

TEMA 6

PROGRAMACIÓN DE MACROS DE EXCEL UTILIZANDO VISUAL BASIC FOR APPLICATIONS



Contenido

1. INTRODUCCIÓN A LAS MACROS DE EXCEL	2
1.1 LA GRABADORA DE MACROS	3
1.2 SEGURIDAD DE MACROS.....	7
2. PRINCIPIOS BÁSICOS DE PROGRAMACIÓN EN VBA	8
2.1 PROGRAMACIÓN ORIENTADA A OBJETOS	8
2.2 EL EDITOR DE VISUAL BASIC.....	12
2.3 TIPOS DE ERRORES EN VBA	15
3. PROGRAMACIÓN EN VBA.....	17
3.1 VARIABLES	17
3.2 ENTRADA DE DATOS Y EMISIÓN DE RESULTADOS.....	18
3.3 FUNCIONES.....	21
3.4 ESTRUCTURAS DE CONTROL DE FLUJO	25
4. FORMULARIOS Y CONTROLES	38
ANEXO:.....	50

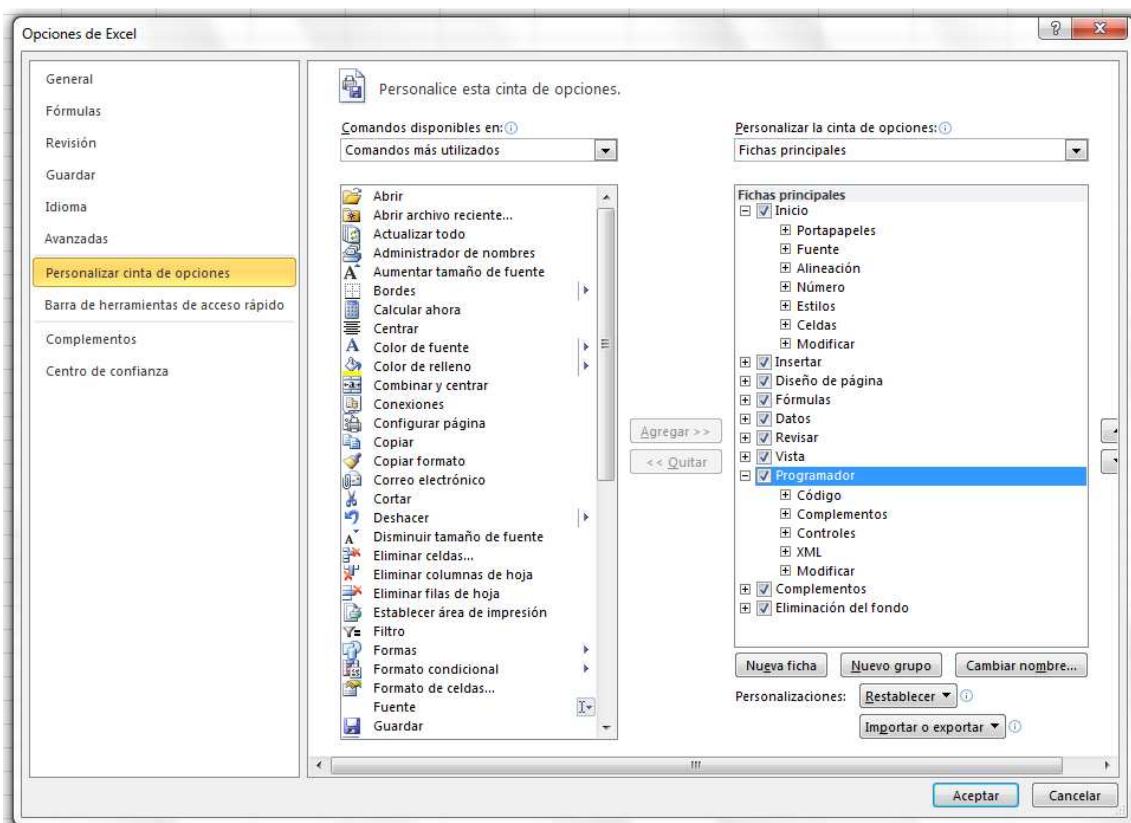
1. INTRODUCCIÓN A LAS MACROS DE EXCEL

Una macro es un conjunto de comandos que se almacenan en Excel de manera que están siempre disponibles cuando se necesita ejecutarlas. Las macros se utilizan principalmente para evitar tener que repetir los pasos de aquellas tareas que se realizan una y otra vez.

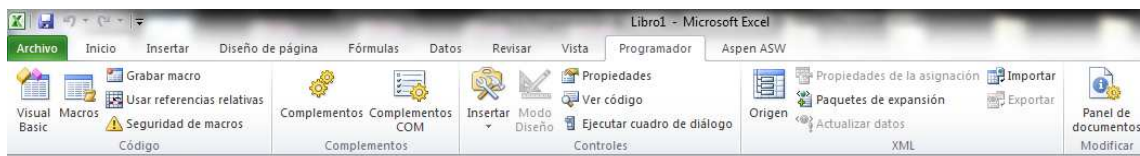
Las macros se escriben en un lenguaje de programación que se denomina *Visual Basic for Applications* (VBA). Aunque no es necesario conocer a fondo este lenguaje para crear una macro, ya que Excel contiene una herramienta especial de programación, manejar este lenguaje permite acceder a todas las funcionalidades de Excel y ampliar las posibilidades del programa.

Como para cualquier otro lenguaje de programación, hay que aprender a utilizar los comandos que le son propios, de manera que la aplicación pueda ejecutar las tareas programadas.

Las macros se pueden crear de dos maneras, utilizando un Editor de Visual Basic o utilizando la Grabadora de Macros de Excel. Ambas opciones se ejecutan desde la Ficha Programador de la cinta de opciones. Al igual que ocurría con el Solver, la Ficha Programador no aparece por defecto al abrir la aplicación, sino que es necesario activarla. Para ello, desde la Ficha Archivo, hay que seleccionar Opciones → Opciones de Excel → Personalizar la cinta de opciones → seleccionar la opción Programador:



La Ficha Programador tiene el siguiente aspecto:

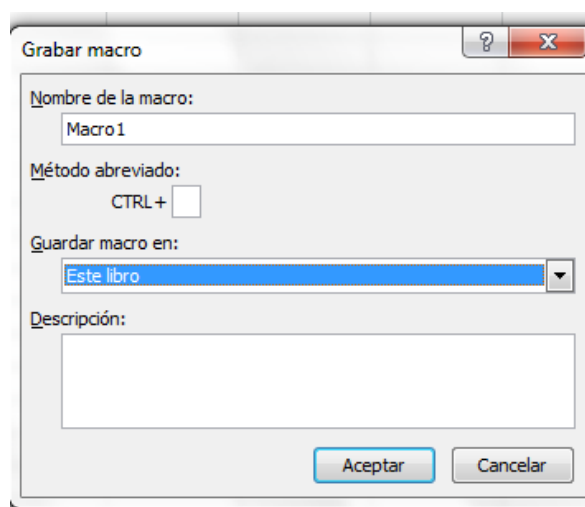


- El Grupo Código contiene los comandos necesarios para inicial el Editor de Visual Basic donde se puede escribir directamente código VBA. También permite ver la lista de macros disponibles para poder ejecutarlas o eliminarlas. También contiene el comando Grabar Macro, que permite crear una nueva macro sin necesidad de conocer programación en VBA.
- El Grupo Complementos permite habilitar y administrar complementos como el Solver.
- El Grupo Controles permite agregar controles especiales a una hoja de Excel, tales como botones, casillas de verificación, botones de opciones, etc.
- El Grupo XML permite trabajar con ficheros XML.
- El Grupo Modificar contiene únicamente el comando Panel de Documentos.

1.1 LA GRABADORA DE MACROS

La Grabadora de Macros es la forma más sencilla de crear una macro. Consiste básicamente en grabar todos los pasos que el usuario ejecuta para resolver un problema y seguir ese mismo esquema para resolver problemas similares en el futuro. Por tanto, al utilizar la grabadora es conveniente planificar previamente los pasos que se van a seguir y no realizar acciones innecesarias mientras se realiza la grabación.

La grabadora se activa con el comando Grabar macro. Al pulsar el botón se abre un cuadro de diálogo en el que hay que introducir el nombre de la macro y la ubicación donde se desea guardar¹. Una vez creada una macro, el comando se transforma en Detener grabación.



¹ Si la macro se guarda en un libro nuevo puede ser ejecutada desde cualquier libro creado durante la sesión actual de Excel. Si se guarda en un libro de macros personal, la macro se podrá utilizar en cualquier momento sin importar el libro de Excel que se esté utilizando.

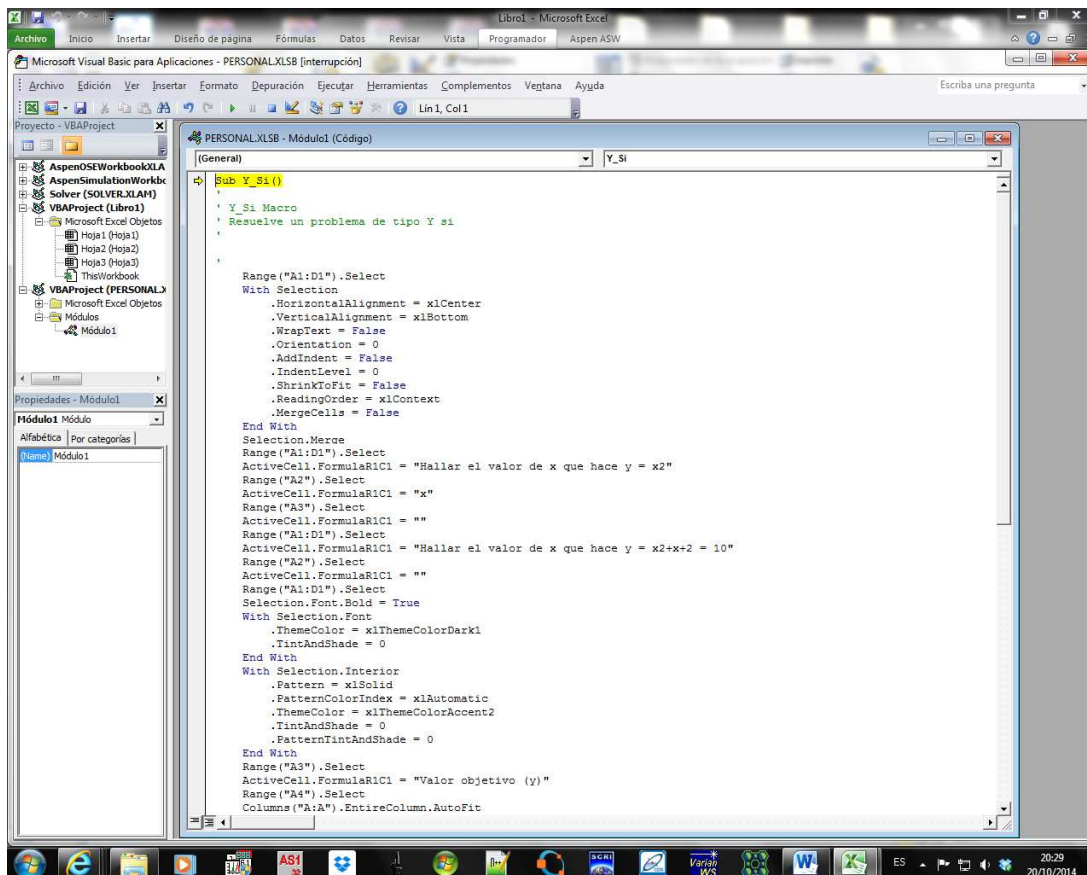
Ejemplo 1: Grabar una macro sencilla.

Una macro puede ser cualquier conjunto de comandos, desde cambiarle el color a una celda, hasta introducir una función y resolverla. En este ejemplo, se ha grabado un ejercicio de tipo Buscar Objetivo igual que uno de los que se hizo en el tema 2. Se ha cambiado el formato de la celda del título para combinar y centrar un conjunto de celdas, cambiar la fuente a negrita en blanco y el fondo a rojo, se ha escrito el título y los rótulos de las casillas, se ha recuadrado el conjunto de casillas y se ha utilizado el comando Buscar objetivo para resolver el problema. La macro se ha guardado con el nombre Y_Si en el libro personal de macros.

El resultado al ejecutarla con el comando Macros es el esperado:

Hallar el valor de x que hace $y = x^2+x+2 = 10$	
Valor objetivo (y)	10,0003829
Variable independiente (x)	2,37234797

Para ver el código exacto que contiene esta macro, en lugar de ejecutarla se puede abrir para modificar. Esto abre el Editor de Visual Basic:



Podemos ver en detalle cada uno de los pasos que incluye el programa:

Sub Y_Si() Se le pide crear una macro con el nombre Y_Si

' Y_Si Macro

' **Resuelve un problema de tipo Y si** Esta es la descripción que se ha incluido al crearla

Range("A1:D1").Select Selecciona las celdas A1 a D1

With Selection Con esas celdas

.HorizontalAlignment = xlCenter centra el contenido tanto horizontalmente

.VerticalAlignment = xlBottom como verticalmente

.WrapText = False

.Orientation = 0

.AddIndent = False

.IndentLevel = 0

.ShrinkToFit = False

.ReadingOrder = xlContext

.MergeCells = False

Esto son más
opciones de alineado,
como añadir una
sangría o ajustar el
texto a la celda, que
por defecto entiende
que no hay que hacer

End With termina con las opciones de alineado

Selection.Merge combina las celdas seleccionadas

Range("A1:D1").Select selecciona la nueva celda combinada como celda objetivo

ActiveCell.FormulaR1C1 = "Hallar el valor de x que hace $y = x^2 + x + 2 = 10$ " y escribe este texto

Selection.Font.Bold = True Sobre la celda seleccionada hace que la fuente sea negrita

With Selection.Font sobre esa celda

.ThemeColor = xlThemeColorDark1 cambia el color

.TintAndShade = 0 para que sea blanco

End With termina con las opciones de la fuente

With Selection.Interior para la celda seleccionada cambia el color del fondo

.Pattern = xlSolid para que toda la celda esté rellena

.PatternColorIndex = xlAutomatic de uno de los colores automáticos

.ThemeColor = xlThemeColorAccent2 en concreto el rojo oscuro

.TintAndShade = 0 sin sombras ni bordes de celda

.PatternTintAndShade = 0 ni ningún patrón de relleno

End With termina con el relleno de la celda

Range("A3").Select para la celda A3

ActiveCell.FormulaR1C1 = "Valor objetivo (y)" escribe este texto

Range("A5").Select para la celda A5

ActiveCell.FormulaR1C1 = "Variable independiente (x)" escribe este texto

Columns("A:A").EntireColumn.AutoFit amplía el ancho de la columna A hasta ajustar el

texto

Range("C3").Select a la celda C3

ActiveCell.FormulaR1C1 = "=R[2]C^2+R[2]C+2" le da el valor de la siguiente función

Range("A1:D5").Select sobre este rango de celdas

Selection.Borders(xlDiagonalDown).LineStyle = xlNone

Selection.Borders(xlDiagonalUp).LineStyle = xlNone

With Selection.Borders(xlEdgeLeft)

.LineStyle = xlContinuous

.ColorIndex = 0

.TintAndShade = 0

.Weight = xlMedium

End With

With Selection.Borders(xlEdgeTop)

.LineStyle = xlContinuous

.ColorIndex = 0

.TintAndShade = 0

.Weight = xlMedium

End With

With Selection.Borders(xlEdgeBottom)

.LineStyle = xlContinuous

.ColorIndex = 0

.TintAndShade = 0

.Weight = xlMedium

End With

With Selection.Borders(xlEdgeRight)

.LineStyle = xlContinuous

.ColorIndex = 0

.TintAndShade = 0

.Weight = xlMedium

End With

Selection.Borders(xlInsideVertical).LineStyle = xlNone

Selection.Borders(xlInsideHorizontal).LineStyle = xlNone

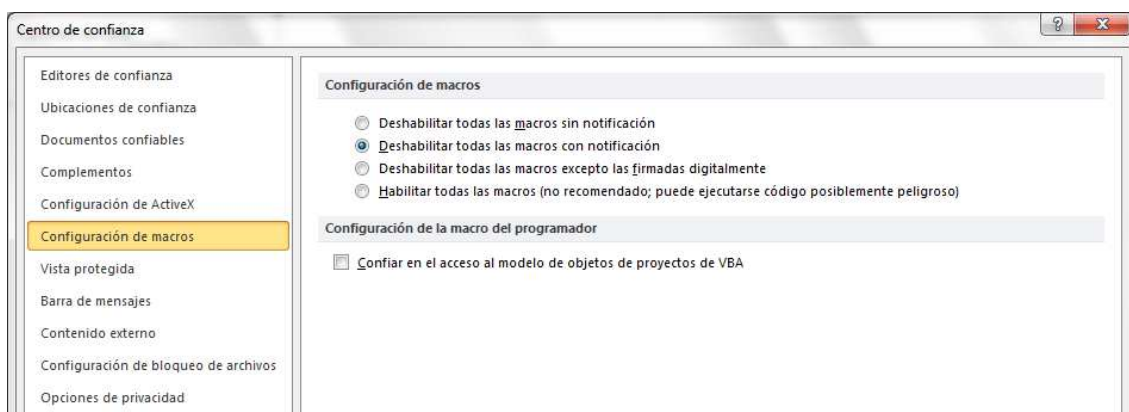
Range("C3").GoalSeek Goal:=10, ChangingCell:=Range("C5") para la celda C3 establece, con el buscador de objetivos, un valor buscado de 10, a base de modificar la celda C5, que es donde se ha referido la x al introducir la función

End Sub fin de la macro

crea un recuadro negro grueso alrededor a base de crear las líneas correspondientes, verticales y horizontales, arriba, abajo, a la derecha y a la izquierda de la selección

