

Hojas de cálculo

1. Programación de macros en Visual Basic

Nicolás Serrano

Organización Industrial

Contenido

- Visual Basic para Aplicaciones
- Primera macro
- Opciones de grabar macro
- Gestor de macros
- Visualizar y editar una macro
- IDE
- Tipos de macros
- Crear función definida por el usuario
- Ejecutar una macro
- Procedimientos
- Debugger

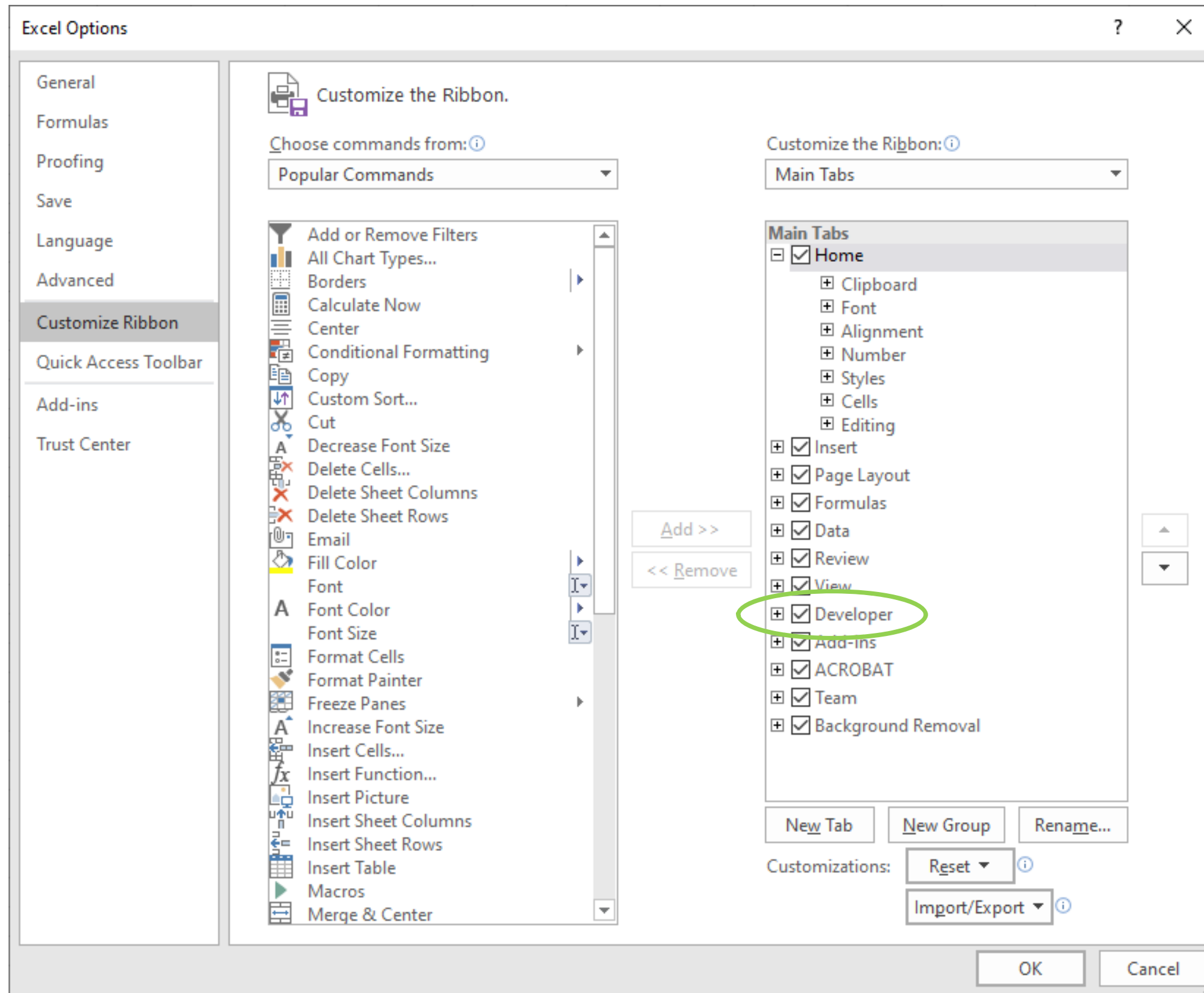
Visual Basic para Aplicaciones

- Herramienta de desarrollo
- Compatible con Visual Basic y común a otras aplicaciones Office
- IDE (Integrated Development Environment)
- Formularios
- Código
 - Módulos de código
 - Módulos de clase
- Lenguaje de script o de macros

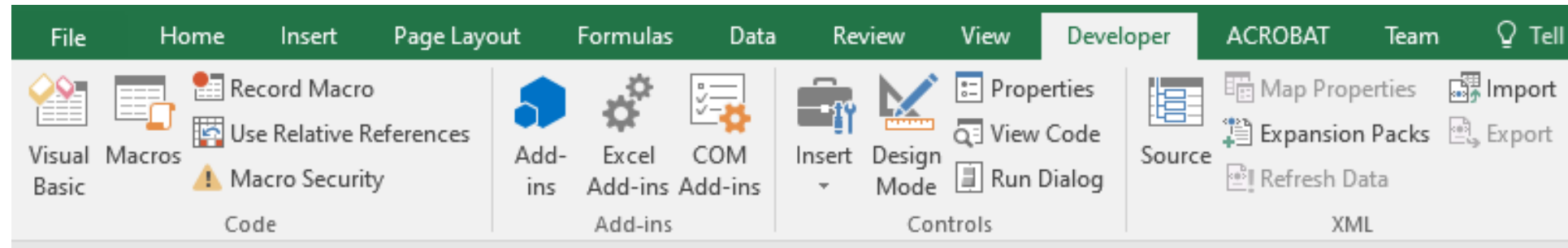
Activar tab de Developer

- Menú File -> Options
- En Customize Ribbon
 - Activar en tabs:
 - Developer
 - Clic en OK

Se muestra Developer en el menu



Primera macro

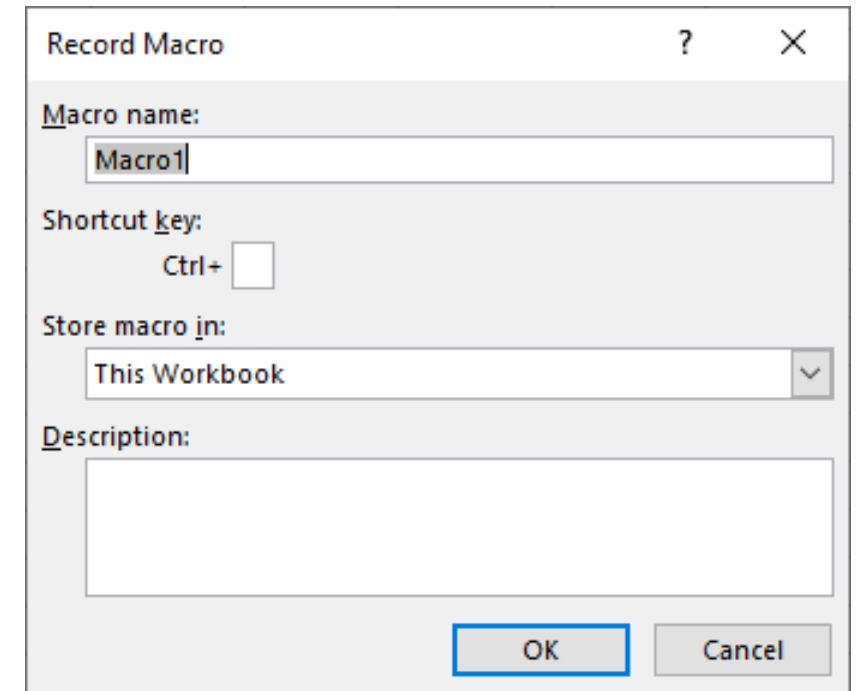


- Grabadora de macros:
-> Developer / Code / Record Macro
- Ejecutar las ordenes deseadas,
escribir en casilla D7 el número 12
- Detener la grabación
-> Developer / Code / Stop Recording
- El código producido se puede ver en:
-> Developer / Code / Macros
Seleccionar Macro1 y clicar en Edit

```
Sub Macro1()  
,  
, Macro1 Macro  
,  
,  
  
Range("D7").Select  
    ActiveCell.FormulaR1C1 = "12"  
End Sub
```

Opciones de grabar macro

- Nombre de la macro (sin espacios)
- Método abreviado: para ejecutar con el teclado
- Guardar macro en: especificar fichero de Excel
 - Libro de macros (Personal Macro Workbook)
 - En otra hoja (New Workbook)
 - Con la propia hoja (This Workbook)
- Descripción

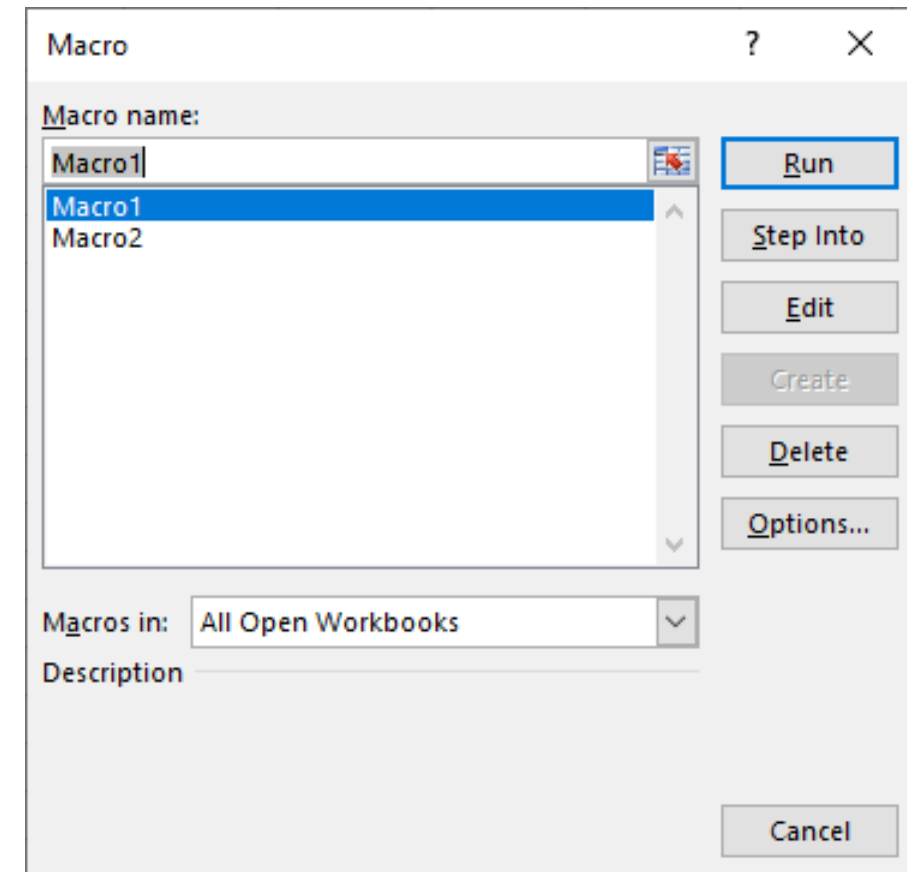


The image shows the 'Record Macro' dialog box from Microsoft Excel. The dialog has a title bar with a question mark and a close button. It contains the following fields and controls:

