EVALUADOR DE FÓRMULAS

REFERENCIA DEL LENGUAJE

Distrito-K, 2018

Contenido

1. TIPOS DE DATOS 8
2. OPERADORES DE CÁLCULO 9
3. OPERADOR DE ASIGNACIÓN 9
4. OPERADORES DE COMPARACIÓN 10
5. OTROS OPERADORES 11
6. ESTRUCTURAS DE LENGUAJE 12
7. CONSTANTES PREDEFINIDAS 12
8. FUNCIONES MATEMÁTICAS 13
8.1. Abs 13
8.2. Decimal 14
8.3. Defecto 15
8.4. Negativo 16
8.5. Redondeo 17
8.6. RedondeoDec 18
8.7. RedondeoFactor 19
8.8. Pi 20
8.9. Entero 21
8.10. LogXY 22
8.11. Positivo 23
8.12. Potencia 24
8.13. Exceso 25
8.14. Máx 26
8.15. Mín 27
9. FUNCIONES DE MANIPULACIÓN DE TEXTOS 28
9.1. Carácter 28
9.2. Intro 29
9.3. TxTamaño 30
9.4. TxEsNúmero 31
9.5. TxEstaVacío 32
9.6. TxConvierte 33
9.7. TxMinúsculas 34
9.8. TxMayúsculas 35
9.9. TxDerecha 36
9.10. TxReplica 37
9.11. TxQuitaFormato 38
9.12. TxPosición 39
9.13. TxLimpia 40
9.14. TxIzquierda 41
9.15. TxLimpiaDer 42
9.16. TxLimpiaIzq 43
9.17. TxPrimeraMayúscula 44
9.18. TxPalabra 45
9.19. TxBorra 46
9.20. TxTrozo 47
9.21. TxRellenaDer 48
9.22. TxRellenaIzq 49

9.23. TxDesdeHasta 50
9.24. TxReemplaza 51
9.25. TxCuantasPalabras 52
9.26. TxSustituye 53
9.27. TxRestoPalabras 54
9.28. TxContiene 55
9.29. TxEmpiezaCon 56
9.30. TxTerminaCon 57
9.31. TxCumpleMáscara 58
9.32. TxIgual 59
9.33. TxCompara 60
10. FUNCIONES DE CONVERSIÓN 61
10.1. EnLetra 61
10.2. EnTexto 62
10.3. EnLetraF 63
10.4. EnLetraM 64
10.5. EnNúmero 65
10.6. Entrecomillar 66
10.7. FormateaHoras 67
10.8. Formatea 68
10.9. FormateaPrecio 69
10.10. EnHexadecimal 70
11. FUNCIONES DE MANIPULACIÓN DE LISTAS (ARRAYS) 71
11.1. LAñade 71
11.2. LAplica 72
11.3. RAñade 73
11.4. LBorra 74
11.5. LCrea 75
11.6. LConvierte 76
11.7. LCarga 77
11.8. LGraba 78
11.9. LInserta 79
11.10. RNombreCampo 80
11.11. LOrdena 81
11.12. Lposición 82
11.13. RPosiciónCampo 83
11.14. LQuita 84
11.15. RQuita 85
11.16. LTamaño 86
11.17. RTamaño 87
11.18. RValorCampo 88
11.19. ChequeaTipoItem 89
11.20. TipoLista 90
11.21. TxEmpaqueta 91
11.22. TxDesempaqueta 92
11.23. RExisteCampo 93
12. FUNCIONES DE MANIPULACIÓN DE FECHAS 94
12.1. MesDe 94
12.2. ReformateaFecha 95
12.3. SemanaDe 96

12.4. DiferenciaEnDías ... 97
12.5. DiferenciaEnHoras ... 98
12.6. SiguienteMes ... 99
12.7. ÚltimoDíaMes ... 100
12.8. AnteriorMes ... 101
12.9. HoraEnTexto ... 102
12.10. HoraEnNúmero ... 103
12.11. PrimerDíaMes ... 104
12.12. TrimestreDe ... 105
12.13. EsAñoBisiesto ... 106
12.14. EsMismoMes ... 107
12.15. DíaDe ... 108
12.16. DíasMes ... 109
12.17. DíasAño ... 110
12.18. DíaDeSemana ... 111
12.19. AñoDe ... 112
12.20. FechaHoy ... 113
12.21. FechaEnTexto ... 114
12.22. FechaEnNúmero ... 115
12.23. FormateaFecha ... 116
12.24. FormateaFechaHora ... 117
12.25. HoraActual ... 118
13. FUNCIONES DE MANIPULACIÓN DE VARIABLES ... 119
13.1. EditarVars ... 119
13.2. DestruirVariable ... 120
13.3. GetVariablePrefix ... 121
13.4. SetVariablePrefix ... 122
13.5. VerVars ... 123
13.6. Tipo ... 124
13.7. BorrarVars ... 125
13.8. EsNada ... 126
13.9. DuplicarVars ... 127
13.10. ExisteVariable ... 128
14. FUNCIONES DE ACCESO A BASE DE DATOS ... 129
14.1. ObtListaSQL ... 129
14.2. Desconecta ... 130
14.3. EjecutarSQL ... 131
14.4. ConectaSQL ... 132
14.5. ConectaBDE ... 133
14.6. ConsultaCampo ... 134
14.7. ConsultaLista ... 135
14.8. ConsultaCuantos ... 136
14.9. ConsultaRegistro ... 137
14.10. CuantosRegistros ... 138
15. FUNCIONES VARIAS ... 139
15.1. Depurador ... 139
15.2. Mensaje ... 140
15.3. ResultadoNada ... 141
15.4. SeleccionarTabla ... 142
15.5. Clona ... 144

15.6. AnteriorControl 145
15.7. Inspeccionar 146
15.8. Confirma 147
15.9. Error 148
15.10. Pregunta 149
15.11. Evalúa 150
15.12. PararImpresión 151
15.13. Imprimir 152
15.14. SiguienteControl 153
15.15. DLL 154
15.16. CumpleRestricción 155
15.17. ActualInsertar 156
15.18. ActualBorrar 157
15.19. ActualLeer 158
15.20. ActualEscribir 159
15.21. SeleccionarConsulta 161
15.22. EnEan 162
15.23. Visor 163
15.24. GenerarInforme 164
15.25. Escape 166
15.26. CargarDocumento 167
15.27. AbrirOpción 169
15.28. ObtenerInformación 170
15.29. ObtenerInformacionFichero 171
15.30. ObtenerListaFicheros 172
15.31. PDF417 173
15.32. LeerBalanza 174
15.33. ReproducirSonido 175
15.34. SeleccionarTipo 176
15.35. EmitirDocumento 177
15.36. EnviarCorreo 178
15.37. EnviarCorreoCuenta 180
15.38. CrearDocumento 182
15.39. CrearApunteCaja 183
15.40. PreguntaNúmeroTáctil 184
15.41. NuevoCorreo 185
15.42. NuevoCorreoCuenta 186
15.43. RLeeXML 187
15.44. RLeeJSON 188
15.45. QRCode 189
15.46. GS1_128 190
15.47. FTPEstáConectado 191
15.48. FTPObtenerDirectorioActual 192
15.49. FTPExisteFichero 193
15.50. FTPTamañoFichero 194
15.51. FTPObtenerListaFicheros 195
15.52. FTPConectar 196
15.53. FTPBajarFichero 197
15.54. CalcularDescuentos 198
15.55. AbrirOpciónVentana 199

15.56. ObtenerFormato 200
15.57. Leer 201
15.58. Escribir 203
15.59. RenombrarFichero 204
15.60. RecalcularImportesConsultaDocumento 205
15.61. BinCrea 206
15.62. BinTipo 207
15.63. BinBorra 208
15.64. BinTamaño 209
15.65. BinGraba 210
15.66. BinCarga 211
15.67. BinDescargaHttp 212
15.68. ImprimirTraspaso 213
15.69. EnviarSMS 214
15.70. BorrarFichero 215
15.71. BorrarDirectorio 216
15.72. BorrarLog 217
15.73. Code128 218
15.74. Code128A 219
15.75. Code128B 220
15.76. Code128C 221
15.77. CodigoUsuarioActual 222
15.78. ConfiguraADO 223
15.79. ConectaADO 224
15.80. ComponerRuta 225
15.81. CrearDirectorio 226
15.82. CreaADO 227
15.83. PreguntaMemo 228
15.84. UsuarioActual 229
15.85. HttpGet 230
15.86. HttpPost 232
15.87. ExtraerRuta 234
15.88. ExisteDirectorio 235
15.89. ExisteFichero 236
15.90. CódigoCarácter 237
15.91. AbrirFichero 238
15.92. MensajeHTML 239
15.93. CopiarFichero 240
15.94. FTPDesconectar 241
15.95. FTPFijarDirectorioSuperior 242
15.96. FTPBorrarFichero 243
15.97. FTPForzarDirectorio 244
15.98. FTPCrearDirectorio 245
15.99. FTPBorrarDirectorio 246
15.100. FTPFijarDirectorioActual 247
15.101. FTPRenombrarFichero 248
15.102. FTPSubirFichero 249
15.103. EnviarMensaje 250
15.104. BinAñade 251
15.105. BinBorra 252

15.106. TrocearTextoImpresión 253
15.107. DatoAdicional 254
15.108. ArtículoUbicaciónCaracterística 255
15.109. NUEVA VARIABLE 256
15.110. FormateaCantidad 257
15.111. TxCarga 258
15.112. EnUPC 259
15.113. EsEAN13 260
15.114. EsUPCA 261
15.115. TxGraba 262
15.116. CUADROS CONCEPTOS RECURSOS Y MATERIALES 263
15.117. Operadores 266
16. FUNCIONES ESPECÍFICAS FORMATOS DE IMPRESIÓN 269
16.1. DimensionesTextoImpresión 269
16.2. ArtículoCaracterística 270
17. FUNCIONES DE CUADROS 271
17.1. ArtículoCuadroCaracterística 271
17.2. CuadroCuantasFilas 272
17.3. CuadroConfiguración 273
17.4. CuadroCrea 275
17.5. CuadroTraspone 276
17.6. CuadroFila 277
17.7. CuadroIdentificación 279
17.8. CuadroCuantasColumnas 280
17.9. CuadroColumnaTítuloFila 281
17.10. CuadroBorrarFila 283
17.11. CuadroColumna 284
17.12. CuadroCelda 286
17.13. CuadroBorrarColumna 288
17.14. CuadroAñadeFila 289
17.15. CuadroAñadeColumna 290
17.16. CuadroExporta 291
18. FORMATO DE MÁSCARAS PARA NÚMEROS 294
19. FORMATO DE MÁSCARAS PARA FECHAS 296
20. FORMATO DE MÁSCARAS PARA HORAS 296
21. FUNCIONES ESPECÍFICAS DE SQLCONTA 297
22. FUNCIONES ESPECÍFICAS DE SQLPYME PARA IMPRESIÓN DE DOCUMENTOS 299
23. EJEMPLOS PRÁCTICOS DE SCRIPTS 302

1. TIPOS DE DATOS

Nada
Tipo nulo

Número
Un número de coma flotante de doble precisión.
Ejemplo: 12777,45 , 0,000056 , 1E18

Texto
Una cadena de texto de cualquier longitud, delimitada entre comillas dobles o simples.
Ejemplo: "hola" , 'sin comentarios'

Lógico
Los valores 'cierto' y 'falso'.
Ejemplo: cierto , falso , a < b

Bloque
Un bloque de código, que especifica una secuencia de instrucciones.
Ejemplo: {a=1;b=2;c=3}

Variable
Una variable, de cualquiera de los tipos indicados.
Ejemplo: &a (siendo a una variable)

Lista
Una lista (array) de elementos, de cualquiera de los tipos indicados.
Ejemplo: #("uno"; "dos"; "tres"; 4; 5)

Tipo
Un tipo de dato, cualquiera de los tipos indicados.
Ejemplo: tipo("uno")

2. OPERADORES DE CÁLCULO

+ (más)
- Suma de dos números
  Ejemplo: 3 + 4 devuelve 7
- Concatenación de dos textos
  Ejemplo: "uno" + "dos" devuelve "unodos"

- (menos)
- Resta de dos números
  Ejemplo: 3 - 4 devuelve -1
- Cambio de signo de un número
  Ejemplo: -(3 + 4) devuelve -7

*** (asterisco)**
Multiplicación de dos números
Ejemplo: 3 * 4 devuelve 12

/ (barra de división)
División de dos números
Ejemplo: 3 / 4 devuelve 0,75

modulo
Resto de la división entera de dos números
Ejemplo 7 modulo 4 devuelve 3

no
negación de un valor lógico
Ejemplo: no (3 > 4) devuelve cierto

y
Combina dos valores lógicos con el operador lógico AND
Ejemplo: (1 > 2) y (3 < 4) devuelve falso

o
Combina dos valores lógicos con el operador lógico OR
Ejemplo: (1 > 2) o (3 < 4) devuelve cierto

3. OPERADOR DE ASIGNACIÓN

= (igual)
Asigna un valor (y un tipo) a una variable, definiéndola si no existe
Ejemplo: a = 7
Nota: No utilizar este operador para comparar dos elementos (un error bastante frecuente)

4. OPERADORES DE COMPARACIÓN

== (doble igual)

Devuelve cierto si los dos elementos comparados son iguales. Es aplicable a todos los tipos, pero si los elementos son de diferente tipo, se produce un "error de tipo".
Ejemplo: 3 == 4 devuelve falso

<> (menor-mayor)

Devuelve cierto si los dos elementos comparados son diferentes. Es aplicable a todos los tipos, pero si los elementos son de diferente tipo, se produce un "error de tipo".
Ejemplo: 3 <> 4 devuelve cierto

> , >= , < , <= (mayor, mayor-igual, menor, menor-igual)

- Si los operandos son números, evalúa las comparaciones según el valor numérico de los operandos
  Ejemplo: 3 > 4 devuelve falso
- Si los operandos son textos, evalúa las comparaciones según el orden alfabético de los operandos, sin tener en cuenta las mayúsculas / minúsculas
  Ejemplo: "abeja" < "mosca" devuelve cierto
- Si los operandos son lógicos, se considera que cierto es mayor a falso
  Ejemplo: cierto > falso devuelve cierto
- Si los operandos son de diferente tipo entre sí, o de cualquier otro tipo que los arriba indicados, se produce un "error de tipo"

5. OTROS OPERADORES

( ) (paréntesis)

- Agrupan operaciones, alterando su precedencia. Se permiten ilimitados niveles
  Ejemplo: (a + b)*(c + (d - e) / 2)
- Indican los argumentos o parámetros de las funciones
  Ejemplo: EnLetras(a)
- Precedidos por el símbolo #, definen una lista de elementos
  Ejemplo: #("uno";"dos";"tres")

[ ] (corchetes)

- Permiten obtener un caracter de un texto
  Ejemplo: a = "distrito-k"; a[3] devuelve "s"
- Permiten obtener un elemento de una lista (array)
  Ejemplo: a = #("uno";"dos";"tres"); a[2] devuelve "dos"
- Evalúan una función predefinida (depende de la aplicación)
  Ejemplo (sólo en Contasoft): [s '430'] devuelve el saldo de la cuenta 430

{ } (llaves)

Delimitan un bloque de código
Ejemplo: si a == b entonces {a = a + 1} sino {a = a - 1}

@ (arroba)

Permite obtener el valor de una variable cuyo nombre se obtiene a partir de una expresión
Ejemplo: a = 25; nombre = "a"; @nombre devuelve 25

& (ampersand)

Permite obtener la variable cuyo nombre se obtiene a partir de una expresión
Ejemplo: a = 25; &a devuelve la variable a, no el valor 25

; (punto y coma)

- Permite separar los elementos de una lista (array)
  Ejemplo: a = #("uno";"dos";"tres")
- Permite separar los argumentos o parámetros de una función
  Ejemplo: Min(a; b)
- Permite ejecutar más de una orden, separándolas entre sí. El valor devuelto por el conjunto de órdenes es el último evaluado.
  Ejemplo: a = 3; b = 4; a + b; Se ejecutan las tres órdenes, y devuelve 7

6. ESTRUCTURAS DE LENGUAJE

funciones

Para llamar a una función, debe indicarse su nombre, y si desean indicarse argumentos o parámetros, deben indicarse a continuación entre paréntesis, separados entre sí por ";"
Ejemplo de llamada a una función sin argumentos: Pi
Ejemplo de llamada a una función con argumentos: Max(a;b)

si <expresión lógica> entonces <bloque de código>

Si <expresión lógica> devuelve cierto, ejecuta <bloque de código>
Ejemplo: si a < b entonces {"a es muy pequeño"}

si <expresión lógica> entonces <bloque de código 1> sino <bloque de código 2>

Si <expresión lógica> devuelve cierto, ejecuta <bloque de código 1>, pero si <expresión logica> devuelve falso, ejecuta <bloque de código 2>
Ejemplo: si a < b entonces {"a es muy pequeño"} sino {"a está bien"}

mientras <bloque de código 1> haz <bloque de código 2>

Si <bloque de código 1> devuelve cierto, y mientras siga siéndolo, ejecuta <bloque de código 2>
Ejemplo: mientras {a < 5} haz {a = a + 1}

haz <bloque de código 1> mientras <bloque de código 2>

Ejecuta el <bloque de código 1>, y sigue ejecutándolo mientras el <bloque de Código 2> devuelva cierto
Ejemplo: haz {a = a + 1} mientras {a < 5}
Nota: La diferencia con la orden anterior es que <bloque de código> se ejecuta al menos una vez.

7. CONSTANTES PREDEFINIDAS

Constante	Tipo	Descripción
Nada	Nada	El valor nulo
Cierto	Lógico	El valor lógico cierto
Falso	Lógico	El valor lógico falso
TNada	Tipo	El tipo Nada
TNumero	Tipo	El tipo Número
TTexto	Tipo	El tipo Texto
TLogico	Tipo	El tipo Lógico
TBloque	Tipo	El tipo Bloque
TVariable	Tipo	El tipo Variable
TLista	Tipo	El tipo Lista
TTipo	Tipo	El tipo Tipo

8. FUNCIONES MATEMÁTICAS

8.1. Abs

Descripción:
Se devuelve el valor absoluto del número pasado como parámetro.

Sintaxis: Abs(<n>)

Tipo de dato devuelto: Número

Ejemplo:
```
a = Abs(-7);     %% a será 7
b = Abs(7);      %% b será 7
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.2. Decimal

Descripción:
Devuelve la parte decimal del número pasado como parámetro.

Sintaxis: Decimal(<n>)

Tipo de dato devuelto: Número

Ejemplo:
a = Decimal(4,9); %% a será 0,9

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.3. Defecto

Descripción:
El entero inmediatamente inferior al número pasado como parámetro.
Puede usarse para truncar a un número de decimales si se multiplica por 1 con el número de decimales deseado y luego se divide por el mismo número.

Sintaxis: Defecto(<n>)

Tipo de dato devuelto: Número

Ejemplo:
```
a = Defecto(4,9);   %% a será 4
b = Defecto(4,3);   %% b será 4

c = Defecto(1000,1265 * 100) / 100; %% c será 1000,12
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.4. Negativo

Descripción:
Si el número pasado como parámetro es negativo se devuelve este, sino 0.

Sintaxis: Negativo(<n>)

Tipo de dato devuelto: Número

Ejemplo:
```
a = Negativo(4,9);    %% a será 0
b = Negativo(-4,9);   %% b será -4,9
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.5.Redondeo

Descripción:
Se devuelve el número entero más próximo al pasado como parámetro.

Sintaxis: Redondeo(<n>)

Tipo de dato devuelto: Número

Ejemplo:
```
a = Redondeo(4,5);   %% a será 5
b = Redondeo(4,6);   %% b será 5
c = Redondeo(4,4);   %% c será 4
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.6.RedondeoDec

Descripción:
Devuelve <n1> redondeado al número de decimales indicado por <n2>

Sintaxis: RedondeoDec(<n1>;<n2>)

Tipo de dato devuelto: Número

Ejemplo:
```
a = RedondeoDec(3,141516; 3);   %% a será 3,142
b = RedondeoDec(3,141516; 2);   %% b será 3,14
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.7.RedondeoFactor

Descripción:
Devuelve el múltiplo de <n2> más próximo a <n1>
Sólo funciona con números enteros en n2.

Sintaxis: RedondeoFactor(<n1>; <n2>)

Tipo de dato devuelto: Número

Ejemplo:
a = RedondeoFactor(15; 6); %% a será 18
b = RedondeoFactor(13; 6); %% b será 12

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.8.Pi

Descripción:
Devuelve el número Pi (3,141516…)

Sintaxis: Pi

Tipo de dato devuelto: Número

Ejemplo:
a = Formatea("0.0000000000"; Pi); %% a será 3,1415926536

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.9. Entero

Descripción:
Devuelve la parte entera del número pasado como parámetro
Permite truncar un número con decimales para quedarse sólo con la parte entera.

Sintaxis: Entero(<n>)

Tipo de dato devuelto: Número

Ejemplo:
a = Entero(4,9); %% a será 4

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.10. LogXY

Descripción:
Logaritmo de <n2> en base <n1>

Sintaxis: LogXY(<n1>; <n2>)

Tipo de dato devuelto: Número

Ejemplo:
```
a = LogXY(10; 100);              %% a valdrá 2
b = LogXY(2,718281828; 10);   %% b valdrá 2,30259
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.11. Positivo

Descripción:
Si el número pasado como parámetro es positivo se devuelve este, sino 0.

Sintaxis: Positivo(<n>)

Tipo de dato devuelto: Número

Ejemplo:
```
a = Positivo(4,9);    %% a será 4,9
b = Positivo(-4,9);   %% b será 0
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.12. Potencia

Descripción:
Potencia de <n1> elevado a <n2>

Sintaxis: Potencia(<n1>; <n2>)

Tipo de dato devuelto: Número

Ejemplo:
a = Potencia(2; 3); %% a valdrá 8
b = Potencia(2; 8); %% b valdrá 256

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.13. Exceso

Descripción:
El entero inmediatamente superior al número pasado como parámetro.

Sintaxis: Exceso(<n>)

Tipo de dato devuelto: Número

Ejemplo:
```
a = Exceso(4,9);   %% a será 5
b = Exceso(4,3);   %% b será 5
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.14. Máx

Descripción:
Se devuelve el mayor de los dos números pasados.

Sintaxis: Máx(<n1>; <n2>)

Tipo de dato devuelto: Número

Ejemplo:
a = Máx(10; 8); %% a valdrá 10
b = Máx(-5; -2); %% b valdrá -2

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

8.15. Mín

Descripción:
Se devuelve el menor de los dos números pasados.

Sintaxis: Mín(<n1>; <n2>)

Tipo de dato devuelto: Número

Ejemplo:
a = Mín(10; 8); %% a valdrá 8
b = Mín(-5; -2); %% b valdrá -5

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones matemáticas

9. FUNCIONES DE MANIPULACIÓN DE TEXTOS

9.1. Carácter

Descripción:

Devuelve el carácter con código ASCII indicado por <n>.
Puede usarse en los formatos de impresión para cambiar la fuente de una variable en base a una condición. Se coloca "Carácter(255)" en vez de "#", después irá "F" y el número de la fuente del formato.

También puede usarse el editor de informes para lo mismo. Se coloca "Carácter(255) en vez de "#", después irá "F", el número de la fuente del formato e Intro. El campo debe tener alto automático.

Sintaxis: Carácter(<n>)

Tipo de dato devuelto: Texto

Ejemplo:

```
%% Ejemplo en formato de impresión:
%% Campo "ArtículoFórmula"
#F16;
si (ArtículoDatoAdicionalValor("Formato") == 'Negrita') entonces {
   ArtículoNombre
};
#F0;
si (ArtículoDatoAdicionalValor("Formato") <> 'Negrita') entonces {
  ArtículoNombre
};

% Ejemplo en editor de informes (el campo tiene que tener alto automático
y el Intro no se imprimirá):%
%% Campo fórmula
si (LíneaDePresupuesto.Nivel == 1) entonces {
  Caracter(255) + "F0" + Intro + LíneaDePresupuesto.Artículo.Código;
} sino {
  Caracter(255) + "F1" + Intro + LíneaDePresupuesto.Artículo.Código;
};
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.2. Intro

Descripción:
Devuelve un retorno de carro.
Se usa para separar un texto por líneas.

Sintaxis: Intro

Tipo de dato devuelto: Nada

Ejemplo:
```
Texto1 = "SQL Pyme";
Texto2 = "SQL Conta";
Texto = Texto1 + Intro + Texto2;
% Texto valdrá:
"SQL Pyme
SQL Conta"
%
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.3. TxTamaño

Descripción:
Se devuelve el número de caracteres de <t>, los espacios en blanco también se cuentan.

Sintaxis: TxTamaño(<t>)

Tipo de dato devuelto: Número

Ejemplo:
```
a = "SQL Commerce ";
b = TxTamaño(a);          %% b valdrá 13
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.4. TxEsNúmero

Descripción:
Devuelve cierto si el texto <t> es un valor numérico.
Ignora los separadores de miles si existen.

Sintaxis: TxEsNúmero(<t>)

Tipo de dato devuelto: Lógico

Ejemplo:

```
x = "1.234,56";
h = "1.234,56 €";
i = "1234.56";

xl = TxEsNúmero(x);   %% x1 valdrá cierto
h1 = TxEsNúmero(h);   %% h1 valdrá falso
i1 = TxEsNúmero(i);   %% i1 valdrá cierto
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.5. TxEstaVacío

Descripción:
Devuelve "cierto" si <t> está vacio o tiene sólo espacios en blanco.

