a mejorar sensiblemente el uso de sus archivos.

Me dediqué a leer mucha documentación en internet sobre VBA Excel. Encontré páginas muy interesantes que enseñaban desde cero. Recopilé mucha información oficial de Microsoft e incluso seguí un par de Cursos On-Line.

A medida que iba aprendiendo, fui tomando apuntes de forma ordenada y metódica, incluyendo teoría y ejemplos prácticos. Poco a poco, casi sin darme cuenta, mis anotaciones se fueron convirtiendo en un auténtico manual que recogía todo lo que aprendía. Así que, lo que empezó como un intento de mejora de unas Hojas de Cálculo de MS Excel, terminó convertido en este Curso que he podido completar con el apoyo y aliento constante de **Paki**, además del arreón final de **Marino**, quién me animó insistentemente a contactar con la Editorial, convencido de que este Manual podría ayudar a muchos a aprender a **desarrollar sus propias Aplicaciones bajo VBA Excel**.

He incluido en este libro gran parte de todo lo que he aprendido y estoy convencido que puede enseñar a adquirir habilidades para que cualquiera desarrolle sus propios programas basados en VBA Excel. Sin duda es el reto que más puede llenar a cualquier **docente vocacional**, como es mi caso.

Los **Conceptos Teóricos** son absolutamente necesarios para sentar las Bases del Aprendizaje de cualquier materia. Pero soy perfectamente consciente de que la **Práctica** es fundamental para **afianzar el conocimiento** y es, sin duda, lo que mayor interés y motivación provoca al lector que quiere aprender. Esta es la razón por la que he llenado el libro de ejemplos de todos los niveles.

Todo lo que he escrito (ejercicios y apuntes) está fundamentado y probado bajo las **Versiones de Microsoft Office 2019 y 2021** de **Microsoft Windows 10 y 11**.

CAPÍTULO 1

1.1 ¿QUÉ ES VBA?

Las siglas **VBA (Visual Basic for *Applications*)** significan **Visual Basic para Aplicaciones**. Según Microsoft, "***Office Visual Basic para Aplicaciones (VBA) es un Lenguaje de Programación Orientado a Eventos que le permite ampliar las aplicaciones de Office***" (*https://learn.microsoft.com/es-es/office/vba/api/overview/*).

VBA es la versión del Lenguaje de Programación Visual Basic que se incluye en **Microsoft Office**, que permite **Grabar**, **Crear** y **Editar Macros** para automatizar tareas sin ser programador, estrictamente hablando. A lo largo del curso, veremos que somos capaces de **Crear Nuevas Aplicaciones** y **Nuevas Funciones** bajo **Microsoft Excel**.

VBA se lanzó por primera vez con MS Excel 5.0 en 1993. A partir de Excel 97, VBA se convirtió en una versión totalmente independiente de Visual Basic y se incluyó en Microsoft Project, Access y Word, reemplazando a AccessBASIC y WordBASIC, respectivamente. Actualmente, las versiones de Microsoft Office incluyen VBA 7.1.

Aprender el uso de VBA Excel y desarrollar Código, no es complicado. Pero creo que es **absolutamente necesario** conocer **Microsoft Excel**, que, como cualquier otra aplicación de **Microsoft Office**, utiliza diversos Objetos (caso de Celdas, Hojas o Libros de Trabajo). Estos Objetos, a su vez, disponen de **Métodos** y **Propiedades** que controlaremos mediante la Programación VBA Excel.

Las aplicaciones de Microsoft Office disponen de un **Editor de VBA integrado** sobre el que trabajaremos para desarrollar el Código de Programación de cada Módulo. Casi cualquier cosa que puedas programar en Visual Basic 5.0 o 6.0 se puede hacer también dentro de un Archivo de MS Office. Pero, teniendo en cuenta que el Editor está integrado, la limitación es que tu Código VBA no se puede compilar separadamente del Documento, Hoja o Base de Datos donde fue creado.

VBA Excel permite automatizar los flujos de trabajo repetitivos y crear herramientas muy útiles para la gestión de todo tipo de proyectos, permitiendo ganar tiempo y competitividad. Nuestro ejercicio final es un claro ejemplo de este concepto.

Si alguna vez has utilizado **Macros de Excel**, se puede afirmar que ya has utilizado **VBA Excel**. Siempre que creamos una Macro, generamos un Programa con Código VBA Excel. De hecho, usaremos las Macros cuando queramos conocer cómo se genera un Procedimiento del que tenemos dudas o, sencillamente, no nos apetece comenzar a escribir Código partiendo de Cero. Pero, sin duda, llegará un momento, muy pronto, en que pretendamos ir mucho más allá de las **limitaciones que nos impone la Grabadora de Macros**.

Con VBA Excel, podremos programar Macros que automatizan tareas que se ejecutan con un simple **Click** de ratón sobre un **Botón de Comando**, por ejemplo. Para empezar, la forma más fácil de empezar a crear Código VBA Excel es utilizar la **Grabadora de Macros** integrada en MS Excel. La Grabadora de Macros genera Código VBA Excel que se puede leer y editar, lo que nos da un punto de partida para aprender a codificar nuestros propios programas partiendo de un proyecto vacío.

Para evitar confusiones entre **VB** (*https://learn.microsoft.com/es-es/office/vba/language/reference/user-interface-help/visual-basic-conceptual-topics*) y **VBA** (*https://learn.microsoft.com/es-es/office/vba/excel/concepts/miscellaneous/concepts-excel-vba-reference*), quiero comentar que Visual Basic (VB) es un Lenguaje de Programación y un entorno de desarrollo integrado (IDE) desarrollado por Microsoft, que permite a los desarrolladores crear aplicaciones y componentes de software de Windows. En cambio, **Visual Basic para Aplicaciones (VBA)** es un subconjunto casi completo de Visual Basic 5.0 y 6.0., diseñado específicamente para automatizar tareas y agregar Funciones Personalizadas a las aplicaciones de Microsoft Office, como Excel, Word, PowerPoint o Access.

Así pues, aunque VB y VBA comparten una sintaxis similar, tienen diferentes propósitos. VB se usa para crear aplicaciones independientes y **VBA** se usa para mejorar y automatizar tareas **dentro de los programas de Microsoft Office**.

Obviamente, aquellos que hayan desarrollado aplicaciones bajo VB, ya tienen un gran avance en la programación de Macros de Excel.

Ya veremos que VBA Excel cuenta con los típicos elementos de programación, como variables, matrices, funciones, decisiones y bucles.

Probablemente, aquellos que conozcan otros Lenguajes de Programación echarán en falta un mejor Control de Errores ya que el de VBA Excel es un poco simple, pero servirá, más que de sobra, para nuestros propósitos.

Debemos saber que **VBA Excel no puede utilizarse para crear programas independientes**, sitios web o servicios. Así que tendremos en cuenta que cuando programamos en VBA Excel, lo hacemos para manejar Libros de Trabajo y todos los demás Objetos dentro MS Excel.

Quien haya oído hablar de **Programación Orientada a Objetos (POO)**, debe saber que VBA Excel tiene un **soporte limitado para algunos conceptos orientados a Objetos**, como la **Encapsulación** (*un grupo de propiedades, métodos y otros miembros relacionados se tratan como una sola unidad u objeto*) y el **Polimorfismo** (*los objetos pueden tener múltiples clases que se pueden usar de manera intercambiable, aunque cada clase implementa las mismas propiedades o los mismos métodos de maneras diferentes*), así que, en sentido estricto, **VBA Excel no es un Lenguaje Orientado a Objetos**. Carece del concepto de **Herencia** (*describe la posibilidad de crear nuevas clases basadas en una clase existente*), lo que limita en gran medida la capacidad de ampliar la funcionalidad de los tipos existentes.

Pero dicho todo lo anterior, puedo asegurarte que **nuestra barrera será nuestra imaginación**, de modo que **VBA Excel no tendrá limitaciones para hacer fantásticas aplicaciones**. Seremos capaces de hacer **aplicaciones muy potentes que resuelvan muchísimos problemas y tareas**, tanto personales como laborales.

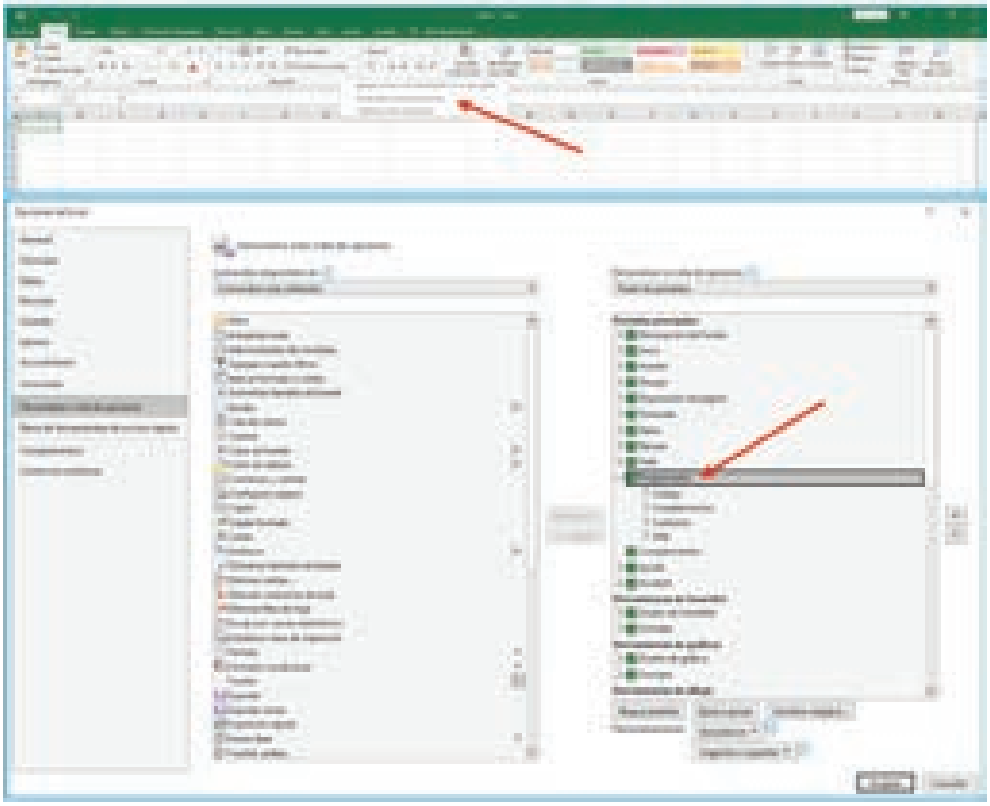
Te aconsejo que leas todo el contenido y que **vayas llevando a cabo todo y cada uno de los ejemplos y ejercicios desarrollados**, tratando de entender por qué se hace cada cosa.

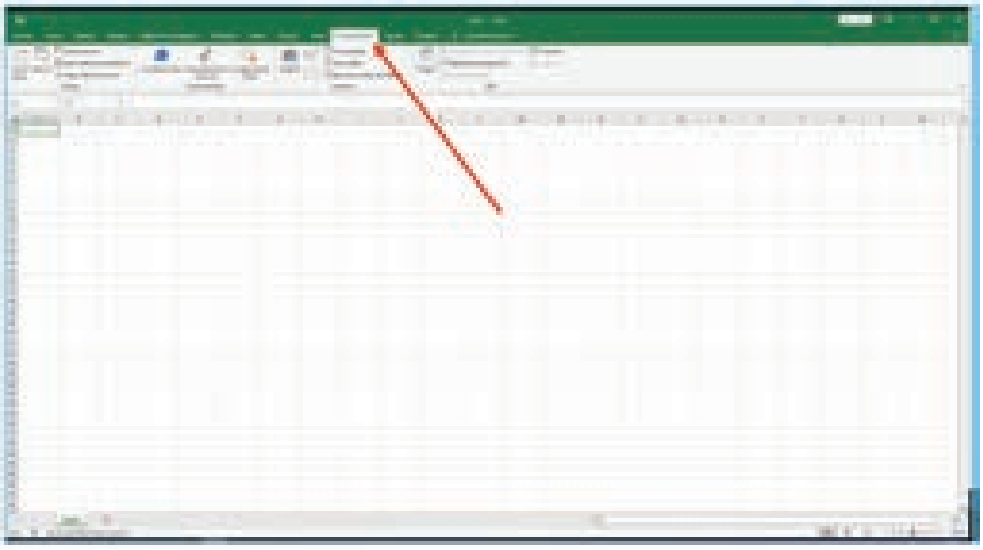
Ten en cuenta que las mismas aplicaciones que ves en este temario, se pueden hacer con otros elementos y con otra filosofía. Pero lo importante es que el resultado resuelva el algoritmo (*conjunto de instrucciones sistemáticas y previamente definidas que se utilizan para realizar y/o solucionar una determinada tarea*) que te hayas planteado para solucionar el problema que se te presenta.