- Macro name:** A text box containing 'Macro1'.
- Shortcut key:** A label followed by 'Ctrl+' and an empty text box for specifying a key.
- Store macro in:** A dropdown menu currently showing 'This Workbook'.
- Description:** A large empty text area for providing a description of the macro.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

Gestor de macros

- Abrir el cuadro de diálogo Macro
 - > Developer / Code / Macros
- Desde esta ventana se permite:
 - Ver las macros existentes
 - Ejecutar (Run)
 - Debugger (Step into)
 - Editar la macro
 - Crear nueva macro
 - Borrar una macro
 - Cambiar las opciones de una macro



Visualizar y editar una macro

- En el cuadro de diálogo Macro
 - > Developer / Code / Macros
 - Seleccionar la macro
 - Clicar Modificar
- Se abre el editor de Visual Basic con el módulo en el que se encuentra la macro.
- Dispone de diversos elementos que se muestran en la imagen siguiente

IDE

Visual Basic for Applications

Explorador de proyectos

Ventana de propiedades

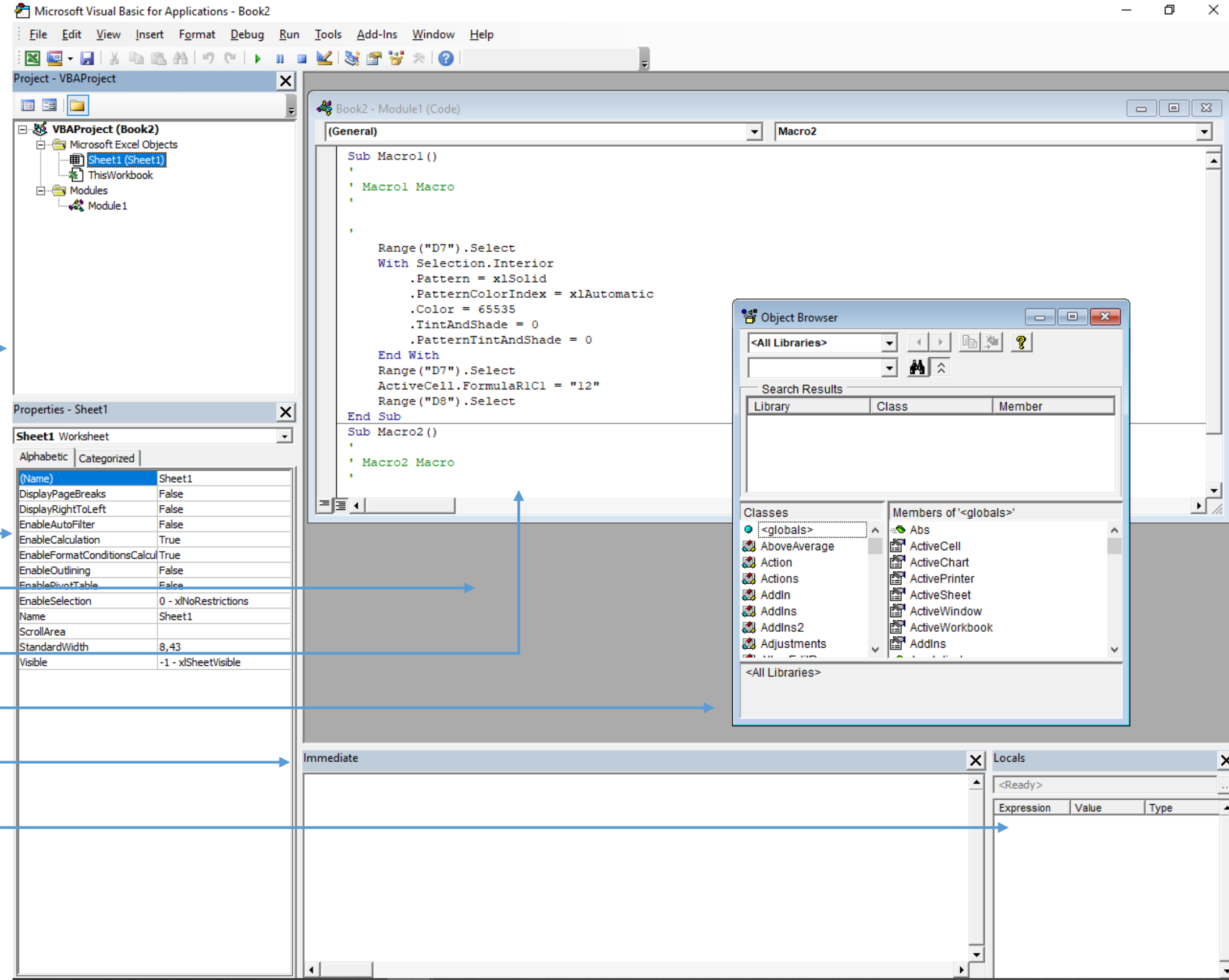
Área de trabajo

Código

Examinador de objetos

Ventanas de expresiones

Ventana de variables



Tipos de macros

- Macros de orden
 - Procedimientos Sub
 - Normalmente ejecutan comandos de menú
- Funciones definidas por el usuario
 - Procedimientos Function
 - Amplían las funciones incorporadas en la aplicación
 - No debe modificar el entorno
 - Devuelve un valor

Crear función definida por el usuario

- En un módulo, escribir `Function nombreFuncion` y los parámetros
- Escribir el cuerpo de la función
- Finalizar con `nombreFuncion = valor del Resultado`
- Ejemplo:

```
Function cuadrado(x)  
    cuadrado = x * x  
End Function
```

Formas de ejecutar una macro

- En el cuadro de diálogo Macro
 - > Developer / Code / Macros
 - Seleccionar la macro y clicar Run
- Clicar en el código de la macro y clicar Run -> Continue (o F5)
- Mediante la tecla de método abreviado
- Si se ha asignado a un botón de la barra de herramientas, mediante dicho botón
- Si es una función, introduciendo la función:
=cuadrado(2)

Procedimientos

- Estructura:

```
Sub NombreProcedimiento (argumento1, argumento2, ...)  
    [sentencias VBA]  
End Sub
```

```
Function NombreProcedimiento (argumento1, argumento2, ...)  
    [sentencias VBA]  
    NombreProcedimiento = valorRetorno  
End Function
```

- Llamada:

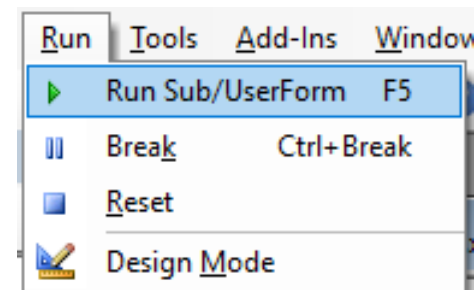
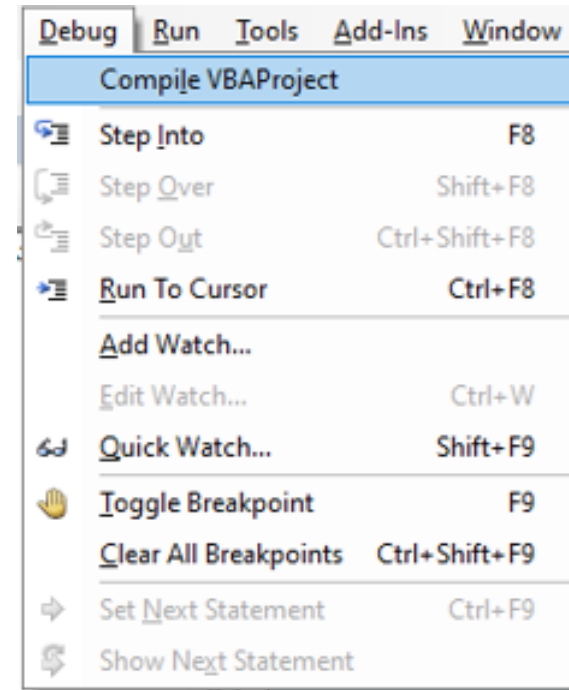
```
NombreProcedimiento (argumento1, argumento2, ...)
```

- Llamada desde otro fichero:

```
NombreFichero.NombreProcedimiento (argumento1, argumento2, ...)
```

Debugger

- Permite situar breakpoints
 - Haciendo clic en la línea
- Ejecutar paso a paso
- Inspeccionar variables
 - Sobre el código
 - En la ventana "Locals "
 - En la ventana "Immediate "
 - Debug.Print (imprime en Immediate)
- Arrancar y parar macros
 - En modo debugger, se para el funcionamiento normal del Excel



Hojas de cálculo

2. Lenguaje VBA

Nicolás Serrano
Organización Industrial

Contenido

- Variables
- Constantes y operadores
- Condiciones
- Bucles For...Next, For Each...Next
- Bucles Do While

Variables

- Declaración:

Dim nombreVariable [As TipoVariable]

- Tipos de variables

- Boolean TRUE/FALSE
- Integer, Long 2 y 4 bytes
- Single, Double 4 y 8 bytes
- String 1 byte por carácter
- Currency, Date 8 bytes
- Object 4 bytes
- Array
- Variant

Referencia en:

<https://docs.microsoft.com/en-us/office/vba/language/reference/user-interface-help/data-type-summary>

- Dimensión de arrays

Dim nombreVariable (limiteInferior1 To limiteSuperior1, limiteInferior2 To limiteSuperior2, ...) [As TipoVariable]

Constantes y operadores

- Constantes de la aplicación
 - Const xlMaximized = -4137 (&HFFFFFFD7)
 - ActiveWindow.WindowState = xlMaximized
- Constantes definidas por el usuario
 - Const NOMBRECONSTANTE = valor
- Operadores aritméticos: +, -, *, /, ^, Mod
- Operadores de comparación: =, >, <, >=, <=, <>
- Operadores lógicos:
 - TRUE Returns the logical value TRUE
 - FALSE Returns the logical value FALSE
 - NOT Reverses the logic of its argument
 - OR Returns TRUE if any argument is TRUE
 - AND Returns TRUE if all its arguments are TRUE
 - IF Specifies a logical test to perform

Sintaxis If...Then...Else

If condition Then

[statements]

[Elself condition-n Then

[elseifstatements]]

[Else

[elsestatements]]

End If

```
Sub test()
```

```
    Dim Half As Integer
```

```
    Dim Module As Integer
```

```
    Dim Number
```

```
    Number = Int((100 * Rnd) + 1) ' Initialize variable.
```

```
    Half = Number / 2
```

```
    If Half * 2 = Number Then
```

```
        Module = 0
```

```
    Else
```

```
        Module = 1
```

```
    End If
```

```
    Debug.Print (Number & " module 2: " & Module)
```

```
End Sub
```

[Reference example](#)

Sintaxis For...Next, For Each...Next

```
For counter = start To end [Step step]
    [statements]
    [Exit For]
    [statements]
Next [counter]
```

```
For Each element In group
    [statements]
    [Exit For]
    [statements]
Next [element]
```

```
Sub testFor()
    Debug.Print ("Funcion testFor")
    For i = 1 To 5
        Call test
    Next
End Sub
```

[Reference example](#)

```
Sub testForEach()
    Debug.Print ("Funcion testForEach")
    For Each cell In Range("A1:A5")
        Debug.Print ("cell " & cell.Value)
    Next
End Sub
```

[Reference example](#)

For Next

```
Sub Ejemplo_For_Next()  
  
For Contador = 0 To 9  
    If Selection.Offset(Contador, 0).Value > 20 Then  
        With Selection.Offset(Contador, 1).Interior  
            .ColorIndex = 6  
            .Pattern = xlSolid  
        End With  
    End If  
Next  
  
End Sub
```

