Factory Soft Venezuela C.A.

Manual de Expresiones LIF

Versión 1.1

04 de abril de 2013

Historial de Cambios

Versión	Fecha	Observaciones	Responsable
V1.0	04/04/2013	Creación del Manual	RJG
V1.1	17/04/2013	Se agregaron funciones adicionales de formato de número y de fecha.	RJG

Introducción

[Contenido]

Valores Literales

Son expresiones que representan un valor directo (constante).

Tipo	Formato	Ejemplos
Cadena	`"[^"]*" | '[^']*'`	`""`
		`"Cadena Válida ñ"`
		`'_(válïda'€) 3+6'`
Numero	`d+[.d*]|.[d]+`	`1234` `12.345` `.123`
Fecha	`#AAAAMMDD[ HH:mm[:ss[.fff]]#`	`#20083112#`
		`#20083112 01:05#`
		`#20083112 23:59#`
		`#20083112 23:59:59#`
		`#20083112 23:59:59.999#`
Lógico	**`TRUE`**`|`**`FALSE`**	**`TRUE`** **`FALSE`**
Nulo	**`NULL`**	**`NULL`**
No Número	**`NAN`**	**`NAN`**

Observaciones:

- Los literales de cadena pueden estar delimitados por comillas simples o dobles.
- Un literal de cadena delimitado por comillas simples no puede contener comillas simples.
- Un literal de cadena delimitado por comillas dobles no puede contener comillas dobles.
- Ningún literal de cadena puede contener caracteres de "Salto de Linea" (ASCII 10) ni "Retorno de carro" (ASCII 13). El carácter de "Tabulación" (ASCII 9) está permitido, pero se recomienda usar una secuencia de escape, como se indica a continuación.
- Para usar comillas simples, comillas dobles, avance de línea, retorno de carro, y tabulaciones, en un literal de cadena se admiten las siguientes secuencias de escape:

Carácter	Código de Escape	Ejemplo	Salida
'	\q	'La mejor opción: \qeFactory\q'	La Mejor Opción: 'eFactory'
"	\Q	"La mejor opción: \QeFactory\Q"	La Mejor Opción: "eFactory"
ENTER	\n|\n\r	"La mejor opción:\n\reFactory"	La Mejor Opción: eFactory
TAB	\t	"La mejor opción:\teFactory"	La Mejor Opción: eFactory

- **Nota:** En sistemas Linux, Unix, Amiga, MAC y similares se utiliza el carácter de `Salto de Línea` (ASCII 10) como estándar para "`Nueva Línea`" o "`ENTER`"; en DOS y Windows, por extensión de las viejas máquinas de escribir que dieron origen al teclado, se utiliza una combinación de dos caracteres para indicar una "`Nueva Línea`" o "`Enter`": un `Salto de Línea` (ASCII 10) seguido de un `Retorno de Carro` (ASCII 13). Se recomienda usar solo la primera forma, es decir, para insertar un "Enter" en un literal de cadena se recomienda hacerlo como sigue: `"La mejor opción:\neFactory"`

Operadores

Operadores Aritméticos

Solo esperan operandos numéricos o tratan todos los operandos como números. El resultado esperado es Numérico, pero algunas operaciones devuelven **NAN** o **NULL**.

Símbolo	Operación	Ejemplo
^	Potencia	`2^5 = 32`
+	Identidad (no hace nada)	`+4.2 = 4.2`
	Suma	`5 + 4.2 = 9.2`
-	Negativo	`-5 = -5`
	Resta	`5 - 4.2 = 0.8`
*	Multiplicación	`5 * 4.2 = 21.0`
/	División	`7.1 / 4 = 1.75`
DIV	División Entera `A DIV B = Piso(A / B)`	`7.1 DIV 4 = 1`
MOD	División Modular (Resto) `A MOD B = A - B * Piso(A`	`7.1 MOD 4 = 3.1`

	/ B)	
&	Concatenación	"alfa " & "beta" = "alfa beta" 2.5 & ":" & NULL & TRUE = "2.5:true"

Observaciones:

