Comencemos a programar con

VBA - Access

Entrega 14

Funciones de VBA II

Eduardo Olaz

Funciones adicionales para formato

Funciones derivadas de VBScript

Las últimas versiones de VBA han incorporado nuevas funciones para el formato de números y fechas.

Estas funciones son

- **FormatNumber**
- **FormatCurrency**
- **FormatPercent**
- **FormatDateTime**
- **MonthName**
- **WeekDayName**

Estas funciones están derivadas de sus correspondientes en **VBScript**; una versión de Visual Basic interpretado desarrollado inicialmente para su implementación en páginas Web.

Función FormatNumber

Devuelve una cadena obtenida al formatear un número.

Su sintaxis es

```
FormatNumber (expresión[, numDígitosDespuésDeDecimal [,
incluirDígitoInicial [,
utilizarParéntesisParaNúmerosNegativos [,
agruparDígitos]]]])
```

La *expresión* puede ser cualquiera que devuelva un valor numérico.

numDígitosDespuésDeDecimal define el número de decimales del formato de salida. Si se pusiera el valor **-1**, usaría la configuración por defecto. Si necesita recortar los decimales de ***expresión***, realizará en su caso, un redondeo de cifras.

incluirDígitoInicial define si se pone un cero inicial, antes del separador de decimales, si el valor de la parte entera fuera cero.

utilizarParéntesisParaNúmerosNegativos define si se utilizan los paréntesis para indicar un valor negativo.

agruparDígitos define si se agrupan los números mediante el delimitador de grupos (miles) especificado en la configuración regional.

Estos tres últimos parámetros toman valores long.

Sus valores válidos son **vbTrue**, **vbFalse**, **vbUseDefault**.

vbUseDefault fuerza a usar la configuración Regional.

El único parámetro obligatorio es la expresión que devuelve el número a formatear.

Si no se pasa otro parámetro el formato se ajusta a lo definido en la Configuración Regional.

FormatNumber admite también la terminación con **$**. *Ver el párrafo "Consideraciones previas sobre el uso del signo Dólar"* en esta misma entrega.

En mi ordenador **FormatNumber** devuelve las siguientes cadenas.

`FormatNumber(1234567890.123)` → `" 1.234.567.890,12"`

`FormatNumber(-123456.1234,3,,vbTrue,vbFalse)` → `"(123456,123)"`

También admite parámetros, ó argumentos, por nombre

`FormatNumber$(Expression:=3*45,NumDigitsAfterDecimal:=2)` → `"135,00"`

Considerando las posibilidades que tiene la función **Format**, la función **FormatNumber** es útil sobre todo para el formato directo de números, sin usar parámetros:

`FormatNumber(1234567890.123)` → `" 1.234.567.890,12"`

Función FormatCurrency

FormatCurrency devuelve una cadena tras formatear un número al modo moneda.

Su sintaxis es

FormatCurrency (*expresión*[, *numDígitosDespuésDeDecimal* [, *incluirDígitoInicial* [, *utilizarParéntesisParaNúmerosNegativos* [, *agruparDígitos*]]]]**)**

Todo lo comentado sobre los parámetros de la función **FormatNumber**, es aplicable para **FormatCurrency**.

`FormatCurrency(1234567890.123)` → `" 1.234.567.890,12 €"`

`FormatCurrency(1234567890.123, 0)` → `" 1.234.567.890 €"`

Función FormatPercent

FormatPercent devuelve una cadena tras formatear un número al modo porcentaje.

Multiplica el valor de *expresión* por **100** y le añade el signo de porcentaje **%**.

Su sintaxis es

FormatPercent (*expresión*[, *numDígitosDespuésDeDecimal* [, *incluirDígitoInicial* [, *utilizarParéntesisParaNúmerosNegativos* [, *agruparDígitos*]]]]**)**

Todo lo comentado sobre los parámetros de la función **FormatNumber**, es aplicable con la función **FormatPercent**.

`FormatPercent (0.123)` → **`"12,3%"`**

`FormatPercent (3.1415926,4)` → **`" 314,1593%"`**

Función FormatDateTime

FormatDateTime Devuelve una cadena tras formatear un número al modo fecha hora, ó fecha.