1.2 PERSONALIZANDO LA CINTA DE OPCIONES DE MS EXCEL

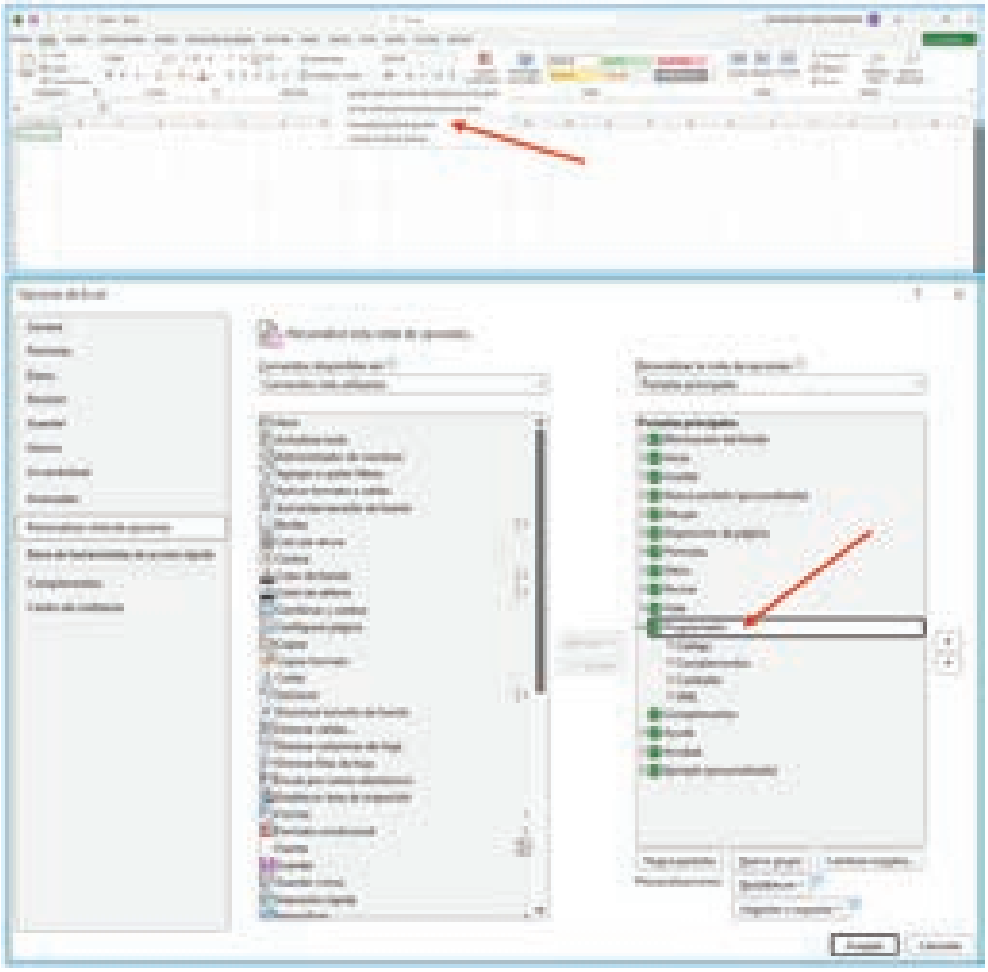
Antes de comenzar a trabajar con VBA Excel, tendremos que activar el **Menú de Programación**. Una vez abierto el **Programa Excel**, con el Botón derecho del ratón sobre la **Cinta de Opciones**, escogeremos **Personalizar la Cinta de Opciones**, activamos la **Pestaña Programador** y aceptamos. Lo vamos a presentar con **ambas versiones de MS Excel**.

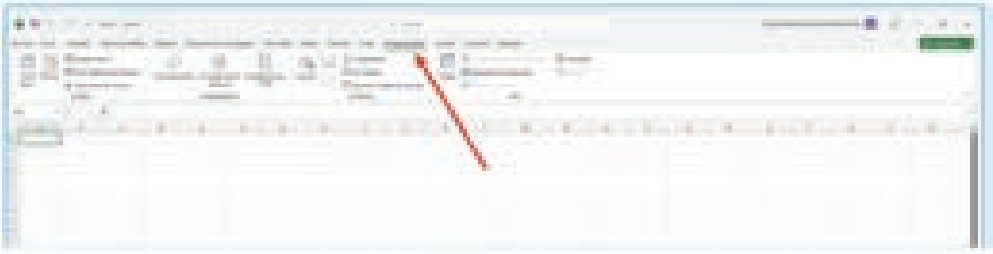
Versión Microsoft Excel 2019





Versión Microsoft Excel 2021





1.3 NUESTRA PRIMERA MACRO EN MS EXCEL

Vamos a comenzar con algo muy sencillo. Haremos nuestra primera Macro que tendrá como misión **sumar varias Celdas**.

> **NOTA**
>
> Antes de comenzar con el ejemplo, te advierto que mostraré pantallas, indistintamente, de **ambas versiones (2019 y 2021)**, siempre y cuando el resultado no varíe.

Las Macros son trozos de Código VBA Excel generados automáticamente. Es realmente fácil crear una Macro, solo se necesita usar la **Grabadora de Macros** y realizar las acciones que queremos repetir más tarde de forma automática.

Incluso cuando seamos usuarios de VBA Excel con buenos conocimientos de programación, es muy aconsejable grabar Macros, de vez en cuando, para conocer cómo se genera ciertos trozos de Código que podamos copiar y usar en nuestros programas, en vez de ir a buscar por Internet cómo se hace. Y esto es un **truco que conviene no olvidar nunca**.

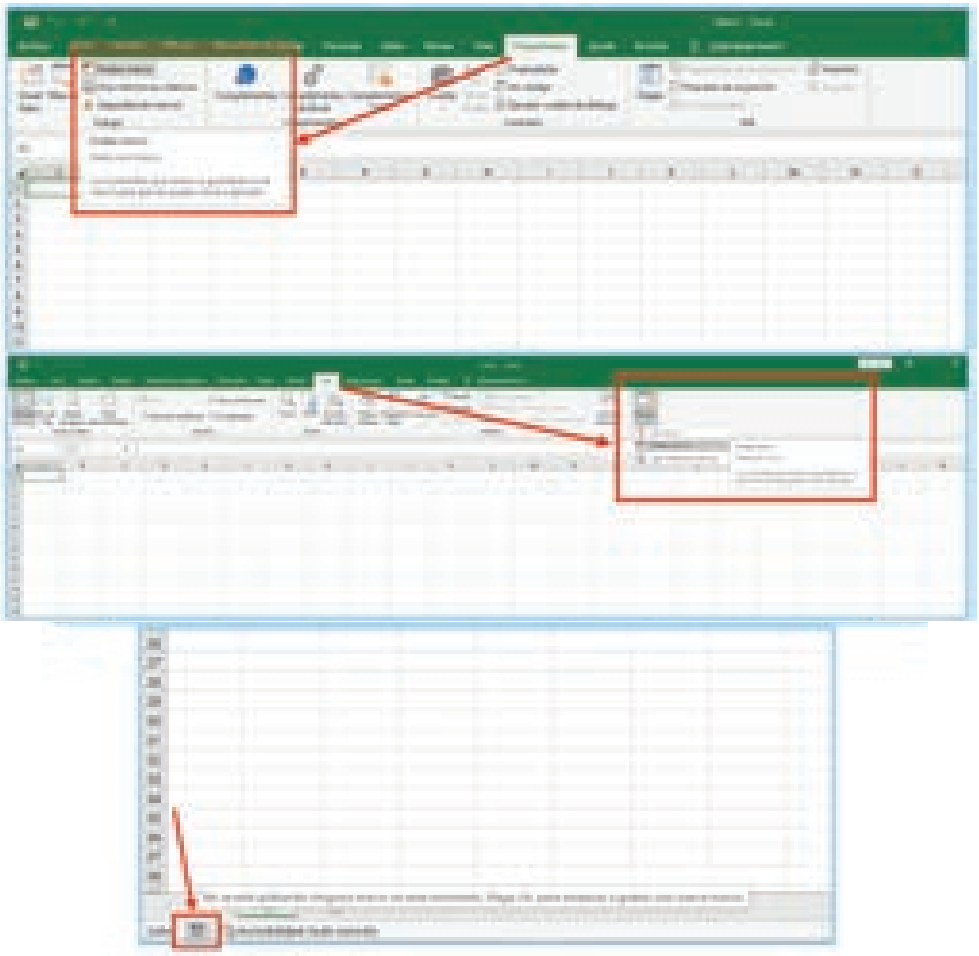
Cuando creamos las Macros o las usamos, en realidad, no tenemos ninguna necesidad de ver su Código y mucho menos de entrar a editarlo.

Debemos tener en cuenta que con las macros
No podemos crear programas
Que generen condicionales o bucles

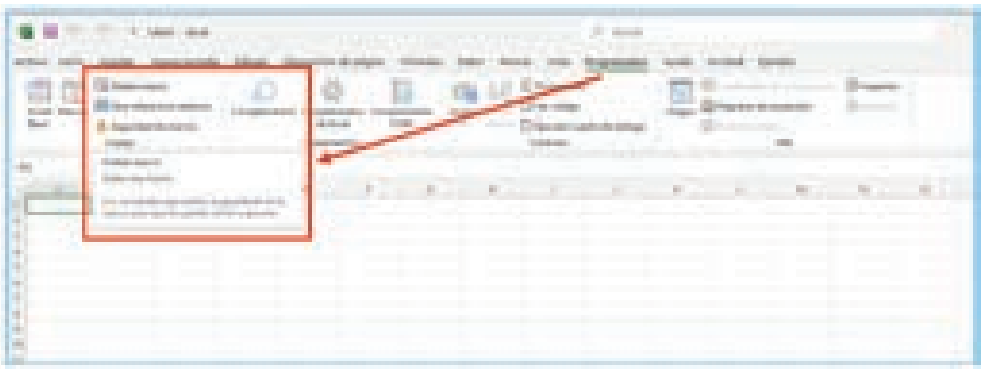
Para usar la **Grabadora de Macros** podemos llegar por varios caminos.