- Las operaciones numéricas no válidas (ej. 0/0) devuelven **NAN**.
- Si uno de los operandos es **NAN**, o es de tipo Fecha el resultado es **NAN**.
- Si uno de los operandos es de tipo Cadena, será convertido en numérico si tiene el formato correcto, o se evaluará como 0 en caso contrario.
- El valor especial **NULL** es evaluado como 0.
- El valor lógico **TRUE** es evaluado como 1. El valor lógico **FALSE** es evaluado como 0.
- El operador de concatenación acepta cualquier tipo de operando, pero si no son cadenas los convierte de acuerdo a las siguientes reglas:
  - Los valores de tipo Decimal se representan con un punto como separador decimal y sin separador de miles, e.g. "1234.56".
  - Los valores de tipo fecha se representan con el formato completo, e.g. "YYYYMMDD HHmmss.fff".
  - Los valores Lógicos **TRUE** y **FALSE** se representan por su nombre en minúsculas: "true" "false"
  - Los valores **NAN** y **NULL** se representan por una cadena vacía: "".

Aunque el operador de concatenación haga una conversión automática es recomendable usar una de las funciones de formato.

Operadores de Comparación

Pueden manejar varios tipos de operandos. El resultado esperado es Lógico, pero algunas comparaciones devuelven **NAN** o **NULL**.

Símbolo	Comparación	Ejemplo
==	Igualdad	3.0 == 3 → **TRUE**
!=	Desigualdad	3.0 != 3 → **FALSE**
>	Mayor que	#20090101# > #20080101# → **TRUE**
<	Menor que	"alfa" < "beta" → **TRUE**
>=	Mayor o igual que	4 >= 4 → **TRUE**

`<=`	Menor o igual que	`8.5 <= 4` → **FALSE**
LIKE	Parecido a	`"alfabeto" LIKE "a[a-z]{2}b"` → **TRUE**

Observaciones:

- Si los operandos son ambos de tipo Decimal, Cadena o Fecha son comparados de forma habitual.
- En las cadenas, el orden depende del código ASCII de los caracteres, eg. `"A"<"a"`, `"19A"<"1A"` y `"."<"5"`.
- Si los operandos son lógicos, se considera **TRUE > FALSE**.
- Para los valores especiales **NAN** y **NULL**, se considera **NAN==NAN**, **NULL==NULL** y **NULL<NAN**.
- Al comparar valores de tipo Decimal con valores tipo Lógico, el valor lógico se considera `1` si es **TRUE** y `0` si es **FALSE**.
- El operador **LIKE** considera **NAN==""** y **NULL==""**. Si alguno de los operandos es de tipo Decimal, Fecha o Lógico, siempre devuelve **FALSE**. El segundo operando debe ser una *Expresión Regular* válida.
- Los casos de comparación (excepto para **LIKE**) se resumen en la siguiente tabla:

Tipo	Decimal	Cadena	Lógico	Fecha	NAN	NULL
Decimal	OK	devuelve **NAN**	devuelve **NAN**	devuelve **NAN**	devuelve **NAN**	**NULL == 0**
Cadena	devuelve **NAN**	OK	**LEN(C) > 0** → **TRUE** **LEN(C) == 0** → **FALSE**	devuelve **NULL**	**NAN == ""**	**NULL == ""**
Lógico	devuelve **NAN**	**LEN(C) > 0** → **TRUE** **LEN(C) == 0** → **FALSE**	**TRUE > FALSE**	devuelve **NULL**	**NAN == FALSE**	**NULL == FALSE**
Fecha	devuelve **NAN**	devuelve **NULL**	devuelve **NULL**	OK	devuelve **NAN**	**NULL == #19000101#**
NAN	devuelve **NAN**	**NAN == ""**	**NAN == FALSE**	devuelve **NAN**	**NAN == NAN**	**NAN > NULL**
NULL	**NULL == 0**	**NULL == ""**	**NULL == FALSE**	**NULL == #19000101#**	**NAN > NULL**	**NULL == NULL**

Operadores Lógicos

Solo esperan operandos de tipo lógico o tratan todos los operandos como lógicos. El resultado de estas operaciones es siempre de tipo lógico.