Su sintaxis es

FormatDateTime (*fecha*[, *formatoConNombre*])

formatoConNombre puede tomar cualquiera de estos valores

vbGeneralDate	Muestra la fecha en **Fecha Corta**, la hora en **Hora Larga** ó ambas en formato si el parámetro es una Fecha con una parte de Hora
vbLongDate	Muestra la fecha en el formato **Fecha Larga** de la configuración.
vbShortDate	Muestra la fecha en el formato **Fecha Corta**.
vbLongTime	Muestra la fecha en el formato especificado en la configuración.
vbShortTime	Muestra la fecha en el formato de **24 Horas (hh:mm)**.

El siguiente procedimiento hace uso de las distintas posibilidades de **FormatDateTime**.

```
Public Sub PruebaFormatDateTime()
    Dim datFechaHora As Date
    Dim datFecha As Date
    Dim datHora As Date

    datFecha = Date
    datHora = Time
    datFechaHora = datFecha + datHora

    Debug.Print FormatDateTime(datFecha)
    Debug.Print FormatDateTime(datHora)
    Debug.Print FormatDateTime(datFechaHora)
    Debug.Print
    Debug.Print FormatDateTime(datFechaHora,vbGeneralDate)
    Debug.Print FormatDateTime(datFechaHora, vbLongDate)
    Debug.Print FormatDateTime(datFechaHora, vbLongTime)
    Debug.Print FormatDateTime(datFechaHora, vbShortDate)
    Debug.Print FormatDateTime(datFechaHora, vbShortTime)
End Sub
```

Tras ejecutarlo, en mi ordenador, muestra los siguientes valores:

```
20/02/2005
21:15:35
20/02/2005 21:15:35

20/02/2005 21:15:35
domingo 20 de febrero de 2005
21:15:35
20/02/2005
21:15
```

El resultado puede cambiar, respecto a otros ordenadores, según su Configuración regional.

Función MonthName

MonthName Devuelve el nombre del mes en el idioma especificado en la configuración regional. Su sintaxis es

MonthName(*mes [, abreviar]***)**

El parámetro que se le pasa es un número que representa al mes. Su valor debe estar entre **1** y **12**; caso contrario dará error de "Argumento o llamada a procedimiento no válida"

El parámetro ***abreviar*** , de tipo boleano, indica si queremos el nombre abreviado.

En mi ordenador devuelve los siguientes valores

MonthName(1) → "enero"
MonthName(2) → "febrero"
- - - - - - - - - -
MonthName(11) → " noviembre"
MonthName(12) → " diciembre"

MonthName(9,True) → " sep"
MonthName(10,True) → " oct"

Función WeekdayName

WeekdayName Devuelve el nombre del día de la semana que se le pasa como parámetro, en el idioma especificado en la configuración regional. Su sintaxis es

WeekdayName (*díaDeLaSemana, abreviar, primerDíaDeLaSemana***)**

El parámetro que se le pasa es un número que representa al día de la semana.

Su valor debe estar entre **1** y **7**.

El parámetro ***abreviar*** , de tipo boleano, indica si queremos el nombre abreviado.

El parámetro ***primerDíaDeLaSemana***, es una de las constantes que vimos en la función Format , indica el día que queremos adoptar como el primero de la semana.

vbUseSystem
vbSunday
vbMonday
vbTuesday
vbWednesday
vbThursday
vbFriday
vbSaturday
vbUseSystemDayOfWeek

En mi equipo esta función devuelve los siguientes valores para distintas llamadas.

```
WeekdayName(1) → " lunes"
WeekdayName(1 ,True ,vbUseSystemDayOfWeek) → " lun"
WeekdayName(1,False,vbSunday) → " domingo"
WeekdayName(1,,vbMonday) → " lunes"
WeekdayName(1,,vbMonday) → " lunes"
```

Se pueden obtener resultados “exóticos” cambiando el parámetro ***primerDíaDeLaSemana***.

Todas estas funciones tienen algo en común:

Facilitan la presentación de datos acorde con la configuración del ordenador de cada usuario, evitando tener que acceder a la configuración regional para efectuar el formateo adecuado del dato.

Manipulación de cadenas

Consideraciones previas sobre el uso del signo Dólar en funciones que devuelven cadenas de texto

Basic, desde sus principios, ya incluía todo un conjunto de procedimientos para el tratamiento de cadenas, entre ellos podemos mencionar **Left$**, **Right$** y **Mid$**.

Estos mismos procedimientos están disponibles con VBA, pero con una salvedad; pueden usarse sin poner el signo del Dólar al final del nombre del procedimiento.

A pesar de que la ayuda de VBA no lo utiliza, podemos afirmar que su uso es muy recomendable.