For Each

```
Sub Ejemplo_For_Each_Next()  
,  
For Each celda_en_bucle In Range("A1:A5")  
    If celda_en_bucle.Value > 20 Then  
        With celda_en_bucle.Offset(0, 1).Interior  
            .ColorIndex = 6  
            .Pattern = xlSolid  
        End With  
    End If  
Next  
End Sub
```

Sintaxis Do Loop

```
Do [{While | Until} condition]
    [statements]
    [Exit Do]
    [statements]
Loop
```

```
Do
    [statements]
    [Exit Do]
    [statements]
Loop [{While | Until} condition]
```

```
Sub testDo()
    Debug.Print ("Funcion testDo")
    i = 1
    Do While i <= 5
        Call test
        i = i + 1
    Loop
End Sub
```

[Reference example](#)

Do Loop

```
Sub Ejemplo_Do_Loop()  
  
    Contador = 0  
    Do Until Selection.Offset(Contador, 0).Row > 100  
        If Selection.Offset(Contador, 0).Value > 20 Then  
            With Selection.Offset(Contador, 1).Interior  
                .ColorIndex = 6  
                .Pattern = xlSolid  
            End With  
        End If  
        Contador = Contador + 1  
    Loop  
  
End Sub
```


Hojas de cálculo

3. Objetos en VBA

Nicolás Serrano
Organización Industrial








Contenido

- Objetos
- Estructura jerárquica
- Métodos
- Eventos
- Colecciones
- Examinador de objetos
- Asignación y propiedades
- Objeto Application
- Objeto Workbook
- Objeto Worksheet
- Objeto Range
- Filas y columnas del objeto Range
- Propiedades y métodos del objeto Range

Objetos

- La interacción con el entorno se realiza mediante objetos
- Objeto en VBA es todo elemento manipulable (Rango, Window)
 - Modificando una propiedad del objeto
 - Ejecutando un método del objeto
 - Definiendo un procedimiento para un evento del objeto
- Ejemplo de objeto: Worksheet
 - Propiedad: Cells, CircularReference
 - Método: Calculate, CheckSpelling
 - Evento: Change

Members of 'Worksheet'

-  Calculate
-  Cells
-  Change
-  ChartObjects
-  CheckSpelling
-  CircleInvalid
-  CircularReference

Propiedades

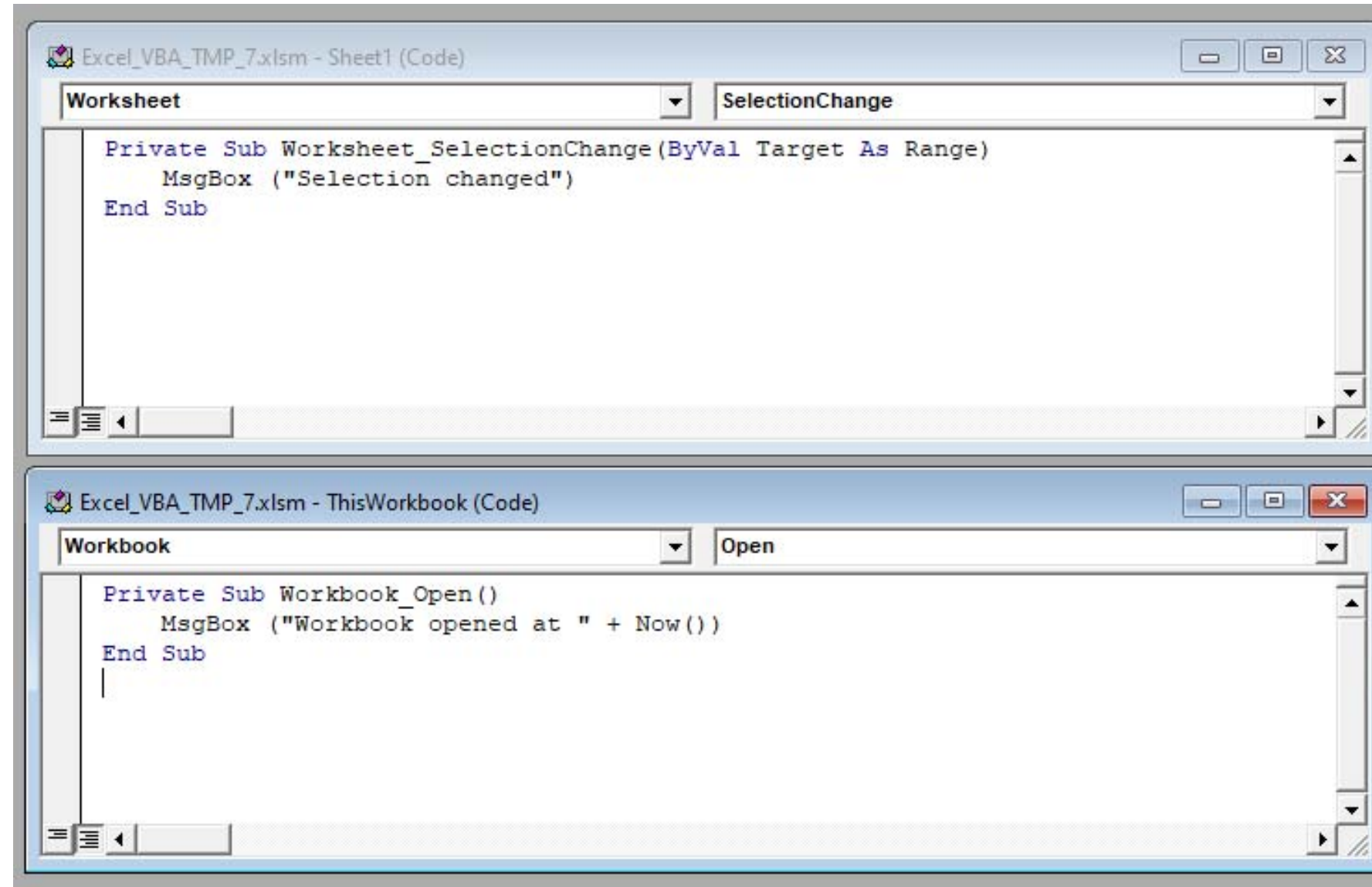
- Una propiedad de un objeto puede ser otro objeto (estructura jerárquica)
- Para especificar un objeto de la jerarquía se indica la ruta desde el origen:
 - `Application.Workbooks("Libro1").Worksheets("Hoja1").Range(A1:A1)`
- Reducción de la ruta:
 - `Application.ActiveWindow.ActiveCell.Font.Italic`
- Se puede reducir a:
 - `ActiveCell.Font.Italic`
- Establecer valores de propiedades
 - `Objeto.Propiedad = valor` (numérico, cadena de caracteres o lógico)
- Obtener un valor
 - `variable = Objeto.Propiedad`

Métodos

- Los métodos dan ordenes a los objetos
- Ejecución de un método:
 - Objeto.Método
 - Ej: `ActiveWorkbook.Save`
- Si tiene argumentos
 - `Objeto.Método (argumento1, argumento2, ...)`
 - `Objeto.Método nombreArgumento1:=argumento1, _ nombreArgumento2:=argumento2, ...`
 - Ej: `ActiveWorkbook.SaveAs([Filename], [FileFormat], [Password], [WriteResPassword], [ReadOnlyRecommended], [CreateBackup], [AccessMode As XlSaveAsAccessMode = xlNoChange], [ConflictResolution], [AddToMru], [TextCodepage], [TextVisualLayout])`

Eventos

- Es algo que le sucede al objeto (por ejemplo Archivo / Abrir)
- Se pueden escribir procedimientos de respuesta al evento
- Se describen en la ventana de módulo cuando se selecciona un objeto en la lista de objetos
- La lista de procedimientos muestra los eventos del objeto

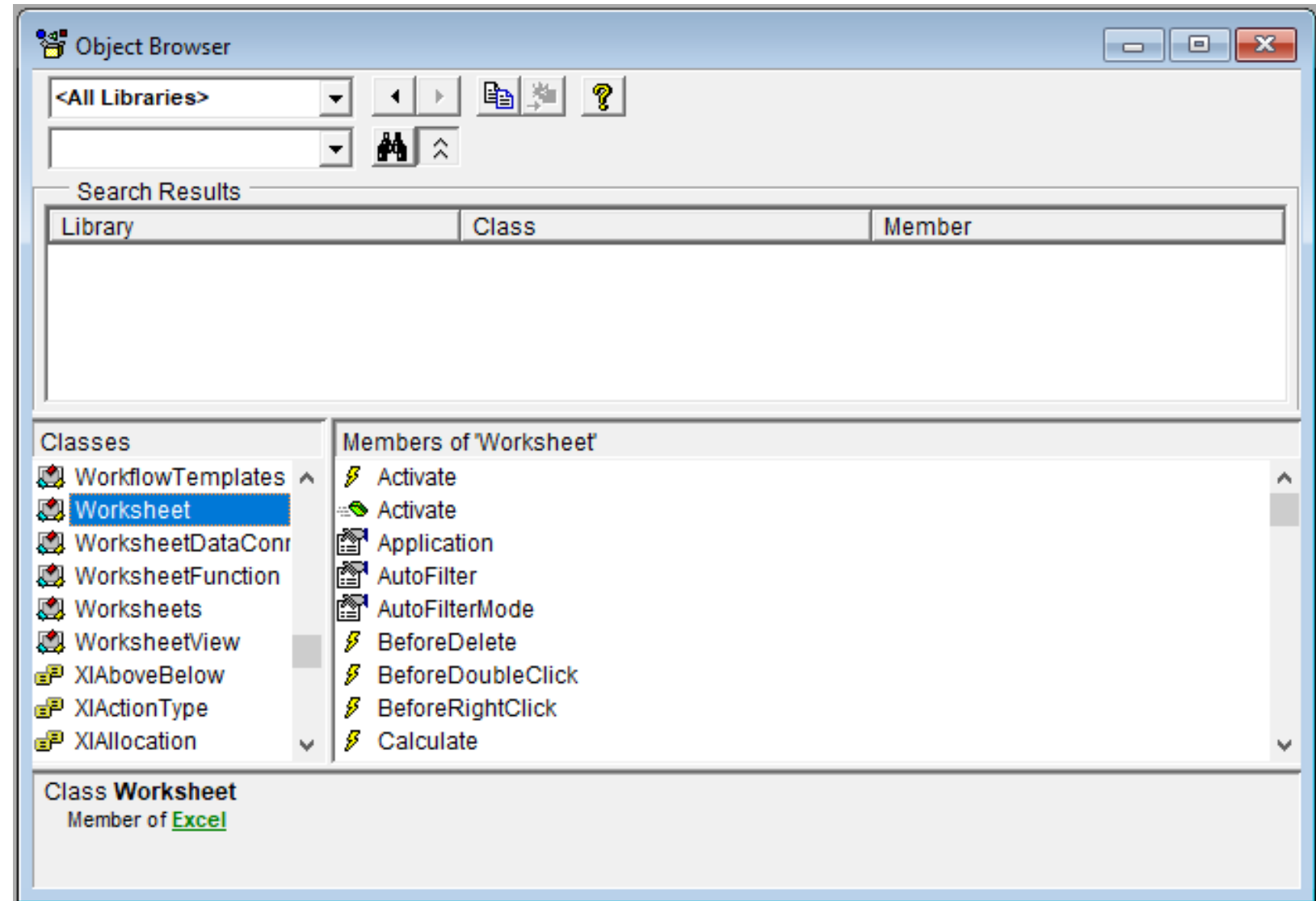


Colecciones

- Es un conjunto de objetos del mismo tipo
- WorkSheets es la colección de hojas de un libro de trabajo
- Una colección es un objeto por lo que se puede manipular la colección o cada uno de los objetos
- Los miembros se denominan elementos y se pueden referenciar por:
 - El nombre del objeto WorkSheets("Hoja1")
 - Índice WorkSheets(1)
- Colecciones de Application
 - AddIns
 - Dialogs
 - Windows
 - Workbooks