Símbolo	Operación	Ejemplo
NOT	Negación Lógica	**NOT(5 > 3)** → **FALSE**

AND	Conjunción Lógica	`(5 >= 3) AND (3 <= 3)` → **TRUE**
XOR	Disyunción Lógica Exclusiva	**TRUE XOR** `(5 > 3)` → **FALSE**
OR	Disyunción Lógica	**TRUE OR** `(5 > 3)` → **TRUE**

Observaciones:

- Al usar operadores lógicos, los valores de tipo no lógico son convertidos automáticamente según la siguiente tabla:

Tipo	Valor Lógico
Decimal	`Sii(D <> 0; TRUE; FALSE)`
Cadena	`Sii(Longitud(C) > 0; TRUE; FALSE)`
Fecha	`Sii(F > #19000101#; TRUE; FALSE)`
NAN	**FALSE**
NULL	**FALSE**

Funciones LIF

Son funciones básicas, que se aplican sobre los cinco tipos básicos de valores usados en LIF.

Funciones Numéricas

Maximo(vParametro1; vParametro2 [; vParametroN]*) → VALOR

Descripción: Devuelve el mayor de los parámetros. Se asume que todos los parámetros (y el resultado) son de mismo tipo que el primero de ellos. Ej.

```
Maximo(4.1; 3; -2) = 4.1
Maximo("alfa";"gamma";"beta")="gamma"
Maximo(#20081031#; #20081225#)= #20081225#
```

Parámetros:

1. **vParametro1, vParametro2…**: Dos o más valores numéricos, de fecha, de cadena, lógicos, **NaN** o **Null**.

Minimo(vParametro1; vParametro2 [; vParametroN]*) → VALOR

Descripción: Devuelve el menor de los parámetros. Se asume que todos los parámetros (y el resultado) son de mismo tipo que el primero de ellos. Ej.

```
Minimo(4.1; 3; -2) = -2
Minimo("alfa";"gamma";"beta")="alfa"
Minimo(#20081031#; #20081225#)= #20081031#
```

Parámetros:

1. **vParametro1, vParametro2…**: Dos o más valores numéricos, de fecha, de cadena, lógicos, **NaN** o **Null**.

EnRango(vParametro1; vParametro2; vParametro3) → LOGICO

Descripción: Devuelve **TRUE** si el primer parámetro es mayor o igual al primero y menor o igual al segundo. Se asume que todos los parámetros (y el resultado) son de mismo tipo que el primero de ellos. Ej.

```
Minimo(5; -1; 8) = True
Minimo(10; -1; 8) = False
Minimo(#20120418#; #20120415#; #20120430#) = True
Minimo("Beta"; "Alfa"; "Gamma") = True
```

Parámetros:

1. **vParametro1**: Valor numérico, de fecha, de cadena, lógico, **NaN** o **Null** que se desea comprobar.
2. **vParametro2**: Valor numérico, de fecha, de cadena, lógico, **NaN** o **Null** que indica el inicio del rango.
3. **vParametro3**: Valor numérico, de fecha, de cadena, lógico, **NaN** o **Null** que indica el fin del rango.

Observaciones:

- Si el primer parámetro es **NaN** o **Null**, la función siempre devolverá **FALSE**.

- El segundo y tercer parámetro se convertirán siempre al tipo del primer parámetro (cadena, número, fecha o lógico) antes de hacer la comparación de rango.

Redondear(nNumero; nDecimales) → NUMERO

Descripción: Devuelve el mismo número, redondeado al número de decimales indicado. El redondeo se hace hacia abajo (hacia "adentro" o hacia el 0) si el siguiente decimal es menor que 5, y hacia arriba ("afuera") si es igual o mayor a 5. Ej.

```
Redondear(4.125; 2) = 4.13
Redondear(227.20; -1) = 230
```

Parámetros:

1. **nNumero**: Valor numérico a ser redondeado.
2. **nDecimales**: Número entero de dígitos significativos a la derecha del punto decimal. Debe ser un entero entre -16 y 28 (la parte fraccional de **nDecimales** es truncada).