La ayuda de VBA indica que estas funciones devuelven un tipo **Variant** (**String**).

Esto supone que internamente VBA tiene que hacer una conversión previa de **Variant** a **String**.

La utilización del signo del Dólar fuerza a la función a devolver directamente un tipo **String**, con lo que la velocidad de proceso se incrementa.

Como veremos después la función **Left** devuelve la parte izquierda de una cadena.

Incluyo aquí el código para hacer una pequeña comprobación de velocidad.

La función **Timer** devuelve el número de segundos que han transcurrido desde las cero horas del día de hoy.

```
Public Sub Prueba()
    Const Bucle As Long = 10000000
    Const conCadena As String = "ABCDEFGHIJKLMN"
    Dim i As Long
    Dim strCadena As String
    Dim tim0 As Single
    tim0 = Timer
    For i = 1 To Bucle
        strCadena = Left(conCadena, 7)
    Next i
    Debug.Print Timer - tim0 & " Segundos"
    tim0 = Timer
    For i = 1 To Bucle
        strCadena = Left$(conCadena, 7)
    Next i
    Debug.Print Timer - tim0 & " Segundos"
    tim0 = Timer
    For i = 1 To Bucle
        strCadena = Left$(String:=conCadena, Length:=7)
    Next i
    Debug.Print Timer - tim0 & " Segundos"
End Sub
```

El resultado, en mi PC, tras ejecutar este código es el siguiente

4,0156 Segundos

1,4531 Segundos

1,4375 Segundos

Lo que indica que **Left$** es casi **3 veces más rápido** que **Left.**

Se puede comprobar que si llamamos a la función utilizando parámetros por nombre, no se penaliza su velocidad; incluso es ligeramente más rápido.

Pruebas realizadas con el resto de funciones **Right$**, **Mid$**, etc. dan resultados semejantes.

La función **Format** que analizamos en la entrega anterior, también admite la utilización del signo Dólar: **Format$(123456.123,"#,##0.00")** .

Tras efectuar una prueba similar pero con un bucle de 1.000.000 de iteraciones.

El resultado ha sido **2,3594** frente a **2,2812** Segundos con **Format$**.

Con lo que se comprueba que aunque el código también es más rápido, no existe la diferencia de velocidad que se consigue por ejemplo con **Left$**. Esto es lógico ya que la asignación de un **String** frente a un **Variant**, es un proceso menor si lo comparamos con las tareas que realiza internamente **Format**. Esa es la razón por la que no he utilizado el signo **$** en la entrega anterior; lo que no quita para que yo recomiende su uso, incluso con la función **Format**.

Función Left

Devuelve los n **caracteres** situados a la izquierda de una cadena dada.

Sintaxis

```
Left(string, length)
Left$(string, length)
Left$("Eduardo Olaz", 7)   →   "Eduardo"
Left(string:= "Eduardo Olaz", length := 7)   →   "Eduardo"
```

length debe ser mayor ó igual a cero. Si es cero devolverá la cadena vacía **""**.

Si fuese menor a cero generará el error 5 "Argumento o llamada a procedimiento no válida".

Función LeftB

Devuelve los n **Bytes** situados a la izquierda de una cadena dada.

Sintaxis

```
LeftB(string, length)
LeftB$(string, length)
LeftB$("Eduardo Olaz", 14)   →   "Eduardo"
```

¿Cómo es posible esto?

Las cadenas que maneja VBA son del tipo **UNICODE** . En este formato cada carácter está formado por **2 Bytes**.

LeftB extrae **Bytes**. **Left** extrae **Caracteres**. Por ello **LeftB** es adecuada para el manejo directo de cadenas de Bytes y **Left** para el manejo de caracteres.

Lo dicho para **LeftB** será igualmente aplicable para las funciones **RightB** y **MidB** que se corresponden con **Right** y **Mid**, funciones que veremos a continuación.

Function Right

Devuelve los n **caracteres** situados a la derecha de una cadena dada.

Sintaxis

```
Right (string, length)
Right$(string, length)
Right( "Eduardo Olaz", 4)   →   " Olaz"
Right$( string:= "Eduardo Olaz", length := 4)   →   "Olaz"
```

length debe ser mayor ó igual a cero. Si es cero devolverá la cadena vacía "".