Sintaxis: TxEstaVacío(<t>)

Tipo de dato devuelto: Lógico

Ejemplo:
```
a = "SQL Pyme";
b = "";
c = "    ";
a1 = TxEstaVacío(a);   %% falso
b1 = TxEstaVacío(b);   %% cierto
c1 = TxEstaVacío(c);   %% cierto
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.6. TxConvierte

Descripción:
Si el texto <t> es un número válido se devuelve convertido en número, si no lo es devuelve el texto <t>.
En cualquier caso los puntos (separador de miles) se eliminan del resultado.

Sintaxis: TxConvierte(<t>)

Tipo de dato devuelto: Texto/Número

Ejemplo:

```
x = "1.234,56";
h = "1.234,56 €";
i = "1234.56";

xl = TxConvierte(x);   %% x1 valdrá 1234,56 (un número)
h1 = TxConvierte(h);   %% h1 valdrá "1234,56 €" (un texto)
i1 = TxConvierte(i);   %% i1 valdrá 123456 (un número)
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.7. TxMinúsculas

Descripción:
Devuelve el texto <t> con todos los caracteres pasados a minúsculas.

Sintaxis: TxMinúsculas(<t>

Tipo de dato devuelto: Texto

Ejemplo:
```
Texto = "SQL PYME";
TextoM = TxMinúsculas(Texto);     %% TextoM valdrá "sql pyme"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.8. TxMayúsculas

Descripción:
Devuelve el texto <t> con todos los caracteres pasados a mayúsculas.

Sintaxis: TxMayúsculas(<t>)

Tipo de dato devuelto: Texto

Ejemplo:
```
Texto = "sql pyme";
TextoM = TxMayúsculas(Texto);    %% TextoM valdrá "SQL PYME"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.9. TxDerecha

Descripción:
Devuelve los <n> últimos caracteres de <t>

Sintaxis: TxDerecha(<t>; <n>)

Tipo de dato devuelto: Texto

Ejemplo:
Texto = "SQL Pyme";
Texto1 = TxDerecha(Texto;4); %% Texto1 valdrá "Pyme"

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.10. TxReplica

Descripción:
Devuelve el texto <t> repetido <n> veces.

Sintaxis: TxReplica(<t>; <n>)

Tipo de dato devuelto: Texto

Ejemplo:
Texto = "-SQL Pyme-";
Texto1 = TxReplica(Texto; 3); %% Texto1 valdrá "-SQL Pyme--SQL Pyme--SQL Pyme-"

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.11. TxQuitaFormato

Descripción:
Devuelve el texto <t> sin las marcas del formato RTF.
Puede utilizarse para acceder a los campos memo de la aplicación que tienen formato.

Sintaxis: TxQuitaFormato(<t>)

Tipo de dato devuelto: Texto

Ejemplo:
```
ObservacionesT = Leer("CLIENTE"; "Observaciones"; "Codigo = 3");
% Observaciones T valdrá:
'{\rtf1\ansi\deff0{\fonttbl{\f0\fnil\fcharset0 @Arial Unicode MS;}}
\viewkind4\uc1\pard\lang3082\f0\fs16 Cliente dedicado a la importaci\'f3n-exportaci\'f3n de verduras.\par
\b Siempre debe firmar los albaranes de la mercanc\'eda recogida.\par
\b0\par
}
%

Observaciones = TxQuitaFormato(ObservacionesT);

% Observaciones valdrá:
'Cliente dedicado a la importación-
exportación de verduras.
Siempre debe firmar los
albaranes de la mercancía
recogida.
'
%
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.12. TxPosición

Descripción:
Devuelve la posición que ocupa <t1> en <t2>. Si <t1> no aparece se devuelve 0.
Se empieza a contar en 1 y se diferencia entre minúsculas y mayúsculas.

Sintaxis: TxPosición(<t1>;<t2>)

Tipo de dato devuelto: Número

Ejemplo:

```
Texto = "SQL Pyme";
p1 = TxPosición("Q"; Texto);       %% p1 valdrá 2
p2 = TxPosición("Pyme"; Texto);    %% p2 valdrá 5
p3 = TxPosición("q"; Texto);       %% p3 valdrá 0
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.13. TxLimpia

Descripción:
Devuelve <t1> sin los caracteres <t2> que pudiera tener por la izquierda y la derecha. Si no se especifica <t2> limpia los caracteres de control y los espacios en blanco.

Sintaxis: TxLimpia(<t1>[;<t2>])

Tipo de dato devuelto: Texto

Ejemplo:
```
t = "####120,25####";
tl = TxLimpia(t; "#");          %% tl valdrá 120,25
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.14. TxIzquierda

Descripción:
Devuelve los <n> primeros caracteres de <t>

Sintaxis: TxIzquierda(<t>; <n>)

Tipo de dato devuelto: Texto

Ejemplo:
```
Texto = "SQL Pyme";
Texto1 = TxIzquierda(Texto;3);    %% Texto1 valdrá "SQL"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.15. TxLimpiaDer

Descripción:
Devuelve <t1> sin los caracteres <t2> que pudiera tener por la derecha. Si no se especifica <t2> limpia los espacios en blanco.

Sintaxis: TxLimpiaDer(<t1>;<t2>)

Tipo de dato devuelto: Texto

Ejemplo:
```
t = "####120,25####";
tl = TxLimpiaDer(t; "#");          %% tl valdrá ####120,25
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.16. TxLimpiaIzq

Descripción:
Devuelve <t1> sin los caracteres <t2> que pudiera tener por la izquierda. Si no se especifica <t2> limpia los espacios en blanco.

Sintaxis: TxLimpiaIzq(<t1>;<t2>)

Tipo de dato devuelto: Texto

Ejemplo:
```
t = "####120,25####";
tl = TxLimpiaIzq(t; "#");          %% tl valdrá 120,25####
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.17. TxPrimeraMayúscula

Descripción:
Devuelve el texto <t> con todos los caracteres pasados a minúsculas menos el primero.

Sintaxis: TxPrimeraMayúscula(<t>)

Tipo de dato devuelto: Texto

Ejemplo:
Texto = "SQL PYME";
TextoM = TxPrimeraMayúscula(Texto); %% TextoM valdrá "Sql pyme"

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.18. TxPalabra

Descripción:
Devuelve la palabra de posición <n>, empezando a contar en 0, del texto <t1>, usando el primer carácter de <t2> como separador de palabras.

Sintaxis: TxPalabra(<t1>; <t2>; <n>)

Tipo de dato devuelto: Texto

Ejemplo:
```
Texto = "SQL PYME";
Palabra = TxPalabra(Texto; " "; 1);     %% Palabra valdrá "PYME"

Texto1 = "abcd;1234;efghi;5678";
Palabra1 = TxPalabra(Texto1; ";"; 3);     %% Palabra valdrá "5678" (un texto)
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.19. TxBorra

Descripción:
Devuelve <t> después de quitarle <n2> caracteres a partir de <n1>. Si no se especifica <n2> se borrarán todos los caracteres hasta el final.

Sintaxis: TxBorra(<t>; <n1>[; <n2>])

Tipo de dato devuelto: Texto

Ejemplo:
```
Texto = "SQL Pyme";
Texto1 = TxBorra(Texto; 4);     %% Texto1 valdrá "SQL"
Texto2 = TxBorra(Texto; 1; 4); %% Texto2 valdrá "Pyme"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.20. TxTrozo

Descripción:
Devuelve <n2> caracteres de <t> empezando a contar en la posición <n1>. Si no se especifica <n2> se devuelve el texto existente desde la posición <n1> (inclusive) hasta el final.
La primera posición es la 1.

Sintaxis: TxTrozo(<t>; <n1>[; <n2>])

Tipo de dato devuelto: Texto

Ejemplo:
```
Texto = "SQL Pyme";
Texto1 = TxTrozo(Texto; 5);    %% Texto1 valdrá "Pyme"
Texto2 = TxTrozo(Texto; 1; 3); %% Texto2 valdrá "SQL"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.21. TxRellenaDer

Descripción:
Se devuelve el texto <t> completado con el carácter <c> por la derecha hasta que tenga <n> caracteres en total.

Sintaxis: TxRellenaDer(<t>; <c>; <n>)

Tipo de dato devuelto: Texto

Ejemplo:
```
a1 = "Codigo";
b1 = "Nombre";
a2 = "SQP";
b2 = "SQL Pyme";
Texto = TxRellenaDer(a1; " "; 10) + TxRellenaDer(b1; " "; 10);
Texto = Texto + Intro + TxRellenaDer(a2; " "; 10) + TxRellenaDer(b2; " "; 10);
%  Texto será:
Codigo    Nombre
SQP       SQL Pyme
%
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.22. TxRellenaIzq

Descripción:
Se devuelve el texto <t> completado con el carácter <c> por la izquierda hasta que tenga <n> caracteres en total.

Sintaxis: TxRellenaIzq(<t>; <c>; <n>)

Tipo de dato devuelto: Texto

Ejemplo:

```
a1 = "Codigo";
b1 = "Nombre";
a2 = "SQP";
b2 = "SQL Pyme";
Texto = TxRellenaIzq(a1; " "; 10) + TxRellenaIzq(b1; " "; 10);
Texto = Texto + Intro + TxRellenaIzq(a2; " "; 10) + TxRellenaIzq(b2; " "; 10);
%  Texto será:
   Codigo    Nombre
      SQP  SQL Pyme
%
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.23. TxDesdeHasta

Descripción:
Devuelve una parte de <t> que comienza en la posición <n1> y finaliza en la posición <n2>. En <n2> pueden especificarse valores negativos que indican hasta que posición contando desde el final. El último carácter es la posición -1. La primera posición es la 1.

Sintaxis: TxDesdeHasta(<t>; <n1>; <n2>)

Tipo de dato devuelto: Texto

Ejemplo:
Texto = "SQL Pyme";
Texto1 = TxDesdeHasta(Texto; 5; 8); %% Texto1 valdrá "Pyme"
Texto2 = TxDesdeHasta(Texto; 1; -6); %% Texto2 valdrá "SQL

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.24. TxReemplaza

Descripción:
Devuelve el texto <t1> sustituyendo todas las apariciones del texto <t2> por el texto <t3>

Sintaxis: TxReemplaza(<t1>; <t2>; <t3>)

Tipo de dato devuelto: Texto

Ejemplo:
Texto = 'I One Pyme';
Texto1 = TxReemplaza(Texto; "I One"; "SQL"); %% Texto1 valdrá "SQL Pyme"

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.25. TxCuantasPalabras

Descripción:
Devuelve cuantas palabras contiene el texto <t1> usando el primer carácter de <t2> como separador de palabras.

Sintaxis: TxCuantasPalabras(<t1>; <t2>)

Tipo de dato devuelto: Número

Ejemplo:
```
Texto = "SQL PYME";
Palabras = TxCuantasPalabras(Texto; " ");    %% Palabras valdrá 2

Texto1 = "abcd;1234;efghi;5678";
Palabras1 = TxCuantasPalabras(Texto1; ";");    %% Palabras valdrá 4
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.26. TxSustituye

Descripción:
Devuelve el texto <t1> sustituyendo todas las apariciones de cada uno de los caracteres de <t2> por el carácter correspondiente de <t3> (el primero de <t2> por el primero de <t3>)

Sintaxis: TxSustituye(<t1>; <t2>; <t3>)

Tipo de dato devuelto: Texto

Ejemplo:
Minúsculas = "abcdefghijklmnñopqrstuvwxyz";
Mayúsculas = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ";
TextoMinúsculas = "sql pyme";
TextoMayúsculas = TxSustituye(TextoMinúsculas; Minúsculas; Mayúsculas); %% TextoMayúsculas valdrá "SQL PYME"

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.27. TxRestoPalabras

Descripción:
Devuelve el texto <t1> a partir de la palabra de posición <n>, sin incluirla.
Se empieza a contar en 0 y se usa el primer carácter de <t2> como separador de palabras.

Sintaxis: TxRestoPalabras(<t1>; <t2>; <n>)

Tipo de dato devuelto: Texto

Ejemplo:
```
Texto = "SQL PYME";
Palabra = TxRestoPalabras(Texto; " "; 0);    %% Palabra valdrá "PYME"

Texto1 = "abcd;1234;efghi;5678";
Palabra1 = TxRestoPalabras(Texto1; ";"; 1);    %% Palabra valdrá "efghi;5678"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de textos

9.28. TxContiene

Descripción:
Compara si texto contiene a subtexto, distinguiendo entre mayúsculas y minúsculas.
Retorna cierto o falso según se cumpla o no.

Sintaxis: TxContiene('texto'; 'subtexto')

Tipo de dato devuelto: Lógico

Ejemplo:
TxContiene('hola'; 'ho') %% retorna cierto
TxContiene('hola'; 'ol') %% retorna cierto
TxContiene('hola'; 'la') %% retorna cierto
TxContiene('hola'; 'm') %% retorna falso
TxContiene('hola'; 'L') %% retorna falso

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones de manipulación de textos

9.29. TxEmpiezaCon

Descripción:
Compara si texto empieza con subtexto, distinguiendo entre mayúsculas y minúsculas.
Retorna cierto o falso según se cumpla o no

Sintaxis: TxEmpiezaCon('texto'; 'subtexto')

Tipo de dato devuelto: Lógico

Ejemplo:
TxEmpiezaCon('hola'; 'ho') %% retorna cierto
TxEmpiezaCon('hola'; 'ol') %% retorna falso
TxEmpiezaCon('hola'; 'la') %% retorna falso
TxEmpiezaCon('Hola'; 'ho') %% retorna falso

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones de manipulación de textos

9.30. TxTerminaCon

Descripción:
Compara si texto finaliza con subtexto, distinguiendo entre mayúsculas y minúsculas.
Retorna cierto o falso según se cumpla o no.

Sintaxis: TxTerminaCon('texto'; 'subtexto')

Tipo de dato devuelto: Lógico

Ejemplo:
TxTerminaCon('hola'; 'ho') %% retorna falso
TxTerminaCon('hola'; 'ol') %% retorna falso
TxTerminaCon('hola'; 'la') %% retorna cierto
TxTerminaCon('hola'; 'LA') %% retorna falso

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones de manipulación de textos

9.31. TxCumpleMáscara

Descripción:
Compara el texto con la máscara, sin distinguir entre mayúsculas y minúsculas.
*Se soportan como comodines * (concuerda con cero o varios caracteres) y ? (concuerda con exactamente 1 carácter).*
Retorna cierto si el texto cumple con la máscara y falso si no.

Sintaxis: TxCumpleMáscara('texto'; 'máscara')

Tipo de dato devuelto: Lógico

Ejemplo:
TxCumpleMáscara('hola'; '') %% retorna cierto*
*TxCumpleMáscara('hola'; 'h*a') %% retorna cierto*
*TxCumpleMáscara('hola'; 'ho*la') %% retorna cierto*
TxCumpleMáscara('hola'; 'ho?la') %% retorna falso
TxCumpleMáscara('hola'; 'HolA') %% retorna cierto
TxCumpleMáscara('hola'; '?o') %% retorna cierto*
TxCumpleMáscara('hola'; '?a') %% retorna falso

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones de manipulación de textos

9.32. TxIgual

Descripción:
Compara los dos textos distinguiendo entre mayúsculas y minúsculas.
Retorna cierto si los dos textos son iguales y falso si no.

Sintaxis: TxIgual('texto1'; 'texto2')

Tipo de dato devuelto: Lógico

Ejemplo:

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones de manipulación de textos

9.33. TxCompara

Descripción:

Compara los dos textos según el criterio de ordenación del idioma actual y distinguiendo entre mayúsculas y minúsculas.
Retorna:

0 si los dos textos son iguales
< 0 (un número negativo) si texto1 < texto2
> 0 (un número positivo) si texto1 > texto2

Sintaxis: TxCompara('texto1'; 'texto2')

Tipo de dato devuelto: Número

Ejemplo:

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones de manipulación de textos

10. FUNCIONES DE CONVERSIÓN

10.1. EnLetra

Descripción:
Devuelve el número <n> expresado en letras en masculino.
Esta función se creó originalmente para expresar el euro en letras.
Es equivalente a la función EnLetraM(<n>)

Sintaxis: EnLetra(<n>)

Tipo de dato devuelto: Texto

Ejemplo:
x = 1234,56;
t = EnLetra(x); %% t valdrá "Mil doscientos treinta y cuatro con cincuenta y seis céntimos"

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de conversión

10.2. EnTexto

Descripción:
Convierte la variable <x> a tipo texto.
En el caso de números se añaden los puntos separadores de los miles. Para evitarlo puede usarse la función "Formatea".
En el caso de fechas se devuelve la fecha en formato "dd/mm/aaaa".

Sintaxis: EnTexto(<x>)

Tipo de dato devuelto: Texto

Ejemplo:
```
a = 1234,56;         at = EnTexto(a);  %% at valdrá "1.234,56"
b = "SQL Pyme";      bt = EnTexto(b);  %% bt valdrá "SQL Pyme"
c = cierto;          ct = EnTexto(c);  %% ct valdrá "cierto"
d = #(1; 2; 3);      dt = EnTexto(d);  %% dt valdrá "#(1;2;3)"
e = |15/09/2012|     et = EnTexto€;    %% et valdrá "15/09/2012"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de conversión

10.3. EnLetraF

Descripción:
Devuelve el número <n> expresado en letras en femenino y sin decimales.
Esta función se creó originalmente para expresar las pesetas en letras.

Sintaxis: EnLetraF(<n>)

Tipo de dato devuelto: Texto

Ejemplo:
```
x = 1234,56;
t = EnLetraF(x); %% t valdrá "Mil doscientas treinta y cinco"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de conversión

10.4. EnLetraM

Descripción:
Devuelve el número <n> expresado en letras en masculino.
Esta función se creó originalmente para expresar el euro en letras.
Es equivalente a la función EnLetra(<n>)

Sintaxis: EnLetraM(<n>)

Tipo de dato devuelto: Texto

Ejemplo:
x = 1234,56;
t = EnLetraM(x); %% t valdrá "Mil doscientos treinta y cuatro con cincuenta y seis céntimos"

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de conversión

10.5. EnNúmero

Descripción:
Convierte <t> a número si es posible. Si no es posible y se especifica <n> se devolverá este número, si no se especifica se devolverá un error.
Se ignoran los puntos separadores de los miles si los hubiera.

Sintaxis: EnNúmero(<t> [; <n>])

Tipo de dato devuelto: Número

Ejemplo:

```
x = "1.234,56";
i = "1234.56";
h = "1.234,56 €";
h2 = "1.234,56 €";


xn  = EnNúmero(x;0);  %% xn valdrá 1234,56
in   = EnNúmero(i;0);   %% i1 valdrá 123456
hn   = EnNúmero(h;0);  %% h1 valdrá 0
h2n = EnNúmero(h2);   %% Se devolverá un error y h2n no tendrá valor
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de conversión

10.6. Entrecomillar

Descripción:
Devuelve un texto con <x> (número, texto o valor lógico) añadiendo al principio y final el carácter que se especifique en <t>, si no se especifica se usará la comilla simple ('). Si <x> es un texto o un número se duplicará el carácter <t> si aparece dentro.

Sintaxis: Entrecomillar(<x>; [<t>])

Tipo de dato devuelto: Texto

Ejemplo:

```
x1 = 1234,12345;
x2 = "SQL Pyme";
x3 = "tendero/a";
x4 = cierto;
x5 = falso;

t1 = Entrecomillar(x1); %% t1 valdrá "'1234,12345'"
t2 = Entrecomillar(x2); %% t2 valdrá "'SQL Pyme'"
t3 = Entrecomillar(x3; "/"); %% t3 valdrá "/tendero//a/"
t4 = Entrecomillar(x4); %% t4 valdrá "'T'"
t5 = Entrecomillar(x5); %% t5 valdrá "'F'"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de conversión

10.7. FormateaHoras

Descripción:
Convierte el número <n> a texto con el formato horas:minutos:segundos.
Permite convertir entre tiempo centesimal y tiempo sexagesimal.
El número debe estar en segundos, las sqls devuelven el resultado en horas y hay que multiplicarlo por 3600 para pasárselo a esta función.
*Ejemplo: 1,395833328 * 3600 = 1:23:45*

Sintaxis: FormateaHoras(<n>)

Tipo de dato devuelto: Texto

Ejemplo:
x = FormateaHoras(7,59); %% x valdrá 07:35:24

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de conversión

10.8. Formatea

Descripción:

Convierte el número <n> a texto usando la máscara <t>.
El número es redondeado a los dígitos indicados en la máscara.
Si el número tiene más de 18 dígitos en la parte entera se usará notación científica independientemente de la máscara.
Si no se especifica <t> se supone la máscara ",0.#####".
Formatos máscara para números:
0: Muestra los 0 no significativos.
#: No muestra los 0 no significativos.
.: Indica la posición del separador decimal.
,: Indica que se incluirá el separador de miles cada tres dígitos. La posición en la que aparezca no importa.
e+: Notación científica. Pueden indicarse de uno a cuatro ceros a continuación para indicar los dígitos del exponente.
e-: Notación científica. Igual que "e+" pero sólo muestra el signo del exponente si es negativo.
;: Separador secciones números positivos, negativos o cero. Debe introducirse dentro de la máscara. Si sólo hay una sección es para todos, si hay dos la primera es para los números positivos y cero y la segunda para los negativos, si hay tres la primera es para positivos, la segunda para negativos y la tercera para cero. Si se especifica máscara para números negativos no se mostrará en esta el signo menos (-). Puede especificarse una máscara para los positivos, ninguna para los negativos (se deja en blanco y se usará la de los positivos) y una para cero.
Cualquier carácter: Si especifica un carácter distinto de los aquí indicados aparecerá tal cual.

Sintaxis: Formatea([<t>;] <n>)

Tipo de dato devuelto: Texto

Ejemplo:

```
x1 = 0,4;
f11 = Formatea("00.00"; x1);              %% f11 valdrá "0,40"
f12 = Formatea("0.##"; x1);               %% f12 valdrá "0,4"
x2 = 1234,123456789;
f21 = Formatea(",0.0000"; x2);            %% f21 valdrá "1.234,1235"
f22 = Formatea(",0.00 €"; x2);            %% f22 valdrá "1.234,12 €"
f23 = Formatea(x2);                       %% f21 valdrá "1.234,12346"
x3 = 1500;
f31 = Formatea("00e-00"; x3);             %% f31 valdrá "15e02"
f32 = Formatea("00e+00"; x3);             %% f32 valdrá "15e+02"
x4 = 0,1234;
f41 = Formatea("0.00E-0"; x4);            %% f41 valdrá "1,23E-1"
x5 = 1234567890123456789;
f51 = Formatea(",0.00"; x5);              %% f51 valdrá "1,23456789012346E18"
x6 = -2500;
f61 = Formatea("0.00"; x6);               %% f61 valdrá "-2500,00"
f62 = Formatea(",0.00;(,0.00)"; x6);      %% f62 valdrá "(2.500,00)"
f63 = Formatea(",0.00;;Cero";x6);         %% f63 valdrá "-2.500"
x7 = 0;
f71 = Formatea(",0.00;(,0.00);Cero"; x7); %% f71 valdrá "Cero"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de conversión

10.9. FormateaPrecio

Descripción:

Devuelve el número <n> como un texto formateado según las siguientes opciones:
<l1>: Separar miles, valores posibles: cierto/falso. Si no se especifica es cierto.
<l2>: Imprimir cero, valores posibles cierto/falso. Si no se especifica es cierto.
<n1>: Número decimales. Si no se especifica son 2.

Sintaxis: FormateaPrecio(<n>;[[[<l1>]; <l2>]; <n1>])

Tipo de dato devuelto: Texto

Ejemplo:

```
x = 1234,12345;
t1 = FormateaPrecio(x);                  %% t1 valdrá "1.234,12"
t2 = FormateaPrecio(x;falso;falso;3);    %% t2 valdrá "1234,123"
t3 = FormateaPrecio(x; falso);           %% t3 valdrá "1234,12"
x2 = 0;
t21 = FormateaPrecio(x2; falso;falso); %% t21 valdrá ""
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de conversión

10.10. EnHexadecimal

Descripción:
Conversión de un número en base decimal a base hexadecimal.
Puede indicarse el número mínimo de dígitos a generar.

Sintaxis: EnHexadecimal(<número> [;<mínimo_dígitos])

Tipo de dato devuelto: Texto

Ejemplo:
```
x1 = EnHexadecimal(10);   %% x1 valdrá A
x2 = EnHexadecimal(10; 3);  %% x2 valdrá 00A
x3 = EnHexadecimal(16777000; 2); %% x3 valdrá FFFF28
x4 = EnHexadecimal("Dk");   %% Se generará un error
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones de conversión

11. FUNCIONES DE MANIPULACIÓN DE LISTAS (ARRAYS)

11.1. LAñade

Descripción:
Se añade <x> a <lista> como un nuevo elemento al final de la misma.
Se devuelve la posición en la que se añadió.

Sintaxis: LAñade(<lista>; <x>)

Tipo de dato devuelto: Número

Ejemplo:
```
Lista = #("A"; "B"; "C");
Lañade(Lista; "D");                 %% Lista valdrá #("A"; "B"; "C; "D")
Posición = Lañade(Lista; "H");      %% Lista valdrá #("A"; "B"; "C; "D"; "H"), Posición valdrá 5
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.2. LAplica

Descripción:
Se itera lista asignando a la variable "params[1]" el índice del item actual y ejecutando bloque.
No debe usarse un "LAplica" dentro de otro porque la variable sería la misma para los dos.

Sintaxis: LAplica(<lista>; <bloque>)

Tipo de dato devuelto: Nada

Ejemplo:

```
Lista = #("A"; "B"; "C"; "D");
Texto = "";
Laplica(Lista; {
 %% Comprobación de primer valor para no incluir el separador
 si (Texto <> "") entonces {
   Texto = Texto + ", ";
 };
 Texto = Texto + Lista[params[1]];
});

%% Texto valdrá "A, B, C, D"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.3. RAñade

Descripción:
Se crea el campo <t> en el registro <r>. Se devuelve la posición del nuevo campo en el registro, empezando a contar en 1 y siguiendo un orden alfabético por nombre del campo.