EnteroSuperior(nNumero) → NUMERO

Descripción: Devuelve el mismo número, redondeado al entero superior (hacia "la derecha"). Ej.

```
EnteroSuperior(4) = 4
EnteroSuperior(4.125) = 5
EnteroSuperior(-4.125) = -4
```

Parámetros:

1. **nNumero**: Valor numérico a ser redondeado.

EnteroInferior(nNumero) → NUMERO

Descripción: Devuelve el mismo número, redondeado al entero inferior (hacia "la izquierda"). Ej.

```
EnteroInferior(4) = 4
EnteroInferior(4.125) = 4
EnteroInferior(-4.125) = -5
```

Parámetros:

1. **nNumero**: Valor numérico a ser redondeado.

Truncar(nNumero) → NUMERO

Descripción: Devuelve el mismo número sin parte decimal. Ej.

```
Truncar(4) = 4
Truncar(4.125) = 4
Truncar(-4.125) = -4
```

Parámetros:

1. **nNumero**: Valor numérico a ser redondeado.

EsNulo(vValor) → VALOR

Descripción: Devuelve **TRUE** si **vValor** es **NULL**

Parámetros:

o **NaN**; devuelve **FALSE** en caso contrario. Ej.
`EsNulo("ABC") = FALSE`
`EsNulo(123) = FALSE`
`EsNulo(NULL) = TRUE`
`EsNulo(NaN) = TRUE`
`EsNulo(1/0) = TRUE`
	1. **vValor**: un valor a verificar.

NoNulo(vValor1[;vValor2]*) → VALOR	

Descripción: Devuelve el primer valor que no sea **NULL** o **NaN**. Si no hay ningún valor que no sea **NULL** o **NaN** entonces devuelve **NULL**. Ej.
`NoNulo("ABC") = "ABC"`
`NoNulo("ABC"; 123) = "ABC"`
`NoNulo(NULL; NaN; 123) = 123`
`NoNulo(NULL; NaN; NULL) = NULL`
	Parámetros:
	2. **vValor1[;vValor2…]***: uno o más valores a verificar. Pueden ser valores de diferentes tipos.

Funciones de Cadena

Longitud(cCadena) → NUMERO	
Descripción: Devuelve la longitud de la cadena especificada.	**Parámetros:** 1. **cCadena**: Cadena de la que se obtendrá la longitud.

Recortar(cCadena) → CADENA	
Descripción: Devuelve la cadena sin los espacios en blanco en los extremos.	**Parámetros:** 1. **cCadena**: Cadena cuyos espacios extremos serán eliminados.

TextoIzquierda(cCadena; nCantidad) → CADENA	
Descripción: Devuelve una subcadena con el número especificado de caracteres contados desde el principio de la cadena original.	**Parámetros:** 1. **cCadena**: Cadena de la que se obtendrá la subcadena. 2. **nCantidad**: Cantidad de caracteres que serán devueltos.

TextoDerecha(cCadena; nCantidad) → CADENA

Descripción: Devuelve una subcadena con el número especificado de caracteres contados desde el final de la cadena original.	**Parámetros:** 1. `cCadena`: Cadena de la que se obtendrá la subcadena. 2. `nCantidad`: Cantidad de caracteres que serán devueltos.

`TextoCentro(cCadena; nDesde [; nCantidad])` → `CADENA`

Descripción: Devuelve una sub-cadena con el número especificado de caracteres contados desde la posición indicada de la cadena original. Ej.
`TextoCentro("Comentario"; 4; 3) = "ent"`
`TextoCentro("Comentario"; 4) = "entario"`
	Parámetros:
	1. `cCadena`: Cadena de la que se obtendrá la subcadena.
	2. `nDesde`: Posición desde la cual se obtendrá la subcadena.
	3. `nCantidad`: Cantidad de caracteres que serán devueltos. Si no se indica una cantidad devuelve todos los caracteres a partir de `nDesde`.

`NoVacio(cCadena1; cCadena2 [; cCadenaN]*)` → `CADENA`

Descripción: Devuelve la primera cadena pasada como parámetro que no esté vacía; si todas las cadenas están vacías devuelve una cadena vacía. Ej.
`NoVacio(""; "q"; " ") = "q"`
`NoVacio(""; " "; "q") = " "`
	Parámetros:
	1. `cCadena1, cCadena2…`: Dos o más cadenas a evaluar.