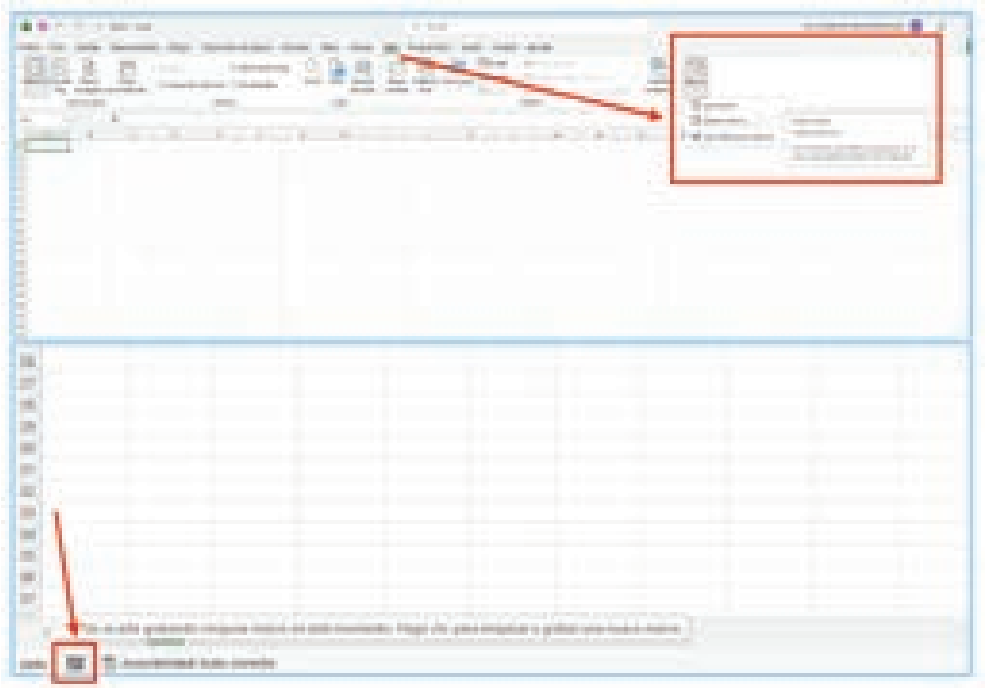
- Ficha Programador.
- Ficha Vista (a la derecha).
- Icono en la Esquina Inferior Izquierda de la Barra de Estado de la Hoja Activa.

Versión Microsoft Excel 2019

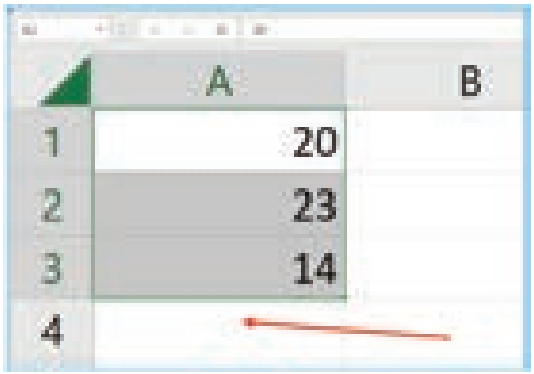


Versión Microsoft Excel 2021





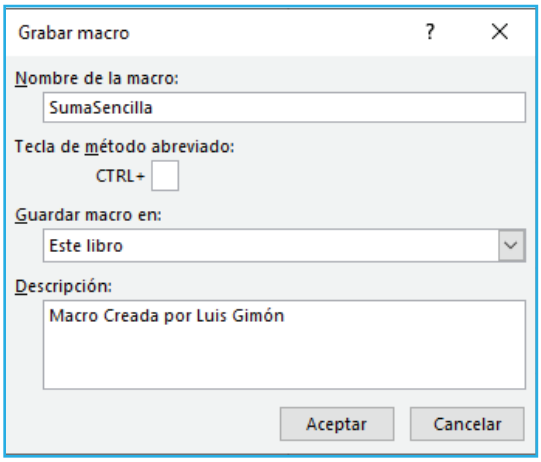
Te propongo comenzar haciendo una Macro que **sume** los siguientes números (**A1:A3**) y que el resultado aparezca debajo (**A4**).

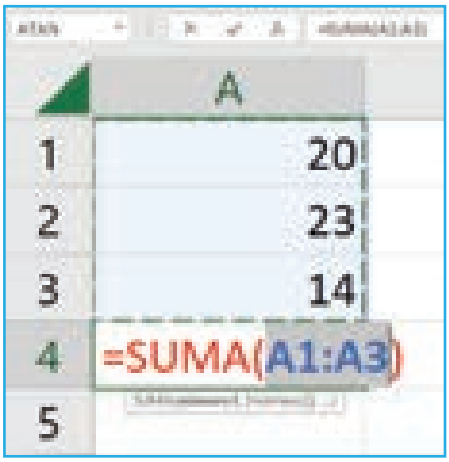


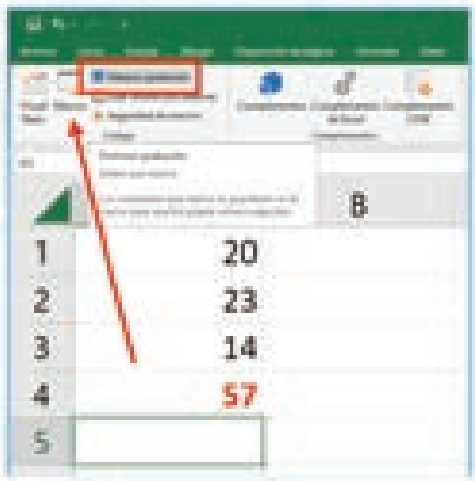
Seguiremos los siguientes pasos **exactamente tal y como te lo indiqué, en este orden y sin saltar ninguno**.

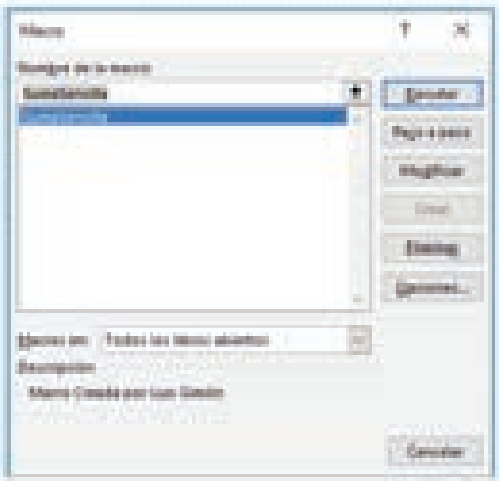
1. Escribimos los **3 números propuestos** para la suma, en las celdas indicadas.
2. Empezamos a grabar la Macro y le damos el nombre: **SumaSencilla**.
3. Guardamos la Macro en "**Este libro**".

4. En la **Descripción**, escribimos "**Macro Creada por Luis Gimón**".
5. Pulsamos **Aceptar**.
6. Seleccionamos la **Celda A4**. (**IMPORTANTE: HACER CLIC SOBRE LA CELDA A4**)
7. Aplicamos **Negrita** y color de fuente **Rojo**.
8. Introducimos la fórmula *=SUMA(A1:A3)*
   - ***Si prefieres, podrías haber escrito algo más sencillo como "=+A1+A2+A3".***
9. Pulsamos **Aceptar**.
10. Detenemos la Macro (Recuerda que la puedes detener en los 3 lugares dónde te indiqué que la podrías empezar a crear).
11. Pulsamos el **Icono Macros** y vemos que tenemos nuestra primera **Macro** (***SumaSencilla***).

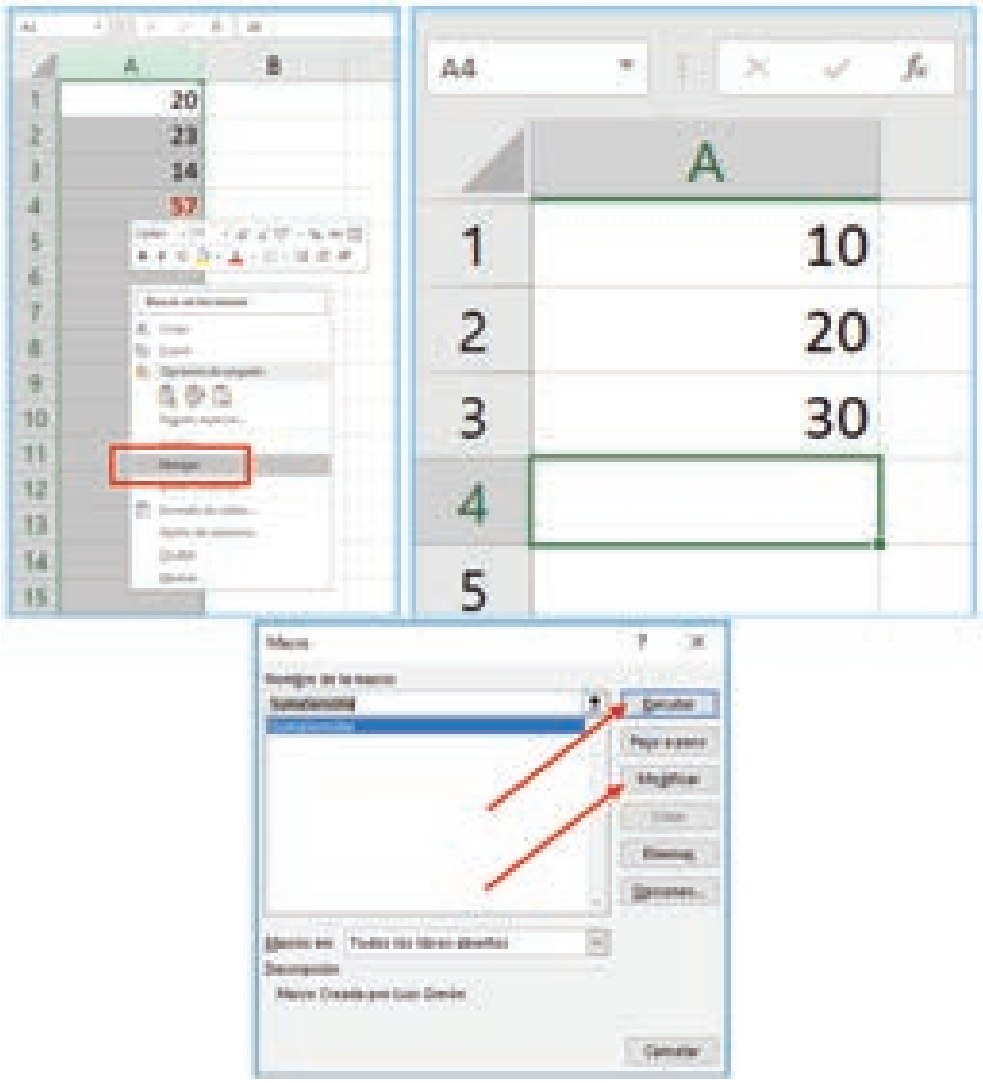






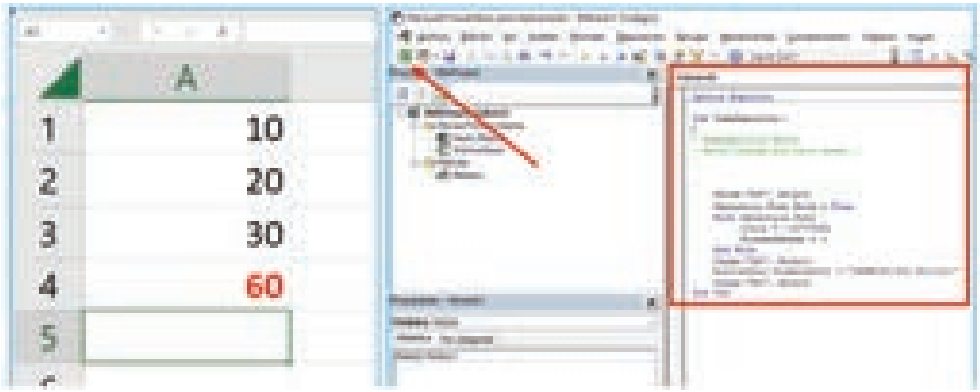


Cerramos el **Cuadro de Diálogo Macro** y **eliminamos la Columna A**, de modo que ya no tenemos ni los 3 números, ni la **Función *Sumar***. Entonces **escribimos 3 nuevos números** y, al terminar, vamos al **Icono Macros** y ejecutamos ***SumaSencilla***.



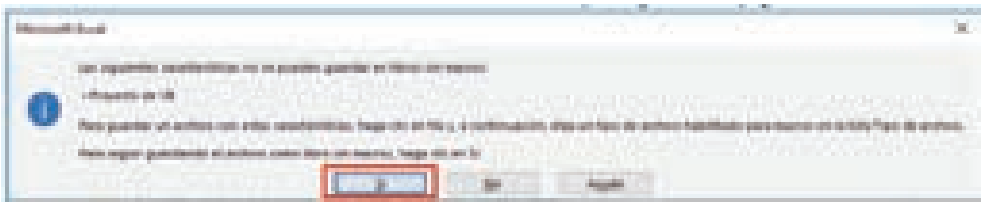
Aparece el resultado en **A4**. O sea, nuestra Macro **siempre** suma los valores que están en el **Rango A1:A3** y lo pone en **A4**, independientemente de cuál sea la Celda seleccionada en ese momento.

