Bloque 2. Visual Basic .NET
UT 3. Confección de interfaces de usuario y creación de componentes visuales

TEMA 7

Funciones del lenguaje.

1. Introducción

Las funciones del lenguaje proporcionan todas las herramientas necesarias para trabajar con los objetos y los formularios. Una vez que sepamos cómo acceder a los formularios y objetos necesitaremos utilizar funciones para preparar, procesar o producir un resultado. Tenemos decenas de funciones que serán muy importantes pero entre todas hay varios grupos que utilizaremos muy a menudo.

1.1. Cómo utilizarlas en el código

Al usar una función de una fórmula, escribiremos el nombre de la función y le proporcionaremos los argumentos requeridos. Por ejemplo, la función Len requiere un argumento de cadena y calcula la longitud de la misma.

```
Dim x As String
x = "hola"
formula = Len(x) 'La fórmula devuelve el número 4
```

Al proporcionar argumentos del tipo incorrecto requerido por la función, se genera un error. Por ejemplo, al llamar a `Len(3)` se produciría un error ya que Len no acepta un argumento numérico.

Algunas veces las funciones pueden aceptar argumentos o tipos de argumentos diferentes. Por ejemplo, la función `CDate` acepta un argumento de cadena simple para formar un valor de fecha, o bien 3 valores numéricos que incluyen el año, mes y día, respectivamente, para formar un valor de fecha a partir de los mismos.

Ejemplo con la función Mid

```
Dim x as String
x = "hola"
'Comenzar en la posición 2 e ir al final de la cadena
formula = Mid(x, 2) 'fórmula es ahora "ola"
'Comenzar en la posición 2 y extraer 1 carácter
formula = Mid (x, 2, 1) 'fórmula es ahora "o"
```

Las clases de funciones son: matemáticas, resumen, finanzas, cadena, fecha/hora, rango de fechas, matriz, conversión de tipos, accesos directos de programación, tiempo de evaluación, estado de impresión, propiedades de documentos y funciones adicionales. Existen también algunas funciones específicas a las fórmulas de formato condicional.

Comenzaremos con las más utilizadas.

2. Funciones de conversión de tipos

Una expresión de conversión convierte una expresión en un tipo determinado. Existen palabras clave de conversión específicas que convierten expresiones en tipos primitivos; también existen dos palabras clave generales de conversión, **CType** y **DirectCast**, que convierten una expresión en cualquier tipo.

Es decir, por un lado tendremos una colección de funciones una para cada tipo de datos y otra general (CType) que convierte a cualquier tipo. Esta función la hemos visto en algún ejemplo. Permite convertir cualquier formato y es más fácil de acordarse que las otras.

DirectCast es un caso especial, porque las conversiones de **Object** a cualquier otro tipo se realizan por conversión directa en orden jerárquico (todos los comportamientos especiales de conversión de **Object** se omiten). Cuando se convierte una expresión de tipo **Object** cuyo tipo en tiempo de ejecución es un tipo de valor primitivo, **DirectCast** inicia una excepción **System.InvalidTypeException** si el tipo especificado no es el mismo que el tipo en tiempo de ejecución de la expresión. No obstante, si el tipo especificado y el tipo de tiempo de ejecución de la expresión son los mismos, el rendimiento en tiempo de ejecución de **DirectCast** es mejor que el de **CType**.

Si no existe conversión del tipo de la expresión en el tipo especificado, se produce un error. En caso contrario, la expresión se clasifica como un valor y el resultado es el valor producido por la conversión.

Estas funciones se compilan en línea, es decir, el código de conversión forma parte del código que evalúa la expresión. La ejecución es más rápida, ya que no existen llamadas a un procedimiento que realiza la conversión. Cada función convierte una expresión a un tipo de datos específico.

- CBool (expresión)
- CByte (expresión)
- CChar (expresión)
- CDate (expresión)
- CDbl (expresión)

- CDec (expresión)
- CInt (expresión)
- CLng (expresión)
- CObj (expresión)
- CShort (expresión)
- CSng (expresión)
- CStr (expresión)

La **expresión** es un dato obligatorio y debe ser una expresión numérica o una cadena de caracteres (string).

Tipos devueltos

El nombre de la función determina el tipo devuelto, como muestra la siguiente tabla:

Nombre de la función	Tipo de valor devuelto	Intervalo de valores del argumento expresión
CBool	**Boolean**	Cualquier expresión numérica o de cadena (String) válida.
CByte	**Byte**	0 a 255; las fracciones se redondean.
CChar	**Char**	Cualquier expresión String válida, valores comprendidos entre 0 y 65535.
CDate	**Date**	Cualquier representación válida de fecha y hora.
CDbl	**Double**	-1,79769313486231E+308 a -4,94065645841247E-324 para valores negativos; 4,94065645841247E-324 a 1,79769313486231E+308 para valores positivos.
CDec	**Decimal**	+/-79.228.162.514.264.337.593.543.950.335 para números a partir de cero, es decir, números sin decimales. Para números con 28 decimales, el rango es +/-7.9228162514264337593543950335. El menor número distinto de cero es 0,0000000000000000000000000001.
CInt	**Integer**	-2.147.483.648 a 2.147.483.647; las fracciones se redondean.
CLng	**Long**	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807; las fracciones se redondean.
CObj	**Object**	Cualquier expresión válida.
CShort	**Short**	-32.768 a 32.767; las fracciones se redondean.
CSng	**Single**	De -3,402823E+38 a –1,401298E-45 para valores negativos; de 1,401298E-45 a 3,402823E+38 para valores positivos.
CStr	**String**	Los valores devueltos para CStr dependen del argumento expression. Mira la siguiente tabla

En la tabla siguiente se describen los valores devueltos por la función **CStr** en varios tipos de *expresión*

Si el tipo de expresión es	CStr devuelve
Boolean	Cadena que contiene "True" o "False".
Date	Cadena que contiene un valor **Date** (fecha y hora) en el formato de fecha corta del sistema.
Numeric	Cadena que representa el número.

Comentarios

Si el argumento expresión que se pasa a la función está fuera del intervalo de valores del tipo de datos al que se va a convertir, se produce un error.

En general, podemos utilizar las funciones de conversión de tipos de datos para expresar el resultado de alguna operación como un tipo de datos concreto en vez de como el tipo de datos predeterminado. Por ejemplo, utilizaremos `CDec` para forzar la ejecución de operaciones con aritmética decimal en los casos en los que se haría con precisión simple, doble precisión o aritmética entera.

Cuando la parte fraccionaria es exactamente 0,5, `CInt` y `CLng` siempre redondean al número par más cercano. Por ejemplo, 0,5 se redondea a 0 y 1,5 a 2. `CInt` y `CLng` se diferencian de las funciones `Fix` e `Int` en que éstas últimas truncan la parte fraccionaria de un número en lugar de redondearla. Además, `Fix` e `Int` siempre devuelven un valor del mismo tipo que reciben.

`CDate` reconoce literales de fecha y literales de hora además de números comprendidos dentro del intervalo de fechas aceptables. CDate reconoce formatos de fecha que se ajusten a la configuración regional del sistema. Debe suministrar el día, mes y año en el orden correcto para la configuración regional, de no hacerlo así, es posible que la fecha no se interprete de forma correcta. No se puede reconocer un formato de fecha largo si contiene la cadena del día de la semana, por ejemplo "Miércoles".

El tipo de datos `Date` siempre contiene información de fecha y hora. Para la conversión de tipos, Visual Basic .NET considera 1/1/1 (1 de enero del año 1) un valor neutral de fecha y 00:00:00 (medianoche) un valor neutral de hora. Si se convierte un valor `Date` a una cadena, `Cstr` no incluye valores neutrales en la cadena resultante. Por ejemplo, si se convierte #January 1, 0001 9:30:00# en una cadena, el resultado sería "9:30:00 a.m.", ya que la información de fecha se omite. No obstante, la información de fecha sigue estando presente en el valor Date original y se puede recuperar mediante funciones como `DatePart`.

La función `CType` acepta un segundo argumento, `typename`, y convierte *expresión* a typename, donde `typename` puede ser cualquier tipo de datos, estructura, clase o interfaz. Luego veremos más cosas de la función `CType`

Ejemplo de la función CBool

Este ejemplo utiliza la función `CBool` para convertir expresiones en valores `Boolean`. Si una expresión se evalúa a un valor distinto de cero, `CBool` devuelve `True`; en caso contrario, devuelve False.

```
Dim A, B, C As Integer
Dim Comprueba As Boolean
A = 5
B = 5
Comprueba = CBool(A = B)   ' Comprueba se establece a True.
' ...
C = 0
Comprueba = CBool(C)   ' Comprueba se establece a False.
```

Ejemplo de la función CByte

Este ejemplo utiliza la función CByte para convertir expresiones en valores Byte.

```
Dim valorDoble As Double
Dim valorByte As Byte
valorDoble = 125.5678
valorByte = CByte(valorDoble)   ' valorByte se establece a 126.
```

Ejemplo de la función CChar

En este ejemplo, la función CChar convierte el primer carácter de una expresión String en un tipo Char.

```
Dim cadena As String
Dim caracter As Char
cadena = "BCD"    ' CChar convierte sólo el primer carácter de la
cadena.
caracter = CChar(cadena)   ' caracter se establece a "B".
```

El argumento de entrada para CChar debe ser el tipo de datos String. No se puede utilizar CChar para convertir un número en un carácter, porque CChar no acepta un tipo de datos numéricos. Este ejemplo obtiene un número que representa un punto de código (código de carácter) y lo convierte en el carácter correspondiente. Utiliza InputBox para obtener la cadena de dígitos, CInt para convertir la cadena en el tipo Integer y ChrW para convertir el número en el tipo Char.

InputBox es una sencilla instrucción que lee un dato por pantalla. Normalmente utilizaremos formularios pero para capturar rápidamente un valor puede ser útil.

```
Dim valor As String   ' Cadena de dígitos de entrada para convertir.
Dim edad As Integer         ' Números a representar como caracteres.
Dim caracter As Char
valor = InputBox("Introduce una cadena:")
edad = CInt(valor)   ' Convierte la cadena entera en Integer.
caracter = ChrW(CodePoint)   ' caracter se establece a tipo Char.
```

Ejemplo de la función CDate

Este ejemplo utiliza la función CDate para convertir cadenas en valores Date. En general, no se recomienda especificar las fechas y horas como cadenas. Tal y como se puede ver en este ejemplo, en su lugar utiliza literales de fecha y hora, por ejemplo #Feb 12, 1969# y #4:45:233 p.m.#.

```
Dim cadena_fecha, cadena_hora As String
Dim fecha, hora As Date
cadena_fecha = "12 Febrero, 1969"
cadena_hora = "4:35:47 PM"
' ...
fecha = CDate(cadena_fecha)   ' Convierte la cadena en tipo Date.
hora = CDate(cadena_hora)  ' Convierte la cadena en tipo Date.
```

Ejemplo de la función CDbl

Este ejemplo utiliza la función CDbl para convertir expresiones en valores Double.

```
Dim MyDec As Decimal
Dim MyDouble As Double
MyDec = 234.456784D     'El carácter D hace que Mydec sea decimal.
MyDouble = CDbl(MyDec * 8.2D * 0.01D)    ' Convierte el resultado a
Double.
```

Ejemplo de la función CDec

Este ejemplo utiliza la función CDec para convertir un valor numérico en Decimal.

```
Dim MyDouble As Double
Dim MyDecimal As Decimal
MyDouble = 10000000.0587
MyDecimal = CDec(MyDouble)    ' Convierte a decimal.
```

Ejemplo de la función CInt

Este ejemplo utiliza la función CInt para convertir un valor en Integer.

```
Dim MyDouble As Double
Dim MyInt As Integer
MyDouble = 2345.5678
MyInt = CInt(MyDouble)   ' MyInt se establece a 2346.
```

