Macros personalizadas en Excel

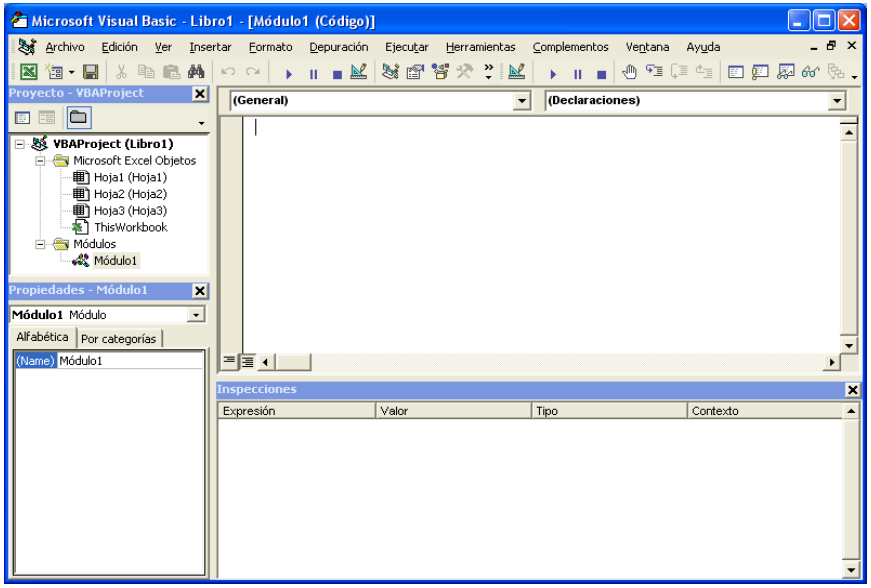
Visual Basic para Aplicaciones

V B A

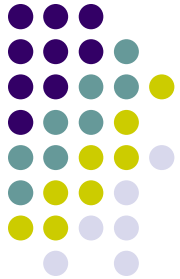
mi primera Macro

- Editor de Visual Basic
  - Alt+F11
  - Barra de Herramientas: Visual Basic
  - Herramientas, Macro, Editor de V.B.
- Insertar Módulo
- Primera macro

```
Sub Hola_Mundo()
    ActiveCell.Value = "Hola Mundo"
End Sub
```



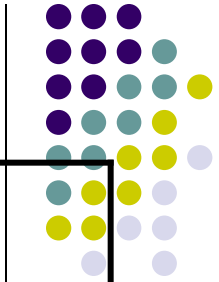
Objeto Rango



- Programa que deja un valor en una celda y modifica su formato

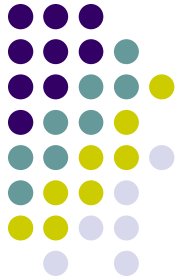
```
Sub Saludo()
    Worksheets("Hoja2").Activate
    ActiveSheet.Range("C5").Value = "¿Cómo esta usted?"
    ActiveSheet.Range("C5").Font.Bold = True
    ActiveSheet.Range("C5").Font.Color = RGB(255, 0, 0)
End Sub
```

Range y Offset



```
Sub primero()
    'Queremos asignar un valor al objeto Range
    Range("B10").Value = "Hola"
    ' Otra forma de trabajar es poniendo el objeto WorkSheets que está por encima de Range
    Worksheets(1).Range("B11").Value = "¿Qué tal?"
    ' Y aún podemos poner los objetos superiores
    ' Application que hace referencia a la aplicación Excel
    ' Y WorkBooks que se refiere al libro de trabajo
    Application.Workbooks(1).Worksheets(1).Range("B12").Value = "Felicidades"
    Application.Workbooks("Mac01.xls").Worksheets("Hoja1").Range("B13").Value = "América"
    ' Application normalmente no se pone porque todo cuelga de Excel
    ' WorkBooks conviene ponerlo cuando se trabaja con varios libros
    ' WorkSheet conviene si se trabaja con varias hojas, aunque muchas veces no se pone
    Range("B14").Value = 8.6 'Los decimales con punto
    Range("B15").Select
    Application.ActiveWindow.ActiveCell.Value = "Adios"
    ' Señale con el ratón ActiveWindow y pulse F1 que es la ayuda
    ActiveCell.Offset(1, 0).Value = "Bye"
    ActiveCell.Offset(2, 0).Activate
    ActiveCell.Value = "Hasta la vista"
    ActiveSheet.Range("A1").Offset(17, 1).Value = "100%"
End Sub
```

La Estructura With - End With



- Sirve para ejecutar una serie de acciones sobre un mismo Objeto, **sin tener que repetir toda su jerarquía**
  - Ej.: Propiedades del objeto Range

```
Sub Escribe()
    ActiveSheet.Range("C7").Value = "Cta. Resultados"
    ActiveSheet.Range("C7").Font.Bold = True
    ActiveSheet.Range("C7").Font.Color = RGB(0, 255, 0)
End Sub
```

```
Sub Escribe_bis()
    With ActiveSheet.Range("C7")
        .Value = "Cta. Resultados"
        .Font.Bold = True
        .Font.Color = RGB(0, 255, 0)
    End With
End Sub
```

Dim e InputBox

- Option Explicit sirve para que nos obliguemos ha definir todas las variables
- Dim permite definir el tipo de variable
- Si no se definen las variables se toman como VARIAN que son las que más ocupan
- InputBox permite capturar datos del usuario
- InputBox devuelve siempre datos tipo String
- Chr(13) es para cambiar de línea

```
Option Explicit

Sub Entrar_Valor()
    Dim Texto As String
    Texto = InputBox("Introducir un texto" & Chr(13) & "Para la Casilla D10", "Entrada de Datos")
    ActiveSheet.Range("D10").Value = Texto
End Sub
```

Dim e InputBox

InputBox(*Mensaje*, *Título*, *Valor por defecto*, *Posición horizontal*, *Posición Vertical*, *Archivoayuda*, *Número de contexto para la ayuda*)

```
Sub Entrar_Valor_Bis()
    'Este procedimiento es igual que el anterior pero no utiliza variables
    ActiveSheet.Range("D11").Value _
    = InputBox("Introducir un texto " & Chr(10) & "Para la casilla D11", "Entrada de datos")
    'El guión bajo permite partir una línea de código demasiado larga. Ver Chr(10)
End Sub
```

```
Sub Entrar_Valor_Tris()
    'En este caso se pide al usuario que entre la casilla donde se introducirá el texto
    Dim Casilla As String 'Casilla puede ser por ejemplo D12
    Dim Texto As String
    Casilla = InputBox("En que casilla quiere entrar el valor", "Entrar Casilla")
    Texto = InputBox("Introducir un texto" & Chr(13) _
    & "Para la casilla " & Casilla, "Entrada de datos") ‘ Operador de concatenación &
    ActiveSheet.Range(Casilla).Value = Texto
End Sub
```

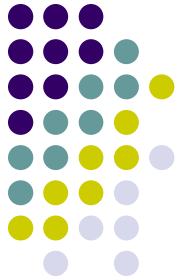
Tipos de variables

Tipo de datos	Tamaño de almacenamiento	Intervalo
Byte	1 byte	0 a 255
Boolean	2 bytes	True o False
Integer	2 bytes	-32.768 a 32.767
Long (entero largo)	4 bytes	-2.147.483.648 a 2.147.483.647
Single (coma flotante/precisión simple)	4 bytes	-3,402823E38 a -1,401298E-45 para valores negativos; 1,401298E-45 a 3,402823E38 para valores positivos
Double (coma flotante/precisión doble)	8 bytes	-1,79769313486232E308 a -4,94065645841247E-324 para valores negativos; 4,94065645841247E-324 a 1,79769313486232E308 para valores positivos
Currency (entero a escala)	8 bytes	-922.337.203.685.477,5808 a 922.337.203.685.477,5807
Decimal	14 bytes	+/-79.228.162.514.264.337.593.543.950.335 sin punto decimal; +/-7,9228162514264337593543950335 con 28 posiciones a la derecha del signo decimal; el número más pequeño distinto de cero es +/-0,0000000000000000000000000001
Date	8 bytes	1 de enero de 100 a 31 de diciembre de 9999
Object	4 bytes	Cualquier referencia a tipo Object
String (longitud variable)	10 bytes + longitud de la cadena	Desde 0 a 2.000 millones
String (longitud fija)	Longitud de la cadena	Desde 1 a 65.400 aproximadamente
Variant (con números)	16 bytes	Cualquier valor numérico hasta el intervalo de un tipo Double
Variant (con caracteres)	22 bytes + longitud de cadena	El mismo intervalo que para un tipo String de longitud variable
Definido por el usuario (utilizando Type)	Número requerido por los elementos	El intervalo de cada elemento es el mismo que el intervalo de su tipo de datos.

Ejercicio 1

- Crear un libro llamado “Rellena.xls”
- Programar un procedimiento que nos pregunte en que hoja queremos situarnos
- Nos pregunte en que celda queremos situarnos
- Nos pregunte lo que queremos escribir
- El programa pone lo que hemos dicho y lo pone de color verde y cursiva, sobre fondo rojo
  - Pista: ActiveCell.Interior.Color=RGB(x,y,z)
- Ejecute el programa
  - Primero, dando una sola celda
  - Segundo. Cuando pida la celda introduzca un rango para ver como funciona Range
- ¿Ha usado With – End With?

Suma dos números



- ***Val***(Cadena). Convierte la cadena a un valor numérico
- ***Str***(Número). Convierte el número a una expresión cadena
  - **CBool**, **CByte**, **CCur**, **CCur**, **CDate**, **CDec**, **CInt, CLng, CSng, CStr, CVar**

```
Sub Sumar()
   'Pide dos números y pone en una celda su suma
   'Dim Numero1 As Integer
   'Dim Numero2 As Integer
   Numero1 = InputBox("Entrar el primer valor", "Entrada de datos")
   Numero2 = InputBox("Entrar el segundo valor", "Entrada de datos")
   Worksheets("Hoja1").Activate 'Esto se pone por si estamos en una hoja distinta de la Hoja1
   ActiveSheet.Range("E10").Value = Numero1 + Numero2
End Sub
```

```
Sub Sumar_Bis()
   'Este procedimiento es similar al anterior
   'En el procedimiento anterior si se mete como variable una palabra, da error.
   'Por eso en este procedimiento añadimos la función Val
   Dim Numero1 As Integer
   Dim Numero2 As Integer
   Numero1 = Val(InputBox("Entrar el primer valor", "Entrada de datos"))
   Numero2 = Val(InputBox("Entrar el segundo valor", "Entrada de datos"))
   ActiveSheet.Range("E11").Value = Numero1 + Numero2
End Sub
```

Ejercicio 2

- El siguiente programa no funciona bien
- El área del un rectángulo de base 4,5 y altura 5,5 es 24,75
- Pero este programa da 24. El problema es que no da ERROR
- Modifique el código del procedimiento para solucionarlo

```
Sub area()
    Dim base As Integer
    Dim altura As Integer
    Dim superficie As Integer
    'Los decimales se introducen con coma en un inputbox, y con punto en el código
    altura = InputBox("Introduzca la altura del rectángulo")
    base = InputBox("Introduzca la base del rectángulo")
    superficie = base * altura
    MsgBox ("El área del rectángulo es " & superficie)
End Sub
```

Public – Private. Cells

- Public. Indica que el procedimiento **Sub** es accesible para todos los demás procedimientos de todos los módulos
- Private. Indica que el procedimiento **Sub** es accesible sólo para otros procedimientos del módulo en el que se declara
- Por defecto los procedimientos son Public
- Cells comienza a contar filas y columnas a partir del rango especificado en el objeto Range

Cells(fila,columna)

```
Private Sub Celda()
    Cells(12, 3).Value = "Solo " & 2
    ActiveSheet.Cells(10, 6).Value = "Paris"
    'La Celda 10,6 es la F10
    Range("C13:D14").Value = "Cuadrado"
    Range(Cells(15, 3), Cells(16, 4)).Value = "Cubo"
    Range("C17:F20").Cells(2, 1).Value = "Elipse" 'Esto solo pone una elipse
End Sub
```

Variables de objetos

- Una variable objeto sirve para hacer referencia a un objeto, esto significa que podremos acceder a las propiedades de un objeto e invocar sus métodos a través de la variable en lugar de hacerlo directamente a través del objeto.
- Para declarar una variable objeto se utiliza también la palabra **Dim**

  Dim Var_Objeto **As** **Objeto**
- Por Ejemplo

  Dim R **As** **Range**

  Dim Hoja **As** **WorkSheet**
- Para asignar un objeto a una variable debe utilizar la instrucción **Set**.

  Set Variable_Objeto = Objeto
- Por Ejemplo

  Set R= ActiveSheet.Range("A1:B10")

  Set Hoja = ActiveSheet

  Set Hoja = WorkSheets(1)
- A veces pude ser interesante desasignar una variable objeto

  Dim Var_Objeto = Nothing

Variables de objetos

- Posiblemente no se utilice demasiado esta clase de variables (dependerá de las preferencias del programador), pero hay casos en los que no hay más remedio que utilizarlas, por ejemplo en estructuras **For Each** ... **Next** como veremos, o cuando sea necesario construir funciones que devuelvan rangos, referencias a hojas, etc.

```
Sub objeto()
   Dim R As Range
   Set R = ActiveSheet.Range("H21:I22")
   R.Value = "Roma"
   R.Font.Bold = True
   R.Font.Color = RGB(0, 255, 100)
End Sub
```

Estructuras Condicionales

```
If Condición Then
   Senténcia1
Senténcia2
      .
      .
SenténciaN
End If
```

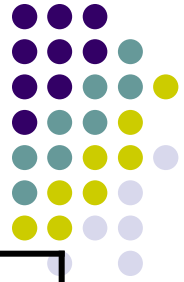
```
Sub Condicional()
   ActiveSheet.Range("E14").Value = 0
   ActiveSheet.Range("E15").Value = 0
   ActiveSheet.Range("E16").Value = 0
   ActiveSheet.Range("E14").Value = Val(InputBox("Entrar el precio", "Entrar"))
   'Si el valor de la casilla E14 es mayor que 1000, entonces pedir descuento
   If ActiveSheet.Range("E14").Value > 1000 Then
      ActiveSheet.Range("E15").Value = Val(InputBox("Entrar Descuento", "Entrar"))
   End If
   ActiveSheet.Range("E16").Value = _
   ActiveSheet.Range("E14").Value - ActiveSheet.Range("E15")
End Sub
```

Estructuras Condicionales

```
Sub Condicional_Bis()
   'Igual que el procedimiento anterior pero ahora usando variables
   Dim Precio As Integer
   Dim Descuento As Integer
   Precio = 0
   Descuento = 0
   Precio = Val(InputBox("Entrar el precio", "Entrar"))
   ' Si el valor de la variable precio es mayor que 1000, entonces, pedir descuento
   If Precio > 1000 Then
      Descuento = Val(InputBox("Entrar descuento", "Entrar"))
   End If
   ActiveSheet.Range("F14").Value = Precio
   ActiveSheet.Range("F15").Value = Descuento
   ActiveSheet.Range("F16").Value = Precio - Descuento
End Sub
```

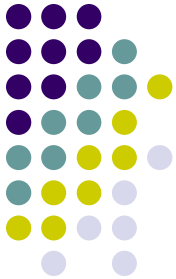
```
Sub Condicional2()
   If ActiveSheet.Range("F14").Value = ActiveSheet.Range("F16").Value Then
      ActiveSheet.Range("F14").Font.Color = RGB(0, 0, 255)
      ActiveSheet.Range("F16").Font.Color = RGB(0, 0, 255)
   End If
End Sub
```

Estructuras Condicionales. Else



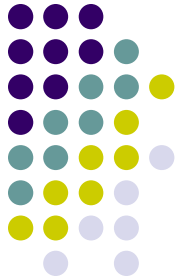
```
Sub Condicional_Else()
    Dim Precio As Single
    Dim Descuento As Single
    Precio = 0
    Descuento = 0
    Precio = Val(InputBox("Entrar el precio", "Entrar"))
    ' Si el valor de la variable precio es mayor que 1000, entonces, aplicar descuento del 10%
    If Precio > 1000 Then
        Descuento = Precio * (10 / 100)
        ActiveSheet.Range("G13").Value = 0.1
    Else ' Sino Aplicar descuento del 5%
        Descuento = Precio * (5 / 100)
        ActiveSheet.Range("G13").Value = 0.05
    End If
    ActiveSheet.Range("G14").Value = Precio
    ActiveSheet.Range("G15").Value = Descuento
    ActiveSheet.Range("G16").Value = Precio - Descuento
End Sub
```

Estructuras Condicionales. Else



```
Sub Condicional_Else2()
    'Ponga valores en G10 y en G11.
    'La macro calcula la diferencia la pone en G12 y asigna color
    ActiveSheet.Range("G12").Value = ActiveSheet.Range("G10").Value -
ActiveSheet.Range("G11").Value
    If Range("G12").Value < 0 Then
        'Si la diferencia es negativa pone color rojo
        ActiveSheet.Range("G12").Font.Color = RGB(255, 0, 0)
    Else
         'En caso contrario pone color azul
        ActiveSheet.Range("G12").Font.Color = RGB(0, 0, 255)
    End If
End Sub
```

El valor Nothing



- Algunas veces puede que sea necesario desasignar una variable del objeto al cual hace referencia, en este caso debe igualar la variable al valor **Nothing** de la forma siguiente.