Sintaxis: RAñade(<r>; <t>)

Tipo de dato devuelto: Número

Ejemplo:
```
Registro = #[Código: 1 Nombre: "Andrea"];
n = RAñade(Registro;"Apellidos");     %% n valdrá 1
Registro.Apellidos = "Suárez Pérez";
%
Registro valdrá: #[Apellidos: "Suárez Pérez" Código: 1 Nombre: "Andrea"
%
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.4. LBorra

Descripción:
Elimina todos los items de <lista>

Sintaxis: LBorra(<lista>)

Tipo de dato devuelto: Nada

Ejemplo:
```
Lista = #("A"; "B"; "C");   %% Lista valdrá #("A"; "B"; "C")
LBorra(Lista);              %% Lista valdrá #()
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.5. LCrea

Descripción:
Crea y devuelve una lista vacía.

Sintaxis: LCrea

Tipo de dato devuelto: Lista

Ejemplo:

```
Lista1 = Lcrea;

%% Otras formas de crear una lista:
Lista2 = #();                %% Es igual a LCrea, una lista vacía
Lista3 = #(1; 2; 3);         %% Lista de números
Lista4 = #("A"; "B"; "C");   %% Lista de textos
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.6. LConvierte

Descripción:
Convierte todos elementos de tipo texto que representan un número en tipos de datos numéricos.

Sintaxis: LConvierte(<lista>)

Tipo de dato devuelto: Nada

Ejemplo:
Lista = #(11;'12,6';"1.000,32"; "005"; 'a');
LConvierte(Lista); %% Lista valdrá #(11; 12,6; 1000,32; 5; "a")

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.7. LCarga

Descripción:

LCarga(lista; 'fichero'[; parametros])
Se lee el fichero de texto indicado en la ruta <t> añadiendo cada línea como un elemento de la lista <lista>.
La lista debe existir con anterioridad. Los elementos añadidos siempre son de tipo texto, no todos los elementos de la lista tienen que ser del mismo tipo.

LCarga(lista; binario[; parametros])
Carga el contenido del fichero o binario en la lista.

Los dos primeros parámetros son obligatorios, el tercero es opcional.
Parametros es un registro que permite especificar diversas opciones:
Codificación (texto) (posibles valores 'ansi', 'utf8', 'utf16'): por defecto ansi.

Si el contenido es utf con marca BOM, detecta automáticamente la codificación sin necesidad de especificarla.

Sintaxis: LCarga(<lista>; <t> | <binario> [;parametros])

Tipo de dato devuelto: Nada

Ejemplo:

```
% Fichero ejemplo:
10 100
20 200
Línea 3
%
Lista = #(10; 20; 30);
LCarga(Lista; "C:\Distrito\Pyme\Fichero.TXT");
%% Lista valdrá #(10; 20; 30; "10 100"; "20 200"; "Línea 3")

%% Ejemplo 2:
texto = 'Hola €9#Ñ&Á?É Í_Ó-Ú.Z$Z!Z\Z/Z%Z<Z>Z áéíóúàèìòùäëïöüâêîôûñ-ÁÉÍÓÚÀÈÌÒÙÄËÏÖÜÂÊÎÔÛÑ-minus-MAYUS çÇ adioS.';
TxGraba(texto; 'C:\Distrito\prueba.txt');

lista = LCrea;
LCarga(lista; 'C:\Distrito\prueba.txt');
LGraba(lista; 'C:\Distrito\prueba_u8.txt'; #[codificación: 'utf8']); %% por defecto le graba el BOM
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.8. LGraba

Descripción:

LGraba(lista; 'fichero'; parametros)
Se graba <lista> en el fichero de texto indicado en la ruta <t>.
Cada elemento de la lista estará en una línea.

LGraba(lista; binario; parametros)
Craba el contenido de la lista en el fichero o binario.

Los dos primeros parámetros son obligatorios, el tercero es opcional.
Parametros es un registro que permite especificar diversas opciones:
Codificación (texto) (posibles valores 'ansi', 'utf8', 'utf16'): por defecto ansi
BOM (lógico): indica si se graba la marca BOM (byte order mark) al principio del fichero si la codificación lo soporta (por defecto cierto)

Sintaxis: LGraba(<lista>; <t> | <binario> [; parametros])

Tipo de dato devuelto: Nada

Ejemplo:

```
Lista = #(10; 20; 30; "Texto 1"; "Texto 2");
LGraba(Lista; "C:\Distrito\Pyme\Fichero.TXT");
% El fichero tendrá este contenido
10
20
30
Texto 1
Texto 2
%

%% Ejemplo 2:
texto = 'Hola €9#Ñ&Á?É Í_Ó-Ú.Z$Z!Z\Z/Z%Z<Z>Z áéíóúàèìòùäëïöüâêîôûñ-ÁÉÍÓÚÀÈÌÒÙÄËÏÖÜÂÊÎÔÛÑ-minus-MAYUS çÇ adioS.';
TxGraba(texto; 'C:\Distrito\prueba.txt');

lista = LCrea;
LCarga(lista; 'C:\Distrito\prueba.txt');
LGraba(lista; 'C:\Distrito\prueba_u8.txt'; #[codificación: 'utf8']); %% por defecto le graba el BOM
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.9. LInserta

Descripción:
Inserta el valor <x> en la posición indicada por <n> de la lista <lista>. Se empieza a contar en 1. Todos los elementos de la lista desde la posición de inserción serán desplazados un lugar.

Sintaxis: LInserta(<lista>; <n>; <x>)

Tipo de dato devuelto: Nada

Ejemplo:
```
Lista = #("A"; "B"; "C");
LInserta(Lista;2; "Z");  %% Lista valdrá #("A"; "Z"; "B"; "C");
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.10. RNombreCampo

Descripción:
Se devuelve el nombre del campo con posición <n> en el registro <r>. La posición de un campo depende de su nombre, se colocan alfabéticamente y se empieza a contar en 1.
Si el campo no existe se produce un error.

Sintaxis: RNombreCampo(<r>; <n>)

Tipo de dato devuelto: Texto

Ejemplo:
```
Registro = #[Código: 25 Nombre: "Verónica" Apellidos: "Rodríguez Pérez"];
t1 = RNombreCampo(Registro; 1);          %% t1 valdrá "Apellidos"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.11. LOrdena

Descripción:
Se ordena <lista>. Los números van antes que los textos.
No se diferencia entre mayúsculas y minúsculas.

Sintaxis: LOrdena(<lista>)

Tipo de dato devuelto: Nada

Ejemplo:

```
%% Ejemplo con una lista
Lista1=#(11;'a';7;12,6;6);
LOrdena(Lista1);            %% Lista1 valdrá #(6;7;11;12,6;'a')

%% Ejemplo con una lista de registros
Lista = #(
 #[Nombre: "Sonia"   Código: 50   Teléfono: "999123456"];
 #[Nombre: "María"   Código: 20   Teléfono: "999123458"];
 #[Nombre: "Irene"   Código: 30   Teléfono: "999123457"]
);
%
Se crea una nueva lista, ListaOrdenada, y se añade cada registro como una lista
con el campo por el que se quiere ordenar como primer elemento, el resto de los
campos se añaden en un registro.
%
ListaOrdenada = #();
LAplica(Lista; {
 LAñade(ListaOrdenada;
  #(Lista[params[1]].Código ;
    #[
     Nombre: Lista[params[1]].Nombre
     Teléfono: Lista[params[1]].Teléfono
    ]
  )
 );
});
LOrdena(ListaOrdenada);
Mensaje(ListaOrdenada[3][2].Nombre);  %% Esto devolverá Sonia
%
La lista quedará así:
ListaOrdenada = #(
 #(20; #[Nombre: María   Teléfono: "999123458"]);
 #(30; #[Nombre: Irene   Teléfono: "999123457"]);
 #(50; #[Nombre: Sonia   Teléfono: "999123456"])
);
%
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.12. Lposición

Descripción:
Busca el valor en la lista y retorna la posición (0 si no lo encuentra)

Sintaxis: LPosición(<lista>; <valor>)

Tipo de dato devuelto: Número

Ejemplo:
```
Lista = #(11;7;12;13);
n1 = Lposición(Lista;7);    %% n1 valdrá 2
n2 = Lposición(Lista;"12");   %% n2 valdrá 0, porque no existe el elemento "12" (tipo texto) en la lista
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.13. RPosiciónCampo

Descripción:
Se devuelve la posición del campo con nombre <t> en el registro <r>. En el caso de no existir el campo se devuelve 0.
La posición de un campo depende de su nombre, se colocan alfabéticamente y se empieza a contar en 1.
No se diferencian mayúsculas y minúsculas ni letras acentuadas o no.

Sintaxis: RPosiciónCampo(<r>; <t>)

Tipo de dato devuelto: Número

Ejemplo:
```
Registro = #[Código: 25 Nombre: "Verónica" Apellidos: "Rodríguez Pérez"];
n1 = RPosiciónCampo(Registro; "CODIGO");        %% n1 valdrá 2
n2 = RPosiciónCampo(Registro; "Teléfono");      %% n2 valdrá 0
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.14. LQuita

Descripción:
Se elimina el item con índice <n> de la lista <lista>.
Las listas siempre empiezan a contar en 1.

Sintaxis: LQuita(<lista>; <n>)

Tipo de dato devuelto: Nada

Ejemplo:
```
Lista = #("A"; "B"; "C");
LQuita(Lista; 2);        %% Lista valdrá #("A"; "C")
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.15. RQuita

Descripción:
Elimina el campo <c> del registro <r>. El campo puede indicarse por su nombre o su posición, la posición de un campo depende de su nombre, se colocan alfabéticamente y empieza a contar en 1.

Sintaxis: RQuita(<r>; <c>)

Tipo de dato devuelto: Nada

Ejemplo:
```
Registro1 = #[Código: 25 Nombre: "Verónica" Apellidos: "Rodríguez Pérez"];
RQuita(Registro1; 2);  %% Registro valdrá #[Apellidos: "Rodríguez Pérez" Nombre: "Verónica"];
Registro2 = #[Código: 25 Nombre: "Verónica" Apellidos: "Rodríguez Pérez"];
RQuita(Registro2; "Apellidos");  %% Registro valdrá #[Código: 25 Nombre: "Verónica"];
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.16. LTamaño

Descripción:
Devuelve el número de items de la lista

Sintaxis: LTamaño(<lista>)

Tipo de dato devuelto: Número

Ejemplo:
```
Lista = #("A"; "B"; "C");
x = LTamaño(Lista);          %% x valdrá 3
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.17. RTamaño

Descripción:
Retorna cuántos campos tiene el registro

Sintaxis: RTamaño(<registro>)

Tipo de dato devuelto: Número

Ejemplo:
Registro = #[Campo1: "valor1" Campo2: 100 Campo3: 101 campo4:"text" t1exto:"valor"];
n = Rtamaño(Registro); %% n valdrá 5

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.18. RValorCampo

Descripción:
Se devuelve el contenido del campo <c> del registro <r>. El campo puede indicarse por su nombre o su posición, la posición de un campo depende de su nombre, se colocan alfabéticamente y se empieza a contar en 1.
Si el campo no existe se produce un error.

Sintaxis: RValorCampo(<r>; <c>)

Tipo de dato devuelto: Variable

Ejemplo:
```
Registro = #[Código: 25 Nombre: "Verónica" Apellidos: "Rodríguez Pérez"];
x1 = RValorCampo(Registro; 2);          %% x1 valdrá 25
x2 = RValorCampo(Registro; "Apellidos");  %% x2 valdrá "Rodríguez Pérez"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.19. ChequeaTipoItem

Descripción:
Comprueba que todos los elementos de la lista <lista> sean del tipo <tipo>. Si lo fueran no devuelve nada y continua la ejecución del script, si no lo fueran da un error y devuelve "Fichero".

Sintaxis: ChequeaTipoItem(<lista>; <tipo>)

Tipo de dato devuelto: Nada/Error

Ejemplo:
```
Lista = #(10; 20; "Texto 1"; "25/02/2009");
Tipos = TipoLista(Lista); %% Tipos valdrá #(TNúmero; TNúmero; TTexto TTexto);
h = ChequeaTipoItem(Lista; TNúmero);
%% h no existirá porque da error la instrucción ChequeaTipoItem
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.20. TipoLista

Descripción:
Devuelve una lista con los tipos de datos de cada elemento de la lista pasada como parámetro.

Sintaxis: TipoLista(<lista>)

Tipo de dato devuelto: Lista

Ejemplo:
Lista = #(10; 20; "Texto 1"; "25/02/2009");
Tipos = TipoLista(Lista); %% Tipos valdrá #(TNúmero; TNúmero; TTexto; TTexto);

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.21. TxEmpaqueta

Descripción:
Devuelve una cadena de texto con los elementos de <lista> separados por el texto <t>.

Sintaxis: TxEmpaqueta(<lista>; <t>)

Tipo de dato devuelto: Texto

Ejemplo:
```
Lista = #("Ana"; "Alicia";20;30;40;50);
t = Txempaqueta(Lista;","); %% t valdrá Ana,Alicia,20,30,40,50
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.22. TxDesempaqueta

Descripción:
Devuelve una lista con cada parte de <texto1> separada por <texto2> como un elemento de la misma.

Sintaxis: TxDesempaqueta(<texto1>; <texto2>);

Tipo de dato devuelto: Lista

Ejemplo:
Texto = "Ana,Alicia,30,40,50";
Lista = Txdesempaqueta(Texto;","); %% Lista valdrá #("Ana";"Alicia";"30";"40";"50")

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de listas

11.23. RExisteCampo

Descripción:
Retorna cierto o falso según exista o no el campo <c> en el registro <r>

Sintaxis: RExisteCampo(<r>; <c>)

Tipo de dato devuelto: lógico

Ejemplo:
```
registro = #[
  campo1: 10
  campo2: 'valor'
];
e1 = RExisteCampo(registro; 'campo2');  %% cierto
e2 = RExisteCampo(registro; 'campo10'); %% falso
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones de manipulación de listas

12. FUNCIONES DE MANIPULACIÓN DE FECHAS

12.1. MesDe

Descripción:
Devuelve el número del mes (1 a 12) de la fecha <t>.

Sintaxis: MesDe(<t>)

Tipo de dato devuelto: Número

Ejemplo:
```
f1 = "7/02/2016";
x1 = MesDe(f1);   %% x1 valdrá 2
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.2. ReformateaFecha

Descripción:

Intenta convertir a fecha el texto <t>. Si le faltan las "/" las añade, si no se indica el año supone el actual.
Pueden usarse los atajos para fechas de la aplicación:
- Indicar el signo y el número de días. Esto devolverá la fecha de tantos días antes o después (- o +) del día actual.
- Indicar el nombre de un día. Valores posibles: hoy, ayer, mañana,

Se produce un error si no puede convertirse el texto introducido a una fecha válida.

Sintaxis: ReformateaFecha(<t>)

Tipo de dato devuelto: Texto

Ejemplo:

```
f1 = ReformateaFecha("1512");     %% f1 valdrá "15/12/2011" suponiendo que la fecha actual sea del 2011
f2 = ReformateaFecha("15/12");    %% f2 valdrá "15/12/2011" suponiendo que la fecha actual sea del 2011
f3 = ReformateaFecha("151216");   %% f3 valdrá "15/12/2016"
f4 = ReformateaFecha("15/12/16"); %% f4 valdrá "15/12/2016"
f5 = ReformateaFecha("+0");       %% f5 valdrá "30/05/2012" suponiendo que la fecha actual sea el 30/05/2012
f6 = ReformateaFecha("Ayer");     %% f6 valdrá "29/05/2012"  suponiendo que la fecha actual sea el 30/05/2012
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.3. SemanaDe

Descripción:
Devuelve el número de semana dentro del año (de 1 a 54) en que se encuentra la fecha <t>.
Se usa el criterio americano para calcular la semana del año, esto es la que contenga el día 1.
En España se usa la normativa ISO 8601 que especifica que la primera semana del año es aquella que contiene el primer jueves del año.
Existe un script para calcular el número de semana acorde a la normativa ISO 8601.

Sintaxis: SemanaDe(<t>)

Tipo de dato devuelto: Número

Ejemplo:
```
f1 = "7/12/2016";
f2 = "7/12/2014";
x1 = SemanaDe(f1);   %% x1 valdrá 50
x2 = SemanaDe(f2);   %% x2 valdrá 49
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.4. DiferenciaEnDías

Descripción:
Devuelve el número de días entre las dos fechas pasadas. Da igual el orden siempre devuelve un valor positivo. Si se quiere obtener un resultado negativo cuando una fecha es menor que la otra puede usarse "FechaEnNúmero" para convertir las fechas y después restarlas como números normales.

Sintaxis: DiferenciaEnDías(<t1>; <t2>)

Tipo de dato devuelto: Número

Ejemplo:
```
f1 = "15/04/16";
f2 = "13/04/16";
d = DiferenciaEnDías(f1; f2);   %% d valdrá 2
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.5. DiferenciaEnHoras

Descripción:

Devuelve el número de horas, minutos y segundos entre dos valores de fecha y hora.
La parte de la fecha se ignora, sólo se comparan las horas. A <fh1> se le resta <fh2> por lo que pueden darse valores negativos si <fh2> es mayor que <fh1>.
Puede pasarse una lista de valores para cada parámetro, en ese caso se devolverá una lista de resultados.
Para poder convertir estas horas a número y poder formular con ellas puede usarse "HoraEnNúmero".

Sintaxis: DiferenciaEnHoras(<fh1>; <fh2>)

Tipo de dato devuelto: Hora

Ejemplo:

```
f1 = "15/04/16 13:05";
f2 = "13/04/16 16:14:55";
d = DiferenciaEnHoras(f1; f2);   %% d valdrá "-3:-09:-55"

f12 = #("15/04/16 13:05"; "12/04/16 12:00");
f22 = #("13/04/16 16:14:55"; "12/04/16 15:00");
dd = DiferenciaEnHoras(f12; f22);   %% dd valdrá #("-3:-09:-55"; "3:00:00")
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.6. SiguienteMes

Descripción:
Devuelve la fecha correspondiente al primer día del mes siguiente a la fecha <t>.

Sintaxis: SiguienteMes(<t>)

Tipo de dato devuelto: Texto

Ejemplo:
```
f1 = "7/03/2016";
f2 = "20/03/2015";
f3 = "14/12/2014";
x1 = SiguienteMes(f1);   %% x1 valdrá "01/04/2016"
x2 = SiguienteMes(f2);   %% x2 valdrá "01/04/2015"
x3 = SiguienteMes(f3);   %% x3 valdrá "01/01/2015"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.7. ÚltimoDíaMes

Descripción:
Devuelve la fecha correspondiente al último día del mes indicado por la fecha <t>.
Nos da el fin de mes.

Sintaxis: ÚltimoDíaMes(<t>)

Tipo de dato devuelto: Texto

Ejemplo:

```
f1 = "7/02/2016";
f2 = "20/02/2015";
f3 = "14/12/2014";
x1 = ÚltimoDíaMes(f1);   %% x1 valdrá "29/02/2016"
x2 = ÚltimoDíaMes(f2);   %% x2 valdrá "28/02/2015"
x3 = ÚltimoDíaMes(f3);   %% x3 valdrá "31/12/2014"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.8. AnteriorMes

Descripción:
Devuelve la fecha correspondiente al último día del mes anterior a la fecha <t>.

Sintaxis: AnteriorMes(<t>)

Tipo de dato devuelto: Texto

Ejemplo:
```
f1 = "7/03/2016";
f2 = "20/03/2015";
f3 = "8/01/2014";
x1 = AnteriorMes(f1);   %% x1 valdrá "29/02/2016"
x2 = AnteriorMes(f2);   %% x2 valdrá "28/02/2015"
x3 = AnteriorMes(f3);   %% x3 valdrá "31/12/2013"

%% Obtención del primer y último día del mes de hace x meses
meses = 3;
fecha = FechaHoy;
final_mes = FechaHoy;
i = 1; mientras {i <= meses} haz {
  final_mes = AnteriorMes(final_mes);
  i = i + 1;
};
principio_mes = PrimerDíaMes(final_mes);
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.9. HoraEnTexto

Descripción:
Devuelve la hora correspondiente al número de segundos <n> transcurridos desde las "00:00".
Para obtener de una hora el número de segundos se usa la instrucción "HoraEnNúmero".

Sintaxis: HoraEnTexto(<n>)

Tipo de dato devuelto: Texto

Ejemplo:
```
x1 = HoraEnTexto(120);  %% x1 valdrá "0:02:00"
x2 = HoraEnTexto(9525); %% x2 valdrá "2:38:45"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.10. HoraEnNúmero

Descripción:

Devuelve la hora convertida en el número de segundos transcurridos desde las "00:00".
Permite operar con horas, minutos y segundos.
Puede usarse para transformar tiempo en sistema sexagesimal a sistema centesimal dividiendo el resultado de la función entre 3600, se obtienen horas en centesimal.
Para obtener el formato de horas a partir del resultado de esta función se usa la instrucción "HoraEnTexto".

Sintaxis: HoraenNúmero(<t>)

Tipo de dato devuelto: Número

Ejemplo:

```
x1 = HoraEnNúmero('0:02:00');       %% x1 valdrá 120
x2 = HoraEnNúmero('07:35:24')/3600;   %% x2 valdrá 7,59
x3 = HoraEnNúmero('09:59:00')/3600;   %% x3 valdrá 9,98333
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.11. PrimerDíaMes

Descripción:
Devuelve la fecha correspondiente al primer día del mes indicado por la fecha <t>.

Sintaxis: PrimerDíaMes(<t>)

Tipo de dato devuelto: Texto

Ejemplo:
```
f1 = "7/03/2016";
x1 = PrimerDíaMes(f1);   %% x1 valdrá "01/03/2016"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.12. TrimestreDe

Descripción:
Devuelve el trimestre (de 1 a 4) en que se encuentra la fecha <t>.

Sintaxis: TrimestreDe(<t>)

Tipo de dato devuelto: Número

Ejemplo:
```
f1 = "7/12/2016";
f2 = "7/8/2014";
x1 = TrimestreDe(f1);   %% x1 valdrá 4
x2 = TrimestreDe(f2);   %% x2 valdrá 3
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.13. EsAñoBisiesto

Descripción:
Devuelve cierto si un año es bisiesto.
Los años divisibles por 4 son bisiestos, pero cada 400 años se deben eliminar 3 bisiestos. Para ello, no son bisiestos los que se dividen por 100, menos los que se dividen por 400, que sí son bisiestos.

Sintaxis: EsAñoBisiesto(<t>)

Tipo de dato devuelto: Lógico

Ejemplo:
```
f1 = "15/04/2100";
f2 = "31/12/2000";
x1 = EsAñoBisiesto(f1);  %% x1 valdrá falso
x2 = EsAñoBisiesto(f2);  %% x2 valdrá cierto
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.14. EsMismoMes

Descripción:
Devuelve cierto si el año y mes de las fechas <t1> y <t2> es el mismo.

Sintaxis: EsMismoMes(<t1>; <t2>)

Tipo de dato devuelto: Lógico

Ejemplo:
```
f1 = "15/04/2100";
f2 = "15/04/2099";
f3 = "31/12/2020";
f4 = "03/12/2020";
x1 = EsMismoMes(f1; f2);  %% x1 valdrá falso
x2 = EsMismoMes(f3; f4);  %% x2 valdrá cierto
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.15. DíaDe

Descripción:
Devuelve el día del mes (1 a 31) de la fecha <t>.

Sintaxis: DíaDe(<t>)

Tipo de dato devuelto: Número

Ejemplo:
```
f1 = "7/02/2016";
x1 = DíaDe(f1);   %% x1 valdrá 7
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.16. DíasMes

Descripción:
Devuelve el número de días del mes indicado por la fecha <t>.

Sintaxis: DíasMes(<t>)

Tipo de dato devuelto: Número

Ejemplo:
```
f1 = "03/02/2016";
f2 = "15/02/2015";
f3 = "1/12/2014";
x1 = DíasMes(f1);   %% x1 valdrá 29
x2 = DíasMes(f2);   %% x2 valdrá 28
x3 = DíasMes(f3);   %% x3 valdrá 31
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.17. DíasAño

Descripción:
Devuelve el número de días del año indicado por la fecha <t>.

Sintaxis: DíasAño(<t>)

Tipo de dato devuelto: Número

Ejemplo:
```
f1 = "03/02/2016";
f2 = "15/02/2015";
f3 = "1/12/2014";
x1 = DíasAño(f1);   %% x1 valdrá 366
x2 = DíasAño(f2);   %% x2 valdrá 365
x3 = DíasAño(f3);   %% x3 valdrá 365
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.18. DíaDeSemana

Descripción:
Devuelve el día de la semana (de 1 a 7) en que se encuentra la fecha <t>.
Empieza el lunes (1) y acaba el domingo (7).

Sintaxis: DíaDeSemana(<t>)

Tipo de dato devuelto: Número

Ejemplo:

```
f1 = "7/12/2016";
f2 = "7/8/2014";
x1 = DíaDeSemana(f1);          %% x1 valdrá 3
x2 = DíaDeSemana(f2);          %% x2 valdrá 4
x3 = FormateaFecha("dddd"; f1);   %% x3 valdrá "miércoles"
x4 = FormateaFecha("dddd"; f2);   %% x4 valdrá "jueves"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.19. AñoDe

Descripción:
Devuelve el año de la fecha <t>.

Sintaxis: AñoDe(<t>)

Tipo de dato devuelto: Número

Ejemplo:
```
f1 = "7/02/2016";
x1 = AñoDe(f1);   %% x1 valdrá 2016
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.20. FechaHoy

Descripción:
Devuelve la fecha de hoy

Sintaxis: FechaHoy

Tipo de dato devuelto: Texto

Ejemplo:
t = FechaHoy; %% t valdrá la fecha de hoy en formato "dd/mm/aaaa"

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.21. FechaEnTexto

Descripción:
Formato fecha devuelto: dd/mm/yyyy
El parámetro pasado se tomará como el número de días transcurridos desde el 30/12/1899.