`EsVacio(cCadena)` → `LOGICO`

Descripción: Devuelve un valor lógico que indica si la cadena indicada está o no vacía. Solo se considera vacía a una cadena de longitud cero. Ej.
`EsVacio("texto de prueba") = false`
`EsVacio("") = true`
`EsVacio("    ") = false`
	Parámetros:
	1. `cCadena`: Cadena a evaluar.

`FormatearFecha(fValor; cFormato [; cReferenciaCultural])` → `CADENA`

Descripción: Devuelve un valor de cadena que representa la fecha indicada con el formato indicado. Ej.	**Parámetros:** 1. `fValor`: Fecha a ser formateada. 2. `cFormato`: cadena con un formato de fecha

```
FormatearFecha(#20090115 14:25:48#;
"dd/MM/yyyy") = "15/01/2009"

FormatearFecha(#20090115 14:25:48#; "hh:mm
tt") = "02:25 PM"

FormatearFecha(#20090115 14:25:48#; "dddd,
dd/MMM/yyyy") = "Thursday, 15/Jan/2009"

FormatearFecha(#20090115 14:25:48#; "dddd,
dd/MMM/yyyy"; "es-VE") = "jueves, 15/ene/2009"
```

válido.

3. **`cReferenciaCultural`** (opcional): cadena con una referencia cultural válida. Usada para mostrar los nombres de meses y días en el idioma apropiado.

Observaciones:

- El formato de fecha debe ser una cadena con una combinación de los siguientes caracteres (solo se listan los más usados):

Cadena	Descripción
`yy`	Muestra el año, con dos o cuatro dígitos respectivamente.
`yyyy`	
`M`	Muestra el mes (se usa mayúscula para distinguirlo del minuto). En el primer caso con uno o dos dígitos según corresponda. En el segundo caso: siempre muestra dos dígitos, rellenando con cero a la izquierda cuando sea necesario. En el tercer caso muestra el mes en forma resumida, según la cultura actual. En el cuarto caso muestra el nombre mes completo, según la cultura actual.
`MM`	
`MMM`	
`MMMM`	
`d`	Muestra el día. Se cumplen las mismas reglas que para el mes para los cuatro casos.
`dd`	
`ddd`	
`dddd`	
`h`	Muestra la hora con uno o dos dígitos (`h`, `H`) según corresponda, o con dos dígitos (`hh`, `HH`) rellenando con cero a la izquierda cuando sea necesario. Las minúsculas (`h`, `hh`) muestran la hora en formato de 12 horas, y las mayúsculas (`H`, `HH`) lo hacen en formato de 24 horas.
`hh`	
`H`	
`HH`	
`m`	Muestra los minutos con uno o dos dígitos (`m`) según corresponda, o con dos dígitos (`mm`) rellenando con cero a la izquierda cuando sea necesario. Se usa minúscula para distinguirlo del mes.
`mm`	
`s`	Muestra los segundos con uno o dos dígitos (`s`) según corresponda, o con dos dígitos (`ss`) rellenando con cero a la izquierda cuando sea necesario.
`ss`	
`f`	Muestra fracciones de segundo. La cantidad de dígitos mostrada depende de la cantidad de veces que se repita.
`t`	Muestra el valor `"a"` o `"p"` (`t`), o el valor `"am"` o `"pm"` (`tt`) que indica si la hora corresponde a la mañana o la tarde. El valor mostrado puede verse afectado por la referencia cultural utilizada. Si la hora se muestra en formato de 24 horas esta cadena produce una cadena vacía.
`tt`	

- Cualquier carácter no incluido en la lista anterior se tomará como una cadena fija, por ejemplo los separadores de fecha y hora `":"`, `"-"`, `"."`, `"/"`, etc.