  Set Variable_Objeto = **Nothing**

- Habitualmente se utiliza **Nothing** en una estructura condicional para comprobar si la variable objeto está asignada. Observe que si se utiliza una variable objeto a la cual todavía no se le ha hecho ninguna asignación el programa dará error y detendrá su ejecución. Es buena práctica hacer este tipo de comprobaciones antes de trabajar con variables objeto.

```
Sub objeto_Bis()
   Dim R As Range
   Set R = ActiveSheet.Range("E12:F13")
   R.Value = "Milan"
   R.Font.Bold = True
   Set R = Nothing 'Nothing permite asigna a la variable objeto un valor nulo.
                   ' Esto es útil junto con un IF para verificar si la variable esta asignada
   If R Is Nothing Then
      MsgBox Prompt:="La variable Objeto no ha sido asignada", Buttons:=vbOK, _
      Title:="Error"
   Else
      R.Value = "Hola"
   End If
End Sub
```

Condicionales anidadas

```
Sub Condicional_doble()
   Dim a As Integer
   Dim b As Integer
   Dim C As String
   a = ActiveSheet.Range("G10").Value
   b = ActiveSheet.Range("G11").Value
   If a = b Then
      C = "Los valores de G10 y G11 son iguales"
   Else
      If a > b Then
         C = "G10 es mayor que G11"
      Else
         C = "G10 es menor que G11"
      End If
   End If
   ActiveSheet.Range("G9").Value = C
End Sub
```

ElseIf

- El procedimiento anterior se puede abreviar con un EsleIf

```
If condición 1 Then
        Sentencia 1
        Sentencia 2
ElseIf condición 2 Then
        Sentencia 3
        Sentencia 4
ElseIf condición 3 Then
        Sentencia 5
        Sentencia 6
ElseIf condición 4 Then
        Sentencia 7
        Sentencia 8
Else
        Sentencia 9
        Sentencia 10
EndIf
```

```
Sub Condicional_doble_2()
   Dim a As Integer
   Dim b As Integer
   Dim C As String
   a = ActiveSheet.Range("G10").Value
   b = ActiveSheet.Range("G11").Value
   If a = b Then
      C = "Los valores de G10 y G11 son iguales“
   'ElseIf abrevia dos condicuonales anidados
   ElseIf a > b Then
         C = "G10 es mayor que G11"
   Else
         C = "G10 es menor que G11"
   End If
   ActiveSheet.Range("G9").Value = C
End Sub
```

Operador Lógico AND

```
Sub YAcero() 'Uso del condicional AND
    Dim Producto As String, Cantidad As String, Precio As Single
    Dim Total As Single, Descuento As Single, Total_Descuento As Single
    Precio = 0 'UCase convierte a mayúsculas
    Producto = UCase(InputBox("Entrar nombre del Producto", "Entrar"))
    Precio = Val(InputBox("Entrar Precio", "Entrar"))
    Cantidad = Val(InputBox("Entrar Cantidad", "Entrar"))
    Total = Precio * Cantidad
    ActiveSheet.Range("H10").Value = Producto
    ActiveSheet.Range("H11").Value = Precio
    ActiveSheet.Range("H12").Value = Cantidad
    ActiveSheet.Range("H13").Value = Total
    'Si el Total es mayor que 10000 y el producto es Acero, aplicar descuento
    If Total > 10000 And Producto = "ACERO" Then
        Descuento = Val(InputBox("Entrar Descuento", "Entrar"))
        Total_Descuento = Total * (Descuento / 100)
        Total = Total - Total_Descuento
        ActiveSheet.Range("H14").Value = Total_Descuento
        ActiveSheet.Range("H15").Value = Total
    End If
    Range("H12").NumberFormat = "#,##0"'Formato de Celdas
    Range("H11,H13,H14,H15").NumberFormat = "#,##0.00 $"
End Sub
```

Operador Lógico OR

```
Sub OAcero() ' Condicional OR
    Dim Producto As String, Cantidad As Integer, Precio As Single
    Dim Total As Single, Descuento As Single, Total_Descuento As Single
    Precio = 0
    'LCase convierte a minúsculas
    Producto = LCase(InputBox("Entrar Nombre del Producto", "Entrar"))
    Precio = Val(InputBox("Entrar el Precio", "Entrar"))
    Cantidad = Val(InputBox("Entrar la Cantidad", "Entrar"))
    Total = Precio * Cantidad
    ActiveSheet.Range("I10").Value = Producto
    ActiveSheet.Range("I11").Value = Precio
    ActiveSheet.Range("I12").Value = Cantidad
    ActiveSheet.Range("I13").Value = Total
    'si Total es mayor de 10.000 o el producto es Acero, aplicar descuento
    If Total > 10000 Or Producto = "acero" Then
        Descuento = Val(InputBox("Entrad Descuento", "Entrar"))
        Total_Descuento = Total * (Descuento / 100)
        Total = Total - Total_Descuento
        ActiveSheet.Range("I14").Value = Total_Descuento
        ActiveSheet.Range("I15").Value = Total
    End If
End Sub
```

Operador Lógico NOT

```
Sub operadorNO()
    Dim Precio As Integer
    Dim Descuento As Integer
    Precio = 0
    Descuento = 0
    Precio = Val(InputBox("Entrar el Precio", "Entrar"))
    ' Si el valor de la variable precio NO es menor o igual que
1000,
    ' entonces pedir descuento
    If Not Precio <= 1000 Then
       Descuento = Val(InputBox("Entrar Descuento", "Entrar"))
    End If
    ActiveSheet.Range("B19").Value = Precio
    ActiveSheet.Range("B20").Value = Descuento
    ActiveSheet.Range("B21").Value = Precio - Descuento
End Sub
```

Tablas de Verdad

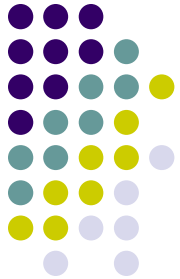
A	B	C	NO(A)	Y(A;B;C)	O(A;B;C)
VERDADERO	VERDADERO	VERDADERO	FALSO	VERDADERO	VERDADERO
VERDADERO	VERDADERO	FALSO	FALSO	FALSO	VERDADERO
VERDADERO	FALSO	VERDADERO	FALSO	FALSO	VERDADERO
VERDADERO	FALSO	FALSO	FALSO	FALSO	VERDADERO
FALSO	VERDADERO	VERDADERO	VERDADERO	FALSO	VERDADERO
FALSO	VERDADERO	FALSO	VERDADERO	FALSO	VERDADERO
FALSO	FALSO	VERDADERO	VERDADERO	FALSO	VERDADERO
FALSO	FALSO	FALSO	VERDADERO	FALSO	FALSO

Calculadora

- Macro que suma, resta, multiplica o divide los valores de las casillas C19 y C20 dependiendo de si C21 contiene el signo +, -, x, :
- El resultado lo deja en C22. Si en C21 no hay ninguno de los signos anteriores en C22 debe dejarse un 0

```
Sub Calculadora()
   Dim Signo As String * 1 'Un solo carácter alfanumérico
   Dim Valor1 As Integer, Valor2 As Integer, Total As Integer
   Valor1 = ActiveSheet.Range("C19").Value
   Valor2 = ActiveSheet.Range("C20").Value
   Signo = ActiveSheet.Range("C21").Value
   Total = 0
   If Signo = "+" Then
      Total = Valor1 + Valor2
   End If
   If Signo = "-" Then
      Total = Valor1 - Valor2
   End If
   If Signo = "x" Then
      Total = Valor1 * Valor2
   End If
   If Signo = ":" Then
      Total = Valor1 / Valor2
   End If
   ActiveSheet.Range("C22").Value = Total
End Sub
```

La estructura Select Case



- La estructura Select Case da mayor legibilidad al programa anterior

```
Sub calcula_case()
   Dim Signo As String * 1
   Dim Valor1 As Integer, Valor2 As Integer, Total As Integer
   Valor1 = ActiveSheet.Range("D19").Value
   Valor2 = ActiveSheet.Range("D20").Value
   Signo = ActiveSheet.Range("D21").Value
   Select Case Signo
      Case "+"
         Total = Valor1 + Valor2
      Case "-"
         Total = Valor1 - Valor2
      Case "x"
         Total = Valor1 * Valor2
      Case ":"
         Total = Valor1 / Valor2
      Case Else
         Total = 0
   End Select
   ActiveSheet.Range("D22").Value = Total
End Sub
```

Ejercicio

- Cree un programa que pregunte la fecha de nacimiento, calcule cuantos días han transcurrido hasta el momento actual y diga en qué día de la semana nació.

Solución Ejercicio

```
Sub nacimiento()
    Dim dias As Integer, Dsemana As Integer, Factual As Date, d As String
    'Dsemana es una variable que da un número que indica el día de la semana
    'dado por la función WEEKDAY, que en Excel es =DIASEM(fecha)
    Static Fnacimiento As Date
    Factual = Date 'Date es la función de VBA equivalente a =HOY()
    Fnacimiento = Factual
    Fnacimiento = InputBox(Prompt:="Introduzca su fecha de nacimiento", Title:="Formato DD-MM-AAAA",
Default:=Fnacimiento)
    dias = Factual - Fnacimiento
    Dsemana = Application.WorksheetFunction.Weekday(Fnacimiento)
    Select Case Dsemana
        Case 1: d = "Domingo"
        Case 2: d = "Lunes"
        Case 3: d = "Martes"
        Case 4: d = "Miercoles"
        Case 5: d = "Jueves"
        Case 6: d = "Viernes"
        Case 7: d = "Sabado"
    End Select
    MsgBox Prompt:="Usted nació un " & d & " hace " & dias & " días" & Chr(10), Title:="Esta información es
correcta siempre que hoy sea " & Factual
End Sub
```

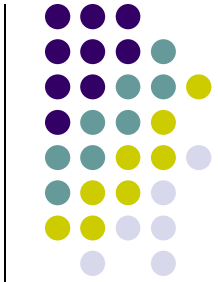
Cada sentencia Case evalúa un rango de valores

```
Sub Media()
    Dim Nota1 As Single, Nota2 As Single, Nota3 As Single
    Dim califica As String, Media As Single
    Nota1 = CSng(InputBox("Entrar Nota primera evaluación", "Nota"))
    Nota2 = CSng(InputBox("Entrar Nota Segunda evaluación", "Nota"))
    Nota3 = CSng(InputBox("Entrar Nota Tercera evaluación", "Nota"))
    Media = (Nota1 + Nota2 + Nota3) / 3
    ActiveSheet.Range("C17").Value = Nota1
    ActiveSheet.Range("D17").Value = Nota2
    ActiveSheet.Range("E17").Value = Nota3
    ActiveSheet.Range("D18").Value = Media
    Select Case Media
        Case Is < 5
            califica = "Suspenso"
        Case 5 To 6.99
            califica = "Aprobado"
        Case 6.99 To 8.99
            califica = "Notable"
        Case Is > 8, 99
            califica = "Sobresaliente"
    End Select
    ActiveSheet.Range("E18").Value = califica
End Sub
```

Select Case y Filtros

```
Sub con_case_y_filtro()
    Dim Signo As String
    Dim Valor1 As Variant, Valor2 As Variant, Total As Single
    Dim Continuar As Boolean
    Valor1 = ActiveSheet.Range("E19").Value
    Valor2 = ActiveSheet.Range("E20").Value
    Signo = ActiveSheet.Range("E21").Value
    Continuar = True
    ' Si en la casilla E19 no hay un valor numérico
    If Not IsNumeric(ActiveSheet.Range("E19")) Then
        MsgBox Prompt:="En E19 no hay ningún valor numérico", Title:="ERROR"
        Continuar = False
    End If
    ' Si en la casilla E20 no hay un valor numérico
    If Not IsNumeric(ActiveSheet.Range("E20")) Then
        MsgBox Prompt:="En E20 no hay ningún valor numérico", Title:="ERROR"
        Continuar = False
    End If
    If IsEmpty(ActiveSheet.Range("E21")) Then
        MsgBox Prompt:="la casilla E21 está vacía", Title:="ERROR"
        Continuar = False
    End If
    If Continuar Then
        Select Case Signo
            Case "+"
                Total = Valor1 + Valor2
            Case "-"
                Total = Valor1 - Valor2
            Case "x"
                Total = Valor1 * Valor2
            Case "/"
                Total = Valor1 / Valor2
            Case Else
                Total = 0
        End Select
        ActiveSheet.Range("E22").Value = Total
    End If
End Sub
```

Lista de Funciones de Comprobación

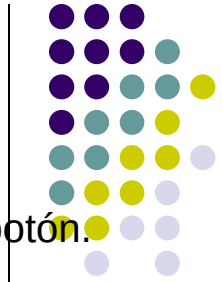


- IsNumériс(Expresión)
  - Comprueba si expresión tiene un valor que se puede interpretar como 'numérico.
- IsDate(Expresión)
  - Comprueba si expresión tiene un valor que se puede interpretar como tipo fecha.
- IsEmpty(Expresión)
  - Comprueba que expresión tenga algún valor, que se haya inicializado.
- IsError(Expresión)
  - Comprueba si expresión devuelve algún valor de error.
- IsArray(Expresión)
  - Comprueba si expresión (una variable) es un array o no.
- IsObject(Expresión)
  - Comprueba si expresión (una variable) representa una variable tipo objeto.
- IsNull(Expresión)
  - Comprueba si expresión contiene un valor nulo debido a datos no válidos.
- Nothing
  - No es propiamente una función, sirve para comprobar si una variable objeto esta 'asociada a un objeto antes de hacer cualquier operación con ella. Recuerde que para trabajar con 'una variable objeto antes debe asignarse a uno (mediante la instrucción Set), en caso contrario se producirá un error en el programa cuando utilice el objeto y se detendrá su ejecución.

Select Case y Filtro

```
Sub con_case_y_filtro_Bis()
' En lugar de los tres If de comprobación se puede utilizar el operador OR de la manera siguiente
    Dim Signo As String
    Dim Valor1 As Variant, Valor2 As Variant, Total As Single
    Dim Continuar As Boolean
    Valor1 = ActiveSheet.Range("F19").Value
    Valor2 = ActiveSheet.Range("F20").Value
    Signo = ActiveSheet.Range("F21").Value
    Continuar = True
    ' Si en la casilla F19 no hay un valor numérico
    If Not IsNumeric(ActiveSheet.Range("F19")) Or Not IsNumeric(ActiveSheet.Range("F20")) Or
IsEmpty(ActiveSheet.Range("F21")) Then
        MsgBox Prompt:="Debe entrar número en F19, F20 y un signo (+,-,x,/) en F21", Title:="ERROR"
    Else
        Select Case Signo
            Case "+“: Total = Valor1 + Valor2
            Case "-“: Total = Valor1 - Valor2
            Case "x“: Total = Valor1 * Valor2
            Case "/“: Total = Valor1 / Valor2
            Case Else: Total = 0
        End Select
        ActiveSheet.Range("F22").Value = Total
    End If
End Sub
```

La función MsgBox (F1)



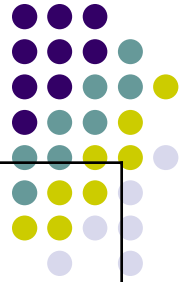
- Muestra un mensaje en un cuadro de diálogo hasta que el usuario pulse un botón. La función devuelve un dato tipo Integer en función del botón pulsado por el usuario. A la hora de invocar está función, se permiten diferentes tipos de botones.