Examinador de objetos

- Elementos del examinador
 - Bibliotecas y proyectos
 - Búsqueda
 - Clases
 - Miembros
 - Propiedades
 - Métodos
 - Eventos
 - Plantilla de código



Asignación y propiedades

- Declaración
 - Dim nombreVariable as Object
 - Ej: Dim Hoja1 As Object
- Asignación
 - Set nombreVariable = nombreObjeto
 - Set Hoja1 = ActiveSheet
- Varias acciones sobre un objeto

With nombreObjeto
 sentencias (.propiedad = valor)
End With

```
With Hoja1.Range("A1:A1").Font  
    .Size = 24  
    .Bold = True  
End With
```

Objeto Application

- Propiedades:
 - Application.Caption = "Curso Hojas de Cálculo"
 - ActiveSheet.Range("A1") = Application.Path
 - ActiveSheet.Range("A2") = Application.UserName
 - ActiveSheet.Range("A3") = Application.Version
- Métodos:
 - Calculate
 - CheckSpelling
 - Wait(time)
 - SaveWorkspace([Filename])

Objeto Workbook

- Se puede acceder a un objeto Workbook:
 - Con la colección Workbooks: `Workbooks("Libro1")` ó `Workbooks(1)`
 - `ActiveWorkbook` (el libro activo)
 - `ThisWorkbook` (el libro en el que se ejecuta el código)
- Abrir un libro:
 - Con el método `Open` de la colección `Workbooks`:
`Workbooks.Open("nombreFichero.xls")`
 - Cambio de disco: `ChDrive "D"`
 - Cambio de directorio `ChDir "\Directorio"`
- Crear un libro
 - Con el método `Add` de la colección `Workbooks`:
`Workbooks.Open[("nombrePlantilla")]`

Objeto Workbook (2)

- Propiedades:
 - `ActiveSheet.Range("A11") = ActiveWorkbook.FullName`
 - `ActiveSheet.Range("A12") = ActiveWorkbook.Name`
 - `ActiveSheet.Range("A13") = ActiveWorkbook.Path`
 - `ActiveSheet.Range("A14") = ActiveWorkbook.Saved`
 - poniendo esta propiedad a True, se puede cerrar un libro sin salvar los cambios y sin que pregunte si se desean guardar
- Métodos:
 - `Activate`
 - `Close` (salvar Cambios, nombre fichero, ruta de envío)
 - `PrintOut` (desde, hasta, copias, preliminar, impresora, a fichero, intercalar)
 - `Save`
 - `SaveAs(nombre fichero)`

Objeto Worksheet

- La colección Worksheets contiene las hojas de un libro de trabajo
- Crear una hoja:
 - Worksheets.Add([Before], [After], [Count], [Type])
 - Ej: Worksheets.Add before:=Worksheets(2)
- Copiar: Copy([Before], [After])
- Propiedades:
 - Name
 - StandardHeight (altura de las filas)
 - StandardWidth (ancho de las columnas)
 - UsedRange
 - Visible
- Métodos
 - Activate
 - Copy([Before], [After])
 - Calculate
 - Move([Before], [After])
 - Select

Objeto Range

- El objeto Range puede ser:
 - Una celda
 - Una fila o columna
 - Una selección de celdas
 - Un rango 3D
- Obtención de un objeto Range
 - [ActiveSheet].Range(name)
 - name es un rango "A1", "A1:B2", "parametros"
 - [ActiveSheet].Range(cell1, cell2)
 - cell1 y cell2 son la superior izquierda y la inferior derecha y representan un rango de una celda o una fila o columna.
 - [ActiveSheet].Cells(rowindex, columnindex)
 - columnindex puede ser numérico o letra
 - se puede aplicar a la hoja o a un rango
 - ActiveSheet.[A1] = "Referencia entre corchetes"

```
For i = 1 To 5
    Cells(18, i) = "Campo " & i
Next i
```

Filas y columnas del objeto Range

- Para referirse a una fila o columna:
 - [ActiveSheet].Rows[(index)]
 - si se omite se devuelve una colección de filas
 - [ActiveSheet].Columns[(index)]
 - EntireRow y EntireColumn devuelven la/s fila/s o columna/s del rango
- Seleccionar un rango: método Select del objeto Range
 - Range("A1 ").Select
 - Es útil sólo en determinadas funciones (charts)
- El rango seleccionado se obtiene con Selection
 - Ej: Selection.Font.Size = 24

Propiedades y métodos del objeto Range

- Propiedades

- Column, Row: devuelve el número de la primera columna o fila del rango
- Count: número de celdas
- Formula, Value: establece u obtiene una fórmula o un valor
- FormulaArray: fórmula matricial
- NumberFormat: formato del rango
 - [A22].Value = [A22].NumberFormat

- Métodos

- Cut([Destination]), Copy([Destination])
- Clear, ClearContents, ClearFormats, ClearNotes
- DataSeries([Rowcol], [Type As xlDataSeriesType = xlDataSeriesLinear], [Date As xlDataSeriesDate = xlDay], [Step], [Stop], [Trend])
- Insert([Shift]), Shift puede ser xlToRight o xlDown
- Address(false, false), devuelve el "A1-style reference" del rango, ej: "A1:C4"

Hojas de cálculo

4. Interfaz de usuario

Nicolás Serrano
Organización Industrial

Contenido

- Mostrar información
- Obtener información
- Formularios de usuario
- Propiedades del formulario
- Controles del formulario
- Propiedades de controles
- Eventos
- Tipos de controles
- Abrir y cerrar un formulario
- Menús y barras de herramientas
- Añadir un botón o elemento de menú

Mostrar información

- Información sonora: Beep
 - Para que se produzca un intervalo:
 - `Application.Wait Now + TimeValue("horas:minutos:segundos")`
- Barra de estado
 - `Application.StatusBar = "cadena de texto"`
- Mensajes
 - `MsgBox(Prompt, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile], [Context]) As VbMsgBoxResult`
 - `VbMsgBoxStyle`:
 - `vbOKOnly` , `vbOKCancel` , `vbYesNo` , `vbYesNoCancel`
 - `vbExclamation` , `vbInformation` , `vbQuestion` , `vbCritical`
 - `VbMsgBoxResult`
 - `vbOK`, `vbCancel`, `vbAbort`, `vbRetry`, `vbIgnore`, `vbYes`, `vbNo`

Obtener información

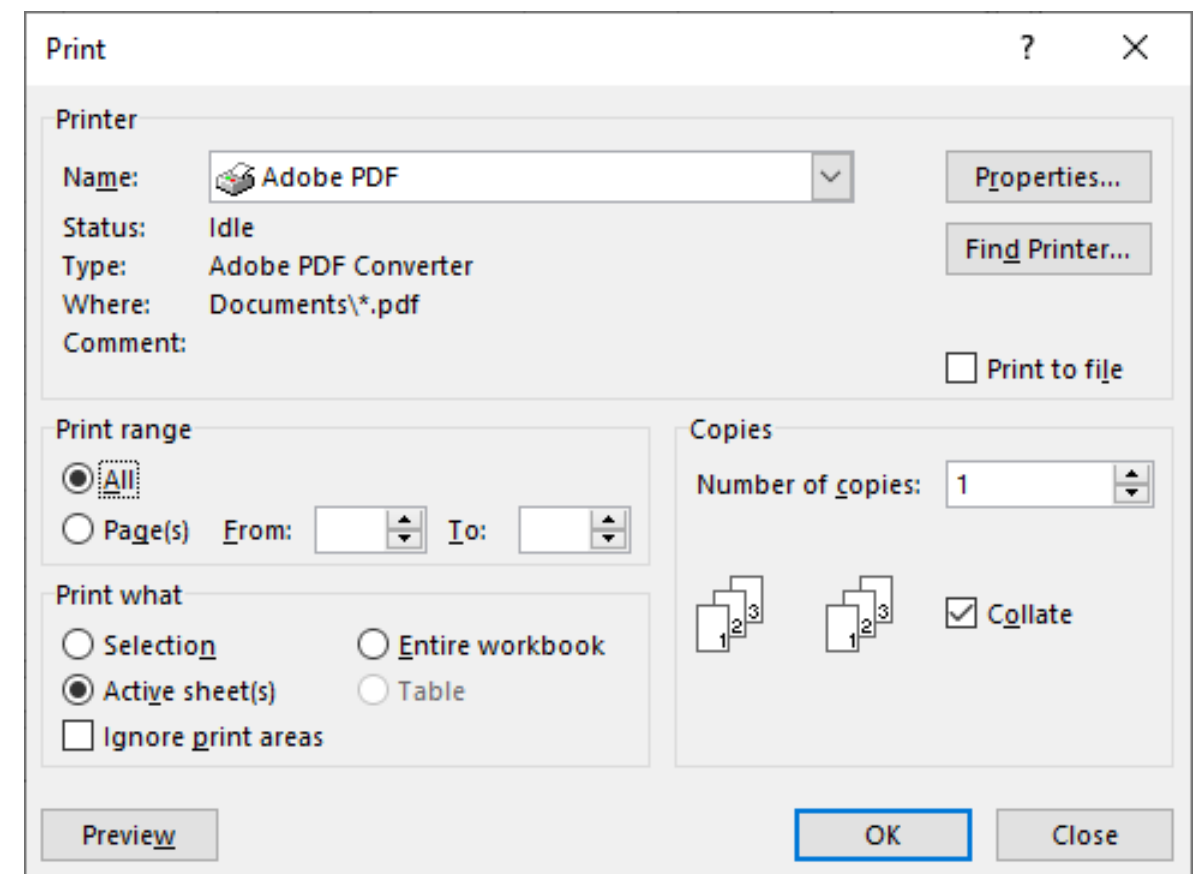
- `InputBox(Prompt As String, [Title], [Default], [Left], [Top], [HelpFile], [HelpContextID], [Type]) As String`

```
InputBox "Prueba de InputBox", "Curso  
Excel "
```