Ejemplo de la función CLng

Este ejemplo utiliza la función CLng para convertir valores en Long.

```
Dim MyDbl1, MyDbl2 As Double
Dim MyLong1, MyLong2 As Long
MyDbl1 = 25427.45
MyDbl2 = 25427.55
MyLong1 = CLng(MyDbl1)   ' MyLong1 contiene 25427.
MyLong2 = CLng(MyDbl2)   ' MyLong2 contiene 25428.
```

Ejemplo de la función CObj

Este ejemplo utiliza la función CObj para convertir un valor numérico en Object. La variable Object en sí contiene sólo un puntero de cuatro bytes, que señala al valor Double que tiene asignado.

```
Dim MyDouble As Double
Dim MyObject As Object
MyDouble = 2.7182818284
MyObject = CObj(MyDouble)       'El valor Double está apuntado por
MyObject.
```

Ejemplo de la función CShort

Este ejemplo utiliza la función CShort para convertir un valor numérico en Short.

```
Dim MyByte as Byte
Dim MyShort as Short
MyByte = 100
MyShort = CShort(MyByte)   ' Convierte a Short.
```

Ejemplo de la función CSng

Este ejemplo utiliza la función CSng para convertir valores en Single.

```
Dim MyDouble1, MyDouble2 As Double
Dim MySingle1, MySingle2 As Single
MyDouble1 = 75.3421105
MyDouble2 = 75.3421567
MySingle1 = CSng(MyDouble1)   ' MySingle1 se establece a 75.34211.
MySingle2 = CSng(MyDouble2)   ' MySingle2 se establece a 75.34216.
```

Ejemplo de la función CStr

Este ejemplo utiliza la función CStr para convertir un valor numérico en String.

```
Dim MyDouble As Double
Dim MyString As String
MyDouble = 437.324
MyString = CStr(MyDouble)   ' MyString se establece a "437.324".
```

Este ejemplo utiliza la función CStr para convertir valores Date en valores String.

```
Dim MyDate As Date
Dim MyString As String
' ...
MyDate = #Febrero 12, 1969 00:00:00#   ' formato inválido.
' Los literales de fecha deben estar en formato #m/d/yyyy# o no será
válido.
' ...
MyDate = #2/12/69 00:00:00#   ' Medianoche.
' El valor neutral de hora 00:00:00 se elimina de la conversión
MyString = CStr(MyDate)   ' MyString se establece a "2/12/1969".
' ...
MyDate = #2/12/69 00:00:01#   ' Un segundo después de medianoche
' El componente de la hora pasa a ser parte del valor convertido
MyString = CStr(MyDate)    ' MyString es ahora "2/12/1969 12:00:01
AM".
```

Cstr siempre procesa un valor Date en el formato corto estándar de la configuración regional actual, por ejemplo, "15/6/2014 4:35:47 p.m.".

2.1. Función CType

Esta función es importante. Devuelve el resultado de convertir explícitamente una expresión a un tipo de datos, objeto, estructura, clase o interfaz.

```
CType(expression, typename)
```

Partes

expression

Cualquier expresión válida. Si el valor de *expression* está fuera del intervalo permitido por *typename* se produce un error.

typename

Cualquier expresión válida dentro de una cláusula **As** de una instrucción **Dim**, es decir, el nombre de cualquier tipo de datos, objeto, estructura, clase o interfaz.

Comentarios

Ctype se compila en línea, es decir, el código de conversión forma parte del código que evalúa la expresión. La ejecución es más rápida, ya que no existen llamadas a un procedimiento que realiza la conversión.

Ejemplo

En este ejemplo se utiliza la función **CType** para convertir una expresión al tipo de datos especificado.

```
Dim Minumero As Long
Dim Minuevonumero As Single
Minumero = 1000
Minuevonumero= CType(Minumero,Single)  'Se establece Minuevonumero a 1000.0
```

2.2. Más sobre conversiones

Hemos comentado que hay dos formas de cambiar de tipo de datos o como se dice en programación hacer un "casting". Visual Basic .NET proporciona una forma para hacer casting con CType que, como hemos visto, convierte un tipo de datos en otro.

La otra función que convierte un tipo en otro es DirectCat, pero con mejor rendimiento que Ctype. Entonces ¿por qué no utilizamos ésta? No lo hacemos porque DirectCat funciona de forma distinta y no realiza comprobación de si el tipo de datos coincide con el tipo de datos que se espera convertir.

CType incluye todas las funciones de conversión de Visual Basic: CBool, CByte, CChar, CDate, CDec, CDbl, CInt, CLng, CObj, CShort, CSng y CStr.

La principal diferencia entre los dos es que DirectCast solo funciona si el tipo de datos especificado y el tipo proporcionado en la expresión en tiempo de ejecución es el mismo. Esta diferencia sólo se presenta cuando estamos convirtiendo desde un tipo objeto a un tipo valor. Por ejemplo, la siguiente operación con DirectCast fallará porque el tipo del objeto O no es un entero:

```
Dim O As Object = 1.5
Dim I As Integer = DirectCast(O, Integer) 'Se produce un error en tiempo
de ejecución
Dim I As Integer = CType(O, Integer)         ' No produce un error
```

En cambio, en este caso si funcionará:

```
Dim O As Object = 1
Dim I As Integer = DirectCast(O, Integer)' Funciona
Dim I As Integer = CType(O, Integer)    ' Funciona pero es más lento
que DirectCast
```

Cuando estamos convirtiendo un objeto a un valor, DirectCast tiene mejor rendimiento que Ctype. El compilador de Visual Basic .NET genera 4 líneas de código para DirectCast. Sin embargo, utilizando CType el compilador genera una llamada a un método de conversión que necesita unas 100 líneas de código interno, esto es porque internamente llama a otros métodos. En aplicaciones donde el rendimiento es crítico la diferencia puede ser sustancial si se realiza un proceso masivo de cálculo.

3. Funciones de comprobación de tipos

Esta colección de funciones las vamos a utilizar en muchas ocasiones en nuestra interfaz. Por ejemplo, tenemos en nuestra interfaz una petición de datos entre los que se encuentra la fecha de nacimiento de la persona. Como es un valor obligatorio comprobamos si ha escrito algo el usuario, es decir:

```
dim mifecha as date
if dato_fecha.text="" then
    msgbox ("Debes escribir un valor en esta casilla")
else
    mifecha=cdate(dato_fecha.text)
    'seguimos proceso...
    ' ...
end if
```

En principio bien porque ha escrito un valor así que se lo asignaré a mi variable de trabajo "mifecha". Por eso la he convertido primero de string a fecha con "cdate" Pero... ¿y si el usuario ha escrito un valor que no es de tipo fecha? Nuestro programa fallará y mostrará una excepción o mensaje de error. Así que tenemos que controlar que además de escribir valores, el usuario los escriba correctamente. Para esto utilizaremos las funciones de comprobación de tipos. Sigamos con nuestro ejemplo... la función **Isdate** nos va a devolver True si puede convertir a fecha el dato que le demos, es decir, si es de tipo fecha, y false si no es de tipo fecha:

```
dim mifecha as date
    if dato_fecha.text="" then
        msgbox ("Debes escribir un valor en esta casilla")
    else
        if isdate (dato_fecha.text) then
            mifecha=cdate(dato_fecha.text)
            'seguimos proceso...
            ' ...
        else
           msgbox ("El valor introducido no es correcto")
        end if
    end if
```

Así que primero comprobamos que el usuario ha escrito un valor y luego si ese valor es de tipo fecha: "if isdate (dato_fecha.text) then"

Nota: En las instrucciones "If then..." se evalúa si se cumple o no una expresión y su resultado sólo puede ser True o False. Además las funciones de conversión de tipo devuelven igualmente true o false dependiendo de si pueden convertir o no la expresión al tipo de datos indicado. Así que es igual escribir esta expresión:

```
If isdate(dato_fecha.text)  <----> if isdate(dato_fecha.text)=true

If not isdate(dato_fecha.text)  <----> if isdate(dato_fecha.text)=false
```

Las funciones para comprobar los tipos de datos son:

3.1. IsNumeric (Función)

Devuelve un valor de tipo **Boolean** que indica si una expresión se puede evaluar como un número.

Sintaxis

IsNumeric(*expresión***)**

El argumento *expresión* requerido, es un tipo de datos que contiene una expresión numérica o una expresión de tipo cadena.

Comentarios

La función **IsNumeric** devuelve **True** si la *expresión* completa se reconoce como un número; en otro caso, devuelve **False**.

Ejemplo

En este ejemplo se utiliza la función IsNumeric para determinar si el contenido de una variable puede evaluarse como un número.

```
Dim MyVar As Object
Dim MyCheck As Boolean
' ...
MyVar = "53" ' Asignamos valor.
MyCheck = IsNumeric(MyVar) ' Devuelve True.
' ...
MyVar = "459.95" ' Asignamos valor.
MyCheck = IsNumeric(MyVar) ' Devuelve True.
' ...
MyVar = "45 Help" ' Asignamos valor.
MyCheck = IsNumeric(MyVar) ' Devuelve False.
```

3.2. IsArray (Función)

Devuelve un valor de tipo **Boolean** que indica si una variable es una matriz.

Sintaxis

IsArray(*nombrevariable***)**

El argumento requerido *nombrevariable,* es un identificador que especifica una variable.

Comentarios

La función **IsArray** devuelve **True** si la variable es una matriz; en caso contrario, devuelve **False**.

La función **IsArray** es especialmente útil en el caso de objetos que contienen matrices.

En este ejemplo se utiliza la función IsArray para comprobar si varias variables hacen referencia a una matriz.

```
Dim MyArray(4), YourArray(3) As Integer ' Declara variables de matriz.
Dim MyString As String
Dim MyCheck As Boolean
MyCheck = IsArray(MyArray) ' Devuelve True.
MyCheck = IsArray(YourArray) ' Devuelve True.
MyCheck = IsArray(MyString) ' Devuelve False.
```

3.3. IsDate (Función)

Devuelve un valor de tipo **Boolean** que indica si una expresión se puede convertir en una fecha.

Sintaxis

IsDate(*expresión***)**

El argumento *expresión* requerido, es un tipo de datos que contiene una expresión de fecha o una expresión de cadena reconocible como una fecha o una hora.

Comentarios

La función **IsDate** devuelve **True** si la expresión es una fecha o se puede reconocer como una fecha válida; en caso contrario, devuelve **False**. En Microsoft Windows, el intervalo de fechas válidas va desde el 1 de enero de 100 D. de C. hasta el 31 de diciembre de 9999 D.de C.; los intervalos varían de un sistema operativo a otro.

Ejemplo

En este ejemplo se utiliza la función IsDate para determinar si varias variables se pueden convertir a fechas.

```
Dim MyDate, YourDate As Date
Dim NoDate As String
```

```
Dim MyCheck As Boolean
MyDate = "February 12, 1969"
YourDate = #2/12/1969#
NoDate = "Hello"
MyCheck = IsDate(MyDate) ' Devuelve True.
MyCheck = IsDate(YourDate) ' Devuelve True.
MyCheck = IsDate(NoDate) ' Devuelve False.
```

3.4. IsNothing (Función)

Devuelve un valor de tipo **Boolean** que indica si una variable ha sido inicializada. Devuelve un valor de tipo Boolean que indica si una expresión no tiene ningún objeto asignado.

```
Public Function IsNothing(ByVal Expression As Object) As Boolean
```

Comentarios

`IsNothing` devuelve True si la expresión representa una variable de tipo Object que no tiene actualmente ningún objeto asignado; en caso contrario, devuelve False.

Ejemplo

En este ejemplo se utiliza la función IsNothing para determinar si una variable de objeto está asociada a alguna instancia de objeto.

```
Dim MyVar As Object ' No hay instancia asignada todavía.
Dim MyCheck As Boolean
' ...
MyCheck = IsNothing(MyVar) ' Devuelve True.
' ...
MyVar = "ABCDEF" ' Asignamos una instancia de string a la variable.
MyCheck = IsNothing(MyVar) ' Devuelve False.
' ...
MyVar = Nothing ' Quita la asociación de la variable desde cualquier instancia
MyCheck = IsNothing(MyVar) ' Devuelve True.
```

3.5. IsError (Función)

Devuelve un valor de tipo **Boolean** que indica si una expresión tiene un valor de error.