1.2 SEGURIDAD DE MACROS

De manera predeterminada Excel no permite ejecutar macros automáticamente. Así se evita que al abrir un archivo que no ha creado el propio usuario se ejecute una macro con código malicioso. No obstante, esta configuración se puede modificar desde la Ficha Archivo → Opciones → Centro de confianza → Configuración del centro de confianza. Se abre un cuadro de diálogo en el que se puede elegir, para el submenú Configuración de macros, lo siguiente:

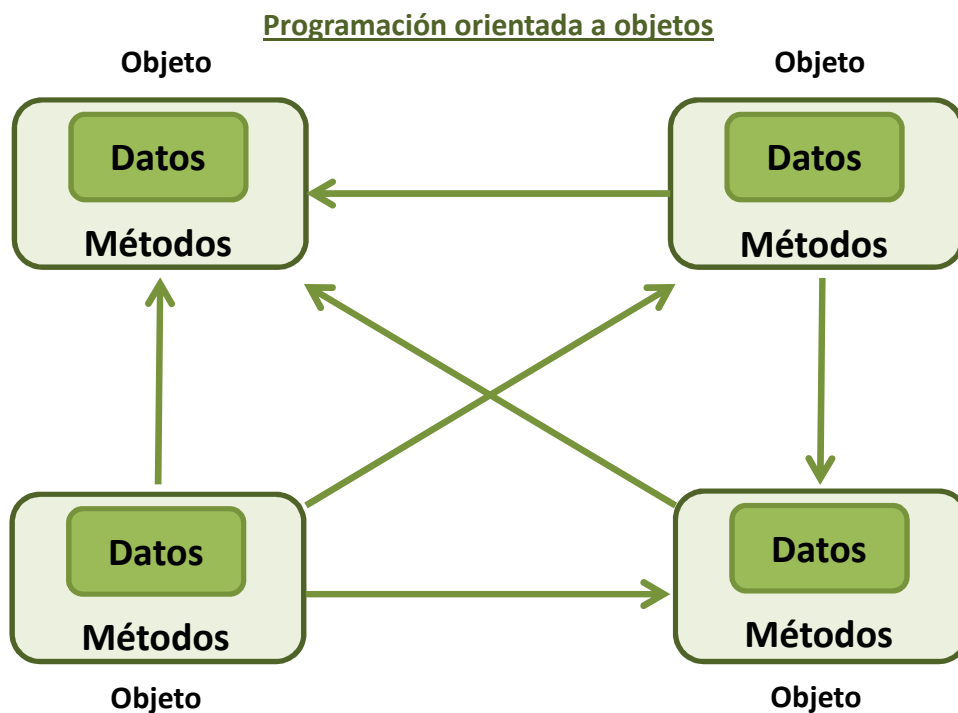


- Deshabilitar todas las macros sin notificación: permite ejecutar únicamente las macros que están almacenadas en un lugar confiable, que, por defecto, son ciertos directorios dentro de la carpeta que contiene los archivos de programa de MS Office en el ordenador del usuario.
- Deshabilitar todas las macros con notificación: muestra una alerta de seguridad advirtiendo de la intención de ejecutar una macro, de manera que el usuario decida si la ejecuta o no. Es la opción predeterminada.
- Deshabilitar todas las macros excepto las firmadas digitalmente: permite ejecutar únicamente las macros con una firma digital y, por tanto, autor conocido.
- Habilitar todas las macros: permite ejecutar todas las macros sin enviar ninguna notificación al usuario, tanto de fuentes conocidas como desconocidas.

2. PRINCIPIOS BÁSICOS DE PROGRAMACIÓN EN VBA

2.1 PROGRAMACIÓN ORIENTADA A OBJETOS

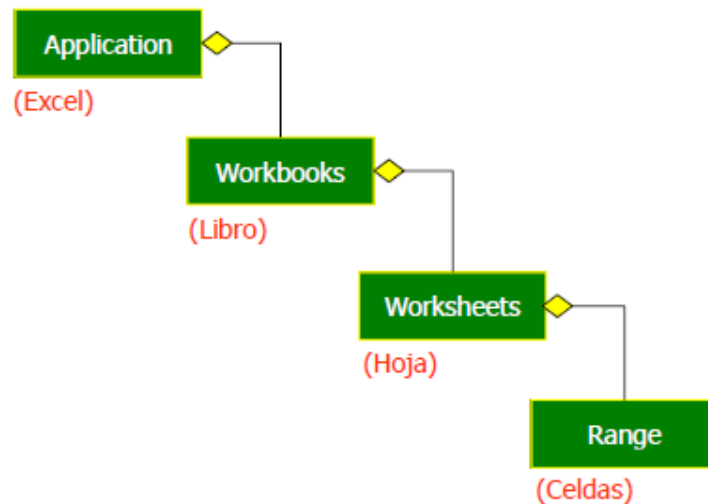
La programación en VBA es una programación orientada a objetos. Un objeto es un elemento provisto de propiedades (datos) y funciones (métodos) que se relaciona con otros objetos de una manera concreta. Todas las acciones o instrucciones que contiene el programa están incluidas en un objeto.



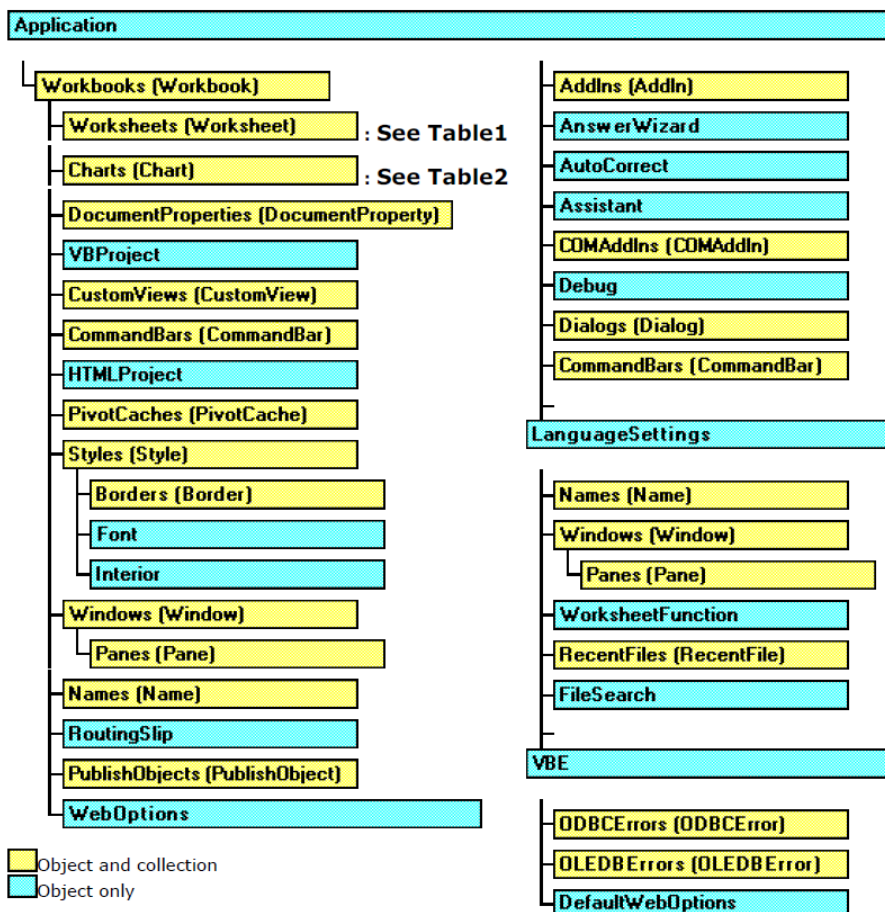
Cada elemento de Excel, por tanto, es un objeto que tiene propiedades y métodos. Por ejemplo, el objeto **Workbook** (libro de Excel) tiene propiedades como **ActiveSheet** (hoja activa), **Name** (nombre), **ReadOnly** (sólo lectura) o **Saved** (guardado); y algunos de sus métodos son **Save** (guardar), **Close** (cerrar), **PrintOut** (imprimir), **Protect** (proteger) o **Unprotect** (desproteger). Otros objetos son **Sheet** (hoja de cálculo), **Range** (celda o rango de celdas) y **Chart** (gráfico).

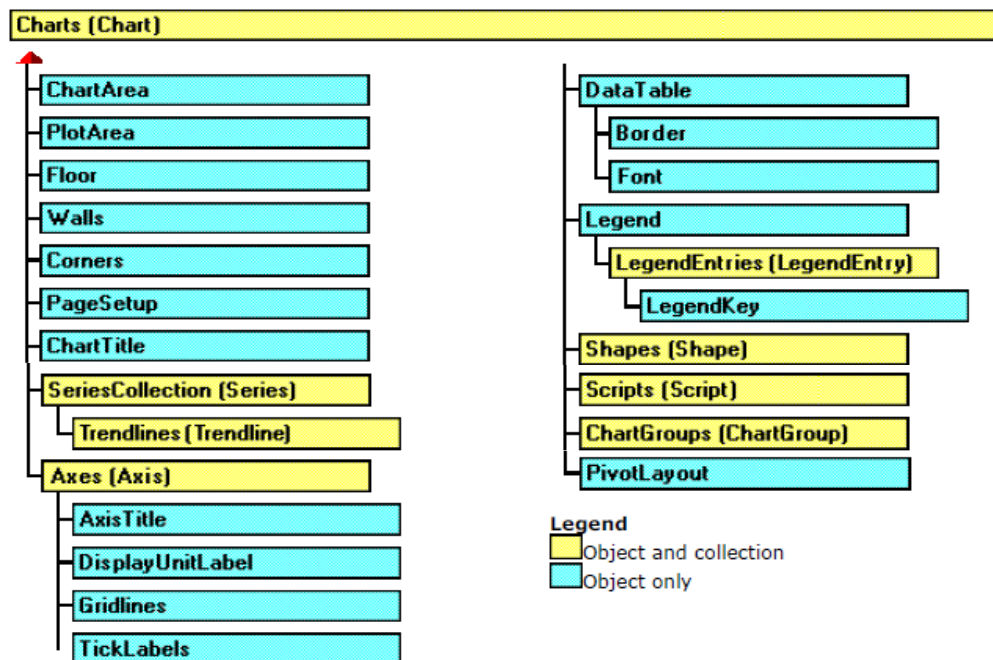
Objetos:

La estructura de **objetos** es jerárquica, unos están contenidos en otros. El objeto **Application** es el de mayor rango, contiene a todos los demás. Representa al propio programa Excel y su uso proporciona toda la información referida a la aplicación que está en uso. Contiene valores y opciones de toda la aplicación y, aunque en ocasiones puede ser necesario declarar este objeto explícitamente, lo normal es no tener que hacerlo. Si no se escribe, se entiende que se trata de Excel. A continuación se encuentran los **Workbook** o libros de Excel. Cada **Workbook** contiene una cantidad de **Worksheet** u hojas de cálculos y éstas, a su vez, contienen un conjunto de **Range**, que son celdas o grupos de celdas.



Pero estos no son los únicos objetos. Por ejemplo, cada Workbook contiene o puede contener, además, Charts (gráficos), Windows (ventanas), CommandBars (barras de herramientas), VBProject (proyectos de Visual Basic), etc. Cada Range, contiene Font, Interior, Border, Column, Row, Text, Cell, Formula, Value, etc.





Tanto **Workbook** como **Worksheet** representan un libro o una hoja de cálculo en concreto, pudiendo existir muchos de estos elementos. Para diferenciarlos entre ellos hay que darles un nombre, del tipo **Workbook**("Libro1.xlsm") o **Worksheet**("Hoja1"). Cuando los elementos son múltiples, se almacenan en colecciones que contienen todos los elementos de esa clase que están abiertos en ese momento. Así, **Workbooks** es la colección de todos los libros de Excel abiertos en ese momento y **Worksheets** es la colección de todas las hojas de cálculo de esos libros. De esta manera se puede hacer referencia a todos ellos con un único nombre en lugar de tener que nombrarlos uno a uno. Por otro lado, **Sheets** es la colección de hojas del libro activo (y no de todos los libros abiertos) y **Sheet** es una hoja del libro activo. Además, cuando se trata del elemento activo, no es necesario nombrarlo, basta con indicar que se trata del objeto activo de la forma **ActiveWorkbook**, **ActiveSheet**, **ActiveCell** o **ThisWorkbook** (el libro que contiene la macro que está siendo ejecutada).

Propiedades y métodos:

Las **propiedades** son las variables que describen los aspectos o características del objeto en el que están incluidas. Pueden ser permanentes o variables. El valor concreto de una propiedad se denomina estado. Por ejemplo, dentro del objeto celda, una de sus propiedades es color del fondo y uno de los posibles estados de esta propiedad es rojo. Esto se escribe de forma genérica como: **NombreDelObjeto.Propiedad = estado**. En el caso de la celda roja:

Selection.Interior.Color = 255 en lugar de selection, que hace referencia a la celda en la que está el cursor, se puede llamar a una celda concreta, por ejemplo **Range("A5")**. En este caso, la propiedad interior (fondo) contiene a su vez otra propiedad, **Color**, y el estado de dicha sub-propiedad es 255, que equivale al rojo brillante.

ActiveSheet.Name = "Resultados" cambia el nombre de la hoja activa por Resultados.

Los **métodos** son acciones que el objeto reconoce y sabe ejecutar. Es un procedimiento asociado a un objeto determinado y cuya ejecución sólo puede desencadenarse a través de un mensaje recibido por el objeto o uno de sus descendientes (objetos que dependen de otro objeto jerárquicamente superior). La forma de escribir un método es **NombreDelObjeto.Método**. Por ejemplo:

Range("A1").Select selecciona y convierte en la celda activa la casilla A1.

Lo mismo ocurre si se especifican las coordenadas de la celda del modo (fila, columna), en este caso **Range (1, 1)**. Si fuera un rango, por ejemplo, de las celdas A1 a C5, el rango sería **Range(Cells(1, 1), Cells(5, 3))** o **Range("A1:C5")**. Otras opciones son **Columns("A:C")**, **Rows("1:5")**, **Range("A:A")**, **Range("3:3")**.

Sheets("Hoja1").Select selecciona la hoja 1 y la convierte en la hoja activa.

Otros ejemplos:

Range("A1").Select selecciona la celda A1

Selection.Copy copia su contenido

Range("C3").Select selecciona la celda C3

ActiveSheet.Paste pega lo copiado en esta celda (no se puede usar selection otra vez porque ya tiene un valor, lo que contiene la celda A1)

Columns("D:D").Clear borra el contenido de todas las celdas de la columna D

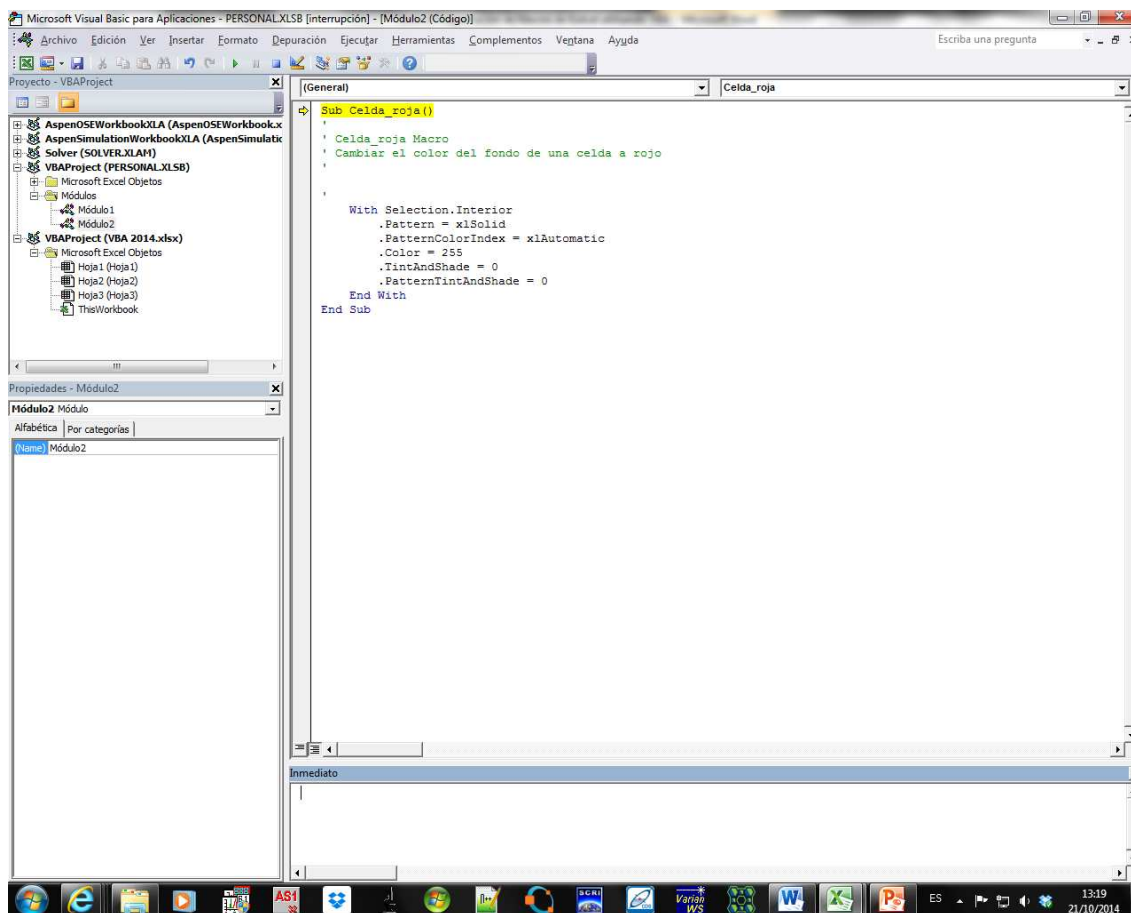
Rows("6:6").Delete elimina la fila 6 y las filas inferiores se desplazan

Un **procedimiento** es un conjunto de instrucciones que conducen a un resultado específico, mientras que un **módulo** es un conjunto de procedimientos que conducen a la resolución de un problema. Un **programa** es un conjunto de módulos que el usuario ha combinado para realizar una actividad compleja. Dicho de otra manera, un procedimiento es, por ejemplo, el conjunto de instrucciones que hacen que una casilla resuelva una función y ponga el resultado en negrita; un módulo es el conjunto de procedimientos que han servido para importar los datos, copiarlos en las celdas, contar el número de datos, leerlos, realizar la operación y ofrecer el resultado; un programa hace todo eso y, además, representa los resultados en una gráfica, genera un informe y lo guarda en una ubicación específica, etc.