Dentro del **Icono Macros**, si pulsamos el **Botón Modificar**, podremos ver el **Código** que hemos generado (***pero no toquemos nada por ahora)***. Pulsa el icono para volver a la Hoja de Cálculo.



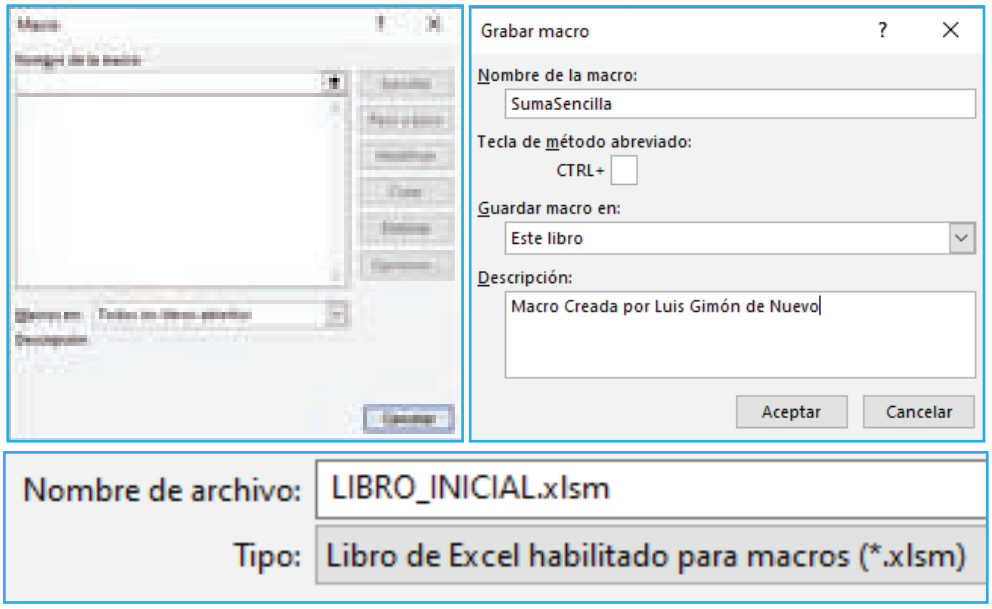
1.3.1 ¿CÓMO GUARDAR NUESTRO LIBRO DE MS EXCEL CON MACROS?

Cuando intentamos guardar el archivo (**Libro de Excel**) que hemos creado con el nombre **LIBRO_INICIAL.xlsx**, recibo el mensaje siguiente y pulso **Sí**.



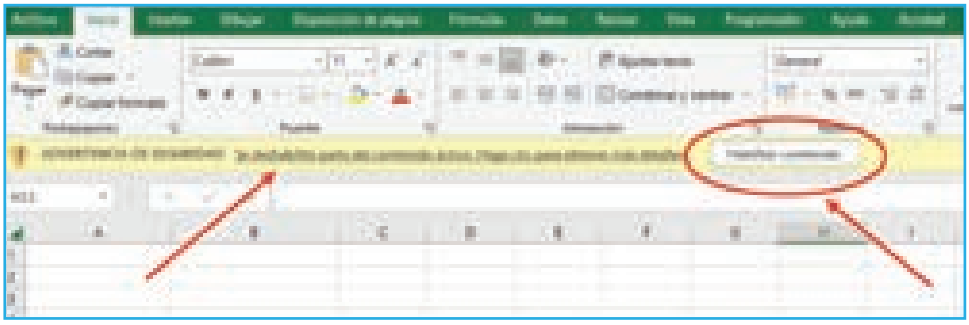
El **Formato de Archivo XLSX no permite almacenar códigos de Macros** de VBA Excel, ni Hojas de Macro de Office Excel 4.0 (.xlm).

Si cierro el archivo y lo abro de nuevo, al intentar ejecutar la Macro **me encuentro que no existe ninguna Macro**. Así que volveremos a repetir los pasos anteriores (***todos y desde el principio***), creando de nuevo la Macro. Pero esta vez, guardaremos el archivo como **LIBRO_INICIAL.xlsm (XLSM–Excel Open XML Macro-Enabled Spreadsheet)**, lo cual **habilita las Macros** en nuestro Libro de Trabajo.

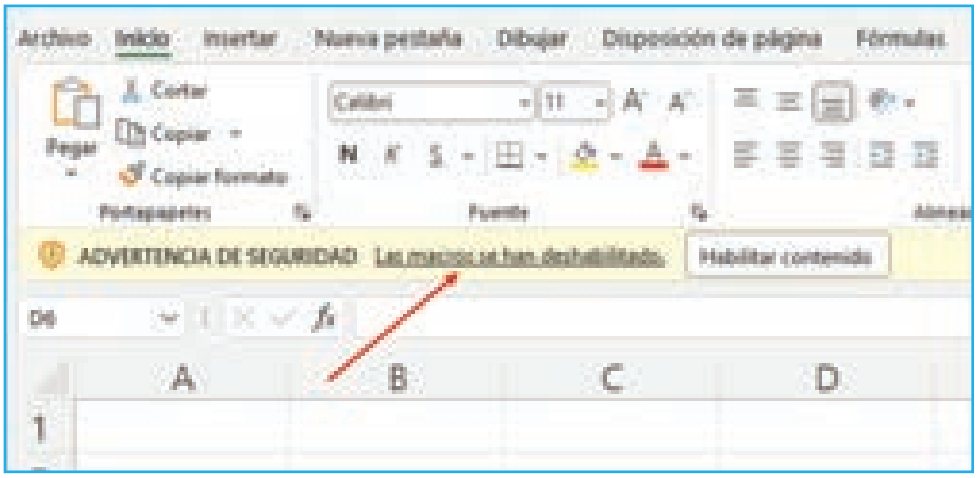


Elimino la Columna A y Cierro y Abro de nuevo el archivo. Entonces recibo el **mensaje de advertencia** que indica que **las Macros están deshabilitadas**. Podríamos habilitar las Macros de este documento, **para siempre**, sin más que pulsar el **Botón Habilitar contenido**. Pero **NO LO PULSARÉ AHORA**, sino que veremos cómo activar las Macros sólo para la **actual sesión de uso del documento**. Esto significará que, cada vez que abramos el documento, nos volverá a preguntar si queremos **Habilitar contenido**. Te lo muestro a continuación en ambas versiones (**2019 y 2021**).