Sintaxis

`IsError(`*`expresión`***`)`**

El argumento *expresión* requerido, puede ser cualquier expresión válida.

Comentarios

Los valores de error se crean al convertir números reales a valores de error utilizando la función **CVErr**. La función **IsError** se utiliza para determinar si una expresión numérica representa un error. La función **IsError** devuelve **True** si el argumento *expresión* indica un error; en caso contrario, devuelve **False**.

Ejemplo:

En este ejemplo se utiliza la función IsError para comprobar si una expresión representa una excepción del sistema.

```
Dim ReturnVal As Object
Dim BadArg As String ' Nombre del argumento fuera de rango
Dim MyCheck As Boolean
' ...
ReturnVal = New System.ArgumentOutOfRangeException(BadArg)
' ...
MyCheck = IsError(ReturnVal) ' Devuelve True
```

3.6. IsDBNull (Función)

Devuelve un valor de tipo **Boolean** que indica si una expresión contiene datos no válidos (Null).

Sintaxis

IsNull(*expresión***)**

El argumento *expresión* requerido, es un tipo de datos que contiene una expresión numérica o una expresión de cadena.

Comentarios

IsDBNull devuelve **True** si el tipo de datos de *Expression* se evalúa como de tipo **DBNull**; si no, **IsDBNull** devuelve **False**.

El valor **System.DBNull** indica que la expresión de tipo **Object** representa datos que no se encuentran o no existen. **DBNull** no es lo mismo que **Nothing**, que indica que una variable no se ha inicializado todavía. Tampoco es lo mismo que una cadena de longitud cero (**""**), a la que a veces se hace referencia como cadena de valor null.

Ejemplo

En este ejemplo se utiliza la función IsDBNull para determinar si una variable se evalúa como DBNull.

```
Dim MyVar As Object
Dim MyCheck As Boolean
```

```
MyCheck = IsDBNull(MyVar) ' Devuelve False.
MyVar = ""
MyCheck = IsDBNull(MyVar) ' Devuelve False.
MyVar = System.DBNull.Value
MyCheck = IsDBNull(MyVar) ' Devuelve True.
MsgBox(MyCheck)
```

3.7. TypeName (Función)

Devuelve una cadena (**String)** que proporciona información acerca de una variable.

Sintaxis

TypeName(*nombreVariable***)**

El argumento *nombreVariable* requerido, es un tipo de datos que contiene cualquier variable excepto una variable de un tipo definido por el usuario.

Comentarios

En la siguiente tabla se muestran los valores **String** devueltos por **TypeName** para distintos contenidos de *VarName*:

Contenido *VarName*	Cadena devuelta
Tipo de valor **True** o **False** de 16 bits	"Boolean"
Valor binario de 8 bits	"Byte"
Carácter de 16 bits	"Char"
Valor de fecha y hora de 64 bits	"Date"
Tipo de referencia que indica que los datos faltan o no existen	"DBNull"
Valor numérico de punto fijo de 128 bits	"Decimal"
Valor numérico de punto flotante de 64 bits	"Double"
Valor entero de 32 bits	"Integer"
Referencia que apunta a un objeto no especializado	"Object"
Referencia que apunta a un objeto especializado creado a partir de la clase <*objectclass*>	"<*objectclass*>"
Valor entero de 64 bits	"Long"
Tipo de referencia sin ningún objeto actualmente asignado	"Nothing"
Valor entero de 16 bits	"Short"

Valor numérico de punto flotante de 32 bits	"Single"
Referencia que apunta a una cadena de caracteres de 16 bits	"String"

Si VarName es una matriz, la cadena devuelta puede ser cualquiera de las cadenas de la tabla anterior seguida de paréntesis vacíos. Por ejemplo, si VarName apunta a una matriz de enteros, TypeName devuelve "Integer()".

Cuando TypeName devuelve el nombre de un tipo de referencia, tal como una clase, sólo devuelve el nombre simple, no el nombre completo. Por ejemplo, si VarName apunta a un objeto de la clase System.Drawing.Printing.PaperSource, TypeName devuelve "PaperSource".

Ejemplo

En este ejemplo se utiliza la función TypeName para devolver información acerca del tipo de datos de varias variables.

```
Dim MyType As String
Dim StrVar As String = "MyString"
Dim DecVar As Decimal
Dim IntVar, ArrayVar(5) As Integer
MyType = TypeName(StrVar) ' Devuelve "String".
MyType = TypeName(IntVar) ' Devuelve "Integer".
MyType = TypeName(ArrayVar) ' Devuelve "Integer()".
```

3.8. VarType (Función)

Devuelve un entero (**Integer)** que indica el subtipo de una variable.

Sintaxis

VarType(*nombrevariable***)**

El argumento *nombrevariable* requerido, es un tipo que contiene cualquier variable excepto una variable de un tipo definido por el usuario.

El valor entero devuelto por **VarType** es un miembro de la enumeración **VariantType**. En la siguiente tabla se muestran los valores devueltos por **VarType** para casos especiales de *VarName*.

Tipo de datos representado por *VarName*	Valor devuelto por *VarType*
Nothing	**VariantType.Object**
DBNull	**VariantType.Null**
Enumeración	Tipo de datos subyacente (**Byte**, **Short**, **Integer** o **Long**)

Matriz	OR bit a bit de tipo de elemento de matriz y **VariantType.Array**
Matriz de matrices	OR bit a bit de **VariantType.Object** y **VariantType.Array**
Estructura (**System.ValueType**)	**VariantType.UserDefinedType**
System.Exception	**VariantType.Error**
Desconocido	**VariantType.Object**

Ejemplo

En este ejemplo se utiliza la función VarType para devolver información acerca de la clasificación del tipo de datos de varias variables.

```
Dim MyString As String = "MyString"
Dim MyObject As Object
Dim MyNumber, MyArray(5) As Integer
Dim MyVarType As VariantType    ' Enumeración de Integer
MyVarType = VarType(MyVarType)  ' Devuelve VariantType.Integer.
MyVarType = VarType(MyString)   ' Devuelve VariantType.String.
MyVarType = VarType(MyObject)   ' Devuelve VariantType.Object.
MyVarType = VarType(MyArray)    ' Devuelve la combinación de
                                VariantType.Array  y
                                VariantType.Integer.
```

4. Funciones aritméticas

En .NET disponemos de una clase llamada **Math** que proporciona constantes y métodos estáticos para operaciones trigonométricas, logarítmicas y otras funciones matemáticas comunes.

Los miembros de esta clase son:

Campos públicos	Descripción
E	Representa la base logarítmica natural, especificada por la constante, **e**
PI	Representa la relación entre la longitud de la circunferencia de un círculo y su diámetro, especificada por la constante π
Métodos públicos	**Descripción**
Abs	Sobrecargado. Devuelve el valor absoluto de un número especificado
Acos	Devuelve el ángulo cuyo coseno es el número especificado
Asin	Devuelve el ángulo cuyo seno es el número especificado
Atan	Devuelve el ángulo cuya tangente corresponde al número especificado

Atan2	Devuelve el ángulo cuya tangente es el cociente de dos números especificados
BigMul	Calcula el producto completo de dos números de 32 bits
Ceiling	Devuelve el número entero más pequeño mayor o igual que el número especificado.
Cos	Devuelve el coseno del ángulo especificado.
Cosh	Devuelve el coseno hiperbólico del ángulo especificado.
DivRem	Sobrecargado. Devuelve el cociente de dos números y pasa también como parámetro de salida el resto de la división.
Exp	Devuelve **e** elevado a la potencia especificada.
Floor	Devuelve el número entero más grande menor o igual que el número especificado
IEEERemainder	Devuelve el resto de la división de dos números especificados
Log	Sobrecargado. Devuelve el logaritmo de un número especificado.
Log10	Devuelve el logaritmo en base 10 de un número especificado.
Max	Sobrecargado. Devuelve el mayor de dos números especificados.
Min	Sobrecargado. Devuelve el menor de dos números.
Pow	Devuelve un número especificado elevado a la potencia especificada.
Round	Sobrecargado. Devuelve el número más próximo al valor especificado.
Sign	Sobrecargado. Devuelve un valor que indica el signo de un número.
Sin	Devuelve el seno del ángulo especificado.
Sinh	Devuelve el seno hiperbólico del ángulo especificado
Sqrt	Devuelve la raíz cuadrada de un número especificado.
Tan	Devuelve la tangente del ángulo especificado.
Tanh	Devuelve la tangente hiperbólica del ángulo especificado

Nos encontramos con que algunos métodos están "sobrecargados".

La palabra **sobrecargado** está dentro de la programación orientada a objetos. La sobrecarga como vimos en el tema anterior consiste en crear más de un procedimiento o función dentro de la misma clase con el mismo nombre y distintos argumentos. Veamos un ejemplo, el primero, la función ABS. Esta es una sencilla función que devuelve el valor absoluto de un número especificado, es decir, el mismo valor sin signo. Extraemos de la ayuda de .NET lo siguiente, que son la "Lista de sobrecargas":

- Devuelve el valor absoluto de un decimal:
  Overloads Public Shared Function Abs(Decimal) As Decimal
- Devuelve el valor absoluto de punto flotante de precisión doble:
  Overloads Public Shared Function Abs(Double) As Double
- Devuelve el valor absoluto de un entero de 16 bits con signo:
  Overloads Public Shared Function Abs(Short) As Short
- ...

Es decir es una función sobrecargada porque está declarada varias veces admitiendo argumentos distintos: decimal, double, short...

Vamos ver unos cuantos ejemplos de estas funciones matemáticas.

Ejemplo con ABS

```
Private m_longBase As Double
Private m_shortBase As Double
Private m_leftLeg As Double
Private m_rightLeg As Double
'Simplemente es procedimiento donde entran cuatro variables
' y se las asigna a las privadas definidas en valor absoluto
Public Sub Nuevo(ByVal longbase As Double, ByVal shortbase As _
Double, ByVal leftLeg As Double, ByVal rightLeg As Double)
  m_longBase = Math.Abs(longbase)
  m_shortBase = Math.Abs(shortbase)
  m_leftLeg = Math.Abs(leftLeg)
  m_rightLeg = Math.Abs(rightLeg)
End Sub
```

Ejemplo con la función Max

La función Max devuelve el mayor de dos números especificados, veamos su funcionamiento con distintos tipos de datos:

```
Public Shared Sub Main()
  Dim str As String = "{0}: El mayor de {1,3} y {2,3} es {3}."

  Dim xByte1 As Byte = 1
  Dim xByte2 As Byte = 51
  Dim xShort1 As Short = - 2
  Dim xShort2 As Short = 52
  Dim xInt1 As Integer = - 3
  Dim xInt2 As Integer = 53
  Dim xLong1 As Long = - 4
  Dim xLong2 As Long = 54
  Dim xSingle1 As Single = 5F
  Dim xSingle2 As Single = 55F
  Dim xDouble1 As Double = 6.0
  Dim xDouble2 As Double = 56.0
  Dim xDecimal1 As [Decimal] = 7D
  Dim xDecimal2 As [Decimal] = 57D

  ' Los siguientes tipos no se pueden utilizar:
  ' SByte, UInt16, UInt32, y UInt64.

  Console.WriteLine("Muestra el mayor de dos valores:")
  Console.WriteLine(str, "Byte ", xByte1, xByte2, Math.Max(xByte1,
xByte2))
  Console.WriteLine(str, "Int16 ", xShort1, xShort2,
Math.Max(xShort1, xShort2))
```

```
  Console.WriteLine(str, "Int32 ", xInt1, xInt2, Math.Max(xInt1,
xInt2))
  Console.WriteLine(str, "Int64 ", xLong1, xLong2, Math.Max(xLong1,
xLong2))
  Console.WriteLine(str, "Single ", xSingle1, xSingle2,
Math.Max(xSingle1, xSingle2))
  Console.WriteLine(str, "Double ", xDouble1, xDouble2,
Math.Max(xDouble1, xDouble2))
  Console.WriteLine(str, "Decimal", xDecimal1, xDecimal2,
Math.Max(xDecimal1, xDecimal2))
'
  Console.WriteLine("No están soportados los siguientes tipos")
  Console.WriteLine("SByte, UInt16, UInt32, y UInt64.")
End Sub 'Main
'
'La salida de esté programa produce:
'
'Muestra el mayor de dos valores:
'
'Byte : El mayor de 1 y 51 es 51.
'Int16 : El mayor de -2 y 52 es 52.
'Int32 : El mayor de -3 y 53 es 53.
'Int64 : El mayor de -4 y 54 es 54.
'Single : El mayor de 5 y 55 es 55.
'Double : El mayor de 6 y 56 es 56.
'Decimal: El mayor de 7 y 57 es 57.
'
```

Ejemplo con la función Round

Esta función muestra el redondeo al entero más próximo:

Math.Round(3.44, 1) 'Devuelve 3.4.