Los procedimientos pueden ser de dos clases: funciones y subrutinas. Las **funciones** (function) son expresiones matemáticas que aceptan como argumentos variables, constantes y expresiones cuyo valor debe estar contenido en una celda y cuyo resultado también va a escribirse en una celda. Las **subrutinas** (sub) son instrucciones de cualquier tipo, que pueden tener como argumento también una variable, una constante o una expresión, pero cuyos valores no tienen por qué estar declarados en celdas, al igual que el resultado final.

2.2 EL EDITOR DE VISUAL BASIC

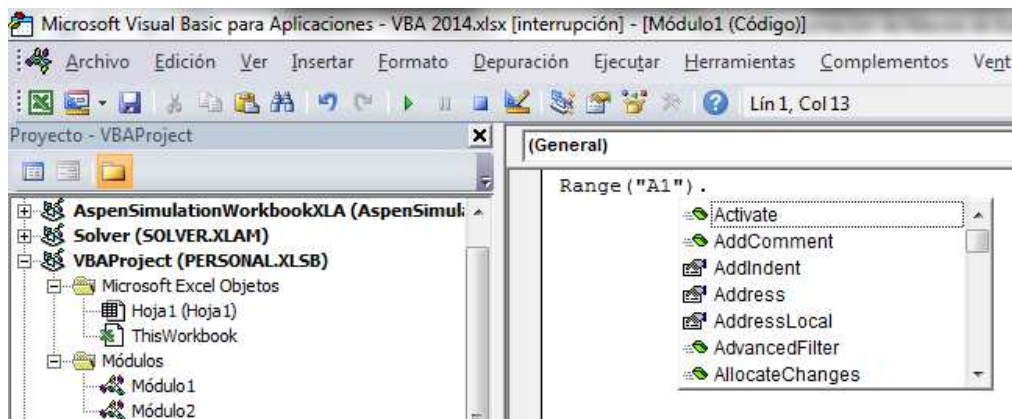
El Editor de Visual Basic es un programa independiente de Excel que permite crear y modificar macros, cambiar las propiedades del libro de trabajo y sus hojas, crear procedimientos o módulos y formularios. El editor se abre desde la Ficha Programador con el comando Visual Basic o pulsando Alt+F11. El aspecto del editor es el siguiente:



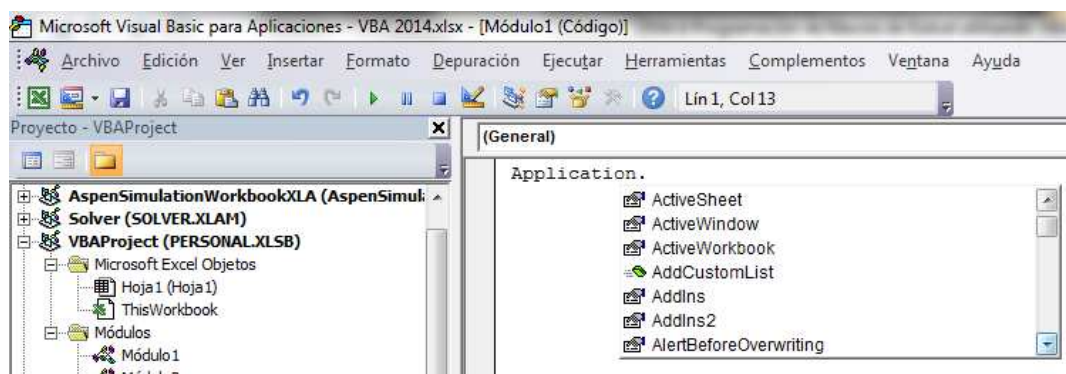
En la parte superior izquierda se muestra el Explorador de proyectos, que muestra el libro de macros y el libro de Excel donde se está trabajando actualmente, con todas sus hojas. En la parte inferior está la ventana Inmediato, donde se pueden introducir instrucciones y observar su resultado inmediatamente. Ambas secciones se pueden hacer visibles, de no aparecer automáticamente, desde el menú Ver. El área principal es donde se escriben y editan las instrucciones que conforman la macro.

Para crear una nueva macro hay que comenzar insertando un módulo nuevo. Un módulo es un conjunto de sentencias que declaran variables y un conjunto de procedimientos a ejecutar para dichas variables. Una macro está formada por varios módulos. Los módulos se insertan desde el menú Insertar → módulo.

Los objetos tienen muchas propiedades y métodos y es difícil imaginar que se pueden memorizar todos por completo. Sin embargo, el Editor de VBA proporciona, mientras se escribe el código (justo después de poner el punto tras el nombre del objeto), la lista completa de propiedades y métodos para el objeto dado. Por ejemplo, al escribir el nombre de una celda, se abre un cuadro de menú con todas las propiedades (una mano señalando un archivo) y métodos (un icono verde) que pueden aplicarse a ese objeto. Por ejemplo, para una celda:



Excel trabaja con un modelo de objetos en el que cada objeto pertenece a un nivel jerárquico. En la parte superior de la jerarquía se encuentra el objeto Application, esto es, el propio programa Excel. Por debajo están ActiveWorkbook > ActiveSheet > Range. Ésta es sólo una de las ramas de la jerarquía. Para conocerlas todas basta con escribir Application. y elegir entre las opciones del cuadro de menú:



Y así con cada objeto. De esta manera, para que una celda quede marcada en negrita basta con escribir **Selection.Font.Bold = True**, aunque podemos especificar toda la jerarquía de esa celda de la forma **Application.ActiveWorkbook.ActiveSheet.Selection.Font.Bold = True**. Cuando no es necesario declarar un objeto para que el código funcione, como en este caso Application, ActiveWorkbook y ActiveSheet, es porque se trata de objetos predeterminados, es decir, Excel entiende por defecto que la jerarquía de los objetos es la anterior, esto es, que la celda seleccionada está sin duda en la hoja activa, la hoja activa en el libro activo y el libro activo pertenece a Excel.

Aun cuando el usuario puede necesitar declarar la ruta completa al objeto, no es necesario hacerlo para cada instrucción. Basta con utilizar el comando With al definir la jerarquía de objetos:

Sub Probar_With() crea una macro con una única subrutina que se llama Probar_With

With Application.ActiveWorkbook.ActiveSheet.Range("A1") la celda A1, dentro de la hoja activa, dentro del libro activo y dentro de la aplicación

- . Value = "Aprender VBA" va a contener este texto
- . Font.Bold = True el texto va a estar en negrita
- . Font.ColorIndex = 2 las letras van a ser de color blanco
- . Interior.ColorIndex = 1 y el interior de la celda de color negro

End With fin de las instrucciones para el objeto indicado

End Sub fin de la subrutina

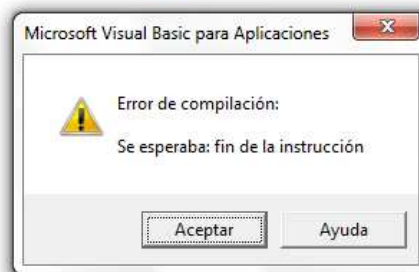
2.3 TIPOS DE ERRORES EN VBA

Existen dos tipos de errores en VBA: errores de sintaxis y errores de tiempo de ejecución.

Errores de sintaxis:

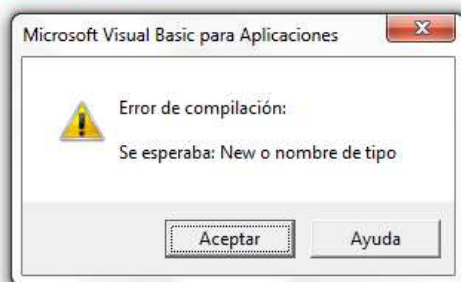
Un error de sintaxis es el que se produce, bien porque se escribe mal una instrucción, bien porque se escribe una instrucción correcta en un lugar inadecuado. En el primer caso, el Editor avisa de un error al escribir Mud el lugar de Mod, que es el comando que corresponde al módulo:

```
Sub CalcularValor()  
    CalcularValor = 100 Mud 5  
End Sub
```



En el segundo caso, el Editor devuelve un error indicando que después del comando ValorBase as debe definirse dicho valor o decir que es una variable nueva, no pudiendo utilizarse ningún otro comando:

```
Sub CalcularValor()  
    Dim ValorBase as Next  
End Sub
```



Errores en tiempos de ejecución:

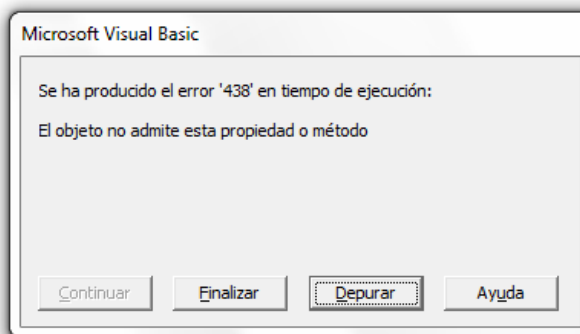
Estos errores son más complejos y ocurren cuando la macro intenta ejecutar una instrucción que no está permitida, de manera que Excel dejará de responder. Algunos de estos errores ocurren por:

- Intentar realizar una operación prohibida en Excel, como dividir entre cero o sumar una cadena de texto.
- Intentar utilizar una librería de código que no está accesible en ese momento.
- Utilizar un bucle con una condición que nunca se cumple.
- Tratar de asignar un valor que está fuera de los límites de la variable.

Para evitar errores, el Editor permite depurar el código. Una manera sencilla es utilizar la opción Depuración → Paso a paso por instrucciones o simplemente pulsar F8. Esto hace que se inicie la ejecución de la primera línea. Si la línea está correcta, el fondo aparece en amarillo y no se muestra ningún mensaje de error. Para pasar a la siguiente línea hay que pulsar de nuevo F8, y así sucesivamente. Por ejemplo, al escribir valor en lugar de Value:

```
Sub CalcularValor()  
  Range("B1").Value = "abcisas"  
  Range("C12").valor = "ordenadas"
```

End Sub



Al pulsar depurar, el Editor nos devuelve a la línea que contiene el error para que lo corrijamos:

```
Sub CalcularValor()  
  Range("B1").Value = "abcisas"  
  Range("C12").valor = "ordenadas"  
End Sub
```

3. PROGRAMACIÓN EN VBA

3.1 VARIABLES

Las variables almacenan valores (datos, constantes, nombres, resultados, etc.). Existen muchos tipos de variables, pero los más habituales son los siguientes:

Tipo de variable	Descripción	Valores
Integer	Número entero	de -32.768 a 32.767
Long	Número entero largo	de -2.147,483.648 a 2.147,483.647
Single	Número real con hasta 6 posiciones decimales	
Decimal	Número real con hasta 28 posiciones decimales	
Date	Fecha	de #1/1/100# a #31/12/9999#
String	Datos de texto	de 0 a $2 \cdot 10^6$ caracteres
Object	Cualquier referencia a un objeto	
Boolean	Lógico del tipo cumple/no cumple	True o False
Variant	El tipo por defecto si no se especifica otro. Admite cualquiera de los formatos anteriores ² .	

Hay cuatro clases de variables:

- Variables locales: son aquellas que se declaran dentro de un módulo y no pueden utilizarse fuera de él. Para declararlas se utiliza el comando **Dim ... As** al principio del módulo correspondiente de la forma: **Dim NombreDeLaVariable As TipoDeVariable**. Si se van a declarar varias variables del mismo tipo, se puede hacer separando su nombre con comas: **Dim Nombre1, Nombre2,... As TipoDeVariable**.
- Variables públicas: están disponibles en todos los módulos de un proyecto. Se definen con el comando **Public NombreDeLaVariable As TipoDeVariable**.
- Variables estáticas: conservan su valor en todos los procedimientos y se declaran como **Static NombreDeLaVariable As TipoDeVariable**.
- Constantes: son valores cuyo valor no cambia al ejecutar un procedimiento. Se declaran como **Const NombreDeLaVariable As TipoDeVariable**. Las constantes también pueden ser públicas de la forma **Public Const...**

Una vez que una variable se ha declarado, se le puede asignar un valor de la forma **NombreDeLaVariable = Valor**.

² El objetivo de especificar el tipo de dato es ahorrar espacio en la memoria, que el programa ocupe menos. Por ejemplo, una variable de tipo Variant ocupa siempre 16 bytes. Si la variable va a ser un número entero, al usar Variant ocupará esos 16 bytes, pero si se define como Integer ocupará sólo 2 bytes.

El comando **Option Explicit** escrito al comienzo de un módulo obliga al usuario a definir todas las variables, de manera que, si el programa encuentra una variable no definida se detiene. Esto permite identificar fácilmente errores en la declaración de variables o usar el mismo nombre para dos variables.

3.2 ENTRADA DE DATOS Y EMISIÓN DE RESULTADOS

Una de las funciones básicas de un programa es pedirle al usuario que defina el valor de las variables, es decir, que rellene los datos con los que se va a resolver el problema. Para ello, una opción es crear un cuadro de diálogo que pida al usuario cierta información y que contenga una casilla que el usuario pueda rellenar, además de botones que le permitan continuar el programa, como aceptar, cancelar, siguiente, sí, no, etc.

Para que el programa le pida información al usuario hay que utilizar el comando `InputBox` de la siguiente forma: **InputBox(mensaje, título, valor por defecto, posición en x, posición en y)**. Donde:

- Mensaje: mensaje que va a recibir el usuario.
- Título: título del cuadro de diálogo que se va a abrir.
- Valor por defecto: valor que se asigna por defecto si el usuario no introduce otra cosa.
- Posición en x: posición horizontal del cuadro de diálogo en la pantalla, en píxeles.
- Posición en y: posición vertical del cuadro de diálogo en la pantalla, en píxeles.

No es necesario incluir todos los datos que se piden, basta, por ejemplo, con escribir el mensaje, pero hay que mantener el orden en el que se escriben.

Ejemplo 2: Crear un cuadro de diálogo para introducir datos.

- a) Crear un cuadro llame Variable X que pida al usuario introducir un dato. En el caso de que no se introduzca nada, que tome el valor 120. Que el cuadro esté en la posición (4830, 2210) de la pantalla:

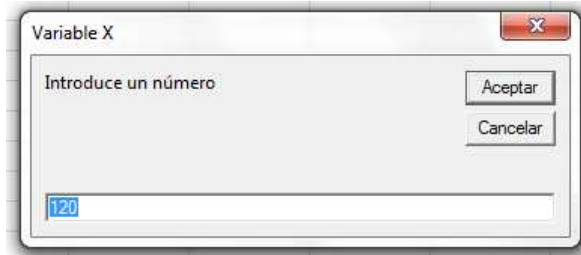
La subrutina es la siguiente:

```
Sub IntroducirDato()
```

```
Numero = InputBox("Introduce un número", "Variable X", 120, 4830, 2210) es necesario darle un nombre a la variable inputbox, en este caso Numero
```

```
End Sub
```

Y el resultado:



El programa no se sigue ejecutando hasta que el usuario pulsa Aceptar o Cancelar, que aparecen por defecto. El valor que se ha elegido por defecto, 120, aparece directamente en el recuadro a la espera de que el usuario escriba otro valor.

Para que le programa muestre el resultado de las operaciones que ha realizado con los datos introducidos hay que crear un nuevo cuadro de diálogo con el comando MsgBox de la forma: **MsgBox(mensaje, botones, título)**. Donde:

- Mensaje: es el mensaje que va a recibir el usuario junto con los resultados. Puede incluir caracteres especiales, del tipo +Chr(13) (retorno de carro) o +Chr(10) (avance de línea).
- Botones: se pueden incluir diferentes botones:

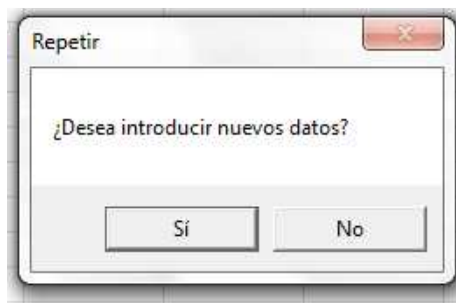
Botones	Comando abreviado	Descripción
VbOKOnly	0	Muestra solamente el botón Aceptar.
VbOKCancel	1	Muestra los botones Aceptar y Cancelar
VbAbortRetryIgnore	2	Muestra los botones Anular, Reintentar e Ignorar
VbYesNoCancel	3	Muestra los botones Sí, No y Cancelar.
VbYesNo	4	Muestra los botones Sí y No.
VbRetryCancel	5	Muestra los botones Reintentar y Cancelar.
VbCritical	16	Muestra el icono de mensaje crítico.
VbQuestion	32	Muestra el icono de pregunta de advertencia.
VbExclamation	48	Muestra el icono de mensaje de advertencia.
VbInformation	64	Muestra el icono de mensaje de información

- Título: rótulo de la ventana que se va a generar.