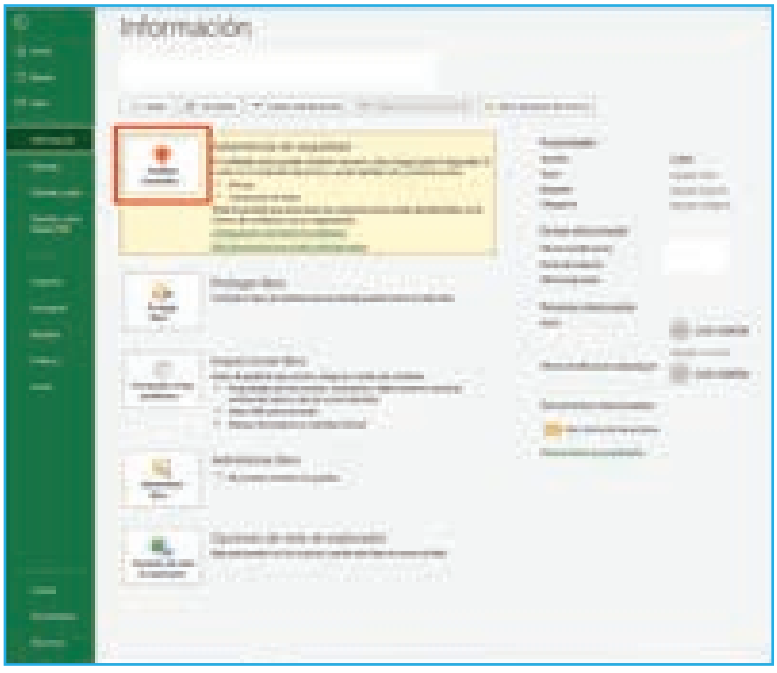
Versión Microsoft Excel 2019

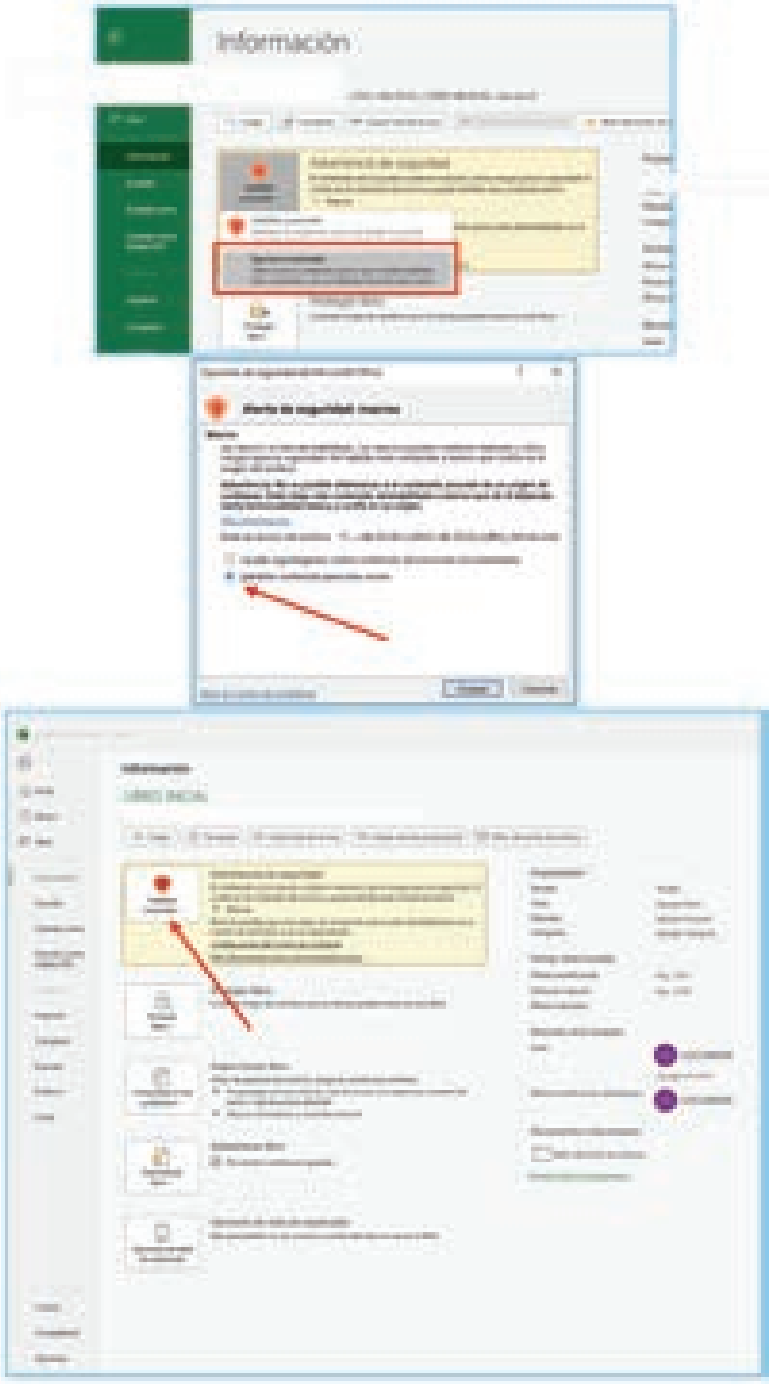


Versión Microsoft Excel 2021



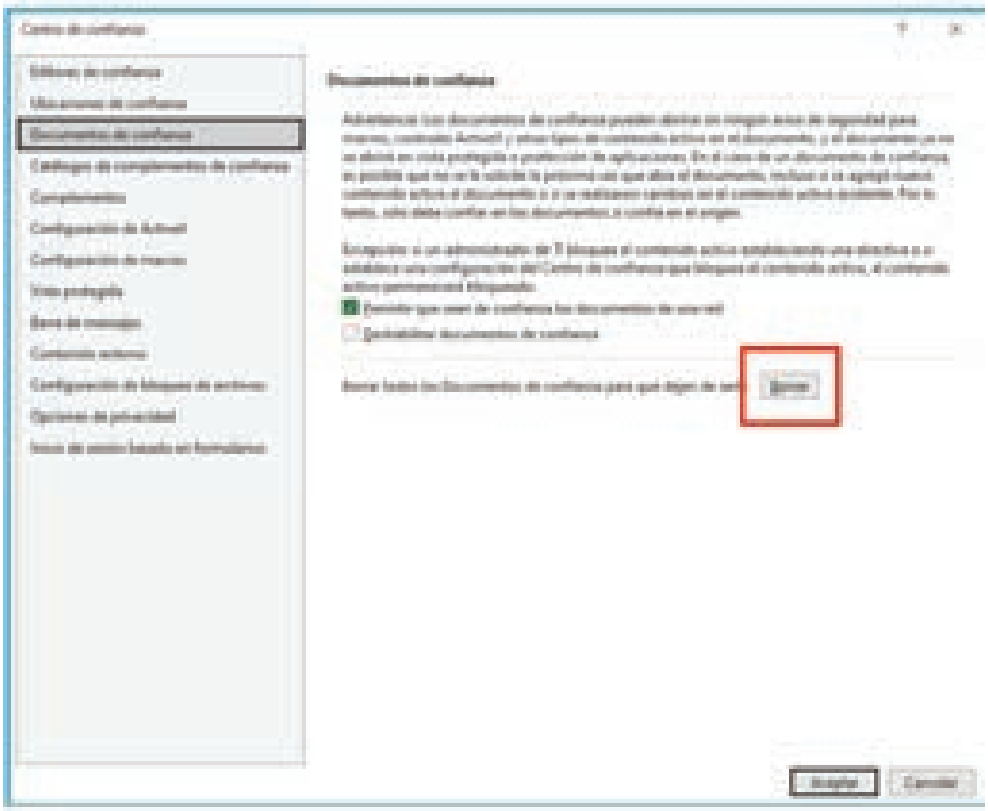
Para que funcione sólo en esta sesión, pulsamos sobre el mensaje de advertencia a la izquierda que nos lleva al siguiente menú. Escogemos **Habilitar contenido/ Opciones Avanzadas** y seleccionamos **Habilitar contenido para esta sesión** y aceptamos.




Información

Cerramos el documento y lo volvemos a abrir. Nos vuelve a bloquear las Macros, pero esta vez pulsaremos **Habilitar contenido**, con lo que se habilitan todas las Macros de este Libro y habremos metido el documento en la **Lista de Documentos de Confianza**. A partir de ahora, si cerramos y volvemos a abrirlo, ya no nos muestra ninguna advertencia.

Si quisiéramos extraer este documento del listado de confianza, tendríamos que ir a ***Opciones de Excel/Centro de Confianza/Configuración del Centro de Confianza/ Documentos de Confianza***. Aquí observamos que tenemos un Botón para ***Borrar todos los Documentos de Confianza*** (**Ojo** Borra Todo).

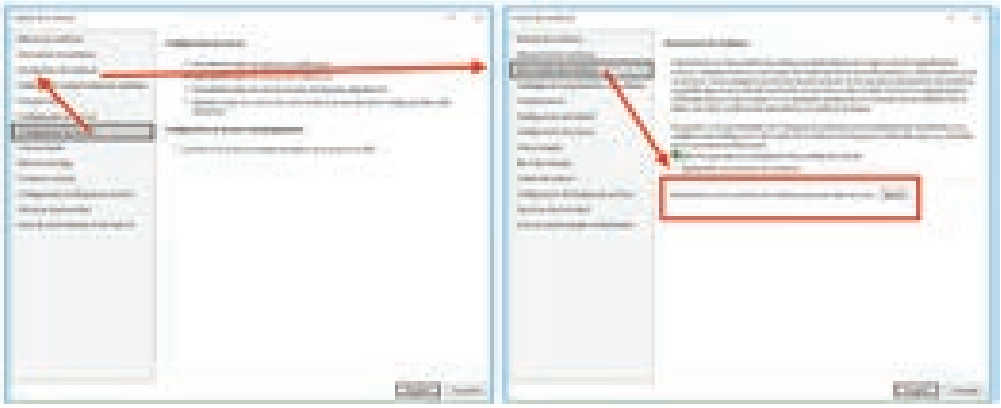


ⓘ NOTA

Si cambiamos el nombre del documento o su ubicación, habría que empezar el proceso de autorización de las Macros porque ya que no es el mismo documento guardado en la **Lista de Confianza**.

NOTA

Quizás sea más rápido acudir a través del **Botón *Seguridad de Macros*** si accedemos por el icono que hay en la **Cinta de Programador**.

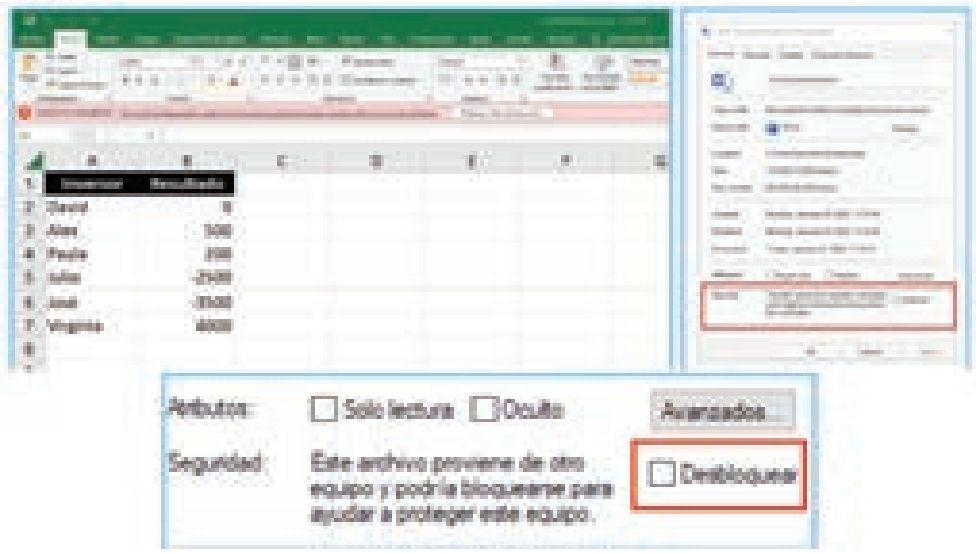


Si activamos la casilla ***Deshabilitar documentos de confianza, ningún documento dispondrá de autorización para abrir las Macros que contenga, sin nuestra autorización expresa***. Podemos comprobarlo y ver que el documento que ya estaba autorizado, al abrirlo de nuevo, nos vuelve a pedir **Habilitarlo**. A partir de ahora, la **Habilitación de Macros sería sólo para cada sesión**. Volvemos a desactivar la casilla y nuestro documento vuelve a activar las Macros.

1.4 DESBLOQUEANDO ARCHIVOS\CARPETAS

Supongamos que nos envían por correo un archivo que contiene Macros y, al intentar abrirlo, nos encontramos con un **mensaje de bloqueo**, que **no nos permite ejecutar las Macros**.

En este caso, antes de abrir el archivo, pulsaremos el Botón derecho del ratón sobre el archivo (Menú Contextual) y escogemos **Propiedades**. En la Pestaña General, veremos una opción de **Seguridad (Security)** que permite **Desbloquear (Unlock)** el archivo. Encontrarás más información al respecto en *https://learn.microsoft.com/es-mx/deployoffice/security/internet-macros-blocked*.

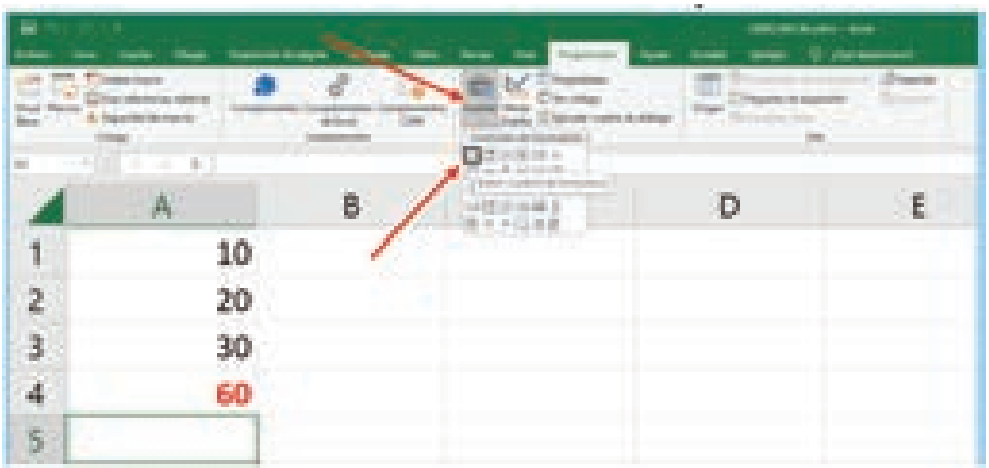


En cambio, **si lo que queremos es desbloquear todos los archivos de una carpeta de confianza en el disco duro del equipo**, podremos agregarla como ubicación de confianza. No olvidemos que tenemos que estar seguro de que son archivos seguros. Para ello, seguiremos los siguientes pasos.

1. Seleccione Archivo > Opciones.
2. Seleccione Centro de confianza > Configuración del Centro de confianza > Ubicaciones de confianza.
3. Seleccione Agregar nueva ubicación.
4. Seleccione Examinar para buscar la carpeta, seleccione una Carpeta (opcionalmente puede incluir las SubCarpetas) y después Aceptar.

1.5 EJECUTAR MACRO INSERTANDO BOTÓN DE FORMULARIO

Vamos a aprovechar que disponemos de una Macro (***SumaSencilla***) en nuestro Libro de Trabajo (**LIBRO_INICIAL.xlsm**). Entramos en la **Pestaña Programador** y elegimos el **Botón** dentro de los **Controles de Formulario**. Cabe decir que los Controles de Formulario están muy limitados, pero por ahora nos vale. Hablaremos con detalle más adelante de este Tipo de Controles.



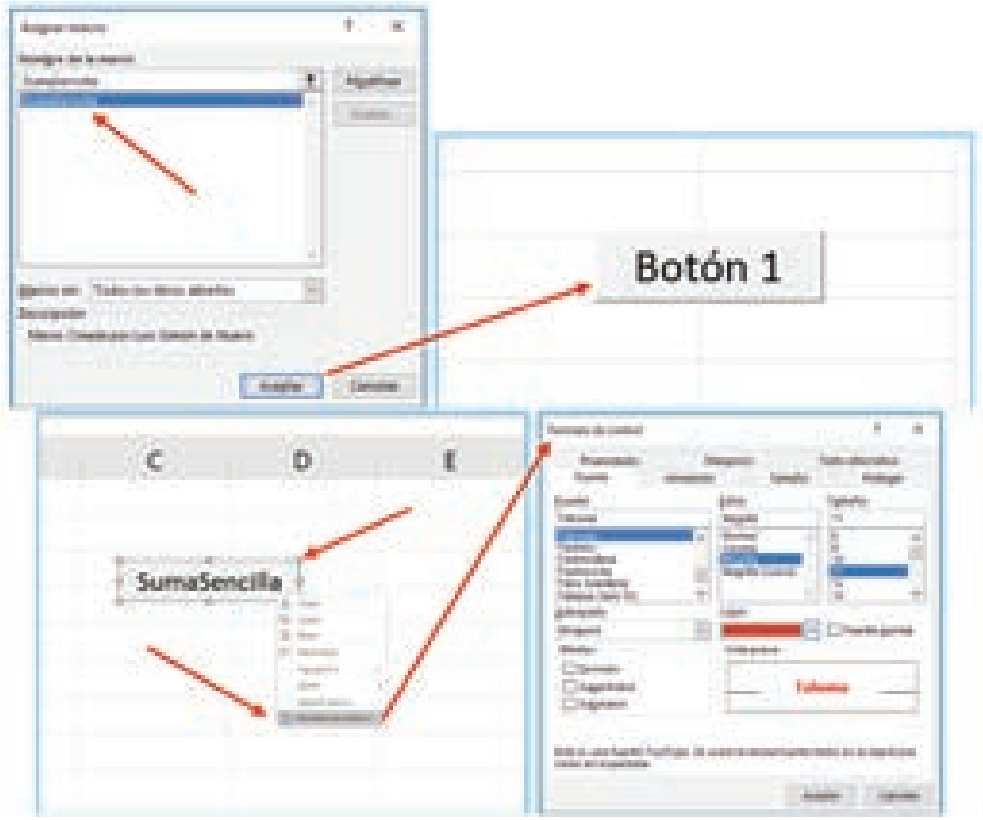
Pulsamos el Botón y elegimos una de las 2 siguientes opciones.