- Las cadenas fijas que no sean separadores deberían especificarse entre comillas simples para evitar que se confundan con los caracteres de la tabla anterior. Por ejemplo, la cadena de formato `"dddd, 'es feriado'"` en el último ejemplo debería generar el resultado como `"jueves, es feriado"`. Si no se usan las comillas simples algunos caracteres de la cadena fija se sustituirían por sus valores: `"jueves, e48 0eria15o"`.

- La referencia cultural es una cadena de al menos 5 caracteres que indica un código de idioma (según ISO 639) y un código de país o región cultural (según ISO 3166) separados por un guión `"-"`. Si no se especifica la referencia cultural, se usará la referencia `"en-US"`.

`FormatearNumero(nValor; nDecimales [; cSepDecimal [; cSepMiles]])` → **`CADENA`**

Descripción: Devuelve un valor de cadena que representa la fecha indicada con el formato indicado. Ej.

```
FormatearNumero(1234.5; 2) = "1,234.50"

FormatearNumero(1234.5; 0) = "1,235"

FormatearNumero(1234.5; 4; ","; "'") =
"1'234,5000"

FormatearNumero(1234.5; 2; ","; "") =
"1234,50"
```

Parámetros:

1. **`nValor`**: Número a ser formateado.
2. **`nDecimales`**: número que indica la cantidad de cifras decimales a mostrar.
3. **`cSepDecimal`** (opcional): carácter que se usará como separador decimal. Por defecto se usará un punto `"."`.
4. **`cSepMiles`** (opcional): carácter que se usará como separador de miles. Por defecto se usará una coma `","`.

Observaciones:

- El número de decimales debe ser un número entero mayor o igual a cero, y menor a cien.

- Si no se indica el separador decimal, si se indica una cadena vacía, o si se indica un carácter de espacio `" "`, se usará un punto `"."` como separador decimal.

- Si no se indica el separador de miles se usará una coma `","`. Si se indica una cadena vacía entonces no se mostrará ningún separador de miles.

- Solo se tomará el primer carácter del separador decimal, y el primer carácter del separador de miles.

- Si se especifican dos separadores iguales, entonces el separador de miles se cambiará

por espacio " " o comilla simple "''" para evitar que coincidan.

Funciones de Fecha

Hoy() → FECHA

Descripción: Devuelve la fecha actual, con las componentes de hora establecidas a 0. Ej. `Hoy() = #20090115#`	**Parámetros:** 1. Ninguno.

Ahora() → FECHA

Descripción: Devuelve la fecha y hora actual. Ej. `Ahora() = #20090115 14:25:22.159#`	**Parámetros:** 1. Ninguno.

NuevaFecha(nAño; nMes; nDia[; nHoras[; nMinutos[; nSegundos[; nMilisegundos]]]]) → FECHA

Descripción: Devuelve un valor de fecha creado a partir de las componentes indicadas. Los valores de las componentes son corregidos en caso de salir del rango de valores válidos. Ej.
`NuevaFecha(2009; 1; 15) = #20090115#`
`NuevaFecha(2009; 1; 15; 14) = #20090115 14:00#`
`NuevaFecha(2009; 1; 15; 14; 25; 10; 22) = #20090115 14:25:10.022#`
	Parámetros:
	1. **nAño**, **nMes**, **nDia**: Parámetros enteros obligatorios; la parte fraccional es truncada.
	2. **nHoras**, **nMinutos**, **nSegundos**, **nMilisegundos**: Parámetros enteros opcionales; la parte fraccional es truncada.

ComponenteFecha(dFecha; cUnidad) → NUMERO

Descripción: Devuelve el componente **cUnidad** de la fecha indicada. El día de la semana devuelve 0 para domingo, 1 para lunes… y 6 para sábado.
`ComponenteFecha(#20090215#; "AÑO") = 2`
`ComponenteFecha(#20090215#; "DIA") = 15`
`ComponenteFecha(#20090215#; "DIA-SEMANA")= 0`
`ComponenteFecha(#20090215 12:35:12#; "MINUTO") = 35`
	Parámetros:
	1. **dFecha**: Fecha de la que se obtendrá el componente.
	2. **cUnidad**: Uno de los siguientes valores de cadena: "SEGUNDO", "MINUTO", "HORA", "DIA", "DIA-SEMANA", "DIA-AÑO", "MES", "AÑO".