  MsgBox(**Mensaje**, Botones, Título, Archivo de ayuda, contexto)

- **Mensaje**: Obligatorio, es el mensaje que se muestra dentro del cuadro de diálogo.
- **Botones**: Opcional. Es un número o una suma de números o constantes, que sirve para mostrar determinados botones e iconos dentro del cuadro de diálogo. Si se omite este argumento asume valor 0 que corresponde a un único Botón OK.
- **Título** : Opcional. Es el texto que se mostrará en la barra del título del cuadro de diálogo.

```
MsgBox Prompt:="En la casilla A1 no hay ningún valor numérico", Title:="ERROR"
MsgBox Prompt := "La variable Objeto no ha sido asignada", Buttons:=vbOk, Title := "Error"
X= MsgBox ("Hola usuario, Ha acabado el proceso", VbOkOnly, "Mensaje")
X=MsgBox("Desea Continuar", vbYesNo + vbQuestion, "Opción",,)
Mas_datos = MsgBox("Otro registro ?", vbYesNo+vbQuestion,"Entrada de datos")
MsgBox Prompt:=Texto, Buttons:=vbOKOnly + vbInformation, Title:=Titulo
MsgBox ("Debe introducir valores numéricos")
```

MsgBox



```
Sub MesajeCaja()
   Dim nom As String, Respuesta As Integer
   nom = "Antonio"
   MsgBox ("Hola " & nom) 'Se pueden poner paréntesis o no
   MsgBox "Hola " & nom
   MsgBox "Mire el Título", , "Aqui se puede poner el título que se desee"
   MsgBox "Observe este texto" & vbCrLf & "que ocupa" & vbCrLf & "tres líneas",, "Título"
   MsgBox "Mire el icono de" & vbCrLf & "Interrogación", vbQuestion, _
   "Icono de Interrogación"
   MsgBox "Otro icono", vbCritical, "Icono Crítico" 'Sonido
   MsgBox "Otro", vbExclamation, "Icono Exclamación" 'Sonido
   MsgBox "Otro más", vbInformation, "Icono Información" 'Sonido
   Respuesta = MsgBox("Observe que al incluir más" & vbCrLf & _
   "de un botón, en el MsgBox" & vbCrLf & "pongo paréntesis y utilizo" _
   & vbCrLf & "una variable que recogerá" & vbCrLf & "el botón que hemos pulsado", _
   vbYesNo + vbQuestion, "Dos Botones")
   MsgBox "La Respuesta ha sido " & Respuesta, , "Respuesta"
   Respuesta = MsgBox("Tres Botones", vbYesNoCancel + vbInformation, _
   "Con icono de Información") 'Con paréntesis necesariamente
   MsgBox "La Respuesta ha sido " & Respuesta, , "Respuesta"
   Respuesta = MsgBox("Tres Botones pero" & vbCrLf & "el activo es el segundo", _
   vbAbortRetryIgnore + vbCritical + vbDefaultButton2, "Icono Crítico")
   MsgBox "La Respuesta ha sido " & Respuesta, , "Respuesta"
End Sub
```

InputBox

- Variable = InputBox (mensaje, Titulo, Defecto, Coordenada Horizontal, Coordenada Vertical)
- Las coordenadas se miden en Twips desde el extremo superior izquierdo de la ventana
  - 1 cm = 566 Twips
  - 1 pixel = 15 Twips

```
Sub InputCaja()
   Dim Respuesta As String
   Respuesta = InputBox("Primera Línea" & vbCrLf & Chr(9) _
   & "Segunda Línea con Tabulador Chr(9)", "Aquí el Título") 'Chr(10) equivale a vbCrLf
   Respuesta = InputBox("Haz clic en [Cancel]", "A ver que pasa si se cacela")
   MsgBox "Al pulsar Calcelar el resultado es = " & Respuesta 'Respuesta nula ""
   Respuesta = InputBox("Aparece un valor por defecto", "Título", "Aparece esto por defecto")
   Respuesta = InputBox("Situo la ventana", "1200 Twips a la derecha y 1400 hacia abajo", "coordenadas
1200x1400", 1200, 1400)
   Respuesta = InputBox("Otra posición", , "1 cm = 566 Twips  y  1 pixel = 15 Twips", 50, 75)
End Sub
```

La instrucción With (repaso)

```
Sub OAcero_with()
    Dim Producto As String
    Dim Cantidad As Integer
    Dim Precio As Single
    Dim Total As Single
    Dim Descuento As Single
    Dim Total_Descuento As Single
    Precio = 0
    Producto = LCase(InputBox("Entrar Nombre del Producto", "Entrar"))
    Precio = Val(InputBox("Entrar el precio", "Entrar"))
    Cantidad = Val(InputBox("Entrar la cantidad", "Entrar"))
    Total = Precio * Cantidad
    With ActiveSheet
        .Range("J10").Value = Producto
        .Range("J11").Value = Precio
        .Range("J12").Value = Cantidad
        .Range("J13").Value = Total
    End With
    ' Si total mayor que 10.000 o el producto es Acero, aplicar descuento.
    If Total > 10000 Or Producto = "Acero" Then
        Descuento = Val(InputBox("Entrar Descuento", "Entrar"))
        Total_Descuento = Total * (Descuento / 100)
        Total = Total - Total_Descuento
        With ActiveSheet
            .Range("J14").Value = Total_Descuento
            .Range("J15").Value = Total
        End With
    End If
End Sub
```

Estructuras repetitivas

```
Sub Media_notas()
    Dim Nota As Integer
    Dim Media As Single

    Media = 0

    'Observe que este programa repite el siguiente bloque de sentencias, 5 veces

    Nota = Val(InputBox("Entrar la Nota 1: ", "Entrar Nota"))
    ActiveSheet.Range("G17").Value = Nota
    Media = Media + Nota

    Nota = Val(InputBox("Entrar la Nota 2: ", "Entrar Nota"))
    ActiveSheet.Range("G18").Value = Nota
    Media = Media + Nota

    Nota = Val(InputBox("Entrar la Nota 3: ", "Entrar Nota"))
    ActiveSheet.Range("G19").Value = Nota
    Media = Media + Nota

    Nota = Val(InputBox("Entrar la Nota 4: ", "Entrar Nota"))
    ActiveSheet.Range("G20").Value = Nota
    Media = Media + Nota

    Nota = Val(InputBox("Entrar la Nota 5: ", "Entrar Nota"))
    ActiveSheet.Range("g21").Value = Nota
    Media = Media + Nota

    Media = Media / 5
    ActiveSheet.Range("G22").Value = Media
End Sub
```

Bucle For … Next

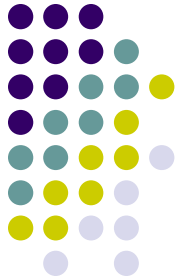
```
Sub Totalizar()

    Dim i As Integer
    Dim Total As Integer
    Dim Valor As Integer

    For i = 1 To 10
        Valor = Val(InputBox("Entrar el valor " & i, "Entrada"))
        Total = Total + Valor
    Next i

    ActiveSheet.Range("C11").Value = Total
End Sub
```

Recorrer casillas de una Hoja

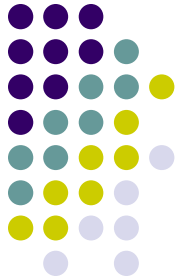


- **Propiedad Cells**
  - sirve para referenciar una celda o un rango de celdas según coordenadas de fila y columna

```
Sub rellenar() 'Rellena de H16 a H20 con los pares del 2 al 10
    Dim Fila As Integer, i As Integer
    Fila = 16
    For i = 2 To 10 Step 2
        ActiveSheet.Cells(Fila, 8).Value = i
        Fila = Fila + 1 'Esto es un contador
    Next i
End Sub
```

```
Sub rellenar_Bis() 'Rellena de H16 a H20 con los pares del 2 al 10, sin contador Fila
    Dim i As Integer
    For i = 16 To 20
        ActiveSheet.Cells(i, 9).Value = i * 2 - 30
    Next i
End Sub
```

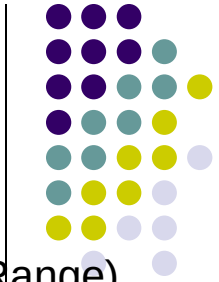
Rellenar una serie



- Llenar un rango de filas, empezando por una celda, que se debe especificar desde teclado, con una serie de 10 valores correlativos (comenzando por el 1).

```
Sub serie()
    Dim Casilla_Inicial As String
    Dim i As Integer
    Dim Fila As Integer, Columna As Integer
    Casilla_Inicial = InputBox("Introducir la casilla Inicial : “ & chr(10) & “Por ejemplo la K10”,
"Casilla Inicial")
    ActiveSheet.Range(Casilla_Inicial).Activate
    Fila = ActiveCell.Row
    Columna = ActiveCell.Column
    'ROW y COLUMN devuelven la fila y la columna de un objeto range.
    'en este caso se utilizan para obtener la fila y la columna de la casilla activa.
    For i = 1 To 10
        ActiveSheet.Cells(Fila, Columna).Value = i
        Fila = Fila + 1
    Next i
End Sub
```

Rellenar una serie



- Recuerde que cuando utilizamos **Cells** como propiedad de un rango (Objeto Range), **Cells** empieza a contar a partir de la casilla referenciada por **Range**

```
Sub serie_Bis()
    Dim Casilla_Inicial As String
    Dim i As Integer
    Dim Fila As Integer, Columna As Integer
    Casilla_Inicial = InputBox("Introducir la casilla Inicial : " & chr(10) & “Por ejemplo la L10”, "Casilla Inicial")
    ActiveSheet.Range(Casilla_Inicial).Activate
    Fila = 1
    For i = 1 To 10
        ActiveSheet.Range(Casilla_Inicial).Cells(Fila, 1).Value = i
        Fila = Fila + 1
    Next i
End Sub
```

Rellenar una serie

- Una variante del programa anterior.
- No se usa Fila, se usa la variable del For

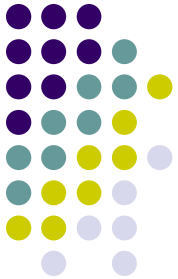
```
Sub serie_Tris()
    Dim Casilla_Inicial As String
    Dim i As Integer
    Dim Fila As Integer, Columna As Integer
    Casilla_Inicial = InputBox("Introducir la casilla Inicial : " & chr(10) &
“Por ejemplo la M10”, "Casilla Inicial")
    ActiveSheet.Range(Casilla_Inicial).Activate
    ‘ Activate (con Range) activa una sola celda. Range("B2").Activate
    ‘ Para seleccionar un rango de celdas, use el método Select. Range("A1:C3").Select
    For i = 1 To 10
        ActiveSheet.Range(Casilla_Inicial).Cells(i, 1).Value = i
    Next i
End Sub
```

For-Next y Cells

- Volvemos a calcular las notas medias, pero usando la estructura **For_Next** y la propiedad **Cells**

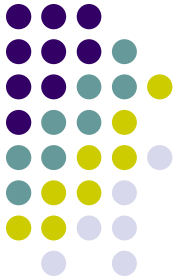
```
Sub Media_notas_Bis()
    Dim Nota As Integer
    Dim Media As Single
    Dim Fila As Integer
    Media = 0
    For Fila = 1 To 5
        Nota = Val(InputBox("Entrar la " & " Nota " & Fila, "Entrar Nota"))
        ActiveSheet.Range(“N10").Cells(Fila, 1) = Nota
        'lo de Range(“N10") se pone para marcar la celda de inicio,
        'si no se pone comienza en A1
        Media = Media + Nota 'esto es un acumulado
    Next Fila
    Media = Media / 5
    ActiveSheet.Range(“N10").Cells(6, 1).Value = Media
End Sub
```

Propiedad Offset



- Esta propiedad es también muy útil a la hora de recorrer rango.
- **Offset**, que significa desplazamiento, es una propiedad del objeto **Range** y se utiliza para referenciar una casilla situada a n Filas y n Columnas de una casilla dada.
- Ejemplos:
  - **ActiveSheet**.**Range**("A1").**Offset**(2, 2).**Value** = "Hola“
    - *Casilla C3 = Hola, 2 filas y 2 columnas desde A1.*
  - **ActiveCell**.**Offset**(5,1).**Value** = "Hola“
    - *5 Filas por debajo de la casilla Activa = Hola*
  - **ActiveCell**.**Offset**(2,2).**Activate**
    - *Activar la casilla que está 2 filas y 2 columnas de la activa*

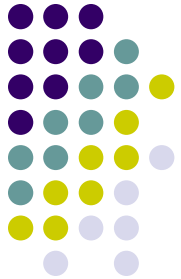
For-Next y Offset.
Sin cambiar celda activa



- Recorrer rangos con la propiedad OffSet (desplazamiento)

```
Sub Media_notas_Tris()
    Dim Nota As Integer
    Dim Media As Single
    Dim Fila As Integer
    Media = 0
    ActiveSheet.Range("O10").Activate 'la casilla activa siempre es la misma
    For Fila = 0 To 4
        Nota = Val(InputBox("Entrar la " & " Nota " & Fila + 1, "Entrar Nota"))
        ActiveCell.Offset(Fila, 0).Value = Nota
        Media = Media + Nota
    Next Fila
    Media = Media / 5
    ActiveCell.Offset(5, 0).Value = Media
End Sub
```

For-Next y Offset.
Cambia Celda Activa



```
Sub Media_notas_Tetra()
    Dim Nota As Integer
    Dim Media As Single
    Dim i As Integer
    Media = 0
    ActiveSheet.Range("P10").Activate
    For i = 1 To 5
        Nota = Val(InputBox("Entrar la " & " Nota " & i, "Entrar Nota"))
        ActiveCell.Value = Nota
        Media = Media + Nota
        'Hacer activa la casilla situada una fila por debajo de la actual
        ActiveCell.Offset(1, 0).Activate
    Next i
    Media = Media / 5
    ActiveCell.Value = Media
End Sub
```

Do While..Loop

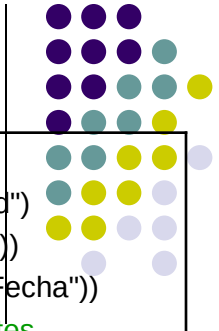
Estructura Repetitiva (Hacer Mientras)

- La estructura repetitiva **FOR** se adapta perfectamente a aquellas situaciones en que se sabe previamente el número de veces que se ha de repetir un proceso
- Do While..Loop es una estructura repetitiva que se repite mientras se cumpla el criterio

```
Do While Condición
Sentencia1
Sentencia2
    .
    .
Sentencia N
Loop
```

- En las sentencias interiores se tiene que producir en algún momento un cambio que haga que la condición deje de cumplirse para así poder salir del bucle.

