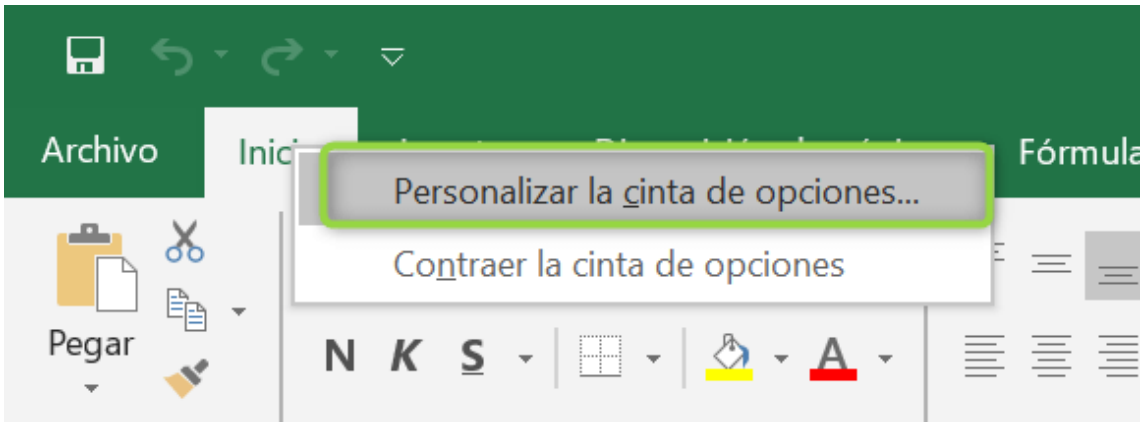
¿Qué es VBA Excel?

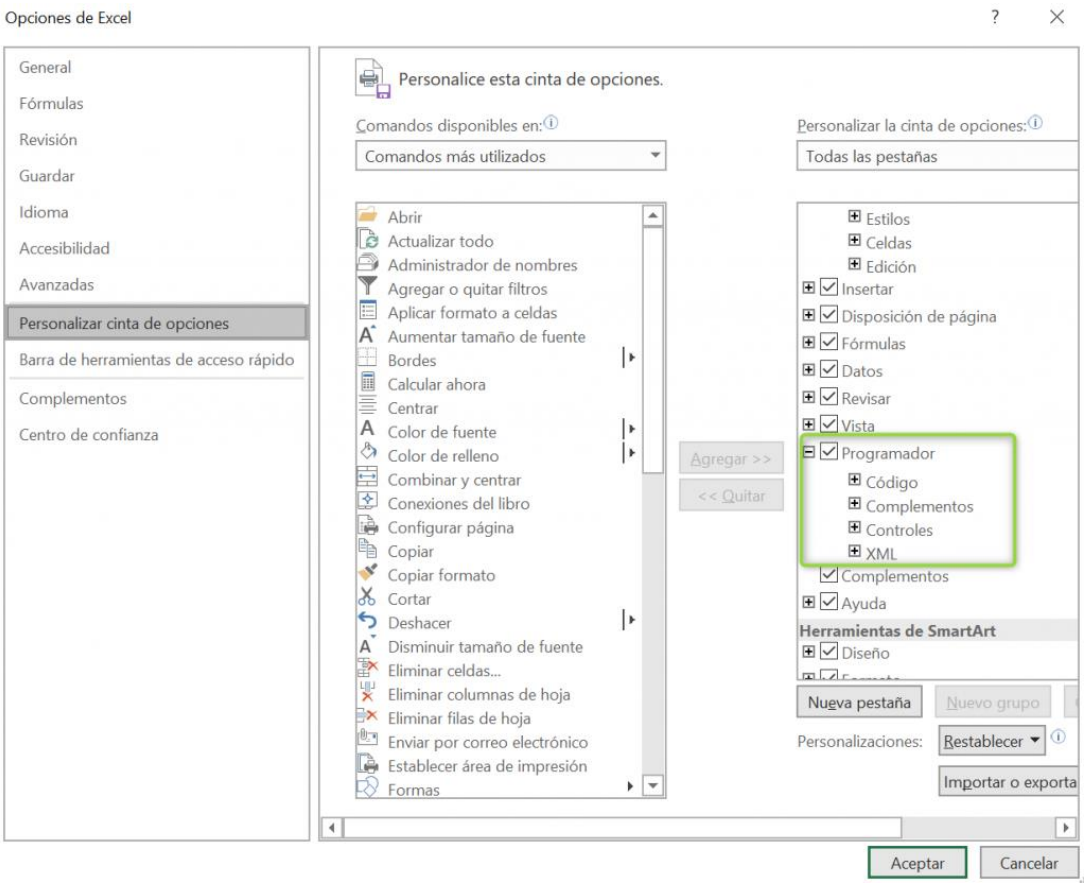
VBA es el lenguaje de programación utilizado por Microsoft para crear aplicaciones en Windows. Viene integrado en la suite de Office: Outlook, Word, Power Point…etc. A continuación, vamos a ver cómo empezar a utilizarlo para crear pequeños programas en Excel.