Math.Round(3.45, 1) 'Devuelve 3.4.

Math.Round(3.46, 1) 'Devuelve 3.5.

Ejemplo con la función Pow

Esta función devuelve un número elevado a una potencia. Necesita dos parámetros, la base y el exponente.

La base es "x" y el exponente "y". En esta tabla muestra los resultados si x=0 o "NaN" (valor no numérico), NegativeInfinity (infinito negativo) o PositiveInfinity (infinito positivo)

Valores del parámetro	Devuelve
x o *y* es igual a Double.NaN	NaN.
x es igual a Double.NegativeInfinity	NegativeInfinity si *y* es un entero impar, en caso contrario, PositiveInfinity.
y es igual a Double.NegativeInfinity	0.

x es igual a Double.PositiveInfinity	0 si *y* es igual a NegativeInfinity; en caso contrario, PositiveInfinity.
y es igual a Double.PositiveInfinity	PositiveInfinity.

Y aplicado en una función:

```
Public Function GetHeight() As Double
    Dim x As Double = GetRightSmallBase()
    GetHeight = Math.Sqrt(Math.Pow(m_rightLeg, 2) - Math.Pow(x, 2))
End Function
```

4.1. Funciones matemáticas derivadas

A continuación, se muestra una lista de funciones matemáticas que pueden derivarse de las funciones matemáticas que hemos visto antes:

Función	Funciones derivadas equivalentes
Secante (Sec(x))	1 / **Cos(**x**)**
Cosecante (Csc(x))	1 / **Sin(**x**)**
Cotangente (Ctan(x))	1 / **Tan(**x**)**
Seno inverso (Asin(x))	**Atan(**x / **Sqrt(**-x * x + 1**))**
Coseno inverso (Acos(x))	**Atan(**-x / **Sqrt(**-x * x + 1**))** + 2 * **Atan(**1**)**
Secante inversa (Asec(x))	2 * **Atan(**1**) – Atan(Sign(**x**)** / **Sqrt(**x * x **–** 1**))**
Cosecante inversa (Acsc(x))	**Atan(Sign(**x**)** / **Sqrt(**x * x **–** 1**))**
Cotangente inversa (Acot(x))	2 * **Atan(**1**)** - **Atan(**x**)**
Seno hiperbólico (Sinh(x))	(**Exp(**x**) – Exp(**-x**)**) / 2
Coseno hiperbólico (Cosh(x))	(**Exp(**x**)** + **Exp(**-x**)**) / 2
Tangente hiperbólica (Tanh(x))	(**Exp(**x**) – Exp(**-x**)**) / (**Exp(**x**)** + **Exp(**-x**)**)
Secante hiperbólica (Sech(x))	2 / (**Exp(**x**)** + **Exp(**-x**)**)
Cosecante hiperbólica (Csch(x))	2 / (**Exp(**x**) – Exp(**-x**)**)
Cotangente hiperbólica (Coth(x))	(**Exp(**x**)** + **Exp(**-x**)**) / (**Exp(**x**) – Exp(**-x**)**)
Seno hiperbólico inverso (Asinh(x))	**Log(**x + **Sqrt(**x * x + 1**))**
Coseno hiperbólico inverso (Acosh(x))	**Log(**x + **Sqrt(**x * x – 1**))**
Tangente hiperbólica inversa (Atanh(x))	**Log(**(1 + x) / (1 – x)**)** / 2
Secante hiperbólica inversa (AsecH(x))	**Log(**(**Sqrt(**-x * x + 1**)** + 1) / x**)**
Cosecante hiperbólica inversa (Acsch(x))	**Log(**(**Sign(**x**)** * **Sqrt(**x * x + 1**)** + 1) / x**)**
Cotangente hiperbólica inversa (Acoth(x))	**Log(**(x + 1) / (x – 1)**)** / 2

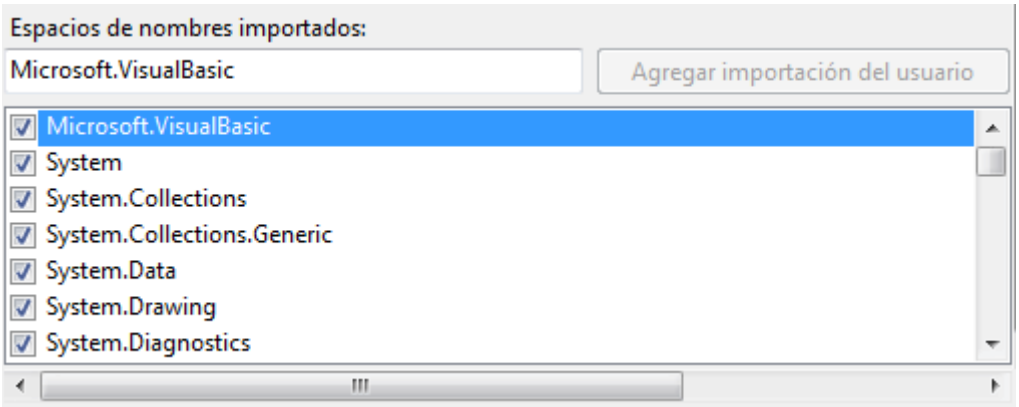
5. Funciones de cadenas de caracteres.

Este es otro de los grupos más importantes ya que las utilizaremos frecuentemente. En nuestra interfaz trabajaremos casi siempre con cadenas de caracteres, en algunas ocasiones tendremos que realizar conversiones de tipos como las que hemos visto y en otras realizar operaciones dentro de las cadenas como extraer subcadenas, contar caracteres, ver su contenido, ...

Al igual que las funciones aritméticas anteriores las funciones de cadenas se encuentran definidas dentro del Spacename o espacio de nombres **Microsoft.VisualBasic.** Este espacio de nombres tiene todas las clases y módulos de Visual Basic .NET. Todo esto viene a que tenemos una función Left que nos devuelve una subcadena de caracteres, pues bien, la palabra "Left" también se utiliza para otras cosas, por ejemplo, es la propiedad que le indica la posición izquierda de los controles. Para distinguir las dos funciones o propiedades y, en general, para resolver este pequeño problema debemos utilizar el nombre completo de la función es decir, su espacio de nombre y la función:

```
Microsoft.VisualBasic.Left
```

Aunque si recuerdas hay una serie de espacios de nombres ya importados por defecto a nuestros proyectos, esto lo veíamos en las propiedades del proyecto:



Vemos que este espacio de nombres está importado así que podemos escribir estas funciones tal y como están explicadas a continuación. Otro detalle a tener en cuenta es que como hemos visto al principio del tema, la clase "string" ya tenía muchos métodos para la manipulación de las cadenas, aquí vemos las instrucciones como funciones generales no como métodos de la clase string. A nuestro gusto está utilizarlas de una forma (como método de la clase string o directamente como veremos ahora) u otra, las dos son correctas.

Por coherencia con esta nueva forma de programación de objetos, deberíamos trabajar con la clase "string" y lo que vamos a ver ahora serían métodos suyos. Veamos ahora las funciones más importantes:

5.1. Len (cadena)

Devuelve un número entero que contiene el número de caracteres de la cadena.

Sintaxis:

Function Len(ByVal *Expression* **As** { **Boolean** | **Byte** | **Char** | **Double** | **Integer** | **Long** | **Object** | **Short** | **Single** | **String** | **DateTime** | **Decimal** } **) As Integer**

Ejemplo:

```
dim micadena as string
dim longitud as integer

micadena="hola mundo”
longitud=len(micadena)
```

5.2. Left (cadena)

Devuelve una cadena que contiene un número especificado de caracteres desde el lado izquierdo de la cadena. Si se usa en un formulario Windows Forms o en cualquier otra clase que tenga una propiedad **Left**, debemos calificar plenamente la función con Microsoft.VisualBasic.Left.

Sintaxis:

```
Function Left (Byval str as String, Byval Longitud as Integer) as String
```

Ejemplo

```
Dim myString As String = "Hola Mundo"
Dim subString As String
subString = Microsoft.VisualBasic.Left(myString, 4) ' Devuelve "Hola"
```

5.3. Right

Devuelve una cadena que contiene un número especificado de caracteres desde el lado derecho de la cadena. Si se usa en un formulario Windows Forms o en cualquier otra clase que tenga una propiedad **Right**, debemos calificar plenamente la función con Microsoft.VisualBasic.Right.

Sintaxis:

Function Right(ByVal *Str* **As String, ByVal** *Length* **As Integer) As String**

Ejemplo

```
Dim myString As String = "Hola Mundo"
Dim subString As String
subString = Microsoft.VisualBasic.Right(myString, 5) ' Devuelve "Mundo"
```

5.4. Mid

Devuelve una cadena que a su vez contiene un número especificado de caracteres de una cadena.

Sintaxis

Function Mid(ByVal *Str* **As String, ByVal** *Start* **As Integer, Optional ByVal** *Length* **As Integer) As String**

Parámetros:

- ***Str.*** Requerido. Expresión **String** de la que se devuelven caracteres.
- ***Start.*** Requerido. Expresión **Integer**. Posición de carácter de *Str* donde comienza la parte que se va a utilizar. Si *Start* es mayor que el número de caracteres de *Str*, la función **Mid** devuelve una cadena de longitud cero (""). *Start* es de base uno.
- ***Length.*** Opcional. Expresión **Integer**. Número de caracteres que se van a devolver. Si se omite o si existen menos caracteres del número especificado por *Length* en el texto (incluido el carácter en la posición *Start*), se devuelven todos los caracteres desde la posición de inicio hasta el final de la cadena

Ejemplo

En este ejemplo se utiliza la función Mid para devolver un número de caracteres especificado de una cadena.

```
Dim MyString, FirstWord, LastWord, MidWords As String
MyString = "Demo de la función MID"          ' Crea una cadena de texto
FirstWord = Mid(MyString, 1, 4)              ' Devuelve "Demo"
LastWord = Mid(MyString, 19, 3)              ' Devuelve "MID"
MidWords = Mid(MyString, 11)                 ' Devuelve "función MID"
```

5.5. Space

Devuelve una cadena que consta del número especificado de espacios.

Sintaxis

Function Space(ByVal *Number* **As Integer) As String**

Ejemplo

En este ejemplo se utiliza la función **Space** para devolver una cadena formada por un número de espacios especificado.

```
Dim MyString As String
' Devuelve una cadena de caracteres con 10 espacios
MyString = Space(10)
' Inserta 10 espacios entre dos cadenas
MyString = "Hola" & Space(10) & "Mundo"
```

5.6. InStr, InStrRev

Devuelve un entero que especifica la posición inicial de la primera aparición de una cadena dentro de otra

Sintaxis

InStr(Optional ByVal *Start* **As Integer, ByVal** *String1* **As String, ByVal** *String2* **As String) As Integer**

Los parámetros son:

- ***Start.*** Opcional. Expresión numérica que establece la posición inicial para cada búsqueda. Si se omite, la búsqueda comienza en la primera posición del carácter. El índice de inicio está basado en 1.
- ***String1.*** Requerido. Expresión **String** en la que se busca.
- ***String2.*** Requerido. Expresión **String** que se busca.