1. Pinchamos en la Hoja y hacemos Click (tamaño estándar del Botón).
2. Pichamos el control Botón y lo arrastramos para darle tamaño personalizado.

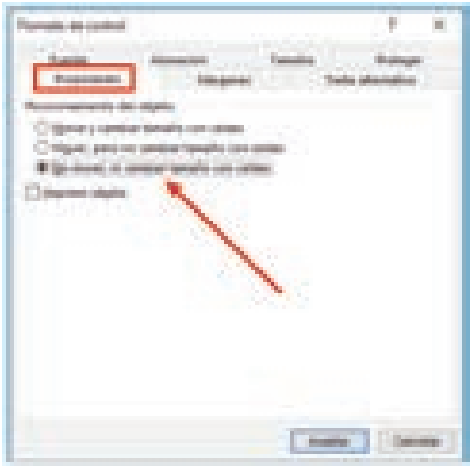
Para crear el Botón **debemos asignarle una Macro y aceptar**. Una vez hecho esto, ya tenemos un Botón que, al pulsarlo, hará la **Suma de las 3 Celdas que hay inmediatamente encima de la Celda Seleccionada**.

Si lo quieres probar, no tienes más que escribir 3 valores en **A1**, **A2** y **A3** y pulsar el Botón asignado a la Macro.

Si pulsas el **Botón derecho del ratón sobre el Botón asignado a la Macro**, lo seleccionas y podrás cambiar el **Texto** del mismo (Click dentro del Botón\opción **Editar texto**), o bien, usar los tiradores en el perímetro del Botón para cambiar el **Tamaño**. Con la opción **Formato de control** del Menú Contextual (Botón derecho del ratón) podrás modificar Propiedades, Fuentes, Tamaño, etc.

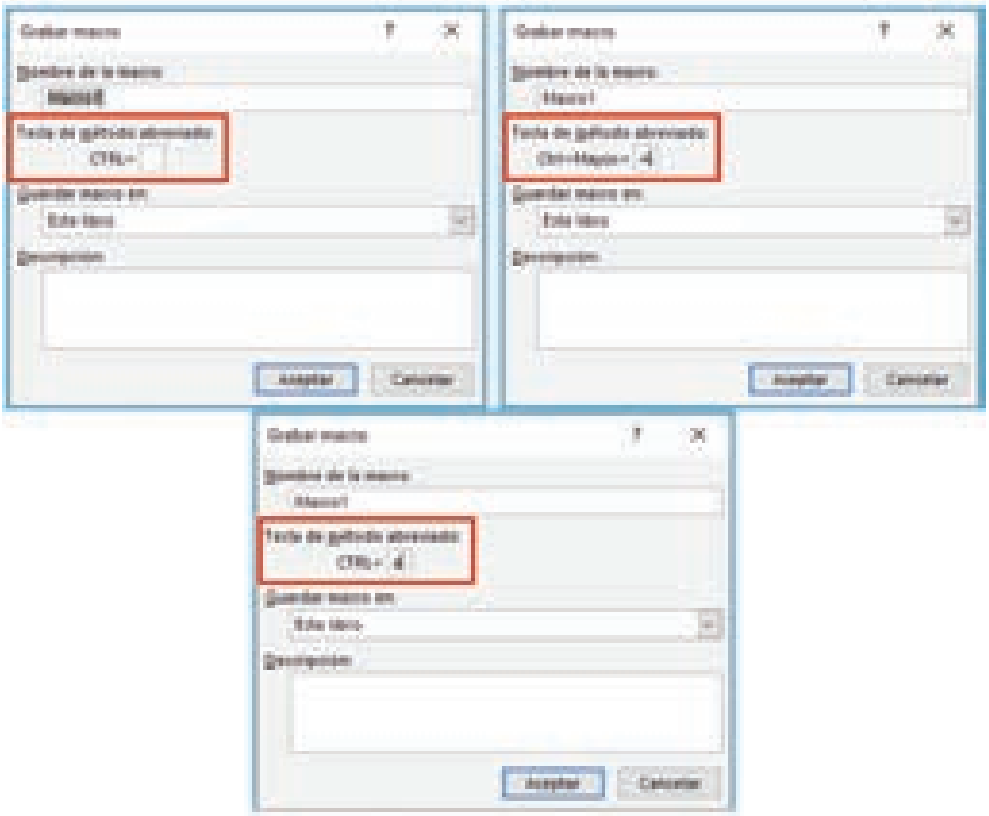


Por defecto, el Botón se moverá si las Celdas alrededor de dónde lo insertamos, se mueven o las eliminamos. Dentro de la **Pestaña *Propiedades***, tenemos una opción para evitar que el Botón se desplace al eliminar Celdas, Filas o Columnas.

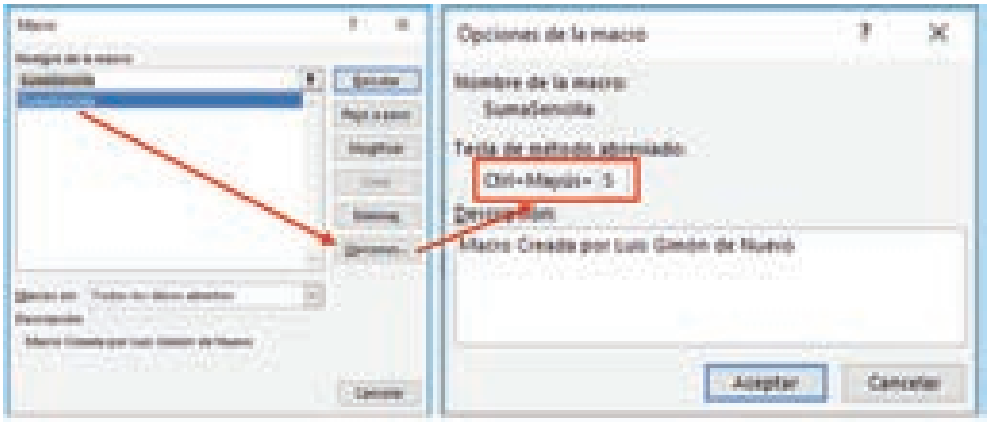


1.6 EJECUTAR MACRO POR COMBINACIÓN DE TECLAS

Vamos a crear una **Combinación de Teclas** que también ejecute la Macro. Cuando creamos una Macro, aparece una opción llamada **Tecla de método abreviado**, escribiremos cualquier letra en minúsculas o mayúsculas que queramos usar como ***Tecla de método abreviado***.



Habrá que tener en cuenta que hay una amplia variedad de combinaciones de teclas del tipo **Ctrl+alguna_tecla** ya **asignada a funciones específicas**. En vez de usar alguna de las combinaciones asignadas, **como consejo**, es preferible usar la combinación de teclas **Ctrl+Mayúsc+Tecla**. En nuestro ejemplo usaremos la combinación **Ctrl+Mayúsc+S** para asignársela a la **Macro *SumaSencilla***, usando el botón ***Opciones***.



Microsoft nos ofrece esta advertencia que conviene tener en cuenta.

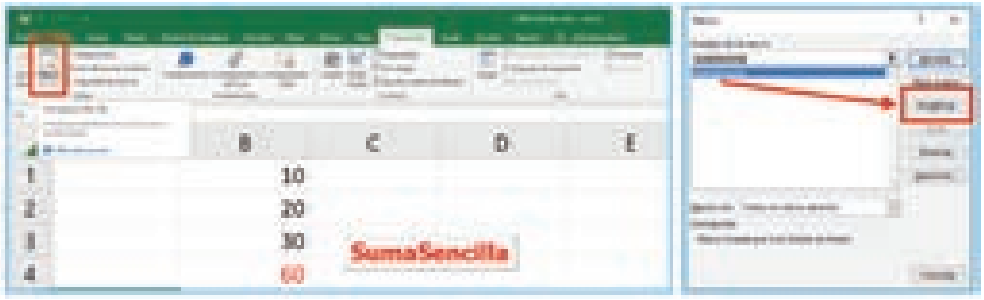
Notas:

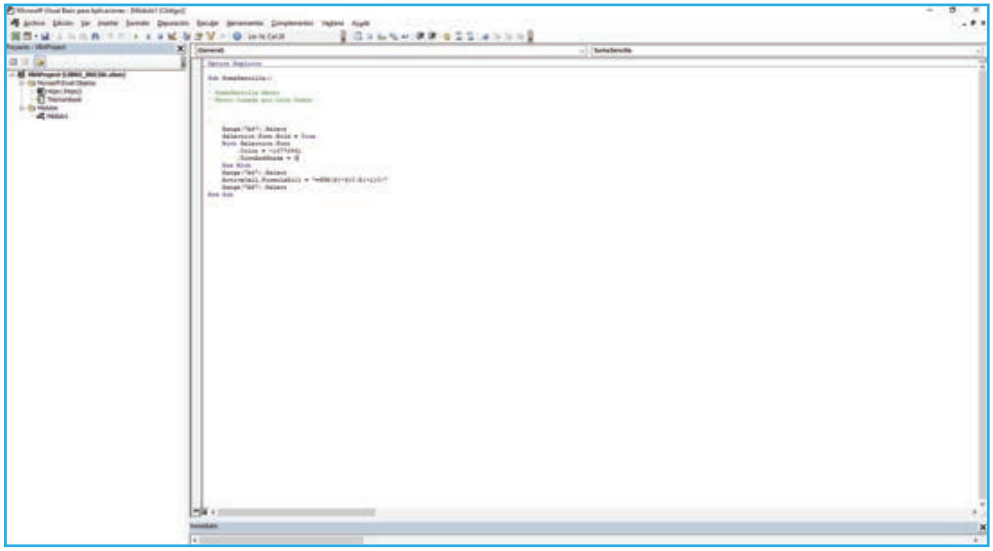
- Para **Windows**, la tecla de método abreviado para las letras en minúsculas es **Ctrl+letra.** Para las letras mayúsculas, es **Ctrl+Mayús+Letra.**
- Para **Mac,** la tecla de método abreviado para las letras en minúsculas es **Opción+Comando+letra,** pero **Ctrl+letra** también funcionará. Para las letras mayúsculas, es **Ctrl+Mayús+Letra.**
- Tenga cuidado al asignar teclas de método abreviado, ya que invalidarán cualquier tecla de acceso directo Excel predeterminada equivalente mientras el libro que contiene la macro está abierto. Por ejemplo, si asigna una macro a **Ctrl+z,** perderá la capacidad de **Deshacer**. Debido a esto, generalmente es una buena idea usar **Ctrl+Mayús+Letra** mayúscula en su lugar, como **Ctrl+Mayús+Z**, que no tiene un método abreviado equivalente en Excel.

Una vez asignada la combinación, volvemos a ejecutar la Macro y comprobamos que el resultado es el mismo que al pulsar el **Botón *SumaSencilla***.