Ejemplo 3: Crear cuadros de resultados con botones:

- Crear un cuadro llamado Repetir que pregunte al usuario si desea introducir nuevos datos y ofrezca dos botones, Sí y No.

```
Sub SiNo()
    Respuesta = MsgBox("¿Desea introducir nuevos datos?", vbYesNo, "Repetir")
End Sub
```

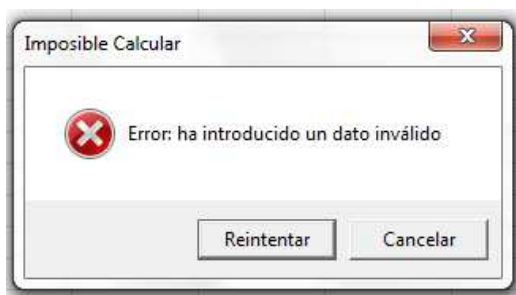


- b) Crear un cuadro llamado Imposible Calcular que avise al usuario de que ha introducido un valor erróneo junto con un icono de mensaje crítico y los botones reintentar y cancelar:

```
Sub Fallo()
```

```
Aviso = MsgBox("Error: ha introducido un dato inválido", 5 + 16, "Imposible Calcular")
```

```
End Sub
```

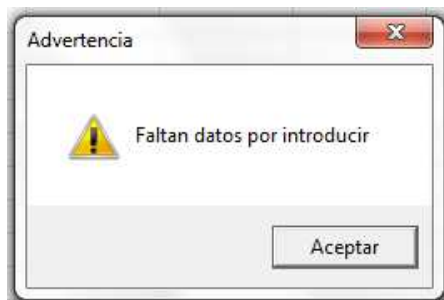


- c) Crear un cuadro llamado Advertencia que indique al usuario que faltan datos por introducir y que muestre un icono de advertencia:

```
Sub Advertencia()
```

```
Advertencia = MsgBox("Faltan datos por introducir", vbExclamation, "Advertencia")
```

```
End Sub
```



- d) Diseñar un programa que calcule el precio total de una compra a partir del número de unidades compradas, el precio unitario y el IVA, creando los correspondientes cuadros para introducir las variables y el cuadro con la respuesta final.

Sub Precio_Compra()

Dim Cantidad As Integer

Dim Precio, IVA, Total As Single

Precio = InputBox("Introduzca el precio unitario en €", "Precio")

Cantidad = InputBox("Introduzca la cantidad que desea comprar", "Cantidad", 1)

IVA = InputBox("Modifique el tipo impositivo si es necesario", "IVA", 0.21)

Total = Cantidad * Precio + Cantidad * Precio * IVA

Resultado = MsgBox("El precio total es:" + Chr(13) + Chr(10) + Chr(13) + Chr(10) & Total, 0, "Precio final") Después del mensaje, tiene que bajar dos líneas y recuperar la tabulación inicial hasta poner el resultado (esto es, dejar una línea vacía entre medias). 0 es que aparezca el botón de aceptar únicamente

End Sub

3.3 FUNCIONES

La expresión general de una función es la siguiente:

Function Nombre(argumentos separados por comas)

```
Instrucciones
```

```
End Function
```

Las funciones se almacenan como elementos independientes y pasan a formar parte del conjunto de funciones predefinidas. De hecho, una vez definida y almacenada la función, basta con escribir su nombre en una celda y ejecutarla, como se haría con cualquiera de las funciones predefinidas. También pueden incluirse en subrutinas.

Ejemplo 2: Crear una función

- a) Crear una función que calcule las raíces de una ecuación de segundo grado:

Es necesario crear dos funciones, ya que hay dos soluciones posibles:

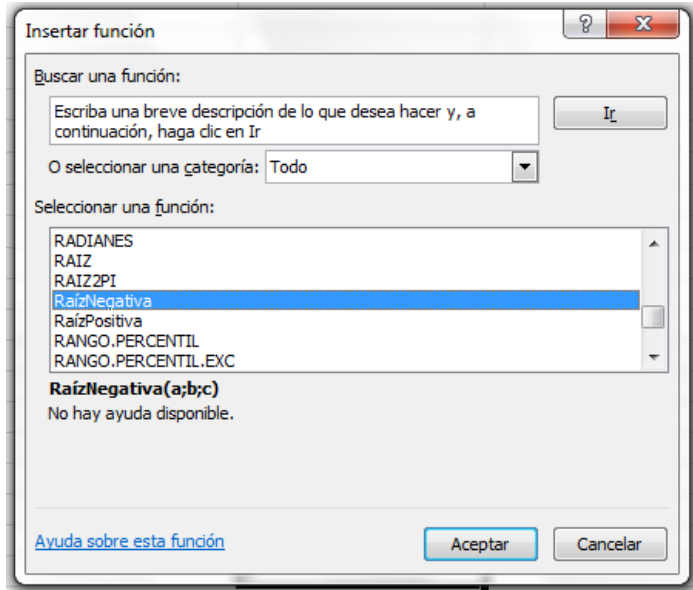
$$x1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad x2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Las funciones serían las siguientes:

```
Function RaízPositiva(a, b, c) se indica el nombre y que va a tener tres argumentos
If (b ^ 2 - 4 * a * c < 0) Then si se cumple que el interior de la raíz sea negativo
Respuesta = MsgBox("Las raíces son imaginarias", 5, "Error") el programa debe dar un
mensaje de error indicando que la solución es un número imaginario
Else de lo contrario
RaízPositiva = (-b + Sqr(b ^ 2 - 4 * a * c)) / (2 * a) debe calcular la raíz de esta manera
End If termina el bucle If-Then
End Function termina la función
```

```
Function RaízNegativa(a, b, c)
If (b ^ 2 - 4 * a * c < 0) Then
Respuesta = MsgBox("Las raíces son imaginarias", 5, "Error")
Else
RaízNegativa = (-b - Sqr(b ^ 2 - 4 * a * c)) / (2 * a)
End If
End Function
```

Una vez definidas, las funciones han quedado integradas dentro de las funciones predeterminadas:



De esta manera, el programa da la opción de elegir las, como a cualquier otra función:

a	b	c	Raíz Positiva	Raíz Negativa
1	-5	6	=Raíz	
2	-7	3		

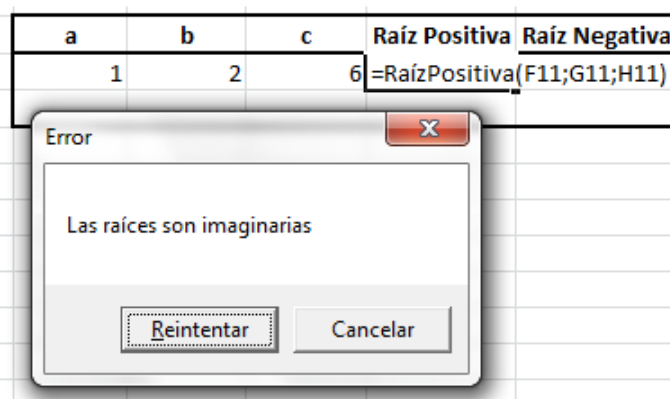
Para ejecutarlas, basta con escribir su nombre de la forma que se ha definido:

a	b	c	Raíz Positiva	Raíz Negativa
1	-5	6	=RaízPositiva(F11;G11;H11)	=RaízNegativa(F11;G11;H11)
2	-7	3	=RaízPositiva(F12;G12;H12)	=RaízNegativa(F12;G12;H12)

Y obtener el resultado:

a	b	c	Raíz Positiva	Raíz Negativa
1	-5	6	3	2
2	-7	3	3	0,5

Cuando las raíces son imaginarias, aparece el mensaje de error:



El lenguaje VBA contiene una serie de funciones predeterminadas de carácter no matemático que pueden resultar útiles y que son las siguientes:

FUNCIÓN	DESCRIPCIÓN
Abs	Regresa el valor absoluto de un número
Asc	Obtiene el valor ASCII del primer carácter de una cadena de texto
CBool	Convierte una expresión a su valor booleano
CByte	Convierte una expresión al tipo de dato Byte
CCur	Convierte una expresión al tipo de dato moneda (Currency)
CDate	Convierte una expresión al tipo de dato fecha (Date)
CDbl	Convierte una expresión al tipo de dato doble (Double)
CDec	Convierte una expresión al tipo de dato decimal (Decimal)
Choose	Selecciona un valor de una lista de argumentos
Chr	Convierte un valor ANSI en valor de tipo texto
CInt	Convierte una expresión en un dato de tipo entero (Int)
CLng	Convierte una expresión en un dato de tipo largo (Long)
CreateObject	Crea un objeto de tipo OLE
CStr	Convierte una expresión en un dato de tipo texto (String)
CurDir	Regresa la ruta actual
CVar	Convierte una expresión en un dato de tipo variante (Var)
Date	Regresa la fecha actual del sistema
DateAdd	Agrega un intervalo de tiempo a una fecha especificada
DateDiff	Obtiene la diferencia entre una fecha y un intervalo de tiempo especificado
DatePart	Regresa una parte específica de una fecha
DateSerial	Convierte una fecha en un número serial
DateValue	Convierte una cadena de texto en una fecha
Day	Regresa el día del mes de una fecha
Dir	Regresa el nombre de un archivo o directorio que concuerde con un patrón
EOF	Regresa verdadero si se ha llegado al final de un archivo
FileDateTime	Regresa la fecha y hora de la última modificación de un archivo
FileLen	Regresa el número de bytes en un archivo
FormatCurrency	Regresa un número como un texto con formato de moneda
FormatPercent	Regresa un número como un texto con formato de porcentaje
Hour	Regresa la hora de un valor de tiempo
IIf	Regresa un de dos partes, dependiendo de la evaluación de una expresión
InputBox	Muestra un cuadro de diálogo que solicita la entrada del usuario
InStr	Regresa la posición de una cadena de texto dentro de otra cadena
InStrRev	Regresa la posición de una cadena de texto dentro de otra cadena pero empezando desde el final
Int	Regresa la parte entera de un número
IsDate	Regresa verdadero si la variable es una fecha
IsEmpty	Regresa verdadero si la variable está vacía
IsError	Regresa verdadero si la expresión es un valor de error
IsNull	Regresa verdadero si la expresión es un valor nulo
IsNumeric	Regresa verdadero si la variable es un valor numérico
Join	Regresa una cadena de texto creada al unir las cadenas contenidas en un vector o matriz

LCase	Regresa una cadena convertida en minúsculas
Left	Regresa un número específico de caracteres a la izquierda de una cadena
Len	Regresa la longitud de una cadena (en caracteres)
LTrim	Remueve los espacios a la izquierda de una cadena
Mid	Extrae un número específico de caracteres de una cadena de texto
Minute	Regresa el minuto de una dato de tiempo
Month	Regresa el mes de una fecha
MsgBox	Despliega un cuadro de dialogo con un mensaje especificado
Now	Regresa la fecha y hora actual del sistema
Replace	Reemplaza una cadena de texto con otra
Space	regresa una cadena de texto con el número de espacios especificados
Split	Regresa un arreglo formado por cadenas de texto que formaban una sola cadena
Str	Regresa la representación en texto de un número
Right	Regresa un número especificado de caracteres a la derecha de una cadena de texto
Rnd	Regresa un número aleatorio entre 0 y 1
Round	Redondea un número a una cantidad específica de decimales
RTrim	Remueve los espacios en blanco a la derecha de una cadena de texto
Second	Regresa los segundos de un dato de tiempo
StrComp	Compara dos cadenas de texto
StrReverse	Invierte el orden de los caracteres de una cadena
Time	Regresa el tiempo actual del sistema
Timer	Regresa el número de segundos desde la media noche
TimeValue	Convierte una cadena de texto a un número de serie de tiempo
Trim	Remueve los espacios en blanco al inicio y final de una cadena de texto
TypeName	Obtiene el nombre del tipo de dato de una variable
Ucase	Convierte una cadena de texto en mayúsculas
Val	Regresa el número contenido en una cadena de texto
Weekday	Regresa un número que representa un día de la semana
WeekdayName	Regresa el nombre de un día de la semana
Year	Obtiene el año de una fecha

3.4 ESTRUCTURAS DE CONTROL DE FLUJO

Las estructuras de control de flujo permiten establecer condiciones para que una parte del código no se ejecute automáticamente, sino que haya que cumplir alguna clase de condición previamente.

Estructuras If-Then:

La declaración If-Then permite validar una condición antes de seguir ejecutando el código. Se usa en ocasiones en las que, sólo si se cumple la condición, hay que ejecutar una acción determinada. La estructura es la siguiente:

If condición Then

```

Instrucción a ejecutar si se cumple la condición
Opción a) Else
        Instrucción a ejecutar si no se cumple
Opción b) Elseif3 siguiente condición
        Instrucción a ejecutar si no se cumple
End If

```

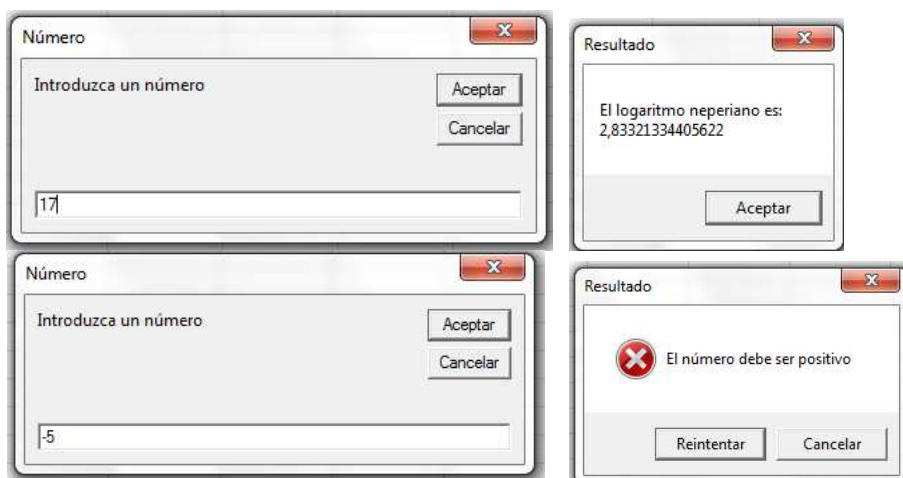
Ejemplo 4: Estructuras If-Then

- a) Crear un programa que calcule el logaritmo neperiano de un número dado por el usuario si éste es positivo. Si es negativo, que cree un cuadro de respuesta con un icono de alerta que advierta al usuario del error y le pida reintentarlo o cancelar.

```

Sub SiEntonces()
Número = InputBox("Introduzca un número", "Número")
If Número > 0 Then
Valor = Log(Número)
Respuesta = MsgBox("El logaritmo neperiano es:" + Chr(13) + Chr(10) & Valor, 0, "Resultado")
Else
Respuesta = MsgBox("El número debe ser positivo", 5 + 16, "Resultado")
End If
End Sub

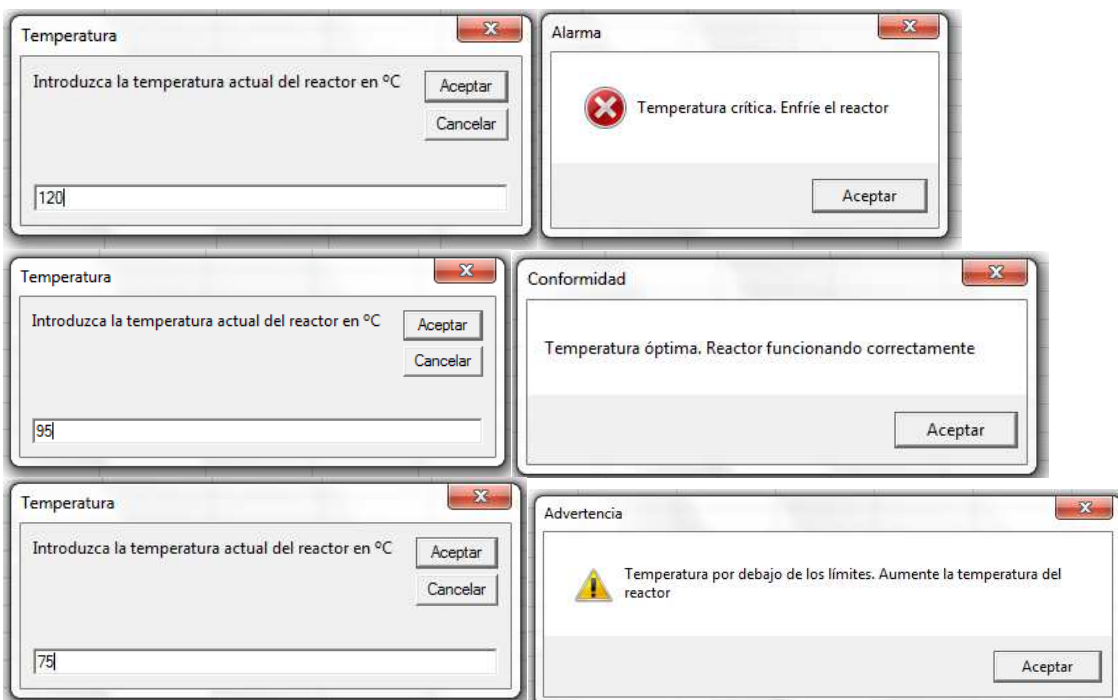
```



³ Si sólo hay dos opciones, se cumple o no se cumple, basta con Else, pero si hay varias condiciones habrá que usar Elseif, es decir, si no se cumple la primera condición pero a su vez hay que valorar si se cumple la segunda.

