Comencemos a programar con

VBA - Access

Entrega 13

Funciones de VBA

Eduardo Olaz

Funciones propias de VBA

VBA incluye un gran número de Funciones y Procedimientos propios que nos simplificarán las tareas de programación.

Funciones para la toma de decisiones

Ya hemos visto en la entrega 9 la estructura de decisión **If – Then – Else – EndIf**, la función **IIF**. Y la estructura **Select – Case – End Select**.

VBA posee funciones adicionales que nos servirán para la toma de decisiones en el código.

Función Choose

La función **Choose**, selecciona y devuelve un valor de entre una lista de argumentos.

Sintaxis

Choose(*índice, opción-1*[, *opción-2, ...* [, *opción-n*]]**)**

Choose busca el argumento situado en la posición definida por el índice.

Si índice fuese menor que **1**, ó mayor que el total de argumentos, devolvería el valor **Null**.

Si *índice* fuese un número no entero, lo redondearía al entero más próximo.

En este ejemplo hemos definido, como constantes enumeradas los posibles puestos de una hipotética empresa.

Las constantes enumeradas contienen un valor long, por lo que vamos a desarrollar una función que nos devuelva, en base a su valor, la descripción literal del cargo.

```
Public Enum Puesto
    eoEmpleado = 1
    eoTecnico
    eoAdministrativo
    eoMandoIntermedio
    eoComercial
    eoDirectivo
    eoGerente
End Enum

Public Function PuestoDeTrabajo( _
                    ByVal Cargo As Puesto _
                    ) As String
    ' Comprobamos si Cargo está en un rango válido
    If Cargo < eoEmpleado Or Cargo > eoGerente Then
        PuestoDeTrabajo = ""
    Else
        PuestoDeTrabajo = Choose(Cargo, _
                          "Empleado", _
                          "Técnico", _
                          "Administrativo", _
                          "Mando Intermedio", _
```

```
                                    "Comercial", _
                                    "Directivo", _
                                    "Gerente")
        End If
    End Function
```

Si escribimos

```
PuestoDeTrabajo(eoMandoIntermedio)
```

Nos devolverá

```
Mando Intermedio
```

Los argumentos de la función **Choose**, en realidad son un **ParamArray**, como el que vimos en la entrega anterior.

Función Switch

La función **Switch**, evalúa una lista de expresiones y devuelve un valor, o una expresión asociada, a la primera expresión de la lista que sea cierta.

Su sintaxis es

Switch(*expresión-1, valor-1*[*, expresión-2, valor-2 … [, expresión-n,valor-n*]]**)**

Supongamos que queremos hacer una función que recibiendo como parámetro el nombre de un país nos devuelva su capital.

```
Public Function Capital(ByVal Pais As String) As String
    Dim varPais As Variant
    ' Uso un variant porque si Switch no encuentra _
      una expresión cierta, devuelve Null
    Pais = Trim(Pais)
    varPais = Switch(Pais = "España", "Madrid", _
                     Pais = "Francia", "París", _
                     Pais = "Portugal", "Lisboa", _
                     Pais = "Italia", "Roma", _
            Len(Pais) = 0, "Tienes que introducir algo", _
            IsNumeric(Pais), "Los números no valen")
    If IsNull(varPais) Then
        Capital = "No conozco la capital de " & Pais
    Else
        Capital = varPais
    End If
End Function
```

En cada caso devolverá:

```
Capital("Italia")   → "Roma"
Capital(3.1416)   → "Los números no valen"
Capital("Japón ")   → " No conozco la capital de Japón"
```

```
Capital("    ")  →  "Tienes que introducir algo"
```

Como puede comprobarse en la función anterior, **Switch** permite un tipo de código simple que posibilita evaluar expresiones de diferentes tipos .

No obstante, para las funciones **Choose**, **Switch** y la función **IIF** que vimos en la entrega 09 hay que tener en cuenta que su velocidad de ejecución es inferior a la de **If – Then** o **Select – Case**. Por ello, en aquellos procedimientos en los que el tiempo de ejecución puede ser crítico, es mejor no usarlas.

Función Format

La función format recibe una expresión que contiene una cadena, un valor numérico, una fecha, un valor boleano ó un valor nulo, y devuelve una cadena formateada de acuerdo a las instrucciones contenidas en la cadena formato.

Su sintaxis es

```
Format(expresión[, formato[, PrimerDíaDeLaSemana[,
PrimeraSemanaDelAño]]])
```

Utilización con cadenas String.