Editor de Visual Basic Application en Excel

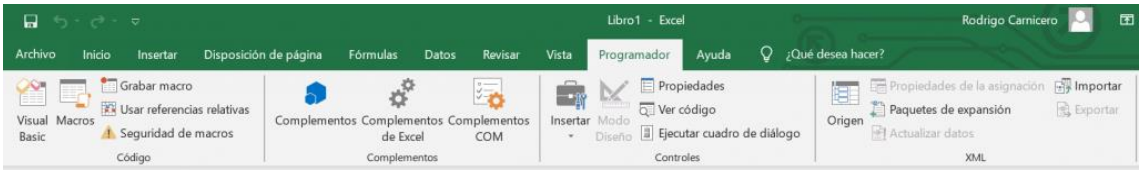
Para poder empezar a programar en VBA necesitamos habilitar la pestaña Programador. Para hacerlo, haz clic derecho en cualquier elemento del menú superior y haz clic en *Personalizar la cinta de opciones*.



Llegarás a una ventana donde debes hacer clic en *Personalizar cinta de opciones* (en el menú de la izquierda) y seleccionar a la derecha la opción *Programador*.



Hecho esto ya te aparecerá el menú con las opciones necesarias para crear tus primeros programas y macros.



Variables

Las variables son la representación de un valor o un objeto. Se declaran utilizando *Dim*, ejemplo:

```
1
2Dim nombreVariable As tipodatos
```

Tipos de datos en VBA

Tipo	Datos que contiene
Boolean	True o False
Byte	Nº enteros entre 0 y 255
Integer	Nº enteros entre -32768 y 327687
Long	Nº enteros entre -2147483648 y 2147483647
Single	Nº decimales de precisión simple
Double	Nº decimales de precisión doble
String	Cadenas de caracteres
Date	Fechas

Tipo	Datos que contiene
Object	Referencias a objetos
Variant	Cualquier tipo de dato

Siguiendo con el ejemplo anterior, podríamos declarar una variable así:

```
1
2Dim strDominio As String
```

Expresiones u operadores

Las expresiones en VBA aritméticas, relacionales o lógicas. A su vez, existen operadores que permiten componer este tipo de expresiones. Los operadores son:

OPERADOR	OPERACIÓN QUE EFECTÚA
Aritméticos	
+	Suma
–	Resta

OPERADOR	OPERACIÓN QUE EFECTÚA
\	División entera
Mod	Resto de la división entera
*	Multiplicación
/	División
^	Potencia del primer operando
Relacionales	
=	Comprueba si los dos operandos son iguales
<>	Comprueba si los dos operandos son distintos
<=	Comprueba si el primero es menor o igual que el segundo

OPERADOR	OPERACIÓN QUE EFECTÚA
>	Comprueba si el primero es mayor que el segundo
>=	Comprueba si el primero es mayor o igual que el segundo
Lógicos	
And	Devuelve true si los dos son true
Or	Devuelve true si uno de los dos, o los dos son true
Xor	Devuelve true si uno de los dos es true y el otro false
Eqv	Devuelve true si los dos son true o false
Not	Devuelve true si el operando es false y viceversa

Sub rutinas

Los procedimientos Sub siempre comienzan con la palabra reservada Sub, el nombre de la macro (cada macro debe tener un nombre único) y un par de paréntesis (estarán vacíos a menos que el procedimiento utilice uno o más argumentos).

La instrucción End Sub señala el final del procedimiento. Las líneas que se encuentran en medio son el código del procedimiento.

Un ejemplo de sub rutina:

```
Sub FechaActual()
    'Inserta la fecha actual en la celda activa al hacer clic en un botón
    ActiveCell.Value = Date
    ActiveCell.NumberFormat = "mmmm d, yyyy"
    ActiveCell.Font.Bold = True
    ActiveCell.Columns.AutoFit
End Sub
```

Funciones

Una función siempre devuelve un único valor (del mismo modo que una función de hoja siempre devuelve un único valor). Una función VBA puede ejecutarse desde otro procedimiento VBA o usarse en fórmulas de hoja, igual que usaría las funciones de hoja integradas en Excel.