- b) Un reactor funciona correctamente entre 90 °C y 100°C; por encima de esa temperatura el sistema se sobrecalienta y hay que enfriarlo para que la reacción no se descontrola; entre 50 y 90 °C el reactor funciona, pero el rendimiento de la reacción es muy bajo; por debajo de 50 °C la reacción no se produce y el reactor puede atascarse, por lo que es necesario pararlo, revisarlo y volver a ponerlo en marcha. Diseñe un programa que pida al usuario la temperatura actual y le devuelva el mensaje correspondiente junto con los iconos que se consideren oportunos:

```
Sub Reactor()  
Temp = InputBox("Introduzca la temperatura actual del reactor en °C", "Temperatura")  
If Temp > 100 Then  
Mensaje = MsgBox("Temperatura crítica. Enfríe el reactor", vbCritical, "Alarma")  
ElseIf Temp > 90 Then  
Mensaje = MsgBox("Temperatura óptima. Reactor funcionando correctamente", 0, "Conformidad")  
ElseIf Temp > 50 Then  
Mensaje = MsgBox("Temperatura por debajo de los límites. Aumente la temperatura del reactor", vbExclamation, "Advertencia")  
ElseIf Temp < 50 Then  
Mensaje = MsgBox("Temperatura crítica. Desconecte y revise el reactor", vbCritical, "Alarma")  
End If  
End Sub
```





Estructuras Select-Case:

Estas estructuras permiten ejecutar una o más instrucciones según el valor que tenga una variable. No se trata sólo de si se cumple o no una condición, sino qué valor concreto tiene esa condición. La condición puede ser un valor, un intervalo o cualquier relación entre los valores que pueda tomar la variable. La estructura general es la siguiente:

Select Case variable o expresión

Case primera opción de valores que puede adquirir la variable

Instrucciones

Case siguiente opción de valores que puede adquirir la variable

Instrucciones

Case última opción de valores que puede adquirir la variable

Instrucciones

Case Else

Instrucciones si la variable no toma ninguno de los valores establecidos anteriormente

End Select

Ejemplo 5: Estructuras Select-Case

- a) Diseñar un programa que indique al usuario el tipo de crudo de que se trata en función de su densidad en grados API en función del siguiente criterio:
- Crudos Livianos: 30-40°
 - Crudos Medianos: 22-29.9°
 - Crudos Pesados: 10-21.9°
 - Crudos Extrapesados: Menos 10°

Sub Crudos()

Dim API As Single

API = InputBox("Introduzca el valor de densidad en grados API", "Densidad")

Select Case API

Case Is < 10

Solución = MsgBox("El tipo de crudo es: Extrapesado", 0, "Tipo de crudo")

Case 10 To 21.9

Solución = MsgBox("El tipo de crudo es: Pesado", 0, "Tipo de crudo")

Case 22 To 29.9

Solución = MsgBox("El tipo de crudo es: Mediano", 0, "Tipo de crudo")

Case 30 To 40

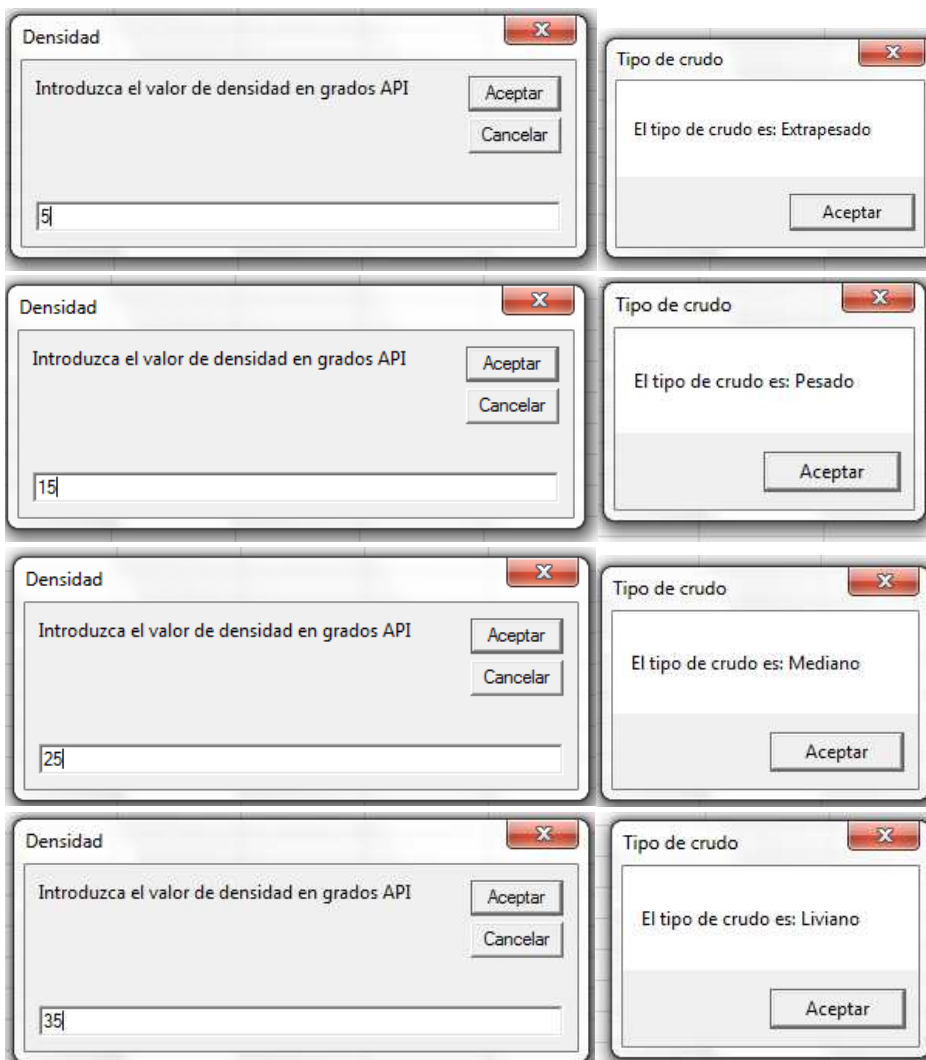
Solución = MsgBox("El tipo de crudo es: Liviano", 0, "Tipo de crudo")

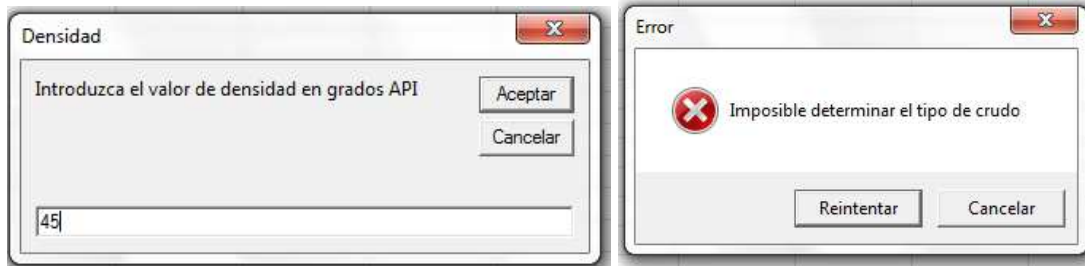
Case Else

Solución = MsgBox("Imposible determinar el tipo de crudo", 5 + 16, "Error")

End Select

End Sub





- b) Diseñar un programa que le pida al usuario el nombre de un elemento del grupo de los halógenos y le devuelva el peso atómico:

```
Sub Halógenos()
```

```
Dim Elemento As String
```

```
Elemento = InputBox("Escriba el elemento que desea evaluar:", "Elemento")
```

```
Select Case Elemento
```

```
Case "Flúor"
```

```
PA = MsgBox("18.9984 g/mol", 0, "Peso atómico")
```

```
Case "Cloro"
```

```
PA = MsgBox("35.453 g/mol", 0, "Peso atómico")
```

```
Case "Bromo"
```

```
PA = MsgBox("79.904 g/mol", 0, "Peso atómico")
```

```
Case "Yodo"
```

```
PA = MsgBox("126.9045 g/mol", 0, "Peso atómico")
```

```
Case "Astatio"
```

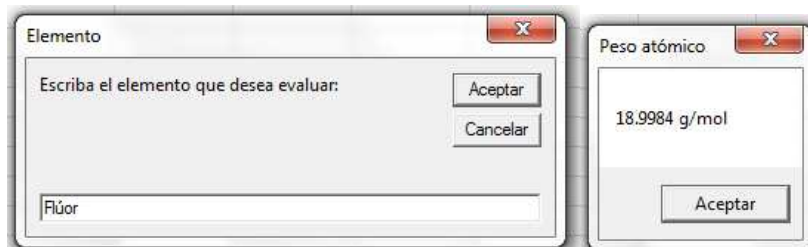
```
PA = MsgBox("210 g/mol", 0, "Peso atómico")
```

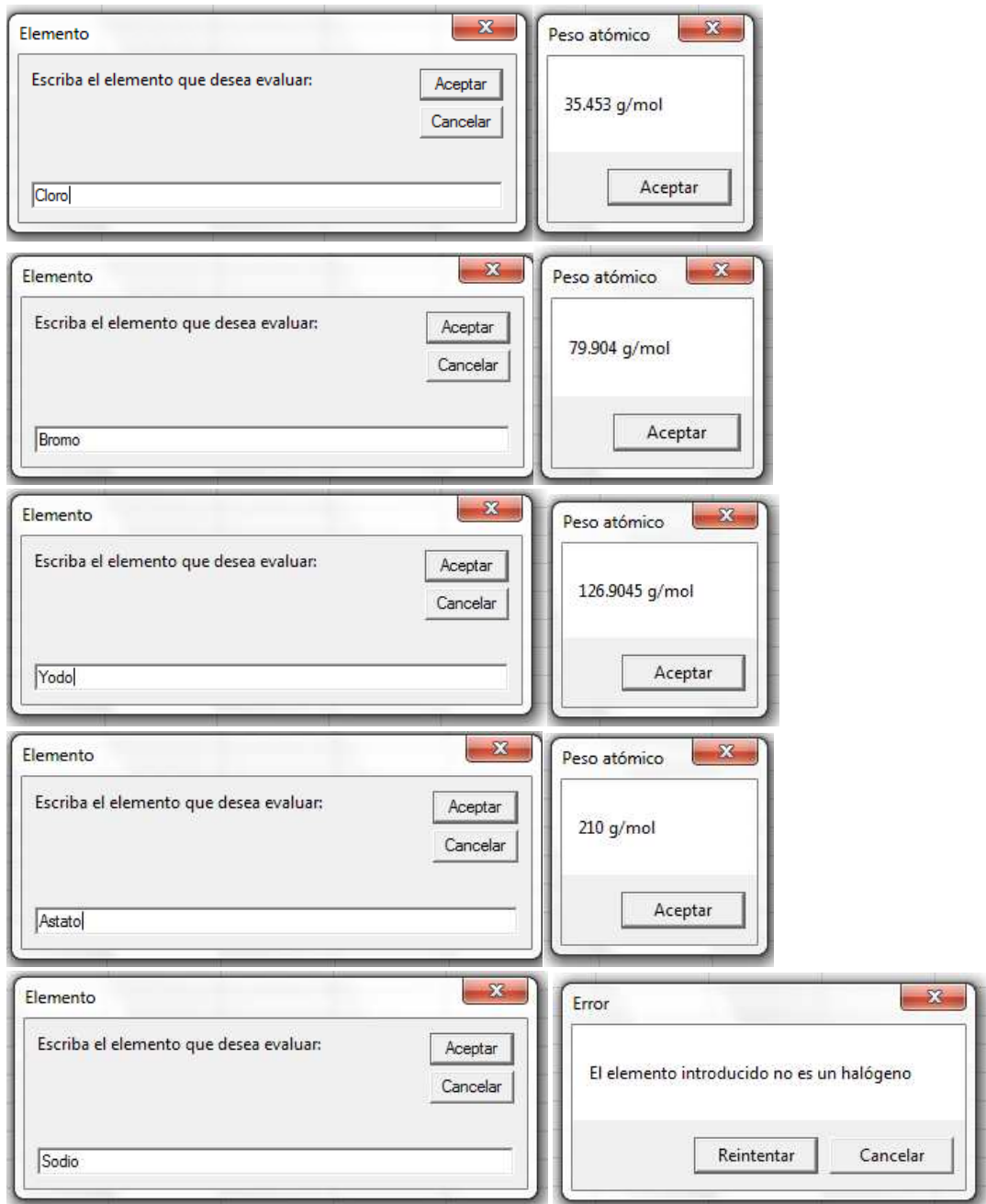
```
Case Else
```

```
PA = MsgBox("El elemento introducido no es un halógeno", vbRetryCancel, "Error")
```

```
End Select
```

```
End Sub
```





Estructuras For-Next:

Permiten ejecutar una instrucción una serie de veces mientras la variable toma los valores definidos dentro de un intervalo, es decir, es una estructura de bucle. Después, continúa con la siguiente instrucción. La estructura general es la siguiente:

```

For variable = valor inicial To valor final Step
Incremento
Instrucción
Next
    
```


Ejemplo 6: Estructuras For-Next

- a) Para un manómetro de mercurio, diseñar un programa que calcule la presión correspondiente (en Pa) a las alturas de columnas entre 1 y 10 cm cada medio cm y devuelva cada valor en una celda de la hoja activa. (Densidad del mercurio = 13560 Kg/m³):

Sub Presión()

Dim Altura, Presión As Single

For Altura = 1 To 10 Step 0.5

Presión = Altura / 100 * 9.81 * 13560

ActiveCell.Value = Presión Escribe en la celda activa el primer valor calculado

ActiveCell.Next.Select pasa a la siguiente celda (a la siguiente columna) y la selecciona como activa para escribir el siguiente valor

Next

End Sub

	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
4																			
5	1995,354	2660,4719	3325,5901	3990,708	4655,8262	5320,9438	5986,062	6651,1802	7316,2979	7981,416	8646,5342	9311,6523	9976,7695	10641,888	11307,006	11972,124	12637,242	13302,36	
6																			

Estructuras While-Wend:

Estas estructuras permiten ejecutar un conjunto de instrucciones hasta se cumpla una determinada condición. A diferencia de For-Next, While-Wend se utiliza cuando no se conoce el número de iteraciones que se va a realizar. La estructura general es:

While condición
Instrucciones
Wend

Ejemplo 7: Estructuras While-Wend

- a) Crear un programa que sume los valores contenidos en un rango de celdas y los muestre en la fila siguiente, así como en un cuadro de mensaje.

Sub WhileWend()

Worksheets("Hoja5").Range("A1").Select selecciona la celda en la que empiezan los valores⁴

Sum = Range("A1") el primer valor de la suma es el de la primera celda

While ActiveCell.Value <> "" a partir de ahí, y mientras el valor de las celdas no sea nulo

⁴ Lo primero que hay que hacer es escribir los valores que se quieren sumar en las celdas correspondientes.

(mientras estén escritas y contengan un número)⁵

ActiveCell.Offset(1, 0).Select⁶ la instrucción es bajar celda por celda para comprobar si son nulas o no (si se cumple la condición)

Sum = Sum + ActiveCell.Value si se cumple, la instrucción es añadir el valor de la celda activa al valor acumulado de la suma

Wend cuando deje de cumplirse la condición, terminar el bucle

With ActiveCell para la última celda activa (la primera que se ha encontrado vacía)

.Value = Sum Darle el valor de la suma

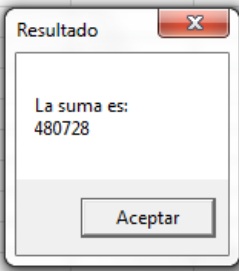
.Font.Bold = True poner el valor en negrita para destacarlo

End With terminar las acciones con esa celda

Resultado = MsgBox("La suma es:" + Chr(13) + Chr(10) & Sum, 0, "Resultado") crea un cuadro de mensaje con el resultado

End Sub

	A	B	C	D
1	452			
2	2365			
3	8956			
4	11773			
5	23546			
6	235			
7	897			
8	235			
9	415			
10	6569			
11	32			
12	414			
13	141613			
14	283226			
15	480728			
16				



Estructuras Do While-Loop:

Es una variante del ciclo While-Wend. La diferencia con éste es que la comprobación de que se cumple o no la condición se realiza al final del bucle, de manera que la instrucción se ejecuta al menos una vez. La estructura general es:

```

Do While condición
Instrucciones
Loop

```

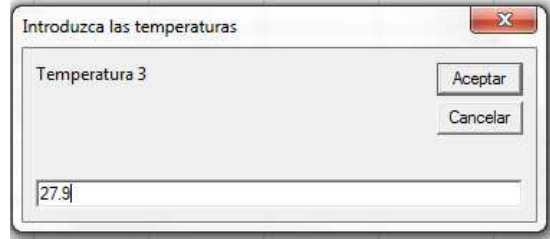
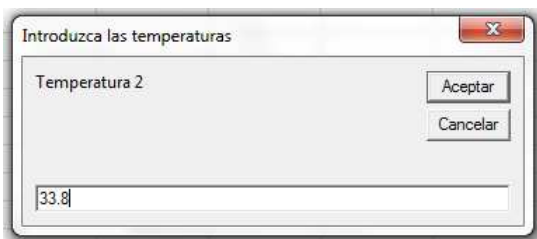
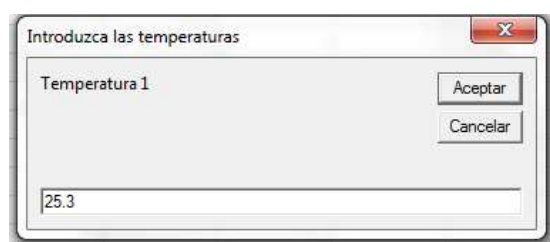
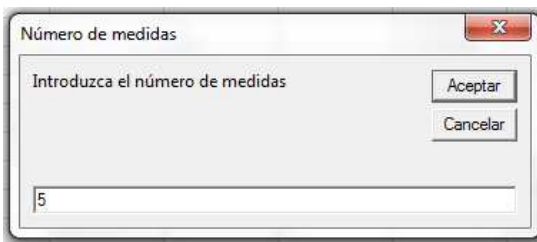
⁵ <> es el símbolo en VBA para distinto de y, al abrir y cerrar comillas sin escribir nada dentro, el lenguaje entiende que el contenido es vacío.

⁶ Offset es un método que indica que hay que desplazarse de una celda a otra. En este caso, hay que bajar una fila y moverse cero columnas a la derecha.