Aunque en VBA existen otras funciones más potentes para el manejo de cadenas, funciones que veremos próximamente, **Format** permite algunos tipos de formateo.

```
Format("Cadena a mayúsculas", ">") "CADENA A MAYÚSCULAS"
```

Si usamos como cadena de formato "**<**" devuelve la cadena convertida a minúsculas.

```
Format("Cadena a MINÚSCULAS", "<") " cadena a minúsculas "
```

Si usamos @ `Format("Cadena", "@@@@@@@@@@@@@")`

Si en la posición donde está el signo @ hay un carácter, se presentará éste, en caso contrario pondrá un espacio en blanco. La cadena se mostrará alineada a la derecha

`Format("Eduardo", "@@@@@@@@@@@@@")` Devolverá

Eduardo He puesto el subrayado para mostrar los espacios en blanco.

Si ponemos delante de la cadena de formato el carácter ! hará que se alinee a la izquierda.

Otro carácter que podemos utilizar es &. Se diferencia de @ en que si en la posición donde está el signo & hay un carácter, se presentará éste, en caso contrario no pondrá nada.

Si la cadena con & es precedida del carácter "!" recortará la parte izquierda de la cadena.

Si ejecutamos el procedimiento PruebaFormat

```
Public Sub PruebaFormat()
   Debug.Print Format("Esta cadena a mayúsculas", ">")
   Debug.Print Format ("ESTA CADENA A minúsculas", "<")
   Debug.Print "#" & Format("Derecha", "@@@@@@@@@@@@@") & "#"
   Debug.Print "#" & Format("Izquierda", "!@@@@@@@@@@@@@") & "#"
   Debug.Print "#" & Format("Cadena Sin Cortar", "&&&&&") & "#"
   Debug.Print "#" & Format("Cadena Sin Cortar", "@@@@@") & "#"
   Debug.Print "#" & Format("Cadena Cortada", "!" & "&&&&&&&") & "#"
   Debug.Print "#" & Format("Eduardo", "@ @ @ @ @ @ @ @ @") & "#"
   Debug.Print "#" & Format("Eduardo", "!@ @ @ @ @ @ @ @ @") & "#"
```

```
End Sub
```

Nos mostrará en la ventana Inmediato

```
ESTA CADENA A MAYÚSCULAS
esta cadena a minúsculas
#     Derecha#
#Izquierda   #
#Cadena Sin Cortar#
#Cadena Sin Cortar#
#Cortada#
#    E d u a r d o#
#E d u a r d o    #
```

Utilización con fechas.

Format está especialmente preparada para manejar valores numéricos y en concreto el formato de fechas.

Como cadena de formato admite tanto una cadena de caracteres como la llamadas cadenas con nombre.

Es precisamente para el formateo de fechas donde adquieren sentido los dos últimos parámetros opcionales de la función **Format**.

Recordemos su sintaxis:

```
Format(expresión[, formato[, PrimerDíaDeLaSemana[,
PrimeraSemanaDelAño]]])
```

El parámetro *PrimerDíaDeLaSemana* sirve para indicar a la función Format qué día de la semana vamos a considerar como el primero.

A diferencia de España, en Estados unidos se considera el Domingo como el primer día de la semana, y ésta es la opción por defecto.

En España se considera como primer día de la semana el lunes.

Se puede introducir, como primer día de la semana, cualquiera de estos valores

Constante	Valor	Significado
vbUseSystem	0	Primer día definido por el sistema
vbSunday	1	Domingo (Valor por defecto)
vbMonday	2	Lunes
vbTuesday	3	Martes
vbWednesday	4	Miércoles
vbThursday	5	Jueves
vbFriday	6	Viernes
vbSaturday	7	Sábado

El definir cuál es el primer día de la semana, puede afectar al último parámetro *PrimeraSemanaDelAño*.

VBA tiene varias formas de considerar cuál es la primera semana del año.

En concreto puede tomar 4 criterios diferentes, configurables mediante 4 constantes.