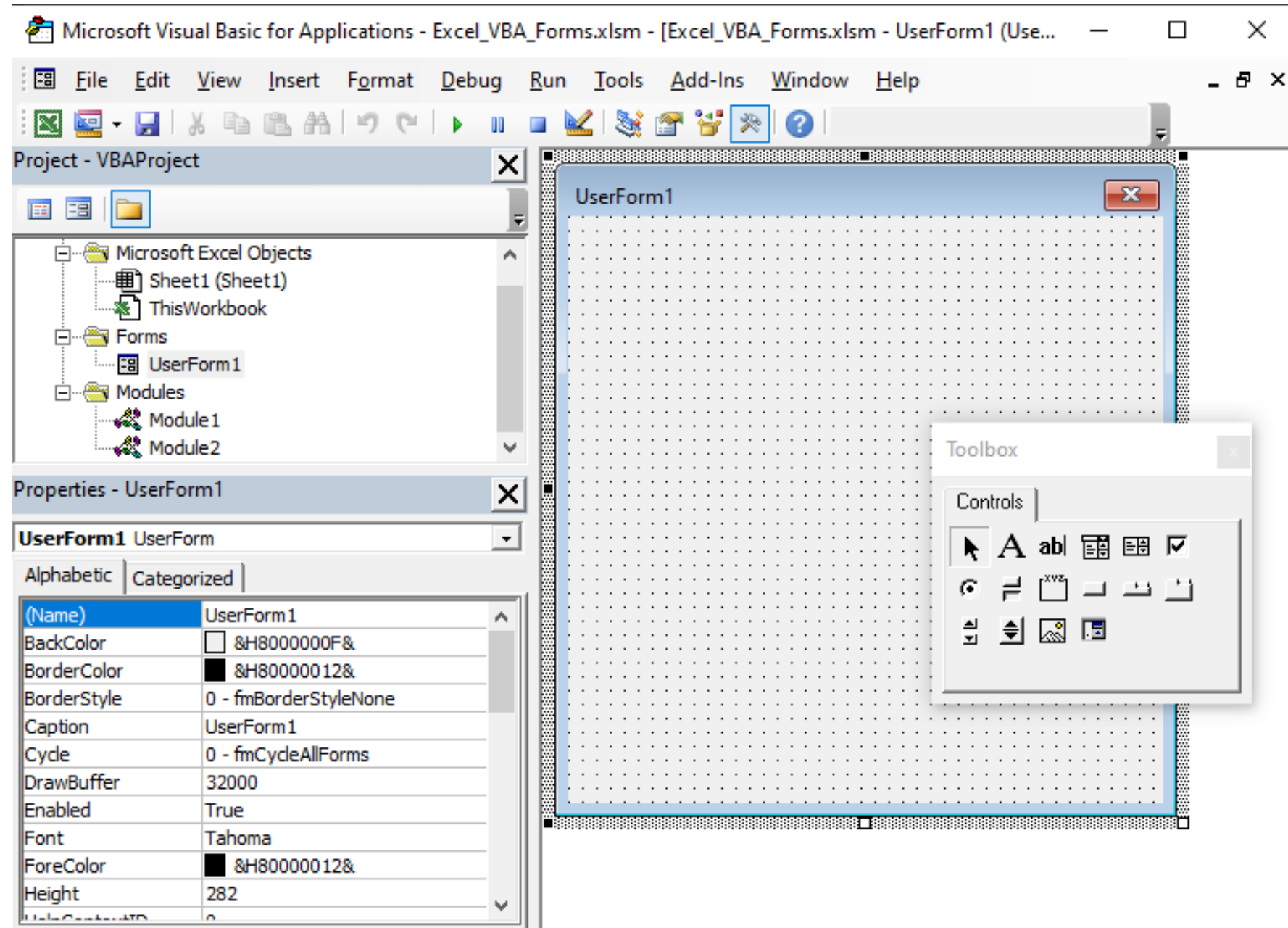
- Cuadros de diálogo de impresión:

```
Application.Dialogs(xlDialogPrint).Show
```



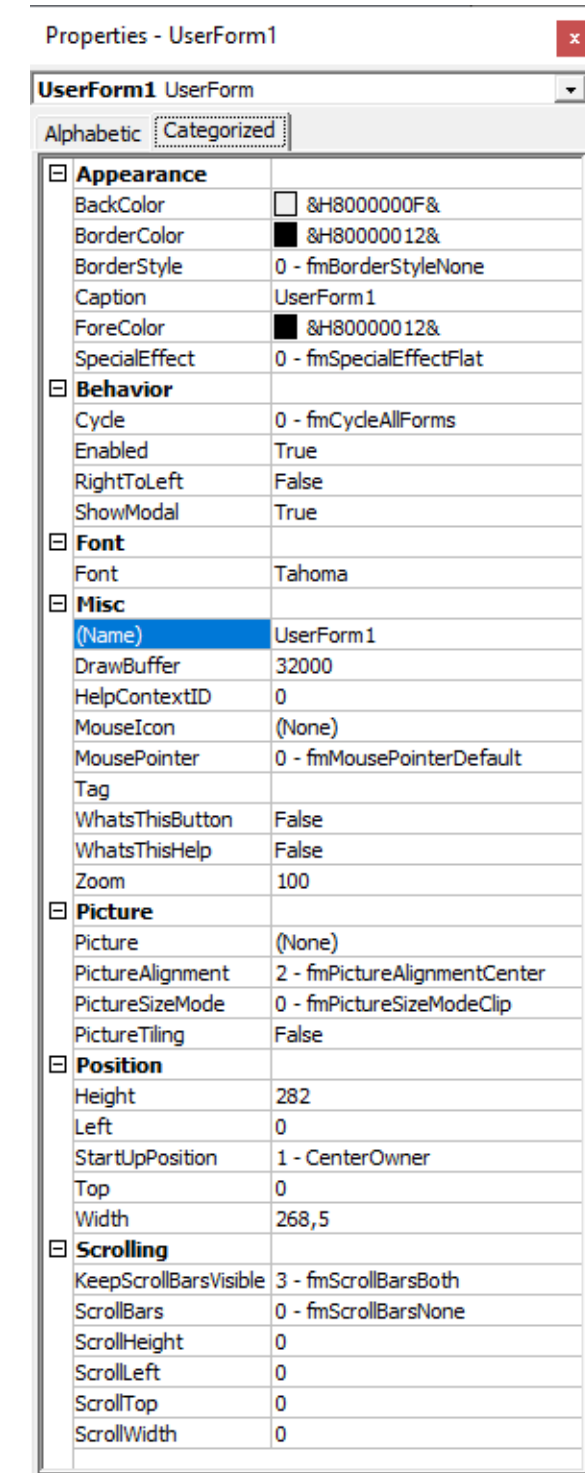
Formularios de usuario

- En el editor de Visual Basic:
 - > Insert / UserForm
- Se muestra
 - La carpeta de formularios
 - El objeto UserForm
 - El formulario
 - El cuadro de herramientas



Propiedades del formulario

- Se agrupan en 7 categorías
 - Apariencia: colores y título
 - Comportamiento: Enabled indica si está activo
 - Fuente
 - Varias: Nombre, puntero del ratón y ayuda
 - Imagen: imagen de fondo
 - Posición: posición y tamaño
 - Desplazamiento: barras de desplazamiento (scroll)

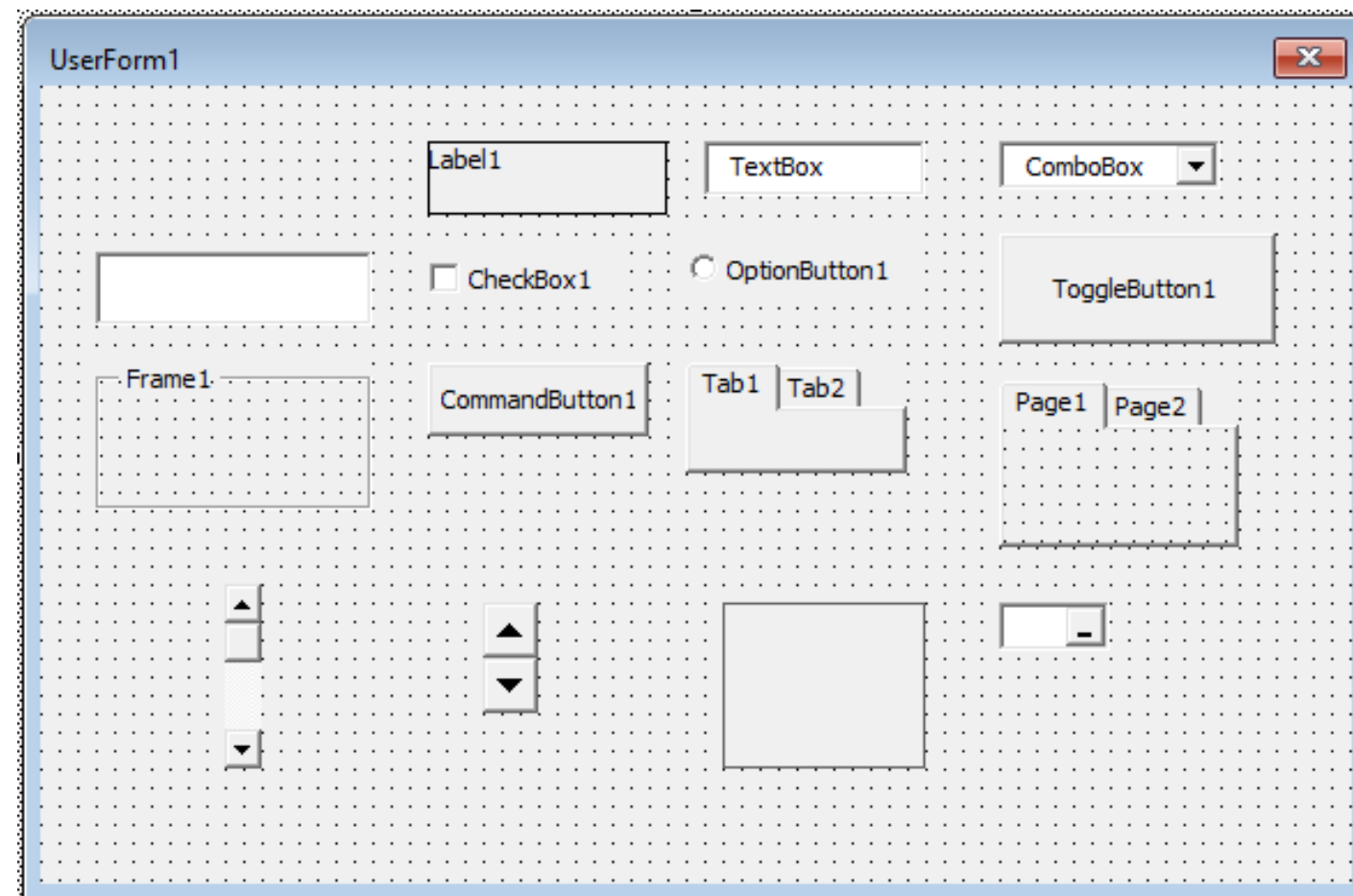
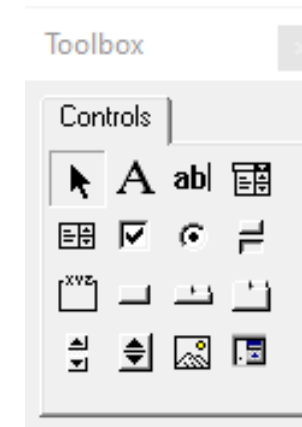


The screenshot shows the 'Properties - UserForm1' window in Visual Basic. The 'UserForm1 UserForm' dropdown is at the top. Below it, the 'Categorized' tab is selected. The properties are grouped into seven categories, each with a minus sign icon to its left:

- Appearance**
 - BackColor: &H8000000F&
 - BorderColor: &H80000012&
 - BorderStyle: 0 - fmBorderStyleNone
 - Caption: UserForm1
 - ForeColor: &H80000012&
 - SpecialEffect: 0 - fmSpecialEffectFlat
- Behavior**
 - Cycle: 0 - fmCycleAllForms
 - Enabled: True
 - RightToLeft: False
 - ShowModal: True
- Font**
 - Font: Tahoma
- Misc**
 - (Name): UserForm1
 - DrawBuffer: 32000
 - HelpContextID: 0
 - MouseIcon: (None)
 - MousePointer: 0 - fmMousePointerDefault
 - Tag:
 - WhatsThisButton: False
 - WhatsThisHelp: False
 - Zoom: 100
- Picture**
 - Picture: (None)
 - PictureAlignment: 2 - fmPictureAlignmentCenter
 - PictureSizeMode: 0 - fmPictureSizeModeClip
 - PictureTiling: False
- Position**
 - Height: 282
 - Left: 0
 - StartPosition: 1 - CenterOwner
 - Top: 0
 - Width: 268,5
- Scrolling**
 - KeepScrollBarsVisible: 3 - fmScrollBarsBoth
 - ScrollBars: 0 - fmScrollBarsNone
 - ScrollHeight: 0
 - ScrollLeft: 0
 - ScrollTop: 0
 - ScrollWidth: 0

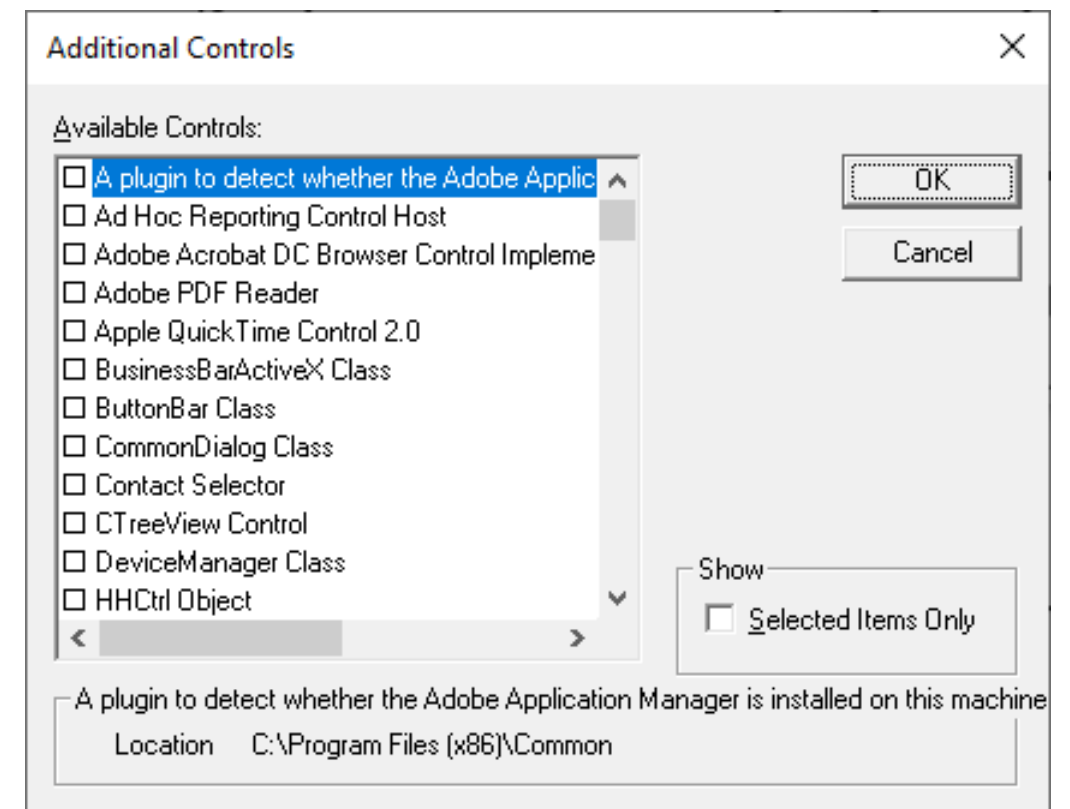
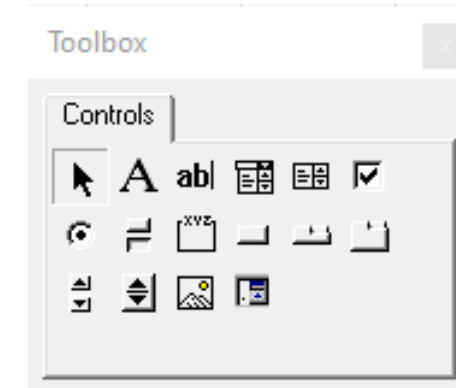
Tipos de controles

- CommandButton
- Label
- TextBox
- Frame
- OptionButton
- CheckBox
- ToggleButton
- ListBox
- ComboBox
- ScrollBar
- SpinButton
- ...



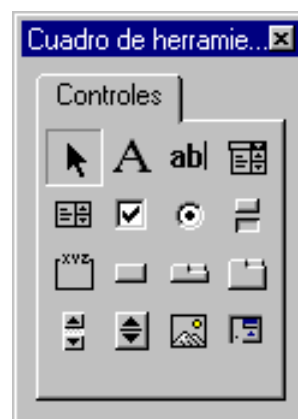
Controles del formulario

- Para añadir controles al formulario
 - Se selecciona en el Cuadro de herramientas
 - Se arrastra sobre el formulario
- Para añadir nuevos controles al cuadro de herramientas
 - Seleccionar un formulario
 - > Tools / Additional Controls
- Propiedades
 - Se pueden editar en el cuadro de propiedades
 - Se pueden modificar en ejecución