Sintaxis: FechaEnTexto(<n>)

Tipo de dato devuelto: Texto

Ejemplo:
```
dia1 = FechaEnTexto(FechaEnNumero(FechaHoy)+21);
%% dia1 contendrá la fecha de dentro de 21 días.

Dia2 = FechaEnTexto(40070);
%% dia2 contendrá '14/09/2009'
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.22. FechaEnNúmero

Descripción:
Convierte una fecha en texto válida en una fecha numérica (permite sumar y restar fechas y sumarles y restarles días).
El número devuelto son los días pasados desde el 30/12/1899.
La fecha pasada por parámetro puede tener las siguientes sintaxis:
DD/MM/AAAA
D/M/AAAA
DD/MM (sobreentiende el año actual)
D/M (sobreentiende el año actual)
Cualquier otro formato de fecha produce un error.

Sintaxis: FechaEnNúmero(<t>)

Tipo de dato devuelto: Número

Ejemplo:
numDia1 = FechaEnNumero('31/12/1899');
%% numDia1 contendrá 1

numDia2 FechaEnNumero(FechaHoy)-FechaEnNumero('18/05/2009');
%% numDia2 contendrá la diferencia de días entre hoy y el 18/05/09.

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.23. FormateaFecha

Descripción:

Conversión del texto <t2> al formato especificado por <t1>. Si no se especifica <t1> se utiliza "dd/mm/yyyy".
Formatos máscara para fechas:
d: Muestra el día como un número de 1 o dos digitos (1-31).
dd Muestra el día com un número de 2 dígitos (01-31).
ddd: Muestra el nombre abreviado del día de la semana (lun, mar, mié, jue, vie, sáb, dom).
dddd: Muestra el nombre completo del día de la semana (lunes-domingo).
ddddd: Muestra la fecha con formato dd/mm/yyyy.
dddddd: Muestra la fecha usando el formato de fecha larga configurado en Windows. Normalmemente es: dddd, dd ' de ' mmmm ' de ' yyyy.
m: Muestra el mes como un número de uno o dos dígitos (1-12).
mm: Muestra el mes como un número de dos dígitos (01-12).
mmm: Muestra el nombre abreviado del mes (ene-dic).
mmmm: Muestra el nombre completo del mes (enero-diciembre).
yy: Muestra el año como un número de dos dígitos.
yyyy: Muestra el año cmo un número de cuatro dígitos (0000-9999).
/: Muestra el separador de fecha configurado en Windows. Normalmente es "/".
'xx' ó "xx": Los caracteres encerrados por comillas simples o dobles se muestran tal como se indican y no afectan al formato.

Sintaxis: FormateaFecha([<t1>;]<t2>)

Tipo de dato devuelto: Texto

Ejemplo:

```
f1 = "15/4/16";
f10 = FormateaFecha(f1);            %% f10 valdrá "15/04/2016"
f11 = FormateaFecha("d"; f1);       %% f11 valdrá "15"
f12 = FormateaFecha("dd"; f1);      %% f12 valdrá "15"
f13 = FormateaFecha("ddd"; f1);     %% f13 valdrá "vie"
f14 = FormateaFecha("dddd"; f1);    %% f14 valdrá "viernes"
f15 = FormateaFecha("ddddd"; f1);   %% f15 valdrá "15/04/2016"
f16 = FormateaFecha("dddddd"; f1);  %% f16 valdrá "viernes, 15 de abril de 2016"
f17 = FormateaFecha("m"; f1);       %% f17 valdrá "4"
f18 = FormateaFecha("mm"; f1);      %% f18 valdrá "04"
f19 = FormateaFecha("mmm"; f1);     %% f19 valdrá "abr"
f20 = FormateaFecha("mmmm"; f1);    %% f20 valdrá "abril"
f21 = FormateaFecha("yy"; f1);      %% f21 valdrá "16"
f22 = FormateaFecha("yyyy"; f1);    %% f22 valdrá "2016"
f23 = FormateaFecha("dd/mm/yyyy"; f1);  %% f23 valdrá "15/04/2016"
f24 = FormateaFecha("dddd, d 'de' mmmm 'de' yyyy"; f1);   %% f24 valdrá viernes, 15 de abril de 2016
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.24. FormateaFechaHora

Descripción:

Convierte la fecha y hora o la hora <t2> al formato especificado por <t1>
Formatos máscara para horas (también soporta todas las de fechas):
h: Muestra la hora del día como un número de uno o dos dígitos (0-23).
hh: Muestra la hora del día como un número de dos dígitos (00-23).
n: Muestra los minutos como un número de uno o dos dígitos (0-59).
nn: Muestra los minutos como un número de dos dígitos (00-59).
s: Muestra los segundos como un número de uno dos dígitos (0-59).
ss: Muestra los segundos como un número de dos dígitos (00-59).
.zzz: Muestra los milisegundos
t: Muestra la hora con formato "hh:nn".
tt: Muestra la hora con el formato definido en Windows (normalmente "h:nn:ss").
am/pm: Usa formato de 12 horas con el especificador h o hh precedente, y muestra "am" o "pm" según corresponda (saldrá en mayúsculas o minúsculas según esté "AM" o "am").
a/p: Usa formato de 12 horas con el especificador h o hh precedente, y muestra "a" o "p" según corresponda (saldrá en mayúsculas o minúsculas según esté el especificador).
:: Muestra el separador de hora configurado en Windows. Normalmente es ":".
'xx' ó "xx": Los caracteres encerrados por comillas simples o dobles se muestran tal como se indican y no afectan al formato.

Sintaxis: FormateaFechaHora(<t1>; <t2>)

Tipo de dato devuelto: Texto

Ejemplo:

```
f1 = "15/4/16 13:9:39";
f10 = FormateaFechaHora(f1);            %% f10 valdrá "15/04/2016 13:09"
f11 = FormateaFechaHora("h"; f1);       %% f11 valdrá "13"
f12 = FormateaFechaHora("hh"; f1);      %% f12 valdrá "13"
f13 = FormateaFechaHora("n"; f1);       %% f13 valdrá "9"
f14 = FormateaFechaHora("nn"; f1);      %% f14 valdrá "09"
f15 = FormateaFechaHora("s"; f1);       %% f15 valdrá "39"
f16 = FormateaFechaHora("ss"; f1);      %% f16 valdrá "39"
f17 = FormateaFechaHora("t"; f1);       %% f17 valdrá "13:09"
f18 = FormateaFechaHora("tt"; f1);      %% f18 valdrá "13:09:39"
f19 = FormateaFechaHora("hh:nn AM/PM"; f1); %% f19 valdrá "01:09 PM"
f20 = FormateaFechaHora("hh:nn a/p"; f1);   %% f20 valdrá "01:09 p"
f23 = FormateaFechaHora("'H'ora hh:nn"; f1);%% f23 valdrá "Hora 13:09"
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

12.25. HoraActual

Descripción:
Se devuelve la hora actual con el formato indicado por <t>. Si no se especifica <t> se usa "HH:MM:SS".
El formato es el mismo que para los campos de hora en la función FormateaFechaHora.

Sintaxis: HoraActual[(<t>)]

Tipo de dato devuelto: Texto

Ejemplo:
t = HoraActual; %% t valdrá la hora actual en formato "HH:MM:SS"

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de fechas

13. FUNCIONES DE MANIPULACIÓN DE VARIABLES

13.1. EditarVars

Descripción:

Muestra un cuadro de diálogo con el título <t1> que permite ver el nombre y valor de la lista de variables indicada en <t2>. Pueden modificarse sus valores.
Para indicar más de una variable se separan por ";" dentro de una cadena de texto sin espacios. Puede usarse el carácter "" como comodín de cualquier carácter al final del nombre de la variable.*
Si no se especifica <t2> se muestran todas las variables. Si se intenta usar una variable que no existe se produce un error.
Con esta instrucción puede tenerse una única ventana para pedirle varios valores al usuario.

Sintaxis: EditarVars(<t1>[;<t2>])

Tipo de dato devuelto: Nada

Ejemplo:

```
Cliente = "";
Desde_fecha   = "";
Hasta_fecha   = "";
EditarVars("Introduzca los valores"; "cliente;desde_fecha;hasta_fecha");

%% El nombre de las variables no puede contener un ;
&"Desde número" = 10;
&"Hasta número" = 30;
mensaje(@"Desde número");
EditarVars("Introduzca el rango"; "Desde número;Hasta número");
desde_número = @'Desde número';
hasta_número = @'Hasta número';
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de variables

13.2. DestruirVariable

Descripción:
Elimina la variable <t> de forma que no pueda seguir usándose.
Si la variable no existe se produce un error.

Sintaxis: DestruirVariable(<t>)

Tipo de dato devuelto: Lógico

Ejemplo:

```
si ExisteVariable("desde") entonces {
  DestruirVariable("desde");
};
i = 1;
lista = #("a"; "b"; "c"; "d"; "e"; "f");
mientras {i <= LTamaño(lista)} haz {
  si No ExisteVariable("desde") entonces {
    desde = Pregunta("Introduzca desde que valor se procesa la lista");
  };
  si (lista[i] >= desde) entonces {
    Mensaje("Procesando: " + lista[i]);
  };
  i = i + 1;
};
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de variables

13.3. GetVariablePrefix

Descripción:
Devuelve la cadena de texto que se añade al principio del nombre de cada nueva variable.

Sintaxis: GetVariablePrefix

Tipo de dato devuelto: Texto

Ejemplo:
```
A = "texto";                %% Se crea la variable A
SetVariablePrefix("Prefijo");   %% Se fija el prefijo para todas las nuevas variables
A = "texto2";               %% Se crea la variable PrefijoA
X = GetVariablePrefix;          %% Se crea la variable PrefijoX con el contenido "Prefijo"
SetVariablePrefix("");          %% Se elimina el prefijo para las nuevas variables
B = "texto3";               %% Se crea la variable B
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de variables

13.4. SetVariablePrefix

Descripción:

Especifica una cadena que se usará como prefijo para el nombre de todas las variables que se creen desde ese momento hasta que se cierra ese evaluador. El nombre de la variable será el que hubiéramos puesto en su definición más el prefijo.

Sintaxis: SetVariablePrefix(<t>)

Tipo de dato devuelto: Nada

Ejemplo:

```
A = "texto";                %% Se crea la variable A
SetVariablePrefix("Prefijo");   %% Se fija el prefijo para todas las nuevas variables
A = "texto2";               %% Se crea la variable PrefijoA
X = GetVariablePrefix;          %% Se crea la variable PrefijoX con el contenido "Prefijo"
SetVariablePrefix("");          %% Se elimina el prefijo para las nuevas variables
B = "texto3";               %% Se crea la variable B
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de variables

13.5. VerVars

Descripción:

Muestra un cuadro de diálogo con el título <t1> que permite ver el nombre y valor de la lista de variables indicada en <t2>. Para indicar más de una variable se separan por ";". Puede usarse el carácter "" como comodín de cualquier carácter al final del nombre de la variable.*
Si no se especifica <t2> se muestran todas las variables. Si se intenta usar una variable que no existe se produce un error.
Para poder inspeccionar las variables cuando se está depurando un script es más cómodo y potente utilizar la instrucción "depurador".
No se puede modificar el valor de las variables. Si puede hacerse con la instrucción "EditarVars".

Sintaxis: VerVars(<t1>[;<t2>])

Tipo de dato devuelto: Nada

Ejemplo:

```
a = "texto";
b = 55;
fecha1 = "15/12/2015";
fecha2 = "5/03/2016";
fecha3 = "22/05/2017";
VerVars("Variables utilizadas"; "a;fecha*");
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de variables

13.6. Tipo

Descripción:
Devuelve el tipo de la variable <t>.
Valores posibles:
TNada: Nulo o vacío.
TNúmero: Un número de coma flotante de doble precisión.
TTexto: Una cadena de texto de cualquier longitud.
TLógico: Los valores "cierto" o "falso".
TLista: Una lista (array) de valores.
TRegistro: Un conjunto de campos.
TBloque: Un bloque de código.
TTipo: Un tipo de dato.

Sintaxis: Tipo(<t>)

Tipo de dato devuelto: Tipo

Ejemplo:
```
v1 = 1;
v2 = "Cadena de texto";
v3 = falso;
v4 = #("a"; 1; "b"; 2; "c");
v5 = #[Nombre: "Sandra" Edad: 21];
v6 = nada;
v7 = {params[1] * params[2]/100};
v8 = v1;
v9 = TNumero;
tv1 = Tipo(v1); %% tv1 valdrá TNúmero
tv2 = Tipo(v2); %% tv2 valdrá TTexto
tv3 = Tipo(v3); %% tv3 valdrá TLógico
tv4 = Tipo(v4); %% tv4 valdrá TLista
tv5 = Tipo(v5); %% tv5 valdrá TRegistro
tv6 = Tipo(v6); %% tv6 valdrá TNada
tv7 = Tipo(v7); %% tv7 valdrá TBloque
tv8 = Tipo(v8); %% tv8 valdrá TNúmero
tv9 = Tipo(v9); %% tv9 valdrá TTipo
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de variables

13.7. BorrarVars

Descripción:

Elimina la lista de variables indicada por <t>.
Para indicar más de una variable se separan por ";". Puede usarse el carácter "" como comodín de cualquier carácter al final del nombre de la variable. Si se intenta usar una variable que no existe se produce un error.*

Sintaxis: BorrarVars(<t>)

Tipo de dato devuelto: Nada

Ejemplo:

```
a = "texto";
b = 25;
c = cierto;
fecha1 = "12/01/2020";
fecha2 = "16/01/2020";
fecha3 = "18/02/2022";
BorrarVars("a;c;fech*");
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de variables

13.8. EsNada

Descripción:
Devuelve cierto si <t> vale "nada", en otro caso devuelve falso.
Es igual a utilizar "Tipo(<t>) == TNada"

Sintaxis: EsNada(<t>)

Tipo de dato devuelto: Lógico

Ejemplo:
```
si (No EsNada(v1) y No EsNada(v3)) entonces {
  v2 = v1 + v3;
} sino {
  v2 = 0;
};
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de variables

13.9. DuplicarVars

Descripción:

Duplica la lista de variables indicada por <t1> añadiendo el prefijo <t2> al nombre de cada variable.
Para indicar más de una variable se separan por ";". Puede usarse el carácter "" como comodín de cualquier carácter al final del nombre de la variable. Si se intenta usar una variable que no existe se produce un error.*
Las nuevas variables tienen el mismo contenido que las antiguas.

Sintaxis: DuplicarVars(<t1>; <t2>)

Tipo de dato devuelto: Nada

Ejemplo:

```
a = "texto";
b = 25;
c = cierto;
fecha1 = "12/01/2020";
fecha2 = "16/01/2020";
fecha3 = "18/02/2022";
DuplicarVars("a;c;fec*"; "nueva_");
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de variables

13.10. ExisteVariable

Descripción:
Devuelve cierto si la variable <t> existe. Debe pasarse el nombre de la variable entre comillas, simples o dobles. Con esta instrucción puede controlarse que determinados procesos se ejecuten sólo una vez.

Sintaxis: ExisteVariable(<t>)

Tipo de dato devuelto: Logico

Ejemplo:

```
i = 1;
lista = #("a"; "b"; "c"; "d"; "e"; "f");
mientras {i <= LTamaño(lista)} haz {
 si No ExisteVariable("desde") entonces {
  desde = Pregunta("Introduzca desde que valor se procesa la lista");
 };
 si (lista[i] >= desde) entonces {
  Mensaje("Procesando: " + lista[i]);
 };
 i = i + 1;
};
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de manipulación de variables

14. FUNCIONES DE ACCESO A BASE DE DATOS

14.1. ObtListaSQL

Descripción:
Ejecuta la sentencia SQL indicada por <t>, que debe ser un select que recupere una única columna, y devuelve una lista que contiene un elemento de tipo texto por cada uno de los registros recuperados.
<alias> indica el nombre de la conexión creada con "ConectaSQL" o "ConectaBDE" a utilizar.

Sintaxis: ObtListaSQL(<t>[;<alias>])

Tipo de dato devuelto: Lista

Ejemplo:
lista = ObtListaSQL("select codigo from cliente where nif is null");
%% lista valdrá, por ejemplo, #('1'; '4'; '5'; '100'; '6000')

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de acceso a base de datos

14.2. Desconecta

Descripción:

Cierra la conexión con la base de datos externa <alias>. Es indiferente que se hubiera lanzado con cualquiera de las instrucciones "ConectaXXX".
Si no se cierra la conexión al finalizar el script esta quedará abierta, bloqueando el, o los ficheros, que se hayan utilizado, hasta que se finalice ese evaluador, generalmente esto se produce al cerrar la ventana desde la que se lanzó el script.

Sintaxis: Desconecta(<alias>)

Tipo de dato devuelto: Nada

Ejemplo:

```
ConectaSQL("Empresa2"; "C:\Distrito\PymeCS\Database\Ejemplo\2006.FDB");
lista_registros = ConsultaLista("select codigo, nombrecomercial from cliente where nif is null"; "Empresa2");
LAplica(lista_registros; {
  mensaje("Cliente: " + Formatea("0"; lista_registros[params[1]].Codigo)
    + " " + lista_registros[params[1]].NombreComercial)
});
Desconecta("Empresa2");
%
  Se mostrará una ventana con el código y nombre de cada cliente que tenga el nif vacío
  en la tabla "cliente" de la base de datos indicada
%
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de acceso a base de datos

14.3. EjecutarSQL

Descripción:

Ejecuta la sentencia sql <t>. <t> Debe ser una sentencia sql que no devuelva ningún registro (insert, update, delete, …).
<alias> indica el nombre de la conexión creada con "ConectaSQL" o "ConectaBDE" a utilizar.
<lista_valores> puede ser una lista o un registro. En el primer caso se sustituirá en la sql cada parámetro, se indican con ":" y el nombre del parámetro a continuación, con el valor correspondiente de la lista según su posición en la sql, el primer parámetro que aparezca en la sql se sustituirá con el primer valor de la lista, segundo parámetro -> segundo valor, etc. No deben usarse nombres repetidos de parámetros en la sentencia sql porque no puede garantizarse su comportamiento con la lista, con el registro no hay problema. En el segundo caso se sustituirá cada parámetro de la sql por el campo del registro que tenga el mismo nombre, independientemente de su posición. Todos los campos que se pasen en el registro deben usarse como variables en la sql o se produce un error. Llamar a una sentencia sql con parámetros es más rápido que tener los valores de esos parámetros dentro de la sentencia sql, por ejemplo "select nombre comercial from cliente where codigo = 3". La diferencia de rendimiento es más grande cuanto más compleja sea la sentencia sql y cuantas más veces se ejecute.

Se devuelve el número de registros afectados por la sentencia sql, igual que se muestran en "Herramientas de administrador".

Sintaxis: EjecutarSQL(<t>[;<lista_valores>][;<alias>])

Tipo de dato devuelto: Número

Ejemplo:

```
EjecutarSQL("update cliente set nif = '99999999R' where nif is null");
%% Se pondrá el nif "99.999.999-R" a todos los clientes que lo tengan vacío

EjecutarSQL("update cliente set nif = :nif where nif is null"; #[nif: '99999999R']);
%% Consulta mejorada
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de acceso a base de datos

14.4. ConectaSQL

Descripción:

Permite lanzar una conexión contra una base de datos en formato Firebird o InterBase (hasta versión 6.X).
<fichero> es el fichero incluyendo la ruta de la base de datos y el nombre del servidor si es necesario.
<alias> es el nombre que se utilizará en las instrucciones de acceso a datos para usar esa conexión en lugar de la empresa abierta en el programa.

Sintaxis: ConectaSQL(<alias>; <fichero>)

Tipo de dato devuelto: Nada

Ejemplo:

```
ConectaSQL("Empresa2"; "C:\Distrito\PymeCS\Database\Ejemplo\2006.FDB");
%% Si se realiza la conexión desde la red sería así:
%% ConectaSQL("Empresa2"; "servidor:C:\Distrito\PymeCS\Database\Ejemplo\2006.FDB");
lista_registros = ConsultaLista("select codigo, nombrecomercial from cliente where nif is null"; "Empresa2");
LAplica(lista_registros; {
  mensaje("Cliente: " + Formatea("0"; lista_registros[params[1]].Codigo)
    + " " + lista_registros[params[1]].NombreComercial)
});
Desconecta("Empresa2");
%
  Se mostrará una ventana con el código y nombre de cada cliente que tenga el nif vacío
  en la tabla "cliente" de la base de datos indicada
%
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de acceso a base de datos

14.5. ConectaBDE

Descripción:

Permite lanzar una conexión contra una base de datos en formato Paradox o DBase III.
<ruta_carpeta> es la ruta de la carpeta donde se encuentran los ficheros de datos.
<alias> es el nombre que se utilizará en las instrucciones de acceso a datos para usar esa conexión en lugar de la empresa abierta en el programa.
En Paradox y DBase III todas las tablas de una base de datos se encuentran en un mismo directorio, la instrucción de acceso a datos buscará un fichero con el nombre de la tabla indicada en la sentencia sql.

Sintaxis: ConectaBDE(<alias>; <ruta_carpeta>)

Tipo de dato devuelto: Nada

Ejemplo:

```
ConectaBDE("BDD_Antigua"; "C:\Distrito\Pyme\Database\Ejemplo\2006");
lista_registros = ConsultaLista("select codigo, nombrecomercial from cliente where nif is null"; "BDE_Antigua");
LAplica(lista_registros; {
  mensaje("Cliente: " + Formatea("0"; lista_registros[params[1]].Codigo)
    + " " + lista_registros[params[1]].NombreComercial)
});
Desconecta("BDD_Antigua");
%
  Se mostrará una ventana con el código y nombre de cada cliente que tenga el nif vacío
  en la tabla "cliente" del directorio indicado
%
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de acceso a base de datos

14.6. ConsultaCampo

Descripción:

Devuelve el valor del campo que devuelva la sentencia sql <t>. El tipo de dato devuelto será el del campo devuelto. Si la sentencia sql devuelve más de un campo la instrucción sólo devolverá el primero.
<alias> indica el nombre de la conexión creada con "ConectaSQL" o "ConectaBDE" a utilizar.
<lista_valores> puede ser una lista o un registro. En el primer caso se sustituirá en la sql cada parámetro, se indican con ":" y el nombre del parámetro a continuación, con el valor correspondiente de la lista según su posición en la sql, el primer parámetro que aparezca en la sql se sustituirá con el primer valor de la lista, segundo parámetro -> segundo valor, etc. No deben usarse nombres repetidos de parámetros en la sentencia sql porque no puede garantizarse su comportamiento con la lista, con el registro no hay problema. En el segundo caso se sustituirá cada parámetro de la sql por el campo del registro que tenga el mismo nombre, independientemente de su posición. Todos los campos que se pasen en el registro deben usarse como variables en la sql o se produce un error. Llamar a una sentencia sql con parámetros es más rápido que tener los valores de esos parámetros dentro de la sentencia sql, por ejemplo "select nombre comercial from cliente where codigo = 3". La diferencia de rendimiento es más grande cuanto más compleja sea la sentencia sql y cuantas más veces se ejecute.

Sintaxis: ConsultaCampo(<t>[;<lista_valores>][;<alias>])

Tipo de dato devuelto: Variable

Ejemplo:

```
código_cliente = 5;
x = ConsultaCampo("select nif from cliente where codigo = " + Formatea("0"; código_cliente));
%% x valdrá lo que contenga el campo nif del cliente 5

x1 = ConsultaCampo("select nif from cliente where codigo = :codigo_cliente"; #(código_cliente));
%% Consulta mejorada
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de acceso a base de datos

14.7. ConsultaLista

Descripción:

Devuelve una lista de registros con los campos, valores y nombre, que devuelva la sentencia sql <t>. El tipo de dato devuelto de cada campo será el del campo devuelto.
<alias> indica el nombre de la conexión creada con "ConectaSQL" o "ConectaBDE" a utilizar.
<lista_valores> puede ser una lista o un registro. En el primer caso se sustituirá en la sql cada parámetro, se indican con ":" y el nombre del parámetro a continuación, con el valor correspondiente de la lista según su posición en la sql, el primer parámetro que aparezca en la sql se sustituirá con el primer valor de la lista, segundo parámetro -> segundo valor, etc. No deben usarse nombres repetidos de parámetros en la sentencia sql porque no puede garantizarse su comportamiento con la lista, con el registro no hay problema. En el segundo caso se sustituirá cada parámetro de la sql por el campo del registro que tenga el mismo nombre, independientemente de su posición. Todos los campos que se pasen en el registro deben usarse como variables en la sql o se produce un error. Llamar a una sentencia sql con parámetros es más rápido que tener los valores de esos parámetros dentro de la sentencia sql, por ejemplo "select nombre comercial from cliente where codigo = 3". La diferencia de rendimiento es más grande cuanto más compleja sea la sentencia sql y cuantas más veces se ejecute.