Tanto para **Left** como para **Right** si ***length*** es mayor que el número de caracteres contenidos en la cadena, devolverá la cadena completa sin generar un error.

```
Left$( "Eduardo Olaz", 50)   →   "Eduardo Olaz"
Right( "Eduardo Olaz", 50)   →   "Eduardo Olaz"
```

Function Mid

Devuelve los n **caracteres** de una cadena dada, situados a partir de una posición.

Sintaxis

```
Mid(string, start[, length])
Mid$(string, start[, length])
Mid$( "Eduardo Olaz", 9, 3)   →   "Ola"
Mid( string:="Eduardo Olaz",start:= 9,length:= 3) → "Ola"
```

String es la cadena de la que vamos a extraer caracteres.

start es la posición a partir de la cual vamos a extraerlos. Es de tipo **Long** y mayor que **0**.

length es el número de caracteres que queremos extraer. Su tipo es **Long**.

Este parámetro es opcional. Si no se incluyera, o su valor fuera superior al número de caracteres que hay desde la posición de partida, incluyendo su carácter, devolvería todos los caracteres que hay desde la posición de ***start***.

```
Mid( "Eduardo Olaz", 9)   →   "Olaz"
Mid$( "Eduardo Olaz", 9, 30)   →   "Olaz"
```

Instrucción Mid

La instrucción **Mid**, Reemplaza determinado número de caracteres de una cadena con los caracteres de otra.

Sintaxis

```
Mid(VariableString, start[, length]) = Cadena
```

Mid$(*VariableString*, *start*[, *length*]) = *Cadena*

La cadena den la que queremos hacer el cambio se debe pasar mediante una variable de tipo **String** ó **Variant** (***VariableString***).

La cadena de la que queremos extraer los caracteres puede ser pasada de forma directa, o mediante una variable.

El número de caracteres reemplazados es menor o como mucho igual al número de caracteres de ***VariableString***.

Tomemos como ejemplo el siguiente código

```
Public Sub PruebaInstruccionMid()
    Dim strAlumno As String

    strAlumno = "Maitane Gaztelu"
    Debug.Print "Cadena inicial    " & strAlumno
    Mid(strAlumno, 1, 7) = "Enrique"
    Debug.Print "Cadena cambiada   " & strAlumno
    Mid(strAlumno, 9, 7) = "Garayoa"
    Debug.Print "Segundo cambio    " & strAlumno
    ' se puede cambiar por una parte de la 2ª cadena
    Mid(strAlumno, 13, 3) = "llámame"
    Debug.Print "Cadena parcial    " & strAlumno
    ' Mid sólo puede cambiar caracteres que ya existan _
      Esto no va a funcionar bien
    Mid(strAlumno, 9, 10) = "Martínez"
    Debug.Print "Cambio incompleto " & strAlumno
End Sub
```

Si lo ejecutamos nos mostrará

```
Cadena inicial    Maitane Gaztelu
Cadena cambiada   Enrique Gaztelu
Segundo cambio    Enrique Garayoa
Cadena parcial    Enrique Garallá
Cambio incompleto Enrique Martíne
```

Si no se indica la longitud (parámetro ***length***) pasará todo el contenido de la segunda cadena, si su longitud es menor o igual al resto de ***VariableString***. Si fuera menor cambiará un número de caracteres igual a los caracteres que restan de la primera, contando desde el que está en la posición ***start***. Hasta el final de ***VariableString***.

```
Public Sub PruebaMid()
    Dim strInicial As String
    strInicial = "Cadena Inicial"
    Debug.Print strInicial
    Mid(strInicial, 7) = "123"
```

```
        Debug.Print strInicial
        Mid(strInicial, 7) = "$$$$$$$$$$$"
        Debug.Print strInicial
    End Sub
```

Al ejecutar **PruebaMid** nos mostrará

```
Cadena Inicial
Cadena123icial
Cadena$$$$$$$
```

Como en el caso de Las funciones anteriores, también existe la versión **MidB** para efectuar cambios a nivel de **Bytes**.

Funciones LTrim, Rtrim y Trim

Devuelve un tipo **Variant** (**String**) que contiene la copia de una cadena determinada a la que se le han eliminado los espacios en blanco de los extremos.

LTrim quita los posibles espacios de la izquierda.

RTrim quita los posibles espacios de la derecha.

Trim quita tanto los de la izquierda como los de la derecha.

Sintaxis

LTrim(*cadena*)
RTrim(*cadena*)
Trim(*cadena*)

El parámetro *cadena* es obligatorio, y se puede pasar cualquier cadena o expresión que devuelva una cadena de texto.