Rellenar una Base de Datos

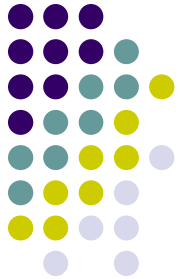


```
Sub Registros()
   'Rellenar los registros de una Base de Datos. Hoja3
   Dim Nombre As String, Ciudad As String
   Dim Edad As Integer, Fecha As Date
   'Activar Hoja3
   Worksheets("Hoja3").Activate
   With ActiveSheet
      .Range("B4").Value = "Nombre"
      .Range("C4").Value = "Ciudad"
      .Range("D4").Value = "Edad"
      .Range("E4").Value = "Fecha"
   End With
   'Para poner negrita y centrar la cabecera
   Range("B4:E4").Select
   With Selection
      .Font.Bold = True
      .HorizontalAlignment = xlCenter
   End With
   'Activar casilla B5
   ActiveSheet.Range("B5").Activate
   Nombre = InputBox("Entre el Nombre (Return para Terminar)
: ", "Nombre")
    'Mientras la variable Nombre sea diferente a cadena vacía
   Do While Nombre <> ""
      Ciudad = InputBox("Entre la Ciudad : ", "Ciudad")
      Edad = Val(InputBox("Entre la Edad : ", "Edad"))
      Fecha = CDate(InputBox("Entra la Fecha : ", "Fecha"))
      'Copiar los datos en las casillas correspondientes
      With ActiveCell
         .Value = Nombre
         .Offset(0, 1).Value = Ciudad
         .Offset(0, 2).Value = Edad
         .Offset(0, 3).Value = Fecha
      End With
      'Hacer activa la celda de la fila siguiente a la actual
      ActiveCell.Offset(1, 0).Activate
      Nombre = InputBox("Entre el Nombre (Return para
Terminar) : ", "Nombre")
   Loop 'pide nuevos datos mientras nombre no este vacío
   'Seleccionamos la Base de Datos y la ponemos amarilla
   Application.Goto Reference:="R4C2"
   Selection.CurrentRegion.Select
   With Selection.Interior
      .ColorIndex = 6
      .Pattern = xlSolid
   End With
End Sub
```

Detecta donde nos hemos quedado

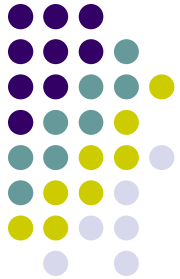
```
Sub Registros_Bis()
    Dim Nombre As String
    Dim Ciudad As String
    Dim Edad As Integer
    Dim Fecha As Date
    Worksheets("Hoja3").Activate
    ActiveSheet.Range("B4").Activate
    'Buscar la primera celda vacía de la columna B y convertirla en activa
    Do While Not IsEmpty(ActiveCell)
        ActiveCell.Offset(1, 0).Activate
    Loop
    Nombre = InputBox("Entre el Nombre (Return para Terminar) : ", "Nombre")
    ' Mientras la variable Nombre sea diferente a cadena vacía
    Do While Nombre <> ""
        Ciudad = InputBox("Entre la Ciudad : ", "Ciudad")
        Edad = Val(InputBox("Entre la Edad : ", "Edad"))
        Fecha = CDate(InputBox("Entra la Fecha : ", "Fecha"))
        With ActiveCell
            .Value = Nombre
            .Offset(0, 1).Value = Ciudad
            .Offset(0, 2).Value = Edad
            .Offset(0, 3).Value = Fecha
        End With
        ActiveCell.Offset(1, 0).Activate
        Nombre = InputBox("Entre el Nombre (Return para Terminar) : ", "Nombre")
    Loop
End Sub
```

¿Desea introducir más datos ?



```
Sub Registros_Tris()
    Dim Nombre As String
    Dim Ciudad As String
    Dim Edad As Integer
    Dim Fecha As Date
    Dim Mas_datos As Integer 'Mas_datos es una variable de tipo Integer
    Worksheets("Hoja3").Activate
    ActiveSheet.Range("B4").Activate
    'Buscar la primera celda vacía de la columna B y convertirla en activa
    Do While Not IsEmpty(ActiveCell)
        ActiveCell.Offset(1, 0).Activate
    Loop
    Mas_datos = vbYes
    'es necesaria la línea anterior al bucle Mas_datos = vbYes, para que cuando se evalúe la
    'condición por vez primera esta se cumpla y se ejecuten las sentencias de dentro del bucle
    Do While Mas_datos = vbYes
        Nombre = InputBox("Entre el Nombre: ", "Nombre")
        Ciudad = InputBox("Entre la Ciudad : ", "Ciudad")
        Edad = Val(InputBox("Entre la Edad : ", "Edad"))
        Fecha = CDate(InputBox("Entra la Fecha : ", "Fecha"))
        With ActiveCell
            .Value = Nombre
            .Offset(0, 1).Value = Ciudad
            .Offset(0, 2).Value = Edad
            .Offset(0, 3).Value = Fecha
        End With
        ActiveCell.Offset(1, 0).Activate
        'Preguntar al usuario si desea entrar otro registro
        Mas_datos = MsgBox("Otro registro ?", vbYesNo + vbQuestion, "Entrada de datos")
    Loop
End Sub
```

Estructura Do..Loop While

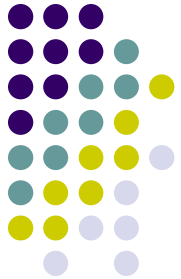


- El funcionamiento de esta estructura repetitiva es similar a la anterior salvo que la condición se evalúa al final, la inmediata consecuencia de esto es que las instrucciones del cuerpo del bucle se ejecutaran al menos una vez.
- Esta estructura es más adecuada para casos como el anterior. Si vamos a entrar datos, al menos uno entraremos, por tanto las instrucciones del cuerpo del bucle se deben ejecutar al menos una vez, luego ya decidiremos si se repiten o no.
- En este caso no es necesario la línea Mas_Datos = vbYes antes de **Do** para forzar la entrada en el bucle ya que la condición va al final.

Do..Loop While

```
Sub Registros_Tetra()
    Dim Nombre As String
    Dim Ciudad As String
    Dim Edad As Integer
    Dim Fecha As Date
    Dim Mas_datos As Integer
    'Mas_datos es una variable de tipo Integer
    Worksheets("Hoja3").Activate
    ActiveSheet.Range("B4").Activate
    'Buscar la primera celda vacía de la columna B y convertirla en activa
    Do While Not IsEmpty(ActiveCell)
        ActiveCell.Offset(1, 0).Activate
    Loop
    Do
        Nombre = InputBox("Entre el Nombre: ", "Nombre")
        Ciudad = InputBox("Entre la Ciudad : ", "Ciudad")
        Edad = Val(InputBox("Entre la Edad : ", "Edad"))
        Fecha = CDate(InputBox("Entra la Fecha : ", "Fecha"))
        With ActiveCell
            .Value = Nombre
            .Offset(0, 1).Value = Ciudad
            .Offset(0, 2).Value = Edad
            .Offset(0, 3).Value = Fecha
        End With
        ActiveCell.Offset(1, 0).Activate
        Mas_datos = MsgBox("Otro registro ?", vbYesNo + vbQuestion, "Entrada de datos")
        'Mientras Mas_datos = vbYes
    Loop While Mas_datos = vbYes
End Sub
```

Estructura Do..Loop Until



- **Hacer.. Hasta que se cumpla la condición**
- Es otra estructura que evalúa la condición al final.
- La interpretación es distinta, ya que el bucle se va repitiendo **HASTA que se cumple la condición**,
- no MIENTRAS se cumple la condición.
- De las dos estructura use la que más le guste

Do..Loop Until

```
Sub Registros_Penta()
    Dim Nombre As String
    Dim Ciudad As String
    Dim Edad As Integer
    Dim Fecha As Date
    Dim Mas_datos As Integer
    'Mas_datos es una variable de tipo Integer
    Worksheets("Hoja3").Activate
    ActiveSheet.Range("B4").Activate
    'Buscar la primera celda vacía de la columna B y convertirla en activa
    Do While Not IsEmpty(ActiveCell)
        ActiveCell.Offset(1, 0).Activate
    Loop
    Do
        Nombre = InputBox("Entre el Nombre: ", "Nombre")
        Ciudad = InputBox("Entre la Ciudad: ", "Ciudad")
        Edad = Val(InputBox("Entre la Edad: ", "Edad"))
        Fecha = CDate(InputBox("Entre la Fecha: ", "Fecha"))
        With ActiveCell
            .Value = Nombre
            .Offset(0, 1).Value = Ciudad
            .Offset(0, 2).Value = Edad
            .Offset(0, 3).Value = Fecha
        End With
        ActiveCell.Offset(1, 0).Activate
        Mas_datos = MsgBox("Otro registro ?", vbYesNo + vbQuestion, "Entrada de datos")
        'Hasta que Mas_Datos sea igual a vbNo
    Loop Until Mas_datos = vbNo
End Sub
```

Estructura For Each

- Este bucle se utiliza básicamente para ejecutar un grupo de sentencias con los elementos de una colección o una matriz.
- Recuerde que una colección es un conjunto de objetos, hojas, rangos, etc.

```
‘Para cambiar los nombres de las hojas de un libro de trabajo
Sub NombraHojas()
   'Programa que pregunta el nombre para cada hoja de un libro de trabajo,
   'si no se pone nombre a la hoja, queda el que tiene.
   Dim Nuevo_Nombre As String
   Dim hoja As Worksheet
   ' Para cada hoja del conjunto WorkSheets
   For Each hoja In Worksheets
      Nuevo_Nombre = InputBox("Nombre de la Hoja : " & hoja.Name, "Nombrar Hojas")
      If Nuevo_Nombre <> "" Then
         hoja.Name = Nuevo_Nombre
      End If
   Next
   '** Hoja va referenciando cada una de las hojas del conjunto WorkSheets a cada paso de bucle
End Sub
```

EXIT FOR

- Esta macro es una variante de la anterior
- Si se pulsa CANCEL o el nombre de hoja esta vacío “” se sale del bucle con un EXIT FOR.
- **EXIT FOR** permite salir de un bucle FOR o FOR EACH, mientras que **EXIT DO** abandona directamente un bucle DO
- Además nos hemos ahorrado el END IF

```
Sub NombraHojas2()
   'Si se pulsa cancelar o no se pone nada en el nombre se sale con el EXIT FOR
   Dim Nuevo_Nombre As String
   Dim hoja As Worksheet
   For Each hoja In Worksheets
      Nuevo_Nombre = InputBox("Nombre de la Hoja : " & hoja.Name, "Nombrar Hojas",
hoja.Name)
      If Nuevo_Nombre = "" Then Exit For 'EXIT FOR sale del bucle
      hoja.Name = Nuevo_Nombre
   Next
End Sub
```

Llenar un Rango

```
Sub Llena_Rango()
    Dim R As Range
    Worksheets("Hoja1").Activate
    ' Para cada celda del rango N16:P19 de la Hoja1
    For Each R In ActiveSheet.Range("N16:P19")
        R.Value = InputBox("Entrar valor para la celda " & R.Address, "Entrada de valores")
    Next
End Sub
```

- Se ha declarado una variable tipo Range, este tipo de datos sirve para guardar Rangos de una o más casillas, estas variables pueden luego utilizar todas las propiedades y métodos propios de los Objetos Range.
- La asignación de las variables que sirven para guardar o referenciar objetos (Range, WorkSheet, etc.) deben inicializarse muchas veces a través de la instrucción SET

Procedimientos

- En los programas largos conviene dividir el trabajo en varios procedimientos.
- Inconvenientes de los procedimientos largos:
  - grandes bloques de código implican mayor complicación del mismo
  - repetición de sentencias
  - mayores problemas de seguimiento a la hora de:
    - depurar errores
    - ampliar funcionalidades
    - incluir modificaciones
- Filosofía de “divide y vencerás”
  - tratar cada problema o tarea de forma más o menos aislada
- Para llamar un procedimiento desde otro se utiliza la instrucción
  Call *Nombre_Procedimiento*

```
Sub P_Uno()
Sentencias
.
Call P_Dos()
.
Sentencias
.
End Sub
```

Call

```
Sub Registros_Hexa()
'el mismo procedimiento que Registros_Bis() pero usando una llamada CALL a otro procedimiento
'el código que salta casilla hasta que se encuentra una vacía se implementa en un procedimiento
'llamado, Saltar_Celdas_Llenas.
'Para entrar valores se ha sustituido Do While..Loop por Do.. Loop While.
    Dim Nombre As String
    Dim Ciudad As String
    Dim Edad As Integer
    Dim fecha As Date
    Dim Mas_datos As Integer
    ' Llamada a la función Saltar_Celdas_Llenas, el programa salta aquí a ejecutar las
    ' instrucciones de este procedimiento y luego vuelve para continuar la ejecución
    ' a partir de la instrucción Do
    Call Saltar_Celdas_Llenas
    Do
        Nombre = InputBox("Entre el Nombre: ", "Nombre")
        Ciudad = InputBox("Entre la Ciudad : ", "Ciudad")
        Edad = Val(InputBox("Entre la Edad : ", "Edad"))
        fecha = CDate(InputBox("Entra la Fecha : ", "Fecha"))
        With ActiveCell
            .Value = Nombre
            .Offset(0, 1).Value = Ciudad
            .Offset(0, 2).Value = Edad
            .Offset(0, 3).Value = fecha
        End With
        ActiveCell.Offset(1, 0).Activate
        Mas_datos = MsgBox("Otro registro ?", vbYesNo + vbQuestion, "Entrada de datos")
    Loop While Mas_datos = vbYes
End Sub
```

Función llamada

```
Sub Saltar_Celdas_Llenas()
    Worksheets("Hoja3").Activate
    ActiveSheet.Range("B4").Activate
    Do While Not IsEmpty(ActiveCell)
        ActiveCell.Offset(1, 0).Activate
    Loop
End Sub
```

- Función que salta celdas de una misma columna.
- Sirve para encontrar la primera celda vacía de la columna

Pasar parámetros

- Los parámetros son el mecanismo por el cual un procedimiento puede pasarle valores a otro y de esta forma condicionar, moldear, etc. las acciones que ejecuta.
- El procedimiento llamado gana entonces en flexibilidad. La sintaxis de llamada de un procedimiento es la siguiente:

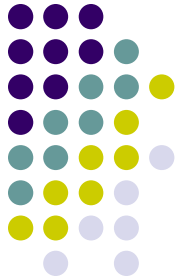
  Call Procedimiento(Parámetro1, Parámetro2,..., ParámetroN)

- Los parámetros pueden ser valores o variables.
- La sintaxis para el procedimiento llamado es la siguiente:

  Sub Procedimiento(Parámetro1 as Tipo,..., ParámetroN As Tipo)

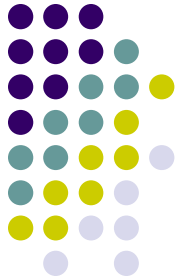
- Observe que aquí los parámetros son variables que recibirán los valores y evidentemente debe haber coincidencia de tipo.
  - Por ejemplo, si el primer parámetro es una variable tipo Integer, el primer valor que se le debe pasar al procedimiento cuando se llama también ha de ser de tipo Integer (recuerde que puede ser un valor directamente o una variable).

Call **Procedimiento(Parámetro1, Parámetro2,..., ParámetroN)**



```
Sub Registros_Septa()
    Dim Nombre As String
    Dim Ciudad As String
    Dim Edad As Integer
    Dim fecha As Date
    Dim Mas_datos As Integer
    ' Llamada a la función Saltar_Celdas_Llenas_Bis
    ' Mediante dos parámetros se comunica al procedimiento llamado en que hoja y celda comenzar
    Call Saltar_Celdas_Llenas_Bis("Hoja3", "B4")
    'Los parámetros pueden ser valores o variables
    Do
        Nombre = InputBox("Entre el Nombre : ", "Nombre")
        Ciudad = InputBox("Entre la Ciudad : ", "Ciudad")
        Edad = Val(InputBox("Entre la Edad : ", "Edad"))
        fecha = CDate(InputBox("Entre la Fecha : ", "Fecha"))
        With ActiveCell
            .Value = Nombre
            .Offset(0, 1).Value = Ciudad
            .Offset(0, 2).Value = Edad
            .Offset(0, 3).Value = fecha
        End With
        ActiveCell.Offset(1, 0).Activate
        Mas_datos = MsgBox("Otro registro ?", vbYesNo + vbQuestion, "Entrada de datos")
    Loop While Mas_datos = vbYes
End Sub
```

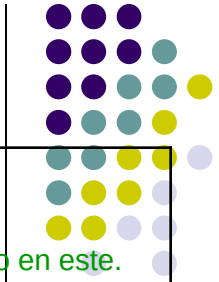
Procedimiento con parámetros



```
Sub Saltar_Celdas_Llenas_Bis(hoja As String, Casilla_Inicial As String)
   'los parámetros son variables que recibirán los valores
   'debe haber coincidencia de tipos.
   Worksheets(hoja).Activate
   ActiveSheet.Range(Casilla_Inicial).Activate
   Do While Not IsEmpty(ActiveCell)
      ActiveCell.Offset(1, 0).Activate
   Loop
End Sub
```

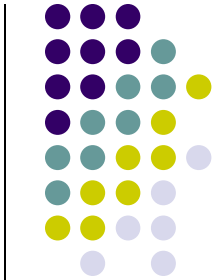
- Sirve para Saltar celdas llenas de una columna hasta encontrar una vacía que se convierte en activa
- Parámetros :
  - Hoja : Hoja donde está el rango a saltar.
  - Casilla_Inicial : Casilla Inicial de la columna
- Gracias a los parámetros, sirve para recorrer cualquier rango en cualquier hoja.