Valores devueltos

Si...	InStr devuelve
String1 es de longitud cero o **Nothing**	0
String2 es de longitud cero o **Nothing**	*start*
String2 no se encuentra	0
String2 se encuentra dentro de *String1*	Posición donde empieza la coincidencia
Start > *String2*	0

Ejemplo

En este ejemplo se usa la función **InStr** para devolver la posición de la primera aparición de una cadena dentro de otra.

```
Dim SearchString, SearchChar As String
Dim MyPos As Integer

SearchString ="XXpXXpXXPXXP"    ' Cadena donde buscar.
SearchChar = "P"                ' Buscamos la "P".

' Una comparación de texto a partir de la posición 4 revuelve 6
MyPos = InStr(4, SearchString, SearchChar)
MyPos = InStr(1, SearchString, "W")   ' Devuelve 0
```

InStrRev es igual que la vista aquí pero la comparación se realiza por el final

5.7. Replace

Devuelve una cadena en la que la subcadena especificada se reemplaza determinado número de veces por otra subcadena

Sintaxis

```
Public Function Replace(ByVal Expression As String, ByVal Find As String,
                        ByVal Replacement As String, Optional ByVal Start As Integer = 1, _
                        Optional ByVal Count As Integer = -1, Optional ByVal Compare _
                        As CompareMethod = CompareMethod.Binary ) As String
```

Parámetros:

- ***Expression.*** Requerido. Expresión de cadena que contiene la subcadena que se va a reemplazar.
- ***Find.*** Requerido. Subcadena que se busca.
- ***Replacement.*** Requerido. Subcadena de reemplazo.
- ***Start.*** Opcional. Posición, dentro de Expression, donde debe empezar la búsqueda de la subcadena. Si se omite, se supone que es 1.
- ***Count.*** Opcional. Número de sustituciones de subcadenas que se deben realizar. Si se omite, el valor predeterminado será –1, con lo que se harán todas las sustituciones posibles.
- ***Compare.*** Opcional. Valor numérico que indica el tipo de comparación que se va a utilizar en la evaluación de subcadenas.

El argumento ***Compare*** puede tener los siguientes valores:

Constante	Descripción
Binary	Realiza una comparación binaria
Text	Realiza una comparación textual

Ejemplo

```
Dim myString As String = "Lista de compra"
Dim aString As String
' Devuelve "Lista de compra".
aString = Replace(myString, "n", "a")
```

5.8. Ltrim, Rtrim, Trim

Devuelve una cadena que contiene una copia de una cadena dada sin espacios iniciales (**LTrim**), sin espacios finales (**RTrim**) o sin espacios iniciales ni finales (**Trim**).

Sintaxis

```
Function LTrim(ByVal Str As String) As String
Function RTrim(ByVal Str As String) As String
Function Trim(ByVal Str As String) As String
```

Ejemplo

En este ejemplo se utiliza la función LTrim para suprimir los espacios iniciales y la función RTrim para suprimir los espacios finales en una variable de cadena. Se utiliza la función Trim para suprimir ambos tipos de espacios.

```
Dim MyString, TrimString As String
MyString = " <-Trim-> "         ' Inicializa el string
TrimString = LTrim(MyString)            ' TrimString = "<-Trim-> ".
TrimString = RTrim(MyString)            ' TrimString = " <-Trim->".
TrimString = LTrim(RTrim(MyString))     ' TrimString = "<-Trim->".
'Utilizando únicamente la función Trim obtenemos el mismo resultado:
TrimString = Trim(MyString)             ' TrimString = "<-Trim->".
```

5.9. UCase, LCase

Devuelve una cadena o un carácter que contiene la cadena especificada convertida en mayúsculas en Ucase y en minúsculas con LCase

Sintaxis

```
Function UCase(ByVal Value As String) As String
Function LCase(ByVal Value As String) As String
```

Ejemplo

En este ejemplo se utiliza la función UCase para devolver la versión en mayúsculas de una cadena.

```
Dim LowerCase, UpperCase As String

LowerCase = "Hola Mundo 1234"           ' Cadena a convertir.
UpperCase = UCase(LowerCase)            ' Devuelve "HOLA MUNDO 1234"
```

5.10. Format

Devuelve una cadena con el formato que especifiquen las instrucciones contenidas en una expresión **String** de formato

Sintaxis

```
Function Format(ByVal Expression As Object, _
Optional ByVal Style As String = "" ) As String
```

La tabla siguiente muestra los nombres de formato numérico predefinidos. Éstos pueden usarse por nombre como argumento de estilo para la función **Format**:

Nombre de formato	Descripción
General Number, **G** o **g**	Muestra el número sin separadores de miles.
Currency, **C** o **c**	Muestra el número con separadores de miles, en su caso; también muestra dos dígitos a la derecha del separador de decimales. El formato de salida dependerá de la configuración regional.
Fixed, **F** o **f**	Muestra al menos un dígito a la izquierda y dos a la derecha del separador de decimales.
Standard, **N** o **n**	Muestra el número con separador de miles, al menos un dígito a la izquierda y dos a la derecha del separador de decimales.
Percent	Muestra el número multiplicado por 100 con un signo de porcentaje (%) a la derecha; siempre muestra dos dígitos a la derecha del separador de decimales.
P o **p**	Muestra el número con separador de miles multiplicado por 100 con un signo de porcentaje (%) a la derecha y separado por un solo espacio; siempre muestra dos dígitos a la derecha del separador de decimales.
Científico	Utiliza notación científica estándar y proporciona dos dígitos significativos.
E o **e**	Utiliza notación científica estándar y proporciona seis dígitos significativos.
D o **d**	Muestra el número como una cadena que contiene el valor del número en formato Decimal (base 10). Esta opción sólo se admite para tipos integrales **(Byte**, **Short**, **Integer**, **Long**).
X o **x**	Muestra el número como una cadena que contiene el valor del número en formato Hexadecimal (base 16). Esta opción sólo se admite para tipos integrales **(Byte**, **Short**, **Integer** y **Long**).
Yes/No	Muestra No si el número es 0; de lo contrario, muestra Yes.
True o False	Muestra False si el número es 0; de lo contrario, muestra True.
On/Off	Muestra Off si el número es 0; de lo contrario, muestra On.

La siguiente tabla identifica caracteres que puede usar para crear formatos de número definidos por el usuario. Éstos pueden usarse para generar el argumento de estilo correspondiente a la función **Format**:

Carácter	Descripción
Ninguno	Muestra el número sin formato alguno.
(**0**)	Marcador de posición de dígito. Muestra un dígito o un cero. Si la expresión tiene un dígito en la posición donde aparece el cero en la cadena de formato, éste se mostrará así; de lo contrario, aparecerá un cero en esa posición.

	Si el número tiene menos dígitos que ceros (a cualquier lado del separador decimal) en la expresión de formato, se mostrarán ceros iniciales o finales. Si el número tiene más dígitos a la derecha del separador decimal que ceros en la expresión de formato, se redondeará el número a tantos decimales como ceros haya. Si el número tiene más dígitos a la derecha del separador decimal que ceros en la expresión de formato, se redondeará el número a tantos decimales como ceros haya.
(#)	Marcador de posición de dígito. Muestra un dígito o nada. Si la expresión tiene un dígito en la posición donde aparece el carácter # en la cadena de formato, se muestra; de lo contrario, no aparece nada en esa posición. Este símbolo funciona como el marcador de posición de dígito **0**, salvo que los ceros iniciales y finales no se mostrarán si el número contiene menos dígitos que caracteres # a cualquiera de los lados del separador decimal en la expresión de formato.
(**.**)	Marcador de posición decimal. El marcador de posición decimal determina cuántos dígitos se mostrarán a la izquierda y derecha del separador decimal. Si la expresión de formato sólo contiene caracteres # a la izquierda de este símbolo, los números inferiores a 1 empezarán con un separador decimal. Par mostrar un cero inicial con números fraccionarios, use el cero como el primer marcador de posición digital a la izquierda del separador decimal. El uso del punto o la coma como separador decimal depende de la configuración regional en cada caso. El mismo carácter utilizado como marcador decimal en virtud del formato de salida dependerá del formato de número reconocido por su sistema. Por tanto, deberá usar el punto como marcador decimal en sus formatos, incluso aunque su configuración regional utilice la coma como separador decimal. La cadena con formato se mostrará con el formato correcto para la configuración regional correspondiente.
(**%)**	Marcador de posición de porcentaje. Multiplica la expresión por 100. El carácter de porcentaje (%) se inserta en la misma posición en la que aparece en la cadena de formato.
(**,**)	Separador de miles. El separador de miles separa las unidades de millar de las centenas con un número que presente cuatro o más dígitos a la izquierda del separador decimal. Se especificará un uso estándar del separador de miles si el formato contiene un separador de miles rodeado de marcadores de posición de dígito (**0** ó #). Un separador de miles situado inmediatamente a la izquierda del separador decimal (se especificará si se trata de un decimal o no) o como el carácter más a la derecha de la cadena significa "reducir el número dividiéndolo por 1000 y redondeándolo en caso necesario". Por ejemplo, puede utilizar el formato de cadena "##0,." para representar 100 millones como 100,000. Los números menores que 1,000 pero mayores o iguales que 500 se muestran como 1; los menores que 500 se muestran como 0. Dos separadores de miles adyacentes en esta posición se reducen por un factor de 1 millón, más otro factor adicional de 1000 por cada separador adicional. En el caso de los separadores múltiples en cualquier posición que no sea inmediatamente a la izquierda del separador decimal o la posición más a la derecha de la cadena, se interpretará que simplemente especifican el uso de un separador de miles. El uso del punto o la coma como separador de miles depende de la configuración regional en cada caso. El mismo carácter utilizado como separador de miles en virtud del formato de salida dependerá del formato de número reconocido por su sistema. Por tanto, deberá usar la coma como marcador de miles en sus

	formatos, incluso aunque su configuración regional utilice el punto como marcador de miles. La cadena con formato se mostrará con el formato correcto para la configuración regional correspondiente.
(**:**)	Separador de hora. En ciertas configuraciones regionales, pueden usarse otros caracteres para representar el separador de hora. Este separador horario separa horas, minutos y segundos cuando se da formato a valores horarios. El carácter real utilizado es el especificado como separador de hora en la configuración de su sistema.
(***/***)	Separador de fecha. En ciertas configuraciones regionales, pueden usarse otros caracteres para representar el separador de fecha. Este separador separa el día, mes y año cuando se da formato a los valores de fecha. El carácter real utilizado es el especificado como separador de fecha en la configuración de su sistema.
(**E- E+ e- e+**)	Formato científico. Si la expresión de formato contiene al menos un marcador de posición de dígito (**0** o #) a la izquierda de **E-**, **E+**, **e-** o **e+**, el número se mostrará en formato científico y se insertará E o e entre el número y su exponente. El número de marcadores de posición digitales a la izquierda determina el número de dígitos en el exponente. Use **E-** o **e-** para colocar un signo menos junto a los exponentes negativos. Use **E+** o **e+** para colocar un signo menos junto a los exponentes negativos y un signo más junto a los positivos. También deberá incluir marcadores de posición digitales a la derecha de este símbolo para obtener un formato correcto.
- + $ ( )	Caracteres literales. Estos caracteres se mostrarán exactamente como se escriben en la cadena de formato. Para mostrar un carácter distinto de los listados, precédase de una barra invertida (\) o escríbase entre comillas (" ").
(****)	Muestra el siguiente carácter de una cadena de formato. Para mostrar un carácter dotado de un significado especial como carácter literal, éste debe ir precedido de una barra invertida (****). La barra invertida en sí no aparecerá. El uso de una barra invertida equivale a incluir el siguiente carácter entre comillas. Para mostrar una barra invertida, úsense dos (****). Los caracteres de formato de fecha y los caracteres de formato de hora (**a**, **c**, **d**, **h**, **m**, **n**, **p**, **q**, **s**, **t**, **w**, **y**, ***/*** y **:**), los caracteres de formato numérico (#, **0**, **%**, **E**, **e**, la coma y el punto) y los caracteres de formato de cadena (@, **&**, **<**, **>** y **!**) son ejemplos de caracteres que no se pueden mostrar como caracteres literales.
(**"ABC"**)	Muestra la cadena entre comillas (" "). Si se desea insertar una cadena en el argumento de estilo desde el código, deberá usar **Chr(34)** para incluir el texto (**34** es el código de caracteres correspondiente a las comillas dobles (")).