Como valor para *cadena* admite **Null**, en este caso devolverá también **Null**.

LTrim , **RTrim** y **Trim** admiten también ser llamadas con el signo de dólar.

En este caso si se les pasa un **Null** como parámetro, generarán un error.

Dim Variable as String * Longitud

LTrim(" 34567890 ") → "34567890 "

RTrim(" 34567890 ") → " 34567890"

Trim(" 34567890 ") → "34567890"

Trim(**Null**) → **Null**

Trim$(" 34567890 ") → "34567890"

Funciones Len y LenB

Devuelven un tipo **Long** que contiene el número de caracteres de una cadena (**Len**) o el número de Bytes (**LenB**). Si *cadenaOVariable* contiene **Null**, devuelve **Null**.

Sintaxis

Len(*cadenaOVariable*)
LenB(*cadenaOVariable*)

El parámetro *cadenaOVariable* puede ser una cadena o una variable.

A Len puede pasársele una variable definida por el usuario.

Si a Len se le pasa un valor **Null**, devuelve **Null**.

Este código devuelve la longitud de diferentes tipos de variables.

Incluso da la longitud del Tipo **Linea** definido por el usuario y a su vez compuesto por dos tipos **Punto**, también definido por el usuario.

```
Public Type Punto
    X As Single
    Y As Single
End Type

Public Type Linea
    pto1 As Punto
    pto2 As Punto
End Type

Public Sub PruebaLen()
    Dim intEntero As Integer
    Dim lngLong As Long
    Dim curMoneda As Currency
    Dim strCadena As String * 6
    Dim ptoCoordenadas As Punto
    Dim linSegmento As Linea
    Dim a(1 To 10) As String

    a(2) = "1234567890"

    Debug.Print "Longitud del tipo Integer " & _
                Len(intEntero)
    Debug.Print "Longitud del tipo Long " & _
                Len(lngLong)
    Debug.Print "Longitud del tipo Currency " & _
                Len(curMoneda)
    Debug.Print "Longitud del tipo Punto " & _
                Len(ptoCoordenadas)
    Debug.Print "Longitud del tipo Linea " & _
                Len(linSegmento)
    Debug.Print "Longitud caracteres de strCadena " & _
                Len(strCadena)
    Debug.Print "Longitud (Bytes) de strCadena " & _
                LenB(strCadena)
    Debug.Print "Longitud de a(1) " & _
                Len(a(1))
```

```
        Debug.Print "Longitud Caracteres de a(2) " & _
                    Len(a(2))
        Debug.Print "Longitud (Bytes) de a(2) " & _
                    LenB(a(2))
    End Sub
```

El resultado de este código es:

```
Longitud del tipo Integer 2
Longitud del tipo Long 4
Longitud del tipo Currency 8
Longitud del tipo Punto 8
Longitud del tipo Linea 16
Longitud caracteres de strCadena 6
Longitud (Bytes) de strCadena 12
Longitud de a(1) 0
Longitud Caracteres de a(2) 10
Longitud (Bytes) de a(2) 20
```

Buscar y sustituir cadenas

Entre el bagaje de funciones que posee VBA existen varias para la búsqueda y sustitución de elementos en cadenas.

Ya hemos visto la Instrucción **Mid** que permite realizar sustituciones en una cadena, aunque con ciertas limitaciones.

Igualmente las funciones **Left**, **Mid** y **Right** podrían usarse para buscar subcadenas dentro de una cadena de texto.

Función InStr

Devuelve un tipo **Variant**, convertido a **Long** que indica la posición en la que podemos encontrar una Subcadena en otra cadena.

Si como cadena buscada, o a buscar, se pasa un **Null**, devolverá un **Null**.

Sintaxis

InStr(*[start,]string1, string2[, comparar*])

Parámetros

start: opcional, posición donde empezar a buscar.

String1: Cadena ó expresión de cadena en la que se busca

string2: Cadena buscada

comparar: opcional, constante que indica qué se consideran cadenas iguales. Si no se escribe toma el valor por defecto.

Las opciones para ***comparar*** son:

vbUseCompareOption	**-1**	Sigue el criterio definido en Option Compare
vbBinaryCompare	**0**	Hace una comparación a nivel de Bytes.
vbTextCompare	**1**	Compara los textos
vbDatabaseCompare	**2**	Sigue el criterio definido en la base de datos Access

Como en el caso de las funciones anteriores, también existe la función **InStrB** que busca valores Byte en una cadena de Bytes.