Los parámetros pueden ser valores o variables



```
Sub Registros_Octa()
    Dim Nombre As String, Ciudad As String, Edad As Integer, fecha As Date, Mas_datos As Integer
    ' Al procedimiento Saltar_Celdas_Llenas_Bis se le pueden pasar valores como en el caso anterior, o variables como en este.
'****************** novedad **********************
    Dim hoja As String
    Dim Casilla_Inicial As String
    hoja = InputBox("En que hoja está la base de datos : ", "Entrar Nombre de Hoja")
    Casilla_Inicial = InputBox("En que casilla comienza la base de datos", "Casilla Inicial")
    ' Observe que los parámetros son dos variables cuyo valor se ha entrado desde teclado en
    ' las dos instrucciones InputBox anteriores.
    Call Saltar_Celdas_Llenas_Bis(hoja, Casilla_Inicial)
'****************** novedad **********************
   Do
       Nombre = InputBox("Entre el Nombre : ", "Nombre")
       Ciudad = InputBox("Entre la Ciudad : ", "Ciudad")
       Edad = Val(InputBox("Entre la Edad : ", "Edad"))
       fecha = CDate(InputBox("Entre la Fecha : ", "Fecha"))
       With ActiveCell
          .Value = Nombre
          .Offset(0, 1).Value = Ciudad
          .Offset(0, 2).Value = Edad
          .Offset(0, 3).Value = fecha
       End With
       ActiveCell.Offset(1, 0).Activate
       Mas_datos = MsgBox("Otro registro ?", vbYesNo + vbQuestion, "Entrada de datos")
    Loop While Mas_datos = vbYes
End Sub
```

Variables Locales y variables Globales



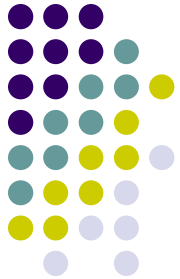
- El ámbito de una variable declarada dentro de una función es la propia función.
- Es decir, no podrá utilizarse fuera de dicha función.
- Así, el siguiente programa que debería sumar las cinco filas siguientes a partir de la casilla activa y guardar el resultado en la sexta es **incorrecto**.

```
Sub Hacer()
        .
        .
  Call Sumar_Cinco_Siguientes
  ActiveCell.Offset(6,0).Value = Suma
        .
        .
End Sub
```

```
Sub Sumar_Cinco_Siguientes()
  Dim i As Integer
  Dim Suma As Single
  Suma=0
  For i=1 To 5
    Suma = Suma+ActiveCell.Offset(i,0).Value
  Next i
End Sub
```

- Es incorrecto porque tanto las variable *i* como la variable *Suma* están declaradas dentro del procedimiento *Sumar_Cinco_Siguientes* consecuentemente, su ámbito de acción es este procedimiento.
- Por tanto, la instrucción *ActiveCell.Offset(6,0).Value = Suma* del procedimiento *Hacer*, generaría un error (con Option Explicit activado) ya que la variable *Suma* no está declarada dentro de él.
- Si piensa en declarar la variable *Suma* dentro del procedimiento *Hacer*, no solucionará nada porque esta será local a dicho procedimiento, en este caso tendría dos variables llamadas *Suma* pero cada una de ellas local a su propio procedimiento y consecuentemente con el ámbito de acción restringido a ellos.

Variables Globales



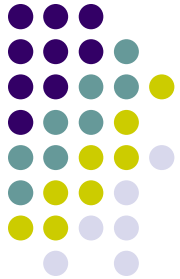
- Una solución seria declarar “suma” como variable global.
- Una variable global se declara fuera de todos los procedimientos y es reconocida por todos los procedimientos del módulo.

```
Option Explicit
Dim Suma As Single  'Suma es una variable global reconocida por todos los procedimientos del módulo
```

```
  Sub Hacer_Bis()
        .
  Call Sumar_Cinco_Siguientes_Bis
  ActiveCell.Offset(6,0).Value = Suma
        .
End Sub
```

```
Sub Sumar_Cinco_Siguientes_Bis()
  Dim i As Integer
  Suma=0
  For i=1 To 5
    Suma = Suma+ActiveCell.Offset(i,0).Value
  Next i
End Sub
```

Pasar variables como parámetros

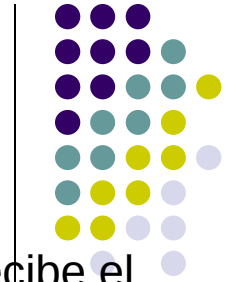


- La variable parámetro *S* (a la que se ha cambiado el nombre adrede) de *Sumar_Cinco_Siguientes_Tris* es la variable *Suma* declarada en *Hacer_Tris*.
- Funcionará porque en Visual Basic, a menos que se indique lo contrario, el paso de parámetros es por referencia.

```
Sub Hacer_Tris()
  Dim Suma As Single
            .
            .
  ' Llamada a la función Sumar_Cinco_Siguientes pasándole la variable Suma
  Call Sumar_Cinco_Siguientes_Tris(Suma)
  ActiveCell.Offset(6,0).Value = Suma
            .
            .
End Sub
```

```
Sub Sumar_Cinco_Siguientes_Tris(S As Single)
  Dim i As Integer
  Suma=0
  For i=1 To 5
    S = S+ActiveCell.Offset(i,0).Value
  Next i
End Sub
```

Paso por referencia y paso por valor

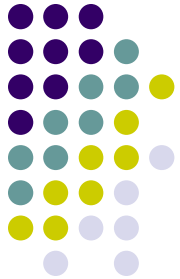


- El paso por valor significa que la variable parámetro del procedimiento recibe el valor de la variable (o directamente el valor) de su parámetro correspondiente de la instrucción de llamada y en el paso por referencia, la variable parámetro del procedimiento es la misma que su parámetro correspondiente de la instrucción de llamada, es decir, la declarada en el procedimiento desde el que se hace la llamada.
- Por defecto, y siempre que en la instrucción de llamada se utilicen variables, las llamadas son por referencia.
- Si desea que el paso de parámetros sea por valor, debe anteponer a la variable parámetro la palabra reservada **ByVal**

```
Sub Saltar_Celdas_Llenas(ByVal Hoja As String, ByVal Casilla_Inicial As String)
```

- Aunque lo elegante y efectivo por razones de memoria seria pasar siempre que sea posible por valor, es poco habitual que así se haga en Visual Basic, seguramente por comodidad.
- Como suponemos que hará como la mayoría, es decir, pasar por referencia, tenga cuidado con los (indeseables) efectos laterales.

Efecto Lateral

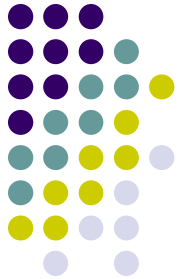


- Este programa **no funciona** bien
- En la Hoja4 disponemos de 5 valores en cada una de las tres columnas B,C,D, y deseamos sumarlos
- Debería sumar los cinco valores de cada columna y poner su suma justo bajo ellos
- El mal funcionamiento se debe a que la variable Fila pasa al procedimiento llamado, como variable y no como valor, pese a que se cambia el nombre por F, sigue siendo la misma

```
Sub Efecto_Lateral()
    Dim Fila As Integer
    Worksheets("Hoja4").Activate
    Fila = 5
    Call Recorrer_Sumar(Fila, 2, 5) ' Columna B
    Call Recorrer_Sumar(Fila, 3, 5) ' Columna C
    Call Recorrer_Sumar(Fila, 4, 5) ' Columna D
End Sub
```

```
Sub Recorrer_Sumar(F As Integer, C As Integer, Q As Integer)
    Dim i As Integer
    Dim Total As Integer
    Total = 0
    For i = 1 To Q
        Total = Total + ActiveSheet.Cells(F, C).Value
        F = F + 1 ' OJO con esta asignación, recuerde que F es la variable Fila declarada en el procedimiento Efecto_Lateral
    Next i
    ActiveSheet.Cells(F, C) = Total
End Sub
```

ByVal

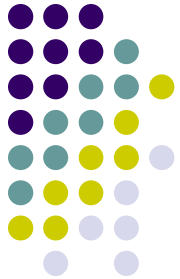


- Se corrige añadiendo ByVal a la variable, lo que hace que pase como valor.

```
Sub Efecto_Lateral_bis() 'Este procedimiento es igual al Efecto_Lateral
'con la salvedad de que en este se llama a Recorrer_Sumar_bis
    Dim Fila As Integer
    Worksheets("Hoja4").Activate
    Fila = 5
    Call Recorrer_Sumar_bis(Fila, 2, 5) ' Columna B
    Call Recorrer_Sumar_bis(Fila, 3, 5) ' Columna C
    Call Recorrer_Sumar_bis(Fila, 4, 5) ' Columna D
End Sub
```

```
Sub Recorrer_Sumar_bis(ByVal F As Integer, C As Integer, Q As Integer)
'Este sub es idéntico al anterior salvo porque en la variable F hemos añadido ByVal,
'que transfiere el parámetro como valor y no como variable
    Dim i As Integer
    Dim Total As Integer
    Total = 0
    For i = 1 To Q
        Total = Total + ActiveSheet.Cells(F, C).Value
        F = F + 1
    Next i
    ActiveSheet.Cells(F, C) = Total
End Sub
```

Funciones



- Las funciones no ejecutan acciones, simplemente dan como resultado un valor
- Las variables de la función se introducen como argumentos

```
Function Area_Cuadrado(x, y)
    Area_Cuadrado = x * y
End Function
```

- En la categoría de Funciones “Definidas por el usuario” encontrará esta función que podrá aplicar normalmente a la hoja de cálculo.
- También se puede usar esta función llamándola desde un procedimiento o desde otra función.

Función llamada por un Sub

- Una función puede ser llamada por un procedimiento u otra función.
- Las funciones tienen tipo (esta es de tipo integer) ya que devuelven un valor

```
Sub Llama_suma() 'Procedimiento que llama a una función de varias formas. Ver distintas formas.
    Dim x As Integer
    Dim n1 As Integer, n2 As Integer
    x = Sumardos(5, 5)
    n1 = Val(InputBox("Entrar un número : ", "Entrada"))
    n2 = Val(InputBox("Entrar otro número : ", "Entrada"))
    x = Sumardos(n1, n2)
    ActiveCell.Value = Sumardos(ActiveSheet.Range("K10").Value, ActiveSheet.Range("K11").Value)
    x = Sumardos(5, 4) + Sumardos(n1, n2)
End Sub
Function Sumardos(V1 As Integer, V2 As Integer) As Integer
    Dim Total As Integer
    Total = V1 + V2
    Sumardos = Total
End Function
```

Ejercicio

- Cree una función que calcule el factorial de un número
- Por ejemplo. Factorial(5)=5x4x3x2x1=120
- Aunque ya existe una función en Excel que calcula el factorial:
- =FACT(numero)

Función Factorial

```
Function factori(n As Long) 'FUNCIÓN que calcula el factorial de un número
    Dim F As Long
    Dim i As Long
    F = 1
    For i = n To 1 Step -1
        F = F * i
    Next
    factori = F
End Function
```

```
Function Factorial(ByVal n As Integer)
    ' Un buen ejemplo del uso de ByVal para transferir variables
    ' Si no se pusiera en este caso no calcularía bien
    n = n - 1
    If n = 0 Then
        Factorial = 1
        Exit Function
    End If
    Factorial = Factorial(n) * (n + 1)
End Function
```


Función que detecta Celda Vacía

```
Sub Detecta_Vacia()
    Dim Casilla As String
    Worksheets("Hoja4").Activate
    Casilla = Casilla_Vacia("B5") 'Llama a la función Casilla_Vacia
    MsgBox Prompt:=Casilla, Title:="La primera celda vacía"
End Sub
```

- Función Casilla_Vacia de Tipo String
- Sirve para Recorrer las filas de una columna hasta encontrar una vacía.
- Parámetros :
- Casilla_Inicio : Casilla donde debe empezar a buscar.
- Devuelve un string que contiene la referencia de la primera casilla

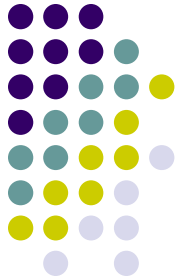
```
Function Casilla_Vacia(Casilla_Inicio As String) As String
    ActiveSheet.Range(Casilla_Inicio).Activate
    Do While Not IsEmpty(ActiveCell)
        ActiveCell.Offset(1, 0).Activate
    Loop
    Casilla_Vacia = ActiveCell.Address
End Function
```

Función que Busca un Valor

```
Sub Busca()
   Dim Casilla As String, Valor As Integer
   Worksheets("Hoja4").Activate
   Valor = CInt(InputBox("Valor buscado: ", "Entrar Datos"))
   Casilla = Buscar_Valor("C5", Valor) ‘Llama a la función Buscar_Valor
   If Casilla = "" Then ' Si valor no encontrado
      MsgBox ("NO se ha encontrado el valor buscado")
   Else 'Valor encontrado
      MsgBox ("El primer " & Valor & " esta en la celda: " & Casilla)
   End If
End Sub
```

- Función que devuelve la dirección de la primera celda vacía de un rango.
- La función es de tipo **String** ya que devuelve la casilla en la forma "FilaColumna ", por ejemplo "A10".
- Utilizaremos la propiedad **Address** del objeto range, esta propiedad devuelve un string que contiene la referencia "FilaColumna" de una casilla o rango de casillas.
- En el caso de un rango devuelve, "FilaColumna_Inicial:FilaColumna_Final", por ejemplo "A1:C10"

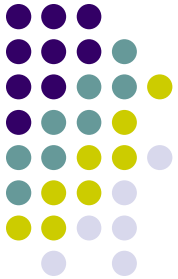
Función que Busca un Valor



- Función Buscar de Tipo String
- Sirve para: Recorrer las filas de una columna hasta encontrar el valor buscado o una de vacía.
- Parámetros :
  - Casilla_Inicial: Casilla donde debe empezar a buscar
  - Valor_Buscado: Valor que se debe encontrar
- Devuelve: Un string que contiene la referencia de la casilla donde se ha encontrado el valor
- También puede devolver "" en caso que el valor buscado no esté

```
Function Buscar_Valor(Casilla_Inicial As String, Valor_Buscado As Integer) As String
    ActiveSheet.Range(Casilla_Inicial).Activate
    ' Mientras casilla no vacía Y valor de casilla diferente al buscado
    Do While Not IsEmpty(ActiveCell) And ActiveCell.Value <> Valor_Buscado
        ActiveCell.Offset(1, 0).Activate
    Loop
    ' Si la casilla donde se ha detenido la búsqueda NO ESTÁ VACÍA es que se ha encontrado
    ' el valor
    If Not IsEmpty(ActiveCell) Then
        Buscar_Valor = ActiveCell.Address ' Devolver la casilla donde se ha encontrado el valor
    Else ' La casilla está vacía, NO se ha encontrado el valor buscado
        Buscar_Valor = "" ' Devolver una cadena vacía
    End If
End Function
```