Ejemplo

La tabla siguiente contiene algunas muestras de expresiones de formato correspondientes a números. En estos ejemplos se presupone que la configuración regional del sistema es Inglés (Estados Unidos). La primera columna contiene las cadenas de formato correspondientes al argumento *Style* de la función **Format**; las otras columnas contienen el formato de salida resultante si los datos con formato contienen el valor asignado en los encabezados de columna.

Formato (*Style*)	"5" con formato como	"-5" con formato como	"0.5" con formato como
Zero-length string ("")	5	-5	0.5

0	5	-5	1
0.00	5.00	-5.00	0.50
#,##0	5	-5	1
$#,##0;($#,##0)	$5	($5)	$1
$#,##0.00; ($#,##0.00)	$5.00	($5.00)	$0.50
0%	500%	-500%	50%
0.00%	500.00%	-500.00%	50.00%
0.00E+00	5.00E+00	-5.00E+00	5.00E-01
0.00E-00	5.00E00	-5.00E00	5.00E-01

La tabla siguiente identifica los nombres de formatos de fecha y hora predefinidos. Éstos pueden usarse por nombre como argumento de estilo para la función **Format**:

Nombre de formato	Descripción
General Date o **G**	Muestra una fecha o una hora. En el caso de números reales, muestra una fecha y una hora; por ejemplo, 4/3/93 05:34 PM. Si no existe parte fraccional, muestra sólo una fecha; por ejemplo, 4/3/93. Si no existe parte entera, muestra sólo una hora; por ejemplo, 05:34 PM. El formato de fecha depende del valor **LocaleID** del sistema.
Long Date o **D**	Muestra una fecha de acuerdo con el formato de fecha larga vigente en su sistema.
Medium Date	Muestra una fecha usando el formato medio que corresponda a la versión de idioma que use la aplicación host.
Short Date o **d**	Muestra una fecha de acuerdo con el formato de fecha corta vigente en su sistema.
Long Time o **T**	Muestra una hora de acuerdo con el formato de fecha larga vigente en su sistema; e incluye horas, minutos y segundos.
Medium Time	Muestra la hora en formato de 12 horas utilizando horas y minutos y la especificación a.m./p.m.
Short Time o **t**	Muestra una hora con el formato de 24 horas, por ejemplo, 17:45.
f	Muestra la fecha larga y la hora corta de acuerdo con el formato vigente en su sistema.
F	Muestra la fecha larga y la hora larga de acuerdo con el formato vigente en su sistema.
g	Muestra la fecha corta y la hora corta de acuerdo con el formato vigente en su sistema.
M, **m**	Muestra el mes y el día de una fecha dada.
R, **r**	Da formato a la fecha y la hora como Hora media de Greenwich (GMT)
s	Da formato a la fecha y la hora como un índice ordenable.
u	Da formato a la fecha y la hora como un índice GMT ordenable.

U	Da formato como GMT a la fecha larga y la hora larga.
Y, y	Da formato a la fecha especificando el año y el mes

En la siguiente tabla se muestran los caracteres que se pueden utilizar para crear formatos de fecha y hora definidos por el usuario. A diferencia de versiones anteriores de Visual Basic, estos caracteres de formato distinguen mayúsculas de minúsculas.

Carácter	Descripción
(**:**)	Separador de hora. En ciertas configuraciones regionales, pueden usarse otros caracteres para representar el separador de hora. Este separador horario separa horas, minutos y segundos cuando se da formato a valores horarios. El carácter real utilizado como separador de hora en los resultados con formato viene determinado por el valor **LocaleID** del sistema.
(*/*)	Separador de fecha. En ciertas configuraciones regionales, pueden usarse otros caracteres para representar el separador de fecha. Este separador separa el día, mes y año cuando se da formato a los valores de fecha. El carácter real utilizado es el especificado como separador de fecha en la configuración local de su sistema.
(**%**)	Se utiliza para indicar que el carácter siguiente debe leerse como formato de una sola letra sin tener en cuenta las posibles letras finales. También se emplea para indicar que un formato de una sola letra se lea como formato definido por el usuario.
d	Muestra el día como un número sin cero a la izquierda (por ejemplo, 1). Usa **%d** si es el único carácter en el formato numérico definido por el usuario.
dd	Muestra el día como un número con cero a la izquierda (por ejemplo, 01).
ddd	Muestra el día de forma abreviada (por ejemplo, Dom).
dddd	Muestra el día de forma completa (por ejemplo, Domingo).
M	Muestra el mes como un número sin cero a la izquierda (por ejemplo, enero se representa como 1). Use **%M** si es el único carácter en el formato numérico definido por el usuario.
MM	Muestra el mes como un número con cero a la izquierda (por ejemplo, 01/12/01), doce de enero de 2001.
MMM	Muestra el mes en forma abreviada (por ejemplo, Ene).
MMMM	Muestra el mes en forma completa (por ejemplo, Enero).
gg	Especifica la era histórica (por ejemplo, A.D.)
h	Muestra la hora como un número sin ceros a la izquierda y en formato de doce horas (por ejemplo, 1:15:15 PM). Usa **%h** si es el único carácter en el formato numérico definido por el usuario.
hh	Muestra la hora como un número con ceros a la izquierda y en formato de doce horas (por ejemplo, 01:15:15 PM).

H	Muestra la hora como un número sin ceros a la izquierda y en formato de doce horas (por ejemplo, 1:15:15 PM). Usa **%H** si es el único carácter en el formato numérico definido por el usuario.
HH	Muestra la hora como un número con ceros a la izquierda y en formato de doce horas (por ejemplo, 01:15:15).
m	Muestra los minutos como un número sin ceros a la izquierda (por ejemplo, 12:1:15). Usa **%m** si es el único carácter en el formato numérico definido por el usuario.
mm	Muestra los minutos como un número con ceros a la izquierda (por ejemplo, 12:01:15).
s	Muestra los segundos como un número sin ceros a la izquierda (por ejemplo, 12:15:5). Usa **%s** si es el único carácter en el formato numérico definido por el usuario.
ss	Muestra los segundos como un número con ceros a la izquierda (por ejemplo, 12:15:05).
F	Muestra fracciones de segundos. Por ejemplo, **ff** muestra centésimas de segundo, mientras que **ffff** muestra diez milésimas de segundo. Puede utilizar hasta siete símbolos **f** en el formato definido por el usuario. Use **%f** si es el único carácter en el formato numérico definido por el usuario.
T	Usa el reloj de doce horas; muestra una A mayúscula para cualquier hora entre medianoche y mediodía, y una P mayúscula para cualquier hora entre mediodía y medianoche. Utilice **%t** si éste es el único carácter del formato numérico definido por el usuario.
tt	Usa el reloj de doce horas y muestra la leyenda AM en mayúsculas para cualquier hora entre medianoche y mediodía; y PM en mayúsculas para cualquier hora entre mediodía y medianoche.
y	Muestra el año sin cero inicial. Utilice **%y** en caso de que éste sea el único carácter en su formato numérico definido por el usuario.
yy	Muestra el año en formato numérico de dos dígitos sin cero inicial, si procede.
yyy	Muestra el año en formato numérico de cuatro dígitos.
yyyy	Muestra el año en formato numérico de cuatro dígitos.
z	Muestra el desplazamiento de zona horaria sin cero a la izquierda (por ejemplo, -8). Use **%z** si es el único carácter en el formato numérico definido por el usuario.
zz	Muestra el desplazamiento de zona horaria con un cero a la izquierda (por ejemplo, -08).
zzz	Muestra el desplazamiento completo de zona horaria (por ejemplo, -08:00).

Ejemplo

A continuación se muestran algunos ejemplos de formatos de hora y fecha definidos por el usuario y correspondientes al 7 de diciembre de 1958, a las ocho horas, cincuenta minutos y treinta y cinco segundos de la tarde (Diciembre 7, 1958, 8:50 PM, 35 segundos):

Formato	Muestra
M/d/yy	12/7/58
d-MMM	7-Dic
d-MMMM-yy	7-Diciembre-58
d MMMM	7 Diciembre
MMMM yy	Diciembre 58
hh:mm tt	08:50 PM
h:mm:ss t	8:50:35 P
H:mm	20:50
H:mm:ss	20:50:35
M/d/yyyy H:mm	12/7/1958 20:50

Ejemplo

En este ejemplo se muestran los diversos usos de la función Format para dar formato a valores tanto con formatos String como otros definidos por el usuario. Para el separador de fecha (/), hora (:) e indicadores de a.m./p.m. (t y tt), el formato de salida que muestre su sistema dependerá de la configuración regional que use el código. Cuando las horas y fechas se muestren en el entorno de desarrollo, se utilizará el formato de fecha y hora corta de la configuración regional del código.

```
Dim MyDateTime As Date = #1/27/2001 5:04:23 PM#
Dim MyStr As String
' Devuelve la hora del sistema en formato de hora larga.
MyStr = Format(Now(), "Long Time")
' Devuelve la fecha del sistema en formato de fecha largo
MyStr = Format(Now(), "Long Date")
' También devuelve la fecha del sistema con formato definido por el
usuario
' Utilizando el caracter D.
MyStr = Format(Now(), "D")
' Devuelve el valor de MyDateTime con formato date/time definido por el
usuario
MyStr = Format(MyDateTime, "h:m:s")         ' Devuelve "5:4:23".
MyStr = Format(MyDateTime, "hh:mm:ss tt")   ' Devuelve "05:04:23 PM".
MyStr = Format(MyDateTime, "dddd, MMM d yyyy") 'Sábado, 27  Enero de
2001.
MyStr = Format(MyDateTime, "HH:mm:ss")      ' Devuelve "17:04:23"
MyStr = Format(23)                          ' Devuelve "23".
' Formatos numéricos definidos por el usuario
MyStr = Format(5459.4, "##,##0.00")         ' Devuelve "5,459.40".
MyStr = Format(334.9, "###0.00")            ' Devuelve "334.90".
MyStr = Format(5, "0.00%")                  ' Devuelve "500.00%".
```

6. Funciones de fechas

Visual Basic .NET reemplaza **Date** y **Time** por las propiedades **Today** y **TimeOfDay**, que utilizan la estructura **DateTime** de ocho bytes. Esto se corresponde con el tipo de datos **Date** en Visual Basic .NET. Ahora veremos estas dos funciones junto a otras de tratamiento de fechas.

6.1. Today

Devuelve o establece un valor **Date** que contiene la fecha actual de acuerdo con el sistema.

El tipo de datos **Date** incluye componentes de tiempo. Al devolver la fecha del sistema, **Today** los establece todos en 0, de tal forma que el valor devuelto representa medianoche (00:00:00). Al establecer la fecha del sistema, **Today** omite los componentes de hora.

Para obtener acceso a la fecha actual del sistema como un valor **String**, se utiliza la propiedad **DateString**.

La propiedad **TimeOfDay** se utiliza para obtener o establecer la hora actual del sistema.

Ejemplo

```
Dim MiFecha As Date
MiFecha = Today
```

6.2. Now

Devuelve un valor **Date** que contiene la fecha y la hora actuales de acuerdo con el sistema. Se utiliza la propiedad **Today** para establecer la fecha del sistema. La propiedad **TimeOfDay** se utiliza para establecer la hora del sistema

Ejemplo

En este ejemplo se utiliza la propiedad Now para mostrar la fecha y hora de sistema actual.

```
Dim Ahora As Date
Ahora= Now ' Establece la fecha y hora actual del sistema
```

6.3. DateString

Devuelve o establece un valor **String** que representa la fecha actual de acuerdo con el sistema.
DateString siempre devuelve la fecha del sistema como "MM-dd-yyyy", es decir, utilizando el nombre de mes abreviado. Los formatos aceptados para configurar la fecha son "M-d-yyyy", "M-d-y", "M/d/yyyy" y "M/d/y".
Si se intenta establecer **DateString** con un valor no válido, se produce un error **InvalidCastException**.