Sintaxis: ConsultaLista(<t>[;<lista_valores>][;<alias>])

Tipo de dato devuelto: Lista

Ejemplo:

```
lista_registros = ConsultaLista("select codigo, nombrecomercial from cliente where nif = '99999999R' ");
LAplica(lista_registros; {
  mensaje("Cliente: " + Formatea("0"; lista_registros[params[1]].Codigo)
    + " " + lista_registros[params[1]].NombreComercial)
});
%% Se mostrará una ventana con el código y nombre de cada cliente que tenga el nif vacío

lista_registros1 = ConsultaLista("select codigo, nombrecomercial from cliente where nif = :nif"; #('99999999R'));
%% Consulta mejorada
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de acceso a base de datos

14.8. ConsultaCuantos

Descripción:

Indica cuantos registros (el número) devolverá la consulta <t>.
<alias> indica el nombre de la conexión creada con "ConectaSQL" o "ConectaBDE" a utilizar.
<lista_valores> puede ser una lista o un registro. En el primer caso se sustituirá en la sql cada parámetro, se indican con ":" y el nombre del parámetro a continuación, con el valor correspondiente de la lista según su posición en la sql, el primer parámetro que aparezca en la sql se sustituirá con el primer valor de la lista, segundo parámetro -> segundo valor, etc. No deben usarse nombres repetidos de parámetros en la sentencia sql porque no puede garantizarse su comportamiento con la lista, con el registro no hay problema. En el segundo caso se sustituirá cada parámetro de la sql por el campo del registro que tenga el mismo nombre, independientemente de su posición. Todos los campos que se pasen en el registro deben usarse como variables en la sql o se produce un error. Llamar a una sentencia sql con parámetros es más rápido que tener los valores de esos parámetros dentro de la sentencia sql, por ejemplo "select nombre comercial from cliente where codigo = 3". La diferencia de rendimiento es más grande cuanto más compleja sea la sentencia sql y cuantas más veces se ejecute.

Sintaxis: ConsultaCuantos(<t>[;<lista_valores>][;<alias])

Tipo de dato devuelto: Número

Ejemplo:

```
v = ConsultaCuantos("select codigo from cliente where nif is null");
%% v será el número de clientes que tienen el nif nulo

v1 = ConsultaCuantos("select codigo from cliente where nif = :nif"; #('99999999R'));
%% Consulta mejorada
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de acceso a base de datos

14.9. ConsultaRegistro

Descripción:

Devuelve un registro con los campos, valores y nombre, que devuelva la sentencia sql <t>. El tipo de dato devuelto de cada campo será el del campo devuelto.
Si la sentencia sql devuelve más de un registro la instrucción sólo devolverá el primero.
<lista_valores> puede ser una lista o un registro. En el primer caso se sustituirá en la sql cada parámetro, se indican con ":" y el nombre del parámetro a continuación, con el valor correspondiente de la lista según su posición en la sql, el primer parámetro que aparezca en la sql se sustituirá con el primer valor de la lista, segundo parámetro -> segundo valor, etc. No deben usarse nombres repetidos de parámetros en la sentencia sql porque no puede garantizarse su comportamiento con la lista, con el registro no hay problema. En el segundo caso se sustituirá
cada parámetro de la sql por el campo del registro que tenga el mismo nombre, independientemente de su posición. Todos los campos que se pasen en el registro deben usarse como variables en la sql o se produce un error. Llamar a una sentencia sql con parámetros es más rápido que tener los valores de esos parámetros dentro de la sentencia sql, por ejemplo "select nombre comercial from cliente where codigo = 3". La diferencia de rendimiento es más grande cuanto más compleja sea la sentencia sql y cuantas más veces se ejecute.

Sintaxis: ConsultaRegistro(<t>[;<lista_valores>][;<alias>])

Tipo de dato devuelto: Registro

Ejemplo:

```
x = ConsultaRegistro("select nif, nombrecomercial from cliente where nif = '99999999R'");
%
  x.NIF valdrá lo que contenga el campo nif del primer cliente devuelto por la sentencia sql
  x.NombreComercial valdrá lo que contenga el campo NombreComercial
    del primer cliente devuelto por la sentencia sql
%
x1 = ConsultaRegistro("select nif, nombrecomercial from cliente where nif = :nif"; #('99999999R'));
%% Consulta mejorada
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de acceso a base de datos

14.10. CuantosRegistros

Descripción:
Devuelve el número de registros de la tabla <t1> que cumplen la condición <t2>.
<t2> debe tener la misma sintaxis que un "where" de una sentencia SQL sin la palabra "where".
<alias> indica el nombre de la conexión creada con "ConectaSQL" o "ConectaBDE" a utilizar.

Sintaxis: CuantosRegistros(<t1>; <t2>[;<alias>])

Tipo de dato devuelto: Número

Ejemplo:
v = CuantosRegistros("cliente"; "nif is null");
%% v será el número de clientes que tienen el nif nulo

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones de acceso a base de datos

15. FUNCIONES VARIAS

15.1. Depurador

Descripción:

Cuando está activado el "debug" (Ctrl + Alt + / (del teclado numérico)) permite detener la ejecución de un script e inspeccionar las variables de memoria correspondientes a ese script.
Cuando se usa en un formato de impresión también pueden verse todas las variables disponibles en el formato.
En la ventana que se abre se verá el tipo de cada variable, y dándole doble clic sobre su valor se abrirá una nueva ventana para ver el contenido completo de la misma.
Si se pulsa "Interrumpir" se finalizará la ejecución del script. Si se pulsa "Depurar" podremos modificar la parte del script que se está visualizando, aunque no se guardan los cambios aquí realizados, de esta manera podremos probar modificaciones sobre el script sin modificar.

Sintaxis: Depurador

Tipo de dato devuelto: Nada

Ejemplo:

```
largo = 5;
ancho = 2;
fórmula = "largo * ancho";
resultado = Evalúa(fórmula);
mensaje(resultado);
depurador;
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.2. Mensaje

Descripción:
Muestra un cuadro de diálgo informativo con el texto <t> y un botón "Aceptar".
Pueden usarse diferentes líneas, $n para cambiar de línea, o tabuladores, $t para añadirlos, dentro del texto de mensaje <t>.

Sintaxis: Mensaje(<t>)

Tipo de dato devuelto: Nada

Ejemplo:
```
número_clientes = ConsultaCampo("select count(*) from cliente");
Mensaje("Existen " + Formatea(",0"; número_clientes) + " clientes en la base de datos");
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.3. ResultadoNada

Descripción:

Evalúa el bloque de código <b> y descarta el valor del resultado devolviendo "Nada".
Es necesario utilizarlo en los formatos de impresión cuando se tienen bloques de código de más de una línea, por ejemplo, dentro de un "si" existe más de una instrucción en la misma rama. Esto es porque los formatos de impresión se evalúan línea a línea. En el resto de lugares donde se usan los scripts no es necesario.

Sintaxis: ResultadoNada(<b>)

Tipo de dato devuelto: Nada

Ejemplo:

```
ResultadoNada({
 txt = "";
 si (TxIzquierda(ArtículoCódigo; 3) == "001") entonces {
  pvr = 0;  %% Inicialización variable
  pvr = ArtículoPrecio + (ArtículoPrecio * 0,30);
  si (pvr < 100) entonces {
   pvr = 100;
  };
  txt = Formatea(",0.00"; pvr);
  si (TxIzquierda(ClienteCódigoPostal; 2)  == "15" y TxTamaño(ClienteCódigoPostal) == 5) entonces {
   txt = txt + " - 10%";
  };
 };
});
txt;
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.4. SeleccionarTabla

Descripción:

SeleccionarTabla("tabla" [; "filtro" [; #(columnas) [; "campo" [; "orden" [; "título" [; #(titulosColumnas)]]]]]])
SeleccionarTabla("tabla" [; parametros])

Muestra una ventana de selección con los registros de la tabla "tabla" que cumplan la condición "filtro" (igual que las condiciones de una sentecia sql sin la palabra "where").
Sólo se muestran los campos indicados en "columnas" (deben introducirse los nombres entre comillas y separados por punto y coma).
Se devolverá el valor de "campo" del registro seleccionado (será del tipo que sea el campo).
"Orden" es el nombre del campo por el cual se ordenarán los registros al mostrarlos, pueden ser varios separados por punto y coma, p.ej.
'serie;numero' (por defecto se utilizará el campo a retornar)
"Título" es el título de la ventana.
TitulosColumnas es una lista con los títulos de las columnas. Si no se envía se usará el nombre del campo.

Parametros es un registro que permite especificar diversas opciones: #[opcion: valor; ...]
Filtro (texto): fragmento de una condición que se utilizará para filtrar la tabla (p.ej. como where del select)
Columnas (lista): lista con los nombres de campos de la tabla que se mostrarán: #('campo'; ...)
Campo (texto): nombre del campo cuyo valor se retornará (por defecto el primero del select)
Orden (texto): Nombre del campo por el cual se ordenarán los registros al mostrarlos, pueden ser varios separados por punto y coma, p.ej.
'serie;numero' (por defecto se utilizará el campo a retornar)
Título (texto): título para la ventana de selección
TítulosColumnas (lista): lista con los títulos para las columnas de la tabla que se mostrarán: #('título'; ...)
Conexión (texto): el nombre de una conexión creada con ConectaSQL o ConectaBDE (por ahora no de ConectaADO porque la ventana de selección no lo soporta)

Retorna
Si el usuario acepta la ventana: el valor del campo especificado
Si el usuario cancela la ventana: nada

Sintaxis: SeleccionarTabla("tabla" [; "filtro" [; #(columnas) [; "campo" [; "orden" [; "título" [; #(titulosColumnas)]]]]]]) /

Tipo de dato devuelto: Variable

Ejemplo:

```
%% Ejemplo 1
SeleccionarTabla( "cliente"; "codigo > 10 and codigo < 100";
   #("codigo"; "nombrecomercial"; "codagente") ;
   "nombrecomercial"  ; "" ;"Seleccione un cliente");

%% Ejemplo 2
SeleccionarTabla(
  'doccab';
  #[
    filtro: 'tipo=2'
    columnas: #('tipo'; 'serie'; 'numero'; 'fecha'; 'codcliente'; 'importetotal'; 'codigo')
    campo: 'codigo'
    valor: 82
    orden: 'serie;numero'
    título: 'Buscar'
    títulosColumnas: #('Tipo'; 'Serie'; 'Número'; 'Fecha'; 'Cód. cliente'; 'Total'; 'Código')
  ]
);
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.5. Clona

Descripción:
Devuelve una copia del objeto igual al pasado. <o> puede ser una variable, una lista, un registro, etc. Es necesario utilizarlo cuando de se trata copiar tipos de datos complejos: listas y registros.

Sintaxis: Clona(<o>)

Tipo de dato devuelto: Variable

Ejemplo:

```
%% Obtención listas con títulos filas y columnas
ListaFilas = #("Color1"; "Color2" ; "Color3"; "Color4");
ListaColumnas = #("Talla1"; "Talla2"; "Talla3");
Tabla = #();
%% Creación de tabla con todos los valores a 0
%% Creación de los valores para cada columna en una fila
ListaValoresFila = #();
LAplica(ListaColumnas; {
  LAñade(ListaValoresFila; 0);
});
%% Creación de las filas
LAplica(ListaFilas; {
  %% Hay que clonar ListaValoresFila porque sino se introduce un objeto
  %%   y al modificar en una fila se modifican todas
  LAñade(Tabla; Clona(ListaValoresFila));
});
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.6. AnteriorControl

Descripción:

Mueve el foco del campo que actualmente lo tiene hasta el que esté <n> posiciones antes.
Sólo se cuentan aquellos campos que tienen "Hacer parada" con valor "Sí".
Es útil para volver a un campo tras detectar un error en él. No sirve para evitar la salida del campo, para conseguir una funcionalidad parecida debe hacerse en los scripts de cabecera, en antes de confirmar, usando la instrucción "Error".

Sintaxis: AnteriorControl(<n>)

Tipo de dato devuelto: Nada

Ejemplo:

```
cliente = Leer("DOCCAB"; "Actual.CodCliente"; "");
si (cliente == 4) entonces {
  Mensaje("Este cliente no admite documentos actualmente");
  AnteriorControl(1);   %% Así al salir del campo volvemos a él.
};
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.7. Inspeccionar

Descripción:
Muestra una ventana con el contenido de la variable. Es útil para ver variables muy grandes. En la mayoría de los casos es más cómodo utilizar la función "Depurador".

Sintaxis: Inspeccionar(<variable>)

Tipo de dato devuelto: Nada

Ejemplo:
```
b = ConsultaLista("select codigo, nombrecomercial from cliente where nif = '99999999R'");
Inspeccionar(b);
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.8. Confirma

Descripción:
Muestra un cuadro de diálogo con el texto <t> y los botones "Sí", "No" y "Cancelar".
Si se selecciona "Sí" se devuelve "cierto", si se selecciona "No" se devuelve "falso". Si se pulsa en "Cancelar" se interrumpe el script.
Pueden usarse diferentes líneas, "$n" o intro para cambiar de línea, o tabuladores, "$t" para añadirlos, dentro del texto de mensaje <t>.

Sintaxis: Confirma(<t>)

Tipo de dato devuelto: Lógico

Ejemplo:
```
lista_códigos_cliente = #();
haz {
 código_cliente = Pregunta("Introduzca el código de cliente para obtener su NIF");
 LAñade(lista_códigos_cliente; código_cliente);
} mientras {Confirma("¿Desea introducir otro cliente?")}
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.9. Error

Descripción:
Muestra un cuadro de diálogo de error con el texto <t> y un botón "Aceptar". Cuando se muestra se finaliza el script.

Sintaxis: Error(<t>)

Tipo de dato devuelto: Nada

Ejemplo:
```
x = Pregunta("¿Introduzca un número entre 1 y 15?");
si (EnNúmero(x;0) > 15 o EnNúmero(x;-1) < 1) entonces {
  Error("No se ha introducido un número válido");
} sino {
  Mensaje("Se ha introducido un número válido");
};
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.10. Pregunta

Descripción:

Muestra un cuadro de diálogo solicitando un texto con el mensaje <t1>.

Si se especifica <t2> se mostrará como valor por defecto.

Pueden usarse diferentes líneas, $n para cambiar de línea, o tabuladores, $t para añadirlos, dentro del texto de mensaje <t>.

Si se indica "cierto" en el tercer parámetro (<l>), al pulsar intro no se pasará el cursor al botón aceptar, directamente cerrará la ventana y finalizará la instrucción. Si no se indica o se pasa "falso", habrá que pulsar intro dos veces para finalizar la instrucción.

Si se indica cierto en el cuarto parámetro <l1> se ocultan los caracteres de manera que no se vean al introducirlos (enmascarado - confidencial).

Aparecerán los botones "Aceptar" y "Cancelar". Si se selecciona "Aceptar" se devuelve el texto introducido, o nada si el usuario aceptó dejando el valor vacío, si se presiona "Cancelar" se interrumpe el script.

Sintaxis: Pregunta(<t1>[;<t2>][;<l>][;<l1>])

Tipo de dato devuelto: Texto

Ejemplo:

```
código_cliente = Pregunta("Introduzca el código de cliente para obtener su NIF");
nif_cliente    = ConsultaCampo("select nif from cliente where codigo = " + código_cliente);
Mensaje("El nif del cliente " + código_cliente + " es " + nif_cliente);

%%Pregunta contraseña
Contraseña=Pregunta("Indica contraseña";"";cierto;cierto);
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.11. Evalúa

Descripción:
Ejecuta el código indicado por <t> y devuelve su resultado.
Permite fórmulas en variables o en campos de la base de datos. Por ejemplo el campo "fórmula de cálculo de cantidades".

Sintaxis: Evalúa(<t>)

Tipo de dato devuelto: Variable

Ejemplo:
```
largo = 5;
ancho = 2;
fórmula = "largo * ancho";
resultado = Evalúa(fórmula);
mensaje(resultado);
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.12. PararImpresión

Descripción:

Anula la impresión del formato actual.
Puede usarse cuando en base a una condición se imprime otro formato y no el actual.
El campo fórmula donde se use tiene que ser obligatoriamente el primer campo del formato si queremos evitar la impresión.
Sólo funciona en los formatos de impresión de documentos (compras, ventas, recuentos y movimientos), recibos y orden de fabricación y reparación.

Sintaxis: PararImpresión

Tipo de dato devuelto: Nada

Ejemplo:

```
si (ClienteDatoAdicionalValor("FormatoEspecial") == "SI") entonces {
  Imprimir("\\Servidor\Distrito\Pyme\FormatoEspecial.FMT");
  PararImpresión;
};
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: SI

Tipo de instrucción: Funciones varias

15.13. Imprimir

Descripción:

Imprime el formato de impresión <t> desde otro. <t> debe incluir la ruta completa al fichero de formato, si estamos en una instalación en red lo normal es que sea la ruta desde la red. El formato a imprimir debe ser del mismo tipo que el que se está imprimiendo. No es relevante la posición del campo "Fórmula" que tiene "Imprimir". Primero se imprimirá el formato que contiene "Imprimir" y a continuación este. Pueden tenerse tantos "Imprimir" dentro de un formato como se quieran.
Si se desea tener un formato que imprima varias hojas debe hacerse con "Imprimir".
Si se quiere en función de una condición imprimir un formato u otro deberá usarse "PararImpresión" después de "Imprimir", en este caso debe ser el primer campo del documento.
Sólo funciona en los formatos de impresión de documentos (compras, ventas, recuentos y movimientos), recibos y orden de fabricación y reparación.
Opcionalmente se puede pasar un registro con variables que estarán disponibles en el nuevo formato a imprimir si estamos imprimiendo un formato de documentos, en el resto de formatos no está soportado.
Las fuentes que se utilizan siempre son las del primer formato, da igual las que tengan los siguientes.
Si los formatos tienen asignadas impresoras diferentes se utilizarán las de cada formato, da igual que se previsualice o no.

Sintaxis: Imprimir(<t>; [registro])

Tipo de dato devuelto: Nada

Ejemplo:

```
Imprimir("\\Servidor\Distrito\Pyme\ContratoVentas.FMT");
Imprimir("\\Servidor\Distrito\Pyme\ContratoVentas.FMT";
 #[variable1: 12 variable2: 'valor']);
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: SI

Tipo de instrucción: Funciones varias

15.14. SiguienteControl

Descripción:

Mueve el foco del campo que actualmente lo tiene hasta el que esté <n> posiciones después.
Sólo se cuentan aquellos campos que tienen "Hacer parada" con valor "Sí".
Si se pone la instrucción a la salida de un campo se ejecutará en el campo siguiente, no en el que estaba.

Sintaxis: SiguienteControl(<n>)

Tipo de dato devuelto: Nada

Ejemplo:

```
cliente = Leer("DOCCAB"; "Actual.CodCliente"; "");
si (cliente == 4) entonces {
  Mensaje("Este cliente debe usar el almacén 4");
  Escribir("DOCCAB"; "Actual.CodAlmacen"; "4"; "");
  SiguienteControl(2);
};
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: SI

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.15. DLL

Descripción:

Se ejecuta una dll desde dentro de un script con los siguientes parámetros:
nombre_dll: Nombre del archivo de la dll, si no incluye la ruta se busca en la carpeta desde la que se ha lanzado el programa. La extensión no es obligatoria. Es un dato de tipo texto.
Parámetros: Cadena de texto a pasar a la dll. Puede ser "".
índice_función: Índice de la función a llamar dentro de la dll. Si no se indica se usa "1".
conexión_BDD: Nombre del alias de la conexión. Si no se especifica se usa la del programa.

Sintaxis: DLL(<nombre_dll>[;<parámetros>[;<índice_función>[;<conexión_BDD>]]])

Tipo de dato devuelto: Nada

Ejemplo:

```
parametros= "email=miemail" + caracter(9) + "pass=password"+caracter(9)+"mobile=655555555"+ caracter(9)+
"id=Empresa"+ caracter(9)+
 "country=34" + caracter(9) + "sms=prueba";
dll("SMSAfilnet"; parametros);
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.16. CumpleRestricción

Descripción:

Devuelve cierto si <valor> cumple los criterios de <restricción>.
<restricción> es una cadena de texto con las condiciones a cumplir. Pueden verse sus posibilidades en el apéndice del manual del editor de informes.
<valor> puede ser una cadena de texto o un número.
Para que valor se compare contra un rango numérico, "1..10" por ejemplo, debe ser un número.
Para poder comparar fechas deben pasarse el valor como tipo fecha. Cuando se obtiene una fecha con ConsultaCampo o ConsultaLista ya es de tipo fecha, en el resto de los casos es de tipo texto y no puede compararse contra una restricción de fechas.

Sintaxis: CumpleRestricción(<valor>; <restricción>)

Tipo de dato devuelto: Lógico

Ejemplo:

```
v1  = 5;
v2  = "5";
v3  = "15/12/2016";
v3f = ConsultaCampo("select fechaalta from cliente where codigo = 1"); %% valor: 31/01/2016
r1   = "1..10; 15..20";
r2   = "3; 5 ; 7";
r3  = "01/01/2015..01/02/2016";

x1  = CumpleRestricción(v1; r1);       %% x1 valdrá cierto
x2  = CumpleRestricción(v2; r1);       %% x2 valdrá falso
x11 = CumpleRestricción(v1; r2);       %% x11 valdrá cierto
x21 = CumpleRestricción(v2; r2);       %% x21 valdrá falso
x3  = CumpleRestricción(v3; r3);       %% x3 valdrá falso
x3f = CumpleRestricción(v3f; r3);      %% x3f valdrá cierto
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.17. ActualInsertar

Descripción:

Se inserta una nueva línea en el documento actual con los valores por defectos rellenados.
Entidad: Tabla en la que insertar.
- DOCLIN, sólo funciona en la ventana de documentos.
- FABFABRILIN, artículos fabricados.

Sintaxis: ActualInsertar(entidad)

Tipo de dato devuelto: Nada

Ejemplo:

```
%% Artículos de regalo en función deL importe del documento (son acumulativos)
ListaRegalos = #(
 #[Importe:  900  Artículo: "JK326"];
 #[Importe: 2000  Artículo: "A525"];
 #[Importe: 3000  Artículo: "YH2500"]
);
%% Obtención base imponible documento
ImporteDocumento = Leer("DOCCAB"; "Actual.BaseImponible"; "");
%% Número líneas documentos
Líneas = Leer("DOCLIN"; "Actual.CuantasLíneas"; "");
%% Bucle inserción regalos
LAplica(ListaRegalos; {
 si (ImporteDocumento >= ListaRegalos[params[1]].Importe) entonces {
   %% Comprobación de que no se ha asignado ya el regalo (existe el artículo con precio 0)
   ExisteRegalo = falso;
   i = 1;
   Mientras {i <= Líneas} haz {
    si (Leer("DOCLIN"; "Actual.CodArticulo"; Formatea("0"; i)) == ListaRegalos[params[1]].Artículo
     y Leer("DOCLIN"; "Actual.Precio"; Formatea("0"; i)) == 0
    ) entonces {
     ExisteRegalo = cierto;
     i = Líneas;   %% Salida del bucle
    };
    i = i + 1;
   };
   si (No ExisteRegalo) entonces {
    %% Inserción línea en documento con valores por defecto
    ActualInsertar("DOCLIN");
    %% Obtención datos para nueva líneas
    Coste = Leer("ARTICULO"; "PrecioCoste"; "Codigo = " + Entrecomillar(ListaRegalos[params[1]].Artículo));
    %% Rellenado campos nueva línea (el resto tendrán 0 o el valor por defecto, como el precio es 0 (regalo) no
afectan)
    %% Si existieran datos adicionales que se quisieran rellenar habría que incluirlos aquí
    Escribir("DOCLIN"; "Actual.CodArticulo"; ListaRegalos[params[1]].Artículo; Formatea("0"; Líneas + params[1]));
    Escribir("DOCLIN"; "Actual.Cantidad"; "1"; Formatea("0"; Líneas + params[1]));
    Escribir("DOCLIN"; "Actual.Coste"; Formatea("0.########"; Coste); Formatea("0"; Líneas + params[1]));
   };
  };
});
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.18. ActualBorrar

Descripción:

Se borra la línea indicada por posición, es un texto, si no se especifica se borra la línea actual.
Entidad: Tabla en la que borrar.
- DOCLIN, sólo funciona en la ventana de documentos.
- FABFABRILIN, artículos fabricados.

Sintaxis: ActualBorrar(entidad[; posición])

Tipo de dato devuelto: Nada

Ejemplo:

```
%% Script a la salida de Cantidad
Cantidad = Leer("DOCLIN";"Actual.Cantidad"; "");
si (Cantidad < 0) entonces {
  Mensaje("No se admiten abonos");
  ActualBorrar("DOCLIN");
};
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.19. ActualLeer

Descripción:

Devuele una lista de registros con los datos de las características
Entidad: Donde se lee.
- DOCLINCAR. Documentos (ventana documentos).
- FABFABRILINCARACT. Artículos fabricados (ventana fabricación, pestaña artículos fabricados).
- FabPrevLinMatCaract. Previstos Fabricación (ventana fabricación).
- FabUtilLinMarCaract. Utilizados Fabricación (ventana fabricación).
- RepPrevLinMatCaract. Previstos Reparación (ventana reparación).
- RepUtilLinMarCaract. Utilizados Reparación (ventana reparación).
- ObraPrevLinMatCaract. Previstos Proyectos (ventana proyectos).
- ObraUtilLinMarCaract. Utilizados Proyectos (ventana proyectos).

Característica: Código de característica a recuperar, "" son todas.
Línea: Número de línea del documento. "" significa la actual.
La lista de registros devuelta tiene este formato: #(#[cantidad: x codigo: y valor: 'tttttt']; #[...]; ...).
El campo código se refiere al código de la característica.
Los valores de las características están siempre en el formato de la base de datos (números: 00000000.########, fechas: aaaammdd), siempre son de tipo texto.
Sólo funciona en la ventana de documentos y artículos fabricados.

Sintaxis: ActualLeer(entidad; característica; línea)

Tipo de dato devuelto: Lista de registros

Ejemplo:

```
%% Teniendo una característica "Fecha Caducidad - Lote" de tipos fecha y número respectivamente
%% Script a la salida de cantidad
ValoresCaracterísticas = ActualLeer("DOCLINCAR"; ""; ""); %% Obtiene los valores de las características de la línea actual
Laplica(ValoresCaracterísticas; {
  %% Bucle para inspeccionar los valores recuperados
  FechaCaducidad = TxPalabra(ValoresCaracterísticas[params[1]].valor; "-"; 0);  %% Obtención fecha de caducidad
  Lote = TxPalabra(ValoresCaracterísticas[params[1]].valor; "-"; 1);          %% Obtención lote
  Hoy = FormateaFecha("yyyymmdd"; FechaHoy);                          %% Formateo fecha para compararla
  si (FechaCaducidad < Hoy) entonces {
    %% Producto caducado. Se muestra mensaje con la fecha formateada y el número de lote
    Mensaje("Se está vendiendo un producto caducado: "
      + TxTrozo(FechaCaducidad; 7; 2) + "/"
      + TxTrozo(FechaCaducidad; 5; 2) + "/"
      + TxTrozo(FechaCaducidad; 1; 4)
      + " Lote: " + Formatea("0"; EnNúmero(Lote))
      );
  };
});
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.20. ActualEscribir

Descripción:

Escribe los datos de las características de una línea del documento actual o un campo concreto de doclin. Cuando se utiliza sobre doclin el efecto es el mismo que teclear ese campo en la pantalla, por ejemplo, si se escribe en la cantidad se solicitarán las características si el artículo las tiene, y se aplicarán las condiciones de precio existentes. Si se usa el campo precio en tickets se hace con IVA incluido, como se usa en la ventana.