Busca Valor por filas y columnas



- Procedimiento idéntico a Buscar() pero que llama a la función Buscar_Valor_Bis que busca por filas y columnas

```
Sub Busca_Bis()
    Dim Casilla As String
    Dim Valor As Integer
    Worksheets("Hoja4").Activate
    Valor = CInt(InputBox("Valor buscado: ", "Entrar Datos"))
    Casilla = Buscar_Valor_Bis("B5", Valor) 'Ver la función Buscar_Valor_Bis
    ' Si valor no encontrado
    If Casilla = "" Then
        MsgBox ("NO se ha encontrado el valor buscado")
    Else 'Valor encontrado
        MsgBox ("El primer " & Valor & " esta en la celda: " & Casilla)
    End If
End Sub
```

Busca Valor por filas y columnas

```
Function Buscar_Valor_Bis(Casilla_Inicial As String, Valor_Buscado As Integer) As String
    Dim Incremento_Columna As Integer
    Dim Continuar As Boolean
    ActiveSheet.Range(Casilla_Inicial).Activate
    Continuar = True
    Do While Continuar
        Incremento_Columna = 0
        ' Buscar el valor por las columnas hasta encontrarlo o encontrar una celda vacía.
        Do While Not IsEmpty(ActiveCell.Offset(0, Incremento_Columna)) And _
            ActiveCell.Offset(0, Incremento_Columna).Value <> Valor_Buscado
            ' Siguiente columna
            Incremento_Columna = Incremento_Columna + 1
        Loop
        ' Si no está vacía la casilla entonces parar la búsqueda, se ha encontrado el valor
        If Not IsEmpty(ActiveCell.Offset(0, Incremento_Columna)) Then
            Continuar = False
        Else ' La casilla está vacía, no se ha encontrado el valor
            ActiveCell.Offset(1, 0).Activate ' Saltar a una nueva fila
            If IsEmpty(ActiveCell) Then ' Si la casilla de la nueva fila está vacía
                Continuar = False ' Parar la búsqueda, no hay más casilla a recorrer
            End If
        End If
    Loop
    ' Si la casilla donde se ha detenido la búsqueda NO ESTÁ VACÍA es que se ha encontrado el valor.
    If Not IsEmpty(ActiveCell) Then
        Buscar_Valor_Bis = ActiveCell(0, Incremento_Columna).Address ' Devolver la casilla donde ' se ha encontrado el valor
    Else ' La casilla está vacía, NO se ha encontrado el valor buscado
        Buscar_Valor_Bis = "" ' Devolver una cadema vacía
    End If
End Function
```

La cláusula Private

- Puede anteponer la cláusula private a todos los procedimientos y funciones que sean llamados sólo desde el mismo módulo.
- Es una forma de ahorrar memoria y hacer que el programa corra un poco más rápido.
- Si necesita llamar un procedimiento o función desde otro módulo, nunca debe precederlo por la cláusula private

' *Módulo 1*	' *Módulo 2*
Sub General .... **End Sub** **Private Sub** Privado .... **End Sub**	**Sub** Procedimiento_de_modulo2 ‘ *Esto es correcto. Llama al procedimiento General definido en Módulo1* **Call** General *' Esto no es correcto. Llama al procedimiento Privado definido en Módulo 1, este* *' procedimiento va precedido pro la cláusula Private, por tanto sólo puede ser llamado* *' desde procedimientos de su propio módulo* **Call** Privado **End Sub**

Ejercicio

- Programe una macro que proporcione las 4!=24 combinaciones de las cuatro letras ABCD

Permutaciones de ABCD

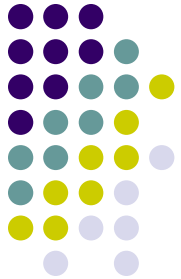
```
Sub permuta()
Dim i As Byte, j As Byte, k As Byte, l As Byte
Dim a() As Byte, mensaje As String
For i = 1 To 4
   For j = 1 To 4
      For k = 1 To 4
         For l = 1 To 4
            If i = j Or i = k Or i = l Or j = k Or j = l Or k = l Then
            Else
               mensaje = mensaje & palabra(i, j, k, l) & vbCrLf
               Exit For
            End If
         Next l
      Next k
   Next j
Next i
MsgBox mensaje
End Sub
```

```
Function palabra(i As Byte, j As Byte, k As Byte, l As Byte) As String
   Dim letra As String * 1
   Dim n As Byte, x As Byte
   Dim a(1 To 4) As Byte
   a(1) = i: a(2) = j: a(3) = k: a(4) = l
   For n = 1 To 4
      x = a(n)
      Select Case x
         Case 1: letra = "A"
         Case 2: letra = "B"
         Case 3: letra = "C"
         Case 4: letra = "D"
      End Select
      palabra = palabra & letra
   Next n
End Function
```

Permutaciones de ABCD con RND

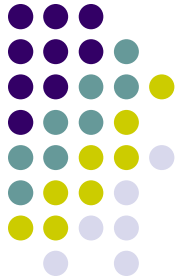
```
Sub permuta_bis()
'Permutaciones de ABCD 4!=24
Dim i As Byte, j As Byte, n As Byte
Dim a(1 To 4) As String
Dim b(1 To 24) As String
Dim frase As String
Dim x As Single
Randomize
For n = 1 To 24
    For i = 1 To 4
        x = Rnd
        Select Case x
            Case Is < 0.25: a(i) = "A"
            Case 0.25 To 0.5: a(i) = "B"
            Case 0.5 To 0.75: a(i) = "C"
            Case Is > 0.75: a(i) = "D"
        End Select
        For j = 1 To i - 1
            If a(i) = a(j) Then i = i - 1: Exit For
        Next j
    Next i
    b(n) = ""
    For i = 1 To 4
        b(n) = b(n) & a(i)
    Next i
    For j = 1 To n - 1
        If b(j) = b(n) Then n = n - 1:  Exit For
    Next j
Next n
For n = 1 To 24
    frase = frase & b(n) & vbCrLf
Next n
MsgBox frase
End Sub
```

Importar y Exportar módulos



- Ciertos procedimientos que pueden ser utilizados en multitud de ocasiones, seria interesante tenerlos disponibles en cualquiera de las hojas que confeccionemos.
- Podría pensar en utilizar las opciones de copiar y pegar para pasar
- procedimientos de una hoja a otra, es un método totalmente válido y efectivo, pero existe otro método más "profesional“.
- Consiste en guardar los procedimientos de un módulo aparte y exportarlo a un archivo **.BAS** que es independiente de cualquier hoja de cálculo.
- Luego, cuando en una nueva hoja necesite estas funciones, solo deberá importar este archivo para incorporarlo.
- Consejo: Aproveche las ventajas que proporciona la programación modular.
- Consejo: agrupe todas las funciones que usted considere de utilización general en uno o dos módulos y luego utilice las opciones de importación y exportación para incorporarlos a sus programas.

Importar y Exportar módulos



Exportar un módulo. Guardar un módulo en un archivo

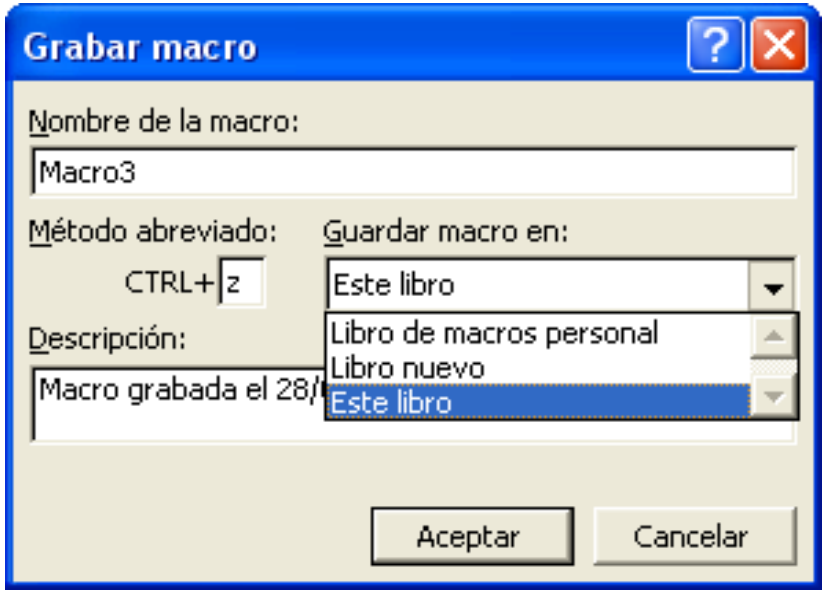
- Abra la hoja donde tiene los procedimientos que desea exportar
- 1. Pase al editor de Visual Basic y active el módulo a exportar.
- 2. Active la opción de la barra de menús **Archivo/ Exportar archivo**. Aparece un cuadro de diálogo.
- 3. En cuadro de edición **Nombre de Archivo**, teclee el nombre para el archivo donde se guardará el módulo, por ejemplo "General.Bas", observe que .BAS es la extensión de estos archivos.
- 4. Pulse sobre el botón **Guardar**.

Importar un módulo

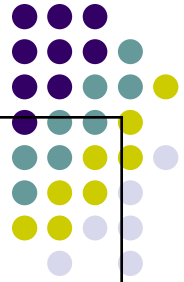
- Cierre todos los archivos de Excel y abra uno nuevo.
- 1. Active el editor Visual Basic.
- 2. Active opción de la barra de menús **Archivo/ Importar Archivo**. Aparece un cuadro de diálogo.
- 3. Seleccione en la lista **Buscar en:** la carpeta donde tiene ubicado el archivo a importar
- 4. Una vez localizada la carpeta, seleccione el archivo a importar (General.Bas en el ejemplo) y pulse sobre **Abrir**.
- Observe como en la ventana de proyecto se ha incorporado un nuevo módulo que contiene todos los procedimientos y funciones del archivo importado.

La grabadora de macros

- Microsoft Excel lleva incluida una utilidad que sirve para registrar acciones que se llevan a cabo en un libro de trabajo y registrarlas en forma de macro.
- Podemos aprovechar esta utilidad para generar código engorroso por su sintaxis un tanto complicada de recordar, además de ahorrar tiempo.
- Casi siempre después deberemos modificarlo para adaptarlo a nuestros programas

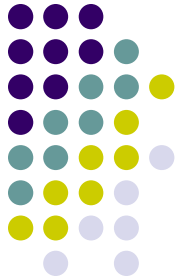


Macro realizada con Grabadora y alguna modificación



```
Sub Poner_Bordes()
    Worksheets("Hoja4").Activate
    ' Seleccionar el rango B5:D10
    Range("B5:D10").Select
    ' No hay borde diagonal hacia abajo
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    ' No hay borde diagonal hacia arriba
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    ' Borde izquierdo de la seleccón
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous 'Estilo de línea continuo
        .Weight = xlMedium ' Ancho de línea Medio
        .ColorIndex = xlAutomatic ' Color de línea automático
(negro)
    End With
    ' Borde superior de la selección
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .Weight = xlMedium
        .ColorIndex = xlAutomatic
    End With
    ' Borde inferior de la selección
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .Weight = xlMedium
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlEdgeRight) ' Borde derecho
        .LineStyle = xlContinuous
        .Weight = xlMedium
        .ColorIndex = xlAutomatic
    End With
' Bordes verticales interiores de la selección
    With Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .Weight = xlThin ' Ancho Simple.
        .ColorIndex = xlAutomatic
    End With
    ' No hay bordes horiontales interiores en la selección
    Selection.Borders(xlInsideHorizontal).LineStyle = xlNone
    ' Seleccionar rango B9:D9
    Range("B9:D9").Select
    ' No hay borde diagonal hacia arriba
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    ' No hay borde diagonal hacia arriba
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    ' Borde inferior de la selección
    With Selection.Borders(xlEdgeBottom) ' Doble línea
        .LineStyle = xlDouble
        .Weight = xlThick
        .ColorIndex = xlAutomatic
    End With
    Range("A1").Select
End Sub
```

Macro con Grabadora y generalizada

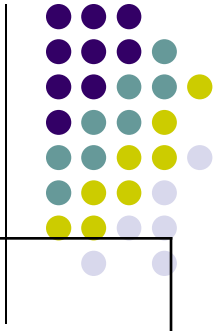


- Fichero decora.bas
- Podemos crear una Macro con la Grabadora y luego perfeccionarla y darle carácter más general introduciendo parámetros.
- Esta macro pone bordes a una tabla, en la primera fila pone doble línea.
- El primer parámetro es el número de hoja (no el nombre), y el segundo la casilla inicial
- La macro se encargará de seleccionar todas las casillas adyacentes y de buscar la primera fila.
- En esta macro además se han incluido funcionalidades como borrar los formatos antes de aplicar las líneas, ajustar el ancho de las columnas, etc.
- Observe la propiedad **CurrentRegion** del objeto **Range**, esta propiedad devuelve el rango de las casillas llenas adyacentes a una dada.
- Por ejemplo imagine una hoja con el rango A1:B10 lleno de valores, la instrucción

  ActiveSheet.Range("A1").CurrentRegion.Select

  Seleccionaria el rango correspondiente a A1:B10

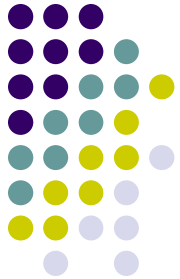
Insertar funciones en una Hoja



```
'va pidiendo números y los va colocando en las celdas de la columna B partir de B12
'al final coloca la función =SUMA para sumar los valores introducidos y la función =PROMEDIO
Sub Sumar()
    Worksheets("Hoja4").Activate
    Dim Valor As Integer
    Dim Casilla_Inicial As String
    Dim Casilla_FInal As String
    ' Hacer activa la casilla B12 de la hoja activa
    ActiveSheet.Range("B12").Activate
    Do
        ' Entrar un valor y convertirlo a numérico
        Valor = Val(InputBox("Entrar un valor", "Entrada"))
        ' Si el valor es distinto de 0
        If Valor <> 0 Then
            ' Guardar el valor en la casilla activa
            ActiveCell.Value = Valor
            ' Hacer activa la casilla de la fila siguiente
            ActiveCell.Offset(1, 0).Activate
        End If
    Loop Until Valor = 0
```

```
' Establecer la casilla inicial del rango a sumar
    Casilla_Inicial = "B12"
    ' Establecer la casilla final del rango a sumar.
    ' Coger la dirección de la casilla activa, la última
    Casilla_FInal = ActiveCell.Address
    ActiveCell.Offset(1, 0).Activate
    ' Poner en la casilla activa la función SUMA
    ActiveCell.Formula = "=Sum(" & Casilla_Inicial & ":" & Casilla_FInal & ")"
    ActiveCell.Offset(1, 0).Activate
    ' Poner en la casilla activa la función promedio
    ActiveCell.Formula = "=Average(" & Casilla_Inicial & ":" & Casilla_FInal & ")"
'Observar que las funciones se ponen en inglés y que al ejecutarse se traducen automáticamente
'Si no se conoce el nombre de una función puede usarse la grabadora
End Sub
```

Operar con Funciones de Excel



- Podemos operar con Funciones de Excel sin necesidad de insertarlas en una celda
- Vea la siguiente Calculadora Préstamos

```
Sub prestamo()
   Static Principal  ‘ Variable estática. No cambia
   Static Tasa
   Static Terminos
   Dim Pago As Double
   Principal = Application.InputBox(Prompt:="Principal (100000 por ejemplo)", _
      Default:=Principal)
   Tasa = Application.InputBox(Prompt:="Tipo de interés nominal anual (4,75 por ejemplo)", _
      Default:=Tasa)
   Terminos = Application.InputBox(Prompt:="Número de años (30 por ejemplo)", _
      Default:=Terminos)
   ‘ Vea como se usa la función de Excel Pmt (Pago) sin necesidad de calcularla en una celda
   Pago = Application.WorksheetFunction.Pmt(Tasa / 1200, Terminos * 12, Principal)
   MsgBox Prompt:="La Mensualidad es " & Format(-Pago, "Currency"), Title:="Calculadora de Préstamos"
End Sub
```