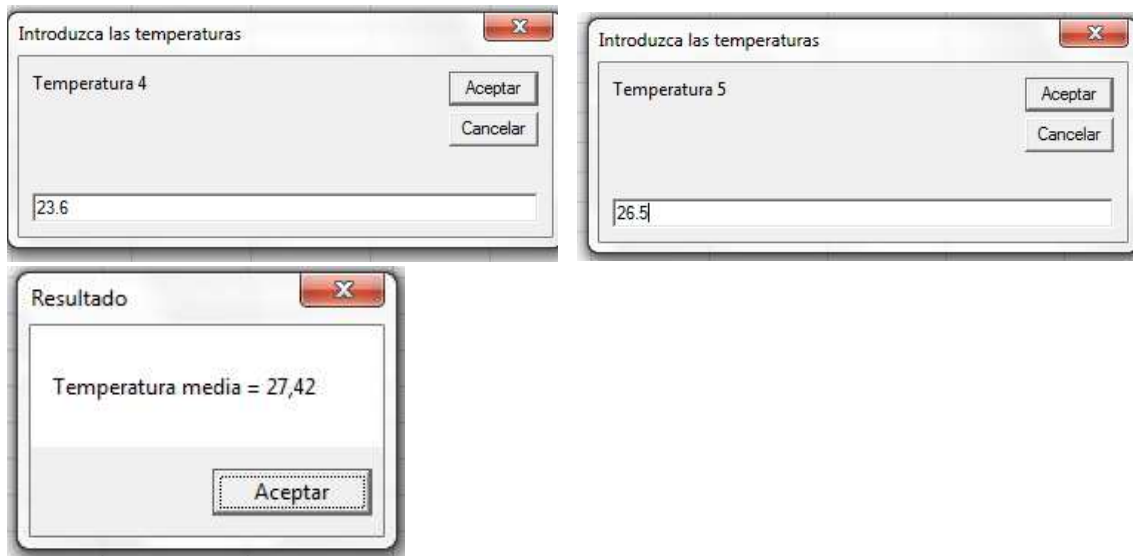
Ejemplo 8: Estructuras Do While-Loop

- a) Diseñar un programa que calcule la temperatura media. Para ello debe pedir al usuario que declare el número de valores que va a introducir y cuáles son esos valores mediante sucesivos cuadros de entrada de datos. El resultado final debe aparece en un cuadro de mensaje.

```
Sub DoWhileLoop()  
Dim Temp, Suma, Media As Single  
Dim i, n As Integer  
Suma = 0  
n = InputBox("Introduzca el número de medidas", "Número de medidas")  
i = 1  
Do While i <= n  
Temp = InputBox("Temperatura " & i, "Introduzca las temperaturas")7  
Suma = Suma + Temp  
i = i + 1  
Loop  
Media = (Suma / n)  
Resultado = MsgBox("Temperatura media = " & Media, 0, "Resultado")  
End Sub
```



⁷ ¡Ojo! No entiende los puntos como separadores decimales, ni siquiera al escribirlos con el teclado numérico, de manera que, por ejemplo 23.5 lo toma como 235.



Estructuras Do-Loop Until:

Estas estructuras crean un bucle de instrucciones que sólo se termina cuando se cumple una condición, pero dicha condición no puede establecerse a priori, sino que es el resultado de alguna de las instrucciones del bucle. La estructura general es la siguiente:

<p>Do Instrucciones Loop Until la respuesta sea la condición requerida</p>
--

Ejemplo 9: Estructuras Do-Loop Until

- a) Diseñar un programa que calcule la suma y la media de un conjunto de números, que el usuario introduce en un cuadro de datos de uno en uno, hasta que el usuario pulsa la opción de no introducir más números:

```

Sub DoLoopUntil()
Dim valor, suma, media As Single
Dim n As Integer
suma = 0
n = 0
Do
valor = InputBox("Introduzca el siguiente número", "Valores")
suma = suma + valor
n = n + 1
respuesta = MsgBox("¿Desea introducir otro número?", vbYesNo, "Siguiete")

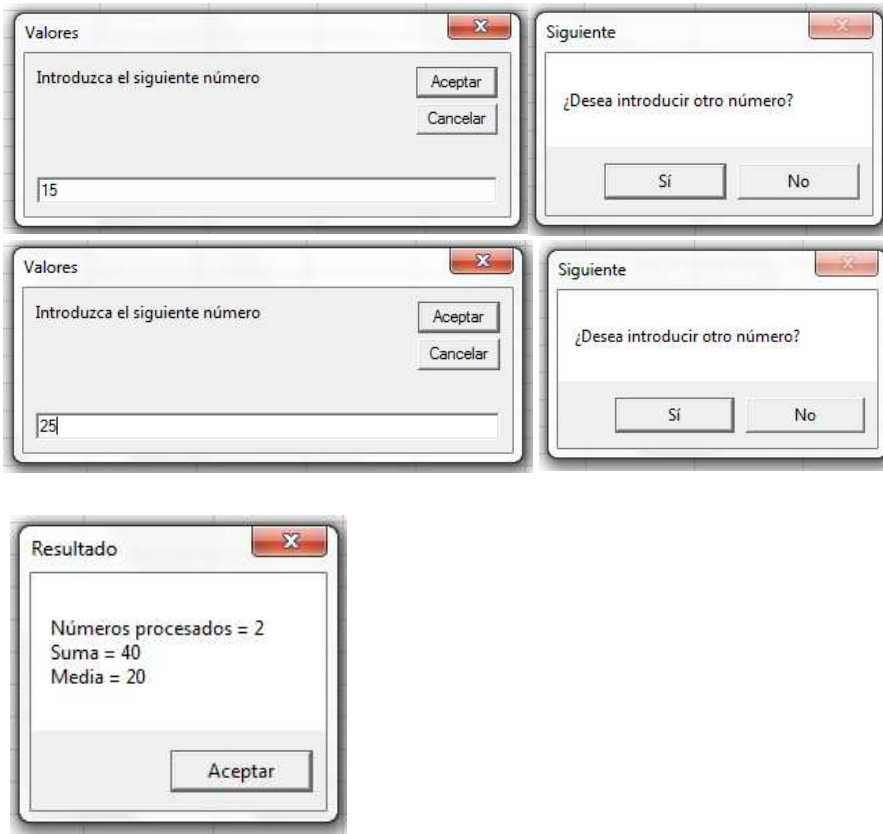
```

```
Loop Until respuesta = vbNo
```

```
media = suma / n
```

```
Final = MsgBox("Números procesados = " & n & Chr(10) & "Suma = " & suma & Chr(10) & "Media = " & media, 0, "Resultado")
```

```
End Sub
```



- b) Diseñar un programa que pregunte al usuario qué temperatura desea como setpoint del reactor. El programa debe identificar como válidas las temperaturas entre 50 y 100°C y, si el usuario introduce un valor fuera del rango, mostrar un mensaje de error.

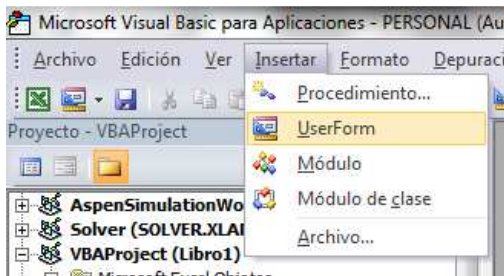
```
Sub TemperaturaConBucle()
Dim Temp As Single
Do
Temp = InputBox("Introduzca la temperatura de operación en °C", "Selección de la Temperatura")
If Temp < 50 Or Temp > 100 Then
Mensaje = MsgBox("La temperatura debe estar entre 50°C y 100°C", vbExclamation + 0, "Error")
End If
Loop Until Temp >= 50 And Temp <= 100
```

End Sub

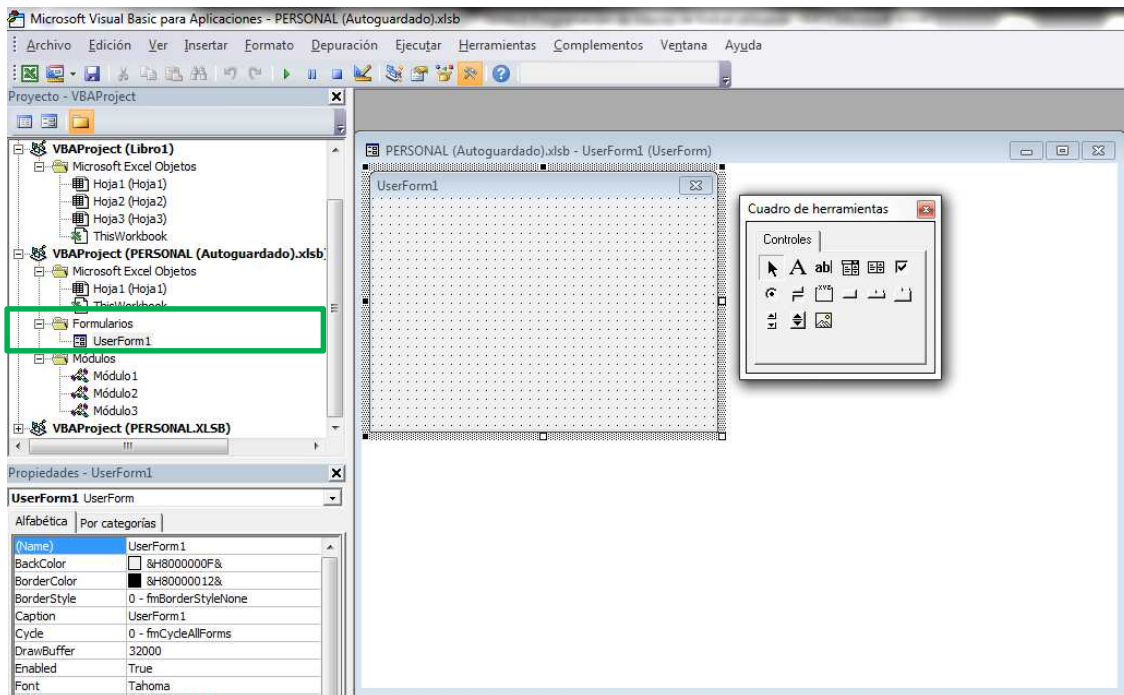


4. FORMULARIOS Y CONTROLES

Un **formulario** es una ventana o cuadro de diálogo que contiene un conjunto de controles insertados por el usuario de entre el conjunto de **controles Activex** disponible. Para crear un formulario nuevo hay que utilizar el comando **UserForm** desde el menú Insertar:



Se abre una nueva ventana que contiene el espacio para diseñar el formulario y los controles que se pueden insertar. Los formularios son elementos independientes de los módulos y se guardan aparte.















Los formularios sirven a tres propósitos principales:

- Como panel para introducir datos y el resultado de un cálculo realizado con dichos datos.
- Para introducir datos en una hoja de cálculo.
- Para recuperar datos almacenados en una hoja de cálculo.

El formulario se crea insertando elementos desde el cuadro de herramientas de controles. Basta con seleccionar el control e insertarlo en el formulario pulsando con el ratón en el lugar

deseado y arrastrando hasta hacer un recuadro. Los controles que se pueden insertar son los siguientes:

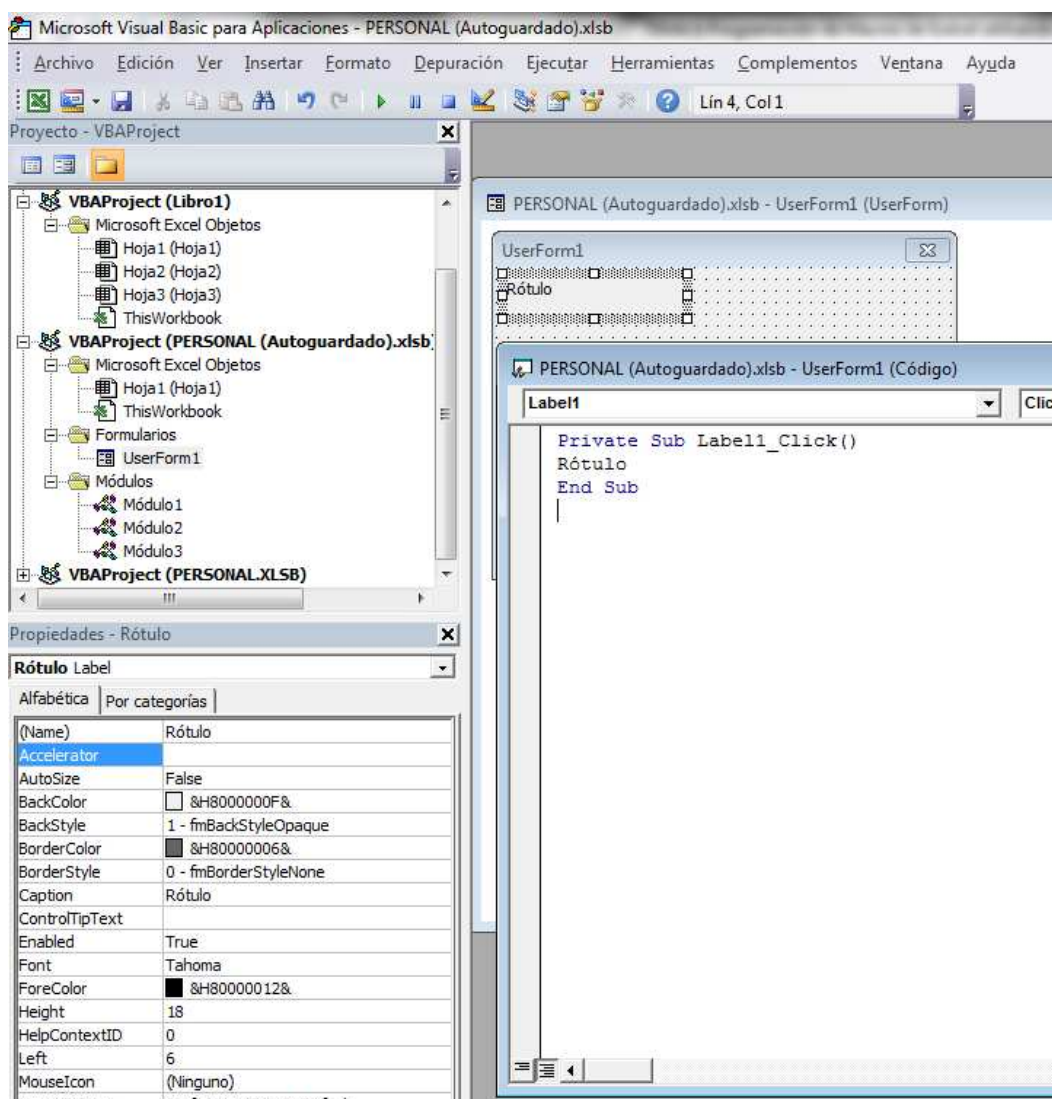
	Control	Qué hace
	Etiqueta	Almacena texto
	Cuadro de texto	Permite introducir texto al usuario
	Cuadro Combinado	Inserta una lista de elementos desplegable
	Cuadro de Lista	Crea una lista de elementos
	Casilla de verificación	Permite seleccionar uno o más elementos
	Botón de opción	Permite seleccionar un solo elemento
	Botón de alternar	Botón on/off
	Marco	Crea un cuadro que contiene otros controles
	Botón de Comando	Crea un botón que se puede presionar
	Barra de tabulaciones	Crea fichas o pestañas
	Página múltiple	Permite crear cuadros de diálogo con fichas
	Barra de desplazamiento	Permite desplazarse en una lista y seleccionar
	Botón de número	Permite desplazarse en una lista y seleccionar
	Imagen	Inserta una imagen
	RefEdit	Permite seleccionar un rango

Los controles tienen propiedades específicas, pero también hay propiedades comunes a todos ellos, que son las siguientes:

Control	Qué hace
Accelerator	Selecciona una letra de entre las del título del control para crear un comando de ejecución rápido del tipo Alt+Letra. La letra seleccionada aparece subrayada en el título.
AutoSize	Si se selecciona True se autoajusta el tamaño del control.
BackColor	Color de fondo.
BackStyle	Estilo del fondo (opaco o transparente).
Caption	Texto que aparece en el control.
Value	Valor del control.
Left and Top	Indica la posición del control.

Width and Height	Determina la anchura y altura del control.
Visible	Si es False, el control se oculta.
Name	Nombre del control. Cada control debe tener un nombre único.
Picture	Imagen para mostrar.

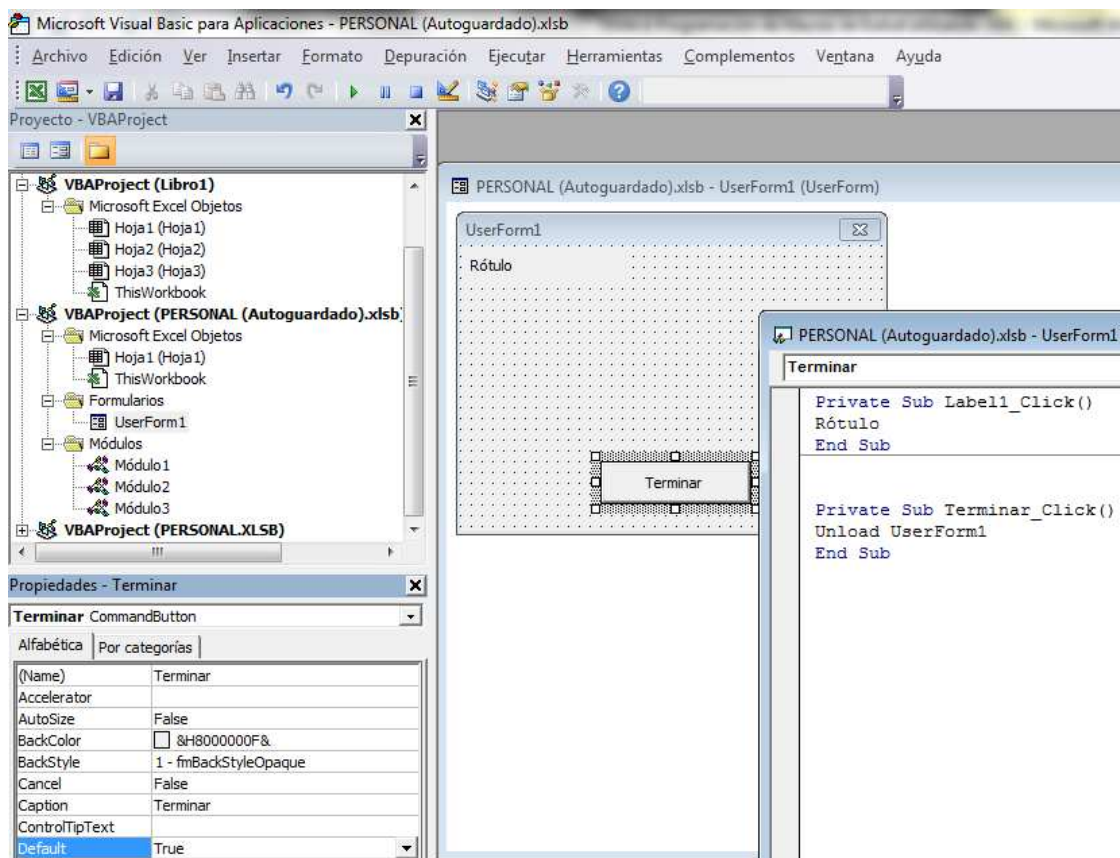
Las propiedades del control pueden ser modificadas bien en el propio control, bien en el código o bien en el cuadro de propiedades del control. En este caso se ha insertado un cuadro de texto y se ha cambiado su nombre por Rótulo.