Como ejemplo, el script que creé para extraer el dominio de una url:

```
Function rcStrDomain(url As String) As String

    'Comprobamos si el parámetro url es verdaderamente una url
    If Left(url, 4) Like "http*" Then
           'Extraemos todo lo que va después de //
           url = Mid(url, InStr(url, "//") + 2)
        Else
           'Si no es una url, lo introducimos en la celda
```

```
            rcStrDomain = "No es una url"
            Exit Function
    End If
    'Comprobamos si el subdominio contiene las tres típicas www (o variaciones)
    If Left(url, 4) Like "[Ww][Ww][Ww0-9]." Then
        'De ser así, extraemos lo que va después de www (o variaciones)
        url = Mid(url, 5)
    End If
    'Finalmente, la función devuelve la cadena de texto hasta que se encuentra el primer /
    rcStrDomain = Split(url, "/")(0)

End Function

```

Condicionales

Las sentencias condicionales en Visual Basic funcionan de forma muy parecida a otros lenguajes de programación. Encontramos dos tipos de sentencias de ramificación, una de las cuales tiene tres variaciones:

La sentencia If simple

```
If condición Then
    sentencias
End If
```

Añadir la sentencias Else

```
If condición Then
    sentencias1
Else
    sentencias2
End If
```

La sentencia ElseIf

```
If condición1 Then
    sentencias1
ElseIf condición2 Then
    sentencias2
Else
    sentencias3
End If
```

La sentencia Select Case

```
Select Case expresión
    Case expresión1
        sentencias1
Case expresión2
        Sentencias2
……
    Case expresiónN
        sentenciasN
    Case Else
        sentencias para la alternativa no coincidente con ninguna anterior
End Select
```

Bucles

Los bucles son la solución para cuando necesitamos repetir una serie de instrucciones un número de veces para resolver un problema.

Existen tres tipos de bucles, uno de los cuales tiene cuatro variedades:

Bucle While ... Wend

```
'Mientras se cumpla una condición
While condición
        'Ejecuta una acción
    sentencias
Wend

'Ejemplo
Dim Contador
Contador = 0 ' Valor inicial.
While Contador < 20 ' Mientras sea menor de 20.
 Contador = Contador + 1 ' Incrementa su valor +1.
Wend ' El bucle termina cuando el Contador > 19.
MsgBox Contador ' Mensaje con el valor de Contador, igual a 20.
```

Bucle Do ... Loop

```
'Ejecuta una acción
Do
Sentencias
'Mientras se cumpla una condición
Loop While/Until condición

'Ejemplo
Dim index As Integer = 0
Do While index <= 10 'Mientras el valor sea menor igual que 10
    Debug.Write(index.ToString & " ") 'Convierte el valor a string y añade un espacio en bl
    index += 1 ' Aumenta el valor del contador +1
Loop

Debug.WriteLine("")
' Resultado: 0 1 2 3 4 5 6 7 8 9 10
```

Bucle For ... Next

```
'Repite una acción un nº limitado de veces. Desde un valor hasta otro.
For variable = expresión1 To expresión2 [Step expresión3]
        'Ejecuta una acción
    Sentencias
Next variable
```

Objetos

Excel cuenta con decenas de objetos que representan a todos los elementos que podamos imaginar: estilos, nombres definidos para celdillas y rangos, gráficos, libros, hojas, rangos, ventanas, complementos y proyectos de Visual Basic son algunos de ellos.

El objeto que actúa como raíz de todos los demás, del cual dependen, es **Application.**

El libro

Cada libro de Excel, que hasta hemos utilizado en calidad de usuarios, es la representación de un objeto Workbook. Partiendo del objeto Application podemos acceder tanto al libro actual, con la propiedad ActiveWorkbook, como a la colección de todos los libros abiertos, con la colección Workbooks.

La hoja

Cada libro contiene una o más hojas de cálculo, cada una de las cuales tiene correspondencia directa con un objeto **Worksheet.** A través de este objeto, con sus propiedades y métodos, podemos acceder a todo el contenido que exista en la hoja: celdillas, gráficos, elementos visuales como los diagramas, tablas dinámicas, etc.

Los rangos

Un rango puede ser una celda, una fila, una columna o una agrupación de cualquiera de éstos. El objeto Range es probablemente el objeto más frecuentemente utilizado en Excel VBA .

¿Cómo hacer referencia a rangos en otras hojas?

```
1 Worksheets(“Hoja1”).Range(“A1”)
```

¿Cómo utilizar la propiedad Cells para seleccionar un rango?

```
1 Cells(5, “C”)
```

¿Cómo utilizar la propiedad Offset para hacer referencia a un rango?

```
1 Range(“A1”).Offset(4, 5)
```

Desplazamientos

INDICADOR	DESPLAZAMIENTOS
xlDown	Hacia abajo
xlUp	Hacia arriba
xlToRight	Hacia la derecha
xlToLeft	Hacia la izquierda