Función InStrReverse

Es similar a **InStr**, salvo que la búsqueda comienza desde el final de la cadena

Como puede comprobarse, su sintaxis difiere ligeramente de la de **InStr**.

```
Sintaxis
InstrRev(cadena1, cadena2[, inicio[, comparar]])
```

inicio define la posición final de la búsqueda. Si se omite empieza por el último carácter.

Al igual que **InStr** si se introdujera un nulo como cadena a buscar o en la que buscar, devolvería un **Null**.

Función StrComp

Devuelve un entero como resultado de la comparación de dos cadenas. Si como parámetro de alguna de las dos cadenas se pasara un nulo, devolvería **Null**.

Sintaxis:

```
StrComp(string1, string2[, comparar])
```

Si ***string1*** es menor que ***string2*** devuelve **-1**

Si ***string1*** es igual a ***string2*** devuelve **0**

Si ***string1*** es mayor que ***string2*** devuelve **1**

Si ***string1*** o ***string2*** es **Null** devuelve **Null**

La forma de comparar dependerá del valor(opcional) de ***comparar***.

Como referencia ver ***comparar*** en **InStr**.

```
StrComp ("curso","VBA")
```
devuelve **-1**

```
StrComp ("curso","VBA", ,vbBinaryCompare)
```
devuelve **1**

Función Replace

Devuelve una cadena en la que se han cambiado una subcadena por otra dada.

Sintaxis:

```
Sintaxis
Replace(expresión, encontrar, reemplazarCon [, inicio[,
contar[, comparar]]])
```

expresión: Cadena o expresión de cadena en la que buscar

encontrar: Subcadena buscada

reemplazarCon: Subcadena que reemplazará a la anterior

inicio: Cadena o expresión

comparar: Forma de comparar. Como referencia ver ***comparar*** en **InStr**.

Este código sirve para ver distintas formas de utilizar la función, y sus resultados:

```
Public Sub PruebaReplace()
    Dim strCadena As String
    strCadena = "aaabbbcccdddeeeEEEfff"
    Debug.Print strCadena
    Debug.Print Replace(strCadena, "a", "B")
    Debug.Print Replace(strCadena, "f", "C", 8)
    Debug.Print Replace(strCadena, "e", "1", , 2)
    Debug.Print Replace(strCadena, _
                    "e", "1", , , vbBinaryCompare)
    Debug.Print Replace(strCadena, "e", "1")
    Debug.Print Replace(strCadena, "cccddd", "$$")
End Sub
```

El resultado de este código es

```
aaabbbcccdddeeeEEEfff
BBBbbbcccdddeeeEEEfff
ccdddeeeEEECCC
aaabbbcccddd11eEEEfff
aaabbbcccddd111EEEfff
aaabbbcccddd111111fff
aaabbb$$eeeEEEfff
```

Función StrReverse

La función **StrReverse** devuelve una cadena con los caracteres invertidos respecto a la cadena pasada como parámetro.

Sintaxis

StrReverse(*cadena***)**

StrReverse("ABCDEFGHIJK") → "KJIHGFEDCBA"

Función Filter

La función **Filter** devuelve un array con los elementos de otro array que contienen (o no contienen) un determinado valor. La cadena devuelta está basada en el índice 0.

Sintaxis

Filter(*sourcesrray*, *match*[, *include*[, *comparar*]])

Sourcesrray es la matriz en donde se va a buscar.

match es el valor con el que queremos comparar.

include Si es **True** hace que se busquen los valores que contengan a ***match***.
Si es **False**, los que no lo contengan.

comparar Forma de comparar. Como referencia ver ***comparar*** en **InStr**.

El siguiente procedimiento primero busca en una matriz llamada **aMatriz** los datos que contengan la palabra **"Jesús"** y se los asigna a la matriz **aFiltrada**. Muestra los datos en la ventana **Inmediato** y seguidamente busca aquéllos elementos que no contengan la palabra **"Eduardo"** y los muestra.

```
Public Sub PruebaFilter()
    Dim aFiltrada() As String
    Dim aMatriz(1 To 6) As String
    Dim i As Long
    aMatriz(1) = "Antonio Martínez"
    aMatriz(2) = "Jesús Martínez"
    aMatriz(3) = "Eduardo Olaz"
    aMatriz(4) = "Eduardo Pérez"
    aMatriz(5) = "Jesús Pérez"
    aMatriz(6) = "Juan Pérez"

    ' Mostramos los elementos _
      que contienen "Jesús"
    aFiltrada = Filter(aMatriz, "Jesús")
    For i = LBound(aFiltrada) To UBound(aFiltrada)
        Debug.Print aFiltrada(i)
    Next i
    Debug.Print

    ' Mostramos los elementos _
      que no contienen "Eduardo"
    aFiltrada = Filter(aMatriz, "Eduardo", False)
    For i = LBound(aFiltrada) To UBound(aFiltrada)
        Debug.Print aFiltrada(i)
    Next i
End Sub
```