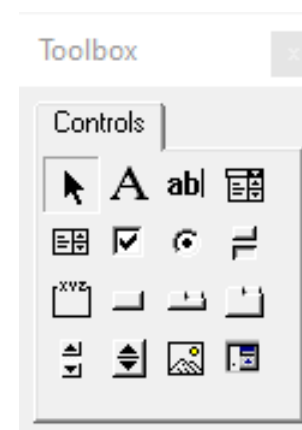


Evolución

Office 97

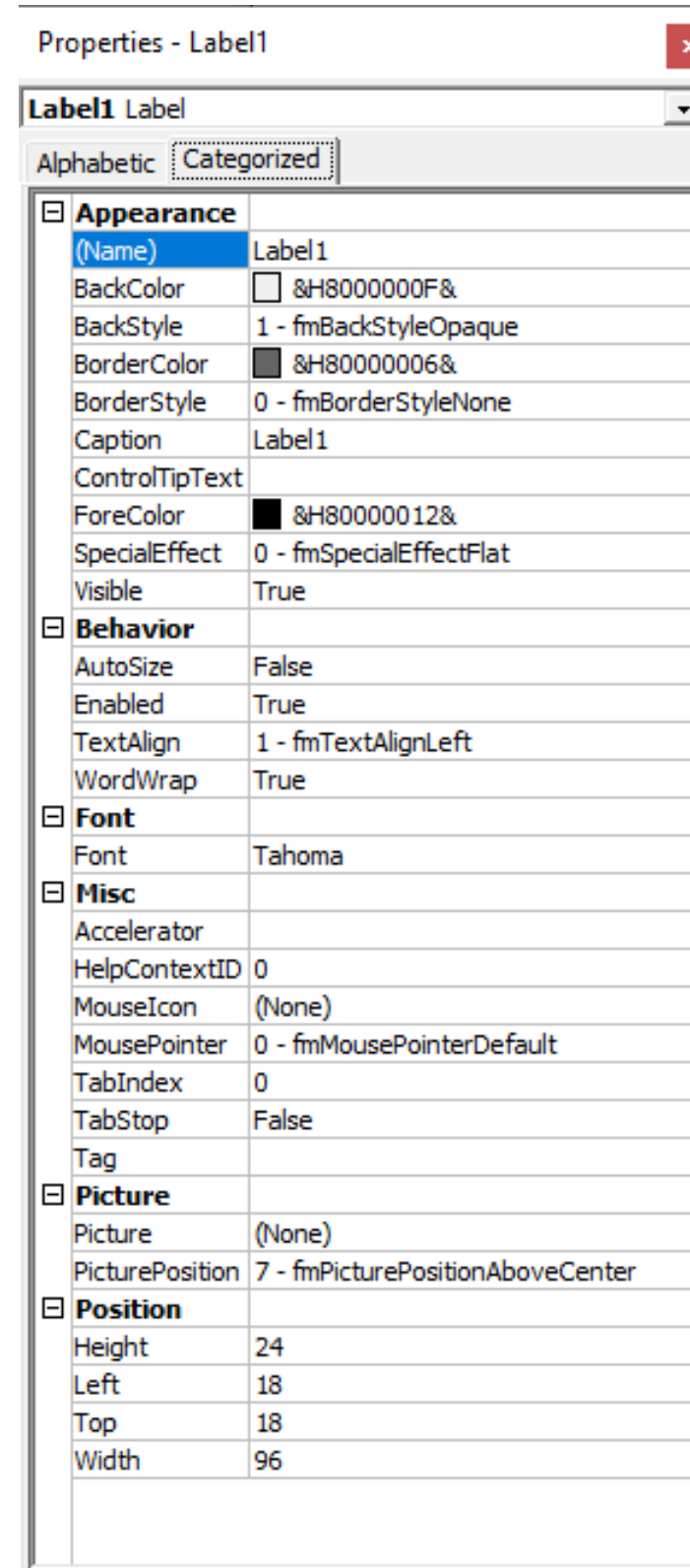


Office 2016



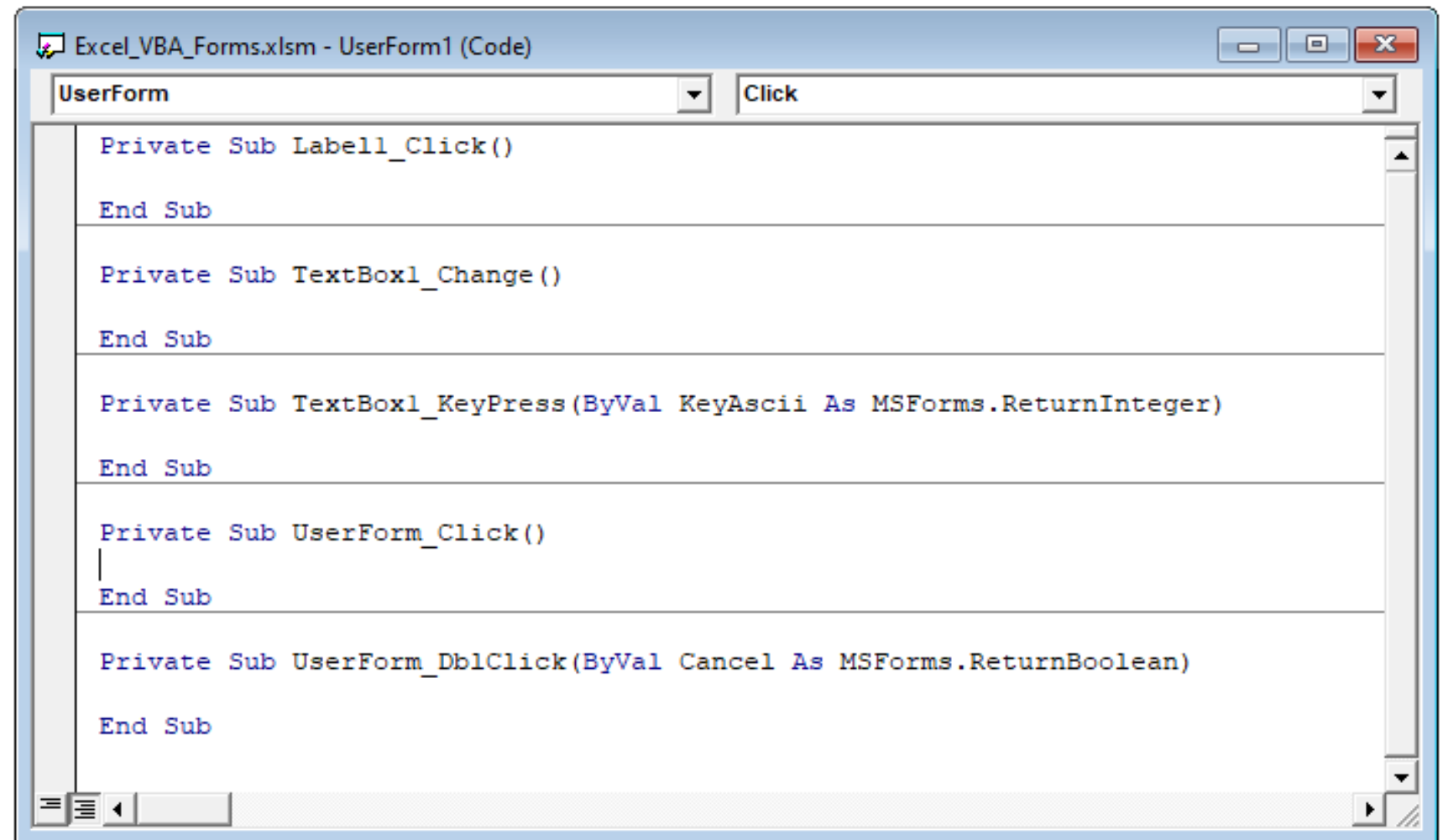
Propiedades de controles

- Accelerator: Alt + tecla
- AutoSize: tamaño función del texto
- BackColor: color de fondo
- Caption: texto del control
- ControlTipText: texto de ayuda
- Enabled: activo
- MousePointer: aspecto del puntero
- TabIndex: orden de tabulación
- TabStop: seleccionar el control mediante tab
- Tag: informativo
- Visible: indica si se muestra
- WordWrap: ruptura de palabras



Eventos

- Se pueden definir eventos para el formulario o sus controles
- Se introduce entre
 Sub
 y
 End Sub
de cada procedimiento
- El código se almacena en el módulo de código del UserForm



```
Excel_VBA_Forms.xlsm - UserForm1 (Code)
UserForm Click
Private Sub Label1_Click()
End Sub
Private Sub TextBox1_Change()
End Sub
Private Sub TextBox1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
End Sub
Private Sub UserForm_Click()
End Sub
Private Sub UserForm_DblClick(ByVal Cancel As MSForms.ReturnBoolean)
End Sub
```

Abrir y cerrar un formulario

- En tiempo de diseño: Run / Run Sub/UserForm o F5
- En código: UserForm1.Show
- Descargar el formulario: Unload Me
- Ejemplo de cierre de formulario con mensaje de verificación:

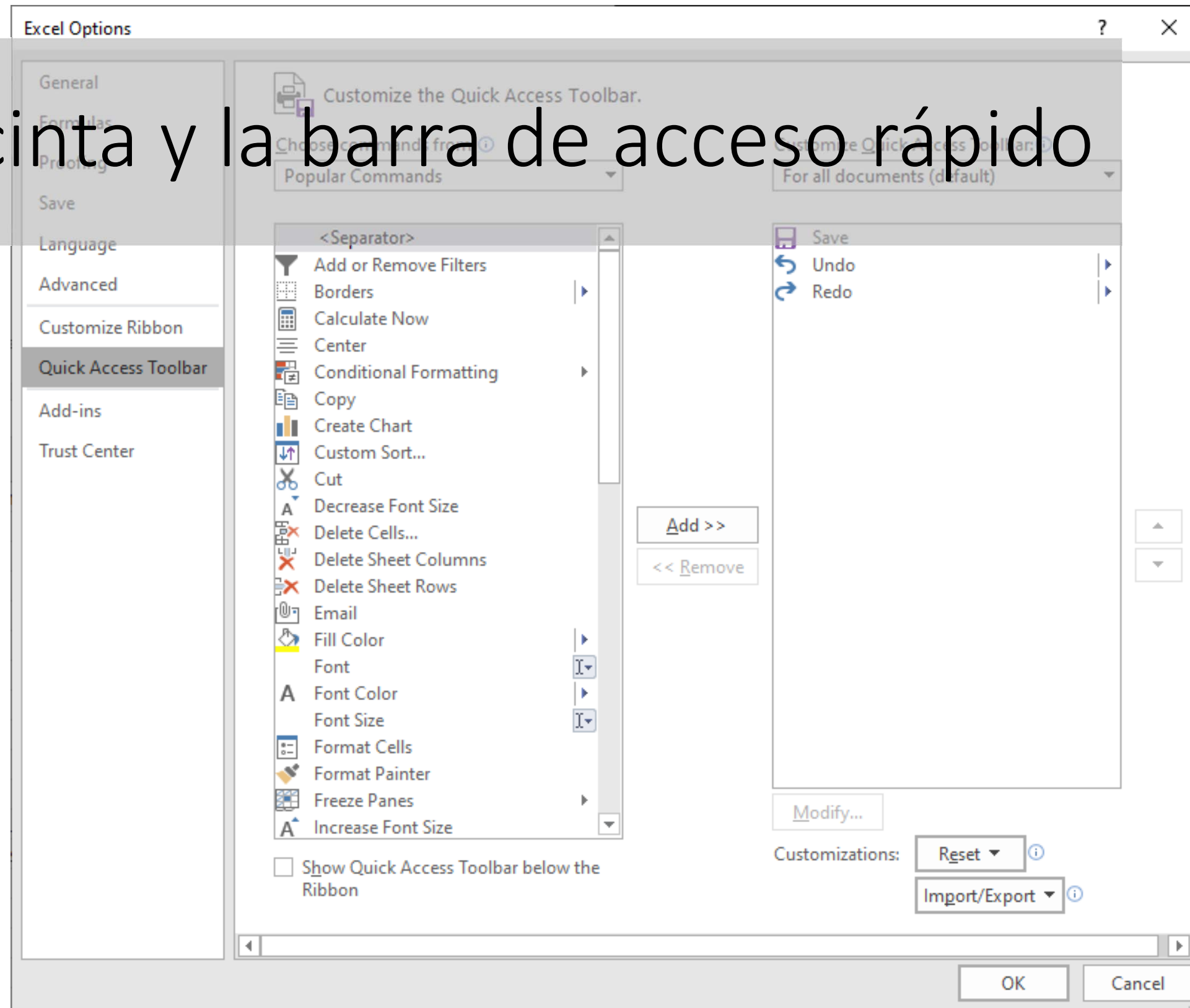
```
Private Sub CommandCancel_Click()
```

```
    If MsgBox("Desea cerrar", vbYesNo) = vbYes Then Unload Me
```

```
End Sub
```

Personalizar la cinta y la barra de acceso rápido

- Como al añadir la pestaña Developer:
Menú File -> Options
 - "Customize Ribbon"
 - "Quick Access Toolbar "



Hojas de cálculo

5. OLE y Acceso a bases de datos

Nicolás Serrano
Organización Industrial

Contenido

- OLE
 - Objeto Word
 - Objeto Outlook
 - Outlook - MailItem
 - Envío de mail
- Acceso a Bases de datos
 - DBEngine
 - Workspaces
 - Base de datos y recordsets
 - Ejemplo de OpenRecordset

Objeto Word

- Este ejemplo copia el rango A1:B20 desde la hoja 1 a un documento nuevo de Microsoft Word.

```
Sub Macro1()  
'  
' Macro1 Macro  
' Macro grabada el 18/06/1998 por Nicolas Serrano  
'  
' Acceso directo: CTRL+a  
Dim wd As Object  
    'Crea una sesión de Microsoft Word  
    Set wd = CreateObject("word.application")  
    'Copia el gráfico en la hoja Rótulos de gráficos  
    Worksheets(1).Range("A1:B20").Copy  
    'Hace visible el documento  
    wd.Visible = True
```

```
'Activa MS Word  
    'AppActivate wd.Name    No hay que  
        activarlo en Excel 2007  
    With wd  
        'Crea un documento nuevo en  
        Microsoft Word  
        .Documents.Add  
        'Inserta un párrafo  
        .Selection.TypeParagraph  
        'Pega el gráfico  
        .Selection.PasteSpecial  
        link:=True, DisplayAsIcon:=False,  
        Placement:=wdInLine  
    End With  
    Set wd = Nothing  
End Sub
```


Outlook

- Este ejemplo crea y agrega información en una tarea nueva de Outlook. Ejecute Outlook y haga clic en Tareas en la barra de Outlook para ver la nueva tarea.
- **NOTA:** La tarea puede demorar unos minutos en aparecer.

```
Sub MS_Outlook()  
Dim ol As Object, miElem As Object  
    'Crea una sesión de Microsoft Outlook  
    Set ol = CreateObject("outlook.application")  
    'Crea una tarea  
    Set miElem = ol.CreateItem(olTaskItem)  
    'Agrega información a la nueva tarea  
    With miElem  
        .Subject = "Nueva tarea de VBA"  
        .Body = "Esta tarea se creó mediante Automatización de Microsoft Excel"  
        .NoAging = True  
        .Close (olSave)  
    End With  
    'Quita el objeto de la memoria  
    Set ol = Nothing  
End Sub
```

Outlook - MailItem

- Se crea con:

```
Dim ol As Object, miElem As Object
'Crea una sesión de Microsoft Outlook
Set ol = CreateObject("outlook.application")
'Crea una tarea
Set miElem = ol.CreateItem(olMailItem)
```

- Class MailItem

- Property To As String
- Property Body As String
- Property Attachments As Attachments
- Property Subject As String
- Método: Sub Send()

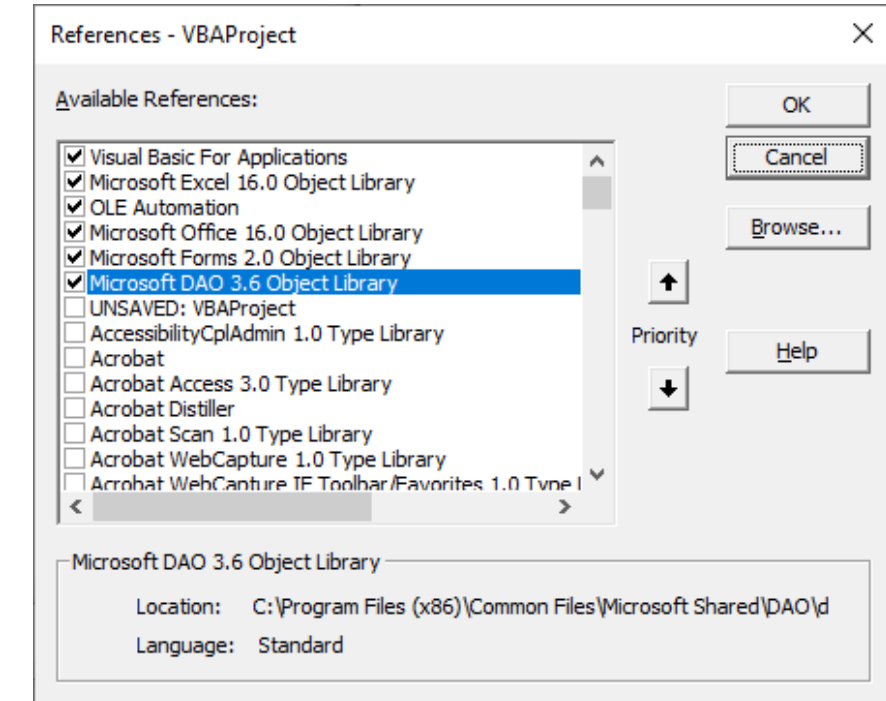
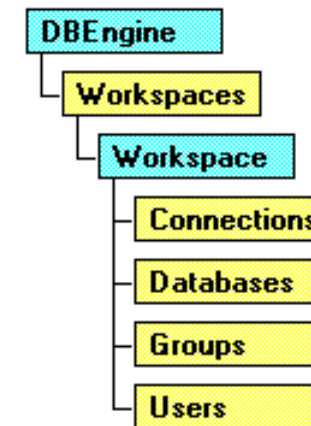
Envío de mail

```
Private Sub Comando0_Click()  
  
Dim ol As Object, miElem As Object  
    'Crea una sesión de Microsoft Outlook  
    Set ol = CreateObject("outlook.application")  
    'Crea el mensaje de mail  
    Set miElem = ol.CreateItem(olMailItem)  
    'Agrega información a la nueva tarea  
    With miElem  
        .Subject = "Prueba de mail de VBA"  
        .Body = "Esta tarea se creó mediante OLE de Microsoft Excel"  
        .To = "Nicolas Serrano"  
    End With  
    miElem.send  
    'Quita el objeto de la memoria  
    Set ol = Nothing  
  
End Sub
```