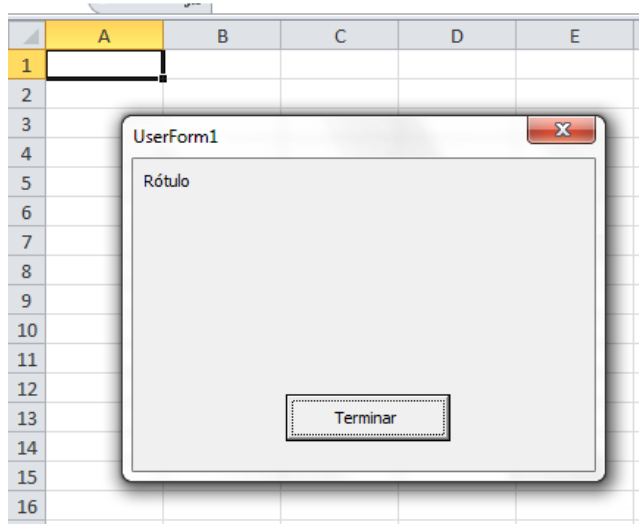
Botón de comando:

Este control permite insertar una determinada acción que se ejecutará al pulsar sobre el botón. Al insertar el control y hacer doble clic en él se abre el módulo donde se tiene que insertar el código a ejecutar. Aunque se puede insertar cualquier acción, lo habitual es que los formularios contengan, entre otras cosas, un botón de comando que permita ejecutar el

código, cerrar el formulario, o ambas cosas. Por ejemplo, se ha creado un botón de comando llamado Terminar, cuyo código es End (finaliza la tarea) y se ha elegido que la propiedad Default⁸ sea True para que al pulsar Enter se ejecute dicho código:



Al ejecutar la macro:

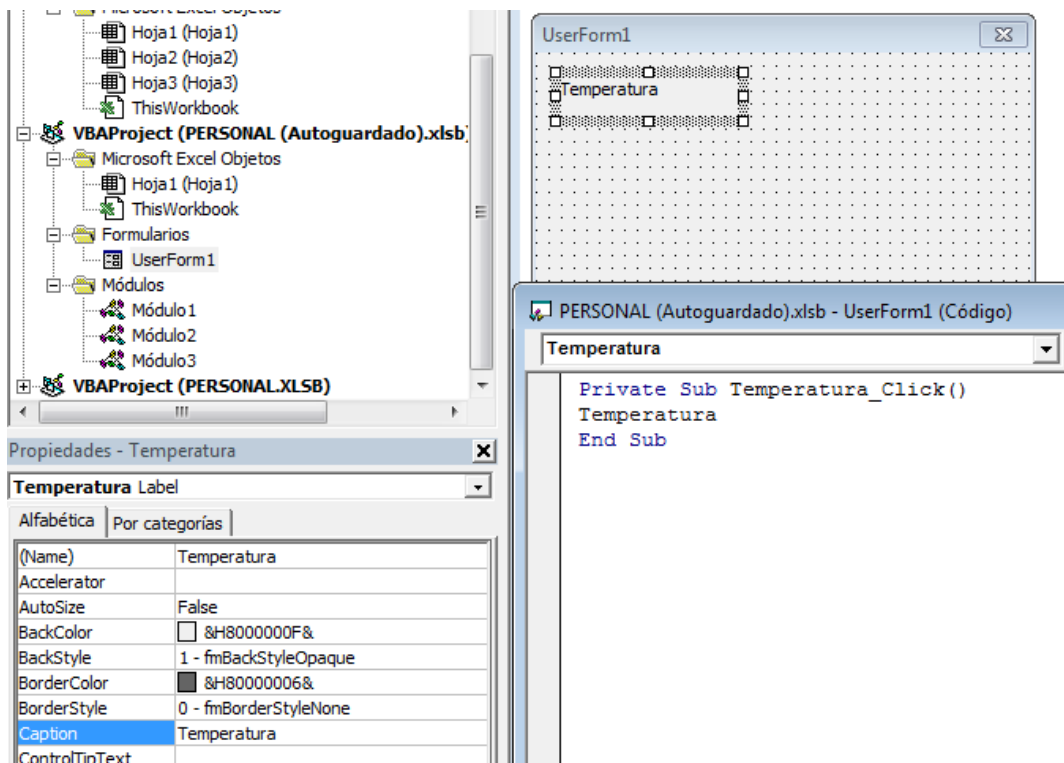


Pulsando el botón o presionando Enter, el formulario se cierra.

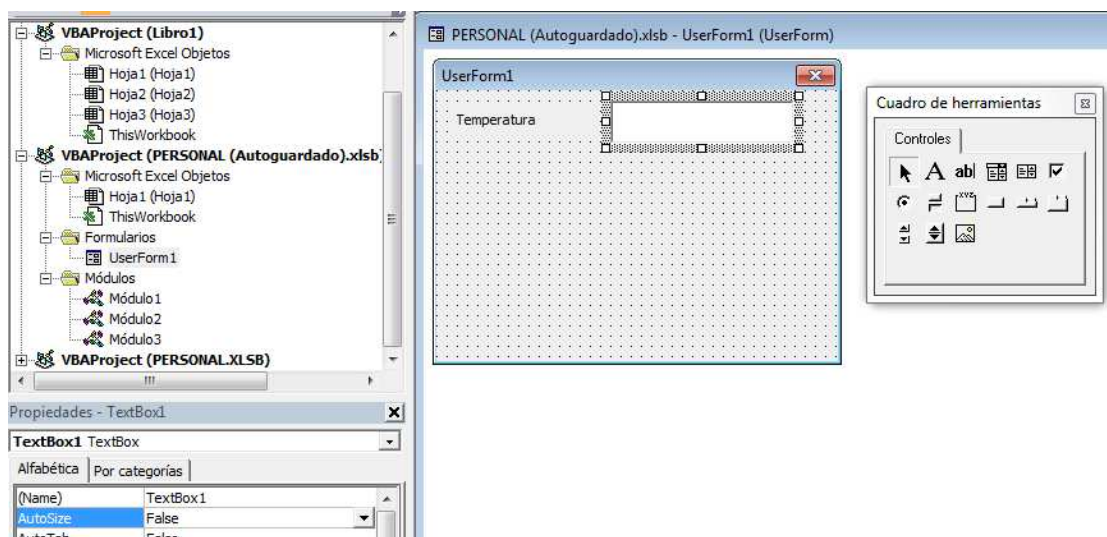
⁸ Si en lugar de la propiedad Default, se selecciona como True la propiedad Cancel, el código se ejecuta pulsando Esc.

Otros comandos habituales son Aceptar y Cancelar. En el siguiente ejemplo se va a crear un formulario con una casilla donde el usuario pueda escribir un valor. Al lado, va a aparecer un rótulo que diga Temperatura. Abajo se van a insertar dos botones de comando: uno que diga Aceptar y que, al pulsarlo, copie el valor dado a la temperatura en la celda A1 de la hoja activa; y otro que diga Cancelar y que cierre el formulario:

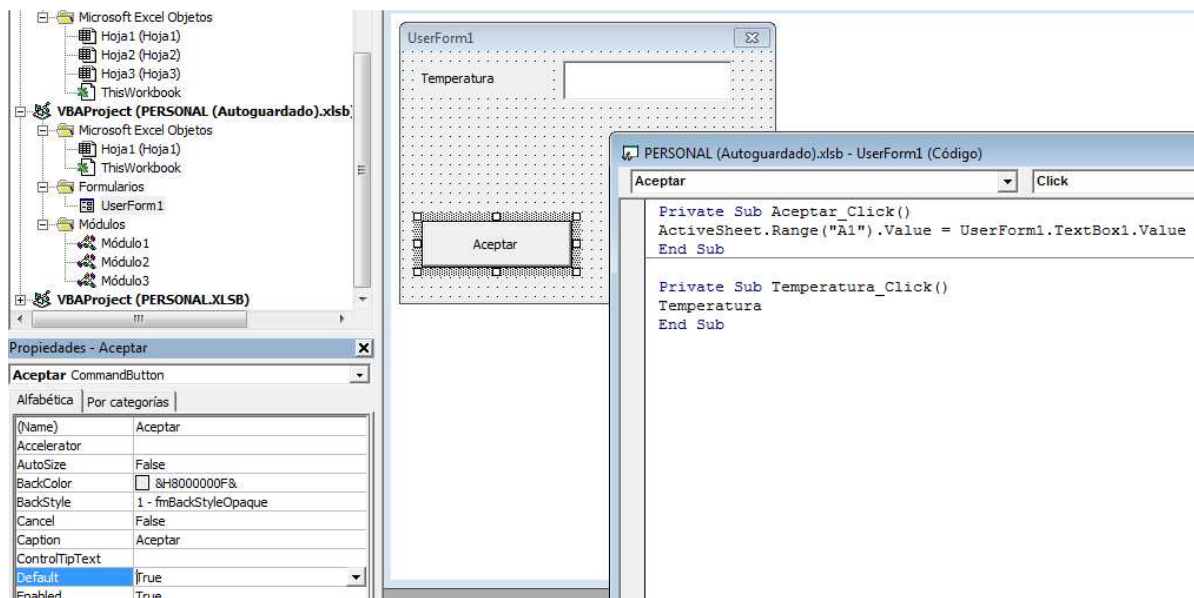
El primer paso es crear la etiqueta y hacer que contenga el rótulo Temperatura (caption):



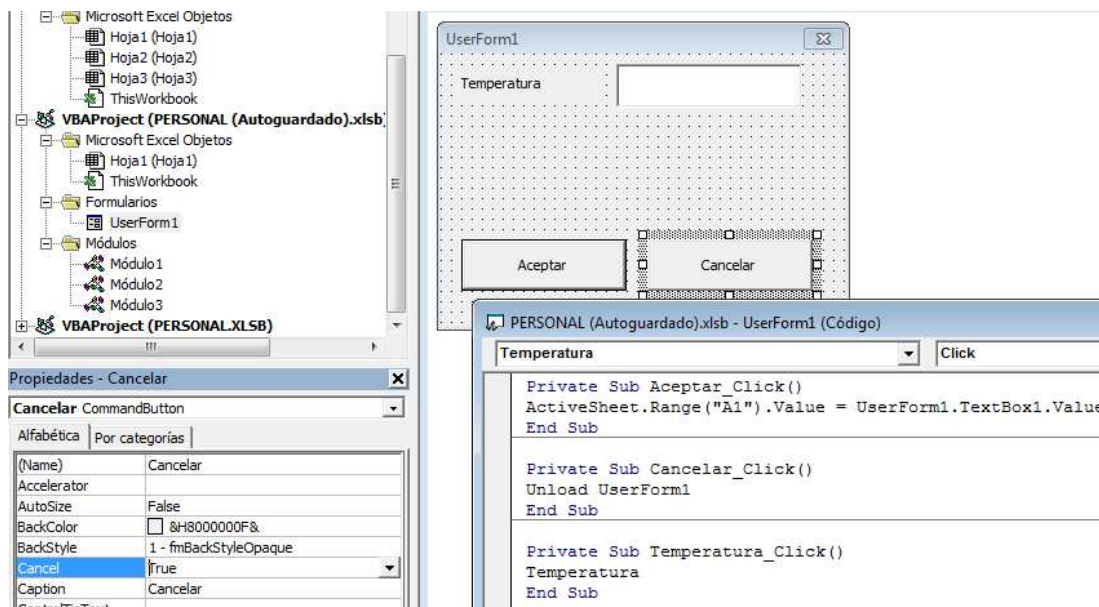
A continuación se inserta el cuadro de texto donde el usuario tiene que escribir la temperatura. No se le ha cambiado el nombre, así que por defecto es TextBox1:



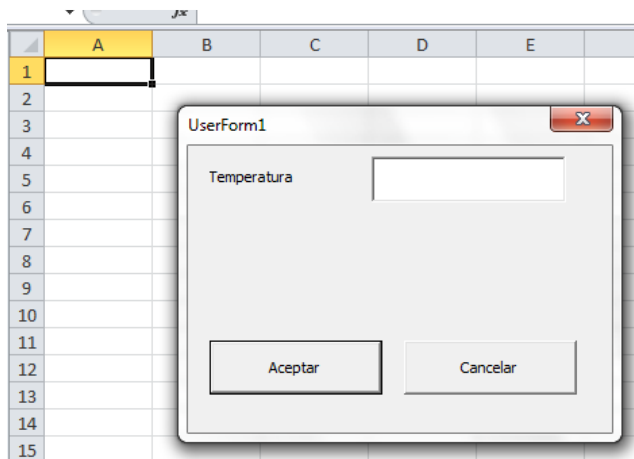
A continuación se inserta un botón de comando, se le llama Aceptar y se usa ese mismo nombre como caption. El código indica que el valor de la celda A1 de la hoja activa debe ser igual al valor del cuadro de texto TextBox1 del formulario UserForm1 que se está creando. Se ha seleccionado Default = True para que se ejecute el comando al pulsar Enter y no sólo al presionar el botón:



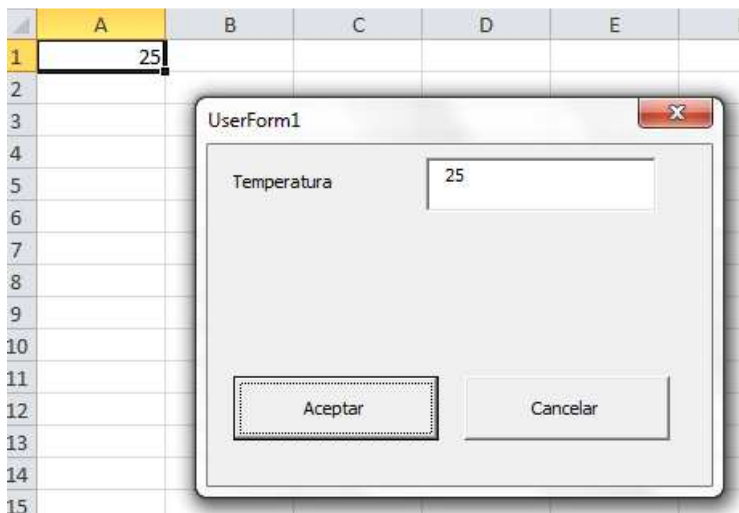
Finalmente, se ha insertado un botón de comando con el nombre y la caption Cancelar, cuyo código es terminar y cerrar el formulario. Se ha elegido Cancel = True para que el código se ejecute también al presionar Esc, como es habitual cuando se trata de cerrar un formulario:



Cuando se ejecuta el formulario aparece esto en pantalla:



Si se escribe un valor en la casilla y se pulsa Aceptar o Enter, el valor se escribe en la celda A1. Si se pulsa Cancelar o Esc, el formulario se cierra y se vuelve al Editor de VB.



Cuadro combinado:

Sirve para crear una lista desplegable de elementos de entre los que se pueda seleccionar uno o varios para realizar alguna acción con ellos. Los elementos se pueden introducir mediante peticiones por teclado o ser importados desde una tabla o rango de celdas en una hoja de cálculo.