El resultado de todo esto es

```
Jesús Martínez
Jesús Pérez

Antonio Martínez
Jesús Martínez
Jesús Pérez
Juan Pérez
```

Función Split

Devuelve una matriz, basada en el índice 0, que contiene un número especificado de subcadenas extraídas de una cadena de texto.

Sintaxis

Split(*expresión*[, *delimitador*[, *contar*[, *comparar*]]]**)**

expresión Una cadena o una expresión que devuelva una cadena de caracteres. La cadena puede contener caracteres especificadores de separación.

delimitador Es un carácter que sirve para definir los límites de las subcadenas dentro de la cadena pasada. Si no se indica, toma como carácter de separación el espacio en blanco.

contar Indica el número de subcadenas que queremos extraer. Si se pasa el valor -1, se especifica que queremos extraer todas las subcadenas. Es el modo por defecto

comparar Al igual que en las funciones anteriores, forma de comparar entre sí las cadenas. Como referencia ver ***comparar*** en **InStr**.

La función Split es útil para obtener datos desde un fichero de texto con los campos separados por Delimitadores.

El siguiente código hace primero un

```
Public Sub PruebaSplit()
    Dim strDatos As String
    Dim aDatos() As String
    Dim i As Long

    strDatos = "Andrés Tomás Iker"
    aDatos = Split(strDatos)
    For i = LBound(aDatos) To UBound(aDatos)
        Debug.Print aDatos(i)
    Next i
    Debug.Print
    strDatos = "Martínez Pérez;Gómez Iturri;García Martín"
    aDatos = Split(strDatos, ";")
    For i = LBound(aDatos) To UBound(aDatos)
        Debug.Print aDatos(i)
    Next i
End Sub
```

Este código mostrará lo siguiente

```
Andrés
Tomás
Iker

Martínez Pérez
Gómez Iturri
García Martín
```

Función Join

La función **Join** se podría decir que es la inversa de la función **Split**.

Toma una matriz de caracteres y los junta en una única cadena.

El siguiente código hace lo siguiente

- Asigna las estaciones del año a una matriz mediante la función **Split**.
- A continuación vuelve a juntar los datos en pero usando como separador un espacio en blanco " ".

```
Public Sub PruebaJoin()
    Dim strDatos As String
    Dim aDatos() As String
    Dim strResultado As String

    strDatos = "Primavera;Verano;Otoño;Invierno"
    aDatos = Split(strDatos, ";")
    strResultado = Join(aDatos, " ")
    Debug.Print strResultado
End Sub
```

Tras ejecutar el código se muestra

```
Primavera Verano Otoño Invierno
```

Operador Like

Este operador se utiliza para comparar dos cadena de caracteres.

resultado **=** *cadena* **Like** *patrón*

Realiza una comparación entre ***cadena*** y ***patrón*** basándose en la configuración **Option Compare** ubicada en el comienzo del módulo. Los valores asignables a Option Compare son:

- `Option Compare` **Text**
- `Option Compare` **Binary**
- `Option Compare` **Database**

Si la cadena responde a la configuración definida en patrón devolverá **True**, en caso contrario devolverá **False**.

Si alguno de los operadores fuese **Null**, devolverá **Null**.

Como ***cadena*** puede usarse una cadena o una expresión que devuelva una cadena.