Constante	Valor	Significado
vbUseSystem	0	Primer semana definida por el sistema
vbFirstJan1	1	Será la que contenga al día 1 de enero
vbFirstFourDays	2	La primera que contenga al menos 4 días.
vbFirstFullWeek	3	La primera que contenga los 7 días

La configuración de estos dos parámetros afecta al resultado de **Format**, si queremos mostrar el número de semana de una fecha, mediante la cadena de formato **"ww"**.

```
format(#12/30/05#,"ww",vbSunday,vbFirstJan1)     → 53
format(#12/30/05#,"ww", vbMonday,vbFirstFullWeek) → 52
format(#12/30/05#,"ww",vbSunday, vbFirstFourDays) → 52
```

Podemos combinar cadenas de formato. Por ejemplo si quisiéramos obtener el número de semana de una fecha poniéndole delante el año con cuatro cifras.

```
format(#12/30/05#," yyyyww",vbSunday, vbFirstFourDays) → 200552
format(#12/30/05#,"yyyy-ww",vbSunday, vbFirstFourDays)→2005-52
```

Los caracteres que podemos poner para formatear una fecha son

Resultados de aplicar Format **Format(#2/8/05 9:5:3#,Cadena)**

Fecha/Hora 9 horas 5 minutos 3 segundos del 8 de febrero de 2005

Cadena	Resultado	Descripción
"d"	"8"	Día del mes con 1 ó 2 dígitos
"dd"	"08"	Día del mes con 2 dígitos
"w"	vbMonday:"2", por defecto:"3"	Número de día de la semana
"ddd"	"mar"	Nombre del día con 3 caracteres
"dddd"	"martes"	Nombre del día de la semana
"ddddd"	"08/02/2005"	Fecha corta
"dddddd"	"martes 8 de febrero de 2005"	Fecha larga
"m" ó "mm"	"2" ó "02"	Número del mes con 1 ó 2 dígitos
"mmm"	"feb"	Nombre del mes con 3 caracteres
"mmmm"	"febrero"	Nombre del mes
"q"	"1"	Trimestre del año
"y"	"39"	Día del año
"yy"	"05"	Año con 2 dígitos

"yyyy"	"2005"	Año con 4 dígitos
"h" ó "hh"	"9" ó "09"	Hora con 1 ó 2 dígitos
"n" ó "nn"	"5" ó "05"	Minutos con 1 ó 2 dígitos
"s" ó "ss"	"3" ó "03"	Segundos con 1 ó 2 dígitos
"ww"	vbMonday vbFirstFourDays "6" (Por defecto da "7")	Número de semana del año
"h/n/s"	"9/5/3"	
"h:nn:ss"	"9:05:03"	

Si quisiéramos mostrar la hora en formato de 12 horas, seguido de **AM** ó **PM**, **A** ó **P**

```
Format(#2/8/05 19:5:3#,"h:nn:ss AM/PM")   → 7:05:03 PM
Format(#2/8/05 19:5:3#,"h:nn:ss A/P")     → 7:05:03 P
```

Podemos poner en la cadena de formato texto que queramos aparezca en el resultado final.

Para que no nos den problemas los caracteres que se pueden usar como formato y los interprete como texto se les pone la barra invertida delante.

```
Format(Now,"Pa\mplo\na, a d \de mmmm \de yyyy")
```

Nos devolverá algo así como:

```
Pamplona, a 18 de febrero de 2005
```

En cambio, si hubiéramos escrito

```
Format(Now,"Pamplona, a d de mmmm de yyyy")
```

Nos hubiera devuelto algo tan incoherente como

```
Pa2plo57a, a 18 18e febrero 18e 2005
```

Esto es así porque interpreta que las **"m"** son para poner el mes, las **"d"** para poner el día y las **"n"** están para indicar dónde poner los segundos.

Cadenas con nombre

Las cadenas **Con Nombre** son unas cadenas predefinidas en VBA, que nos ahorran el construir la secuencia de caracteres en la cadena de formato.

Tienen la ventaja de que son genéricas y se adaptan a la configuración regional de Windows que tenga el usuario, con lo que las podemos usar para formatos internacionales.

Existen tanto para el manejo de fechas como de números.

Los resultados siguientes son propios de la configuración de mi ordenador.

En otros ordenadores el resultado podría cambiar.

Resultado para diversas cadenas de: **Format(#2/8/05 9:5:3#,CadenaConNombre)**

Cadena con Nombre	Resultado
"General Number"	"38391,3785069444"
"Standard"	"38.391,38"
"General Date"	"08/02/2005 9:05:03"

"Long Date"	"martes 8 de febrero de 2005"
"Medium Date"	"08-feb-05"
"Short Date"	"08/02/2005"
"Long Time"	"9:05:03"
"Medium Time"	"09:05 AM"
"Short Time"	"09:05"

El procedimiento **PruebaFormatoFechas**, nos muestra varias maneras de dar formato a una fecha, utilizando las cadenas de formato propuestas en los puntos anteriores.