Ejercicio

- El siguiente programa calcula la longitud de una circunferencia
- Cree otros dos programas que efectúen ese mismo cálculo usando la función de Excel =PI()
  - Uno de ellos insertando la función en una celda
  - El otro sin necesidad de usar la hoja de Excel

```
Sub perimetro()
    Dim radio As Double, longitud As Double
    Const pi = 3.141592
    radio = InputBox("Introduzca el radio de la circunferencia")
    longitud = 2 * pi * radio
    ActiveCell.Value = longitud
End Sub
```

Solución Ejercicio

```
Sub perimetro2()
    Dim radio As Double, longitud As Double
    Dim pi As Double    Worksheets("Hoja1").Range("B3").Formula = "=pi()"
    pi = Range("B3").Value
    radio = InputBox("Introduzca el radio de la circunferencia")
    longitud = 2 * pi * radio
    ActiveCell.Value = longitud
End Sub
```

```
Sub perimetro3()
    Dim radio As Double, longitud As Double
    Dim pi As Double
    pi = Application.WorksheetFunction.pi()
    radio = InputBox("Introduzca el radio de la circunferencia")
    longitud = 2 * pi * radio
    MsgBox = longitud
End Sub
```

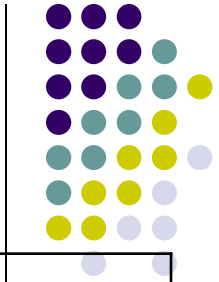
Array = Matriz

```
Sub Array1()
    'Declarar una matriz de tamaño 10
    Dim x(1 To 10) As Double
    'Calcular valores aleatorios
    For j = 1 To 10
        x(j) = Round(Rnd() * 100, 0)
        'Los valores aleatorios se calculan usando formulas VBA
    Next j
    'Transferir el contenido de la matriz a una fila
    Range(Cells(2, 2), Cells(2, 11)).FormulaArray = x
End Sub

Sub Array3()
  'Calcular valores aleatorios
  'En este caso, los valores aleatorios se calculan usando formulas Excel
  Range(Cells(4, 3), Cells(13, 3)).FormulaArray = "=Round(Rand() * 100, 0)"
  'Vea la diferencia entre Rnd (del caso anterior) y Rand.
  'Una es una fórmula VBA la otra es una fórmula Excel en inglés
End Sub
```

```
Sub Array2()      'Para una matriz verticar usando dos dimensiones
  'Declarar una matriz de 10 filas y una columna
  Dim x(1 To 10, 1 To 1) As Double
  'Calcular valores aleatorios
  For j = 1 To 10
    x(j, 1) = Round(Rnd() * 100, 0)
  Next j
  'Transferir el contenido de la matriz a una columna
  Range(Cells(4, 2), Cells(13, 2)).FormulaArray = x
End Sub
```

Horas Semanales Trabajadas

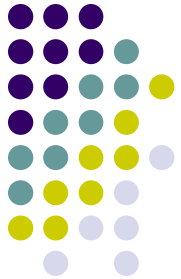


```
Sub HorasSemanales()
'Calcula las horas semanales trabajadas
Dim a(1 To 5, 1 To 2) As Single
Dim dia As String, jornada As String
Dim i As Byte, j As Byte
'Tipo byte entre 0 y 255
Dim horas As Single
For i = 1 To 5
   Select Case i
      Case 1: dia = "Lunes"
      Case 2: dia = "Martes"
      Case 3: dia = "Miercoles"
      Case 4: dia = "Jueves"
      Case 5: dia = "Viernes"
   End Select
   For j = 1 To 2
      If j = 1 Then
         jornada = "INICIO"
      Else
         jornada = "FINAL"
      End If
      a(i, j) = InputBox("Introduzca la hora de " & jornada _
      & " de la jornada de " & dia & "," & vbCrLf & _
      "en formato decimal. Por ejemplo 17:30 son las 17,5")
      If j = 2 Then horas = horas + a(i, 2) - a(i, 1)
   Next j
Next i
MsgBox "Horas semanales = " & horas
End Sub
```

Beneficio Medio de un Grupo de Empresas

```
Sub BeneficioMedio()
'Calcula el beneficio medio de un grupo de empresas
    Dim a() As Double 'Define una matriz sin decir aún la dimensión
    Dim n As Byte
    Dim i As Byte
    Dim media As Double
    n = InputBox("Número de empresas del Grupo =")
    ReDim a(n) 'Redimensiona una matriz dinámica
    For i = 1 To n
        a(i) = InputBox("Beneficio de la Empresa " & i & " = ", "La Media hasta ahora
es " & media)
        media = (media * (i - 1) + a(i)) / i
    Next
    MsgBox "Beneficio Medio del Grupo de Empresas= " & media
End Sub
```

Detección de Errores



- **Errores en tiempo de compilación.** Son los típicos errores que impiden hacer funcionar el programa debido, por ejemplo, a errores de sintaxis en las instrucciones, llamadas a funciones que no existen o llamadas con el tipo o el número de parámetros incorrectos, etc. Este tipo de errores no dan demasiados problemas, primero porque el compilador avisa de donde se han producido y luego porque simplemente revisando la sintaxis se solucionan rápidamente.
- **Errores en tiempo de ejecución**. Estos errores se producen por una mala programación del código al no haber previsto determinados casos concretos o especiales, como por ejemplo intentar abrir un archivo que no existe, imprimir sin comprobar que la impresora está conectada, definir mal la dimensión de un array e intentar acceder a miembros que no existen, etc. Cuando se produce este tipo de errores se detiene la ejecución del programa y normalmente se informa del tipo de error que se ha producido. Muchos de estos errores se pueden solucionar mediante rutinas o funciones de tratamiento de errores.
- **Errores de función**. Son los más complicados de detectar ya que ni se detectan en la fase de ejecución, ni provocan la detención del programa, son debidos a la incorrecta programación de algún proceso y como resultado se obtienen datos erróneos. Errores de este tipo son cálculos mal hechos, bucles infinitos, devolución de valores incorrectos, etc. Como ni los detecta el compilador, ni provocan la interrupción del programa deben revisarse a mano usando las herramientas de depuración.

- Consejo: modularice su programa utilizando procedimientos cortos que realicen trabajos concretos y precisos, de esta forma conseguirá, además de que el programa quede más elegante y en un futuro sea más sencillo modificarlo y depurarlo.

Ejercicio

- Cree una Macro con un juego que consiste en que la máquina piensa un número entre 0 y 100, y el jugador debe adivinarlo. Para ello, dispone de 10 tiradas, y el programa le indica si el número secreto es mayor o menor al introducido.

Solución Ejercicio: Adivina

```
Sub adivina()
   Dim zona As String
   Dim x As Byte, n As Byte
   Dim tirada As Byte
   Randomize
   x = Fix(Rnd * 101) : tirada = 1  'FIX=INT=parte entera
   Do
      If zona = "" Then
         n = InputBox("Introduzca un número entero del 0 al 100" & vbCrLf _
         & "Dispone de 10 tiradas para lograrlo", "Tirada número " & tirada)
      Else
         n = InputBox("El número secreto es " & zona & vbCrLf & "Introduzca otro", "Tirada número " & tirada)
      End If
      If n = x Then
         MsgBox "Felicidades!!!" & vbCrLf & "Ha adivinado el número secreto " & x & ", en " & tirada & " tiradas"
         Exit Sub
      End If
      If x < n Then
         zona = "Inferior"
      Else
         zona = "Superior"
      End If
      tirada = tirada + 1
   Loop Until tirada > 10
   MsgBox "Ha agotado las 10 tiradas disponibles" & vbCrLf & "El número secreto es " & x
End Sub
```

Depuración de programas

- Estas herramientas son muy útiles a la hora de comprobar paso a paso el funcionamiento del programa y detectar los procesos que provocan un mal funcionamiento del mismo.
- Importe Módulo5.bas
- Active la barra de depuración
  - (Ver/ Barras de Herramientas/ Depuración)
- Modo Ejecución paso a paso
  - Paso a Paso por Instrucciones. **F8**
  - Paso a Paso por Procedimientos. MAY+F8

    - Sirve para ejecutar todo un procedimiento. Cuando en la ejecución de un procedimiento, se llega a una línea que llama a otro procedimiento o función, pulsando este botón se puede provocar la ejecución de todo el código de esta función para luego continuar con el modo paso a paso.

- Modo Interrupción
  - En programas largos resulta fastidioso tener que ejecutarlos paso a paso, sobretodo si sabemos que el error se produce en una parte avanzada del programa. El modo interrupción, permite la ejecución del programa hasta una instrucción determinada para, a partir de esta, ejecutar paso a paso y así poder detectar el error.

  - Definir puntos de interrupción
    - 1. Sitúe el cursor sobre la instrucción en la cual debe detenerse el programa para continuar paso a paso.
    - 2. Pulse sobre el botón . También puede activar la opción **Depuración/ Alternar punto de interrupción**, pulsar la tecla **F9** o bien hacer un clic en la parte izquierda de la ventana del módulo (la franja vertical en color gris).
    - *Para desactivar un punto de interrupción siga los mismos pasos*

La Ventana de Inspección

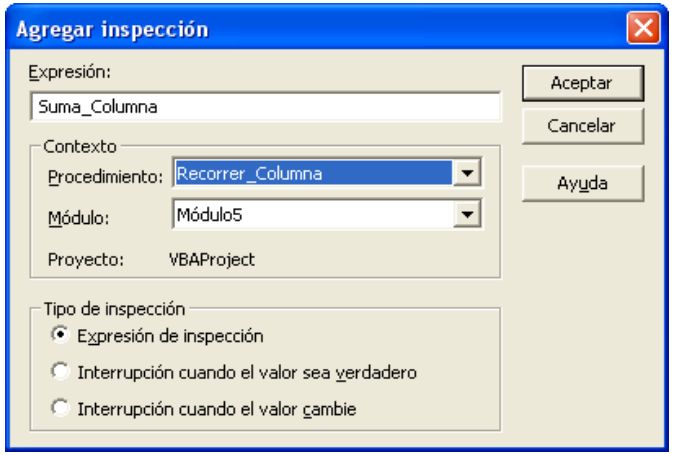
- **Inspecciones rápidas de variables**
  - Estas opciones sirven para revisar el valor de las variables a medida que se va ejecutando el programa.
  - Para ver los valores que van tomando las variables es conveniente tener visible la **Ventana de inspección**, para activarla **Ver/ Ventana de Inspección**

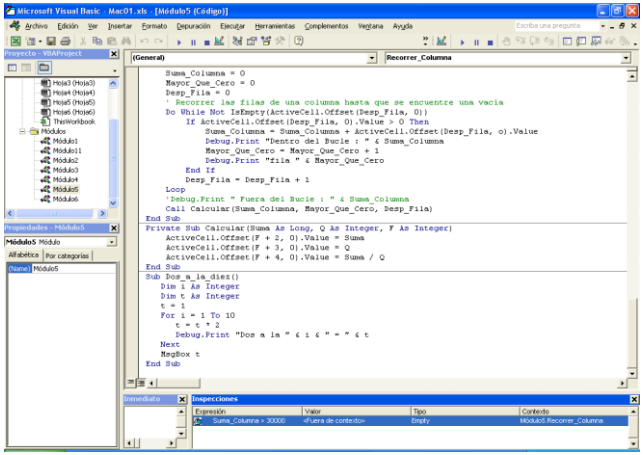
- **Añadir una variable a la ventana de inspección**
  - 1. Seleccione la variable que desee añadir a la ventana haciendo un clic sobre ella.
  - 2. Activar **Depuración/ Inspección rápida** o **MaY+F9**.
    - Aparece un cuadro de diálogo donde se muestra el valor actual de la variable. Si no está ejecutando el programa paso a paso, aparecerá el valor **Fuera de Contexto**.

  - 3. Pulse sobre el botón **Agregar** para añadir la variable a la ventana de inspección.
    - Debe tener en cuenta que para revisar las variables las expresiones que les asignan valores deben de ejecutarse al menos una vez.
- *Cuando ejecuta el programa paso a paso, si sitúa el puntero de ratón sobre una variable, se muestra el valor de la misma*
- **Borrar una variable de la ventana de Inspección**
  - Sólo debe seleccionarla en la ventana de inspección y pulsar sobre la tecla **SUPR**.
- **Modificar el valor de una variable en tiempo de ejecución**
  - A veces resulta interesante cambiar el valor de alguna variable cuando se está ejecutando el programa, para ver que ocurre si coge determinados valores, para terminar un bucle,…

Expresiones de Revisión

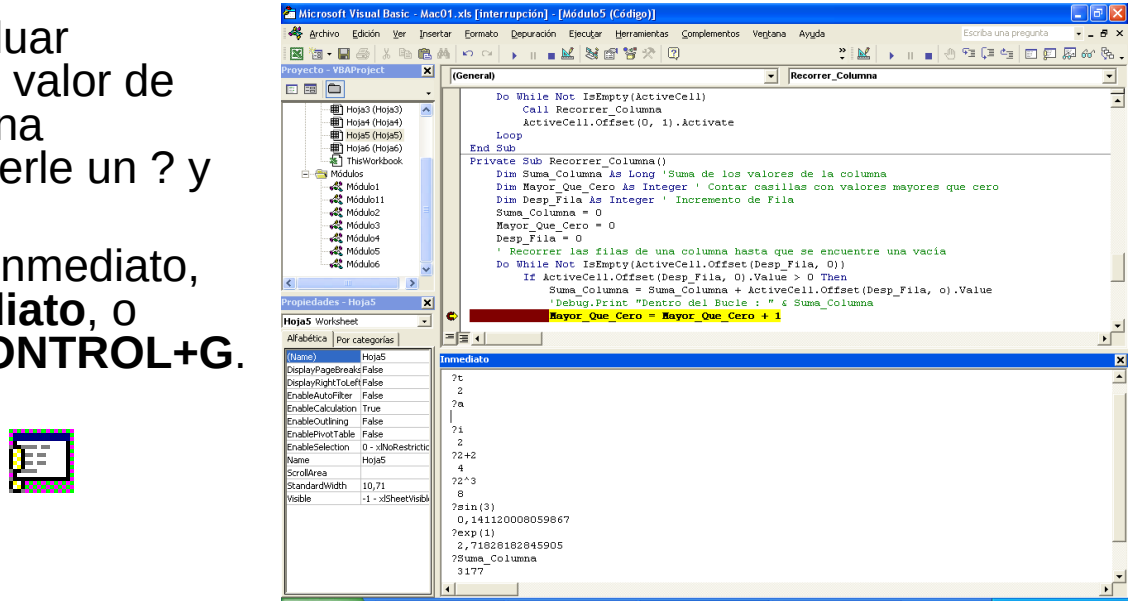
- Además de permitir añadir una variable o expresión dentro de la **Ventana Inmediato**, una **Expresión de Revisión** permite interrumpir la ejecución del programa cuando una variable coge determinado valor.
- Piense que muchas veces un programa deja de funcionar, o funciona mal cuando una variable coge determinados valores.
- Con una expresión de revisión, podremos detener la ejecución del programa cuando una variable contiene determinado valor (a partir de determinado valor), luego, podremos continuar con la ejecución paso a paso para ver que ocurre a partir de este punto.
- Sitúe el cursor sobre una variable y seleccione Agregar Inspección, Interrupción cuando el valor sea verdadero, y luego en la ventana de inspecciones, editar la variable y añadirla una condición lógica que al cumplirse parará el procedimiento.