Para obtener o establecer la hora actual del sistema como un valor **String**, se utiliza la propiedad **TimeString**. Para obtener acceso a la fecha actual del sistema como un valor **Date**, se utiliza la propiedad **Today**

Ejemplo

En este ejemplo se utiliza la propiedad **DateString** para mostrar la fecha de sistema actual.

```
MsgBox("La fecha actual es: " & DateString)
```

6.4. TimeOfDay

Devuelve o establece un valor **Date** que contiene la hora del día actual de acuerdo con el sistema

Ejemplo

En este ejemplo se utiliza la propiedad **TimeOfDay** para mostrar la hora del sistema actual.
El tipo de datos **Date** incluye componentes de fecha. Al devolver la hora del sistema, **TimeOfDay** los establece todos en 1, de tal forma que el valor devuelto representa el primer día del año 1. Al establecer la hora del sistema, **TimeOfDay** omite los componentes de fecha.
Para obtener acceso a la hora actual del sistema como un valor **String**, se utiliza la propiedad **TimeString**.

```
Dim hora_actual As Date
Hora_actual = TimeOfDay    ' Devuelve la hora actual del sistema.
```

6.5. TimeString

Devuelve o establece un valor **String** que representa la hora del día actual de acuerdo con el sistema.

TimeString devuelve siempre la hora del sistema como "HH:mm:ss", que es un formato de 24 horas. Si intentamos establecer **TimeString** con un valor no válido, se produce un error **InvalidCastException**.

Para obtener o establecer la fecha actual del sistema como un valor **String**, se utiliza la propiedad **DateString**. Para obtener acceso a la hora actual del sistema como un valor **Date**, se utiliza la propiedad **TimeOfDay**

Ejemplo

En este ejemplo se utiliza la propiedad **TimeString** para mostrar la hora de sistema actual.

```
MsgBox("La hora actual es: " & TimeString)
```

6.6. Year

Devuelve un valor **Integer** entre 1 y 9999 que representa el año.

Ejemplo

En este ejemplo se utiliza la función **Year** para obtener el año de una fecha especificada. En el entorno de desarrollo, el literal de fecha se muestra en formato corto de fecha con los valores de configuración regional del código correspondiente.

```
Dim fecha As Date
Dim anio As Integer
fecha = #2/12/1969#      ' Asigna una fecha.
anio = Year(fecha)       ' Asigna el año 1969
```

6.7. Month

Devuelve un valor **Integer** entre 1 y 12 que representa el mes del año

Ejemplo

En este ejemplo se utiliza la función **Month** para obtener el mes de una fecha especificada. En el entorno de desarrollo, el literal de fecha se muestra en formato corto de fecha con los valores de configuración regional del código correspondiente.

```
Dim MyDate As Date
Dim MyMonth As Integer
MyDate = #2/12/1969#            ' Asigna una fecha.
MyMonth = Month(MyDate)                 ' Asigna el valor 2
```

6.8. Day

Devuelve un valor **Integer** entre 1 y 31 que representa el día del mes

Ejemplo

En este ejemplo se utiliza la función **Day** para obtener el día del mes de una fecha especificada. En el entorno de desarrollo, el literal de fecha se muestra en formato corto estándar (por ejemplo "12/02/1969") con los valores de configuración regional del código correspondiente.

```
Dim fecha As Date
Dim dia As Integer
fecha = #2/12/1969#   'Asigna una fecha utilizando el formato estándar
corto.
dia = Day(fecha)    ' dia contiene 12.
```

Day se califica para distinguirlo de la enumeración **System.Windows.Forms.Day**

6.9. WeekDay

Devuelve un valor **Integer** que contiene un número que representa el día de la semana.

El valor devuelto por la función **Weekday** corresponde a los valores de la enumeración **FirstDayOfWeek**; es decir, 1 indica domingo y 7 indica sábado.

Si *DayOfWeek* es inferior a 0 o mayor que 7, se produce un error **ArgumentException**.

Nota: **Weekday** utiliza la configuración de calendario actual de la propiedad **CurrentCulture** de la clase **CultureInfo** en el espacio de nombres **System.Globalization**. Los valores **CurrentCulture** predeterminados están determinados por la configuración del **Panel de control**.

Sintaxis

```
Function Weekday(  ByVal DateValue As
DateTime,  Optional ByVal DayOfWeek As FirstDayOfWeek
     = FirstDayOfWeek.Sunday ) As Integer
```

Argumentos

- ***DateValue.*** Requerido. Valor **Date** del cual se desea determinar el día de la semana.
- ***DayOfWeek.*** Opcional. Valor elegido de la enumeración **FirstDayOfWeek** que especifica el primer día de la semana. Si no se especifica ningún valor, se utiliza **FirstDayOfWeek.Sunday**.

El argumento *DayOfWeek* puede tener uno de los siguientes valores:

Valor de enumeración	Valor	Descripción
FirstDayOfWeek.System	0	Primer día de la semana especificado en la configuración del sistema
FirstDayOfWeek.Sunday	1	Domingo (predeterminado)
FirstDayOfWeek.Monday	2	Lunes (de acuerdo con la norma ISO 8601, sección 3.17)
FirstDayOfWeek.Tuesday	3	Martes
FirstDayOfWeek.Wednesday	4	Miércoles
FirstDayOfWeek.Thursday	5	Jueves
FirstDayOfWeek.Friday	6	Viernes
FirstDayOfWeek.Saturday	7	Sábado

Ejemplo

En este ejemplo se utiliza la función **Weekday** para obtener el día de la semana de una fecha especificada.

```
Dim fecha As Date
```

```
Dim dia_semana As Integer
fecha = #2/12/1978#          'Asigna una fecha
dia_semana = Weekday(MyDate) 'dia_semana contiene 4
```

6.10. WeekDayName

Devuelve un valor **String** que contiene el nombre del día de la semana especificado.

```
Public Function WeekdayName( ByVal WeekDay As Integer, _
Optional ByVal Abbreviate As Boolean  =  False,  _
Optional ByVal FirstDayOfWeekValue As FirstDayOfWeek = FirstDayOfWeek.System _
) As String
```

Parámetros

- ***WeekDay.*** Requerido. **Integer**. Designación numérica del día de la semana, entre 1 y 7; 1 indica el primer día de la semana y 7 indica el último día de la semana. Las identidades del primer y último día dependen de la configuración de *FirstDayOfWeekValue*.
- ***Abbreviate.*** Opcional. Valor **Boolean** que indica si se abrevia el nombre del día de la semana. Si se omite, el valor predeterminado es **False**, que significa que el nombre del día de la semana no se abrevia.
- ***FirstDayOfWeekValue.*** Opcional. Valor elegido de la enumeración **FirstDayOfWeek** que especifica el primer día de la semana. Si no se especifica ningún valor, se utiliza **FirstDayOfWeek.System**.

Configuración

El argumento *FirstDayOfWeekValue* puede tener uno de los siguientes valores:

Valor de enumeración	Valor	Descripción
FirstDayOfWeek.System	0	Primer día de la semana especificado en la configuración del sistema (valor predeterminado)
FirstDayOfWeek.Sunday	1	Domingo
FirstDayOfWeek.Monday	2	Lunes (de acuerdo con la norma ISO 8601, sección 3.17)
FirstDayOfWeek.Tuesday	3	Martes
FirstDayOfWeek.Wednesday	4	Miércoles
FirstDayOfWeek.Thursday	5	Jueves
FirstDayOfWeek.Friday	6	Viernes
FirstDayOfWeek.Saturday	7	Sábado

Comentarios

La cadena devuelta por **WeekdayName** no depende únicamente de los argumentos de entrada, sino también de los valores de la Configuración regional especificados en el Panel de control de Windows.

Si *WeekDay* es menor que 1 o mayor que 7, o si *FirstDayOfWeekValue* es menor que 0 o mayor que 7, se produce un error **ArgumentException**.

Nota: **WeekdayName** utiliza la configuración de calendario actual de la propiedad **CurrentCulture** de la clase **CultureInfo** en el espacio de nombres **System.Globalization**. Los valores **CurrentCulture** predeterminados están determinados por la configuración del **Panel de control**

6.11. MonthName

Devuelve un valor **String** que contiene el nombre del mes especificado.

Public Function MonthName(ByVal *Month* **As Integer, Optional ByVal** *Abbreviate* **As Boolean = False) As String**

Parámetros

- ***Month.*** Requerido. **Integer**. Designación numérica del mes, entre 1 y 13; 1 indica el mes de enero y 12 indica el mes de diciembre. Se puede utilizar el valor 13 con un calendario de 13 meses. Si el sistema está utilizando un calendario de 12 meses y *Month* es 13, **MonthName** devuelve una cadena vacía.
- ***Abbreviate.*** Opcional. Valor **Boolean** que indica si se va a abreviar el nombre del mes. Si se omite, el valor predeterminado es **False**, que significa que el nombre del mes no se abrevia.

Excepciones o errores

Tipo de excepción	Número de error	Condición
ArgumentException	5	*Month* es menor que 1 o mayor que 13.

Comentarios

La cadena devuelta por **MonthName** no depende únicamente de los argumentos de entrada, sino también de los valores de la Configuración regional especificados en el Panel de control de Windows.

Si *Month* es inferior a 1 o mayor que 13, se produce un error **ArgumentException**.

Nota: **MonthName** utiliza la configuración de calendario actual de la propiedad **CurrentCulture** de la clase **CultureInfo** en el espacio de nombres **System.Globalization**. Los valores **CurrentCulture** predeterminados están determinados por la configuración del **Panel de control**.

Ejemplo

En este ejemplo se utiliza la función **MonthName** para determinar el nombre del mes, a partir del entero dado. El valor Boolean determinará si se muestra el nombre completo (**False**) o el nombre abreviado (**True**).

```
Dim mes As Integer
Dim nombre As String
mes = 4
nombre = MonthName(mes, True)  ' "True" devuelve un valor abreviado.
Msgbox(nombre)                      ' nombre contiene "Abr".
```

6.12. DatePart

Devuelve un valor **Integer** que contiene el componente especificado de un valor **Date** dado.

```
Public Overloads Function DatePart(ByVal Interval As DateInterval, _
ByVal DateValue As DateTime, _
Optional ByVal FirstDayOfWeekValue As FirstDayOfWeek = VbSunday, _
   Optional ByVal FirstWeekOfYearValue As FirstWeekOfYear = VbFirstJan1 _
) As Integer
```

O bien

```
Public Overloads Function DatePart(ByVal Interval As String,
ByVal DateValue As Object, _
   Optional ByVal DayOfWeek As FirstDayOfWeek = FirstDayOfWeek.Sunday, _
   Optional ByVal WeekOfYear As FirstWeekOfYear
= FirstWeekOfYear.Jan1) As Integer
```

Parámetros

- ***Interval.*** Requerido. Valor de enumeración **DateInterval** o expresión **String** que representa la parte del valor de fecha u hora que se desea devolver.
- ***DateValue.*** Requerido. Valor **Date** que se desea evaluar.
- ***FirstDayOfWeekValue.*** Opcional. Valor elegido de la enumeración **FirstDayOfWeek** que especifica el primer día de la semana. Si no se especifica ningún valor, se utiliza **FirstDayOfWeek.Sunday**.
- ***FirstWeekOfYearValue*.** Opcional. Valor elegido de la enumeración **FirstWeekOfYear** que especifica la primera semana del año. Si no se especifica ningún valor, se utiliza **FirstWeekOfYear.Jan1**.