Para una comparación con `Option Compare` **Text**, las siguientes expresiones devolverán:

`left("Eduardo Olaz", 7) like "EDUARDO"` → **True**

`left(" Eduardo Olaz", 7) like "EDUARDO"` → **False**

```
"Eduardo Olaz" like Null → Null
```

El operador **Like** admite “comodines”

Estos comodines son

?	Representa a cualquier carácter ó número (uno por cada ?).
*****	Representa a cualquier número de caracteres (incluso ninguno)
#	Representa un dígito (del 0 al 9)
[***listacaracteres***]	Un carácter cualquiera incluido en ***listacaracteres***
[!***listacaracteres***]	Ninguno de los caracteres incluido en ***listacaracteres***

```
"BMP1956" like "???1956" → True
"BMP1956" like "???####" → True
"BMP1956" like "???????" → True
"BMP1956" like "*####" → True
"BMP1956" like "*##57" → False
"Cualquier cadena" like "*" → True
"Referencia de producto BMP1956" like "*1956" → True
"BMP1956" like "???1956" → True
"BMP1956" like "*####" → True
left("Eduardo Olaz", 7) like "EDUARDO" → True
"Eduardo Olaz" like Null → Null
Null like Null → Null
```

Funciones Asc y AscB

La función **Asc** devuelve un entero con el código del primer carácter de la cadena pasada como parámetro.

AscB devuelve un entero con el código del primer Byte de la cadena.

Sintaxis

Asc(*cadena***)**

AscB(*cadena***)**

```
Asc("ABCDEFGHIJK") → 65
```

Funciones Chr y Chr$

La función Chr y su equivalente Chr$, actúan de forma inversa a la función Asc.

Devuelven un carácter que tiene como código de carácter el valor que se le pasa como parámetro.

Sintaxis

Chr(*códigocar***)**

Así como `Asc("A")` devuelve **65**, `Chr(65)` → **"A"**

El procedimiento **PruebaChrAsc**, muestra los códigos de carácter correspondientes a las letras que van de la "A" a la "Z".

```
Public Sub PruebaChrAsc()
    Dim i As Long
    For i = Asc("A") To Asc("Z")
        Debug.Print Format(i, "00 - ") & Chr$(i)
    Next i
End Sub
```

El resultado será

```
65 - A
66 - B
67 - C
68 - D
69 - E
70 - F
. . . .
. . . .
84 - T
85 - U
86 - V
87 - W
88 - X
89 - Y
90 - Z
```

Diferencia entre funciones que trabajan en modo Carácter y en modo Byte.

Ya he comentado que existen funciones intrínsecas de VBA, como **Asc**, **Left**, **Mid** y **Right** que trabajan sobre caracteres y que tienen sus equivalentes **AscB**, **LeftB**, **MidB** y **RightB** que trabajan sobre Bytes.

Con ellas podemos ver que las cadenas de VBA están en realidad formadas por caracteres que utilizan 2 Bytes para ser representados.

Tras ejecutar este código podemos ver la diferencia del resultado utilizando **Asc** y **Mid**, respecto a **AscB** y **MidB**.

```
Public Sub ComparaFuncionB(ByVal Cadena As String)
    Dim i As Long
    Dim intCodigo As Integer
    Dim strCaracter
    Debug.Print "Códigos de Caracteres"
    For i = 1 To Len(Cadena)
        intCodigo = Asc(Mid(Cadena, i, 1))
        strCaracter = Mid(Cadena, i, 1)
        Debug.Print Format(intCodigo, "000  ") _
```

```
                    & strCaracter & "  ";
        Next i
        Debug.Print
        Debug.Print "Códigos Bytes"
        For i = 1 To 2 * Len(Cadena)
            intCodigo = AscB(MidB(Cadena, i, 1))
            Debug.Print Format(intCodigo, "000 ");
        Next i
        Debug.Print
    End Sub
```

Recordemos que

MidB va extrayendo **Byte** a **Byte**

Mid extrae **carácter** a **carácter**.

Ejecutemos este procedimiento mediante

```
ComparaFuncionB "VBA €"
```

Imprimirá lo siguiente en la ventana inmediato:

```
Códigos de Caracteres
086  V  066  B  065  A  032     128  €
Códigos Bytes
086 000 066 000 065 000 032 000 172 032
```

A primera vista vemos que por cada carácter que extrae la función **Mid**, la función **MidB** extrae 2.

Esto se produce porque las cadenas **String** de VBA manejan caracteres **Unicode**, y un carácter **Unicode** está compuesto de **2 Bytes**.

Voy a hacer una reflexión especial sobre el carácter **€**, correspondiente al Euro.

La función **Asc**("**€**") devuelve 128, que es el número de carácter que maneja Windows.

En cambio, de forma similar a lo que hemos visto en el procedimiento anterior

AscB(**LeftB**("**€**",1)) → **172**

AscB(**RightB**("**€**",1)) → **32**

Lo que nos indica que internamente VBA utiliza dos bytes, siendo el primero 172 y el segundo 32.