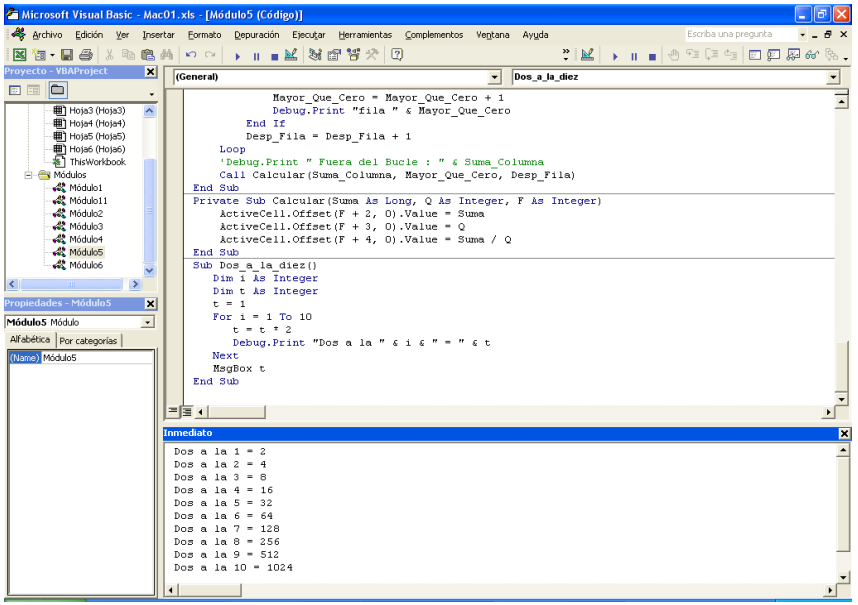
La Ventana Inmediato

- Es otra forma de inspeccionar variables cuando el programa está en modo interrupción (ejecutándose paso a paso)
- Además, ofrece la posibilidad de cambiar valores de las variables
- E incluso ejecutar o evaluar expresiones. Para ver el valor de una variable en la ventana inmediato debe anteponerle un ? y luego pulsar **Enter**.
- Para activar la ventana Inmediato, active opción **Ver/Inmediato**, o pulse la combinación **CONTROL+G**.
- Pruebe
  - ?2+3
  - ?2^3
  - ?exp(1)
  - ?Suma_Columna



La instrucción Debug.Print

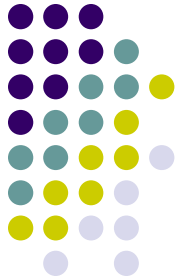
- Esta instrucción se utiliza directamente sobre el código del programa
- Permite ver todos los valores que ha ido tomando una variable o expresión durante la ejecución del programa.
- Los valores se mostrarán en la ventana Inmediato una vez finalizado el programa.
- Esta expresión resulta útil en una fase avanzada de depuración ya que permite ir viendo la evolución de una variable o expresión sin necesidad de poner puntos de interrupción.
- Cuando el programa esté listo deben eliminarse.
- Ejecute Dos_a_la_diez()



Formularios

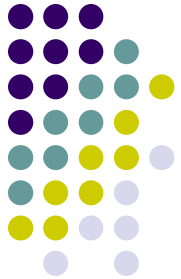
- Mostrar la barra de herramientas para cuadros de control
- Cuadro de Texto
- Etiqueta
- Botón de Comando
- Modo Diseño
- Propiedades
  - En la propiedad **Caption**, cambien el texto ***Label1*** por ***Datos a Buscar***

Los eventos



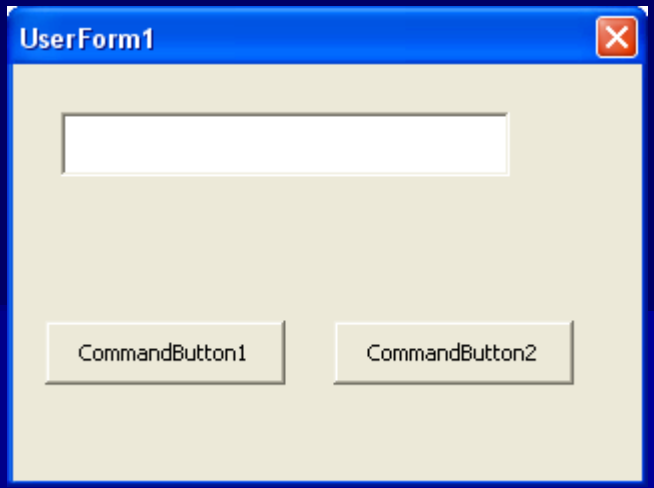
- Cuando se programan controles bien sea directamente en la hoja o desde un formulario, debe tener en cuenta los eventos.
- Un evento se da cuando ocurre algo sobre un objeto
- En entornos Windows constantemente se están produciendo eventos que son recogidos por el sistema.
  - Clicks con el ratón sobre un control
  - Teclear sobre un cuadro de texto, etc.
- Programar un evento significa hacer que se ejecuten determinadas instrucciones cuando ocurra dicho evento.
- En general, todos los controles son capaces de capturar diferentes eventos.

Cuadros Combinados (ComboBox)



- Con un ComboBox podremos escoger el campo, es decir, podremos extraer coincidencias de *Nombre*, *Apellidos*, la *Ciudad*, etc.
- Para ello incluiremos un cuadro combinado que permita escoger en que campo o columna tiene que buscarse la coincidencia.
- La lista, por supuesto, mostrará los nombres de las columnas.

Formularios y Controles



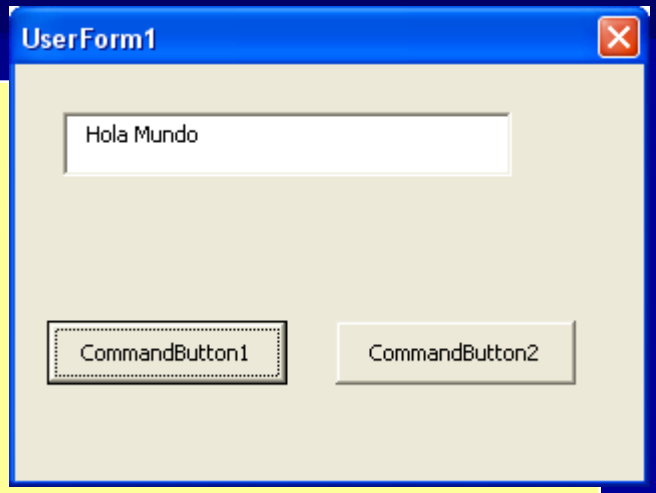
- Cree un nuevo libro (mundo.xls)
- Acceda al Editor de Visual Basic
- Menú, Insertar, UserForm
- En el formulario que aparece UserForm1 se insertarán los controles del Cuadro de Herramientas que también ha aparecido. Si no aparece haga clic en el icono "Cuadro de Herramientas".
- Insertemos Controles
- Clic en el "Cuadro de Texto" del cuadro de herramientas
- Marcar un recuadro en el UserForm1
- Clic en el "Botón de Comando"
- Marcar un pequeño recuadro en el UserForm1
- Inserte un segundo Botón de Comando

Hola Mundo

- Seleccione el CommandButton1 y vea las propiedades. Si no aparecen pulse el icono Ventana de Propiedades
- En la propiedad Caption escriba Saludo
- En la propiedad Caption del CommandButton2 escriba Borrar
- Haga clic en el icono Ver Código de la Ventana de Proyecto
- Observe que aparece un área para introducir e código de los eventos asociados al formulario
- Arriba aparecen dos desplegables. El de la izquierda contiene los Objetos y el de la derecha los Procedimientos
- Seleccione del desplegable el objeto CommandButton1.
- En el otro desplegable aparece automáticamente Click
- Escribe el siguiente código

Código para el UserForm

```
Private Sub CommandButton1_Click()
    TextBox1.Text = "Hola Mundo"
End Sub
Private Sub CommandButton2_Click()
    TextBox1.Text = ""
End Sub
```



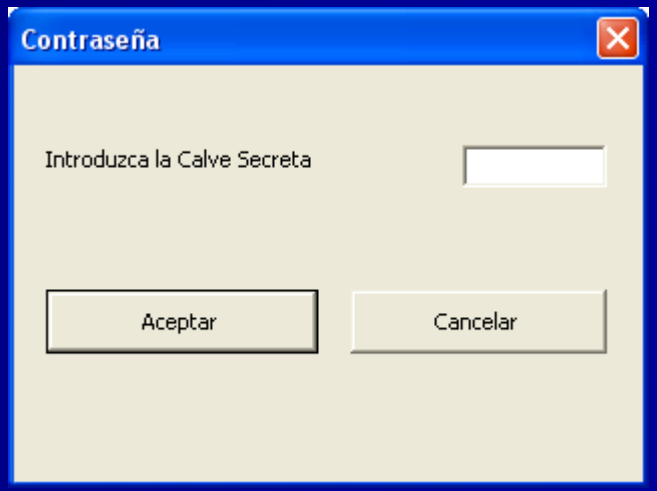
- Vuelva al formulario. Basta cerrar la ventna, o mejor hacer clic en el icono Ver Objeto
- Grabar
- Ejecutar el programa
- Son procedimientos de evento

Programar User Forms

- El procedimiento a seguir es:
  1. Menú Insertar UserForm
     - Cuadro de Herramientas
  2. Establecer Propiedades de los objetos
  3. Escribir el código

Contraseña

- Insertar un nuevo formulario. UserForm2
- Propiedades del UserForm2
  - Name = frmClave
  - Caption = Contraseña
- Insertar una Etiqueta (label)
  - Name =lblClave
  - Caption = Introduzca la clave secreta
- Inserte un Cuadro de Texto (TextBox)
  - Name = txtContraseña
  - MaxLength = 6
  - PasswordChar = *
- Inserte el CommandBooton1
  - Name = cdmAceptar
  - Default = True
  - Caption = Aceptar
- Inserte el CommandBooton2
  - Name = cdmCancelar
  - Caption = Cancelar
  - Cancel = True



- Si un botón se pone como Dafault = True, automáticamente los demás se ponen Default = False
- En un formulario solo puede haber un botón con la propiedad Cancel = True

Código del UserForm Contraseña

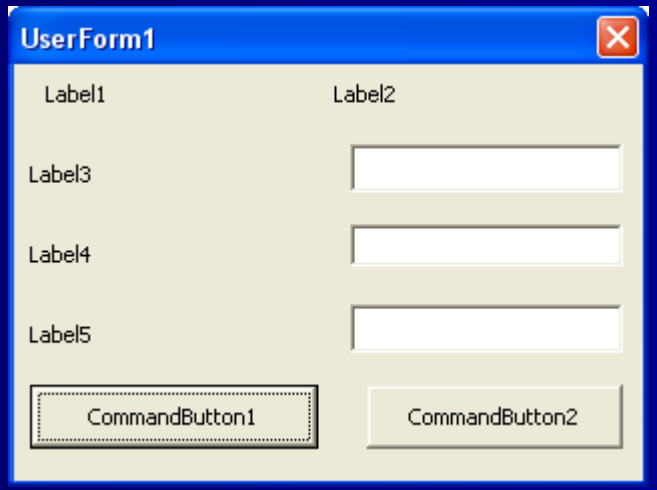
- Escriba los siguientes procedimientos de evento
- Y ejecute el formulario para ver el funcionamiento

```
Private Sub cmdAceptar_Click()
    If UCase(txtContraseña.Text) <> "MACROS" Then
        MsgBox "Contraseña Incorrecta", vbCritical
        End
    Else
        MsgBox "Contraseña Aceptada", vbExclamation
        End
    End If
End Sub

Private Sub cmdCancelar_Click()
    End
End Sub
```

Formulario Fechas

- Nuevo Libro Fechas.xls
- Abrir el Editor de Visual Basic
- Insertar un UserForm
- Incrustar los controles de la imagen



Propiedades

- UserForm1
  - Name = frmFechas
  - Caption = Fechas
- Label1
  - Name = lblEtiq1
  - Caption = Hoy es
- Label2
  - Name = lblHoy
- Label3
  - Name = lblEtiq2
  - Caption = Primero del mes que viene
- Label4
  - Name = lblEtiq3
  - Caption = Escribe una fecha
- TextBox1
  - Name = txtPrimeroMesViene
- TextBox2
  - Name = txtFecha
- Label5
  - Name = lblEtiq4
  - Caption = 1º Mes Siguiente
- TextBox3
  - Name = txtSiguiente
- CommanButton1
  - Name = cmdCalcular
  - Caption = Calcular
- CommandButton2
  - Name = cmdOtra
  - Caption = Otra

Procedimientos de Evento

```
Private Sub UserForm_Activate()
    lblHoy.Caption = Date
    txtPrimeroMesViene.Text = PrimeroMes
    txtFecha.Text = Date
    txtFecha.SetFocus
End Sub
```

```
Private Sub cdmCalcula_Click()
    txtSiguiente.Text = PrimeroMesCualquiera(txtFecha.Text)
End Sub
Private Sub cmdOtra_Click()
    txtFecha.Text = ""
    txtSiguiente.Text = ""
    txtFecha.SetFocus
End Sub
```

Modulo1

```
Function PrimeroMes()
    PrimeroMes = DateSerial(Year(Now), Month(Now) + 1, 1)
End Function
Function PrimeroMesCualquiera(Cual As Date) As Date
    PrimeroMesCualquiera = DateSerial(Year(Cual), Month(Cual) + 1, 1)
End Function
```

Pasar una Matriz a una Función

```
Sub SumaCien()
    Dim i As Byte
    Dim a(100) As Byte
    'Genera una matriz de números aleatorios enteros entre 0 y 100
    Randomize
    'Si no se pone Randomize los valores aleatorios siempre son los mismos
    'Pruebelo. Abra el libro y vuelva a lanzar la macro. Los resultado serán los mismos.
    For i = 1 To 100
        a(i) = Int(Rnd * 101)
    Next i
    MsgBox "Suma de 100 números aleatorios" & vbCrLf & _
    "enteros entre 0 y 100" & vbCrLf & vbCrLf & Chr(9) & calcula(a())
End Sub
```

```
Function calcula(a() As Byte) As Single
    Dim i As Byte
    Dim s As Single
    'Calcula la suma de los 100 números que contiene la matriz
    For i = 1 To 100
        s = s + a(i)
    Next i
    calcula = s
End Function
```

Consulta News

- **Asunto:** Consulta como buscar ultimo
- Quiero averiguar como se hace una búsqueda de un ultimo registro dentro de un rango determinado. Ej. Tengo un rango de A1:A10 en donde hay datos desde A1 hasta A6. En la celda A15 quiero obtener el valor o dato que haya en la última celda ocupada del rango A1:A10, es decir que me escriba en este caso lo que hay en A6.

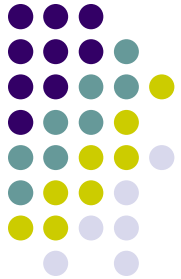
Solución

```
Sub UltimoValor()
    Worksheets("Hoja1").Activate 'Aqui le dices la hoja
    ActiveSheet.Range("A10").Activate 'Aqui le dices el final del rango
    Do While IsEmpty(ActiveCell)
        ActiveCell.Offset(-1, 0).Activate
    Loop
    Range("A15").Value = ActiveCell.Value
End Sub
```

- Se pude hacer con una formula, pero si tiene celdas vacías intermedias no funciona

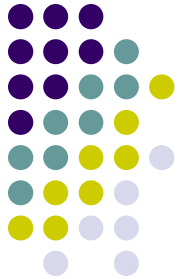
  =INDIRECTO(CONCATENAR("A";CONTAR(A1:A10)))

Pregunta Color



- **Asunto:** codigo para contar celdas de X
- alguien me puede pasar el codigo - formula para contar el numero de celdas que tienen X's color (trama) saludos!!

Respuesta Color 1



- Primero vamos a colorear unas cuantas celdas del rango D1:D30. Para ello ejecuta el siguiente código:

```
Sub colorea()
   Dim Celda
   Dim R As Range
   Set R = Range("D1.D30")
   R.Select
   For Each Celda In R
      Celda.Interior.ColorIndex = Int(Rnd * 10) + 1
   Next

End Sub
```

- En el código anterior hemos pedido que nos coloree con un máximo de 10 colores, aunque sabemos que existen 56 colores distintos.

Respuesta Color 2

- En segundo lugar ejecuta esta macro:

```
Sub pru()
   MsgBox "Celdas de color Rojo (3): " & CuentaColor(Range("D6.D30"), 3)
End Sub
```

- Se basa en la función CuentaColor, que cuenta el color rojo (# 3)

```
Function CuentaColor(R As Range, tono As Byte) As Byte
   Dim num As Long
   Dim Celda
   For Each Celda In R
      If Celda.Interior.ColorIndex = 3 Then num = num + 1
   Next
   CuentaColor = num
End Function
```

- Ver fichero CeldaColor.xls