DBEngine

- El objeto DBEngine es el origen en el modelo de objeto DAO. Existe un solo DBEngine y no es elemento de una colección.
- Activar: Tools -> References: Microsoft DAO 3.6 Object Library

```
Sub DBEngineProperties()  
  
    Dim wrkBucle As Workspace  
    Dim prpBucle As Property  
  
    With DBEngine  
        Debug.Print "Propiedades de DBEngine"  
  
        ' Enumera la colección Properties de DBEngine,  
        ' para interceptar propiedades con valores no  
        ' válidos en este contexto.  
        For Each prpBucle In .Properties  
            On Error Resume Next  
            Debug.Print " " & prpBucle.Name & " = " _  
                & prpBucle  
            On Error GoTo 0  
        Next prpBucle  
  
    End With  
  
End Sub
```



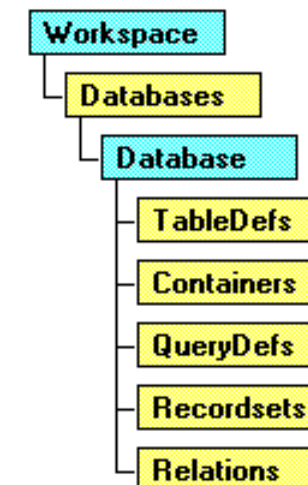
Workspaces

- La colección Workspaces contiene todos los objetos Workspace. Un objeto Workspace define una sesión para un usuario. Contiene las bases de datos abiertas y proporciona mecanismos para realizar transacciones

```
Sub WorkspacesColection()  
  
    ' Enumera la colección Workspaces de DBEngine.  
    Debug.Print "Colección Workspaces de DBEngine"  
    For Each wrkBucle In DBEngine.Workspaces  
        Debug.Print "      " & wrkBucle.Name  
        ' Enumera la colección Properties de cada  
        ' objeto Workspace, para interceptar  
        ' propiedades con valores no válidos en este contexto.  
        For Each prpBucle In wrkBucle.Properties  
            On Error Resume Next  
            Debug.Print "          " & prpBucle.Name & _  
                " = " & prpBucle  
            On Error GoTo 0  
        Next prpBucle  
    Next wrkBucle  
  
End Sub
```

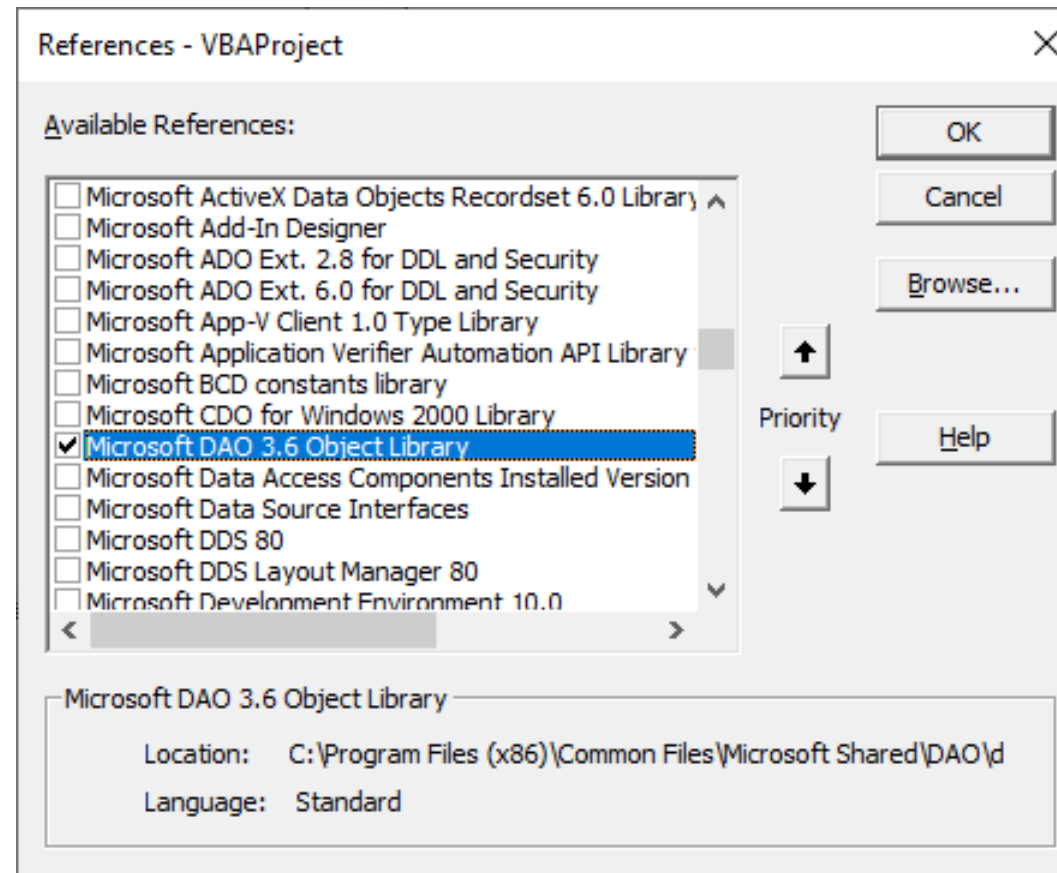
Base de datos y recordsets

- Creación de objeto Base de datos (Database)
- Set basededatos = [espaciode trabajo.]OpenDatabase (nombrebasededatos, opciones, sólolectura, conexión)
 - Nombre base de datos: Es un String que representa el nombre de un archivo de base de datos Microsoft Jet existente o el nombre del origen de datos (DSN) de un origen de datos ODBC existente.
- Para crear un nuevo objeto Recordset y añadirlo a la colección Recordsets:
- Set variable = objeto.OpenRecordset (origen, tipo, opciones, bloquear modificaciones)
 - El origen es un String que especifica el origen de los registros para el nuevo Recordset. El origen puede ser un nombre de tabla, un nombre de consulta o una instrucción SQL que devuelve registros.



Reference

- Tools / References ...
 - Select the DAO object:



Ejemplo de OpenRecordset

```
Sub RecordsetExcel()  
Dim bd As Database  
Dim rs As Recordset  
Dim HojaNueva As Object  
Dim Neptuno As String  
    'Ruta predeterminada a la base de datos de ejemplo Neptuno.mdb  
    Neptuno = "C:\Nicolas\Excel\biblio.mdb"  
    'Abre la base de datos Neptuno.mdb  
    Set bd = DBEngine.Workspaces(0).OpenDatabase(Neptuno)  
    'Abre un conjunto de registros con todos los registros de la tabla clientes  
    Set rs = bd.OpenRecordset("titles")  
    'Inserta una nueva hoja de cálculo en el libro activo  
    Set HojaNueva = ThisWorkbook.Sheets.Add(Type:=xlWorksheet)  
    'Sitúa los nombres de campo en la fila 1 de la nueva hoja de cálculo  
    For h = 0 To rs.Fields.Count - 1  
        HojaNueva.[a1].Offset(0, h).Value = rs.Fields(h).Name  
    Next h  
    'Copia el conjunto de registros en Excel  
    HojaNueva.[a2].CopyFromRecordset rs  
    'Cierra el conjunto de registros  
    rs.Close  
    'Cierra la base de datos  
    bd.Close  
End Sub
```


Recorrer un Recordset

- La copia de los valores en lugar de realizarse con la sentencia:

```
HojaNueva.[a2].CopyFromRecordset rs
```

- Se puede realizar recorriendo los registros del recordet y los campos de cada registro:

```
rs.MoveNext
j = 1
Do While (Not rs.EOF)
    For h = 0 To rs.Fields.Count - 1
        HojaNueva.[a1].Offset(j, h).Value = rs.Fields(h).Value
    Next h
    j = j + 1
    rs.MoveNext
Loop
```

XMLHttpRequest

```
Sub GetTedScript(id As Integer, lang As String, col As Integer)

    Dim i As Integer
    Dim URLstr As String, sHTML As String, sAllPosts As String
    Dim Http As Object
    Dim start As Long, endInt As Long
    Dim blWSExists As Boolean

    Dim fileStr As String
    fileStr = id
    'Create a new Worksheet "id-lang" if it doesnt'exist already.
    For i = 1 To Worksheets.Count
        If Worksheets(i).Name = fileStr Then
            blWSExists = True
            Worksheets(i).Activate
        End If
    Next
    If Not blWSExists Then
        Worksheets.Add.Move after:=Worksheets(Worksheets.Count)
        Worksheets(Worksheets.Count).Name = fileStr
    End If
```

Crear hoja

XMLHttpRequest

Nueva URL (2019):

<https://www.ted.com/talks/subtitles/id/70/lang/es/format/text>

```
'URL to open: http://www.ted.com/talks/subtitles/id/70/lang/eng
URLstr = "http://www.ted.com/talks/subtitles/id/" & id & "/lang/" & lang

' Create an XMLHttpRequest object and add some error trapping
On Error Resume Next
Set Http = CreateObject("MSXML2.XMLHTTP")
If Err.Number <> 0 Then
    Set Http = CreateObject("MSXML.XMLHttpRequest")
    MsgBox "Error 0 has occurred while creating a MSXML.XMLHttpRequest object"
End If
On Error GoTo 0
If Http Is Nothing Then
    MsgBox "For some reason I wasn't able to make a MSXML2.XMLHTTP object"
    Exit Sub
End If

'Open the URL in browser object
Http.Open "GET", URLstr, False
Http.Send
sHTML = Http.responseText
```

Leer URL

XMLHttpRequest

```
'Now extract all text within "content": and startTime
'because they represent the topics
i = 1
start = 1
endInt = 1
Do While start <> 0
    i = i + 1
    start = InStr(endInt, sHTML, ""content"":", vbTextCompare)
    If start <> 0 Then
        start = start + 11
        endInt = InStr(start, sHTML, ""startTime"":", vbTextCompare) - 2
        phraseStr = Mid(sHTML, start, endInt - start)
        phraseStr = Replace(phraseStr, "\"", "")
        Worksheets(fileStr).Range("A2").Offset(i, col).Value = _
            phraseStr
    End If
Loop

'Clean up
Set Http = Nothing
```

End Sub

Procesar texto