Entidad: Donde se va a escribir.
- *DOCLINCAR. Documentos (ventana documentos).*
- *FABFABRILINCARACT. Artículos fabricados (ventana fabricación, pestaña artículos fabricados).*
- *DOCLIN. Documentos (ventana documentos).*
- *FabPrevLinMatCaract. Previstos Fabricación (ventana fabricación).*
- *FabUtilLinMatCaract. Utilizados Fabricación (ventana fabricación).*
- *RepPrevLinMatCaract. Previstos Reparación (ventana reparación).*
- *RepUtilLinMarCaract. Utilizados Reparación (ventana reparación).*
- *ObraPrevLinMatCaract. Previstos Proyectos (ventana proyectos).*
- *ObraUtilLinMarCaract. Utilizados Proyectos (ventana proyectos).*

Campo: En el caso de DOCLINCAR y FABFABRILINCARACT es el código de característica a escribir. Si se indica se sustituyen los valores de ese código por la lista pasada en valor, esto implica que se borrarán todos los valores que no se indiquen explicitamente. Si se indica todas ("") se borran todas y sólo queda lo que se pase como valor. Tiene que indicarse como un texto. En el caso de DOCLIN es el nombre del campo donde grabar. No tiene porque coincidir con el nombre del campo en la tabla, ni siquiera con el de "Actual." Están documentados los campos posibles.
Valor: En el caso de DOCLINCAR y FABFABRILINCARACT es un lista de registros con el formato #(#[cantidad: x codigo: y valor: 'tttttt']; #[...];...).
Deben pasarse todos los campos. El valor siempre se pasa como texto y el código es el de la característica (tipo numérico). Un mismo artículo puede tener varias características. En el caso de DOCLIN es el valor a introducir en el campo. El código de la característica se introduce dos veces, en campo y en valor.
Línea: Número de línea del documento en texto. "" significa la actual.

Los valores de las características están siempre en el formato de la base de datos (números: 00000000.########, fechas: aaaammdd).

Sintaxis: ActualEscribir(entidad; campo; valor; línea)

Tipo de dato devuelto: Nada

Ejemplo:

```
%% Teniendo una característica "Fecha Caducidad - Lote" de tipos fecha y número respectivamente
%% Script a la salida de cantidad
ValoresCaracterísticas = ActualLeer("DOCLINCAR"; ""; ""); %% Obtiene los valores de las carácterísticas de la línea actual
Errores = falso;                          %% Para comprobar si existen valores caducados
ValoresCaracterísticasCorrectos = LCrea;          %% Para grabar los valores no caducados
Cantidad = 0;                             %% Para actualizar la cantidad si se eliminan valores
Laplica(ValoresCaracterísticas; {
 %% Bucle para inspeccionar los valores recuperados
 FechaCaducidad = TxPalabra(ValoresCaracterísticas[params[1]].valor; "-"; 0);  %% Obtención fecha de caducidad
 Lote = TxPalabra(ValoresCaracterísticas[params[1]].valor; "-"; 1);       %% Obtención lote
 Hoy = FormateaFecha("yyyymmdd"; FechaHoy);                    %% Formateo fecha para compararla
 si (FechaCaducidad < Hoy) entonces {
  %% Producto caducado. Se muestra mensaje con la fecha formateada y el número de lote
  Mensaje("Se está vendiendo un producto caducado: "
   + TxTrozo(FechaCaducidad; 7; 2) + "/"
   + TxTrozo(FechaCaducidad; 5; 2) + "/"
   + TxTrozo(FechaCaducidad; 1; 4)
   + " Lote: " + Formatea("0"; EnNúmero(Lote))
   );
   Errores = cierto;
 } sino {
```

```
    %% No es un producto caducado, se guarda en una nueva lista.
    LAñade(ValoresCaracterísticasCorrectos; ValoresCaracterísticas[params[1]]);
    %% Se guarda cantidad para actualizar la de la línea
    Cantidad = Cantidad + ValoresCaracterísticas[params[1]].Cantidad;
  };
});
%% Si ha habido valores caducados se graba la lista que no los contiene y se actualiza la cantidad de la línea
%% No se soportan las condiciones por cantidad al hacer esto.
si Errores entonces {
  ActualEscribir("DOCLINCAR"; ""; ValoresCaracterísticasCorrectos; "");
  Escribir("DOCLIN"; "Actual.Cantidad"; Formatea("0.########"; Cantidad); "");
};
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.21. SeleccionarConsulta

Descripción:

SeleccionarConsulta("select..." [; "campo" [; "orden" [; "título"]]])
SeleccionarConsulta("select..." [; parametros])

Muestra una ventana de selección con los registros devueltos por la sentencia sql "query" y devuelve el valor de la columna indicada en "campo" del registro seleccionado. Si no se indica "campo" se devolverá el primero de la consulta.
"orden" indica el nombre del campo por el que se ordenan los registros de la ventana, puede cambiarse el orden en la ventana pinchando en el título de cualquier columna.
"Título" es el título de la ventana.

Parametros es un registro que permite especificar diversas opciones: #[opcion: valor; ...]
Campo (texto): nombre del campo cuyo valor se retornará (por defecto el primero del select)
Valor: valor por el que se buscará en el Campo para que la ventana aparezca posicionada en él
Orden (texto): nombre del campo por el cual se ordenarán los registros al mostrarlos, pueden ser varios separados por punto y coma, p.ej.
'serie;numero', sin espacios en blanco (por defecto se utilizará el campo a retornar).
Título (texto): título para la ventana de selección
Conexión (texto): el nombre de una conexión creada con ConectaSQL o ConectaBDE (por ahora no de ConectaADO porque la ventana de selección no lo soporta)

Retorna:
Si el usuario acepta la ventana: el valor del campo especificado. El tipo de dato devuelto será el del valor seleccionado.
Si el usuario cancela la ventana: nada

Sintaxis: SeleccionarConsulta(query [; campo [; orden [; título]]])

Tipo de dato devuelto: Variable

Ejemplo:

```
%% Ejemplo 1
CódigoCliente = SeleccionarConsulta("select nombrecomercial, codigo from cliente";"codigo"; ""; "Selección cliente");

%% Ejemplo 2
ConectaSQL('conta'; 'C:\Distrito\ContaCS\Database\Ejemplo\2018.FDB');
seleccionado = SeleccionarConsulta('select * from apuntes'; #[conexion: 'conta']);
Desconecta('conta');
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.22. EnEan

Descripción:

Devuelve <t> de forma que al usarlo con una fuente de códigos de barras EAN sea legible para los lectores de códigos de barras.

Retorna el texto codificado para poder utilizar con las fuentes EAN13_36.ttf y EAN13_72.ttf (sirven para EAN-13 y EAN-8)
si texto no es un EAN-13 o un EAN-8 válido, retorna un texto vacío
el parámetro es obligatorio

Sintaxis: EnEan(<t>)

Tipo de dato devuelto: Texto

Ejemplo:

```
x = EnEAN("8480000276346");
%% x valdrá '+!4I0AA0-chgdeg!'
```

Instrucción específica de contabilidad: NO

Instrucción específica de formatos de edición: NO

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.23. Visor

Descripción:

Envía al visor el texto de cada elemento de la lista, cada elemento en una línea.
En la configuración de tpv no debe estar indicado ningún script en los eventos para el visor para que no se repita la información.

Sintaxis: visor(<lista>)

Tipo de dato devuelto: Nada

Ejemplo:

```
%% Visor
ArtículoCantidad = Leer("DOCLIN"; "Actual.Cantidad"; "");
ArtículoNombre = Leer("DOCLIN"; "Actual.NombreArticulo"; "");
ArtículoImporteIvaIncluido = Leer("DOCLIN"; "Actual.IIIMPORTE"; "");

Visor(
 #(
  si ArtículoCantidad == 1 entonces {
   ArtículoNombre
  } sino {
   Formatea(",0.####"; ArtículoCantidad) + " X " + ArtículoNombre
  };
  ArtículoImporteIvaIncluido
 )
);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición: SI

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.24. GenerarInforme

Descripción:

Ejecuta un informe y lo imprime, visualiza o guarda su resultado como fichero pdf, xls, csv o rtf sin cuadros de texto.
Parámetros:
informe.idk: Ruta y nombre del fichero del informe.
informe.ext: Puede ser un texto o un registro. Si es un texto será la ruta y nombre del fichero donde se guardará el resultado del informe. Si existe se sobreescribirá.Puede ser pdf, xls, csv o rtf. Si se pasa '' se mostrará el visor de impresión, igual que cuando se emite a pantalla. Si es un registro puede tener un campo "fichero", en cuyo caso funciona igual que el texto, si el campo es "impresora", se indica el nombre de la impresora, igual que se ve en los formatos de impresión al seleccionarla, si se indica '' se imprimirá por la impresora predeterminada del usuario que ejecuta la instrucción. Si se pasa un registro vacío es igual que pasar un texto vacío, sale el visor.
Se puede pasar una lista de registros con los siguientes campos: nombre y valor. El nombre será el mismo que aparece en la ventana "Modificar las entidades", pestaña "Restricción" sin los espacios en blanco que pudiera tener, también se puede poner Rx, sustiyendo la x por la posición de la restricción en la ventana, empezando a contar en 1.
El valor será lo que se introduce en la restricción, puede ponerse una fecha para que se interprete, por ejemplo 1010 se convertirá en 10/10/xxxx, siendo xxxx el año actual.

En la opción del menú "Configuración -> Copia de seguridad automática" puede programarse un tipo de operación "Ejecutar script" para programar la ejecución de un script que genere el informe.

Sintaxis: GenerarInforme(informe.idk; informe.ext'; #(#[nombre: nombre_restricción valor: valor_restricción]; #[]; ...))

Tipo de dato devuelto: Nada

Ejemplo:

```
diaLunes = FechaEnNúmero(FechaHoy) - DíaDeSemana(FechaHoy) + 1; %% lunes de la semana actual
lunes = FechaEnTexto(diaLunes);
restriccionFecha = lunes+'..'+FechaEnTexto(diaLunes+6); %% de lunes a domingo
pdf = 'C:\Distrito\Informes\Albaranes-Semana ' + EnTexto(SemanaDe(lunes)) + '-' + Formatea('0'; AñoDe(lunes)) +
'.pdf';

GenerarInforme(
 'C:\Distrito\Pyme\Informes\Informe de documentos de venta\Documentos de venta por fecha y cliente.IDK';
 pdf;
 #(
  #[nombre: 'R1' valor: '2' %albarán%];
  #[nombre: 'DocumentoDeVenta.fecha' valor: restriccionFecha]
 )
);

destinatario = "gerente@empresa.com";
asunto = "Informe Albaranes-Semana " + EnTexto(SemanaDe(lunes)) + '-' + Formatea('0'; AñoDe(lunes));
texto = "Informe adjunto";

resultado = EnviarCorreo(destinatario; asunto; texto; ''; #(pdf));
si no TxEstaVacío(resultado) entonces {
 error(resultado);
};


%% Imprimiendo directamente
GenerarInforme(
 'C:\Distrito\Pyme\Informes\Informe de documentos de venta\Documentos de venta por fecha y cliente.IDK';
 #[impresora: "Genérica / Sólo texto"];
 #(
```

```
  #[nombre: 'R1' valor: '2' %albarán%];
  #[nombre: 'DocumentoDeVenta.fecha' valor: restriccionFecha]
 )
);


%% Impresión directa en red
GenerarInforme(
 "\\Servidor\distrito\Pyme\Informes\Etiquetas artículos.IDK";
 #[impresora: "\\Caja1\Posiflex PP6800 Partial Cut v3.01"];
 #(#[nombre: "Artículo.Código" valor: artículo])
);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones varias

15.25. Escape

Descripción:
Permite, desde un formato de impresión, enviar a la impresora una lista de caracteres ASCII, para poder enviar secuencias de escape, caracteres de control, etc.

Sintaxis: Escape(nASCII1 [, nASCII2 [, nASCII3 [, ...]]])

Tipo de dato devuelto:

Ejemplo:
```
Escape(27, "E");          %% Hace un reset en la impresora (PCL)
Escape(27, 112, 0, 25, 250);  %% Abre el cajón portamonedas (Epson)
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión: SI

Tipo de instrucción: Funciones varias

15.26. CargarDocumento

Descripción:

Recibe como parámetro el código de un documento y nos devuelve un registro con múltiples campos y listas relativas a los importes, bases, cuotas, etc. del documento, tal y como se calcularían en caso de imprimirlo.

El registro devuelto tiene los siguientes campos:

Coste
PorcBenef
Beneficio
Bruto
Neto
Descuento
DescuentoLin
DescuentoDoc
DescuentoPP
Base
CuotaIva (0 si es iva soportado que soporte y repercuta)
CuotaIvaContable (mismo valor que CuotaIva excepto en iva soportado que soporte y repercuta)
CuotaRecEquiv
CuotaIrpf
Importe
ImportePendiente
IIPortes
IIGastosVarios
IIBruto
IINeto
IIDescuento
IIDescuentoLin
IIDescuentoDoc
IIDescuentoPP
ListaIva La lista de ivas es una lista de registros con los siguientes campos:
TipoIva
TipoRecEquiv
Base
CuotaIva (0 si es iva soportado que soporte y repercuta)
CuotaIvaContable (mismo valor que CuotaIva excepto en iva soportado que soporte y repercuta)
CuotaRecEquiv
CuotaRetencionBase (sólo aplicable a documentos enlazados a un proyecto con retención base)
ListaIrpf La lista de irpfs es una lista de registros con los siguientes campos:
TipoIrpf
Base
CuotaIrpf

Sintaxis: CargarDocumento(nCodigoDocumento)

Tipo de dato devuelto: Registro

Ejemplo:

```
RDatos = CargarDocumento(DocumentoCódigoInterno);
txBases = "";
txTipos = "";
txCuotas = "";
i = 1;
Mientras {i <= ltamaño(RDatos.ListaIva)} haz {
 si i > 1 entonces {
   txBases = txBases + intro;
   txTipos = txTipos + intro;
   txCuotas = txCuotas + intro;
```

```
};

txBases = txBases + Formatea(",0.00"; RDatos.ListaIva[i].Base);
txTipos = txTipos + Formatea("0.##%"; RDatos.ListaIva[i].TipoIva);
txCuotas = txCuotas + Formatea(",0.00"; RDatos.ListaIva[i].CuotaIva);

i = i + 1;
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.27. AbrirOpción

Descripción:
Abre una opción del menú general de la aplicación, igual que si se hubiera pulsado sobre ella en el menú.
El nombre de la ventana puede verse en el editor de menú al pulsar en el icono redondo azul a la derecha de "Opción menú".
Si se llama desde un script y se abre una ventana modal, el script detiene su ejecución hasta que se cierra la ventana abierta por esta opción, si no es modal el script continua justo después de abrir la ventana.

Sintaxis: AbrirOpción(<acción>;<parametros>)

Tipo de dato devuelto: Nada

Ejemplo:
AbrirOpción("ActionArchivoArticulos");

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.28. ObtenerInformación

Descripción:

Obtiene la ruta o nombre de fichero que se le especifique con el parámetro "acción";
Valores posibles:
- RutaTemp: Ruta de la carpeta Temp que se está usando.
- FicheroTemp: Genera un nombre de fichero en la carpeta temp que no existe. Es para poder crear un fichero temporal o una carpeta con un nombre aleatorio. Debe crearse el fichero en el momento que se ejecuta esta instrucción, sino se podría crear el mismo fichero desde otro proceso.
- RutaTrabajo: Ruta desde la que se ha lanzado el programa.
- RutaConfigLocal: Ruta donde se guarda el fichero de configuración del equipo para el programa.
- RutaCompart: Ruta de la carpeta compart que figura en el ini de la aplicación.
- FicheroBaseDatos: Ruta y nombre del fichero fdb de la base de datos de la empresa en la que se ejecuta el script. Siempre es la ruta local desde el
servidor. Ejemplo: "c:\distrito\pyme\database\ejemplo\2014.fdb".
- ServidorBaseDatos: Nombre del servidor incluyendo el puerto si se ha configurado en el ini. Ejemplo: "servidor/3051".
- RutaBaseDatos: ServidorBaseDatos y FicheroBaseDatos juntos, separados por ":". Ejemplo:
"servidor/3051:c:\distrito\pyme\database\ejemplo\2014.fdb".
- NombreAplicación: Devuelve el nombre de la aplicación según esta lista: Pyme, Obras, TPV, Caffe, Conta, Horizontal, Inmobiliarias, Vertical
- EjecutableAplicación: Ruta desde la que se ha lanzado el ejecutable de la aplicación.
- VersiónAplicación: Fecha de la versión. Se devuelve como tipo texto. Ejemplo: "23/12/2016".
- RutaDkGlobal: Ruta del archivo dk_global.fdb.Incluye el servidor y el puerto si existen.

Sintaxis: ObtenerInformacion(acción)

Tipo de dato devuelto: Texto

Ejemplo:

```
a = ObtenerInformación('RutaTemp');
b = ObtenerInformación('FicheroTemp');
c = ObtenerInformación('RutaTrabajo');
d = ObtenerInformación('RutaConfigLocal');
e = ObtenerInformación('RutaCompart');
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.29. ObtenerInformacionFichero

Descripción:
ObtenerInformacionFichero('fichero')

Obtiene información sobre el fichero o directorio que se le especifique
Retorna nada si el fichero no existe
Retorna un registro con los siguientes campos si el fichero existe:
- *Tamaño (valor de tipo numérico, en bytes; si es un directorio, este campo NO lo hay)*
- *FechaModificación (valor de tipo fecha-hora, se puede convertir a tipo texto usando EnTexto)*
- *FechaCreación (valor de tipo fecha-hora, se puede convertir a tipo texto usando EnTexto)*
- *EsDirectorio (valor de tipo lógico)*

Sintaxis: ObtenerInformacionFichero('fichero')

Tipo de dato devuelto: TRegistro

Ejemplo:
info = ObtenerInformaciónFichero('C:\Distrito\Pyme\PYME.EXE');

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.30. ObtenerListaFicheros

Descripción:

ObtenerListaFicheros('ruta'; 'máscara')

Averigua información sobre los ficheros y subdirectorios del directorio cuya ruta se indica
El primer parámetro es obligatorio, el segundo es opcional
Máscara es opcional
- *Si no se indica o está vacía, retorna todo el contenido del directorio*
- *Si se indica, retorna información sólo sobre ese fichero/directorio*

*también se puede indicar una máscara del tipo *.txt o similar*
Retorna una lista de registros, uno por cada fichero/subdirectorio, cada uno con los siguientes campos:
- *Nombre*
- *Tamaño (valor de tipo numérico, en bytes; si es un directorio, este campo NO lo hay)*
- *FechaModificación (valor de tipo fecha-hora, se puede convertir a tipo texto usando EnTexto)*
- *FechaCreación (valor de tipo fecha-hora, se puede convertir a tipo texto usando EnTexto)*
- *EsDirectorio (valor de tipo lógico)*

Sintaxis: ObtenerListaFicheros('ruta' [; 'máscara'])

Tipo de dato devuelto: TLista

Ejemplo:

listaExe = ObtenerListaFicheros('C:\Distrito\Pyme'; '.exe');*

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.31. PDF417

Descripción:

Instrucción para imprimir códigos 2D en formato pdf417 en formatos de impresión de documentos e informes del generador.

Esta instrucción hace que el campo fórmula en el que se coloque la instrucción, tiene que ser obligatoriamente la última instrucción del script, imprima un código de barras 2D en formato pdf417 con el texto pasado como parámetro a la función.

El primer parámetro es obligatorio, el segundo opcional
Si el primer parámetro es una lista, se irán concatenando sus elementos, que serán trozos de texto o códigos ascii de caracteres

El segundo parámetro indica el ratio de altura (p.ej. 3 indica el triple de la altura normal); por defecto es 1.
Si el campo en el formato de impresión respeta la anchura del código de barras y reescala la altura, aumentar la escala tiene el efecto de aumentar la altura. El campo ArtículoImagenFórmula funciona así.
Si el campo en el formato de impresión respeta la altura del código de barras y reescala la anchura, aumentar la escala tiene el efecto de encoger la anchura. La mayoría de los campos fórmula funcionan de esta manera.

Sintaxis: pdf417(<t>[; <n>])

Tipo de dato devuelto: Texto

Ejemplo:

PDF417('114491 2 1492071 X-2015035-001'; 3);

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.32. LeerBalanza

Descripción:

Ejecuta la acción configurada en la opción "Otros dispositivos" y la adquisición asociada.
No recibe ningún parámetro
Retorna un registro con los siguientes campos numéricos:
Cantidad
Precio
Descuento

Sintaxis: LeerBalanza

Tipo de dato devuelto: TRegistro

Ejemplo:

```
RBalanza = LeerBalanza;
Escribir("DocLin"; "Actual.Cantidad"; Formatea("0.###"; RBalanza.Peso); "");
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.33. ReproducirSonido

Descripción:
Reproduce ficheros wav

Sintaxis: ReproducirSonido(<t>)

Tipo de dato devuelto: Nada

Ejemplo:
ReproducirSonido("C:\Windows\Media\notify.wav")

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.34. SeleccionarTipo

Descripción:
Selecciona un elemento de un árbol de la tabla tipo
El parámetro tipo es obligatorio, el título es opcional
Retorna el código, 0 si se canceló

Sintaxis: SeleccionarTipo(tipo; 'título')

Tipo de dato devuelto: Número

Ejemplo:
familia = SeleccionarTipo(13; "Seleccione familia");

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.35. EmitirDocumento

Descripción:

Emite el documento con el código pasado como parámetro, si no está emitido generará los vencimientos y los cobros si así lo indica la forma de pago. Igual que funciona la opción de reemitir documentos.
Parámetros:
códigoDocumento: Obligatorio. Código interno del documento. Campo código en tabla DocCab.
formato.fmt: Obligatorio. Ruta y nombre del fichero de formato a utilizar.
documento.ext: Opcional. Puede ser un texto o un registro.
Si es un texto será la ruta y nombre del fichero donde se guardará el formato impreso. Si existe se sobreescribirá.Puede ser pdf, xls, csv o rtf. Si se pasa '' se mostrará el visor de impresión, igual que cuando se emite a pantalla.
Si es un registro puede tener un campo "fichero", en cuyo caso funciona igual que el texto, si el campo es "impresora", se indica el nombre de la impresora, igual que se ve en los formatos de impresión al seleccionarla, si se indica '' se imprimirá por la impresora predeterminada del usuario que ejecuta la instrucción, si el formato tiene asignada una impresora se usará esta, independientemente de lo que se diga aquí. Si se pasa un registro vacío es igual que pasar un texto vacío, sale el visor.

Sintaxis: EmitirDocumento(códigoDocumento; 'formato.fmt ['; 'documento.ext']])

Tipo de dato devuelto: Nada

Ejemplo:

```
EmitirDocumento(
  1228;
  'C:\Distrito\PymeVentas.fmt';
  'C:\Distrito\Pyme\Documento 28547.pdf'
);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.36. EnviarCorreo

Descripción:

Antes de usar esta instrucción debe configurarse el correo en el menú comunicaciones.

Parámetros:
correo_destinatario: Obligatorio. Dirección de correo a la que enviar.
asunto: Obligatorio. Texto del asunto.
texto: Texto del mensaje en html. Si no se indica se usará la primera plantilla definida.
remitente: Dirección de correo del remitente. Si no se indica se usa el del campo "E-mail" de los datos de la empresa.
#(fichero1; fichero2; ...): Opcional. Nombre de los ficheros con su ruta para adjuntar al correo electrónico.

Retorna el mensaje de error del servidor si se ha producido alguno y si no un texto vacío. Esto sólo pasa si la cuenta de correo no está configurada con "Enviar los correos desde la bandeja de salida", en ese caso, si se produce algún error se verá en la ventana de alertas del servicio.
Si hay configuradas varias cuentas de correo se utiliza la primera según el orden de la ventana de "Configuración de cuentas".

Sintaxis: EnviarCorreo('correo_destinatario'; 'asunto'; 'texto'; 'remitente'; #(fichero1; fichero2; ...))

Tipo de dato devuelto: Texto

Ejemplo:

```
TxtError = EnviarCorreo("cliente@empresa.com"; "Prueba correo"; "Prueba mensaje"; "consultas@distritok.com");
si (TxtError <> "") entonces {
  Mensaje("Error al enviar correo: " + TxtError);
};

%
Ejemplo para enviar un correo a través de etiquetas de envío si se pone un dato adicional
de cabecera a Sí y sólo con la 1ª etiqueta. Se envia con un texto fijo y los datos del cliente
del albarán.
%

ResultadoNada({
  codigo=Leer("Doccab";"Actual.Codformaenvio";"");
  cod_cliente=Leer("Doccab";"Actual.Codcliente";"");
  correo=Leer("Tipo";"OBSERVACIONES";"Tipo=3 and codigo="+formatea("0";codigo));
  cod_documento=Leer("Doccab";"Actual.Codigo";"");
  texto=Leer("Carvalor";"valor";"codcaract=912 and codclase=1 and codobjeto="+formatea("0";cod_documento));

  si (texto=='Sí' Y Bulto==1) entonces{
    TxtError = EnviarCorreo(correo; "Recogida Paquete"; " Solicitamos recogida en camiño cruceiro 14, Vigo de "
+EnTexto(TotalBultos)+" bultos" + "
  con destino "
    +"<br> " + @"Nombre Comercial"+"<br> " + dirección + "<br> "+CP +" "+ Poblacion+ "<br> " +Provincia +"<br>
"+"<br> " +"Por favor, confirmen la
  recepcion de este email"
     ; "envios@empresa.com");
    si (TxtError <> "") entonces {
      Mensaje("Error al enviar correo: " + TxtError);
    };
  };
});
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.37. EnviarCorreoCuenta

Descripción:

Antes de usar esta instrucción debe configurarse el correo en el menú comunicaciones.