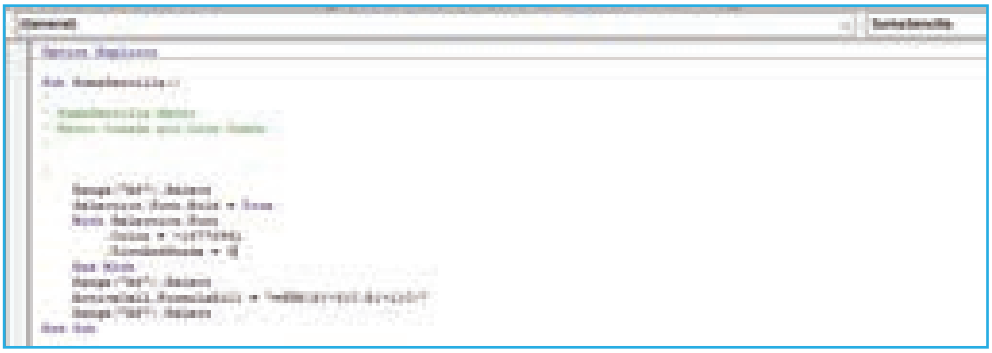
Aunque **ahora no vamos a desarrollar este apartado**, diremos que el **Editor de VBA Excel**, forma parte de la **Suite Microsoft Office** y permite programar aplicaciones Windows para los archivos de Office. Si lo usamos bajo MS Excel, nos permitirá crear y editar Macros. El Editor tiene su propia ventana, separada de la interfaz de MS Excel (Hojas).

Pulsaremos el Botón **Macros**, seleccionamos ***SumaSencilla*** y escogemos el Botón ***Modificar*** para entrar en el **Editor de VBA Excel**. Si te fijas, ahora tienes 2 ventanas abiertas relacionadas con tu Libro de Trabajo de Excel (**LIBRO_INICIAL.xlsm**). Por un lado, está la Hoja de MS Excel y por otro lado aparece el **Editor de Programación VBA Excel**.



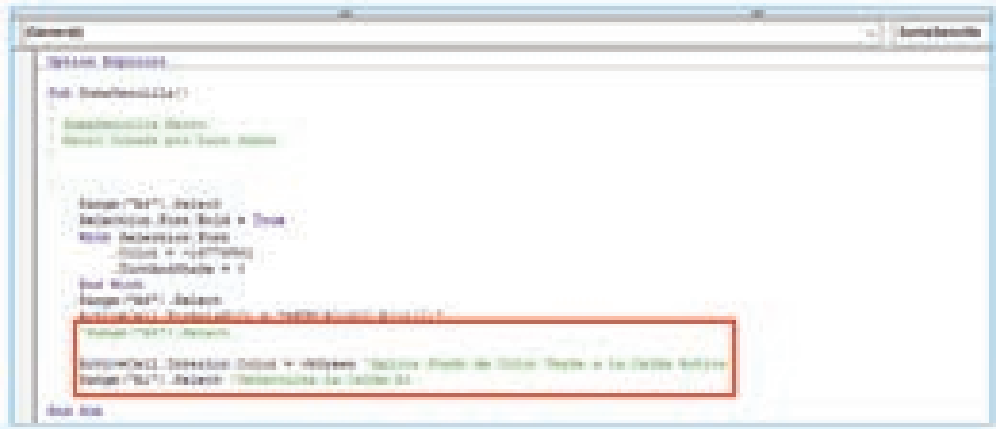


Nos damos cuenta que, para modificar la Macro, hay que conocer el lenguaje **Visual Basic for *Applications* (VBA)**. La Grabadora ha servido para generar un programa, automatizando pasos, pero, para ir un poco más allá, hay que recurrir a la programación pura y dura. Este es el Código que hemos generado con la Macro.



Cambiaremos ahora el Código por este otro. No te preocupes de escribir lo que está en color Verde porque son **comentarios** sobre lo que hace cada línea y no afectan a la ejecución de la Macro.

- ***ActiveCell.Interior.Color = vbGreen 'Aplica fondo de Color Verde a la Celda Activa.***
- ***Range("A1").Select 'Selecciona la Celda A1.***

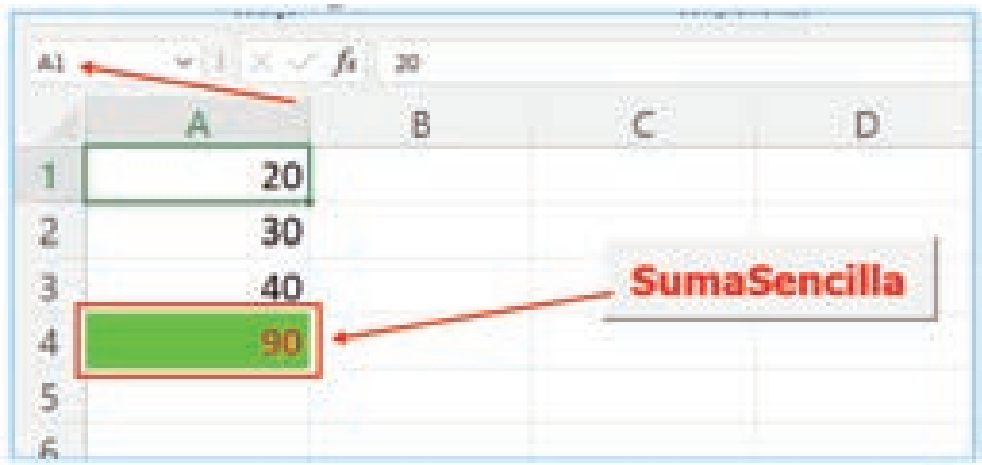


NOTA

No olvides poner un **Apóstrofe (')** al principio de la línea ***Range ("A5").Select***. De esta manera, **no se ejecuta esa línea** porque se convierte en un **Comentario**.

Ahora vuelve a tu Hoja, **escribe 3 números** y pulsa el Botón **SumaSencilla**. Se observa cómo se rellena en color verde el fondo de la Celda que contiene el resultado

de la Suma y la selección termina en **A1**, que es justo lo que indican las 2 últimas líneas del Código.



1.7 MACROS CON REFERENCIAS ABSOLUTAS\RELATIVAS

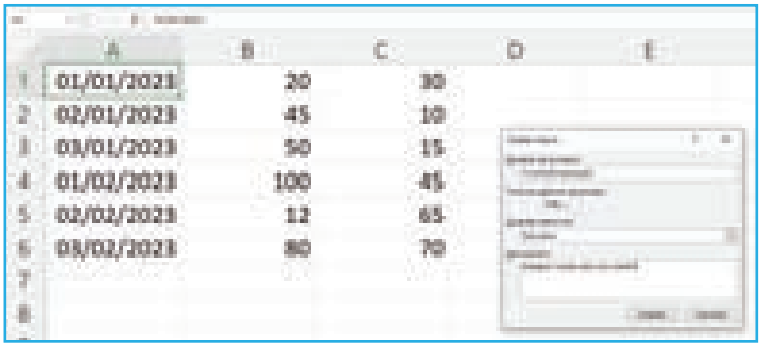
Hasta ahora, hemos grabado las Macros con las **Referencias Absolutas** de las Celdas implicadas. Ya sabemos que una referencia identifica a una Celda (o a un rango de Celdas) de manera única en MS Excel (siempre tendremos al menos una celda seleccionada). Las referencias son como direcciones dentro de cualquier Libro de Trabajo de MS Excel que permite encontrar cualquier celda con la que poder realizar operaciones y cálculos.

Las **Referencias Relativas** guardan una relación con respecto a la **Columna\Fila** donde se encuentre la Selección. Por ejemplo, si copiamos el valor de una Celda que está 3 Columnas a la derecha y 2 Filas por debajo, respecto a la Celda Seleccionada, siempre que hagamos esa operación, desde cualquier otra Celda Seleccionada, se mantendrá la relación, sin variar, cuando trabajamos con Referencias Relativas. Excel se encarga de ajustar inmediatamente la Columna y la Fila correspondiente.

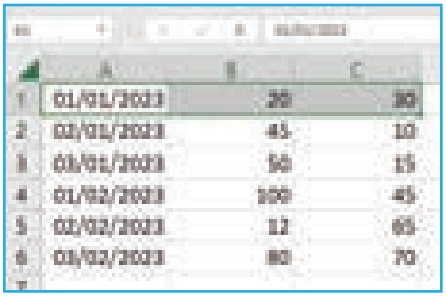
En cambio, las **Referencias Absolutas** permanecen fijas, independientemente de la Celda Seleccionada. Si copiamos el valor de una Celda, con Referencia Absoluta, siempre estaremos haciendo referencia a esa Celda (y no a otra).

Recordemos que se utiliza el **Símbolo $** para convertir en Referencia Absoluta una Celda (ya sea su Columna, su Fila, o ambas a la vez), como por ejemplo ***A1***, ***$B4*** *o* ***C$5***.

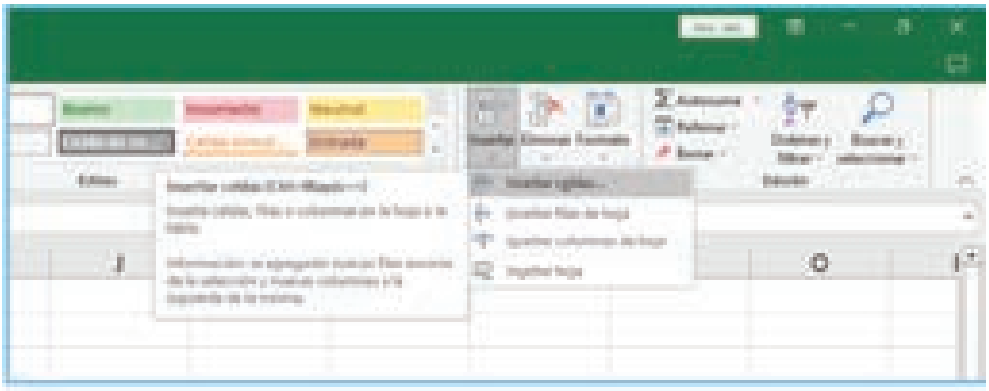
Empezaremos un nuevo Libro de Trabajo y vamos a hacer una nueva Macro que seleccione las **Celdas A1:C1** e **inserte un encabezado (FECHA/VENTA/COMPRA) encima de la selección**.

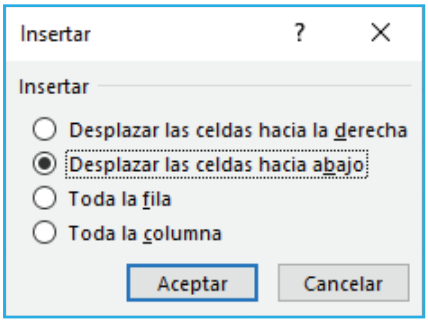


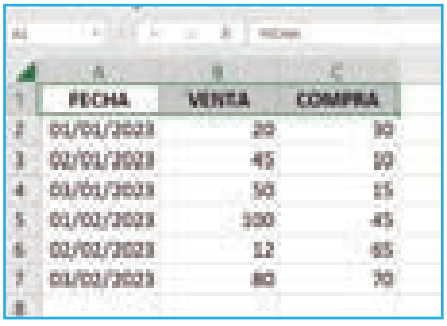
	A	B	C
1	01/01/2023	20	30
2	02/01/2023	45	10
3	03/01/2023	50	15
4	01/02/2023	100	45
5	02/02/2023	12	65
6	03/02/2023	80	70



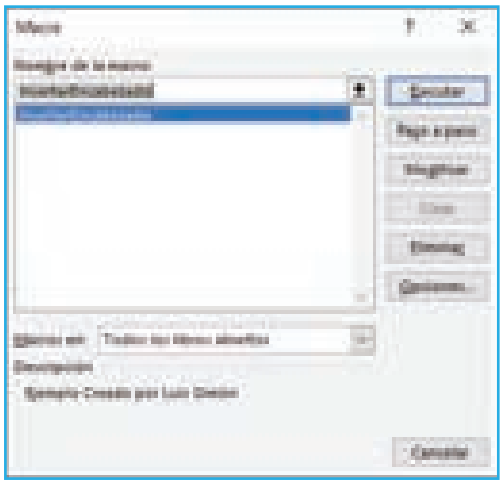
	A	B	C
1	01/01/2023	20	30
2	02/01/2023	45	10
3	03/01/2023	50	15
4	01/02/2023	100	45
5	02/02/2023	12	65
6	03/02/2023	80	70