Así mismo muestra diversas Cadenas con Nombre y sus equivalentes en cadenas de formato con texto.

```
Public Sub PruebaFormatoFechas()
    Dim datFecha As Date
    Dim strFecha As String

    datFecha = #9/3/1953 3:34:20 PM#
    Debug.Print "***** Modo 1 *****"
    strFecha = Format(datFecha, "Medium Date")
    Debug.Print strFecha
    strFecha = Format(datFecha, "dd-mmm-yy")
    Debug.Print strFecha

    Debug.Print "***** Modo 2 *****"
    strFecha = Format(datFecha, "Short Date")
    Debug.Print strFecha
    strFecha = Format(datFecha, "dd/mm/yyyy")
    Debug.Print strFecha

    Debug.Print "***** Modo 3 *****"
    strFecha = Format(datFecha, "Long Date")
    Debug.Print strFecha
    strFecha = Format(datFecha, "dddd d ") _
               & "de " _
               & Format(datFecha, "mmmm ") _
               & "de " _
               & Format(datFecha, "yyyy")
    Debug.Print strFecha
    strFecha = Format(datFecha, "ddddd")
    Debug.Print strFecha

    Debug.Print "***** Modo 4 *****"
```

```
    strFecha = Format(datFecha, "General Date")
    Debug.Print strFecha
     strFecha = Format(datFecha, "ddddd hh:nn:ss")
    Debug.Print strFecha
    strFecha = Format(datFecha, "dd/mm/yyyy hh:nn:ss")
    Debug.Print strFecha
End Sub
```

El resultado de este código es

```
***** Modo 1 *****
03-sep-53
03-sep-53
***** Modo 2 *****
03/09/1953
03/09/1953
***** Modo 3 *****
jueves 3 de septiembre de 1953
jueves 3 de septiembre de 1953
03/09/1953
***** Modo 4 *****
03/09/1953 15:34:20
03/09/1953 15:34:20
03/09/1953 15:34:20
```

Utilización de Format con números.

La utilización de Format con números, nos concede una gran flexibilidad a la hora de presentar los datos numéricos con el formato más adecuado a cada caso.

Tanto si usamos números, como en el caso de las cadenas de texto, no necesitaremos los parámetros para definir el primer día de la semana ni la primera semana del año.

Por ello su sintaxis quedará así:

Format(*expresión*[, *formato*])

Si ejecutamos directamente la orden Format sobre un número, nos lo convertirá simplemente a una cadena cambiando, el punto decimal por el carácter definido en la configuración regional.

```
Format(3.141592) → "3,141592"
```

Con los números podemos definir si queremos utilizar separadores de miles.

Al contrario que en el formato español, el separador de miles, en la cadena de formato, es la coma y el separador de los enteros con los decimales es el punto.

```
Format(1234567890,"#,###") → " 1.234.567.890"
Format(3.141592,"#,###") → " 3"
Format(3.141592,"#,###") → " 3"
Format(3.141592,"#,###.##") → " 3.14"
```

```
Format(1234567890,"#,###") → " 1.234.567.890,"
Format(1234567890,"#,###.##") → "1.234.567.890,"
Format(1234567890,"#,###.00") → "1.234.567.890,00"
Format(3.141592654,"#,###.0000") → " 3,1416"
Format(0.008,"#,###.0000") → ",0080"
Format(0.008,"#,##0.0000") → "0,0080"
Format(023.00800,"#,###.##") → "23,01"
```

El símbolo **#** representa un dígito cualquiera, salvo los ceros no significativos (los que están a la izquierda de la parte entera y a la derecha de la parte decimal).

El símbolo **0** muestra un dígito o un cero. Si el número a formatear tiene en la posición del cero de la cadena de formato alguna cifra, se mostrará ésta. Si no hubiera ninguna mostrará un cero.

```
Format(23.1,"0,000.00") → "0.023,10"
Format(3.14159,"#,##0.0000") → "3,1416"
```

Si el número tuviera más cifras significativas a la izquierda del punto decimal que caracteres de formato la cadena de formato, se mostrarán todos los caracteres del número.

Si el número tuviera más cifras significativas a la derecha del separador de decimales, que caracteres la cadena de formato, redondeará los decimales al mismo número que los caracteres de formato.

Se pueden además incluir caracteres como el signo menos, el signo más, el signo del dólar, el signo de tanto por ciento, los paréntesis y el espacio en blanco.