Por ejemplo, se puede crear un cuadro combinado con una lista de elementos que se van introduciendo a través de un cuadro de texto:

El formulario contiene una etiqueta cuya caption es Introduzca un elemento. Después se ha añadido un cuadro de texto llamado Elementos donde el usuario tiene que escribir dichos elementos. Después se ha insertado un cuadro combinado con nombre Lista. De entre sus propiedades se ha elegido ListRows = 3, es decir, que muestre sólo 3 elementos a la vez y, si hay más elementos, aparezca una barra de desplazamiento. A continuación se ha añadido un botón de comando con el nombre OK y caption Aceptar. El código de ese botón incluye:

Private Sub OK_Click() al hacer click en el botón

Lista.AddItem Elementos.Text el elemento escrito en el cuadro de texto Elementos se añade al cuadro combinado de nombre Lista

Elementos.Text = "" después se borra el contenido del cuadro Elementos

Elementos.SetFocus y se vuelve a poner el cursor en dicho cuadro para escribir el siguiente elemento

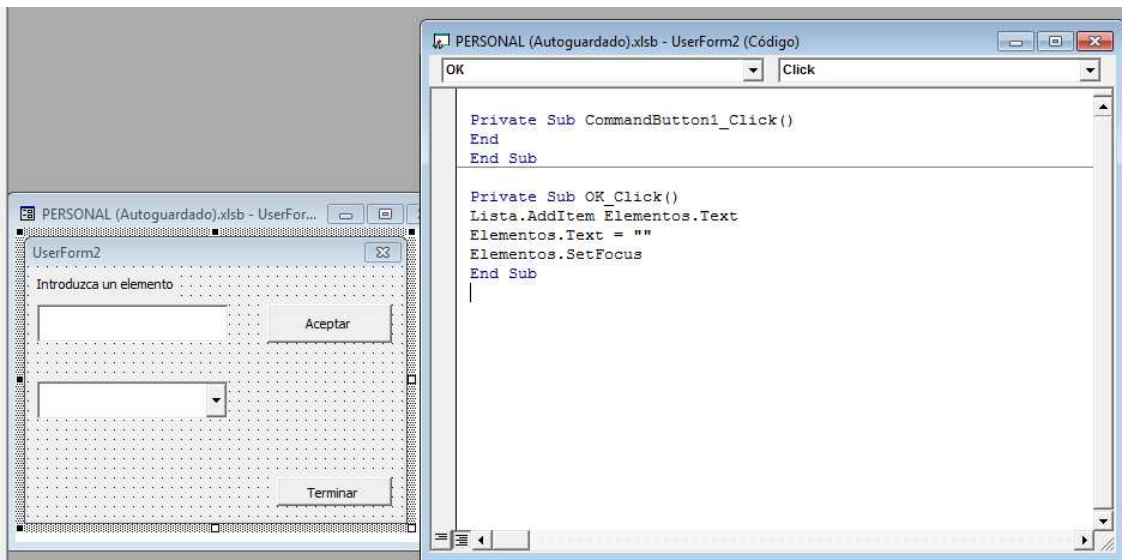
End Sub

Finalmente, se ha añadido un botón de comando con la caption Terminar y cuyo código es:

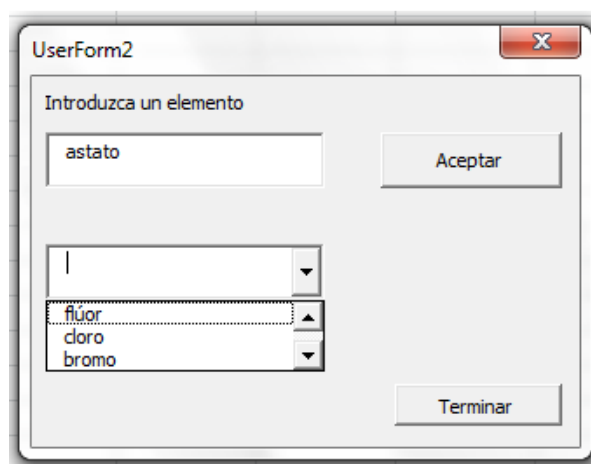
Private Sub CommandButton1_Click() al hacer clic en el botón

End termina de ejecutar la macro y cierra el formulario

End Sub



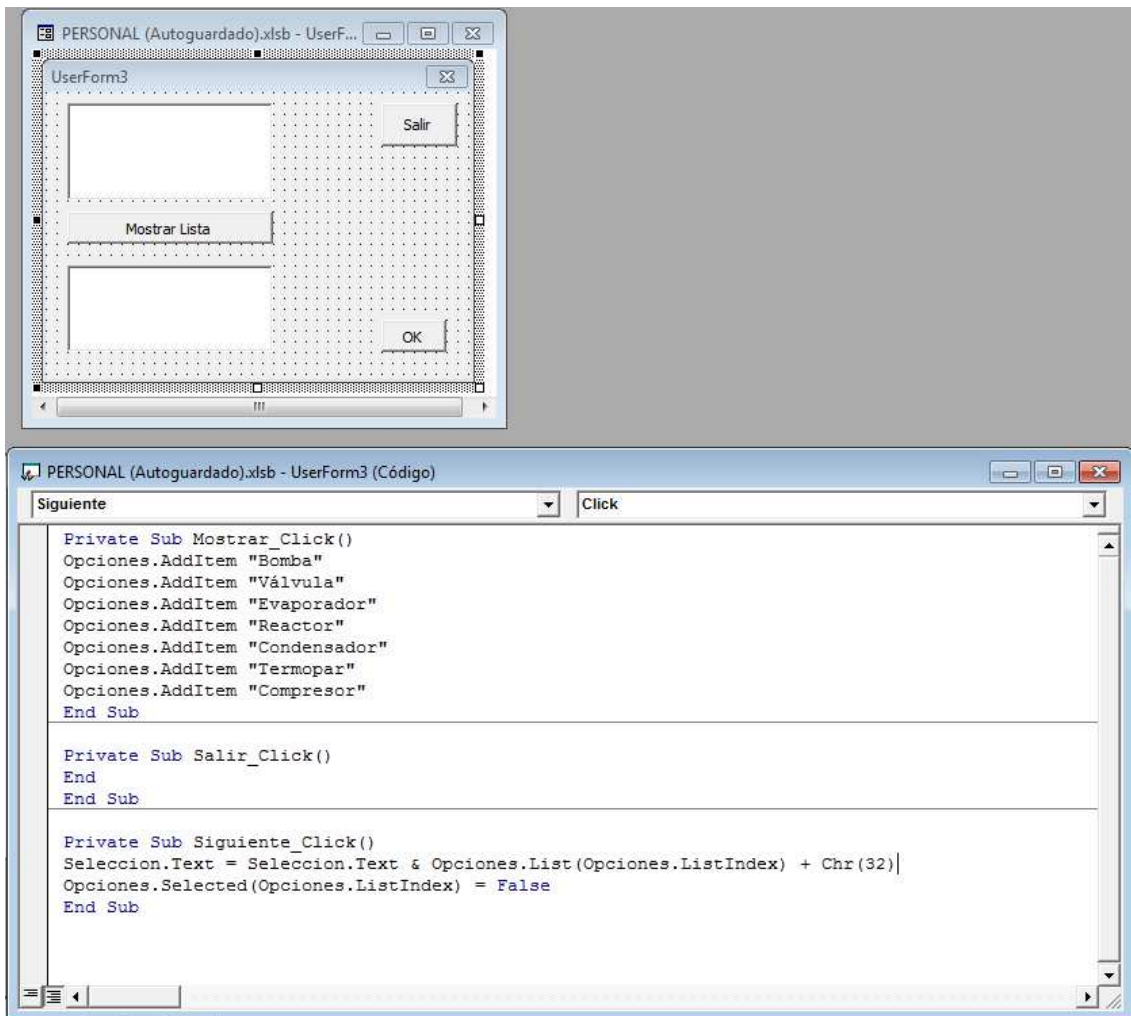
El resultado es:



Cuadro de lista:

Permite insertar un conjunto de elementos que pueden ser seleccionados. Los elementos seleccionados se pueden mostrar en forma de lista en un cuadro de texto del formulario o en un rango de celdas de una hoja de Excel.

Por ejemplo, se puede crear una lista de equipos de entre la que se puedan seleccionar varios elementos para confeccionar otra lista que aparezca en un cuadro de diálogo.

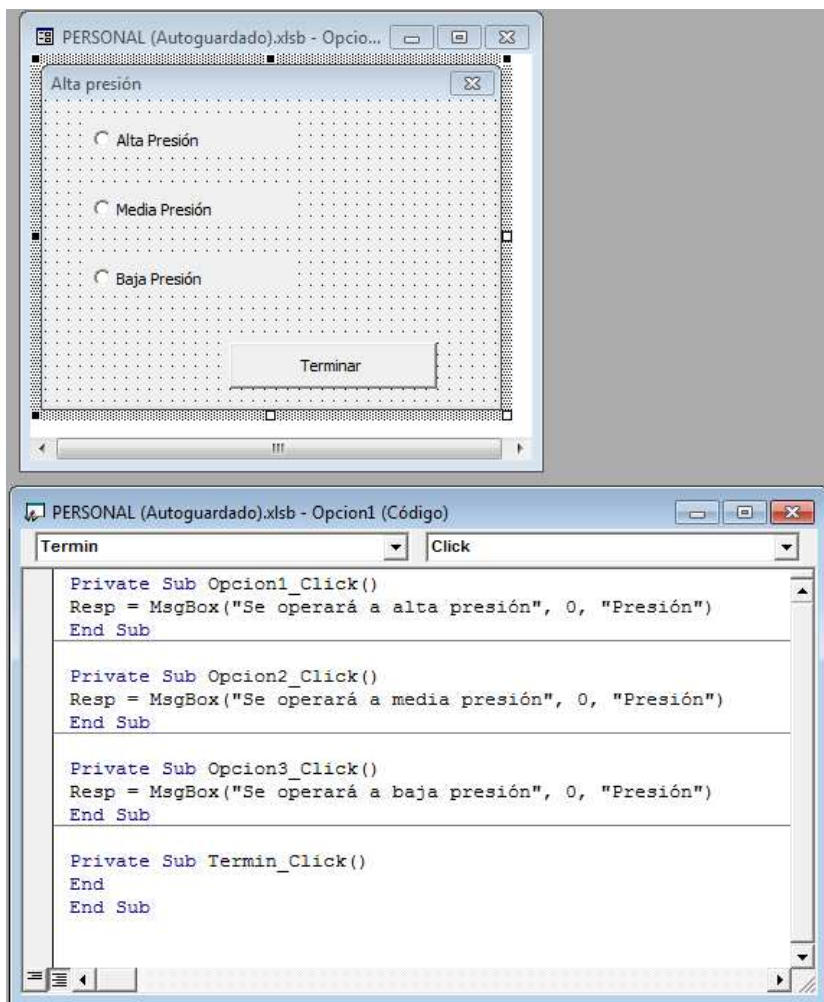


Al presionar el botón Mostrar lista se hace visible la lista y al apretar Ok el elemento seleccionado se incorpora al recuadro:

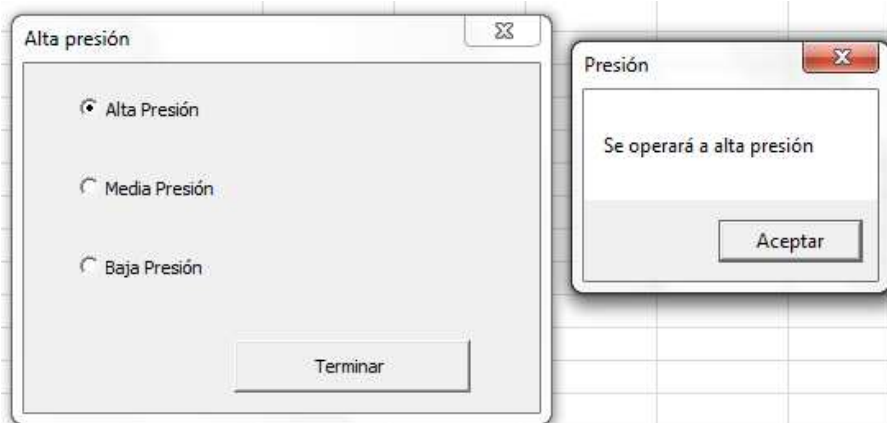


Botón de opción:

Este control permite seleccionar un único elemento de entre una lista de opciones. Por ejemplo, se puede crear un formulario que dé a elegir al usuario si el reactor tiene que operar a alta, media o baja presión y que muestre un cuadro de mensaje distinto en función de la opción seleccionada:

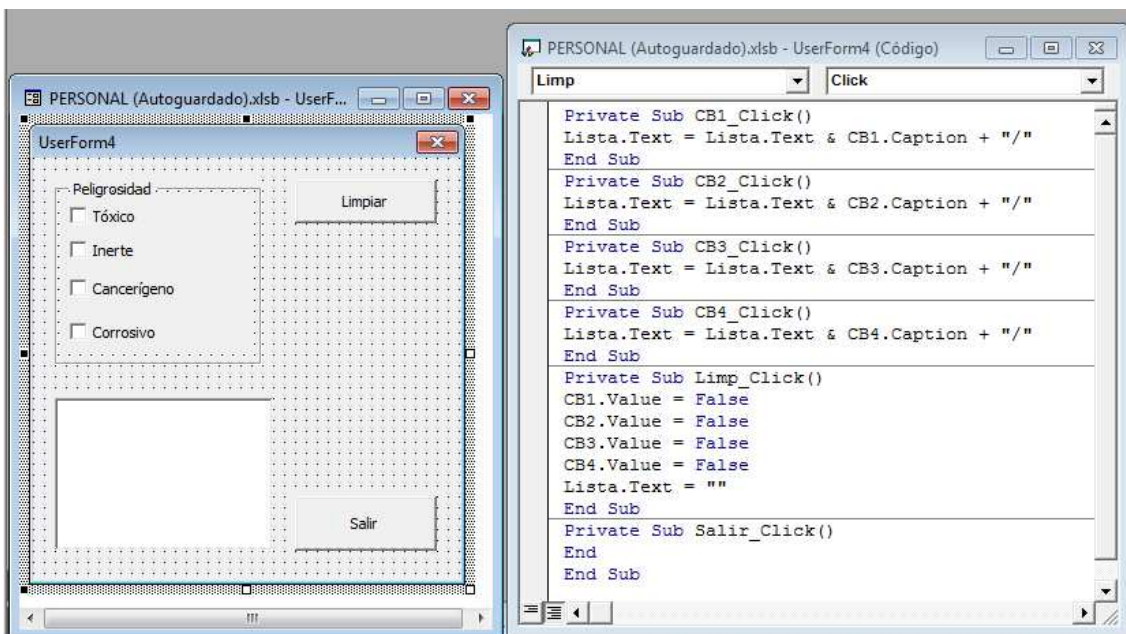


El resultado, al presionar una opción, es:



Casilla de verificación:

Permite seleccionar varios elementos a la vez de entre las opciones posibles, ya que no son excluyentes entre sí. Por ejemplo, se puede crear una lista de opciones que contenga las categorías de peligrosidad de un reactivo, de manera que se puedan seleccionar varias categorías que se muestren en un cuadro de texto:



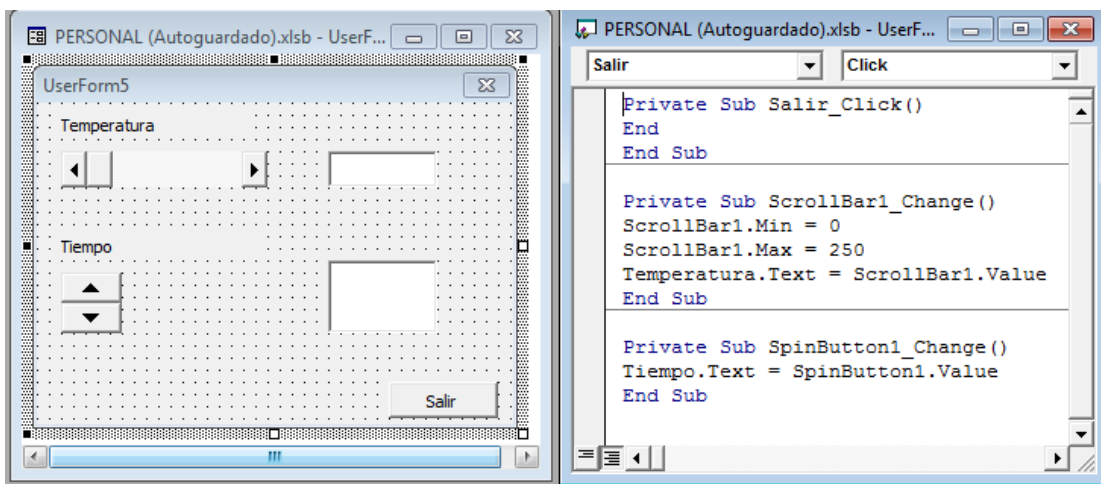
Las opciones se han incluido en un marco o frame con la caption Peligrosidad. Cada casilla de verificación se ha llamado CB1 a CB4 y al seleccionar cada una, la caption correspondiente se escribe en el cuadro de texto, que se ha llamado Lista, junto con una barra de separación.



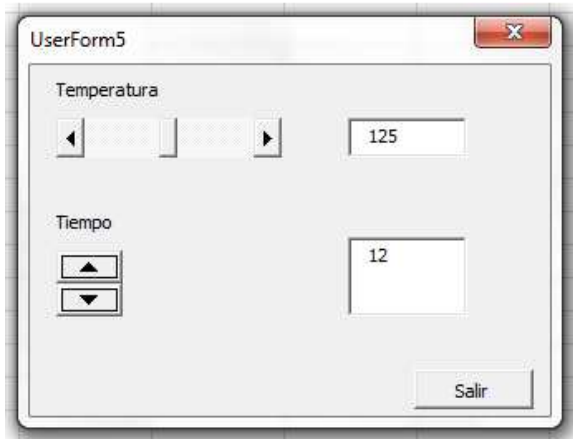
Al pulsar en el botón Limpiar, se deseleccionan las opciones marcadas y se borra el contenido del cuadro de texto. Al pulsar Salir, se termina la macro.

Barra de desplazamiento y botón de número:

Estos controles permiten dar un valor a una variable en función del desplazamiento que se haya realizado en dicho botón o del número de veces que se haya pulsado. Por ejemplo, se puede insertar una barra de desplazamiento que ajuste la temperatura de un reactor y un botón de número que seleccione el tiempo de reacción, mostrando los valores seleccionados en cuadros de texto:



Se ha insertado una barra de desplazamiento llamada ScrollBar1 que puede tomar valores entre 0 y 250, escribiendo dicho valor en un cuadro de texto llamado Temperatura. Se ha insertado, además, un botón de número, llamado SpinButton1, cuyo valor se va a escribir en otro cuadro de texto llamado Tiempo. Finalmente, el botón salir cierra la aplicación.



ANEXO:

Caracteres Ascii más habituales:

Código	vba	Descripción	Código	vba	Descripción
9	vbTab	tabulación horizontal	46		.
10	vbLf	salto de línea	47		/
11	vbVerticalTab	tabulación vertical	58		:
12		página nueva	59		;
13	vbCr	retorno del carro	60		<
24		cancelar	61		=
27		escape	62		>
32		espacio	63		?
33		!	64		@
34		"	91		[
35		#	92		\
36		\$	93]
37		%	94		^
38		&	95		_
39		'	96		`
40		(123		{
41)	124		
42		*	125		}
43		+	126		~
44		,	127		borrar
45		-			

Colores usando la propiedad ColorIndex:

	A	B	C	D	E	F	G	H	I
1	1	2	3	4	5	6	7	8	
2	9	10	11	12	13	14	15	16	
3	17	18	19	20	21	22	23	24	
4	25	26	27	28	29	30	31	32	
5	33	34	35	36	37	38	39	40	
6	41	42	43	44	45	46	47	48	
7	49	50	51	52	53	54	55	56	

© Excel-Pratique.com

Colores usando la propiedad Color = RGB(R, G, B):

RGB(0, 0, 0): negro

RGB(255, 255, 255): blanco

RGB(255, 0, 0): rojo

RGB(0, 255, 0): verde

RGB(0, 0, 255): azul