ContarDias(dFechaInicial; dFechaFinal) → NUMERO	

Descripción: Devuelve la cantidad de días desde **dFechaInicial** hasta **dFechaFinal**. La hora de las fechas no es tomada en cuenta en los cálculos. Si **dFechaInicial** es mayor que **dFechaFinal** el resultado es negativo. Ej.
`ContarDias(#20080401#; #20080401#)=1`
`ContarDias(#20080401#; #20080402#)=2`
`ContarDias(#20080401#; #20080430#)=30`
`ContarDias(#20080401#; #20080501#)=31`
`ContarDias(#20080402#; #20080401#)=-2`
	Parámetros:
	1. **dFechaInicial**: Fecha desde la cual se contará.
	2. **dFechaFinal**: Fecha hasta la cual se contará.

Observaciones:	• Tanto la fecha inicial como la fecha final se incluirán en los cálculos. Ver la función **DiferenciaFechas()**.

ContarSemanas(dFechaInicial; dFechaFinal) → NUMERO	

Descripción: Devuelve la cantidad de semanas completas (periodos de 7 días) desde **dFechaInicial** hasta **dFechaFinal**. La hora de las fechas no es tomada en cuenta en los cálculos. Si **dFechaInicial** es mayor que **dFechaFinal** el resultado es negativo. Ej.
`ContarSemanas(#20080401#; #20080420#)=2`
`ContarSemanas(#20080401#; #20080421#)=3`
`ContarSemanas(#20080407#; #20080401#)=-1`
	Parámetros:
	3. **dFechaInicial**: Fecha desde la cual se contará.
	4. **dFechaFinal**: Fecha hasta la cual se contará.

ContarDiasSemana(dFechaInicial; dFechaFinal; vDiaSemana) → FECHA	

Descripción: Devuelve la cantidad de veces que un día de semana indicado aparece desde **dFechaInicial** hasta **dFechaFinal**. Si **dFechaInicial** es mayor que **dFechaFinal** el resultado es negativo. Ej.
`ContarDiasSemana(#20080401#; #20080401#; "MARTES")=1`
`ContarDiasSemana(#20080401#; #20080415#; 2)=2`
`ContarDiasSemana(#20080415#; #20080407#)=-16`
	Parámetros:
	1. **dFechaInicial**: Fecha desde la cual se contará.
	2. **dFechaFinal**: Fecha hasta la cual se contará.
	3. **vDiaSemana**: Un número entero entre 0 y 6, o uno de los siguientes valores de cadena: "LUNES", "MARTES", "MIERCOLES", "JUEVES", "MES", "AÑO".

DiferenciaFechas(dFechaInicial; dFechaFinal [; cUnidad]) → FECHA

Descripción: Devuelve la cantidad de unidades de tiempo (enteras) desde **dFechaInicial** hasta **dFechaFinal**; Si **cUnidad** es "DIA", "MES" o "AÑO" las horas no se toman en cuenta para el cálculo de la diferencia. Si **dFechaInicial** es mayor que **dFechaFinal** el resultado es negativo. Ej.

```
DiferenciaFechas(#20081220#; #20081220#)=0

DiferenciaFechas(#20081220#; #20081221#)=1

DiferenciaFechas(#20081220#; #20081230#)=10

DiferenciaFechas(#20081220 23:59#;
#20081230 00:00#)=10

DiferenciaFechas(#20090115#; #20081230#)=-
16

DiferenciaFechas(#20081220#; #20090119#;
"MES")=0

DiferenciaFechas(#20081220#; #20090120#;
"MES")=1
```

Parámetros:

1. **dFechaInicial**: Fecha desde la cual se contará.
2. **dFechaFinal**: Fecha hasta la cual se contará.
3. **cUnidad**: Uno de los siguientes valores de cadena: "SEGUNDO", "MINUTO", "HORA", "DIA", "MES", "AÑO". Si se omite se usará "DIA".