Configuración

El argumento *Interval* puede tener uno de los siguientes valores:

Valor de enumeración	Cadena	Parte del valor de fecha u hora devuelta
DateInterval.Day	d	Día del mes (del 1 al 31)
DateInterval.DayOfYear	y	Día del año (del 1 al 366)
DateInterval.Hour	h	Hora
DateInterval.Minute	n	Minuto
DateInterval.Month	m	Mes
DateInterval.Quarter	q	Trimestre
DateInterval.Second	s	Segundo
DateInterval.Weekday	w	Día de la semana (del 1 al 7)
DateInterval.WeekOfYear	ww	Semana del año (de la 1 a la 53)
DateInterval.Year	yyyy	Año

El argumento *FirstDayOfWeekValue* puede tener uno de los siguientes valores:

Valor de enumeración	Valor	Descripción
FirstDayOfWeek.System	0	Primer día de la semana especificado en la configuración del sistema
FirstDayOfWeek.Sunday	1	Domingo (predeterminado)
FirstDayOfWeek.Monday	2	Lunes (de acuerdo con la norma ISO 8601, sección 3.17)
FirstDayOfWeek.Tuesday	3	Martes
FirstDayOfWeek.Wednesday	4	Miércoles
FirstDayOfWeek.Thursday	5	Jueves
FirstDayOfWeek.Friday	6	Viernes
FirstDayOfWeek.Saturday	7	Sábado

El argumento *FirstWeekOfYearValue* puede tener uno de los siguientes valores:

Valor de enumeración	Valor	Descripción
FirstWeekOfYear.System	0	Primera semana del año especificada en la configuración del sistema
FirstWeekOfYear.Jan1	1	Semana en la que se encuentra el 1 de enero (predeterminado)
FirstWeekOfYear.FirstFourDays	2	Semana que contiene al menos cuatro días del nuevo año (de acuerdo con la norma ISO 8601, sección 3.17)
FirstWeekOfYear.FirstFullWeek	3	Primera semana completa del nuevo año

Excepciones o errores

Tipo de excepción	Número de error	Condición
ArgumentException	5	*Interval* no es válido.
InvalidCastException	13	*DateValue* no se puede convertir a **Date**.

Comentarios

Se puede utilizar la función **DatePart** para evaluar un valor de fecha u hora y devolver un componente específico. Por ejemplo, se podría utilizar **DatePart** para calcular el día de la semana o la hora actual.

Si se elige **DateInterval.Weekday** para el argumento *Interval*, el valor devuelto es coherente con los valores de la enumeración **FirstDayOfWeek**. Si se elige **DateInterval.WeekOfYear**, **DatePart** utiliza las clases **Calendar** y **CultureInfo** del espacio de nombres **System.Globalization** para determinar la configuración actual.

El argumento *FirstDayOfWeekValue* afecta a los cálculos que utilizan los valores **DateInterval.Weekday** y **DateInterval.WeekOfYear** para *Interval*. El argumento *FirstWeekOfYearValue* afecta a los cálculos que especifican **DateInterval.WeekOfYear** para *Interval*.

Si algún argumento tiene un valor no válido, se produce un error **ArgumentException**. Si el argumento *DateValue* tiene un valor que no puede convertirse a un valor **Date** válido, se produce un error **InvalidCastException**.

Puesto que todos los valores **Date** se basan en una estructura **DateTime**, sus métodos proporcionan opciones adicionales para recuperar partes de fecha u hora. Por ejemplo, se puede obtener el valor de fecha completo de una variable **Date**, con el valor de hora establecido en medianoche, como se muestra a continuación:

```
Dim Ahora As Date = Now                    'Fecha y hora actuales
Dim ultima_Medianoche As Date = Ahora.Date      'Medianoche.
```

Ejemplo

En este ejemplo se utiliza la función **DatePart** sobre una fecha para mostrar el trimestre del año en que se produce.

```
Dim primera_fecha, Msg As String   'Declara variables.
Dim segunda_fecha As Date
primera_fecha = InputBox("Introduce una fecha:")
segunda_fecha = CDate(primera_fecha)
Msg = "Trimestre: " & DatePart(DateInterval.Quarter, segunda_fecha)
MsgBox (Msg)
```

6.13. Hour

Devuelve un valor **Integer** entre 0 y 23 que representa la hora del día.

Ejemplo

En este ejemplo se utiliza la función **Hour** para obtener la hora de una hora especificada. En el entorno de desarrollo, el literal de hora se muestra en formato de hora corto con los valores de configuración regional del código correspondiente.

```
Dim Tiempo As Date
Dim hora As Integer
tiempo = #4:35:17 PM#    ' Asigna una hora
hora = Hour(tiempo)   ' MyHour contiene 16.
```

6.14. Minute

Devuelve un valor **Integer** entre 0 y 59 que representa el minuto de la hora

Ejemplo

En este ejemplo se utiliza la función **Minute** para obtener el minuto de una hora especificada. En el entorno de desarrollo, el literal de hora se muestra en formato de hora corto con los valores de configuración regional del código correspondiente.

```
Dim tiempo As Date
Dim minuto As Integer
tiempo = #4:35:17 PM#    ' Asigna una hora.
minuto = Minute(hora)   ' hora contiene 35.
```

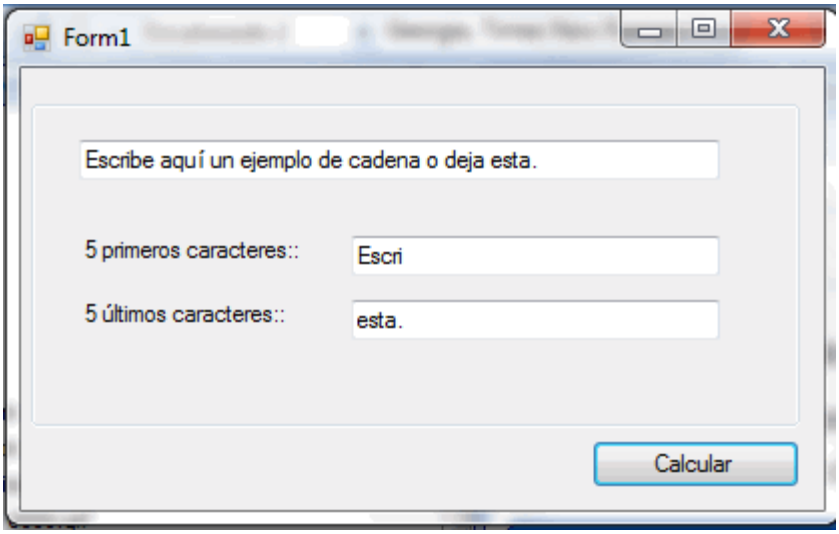
Ejercicios

Ejercicio 1

Haz un programa Windows que lea una cadena de caracteres en un cuadro de texto y escriba en otros:

- Los 5 primeros caracteres
- Los 5 últimos caracteres

Pon nombres a los controles para sustituir a los nombres que pone el IDE y así mejorar la legibilidad del código.



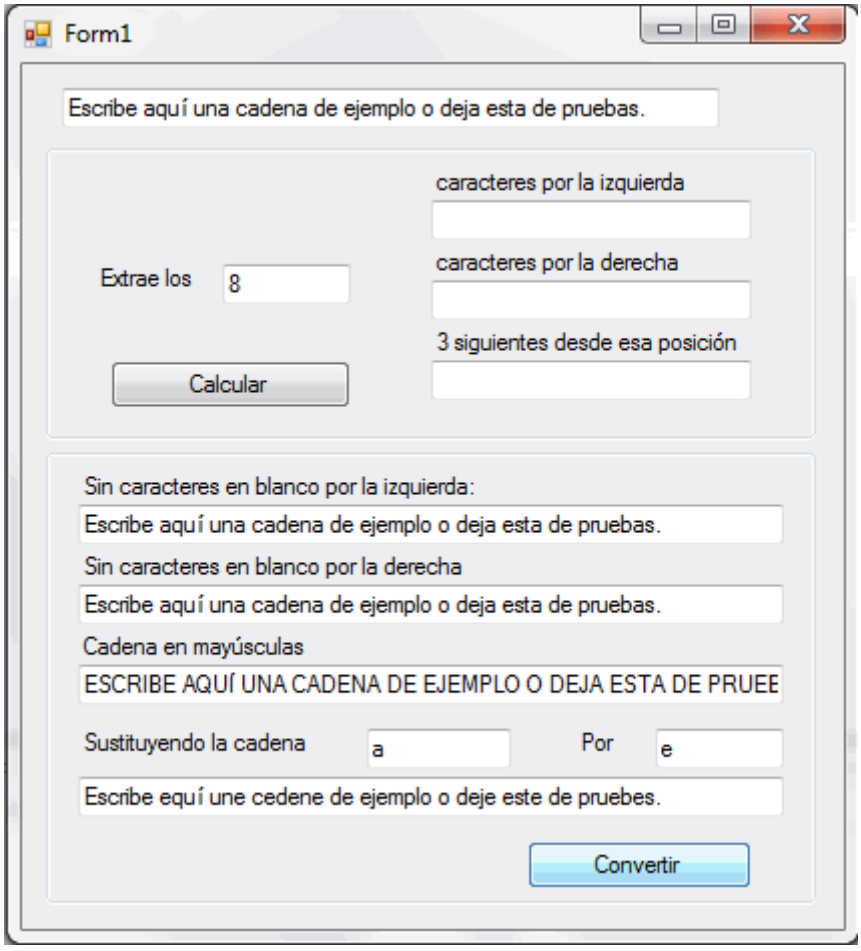
Nota: Recuerda que para escribir el código debes localizar el evento clic del botón:

```
Private Sub btn_calcular_Click( sender As Object, e As EventArgs) Handles btn_calcular.Click
    'Escribe aquí el código...
End Sub
```

Ejercicio 2

Haz un programa Windows que lea una cadena de caracteres en un cuadro de texto y un valor numérico (n) y escriba en otros:

- Los "n" primeros caracteres
- Los "n" últimos caracteres
- Los 3 siguientes caracteres a la posición "n"
- Convertir la cadena a mayúsculas
- Que quite lo espacios en blanco por la izquierda
- Que quite lo espacios en blanco por la derecha
- Que sustituya una cadena por una propuesta. Por ejemplo la letra "a" por la "e"

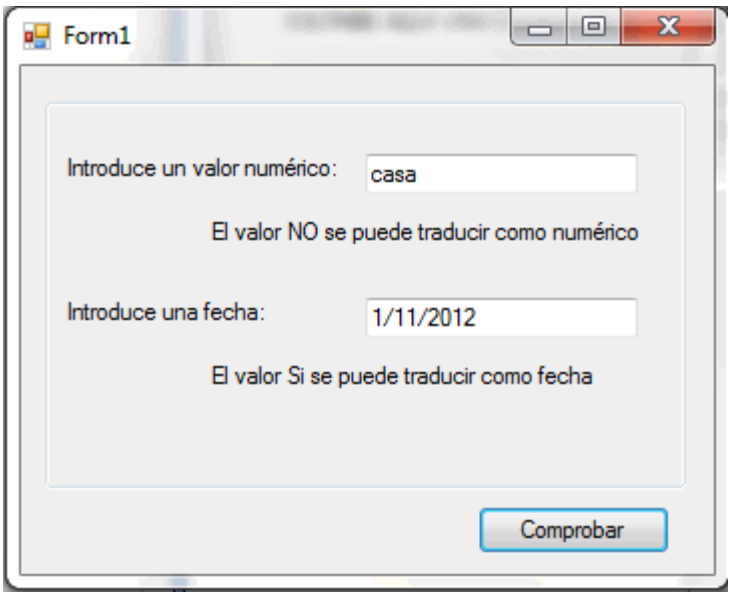


Nota: Convierte a numérico el valor leído en n para trabajar más cómodamente con él.

Ejercicio 3

Haz un programa Windows que lea dos valores y que diga si el valor introducido se puede traducir como:

- Entero
- Fecha



Ejercicio 4

Haz un programa Windows que lea una fecha (por defecto que escriba la fecha de hoy), comprueba que es correcta y que extraiga: el día, mes, año, día de la semana y nombre del mes.
Haz lo mismo con la hora.