Parámetros:
cuenta: Obligatorio. Dirección de correo de la cuenta que se usará para enviar. Si se indica una cadena vacía ("") se usará la primera según el orden de la ventana de "Configuración de cuentas".
correo_destinatario: Obligatorio. Dirección de correo a la que enviar.
asunto: Obligatorio. Texto del asunto.
texto: Texto del mensaje o código de plantilla. Si es un número se entiende que es una plantilla y si no es html. Si no se indica se usará la primera
plantilla definida. No se usa el asunto de la plantilla.
remitente: Dirección de correo del remitente. Si no se indica se usa el del campo "E-mail" de los datos de la empresa.
#(fichero1; fichero2; ...): Opcional. Nombre de los ficheros con su ruta para adjuntar al correo electrónico.

Retorna el mensaje de error del servidor si se ha producido alguno y si no un texto vacío. Esto sólo pasa si la cuenta de correo no está configurada con "Enviar los correos desde la bandeja de salida", en ese caso, si se produce algún error se verá en la ventana de alertas del servicio.

Sintaxis: EnviarCorreoCuenta('cuenta';'correo_destinatario'; 'asunto'; 'texto'; 'remitente'; #(fichero1; fichero2; ...))

Tipo de dato devuelto: Texto

Ejemplo:

```
TxtError = EnviarCorreoCuenta("consultas@distritok.com";"cliente@empresa.com"; "Prueba correo"; "Prueba mensaje";
"consultas@distritok.com");
si (TxtError <> "") entonces {
 Mensaje("Error al enviar correo: " + TxtError);
};

%
Ejemplo para enviar un correo a través de etiquetas de envío si se pone un dato adicional
de cabecera a Sí y sólo con la 1ª etiqueta. Se envía con un texto fijo y los datos del cliente
del albarán.
%

ResultadoNada({
 codigo=Leer("Doccab";"Actual.Codformaenvio";"");
 cod_cliente=Leer("Doccab";"Actual.Codcliente";"");
 correo=Leer("Tipo";"OBSERVACIONES";"Tipo=3 and codigo="+formatea("0";codigo));
 cod_documento=Leer("Doccab";"Actual.Codigo";"");
 texto=Leer("Carvalor";"valor";"codcaract=912 and codclase=1 and codobjeto="+formatea("0";cod_documento));

 si (texto=='Si' Y Bulto==1) entonces{
  TxtError = EnviarCorreoCuenta(""; correo; "Recogida Paquete"; " Solicitamos recogida en camiño cruceiro 14,
Vigo de " +EnTexto(TotalBultos)+"
bultos" + " con destino "
  +"<br> " + @"Nombre Comercial"+"<br> " + dirección + "<br> "+CP +" "+ Poblacion+ "<br> " +Provincia +"<br>
"+"<br> " +"Por favor, confirmen la
recepcion de este email"
   ; "envios@empresa.com");
  si (TxtError <> "") entonces {
   Mensaje("Error al enviar correo: " + TxtError);
  };
 };
```

```
});
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.38. CrearDocumento

Descripción:

CrearDocumento(#[campo1: valor1 campo2: valor2 ...])

Inserta un registro en doccab dando valor a ciertos campos (es obligatorio darle valor al campo tipo y no dárselo a codigo) a los campos que no se les dé valor, si es posible se les pone uno por defecto automáticamente.
Retorna el código del documento generado.

No carga automáticamente los datos del cliente (régimen iva, forma de pago, portes, agente, dirección, forma de envío y descuentos).
Si no se manda el campo serie, el documento se generará con la serie en blanco. El numerador se obtiene automáticamente según la serie enviada.

Tras haber grabado las líneas, es recomendable llamar a RecalcularImportesConsultaDocumento para que recalcule y grabe los campos ImporteXXX de la cabecera.

Sintaxis: CrearDocumento(#[campo1: valor1 campo2: valor2 ...])

Tipo de dato devuelto: Número

Ejemplo:

```
codDoc = CrearDocumento(#[tipo: 2; codcliente: 2; codalmacen: 1; codformapago: 1]);
EjecutarSQL(
  'insert into doclin (coddocumento, codigo, nivel, codarticulo, cantidad, precio, coste, tipoiva, codalmacen) values
(:coddocumento, :codigo,
:nivel, :codarticulo, :cantidad, :precio, :coste, :tipoiva, :codalmacen)';
  #[coddocumento: codDoc; codigo: 10; nivel: 1; codarticulo: 'alfombrilla'; cantidad: 1,2; precio: 10; coste: 9;
tipoiva: 21; codalmacen: 1]
);
RecalcularImportesConsultaDocumento(codDoc);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.39. CrearApunteCaja

Descripción:
Inserta un registro en caja. Se pasa un registro con los campos a rellenar y sus valores. Es obligatorio indicar el campo fecha y no indicar el campo codapunte, el resto de campos son opcionales.
Retorna el codapunte generado.

Sintaxis: CrearApunteCaja(#[campo1: valor1 campo2: valor2 ...])

Tipo de dato devuelto: Número

Ejemplo:
```
fecha = FechaHoy;
codApunte = CrearApunteCaja(#[fecha: fecha; tipo: 7; estado: 'T'; descripcion: 'prueba'; importe: 1234]);
ImprimirTraspaso(fecha; codApunte);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.40. PreguntaNúmeroTáctil

Descripción:
Muestra un cuadro de diálogo solicitando un texto con el mensaje <t1>.

Si se especifica <t2> se mostrará como valor por defecto.

Si se indica cierto en el tercer parámetro <l> se ocultan los caracteres de manera que no se vean al introducirlos (enmascarado).

Aparecerán los botones "Aceptar" y "Cancelar". Si se selecciona "Aceptar" se devuelve el texto introducido, si se presiona "Cancelar" se interrumpe el script.

Pueden usarse diferentes líneas, $n para cambiar de línea, o tabuladores, $t para añadirlos, dentro del texto de mensaje <t>.

Sintaxis: PreguntaNúmeroTáctil(<t1>[;<t2>][;<l>])

Tipo de dato devuelto: Texto

Ejemplo:
cp = PreguntaNúmeroTáctil("Introduzca el código postal"; "15005");

%%Pregunta contraseña
Contraseña=PreguntaNumeroTactil("Indica contraseña";"";cierto);

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.41. NuevoCorreo

Descripción:

Antes de usar esta instrucción debe configurarse el correo en el menú comunicaciones.
Es igual que EnviarCorreo pero abre una ventana con el correo que el usuario podrá modificar. Para que se realice el envío tiene que pulsar en "Enviar".
Si esta instrucción está dentro de un script, el mismo se detiene hasta que le enviamos el correo a nuestro servidor de correo saliente.
Se usará la primera cuenta según el orden de la ventana de "Configuración de cuentas".
El remitente del correo será la cuenta utilizada para enviar.

Parámetros:
correo_destinatario: Obligatorio. Dirección de correo a la que enviar.
asunto: Obligatorio. Texto del asunto.
texto: Texto del mensaje o código de plantilla. Si es un número se entiende que es una plantilla y si no es html. Si no se indica se usará la primera
plantilla definida. No se usa el asunto de la plantilla.
#(fichero1; fichero2; ...): Opcional. Nombre de los ficheros con su ruta para adjuntar al correo electrónico.

Si la cuenta de correo está configurada para enviar desde la bandeja de salida y se produce algún error se verá en la ventana de alertas del servicio.
Si no está configurada la bandeja se muestra el error en una ventana sin cerrar el correo.

Si se ha pulsado "Enviar" se devuelve el código del correo generado (Número)
Si se ha cerrado la ventana sin enviar se devuelve -1 (Número)
Si se produce algún error al abrir la ventana del correo se devuelve el error generado.

Sintaxis: NuevoCorreo('correo_destinatario'; 'asunto'; 'texto'; #(fichero1; fichero2; ...))

Tipo de dato devuelto: Texto

Ejemplo:

```
TxtError = NuevoCorreo("cliente@empresa.com"; "Prueba correo"; "Prueba mensaje");
si (TxtError <> "") entonces {
  Mensaje("Error al enviar correo: " + TxtError);
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.42. NuevoCorreoCuenta

Descripción:

Antes de usar esta instrucción debe configurarse el correo en el menú comunicaciones.
Es igual que EnviarCorreoCuenta pero abre una ventana con el correo que el usuario podrá modificar. Para que se realice el envío tiene que pulsar en "Enviar".
Si esta instrucción está dentro de un script, el mismo se detiene hasta que le enviamos el correo a nuestro servidor de correo saliente.
El remitente del correo será la cuenta utilizada para enviar.

Parámetros:
cuenta: Obligatorio. Dirección de correo de la cuenta que se usará para enviar. Si se indica una cadena vacía ("") se usará la primera según el orden de la ventana de "Configuración de cuentas".
correo_destinatario: Obligatorio. Dirección de correo a la que enviar.
asunto: Obligatorio. Texto del asunto.
texto: Texto del mensaje o código de plantilla. Si es un número se entiende que es una plantilla y si no es html. Si no se indica se usará la primera plantilla definida. No se usa el asunto de la plantilla.
#(fichero1; fichero2; ...): Opcional. Nombre de los ficheros con su ruta para adjuntar al correo electrónico.

Si la cuenta de correo está configurada para enviar desde la bandeja de salida y se produce algún error se verá en la ventana de alertas del servicio.
Si no está configurada la bandeja se muestra el error en una ventana sin cerrar el correo.

Si se ha pulsado "Enviar" se devuelve el código del correo generado (Número)
Si se ha cerrado la ventana sin enviar se devuelve -1 (Número)

Sintaxis: NuevoCorreoCuenta('cuenta';'correo_destinatario'; 'asunto'; 'texto'; #(fichero1; fichero2; ...))

Tipo de dato devuelto: Texto

Ejemplo:

```
TxtError = NuevoCorreoCuenta("consultas@distritok.com";"cliente@empresa.com"; "Prueba correo"; "Prueba mensaje");
si (TxtError <> "") entonces {
  Mensaje("Error al enviar correo: " + TxtError);
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

Si se produce algún error al abrir la ventana del correo se devuelve el error generado.

15.43. RLeeXML

Descripción:

Parsea un xml y lo carga en un registro.
El registro debe existir, si tiene contenido se borrarán sus datos.
No se cargan los atributos, sólo los tags.
Si el tag no tiene hijos se carga el valor del tag como cadena de texto.
Si los tiene se pierde el valor del tag y se carga un registro en ese campo con los hijos como campos. Si un hijo aparece varias veces se creará una lista donde cada elemento será el valor del tag de los hijos.

Sintaxis: RLeeXML(<r>; <t>)

Tipo de dato devuelto: Nada

Ejemplo:

```
xml1 = '<literales><literal>Hola</literal></literales>';
RLeeXML(regxml1; xml1); %% regxml1 valdrá #[literales: #[literal: 'Hola']]
xml2 = '<literales><literal>Hola</literal><literal>Adios</literal></literales>'
RLeeXML(regxml2; xml2); %% regxml2 valdrá #[literales: #[literal: #('Hola'; 'Adios')]]
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.44. RLeeJSON

Descripción:

Parsea un json y lo carga en un registro.
El registro debe existir, si tiene contenido se borrarán sus datos.

Sintaxis: RLeeJSON(registro; "texto JSON")

Tipo de dato devuelto: Nada

Ejemplo:

```
tx_json = '{"departamento":8,"nombredepto":"Ventas","director": "juan
rodriguez","empleados":[{"nombre":"Pedro","apellido":"Fernandez"},{"nombre":"Jacinto","apellido":"Benavente"}
]}';
r_json = #[];
RLeeJSON(r_json; tx_json);
%
Valor r_json:
#[
  departamento: 8
  director: 'juan rodriguez'
  empleados: #(
      #[
          apellido: 'Fernandez'
          nombre: 'Pedro'
      ];
      #[
          apellido: 'Benavente'
          nombre: 'Jacinto'
      ]
  )
  nombredepto: 'Ventas'
]
%
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.45. QRCode

Descripción:

Instrucción para imprimir códigos QR en formatos de impresión de documentos e informes del editor.

Esta instrucción hace que el campo fórmula en el que se coloque la instrucción, tiene que ser obligatoriamente la última instrucción del script, imprima un código QR con el texto pasado como parámetro a la función.
No puede utilizarse en los siguientes campos fórmula: ArtículoFórmula, MaterialArtículoFórmula, RecursoFórmula, RecursosPrevistosFórmula, RecursosUtilizadosFórmula, MaterialesPrevistosFórmula, MaterialesUtilizadosFórmula, ArtículoObraFórmula, ArtículosFabricadosFórmula.

Si puede usarse con ArtículoImagenFórmula.
Puede utilizarse en los campos fórmula del editor de informes.

Sintaxis: QRCode(<texto>)

Tipo de dato devuelto: Texto

Ejemplo:

```
%% Texto fijo
QRCode('MECARD:N:Distrito K;TEL:902119554;URL:www.distritok.com;EMAIL:distritok@distritok.com;ADR:Juan Florez 129;;')

%% Para utilizar dentro de un campo DocumentoFórmula
%% Impresión datos agente y empresa como tarjeta de visita
ResultadoNada({
 %% Tarjeta de visita VCard
 %% En las tarjetas VCARD la "," debe precederse de una "\"
 %% Obtención datos
 agente_nombre = TxReemplaza(AgenteNombre; ","; "\,");
 agente_teléfono = TxReemplaza(
  ConsultaCampo("select tel from agente where codigo = :codigo"; #(AgenteCódigo));
  ","; "\,"
 );
 agente_correo = ConsultaCampo("select email from agente where codigo = :codigo"; #(AgenteCódigo));
 empresa_nombre = TxReemplaza(EmpresaNombre; ","; "\,");
 empresa_url = "http://www.distritok.com";
 empresa_dirección = TxReemplaza(EmpresaDirección; ","; "\,")
  + " " + EmpresaCódigoPostal + " " + TxReemplaza(EmpresaLocalidad; ","; "\,");
 %% Composicón texto a codificar
 vcard = "";
 vcard = "BEGIN:VCARD" + Intro;
 vcard = vcard + "N:" + agente_nombre + Intro;
 vcard = vcard + "ORG:" + empresa_nombre + Intro;
 vcard = vcard + "TEL:" + agente_teléfono + Intro;
 vcard = vcard + "URL:" + empresa_url + Intro;
 vcard = vcard + "EMAIL:" + agente_correo + Intro;
 vcard = vcard + "ADR:" + empresadirección + Intro;
 vcard = vcard + "END:VCARD";
});
QRCode(vcard);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.46. GS1_128

Descripción:

El código GS1-128, antes llamado EAN-128, es una aplicación estándar de GS1 para la transmisión de información entre los agentes de la cadena de suministro bajo las especificaciones del código de barras Code 128. Inicialmente se denominaba UCC/EAN-128, no obstante, desde la unión de UCC y EAN en GS1 su nombre oficial es GS1-128.

Es un sistema para la identificación que se utiliza para el entorno logístico y no para el entorno detallista. Así, es ideal para la identificación de cajas y pallets que viajan y se mueven dentro de una cadena.

El código GS1-128 utiliza una serie de Identificadores de Aplicación (IA), que actúan como prefijos, para dar el significado de los datos como fechas de caducidad, números de lote, cantidades, peso y muchos otros atributos que el usuario pudiera necesitar. Estos identificativos permiten clasificar de una forma estandarizada las características del producto que representan.

A la instrucción pueden pasarse varios textos, se representa cada uno con su AI (application identifier) al principio del valor, se separan las diferentes parejas de AI y valor con ";".

Sintaxis: GS1_128(<t>[;<t>; ...])

Tipo de dato devuelto: Texto

Ejemplo:

```
%% Secuencia a pasar (251)123456(17)201214
x = GS1_128("251123456"; "17201214");
%% x valdrá 'Òï9+7MÍ6ìï14,.oÓ'
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.47. FTPEstáConectado

Descripción:
Retorna si está conectado o no al ftp
El parámetro es obligatorio
"ftp" es el valor retornado al llamar a FTPConecta

Sintaxis: FTPEstáConectado(ftp)

Tipo de dato devuelto: Lógico

Ejemplo:
```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

si FTPEstáConectado(ftp) entonces {
  FTPBajarFichero(ftp; 'prueba.txt'; 'C:\Distrito');
  FTPDesconectar(ftp);
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.48. FTPObtenerDirectorioActual

Descripción:
Retorna el directorio actual
El parámetro es obligatorio
"ftp" es el valor retornado al llamar a FTPConecta

Sintaxis: FTPObtenerDirectorioActual(ftp)

Tipo de dato devuelto: Texto

Ejemplo:
```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

ftpdir = FTPObtenerDirectorioActual(ftp);

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.49. FTPExisteFichero

Descripción:
Averigua si el fichero especificado existe o no
Los dos parámetros son obligatorios
"ftp" es el valor retornado al llamar a FTPConecta
Retorna un valor lógico

Sintaxis: FTPExisteFichero(ftp; 'fichero')

Tipo de dato devuelto: Lógico

Ejemplo:
```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

si FTPExisteFichero(ftp; 'prueba.txt') entonces {
  FTPBorrarFichero(ftp; 'prueba.txt');
};

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.50. FTPTamañoFichero

Descripción:
Retorna el tamaño (en bytes) del fichero especificado
Los dos parámetros son obligatorios
"ftp" es el valor retornado al llamar a FTPConecta

Sintaxis: FTPTamañoFichero(ftp; 'fichero')

Tipo de dato devuelto: Número

Ejemplo:
```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

Tamaño = FTPTamañoFichero(ftp; 'prueba.txt');

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.51. FTPObtenerListaFicheros

Descripción:

Averigua información sobre los ficheros y subdirectorios del directorio actual
El primer parámetro es obligatorio, el segundo es opcional
"ftp" es el valor retornado al llamar a FTPConecta
"máscara" es opcional
Si no se indica o está vacía, retorna todo el contenido del directorio
Si se indica, retorna información sólo sobre ese fichero/directorio
*También se puede indicar una máscara del tipo *.txt o similar, aunque no es recomendado porque MLS no lo soporta (es mejor filtrar en el script)*
Retorna una lista de registros, cada uno con los siguientes campos:
Nombre
Tamaño (valor de tipo numérico, en bytes)
FechaModificación (valor de tipo fecha-hora, se puede convertir a tipo texto usando EnTexto)

Sintaxis: FTPObtenerListaFicheros(ftp; 'máscara')

Tipo de dato devuelto: Lista

Ejemplo:

```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

lista = FTPObtenerListaFicheros(ftp);

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.52. FTPConectar

Descripción:

Se conecta al ftp especificado y retorna un objeto que se utilizará con las otras funciones FTP.

Los tres primeros parámetros son obligatorios, el cuarto es opcional
"parametros" es un registro que permite especificar diversas opciones:
Puerto (numérico): por defecto 21
Binario (lógico): por defecto cierto (lo normal es dejarlo así)
Pasivo (lógico): por defecto cierto salvo que en el ini se especifique lo contrario
MLS (lógico): por defecto cierto salvo que en el ini se especifique lo contrario
MáscaraTodo: por defecto vacío salvo que en el ini se especifique lo contrario

Sintaxis: FTPConectar('host'; 'usuario'; 'contraseña'; parametros)

Tipo de dato devuelto: Objeto

Ejemplo:

```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

FTPBajarFichero(ftp; 'prueba.txt'; 'C:\Distrito');

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.53. FTPBajarFichero

Descripción:

Baja un fichero del ftp
Los dos primeros parámetros son obligatorios, el tercero es opcional
"ftp" es el valor retornado al llamar a FTPConecta
"ficheroOrigenFtp" es el fichero remoto que queremos descargar (no puede incluir ruta, utiliza la actual en el ftp)
"ficheroDestinoLocal" es un parámetro opcional
Si se incluye, indica el fichero local donde se grabará la descarga (puede y debe incluir ruta)
Si sólo se le pasa una ruta (el valor corresponde a un directorio), se grabará en esa ruta manteniendo el nombre del fichero remoto.
Si no se incluye, la función retorna un valor binario con el contenido de la descarga (que no se habrá grabado a disco)

Sintaxis: FTPBajarFichero(ftp; ficheroOrigenFtp; ficheroDestinoLocal)

Tipo de dato devuelto: Binario

Ejemplo:

```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

FTPBajarFichero(ftp; 'prueba.txt'; 'C:\Distrito');

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.54. CalcularDescuentos

Descripción:
Retorna el descuento numérico equivalente a los descuentos en texto separados por ';' que se le pasen como parámetro.

Sintaxis: CalcularDescuentos('descuentos')

Tipo de dato devuelto: Número

Ejemplo:
x = CalcularDescuentos('10;5'); %% x valdrá 14,5

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.55. AbrirOpciónVentana

Descripción:
Abre una opción del menú de la ventana en la que estamos, igual que si se hubiera pulsado sobre ella en el menú.
El nombre de la ventana puede verse en el editor de menú al pulsar en el icono redondo azul a la derecha de "Opción menú".
La ejecución del script continúa una vez abierta la ventana.

Sólo es válida para las ventanas que tienen opciones específicas (artículos, documentos, etc.)

Si se llama desde un script y se abre una ventana modal, el script detiene su ejecución hasta que se cierra la ventana abierta por esta opción, si no es modal el script continua justo después de abrir la ventana.

Sintaxis: AbrirOpciónVentana(<acción>)

Tipo de dato devuelto: Nada

Ejemplo:
AbrirOpciónVentana("ActionImprimirCatalogos");

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.56. ObtenerFormato

Descripción:
Obtiene la ruta de un formato de la tabla catálogo.
Parámetros:
clase: Obligatorio. Clase del formato. Valores posibles: 0 - Empresa, 1 - Usuario, 2 - Cliente, 3 - Proveedor, 4 - Artículo. Si no existe formato en la clase pasada se mira en las siguientes siguiendo este orden: artículo -> cliente/proveedor -> usuario -> empresa.
tipo: Obligatorio. Tipo de formato. Valores posibles: 0 - Impresión documento, 66 - Impresión documento web
subtipo: Para formatos de impresión es el tipo de documento. Valores posibles:

0 presupuesto venta, 1 pedido venta, 2 albaran venta, 3 factura venta, 4 ticket venta
10 presupuesto compra, 11 pedido compra, 12 albaran compra, 13 factura compra
21 movimiento almacén, 31 recuento
41 artículos fabricados
51 certificación

entidad: Nombre de usuario o código de cliente o código de proveedor o código de artículo, depende de lo indicado en el campo "clase".

Sintaxis: ObtenerFormato(clase; tipo; subtipo[; 'entidad'])

Tipo de dato devuelto: Texto

Ejemplo:
ObtenerFormato(0; 0; 3)

ObtenerFormato(1; 0; 4; UsuarioActual); %% Formato ticket usuario
ObtenerFormato(0; 0; 4) %% Formato ticket empresa

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.57. Leer

Descripción:

Permite obtener un dato de una tabla o de un campo de un formulario.
Para acceder a un campo de un formulario en pantalla debe usarse "Actual." antes del nombre del campo. En este caso debe usarse el nombre del campo para el formulario, no tiene porque coincidir con el nombre del campo en la tabla, aunque la mayoría si lo hacen.
En "posición" (tipo texto) puede pasarse "" para refererirse a la línea actual, siempre debe usarse con "Actual." en el nombre del campo.
Si se pasa un número (positivo o negativo) se refiere a la posterior o anterior a la actual.
Cuando se usa un campo de una tabla debe indicarse en posición el mismo texto que se incluiría en una cláusula where de una sentencia sql.
Puede usarse "alias" para referirse a una conexión a otra base de datos, sobre la cual se ejecutará la instrucción.

El tipo de dato devuelto dependerá del campo leido.

Campos especiales en DOCLIN a usar con "Actual.":
CuantasLíneas: Número de líneas en el documento actual
NúmeroLínea: Número de línea de la línea actual. Empezando a contar en 1.