Observaciones:

- Si **cUnidad** es igual a "DIA" la fecha inicial se incluirá en los cálculos, pero no la final. Ver la función **ContarDias()**.
- Si **cUnidad** es igual a "MES" contará la cantidad de meses calendario desde la fecha inicial hasta la fecha final.
- Si **cUnidad** es igual a "AÑO" contará la cantidad de años calendario desde la fecha inicial hasta la fecha final.

TruncarFecha(dFecha) → FECHA

Descripción: Devuelve la misma fecha con las componentes de hora establecidas a 0. Ej.

```
TruncarFecha(#20081220 15:00:25#) =
#20081220#

TruncarFecha(#20081220 00:00:00#) =
#20081220#
```

Parámetros:

1. **dFecha**: Fecha a la cual se eliminará (o establecerá a cero) las componentes de hora.

SumarDia(dFecha; nCantidad) → FECHA

Descripción: Devuelve la fecha **dFecha** luego de sumarle **dCantidad** días. La suma se realiza considerando los años bisiestros y el número de días máximo de cada mes. Ej.

Parámetros:

1. **dFecha**: A la cual se le sumará (o restará) el número de unidades indicadas.
2. **nCantidad**: Cantidad de días que se sumará o restará a **dFecha**. La parte decimal de este

`SumarDia(#20081231#; 1) = #20090101#`
`SumarDia(#20081231#; -1) = #20081230#`
`SumarDia(#20090228#; 1) = #20090301#`
	parámetro, si la hay, se ignorará.

SumarComponenteFecha(dFecha; nCantidad; cUnidad) → FECHA

Descripción: Devuelve la fecha **dFecha** luego de sumarle **dCantidad** unidades de tiempo de tipo **cUnidad**. La suma se realiza considerando los años bisiestros y el número de días máximo de cada mes: Si al sumar meses la componente de día no es válida, ésta es ajustada a una fecha válida (por ejemplo, al sumar un mes al 31 de Enero se obtiene 28 de Febrero). Ej.

```
SumarComponenteFecha(#20081231#; 1; "DIA") = #20090101#

SumarComponenteFecha(#20081231#; 1.75; "MINUTO") = #20081231 00:01:45#

SumarComponenteFecha(#20090131 14:25:05#; -3; "MES") = #20081031 14:25:05#

SumarComponenteFecha(#20090131 14:25:05#; 1; "MES") = #20090228 14:25:05#
```

Parámetros:

3. **dFecha**: A la cual se le sumará (o restará) el número de unidades indicadas.
4. **nCantidad**: Cantidad de unidades que se sumará o restará a **dFecha**. Si **cUnidad** es "MES" o "AÑO" se truncará la parte decimal de **nCantidad**, en caso contrario la parte decimal se agregará coherentemente a las unidades menores
5. **cUnidad**: Uno de los siguientes valores de cadena: "SEGUNDO", "MINUTO", "HORA", "DIA", "MES", "AÑO".

ObtenerAño(dFecha) → NUMERO

Descripción: Devuelve el valor del año de la fecha indicada. Ej.

```
ObtenerAño(#20081220 15:00:25#) = 2008
```

Parámetros:

1. **dFecha**: Fecha de la cual se obtendrá el valor del año.

ObtenerMes(dFecha) → NUMERO

Descripción: Devuelve el valor del mes de la fecha indicada. Ej.

```
ObtenerMes(#20081220 15:00:25#) = 12
```

Parámetros:

1. **dFecha**: Fecha de la cual se obtendrá el valor del año.

ObtenerDia(dFecha) → NUMERO

Descripción: Devuelve el valor del día del mes de la fecha indicada. Ej.

```
ObtenerDia(#20081220 15:00:25#) = 20
```

Parámetros:

1. **dFecha**: Fecha de la cual se obtendrá el valor del día del mes.

ObtenerDiaSemana(dFecha) → NUMERO	
Descripción: Devuelve el valor del día de la semana de la fecha indicada. El valor 0 corresponde a Domingo, el valor 1 al Lunes, etc. Ej. `ObtenerDiaSemana(#20080401 18:30:45#) = 2`	**Parámetros:** 1. **dFecha**: Fecha de la cual se obtendrá el valor del día de la semana.