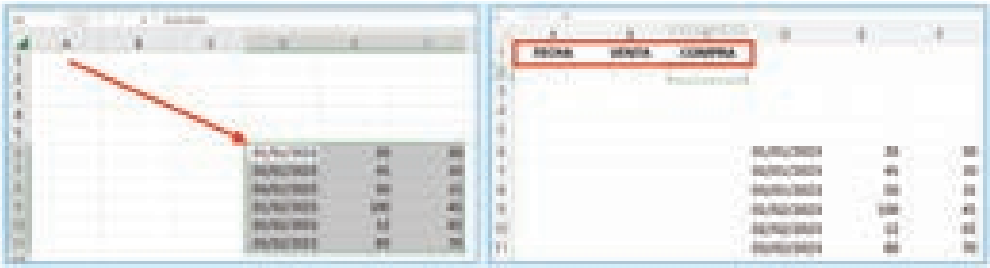




	FECHA	VENTA	COMPRA
2	01/01/2023	20	30
3	02/01/2023	45	10
4	03/01/2023	50	15
5	01/02/2023	100	45
6	02/02/2023	12	65
7	03/02/2023	80	70

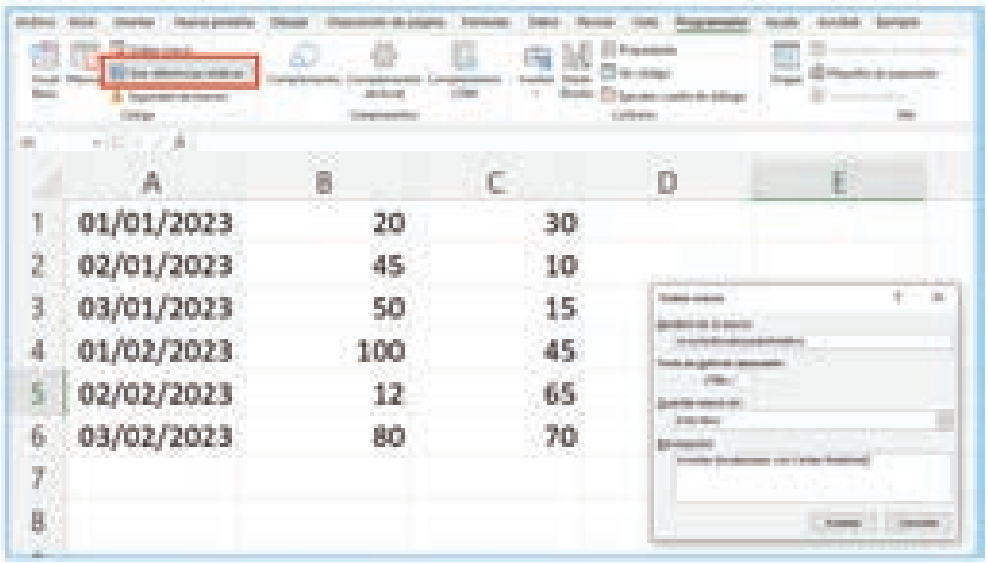


Una vez hecho esto, paramos la Grabación de la Macro. A continuación, eliminamos la primera Fila (Encabezados) y **movemos** la Tabla (Rango de Datos) a otra Celda, procurando no usar las Columnas A, B o C (**para evitar un efecto indeseado**). Yo he trasladado el Rango a **D6**. Seleccionamos el Rango completo o, simplemente, la Celda D6 y ejecutamos la Macro ***InsertarEncabezado***.

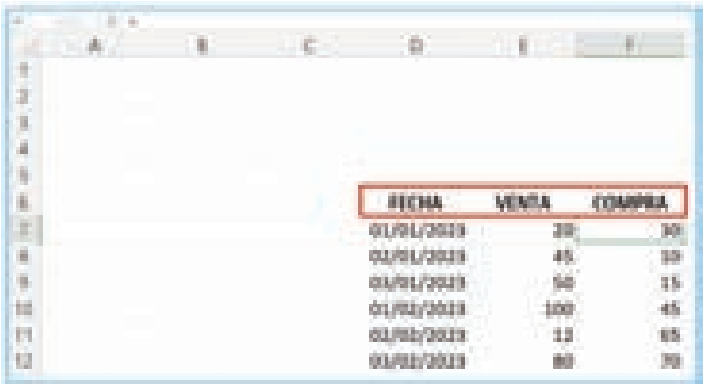


Como vemos, el encabezado se ha insertado en **A1:C1**, pero no se ha colocado sobre la Tabla, **como sería lo deseable**. Si miramos el Código generado, es lógico, ya que observamos que, nada más empezar, se selecciona el **Rango (A1:C1)**. Esto indica que la Macro ha utilizado Referencias Absolutas y es por ello que siempre coloca el encabezado en **A1:C1**.

Volvemos a colocar los Datos como al principio, en **A1**, y creamos una nueva Macro, pero, esta vez pulsaremos la opción **Usar referencias relativas (podemos activarlo antes de empezar a grabar)**. Llamaremos a la Macro con el nombre ***InsertarEncabezadoRelativo***. Seguimos los mismos pasos que antes.



Una vez creada la Macro, volvemos a eliminar la primera Fila y movemos de lugar la Tabla con los Datos (igual que anteriormente), **seleccionamos la Celda D6 y ejecutamos la Macro *InsertarEncabezadoRelativo***. El resultado es que ahora **sí coloca los encabezados donde corresponde**. Lo podría probar colocando la Tabla en cualquier otro lugar de la Hoja y comprobaría que **funciona siempre**. Debajo se añade el Código generado por esta nueva Macro.

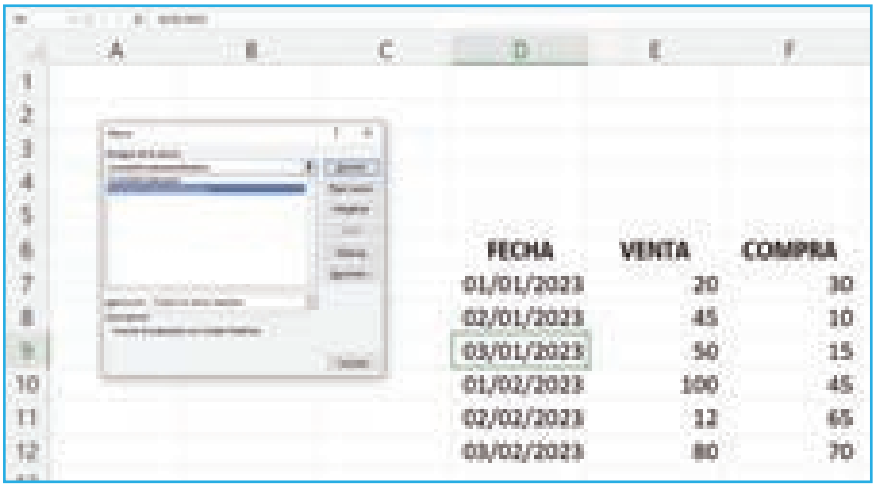


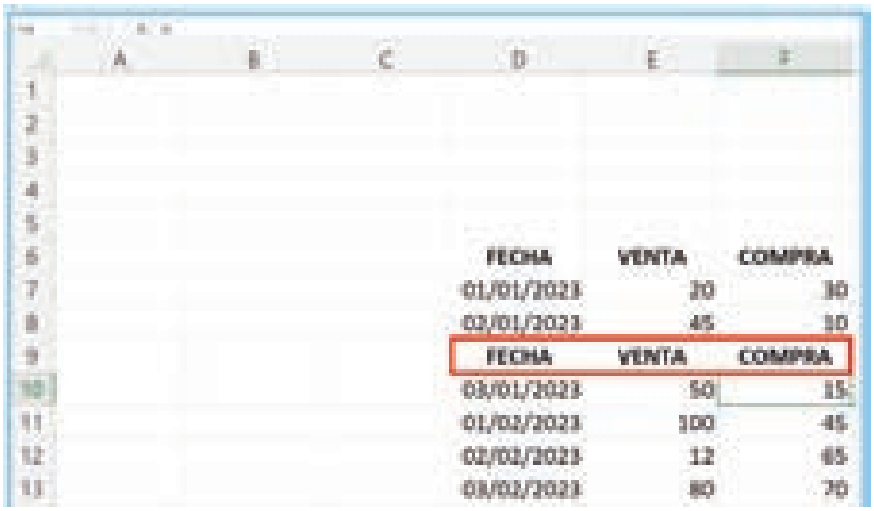

```
Sub InsertarEncabezadoRelativo()
'
' InsertarEncabezadoRelativo Macro
' Insertar Encabezado con Celdas Relativas
'

'
    ActiveCell.Range("A1:C1").Select
    Selection.Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlCenter
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Selection.Font.Bold = True
    ActiveCell.Select
    ActiveCell.FormulaR1C1 = "FECHA"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "VENTA"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "COMPRA"
    ActiveCell.Offset(1, 0).Range("A1").Select
End Sub
```

Ahora seleccionaremos la **Celda D9 (donde hay un cambio de mes)** y ejecutamos de nuevo la misma Macro.

NOTA

Para ejecutar la Macro, ya no es necesario activar **usar referencias relativas**.





Si copias el contenido de la Tabla de Datos **en otra Hoja de este Libro**, podrás ejecutar la Macro (sin olvidar seleccionar previamente la Celda con el cambio de mes) y

funcionará perfectamente. Por tanto, por ahora, las Macros se pueden usar en cualquier Hoja del **Libro de Trabajo actual**.

Aunque adelanto futuras explicaciones, comentaré que el éxito de la Macro se debe a la **Función *Offset*** que permite acceder a Celdas partiendo desde otras Celdas.

La **Función *Offset*** sirve para poder moverse a través de la Hoja de Cálculo, tal y como lo haríamos con las **Flechas de dirección del Teclado** (arriba, abajo, izquierda, derecha). Su configuración es ***Offset* (número de Filas, número de Columnas)**. Veamos algunos ejemplos.

- *ActiveCell.Offset(1, 0).Select* => Avanza una Fila (abajo).
- *ActiveCell.Offset(0, 1).Select* => Avanza una Columna (derecha).
- *ActiveCell.Offset(2, -1).Select* => Avanza 2 Filas (abajo) y retrocede 1 Columna (izquierda).
- *ActiveCell.Offset(0, 0).Select* => Se mantiene en la Celda activa.

ⓘ NOTA

No nos debe confundir, ni preocupar el Comando siguiente que aparece en el Código.

ActiveCell.Offset(0,1).Range("A1").Activate

Nuestros datos están recogidos en el Rango **D6:F11**, pero, al insertar el espacio para el Encabezado, nuestro Rango sería **D6:F12**. O sea, el Encabezado ocupa las **Celdas D6, E6 y F6**.

En este caso, no se trata de avanzar una Columna a la derecha y activar la **Celda Absoluta A1 de nuestra hoja**. Lo que hacemos es avanzar una Columna a la derecha (en la misma Fila) y activar la Primera Celda de la Selección (o Rango) que esté establecido (podría ser el Rango **D6:F12**). En este ejemplo, como el Rango es una Celda (**D6**), lo que hace es avanzar a la **Celda E6** y posicionarse en esa propia Celda, lo cual se podría haber hecho, simplemente, con el siguiente Comando.

ActiveCell.Offset(0,1).Activate

ⓘ IMPORTANTE

Uno de los efectos que tienen las Macros, una vez ejecutadas, es que **sus acciones no se pueden Deshacer**.

Observación. Para no generar dudas, ni en este apartado, ni más adelante. Cuando he hablado de **Tabla de Datos**, no me refería al **Concepto de Tablas de Excel** (que veremos y usaremos al final de presente documento). Se trata simplemente de un **Rango de Celdas con Datos** con los que he trabajado, pero que, coloquialmente, me he referido a dicho Rango como Tabla.