El signo de % nos devuelve el Tanto por Ciento de la cantidad formateada.

```
Format(0.08120,"0.00 %") → "8,12 %"
```

Para dar formateo tipo científico se puede usar la cadena de formato `"0.00E+00"`.

```
Format(0.08120,"0.00E+00") → "8,12E-02"
```

Como en el caso de las cadenas podemos utilizar @ con ó sin el signo de exclamación

```
Format(1234567890,"@ @@@ @@@ @@@") → "1 234 567 890"
Format(12345,"@ @@@ @@@ @@@")      → "       12 345"
Format(12345,"!@ @@@ @@@ @@@")     → "1 234 5      "
```

Cadenas con nombre para números

Al igual que con las fechas, podemos aplicar a los números formatos con nombre.

Resultado para diversas cadenas de: **`Format(1234567.8901,CadenaConNombre)`**

Cadena con Nombre	Resultado
"General Number"	"1234567,89"
"Currency"	"1.234.567,89 € "
"Fixed"	" 1234567,89"
"Standard"	"1.234.567,89 "

"Percent"	"123456789,01%"		
"Scientific"	"1,23E+06"		
"Yes/No"	"Sí"	Si Cero	"No"
"True/False"	"Verdadero"	Si Cero	"Falso"
"On/Off"	"Activado"	Si Cero	"Desactivado"

Formato compuesto

Como los cuadros de texto de los formularios de Access, Format admite cadenas de formato compuesto, es decir formato diferente para distintos tipos de valores.

Estas cadenas pueden constar de 4 secciones, separadas cada una de ellas con un punto y coma.

En el caso de los números, la primera define cómo se mostrarán los valores Positivos, la segunda los negativos, la tercera para el cero y la cuarta para el valor nulo.

```
Format(3.14,"+ #,##0.00;(#,##0.00);Cero pelotero;Valor Nulo")
```

Devuelve "+ 3,14"

```
Format(-28,"+ #,##0.00;(#,##0.00);Cero pelotero;Valor Nulo")
```

Devuelve "(28,00)"

```
Format(0,"+ #,##0.00;(#,##0.00);Cero pelotero;Valor Nulo")
```

Devuelve "Cero pelotero"

```
Format(Null,"+ #,##0.00;(#,##0.00);Cero pelotero;Valor Nulo")
```

Devuelve " Valor Nulo"

Si alguna de las secciones no estuviera definida, se utilizaría para ella el formato positivo.

```
Format(-28, "#,##0.00 €;;Grati\s;Valor Nulo")
```

Devuelve "-28,00 €"

```
Format(0, "#,##0.00 €;;Grati\s;Valor Nulo")
```

Devuelve "Gratis"

Configuración Regional

La función **Format** nos permite averiguar cuál es la Configuración Regional en algunas de sus propiedades sin tener, por ejemplo, que acceder al registro de Windows.

Si ejecutamos el procedimiento **Configuración** nos mostrará algunos elementos de la Configuración Regional.

```
Public Sub Configuracion()
    Dim datPrueba As Date
    Dim sngDato As Single
    Dim strFormato As String
    Dim strSeparador As String
    Dim strDato As String
    Dim strSeparadorMiles As String
    Dim strSeparadorDecimal As String
```

```
    datPrueba = #12/31/2000#
    strDato = Format(datPrueba, "Short Date")
    strSeparador = Mid(strDato, 3, 1)
    If Left(strDato, 2) = "31" Then
        strFormato = "dd" & strSeparador _
                   & "mm" & strSeparador & "yy"
    Else
        strFormato = "mm" & strSeparador _
                   & "dd" & strSeparador & "yy"
    End If
    Debug.Print "Separador fecha " & strSeparador
    Debug.Print "Formato fecha " & strDato
    sngDato = 1234.5
    strDato = Format(sngDato, "Standard")
    strSeparadorMiles = Mid(strDato, 2, 1)
    strSeparadorDecimal = Mid(strDato, 6, 1)
    Debug.Print "Separador de millares " _
              & strSeparadorMiles
    Debug.Print "Separador de decimales " _
              & strSeparadorDecimal
    Debug.Print "Formato numérico " & strDato
    strDato = Format(sngDato, "Currency")
    Debug.Print "Formato moneda " & strDato
End Sub
```

Este procedimiento en mi ordenador muestra:

```
Separador fecha /
Formato fecha 31/12/2000
Separador de millares .
Separador de decimales ,
Formato numérico 1.234,50
Formato moneda 1.234,50 €
```

El resultado será diferente en función de la Configuración Regional propia de cada equipo.