Sintaxis: Leer(<tabla>; <campo>; <posición> [; <alias>])

Tipo de dato devuelto: Variable

Ejemplo:
```
Líneas = Leer("DOCLIN"; "Actual.CuantasLíneas"; "");
Línea_actual = Leer("DOCLIN"; "Actual.NúmeroLínea"; "");

cod_cli = Leer("DOCCAB"; "Actual.Codcliente"; "");
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

Campos especiales para las líneas de documentos:

Nombrearticulo: Nombre del artículo.
Codarticulo: Código del artículo.
Cantidadreal: Cantidad real del artículo(compuestos).
Coddocumentoorigen: Código del documento origen.
Coddocumento: Código del documento.
Codllinea: Código de la línea.
Nivel: Nivel del árbol.
Cantidad: Cantidad del artículo.
Coste: Precio de coste del artículo.
Precio: Precio del artículo.
Porcentajecomision: Porcentaje de comisión.
Descuentos: Descuentos en forma de cadena.

Iva: Porcentaje de iva.
Recequiv: Porcentaje de recargo de equivalencia.
Irpf: Porcentaje de irpf.
Beneficio: Beneficio.
Porcbeneficio: Porcentaje de beneficio.
Importe: Importe.
Iiprecio: Precio iva incluido.
Iiimporte: Importe iva incluido.
Costepuntoverde: Coste punto verde.
Puntoverde: Punto verde del artículo.
Codclienteorigen: Código del cliente origen.
Tipoorigen: Tipo de documento origen
Serieorigen: Serie del documento origen.
Numeroorigen: Número del documento origen.
Fechaorigen: Fecha del documento origen.
Cantidadorigen: Cantidad de la línea de origen.
Cantidadorigenreal: Cantidad real de la línea de origen(compuestos).
Referenciaproveedor: Referencia del proveedor.
Codbarras: Código de barras.
Nombreoriginal: Nombre original del artículo.
Unidadarticulo: Unidad del artículo.
Unidaddecarticulo: Decimales unidad del artículo.
Preciodecarticulo: Decimales de precio del artículo.

CuantasLíneas: Número de líneas del documento actual.
NúmeroLínea: Posición de la línea actual del documento (se empieza a contar en 1)

15.58. Escribir

Descripción:

Permite modificar un dato de una tabla o de un campo de un formulario.
Para acceder a un campo de un formulario en pantalla debe usarse "Actual." antes del nombre del campo. En este caso debe usarse el nombre del campo para el formulario, no tiene porque coincidir con el nombre del campo en la tabla, aunque la mayoría si lo hacen.
Cuando se usa un campo de una tabla el registro sobre el que se escribe debe existir. Es equivalente a un "update" del campo pasado.
El "valor" a escribir siempre es de tipo texto, aunque se haga sobre un campo de tipo numérico.
En "posición" (tipo texto) puede pasarse "" para refererirse a la línea actual, siempre debe usarse con "Actual." en el nombre del campo.
Si se pasa un número (positivo o negativo) se refiere a la posterior o anterior a la actual.
Cuando se usa un campo de una tabla debe indicarse en posición el mismo texto que se incluiría en una cláusula where de una sentencia sql.
Puede usarse "alias" para referirse a una conexión a otra base de datos, sobre la cual se ejecutará la instrucción.

Sintaxis: *Escribir(<tabla>; <campo>; <valor>; <posición> [; <alias>])*

Tipo de dato devuelto: Nada

Ejemplo:

```
cant = Leer("DOCLIN"; "Actual.Cantidad"; "");
cant = cant * 10;
Escribir("DOCLIN"; "Actual.Cantidad"; Formatea("0.#########"; cant); "");
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

Campos especiales para las líneas de documentos:

Nombrearticulo: Nombre del artículo.
Codarticulo: Código del artículo.
Cantidad: Cantidad del artículo.
Coste: Precio de coste del artículo.
Precio: Precio del artículo.
Porcentajecomision: Porcentaje de comisión.
Descuentos: Descuentos en forma de cadena.
Iva: Porcentaje de iva.
Recequiv: Porcentaje de recargo de equivalencia.
Irpf: Porcentaje de irpf.
Costepuntoverde: Coste punto verde.
Referenciaproveedor: Referencia del proveedor.

15.59. RenombrarFichero

Descripción:
Cambia el nombre del fichero "nombre_origen" por "nombre_destino".
Si no existe el archivo de origen o existe el archivo de destino se muestra un error y se detiene el script.

Sintaxis: RenombrarFichero(nombre_origen; nombre_destino);

Tipo de dato devuelto: Nada

Ejemplo:
```
fichero_origen = "c:\distrito\Dibujo.bmp"
fichero_destino = "c:\distrito\Dibujo Modificado.bmp";
si (ExisteFichero(fichero_origen) y no ExisteFichero(fichero_destino)) entonces {
  RenombrarFichero(fichero_origen; fichero_destino);
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.60. RecalcularImportesConsultaDocumento

Descripción:

Recalcula y graba los campos ImporteXXX de doccab del documento indicado.
Conviene llamarla después de crear un documento con CrearDocumento()

Sintaxis: RecalcularImportesConsultaDocumento(codigo_documento)

Tipo de dato devuelto: Nada

Ejemplo:

```
codDoc = CrearDocumento(#[tipo: 2; codcliente: 2; codalmacen: 1; codformapago: 1]);
EjecutarSQL(
  'insert into doclin (coddocumento, codigo, nivel, codarticulo, cantidad, precio, coste, tipoiva, codalmacen) values
(:coddocumento, :codigo,
 :nivel, :codarticulo, :cantidad, :precio, :coste, :tipoiva, :codalmacen)';
  #[coddocumento: codDoc; codigo: 10; nivel: 1; codarticulo: 'alfombrilla'; cantidad: 1,2; precio: 10; coste: 9;
tipoiva: 21; codalmacen: 1]
);
RecalcularImportesConsultaDocumento(codDoc);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.61. BinCrea

Descripción:
Devuelve un objeto binario vacío.

Sintaxis: BinCrea

Tipo de dato devuelto: Binario

Ejemplo:
foto = BinCrea;
t = Tipo(foto); %% t valdrá TBinario

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.62. BinTipo

Descripción:
Devuelve la extensión del fichero correspondiente al contenido del binario, si no es capaz de reconocerla devuelve un texto vacío.
Actualmente reconoce las imágenes de tipo: gif, png, jpg, art, bmp e ico.

Sintaxis: BinTipo(<bin>)

Tipo de dato devuelto: Texto

Ejemplo:
```
foto = BinCrea;
url = "http://imagenes.distritok.com/distritok/logocabecera.min.jpg";
BinDescargaHttp(foto; url);
ext_bin = BinTipo(foto); %% ext_bin valdrá "jpg"
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.63. BinBorra

Descripción:
Borra el contenido de la variable binaria pasada.
Cuando se asigna a una variable binaria un nuevo contenido se libera automáticamente la memoria ocupada por el objeto anterior.

Sintaxis: BinBorra(<bin>)

Tipo de dato devuelto: Nada

Ejemplo:
```
foto = ConsultaCampo("select first 1 foto from fotograf order by codarticulo");
BinBorra(foto);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.64. BinTamaño

Descripción:
Tamaño en bytes de una variable de tipo binario

Sintaxis: BinTamaño(<bin>)

Tipo de dato devuelto: Número

Ejemplo:
```
foto = ConsultaCampo("select first 1 foto from fotograf order by codarticulo");
bytes = BinTamaño(foto);
kb = bytes / 1024;
mb = kb / 1024;
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.65. BinGraba

Descripción:
Graba en un fichero un objeto binario.
Con la función BinTipo puede saberse si el objeto binario se trata de una imagen y la extensión del fichero.
Serviría para exportar las fotografías.

Sintaxis: BinGraba(<bin>; <ruta>)

Tipo de dato devuelto: Nada

Ejemplo:
```
foto = ConsultaCampo("select first 1 foto from fotograf order by codarticulo");
BinGraba(foto; "C:\Distrito\foto.jpg");
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.66. BinCarga

Descripción:
Carga el fichero indicado en la variable. La variable debe existir antes de llamar a la instrucción y ser de tipo binario.

Sintaxis: BinCarga(<binario>; <fichero>);

Tipo de dato devuelto: Nada

Ejemplo:
```
foto = BinCrea;
BinCarga(foto; "C:\Temp\Foto.jpg");
EjecutarSQL("
  insert into fotograf
    (codarticulo, codigo, foto, orden)
    values (:codarticulo, :codigo, :foto, :orden)";
  #('REF001'; 1; foto; 1)
);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.67. BinDescargaHttp

Descripción:

Descarga el contenido de la url pasada como parámetro y lo almacena en la variable de tipo binario indicada.
La variable debe existir.
El parámetro <rcr> es un valor lógico opcional, si no se indica vale falso. Si no se indica el parámetro o es falso, si se produce un error intentando descargar la imagen se finaliza la ejecución del script. Si el valor es cierto se devuelve el código de error http o socket, si se produce otro tipo de error seguirá deteniéndose el script.
Los códigos de respuesta http son de 3 dígitos, los valores más comunes son: 200 (éxito), 404 (no existe) y 401/403 (sin permiso).
Los códigos de error de socket son mayores de 10.000. Algunos típicos son: 11001 (host desconocido), 10061 (puerto no accesible, porque no hay nadie escuchando o un firewall impide el paso)
Se soportan los protocolos http y https.

Sintaxis: BinDescargaHttp(<bin>; <url> [; <rcr>])

Tipo de dato devuelto: txt

Ejemplo:

```
foto = BinCrea;
url = "http://imagenes.distritok.com/distritok/logocabecera.min.jpg";
BinDescargaHttp(foto; url);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.68. ImprimirTraspaso

Descripción:

Imprime el traspaso indicado.
Parámetros:
fecha_apunte: Obligatorio. Fecha del traspaso a imprimir
codigo_apunte: Obligatorio. Código del apunte con el traspaso.
impresion_directa: Opcional.Cierto o falso. Por defecto: falso. Se previsualiza la impresión o se manda directamente a la impresora.
botones_grandes: Opcional. Cierto o falso. Por defecto: falso. Botones grandes en la previsualización de la impresión. Si no se previsualiza con el parámetro anterior da igual lo que se ponga. Si se habilitan los botones grandes no podrá exportarse a Word, Excel, etc.
mostrar_modal: Opcional. Cierto o falso. Por defecto: falso. No deja hacer nada más con el programa hasta que se cierra la ventana de previsualización. Como en el anterior parámetro, si no se previsualiza no tiene efecto.

Sintaxis: ImprimirTraspaso(fecha_apunte; codigo_apunte[; impresion_directa; botones_grandes; mostrar_modal])

Tipo de dato devuelto: Nada

Ejemplo:

```
fecha = FechaHoy;
codApunte = CrearApunteCaja(#[fecha: fecha; tipo: 7; estado: 'T'; descripcion: 'prueba'; importe: 1234]);
ImprimirTraspaso(fecha; codApunte);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.69. EnviarSMS

Descripción:

Antes de usar esta instrucción debe configurarse en las opciones de configuración de la aplicación los "SMS".
El único parámetro obligatorio es "movil_destinatario". Si quiere no pasarse un parámetro

Parámetros:

movil_destinatario: Móvil al que enviar el sms.
texto: Texto del SMS. Si no se indica se usará el texto por defecto configurado en las opciones de la aplicación. No hay control del tamaño, se supone que si se pasa de 150 caracteres se enviarán tantos sms como sean necesarios.
remitente: En función de la plataforma se usará o no. Si es "AfilNet" es lo que recibirá el destinatario como remitente del mensaje, si se trata de un número dependerá de si el cliente lo tiene en su agenda para que le muestre el nombre. Puede enviarse un nombre y el destinatario lo recibirá así, como si fuera un número de teléfono. Si se indica "" se usará el de la configuración.
código_plataforma: Es el código que figura en la configuración de la aplicación. Si no se indica se usará la configurada por defecto.

Retorna el mensaje de error de la plataforma si se ha producido alguno y si no un texto vacío.

Sintaxis: EnviarSMS('movil_destinatario'; 'texto'; 'remitente'; codigo_plataforma)

Tipo de dato devuelto: Texto

Ejemplo:

```
TxtError = EnviarSMS("678123456"; "Prueba SMS"; ""; 1);
si (TxtError <> "") entonces {
  Mensaje("Error al enviar sms: " + TxtError);
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.70. BorrarFichero

Descripción:
Si no se puede borrar el fichero se genera un error.
Eliminación archivos.

Sintaxis: BorrarFichero(<t>)

Tipo de dato devuelto: Nada

Ejemplo:
BorrarFichero("c:\distrito\temp\fichero_temporal.txt");

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.71. BorrarDirectorio

Descripción:
Borra el directorio pasado como ruta siempre y cuando esté vacío. Si no lo está o no se tienen permisos se genera un error y se interrumpe el script.

Sintaxis: BorrarDirectorio(ruta)

Tipo de dato devuelto: Nada

Ejemplo:
BorrarDirectorio("c:\distrito\pyme\salida informes");

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.72. BorrarLog

Descripción:

Borra del log aquellos registros que se pueden borrar sin afectar a los módulos que los utilizan (pda, sql commerce y delegaciones).
Hace lo mismo que la opción en el menú utilidades -> herramientas de administrador -> borrar log.
Todos los parámetros son opcionales y de tipo texto.
Devuelve el número de registros borrados en la tabla DK$OperationLog

Sintaxis: BorrarLog(<desde_fecha>; <hasta_fecha>; <desde_usuario>; <hasta_usuario>)

Tipo de dato devuelto: Número

Ejemplo:

```
%% Eliminación del log de todos los movimientos anteriores a 45 días
%% Después de esta tarea debe programarse una compactación de la base de datos
%% para recuperar el espacio liberado con esta instrucción.
desde_fecha = "01/01/1900";
hasta_fecha = FechaEnTexto(FechaEnNúmero(FechaHoy) - 45);
BorrarLog(desde_fecha; hasta_fecha);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.73. Code128

Descripción:
Devuelve <t> de forma que al usarlo con una fuente de códigos de barras code128 sea legible para los lectores de códigos de barras.
Permite caracteres con código ASCII entre 0 y 127. Se va cambiando la codificación entre A,B y C según se necesita.
También se conoce como code128 auto switch.
Hay que tener instalada en el ordenador que imprima una fuente "Code 128". Puede descargarse de: http:www.distritok.com/descargas/fuentes.exe

Sintaxis: Code128(<t>)

Tipo de dato devuelto: Texto

Ejemplo:
```
x = Code128('PJJ123C');
%% x valdrá'ÑPJJÌ,Í3CpÓ'
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.74. Code128A

Descripción:
Devuelve <t> de forma que al usarlo con una fuente de códigos de barras code128 sea legible para los lectores de códigos de barras.
Permite caracteres con código ASCII entre 0 y 95.
Hay que tener instalada en el ordenador que imprima una fuente "Code 128". Puede descargarse de: http:www.distritok.com/descargas/fuentes.exe

Sintaxis: Code128A(<t>)

Tipo de dato devuelto: Texto

Ejemplo:
x = Code128A('PJJ123C');
%% x valdrá 'ÐPJJ123CVÓ'

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.75. Code128B

Descripción:
Devuelve <t> de forma que al usarlo con una fuente de códigos de barras code128 sea legible para los lectores de códigos de barras.
Permite caracteres con código ASCII entre 32 y 127.
Hay que tener instalada en el ordenador que imprima una fuente "Code 128". Puede descargarse de:
http:www.distritok.com/descargas/fuentes.exe

Sintaxis: Code128B(<t>)

Tipo de dato devuelto: Texto

Ejemplo:
```
x = Code128B('PJJ123C');
%% x valdrá 'ÑPJJ123CWÓ'
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.76. Code128C

Descripción:
Devuelve <t> de forma que al usarlo con una fuente de códigos de barras code128 sea legible para los lectores de códigos de barras.
Permite sólamente números.
Los códigos de barras generados serán más compactos que con las otras variantes de code128
Hay que tener instalada en el ordenador que imprima una fuente "Code 128". Puede descargarse de:
http:www.distritok.com/descargas/fuentes.exe

Sintaxis: Code128C(<t>)

Tipo de dato devuelto: Texto

Ejemplo:
```
x = Code128C('123456');
%% x valdrá 'Ò,BXLÓ'
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.77. CodigoUsuarioActual

Descripción:
Código del usuario actual

Sintaxis: CodigoUsuarioActual

Tipo de dato devuelto: Número

Ejemplo:
```
si (CódigoUsuarioActual <> 19) entonces {
  Error("No está autorizado a usar esta opción.");
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.78. ConfiguraADO

Descripción:

Permite configurar una conexión ADO.Se muestra un cuadro de diálogo para configurarla. Si se ha pasado el parámetro con una cadena de conexión ya se muestra la ventana configurada con los parámetros de esa conexión.
Si se cancela se devuelve un texto vacío si no se pasó ninguna cadena de conexión, si se hizo devuelve esa misma cadena. Si acepta sin cambiar nada es lo mismo que darle a cancelar.

Sintaxis: ConfiguraADO(<cadena_conexión>)

Tipo de dato devuelto: Texto

Ejemplo:

```
ruta = "C:\Distrito";
fichero_mdb = ComponerRuta(ruta; 'Prueba.mdb');
cadena_conexión = 'Provider=Microsoft.Jet.OLEDB.4.0;Data Source=' + fichero_mdb;
cadena_conexión = ConfiguraADO(cadena_conexion);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.79. ConectaADO

Descripción:

Crea una conexión con un objeto ADO.
<alias> es el nombre que se utilizará en las instrucciones de acceso a datos para usar esa conexión en lugar de la empresa abierta en el programa.
Si se intenta lanzar una conexión con un alias que ya existe no se genera un error, primero se desconecta la existente y luego se crea la indicada.

Sintaxis: ConectaADO(<alias>; <cadena_conexión>)

Tipo de dato devuelto: Nada

Ejemplo:

```
ruta = 'C:\Distrito';
fichero_mdb = ComponerRuta(ruta; 'Prueba.mdb');
si no ExisteFichero(fichero_mdb) entonces {
  CreaADO('Provider=Microsoft.Jet.OLEDB.4.0;Data Source=' + fichero_mdb);
  ConectaADO('ado'; 'Provider=Microsoft.Jet.OLEDB.4.0;Data Source=' + fichero_mdb);
  EjecutarSQL('create table tabla1 ( codigo integer not null primary key, nombre text )'; 'ado');
  Desconecta('ado');
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.80. ComponerRuta

Descripción:
Se añade una "\" al final de cada ruta pasada si no la tiene ya, el último parámetro sólo se concatena al final.

Sintaxis: ComponerRuta(ruta1[; ruta2; ...]; fichero)

Tipo de dato devuelto: Texto

Ejemplo:

```
ComponerRuta(ObtenerInformacion('RutaTrabajo'); 'fichero.txt');

GenerarInforme(
  ComponerRuta(ObtenerInformación("RutaTrabajo"); "Informes"; "Informe de documentos de venta" ;
"Documentos de venta por fecha y
cliente.IDK");
  "C:\temp";
  #(
    #[nombre: 'R1' valor: '2' %albarán%];
    #[nombre: 'DocumentoDeVenta.fecha' valor: restriccionFecha]
  )
);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.81. CrearDirectorio

Descripción:
Crea el directorio pasado como ruta. Si ya existe o no se tienen permisos se genera un error que interrumpe el script.

Sintaxis: CrearDirectorio(ruta)

Tipo de dato devuelto: Nada

Ejemplo:
```
ruta_aplicación = ObtenerInformación("RutaTrabajo");
ruta_salida_informes = ComponerRuta(ruta_aplicación; "Salida informes");
si No ExisteDirectorio(ruta_salida_informes) entonces {
  CrearDirectorio(ruta_salida_informes);
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.82. CreaADO

Descripción:
Permite crear un objeto ADO

Sintaxis: CreaADO(<cadena_conexión>)

Tipo de dato devuelto: Nada

Ejemplo:
```
ruta = 'C:\Distrito';
fichero_mdb = ComponerRuta(ruta; 'Prueba.mdb');
si no ExisteFichero(fichero_mdb) entonces {
  CreaADO('Provider=Microsoft.Jet.OLEDB.4.0;Data Source=' + fichero_mdb);
  ConectaADO('ado'; 'Provider=Microsoft.Jet.OLEDB.4.0;Data Source=' + fichero_mdb);
  EjecutarSQL('create table tabla1 ( codigo integer not null primary key, nombre text )'; 'ado');
  Desconecta('ado');
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.83. PreguntaMemo

Descripción:
Detiene la ejecución del script mientras solicita por pantalla la entrada de un texto multilínea.
Si se cancela se cancela la ejecución del script.

Sintaxis: PreguntaMemo('título_ventana'; 'texto' [; valor])

Tipo de dato devuelto: Texto

Ejemplo:
texto = preguntamemo("Título ventana"; "Titulo del campo$nPuede ser de mas de una línea"; "Texto por defecto");
Mensaje(Texto);

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.84. UsuarioActual

Descripción:
Devuelve el nombre del usuario que ha iniciado la sesión en el programa.
Lo devuelve tal y como está guardado en la tabla acc_usr, no como se ha escrito en la ventana de solicitud de usuario y contraseña.

Sintaxis: UsuarioActual

Tipo de dato devuelto: Texto

Ejemplo:
```
si (UsuarioActual <> 'Andrea') entonces {
  Error("No está autorizado a usar esta opción.");
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.85. HttpGet

Descripción:

Permite enviar una petición a un servidor web a través del método get.

Parámetros:
url: Obligatorio. Texto. Dirección a la que enviar el get indicado en parámetros. Debe indicarse el protocolo a utilizar, se soporta http y https.

Opciones: Opcional. Es un registro que permite especificar diversas opciones:
- RetornarCodigoRespuesta (lógico): por defecto falso
- RetornarBinario (lógico): indica si retorna un binario o un texto (por defecto falso para que retorne texto)
- Usuario (texto): por defecto sin usuario
- Clave (texto): por defecto sin contraseña
- ContentType (texto): el tipo de contenido que se envía en el post
- Codificación (texto):
· Convierte el texto que se está enviando a la codificación especificada y fija el CharSet correspondiente
· Sólo se debe especificar si se está enviando un texto
· Posibles valores 'ansi', 'utf8', 'utf16' (por defecto ansi)
- CharSet (texto): codificación del contenido que se envía (por defecto ISO-8859-1); sólo se debería especificar si se está enviando un binario
- Accept (texto): el tipo de contenido que se acepta en la respuesta
- Agent (texto): el agente que hace la petición (normalmente la identificación del navegador)
- Timeout (numérico): cuántos milisegundos espera por la respuesta (por defecto espera indefinidamente)
- Cacheable (lógico): por defecto cierto
- Cabeceras (registro): permite añadir cabeceras a la petición http

Si tiene éxito, retorna un registro con diversos campos:
- Código: el código de respuesta http (normalmente 200)
- Respuesta: el resultado de la respuesta (un texto o un binario según se haya especificado en las opciones), salvo que dé error http, que entonces será el texto del error (normalmente html)
- Cabeceras: los campos más relevantes de la cabecera de respuesta http (Date, Expires, LastModified, Server, ContentType, CharSet)

Si hay error salta el error salvo si retornarCodigoRespuesta es cierto, que indica que para los errores http y de socket retorne su código de error (los demás errores seguirán saltando)
- Los códigos de respuesta http tiene 3 dígitos y los más habituales son 200 (éxito), 404 (no existe) y 401/403 (sin permiso).
- Los códigos de error de socket son mayores de 10000 y algunos típicos son 11001 (host desconocido), 10061 (puerto no accesible, porque no hay nadie escuchando o un firewall impide el paso)

Sintaxis: HttpGet('url' [, opciones])

Tipo de dato devuelto: registro

Ejemplo:

```
b = BinCrea;
resultado = BinDescargaHttp(b; 'http://localhost');
MensajeHtml(b; resultado);


resultado = HttpGet(
  'http://localhost/test-headers.php';
  #[
    retornarCodigoRespuesta: cierto;
    cabeceras: #[Authorization: 'prueba prueba']
  ]
);
Mensaje(resultado);
```

resultado.Código

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.86. HttpPost

Descripción:

Permite enviar una petición a un servidor web a través del método post.

Parámetros:
url: Obligatorio. Texto. Dirección a la que enviar el post indicado en parámetros. Debe indicarse el protocolo a utilizar, se soporta http y https.

Contenido: Obligatorio. Indica el contenido que envía el post:
- Si es un registro especifica los campos del formulario y su valor
- Si es un binario se subirá directamente su contenido

Opciones: Opcional. Es un registro que permite especificar diversas opciones:
- RetornarCodigoRespuesta (lógico): por defecto falso
- RetornarBinario (lógico): indica si retorna un binario o un texto (por defecto falso para que retorne texto)
- Usuario (texto): por defecto sin usuario
- Clave (texto): por defecto sin contraseña
- ContentType (texto): el tipo de contenido que se envía en el post
- Codificación (texto):
· Convierte el texto que se está enviando a la codificación especificada y fija el CharSet correspondiente
· Sólo se debe especificar si se está enviando un texto
· Posibles valores 'ansi', 'utf8', 'utf16' (por defecto ansi)
- CharSet (texto): codificación del contenido que se envía (por defecto ISO-8859-1); sólo se debería especificar si se está enviando un binario
- Accept (texto): el tipo de contenido que se acepta en la respuesta
- Agent (texto): el agente que hace la petición (normalmente la identificación del navegador)
- Timeout (numérico): cuántos milisegundos espera por la respuesta (por defecto espera indefinidamente)
- Cacheable (lógico): por defecto cierto
- Cabeceras (registro): permite añadir cabeceras a la petición http

Nota: Por compatibilidad con versiones anteriores también se puede indicar un valor lógico directamente en el tercer parámetro, que indica el valor para la opción retornarCodigoRespuesta
Si no hay tercer parámetero, se considera que retornarCodigoRespuesta es falso.
Si se especifica retornarCódigoRespuesta: Opcional. Lógico. Por defecto falso.
- Si es cierto se devuelve un registro con dos campos: código (número) código http de la respuesta, respuesta (texto) texto devuelto por el servidor.
- Si es falso devolverá lo mismo si no se produce ningún error, si se produce se detendrá la ejecución del script y se mostrará el mensaje de error.

Si tiene éxito, retorna un registro con diversos campos:
- Código: el código de respuesta http (normalmente 200)
- Respuesta: el resultado de la respuesta (un texto o un binario según se haya especificado en las opciones), salvo que dé error http, que entonces será el texto del error (normalmente html)
- Cabeceras: los campos más relevantes de la cabecera de respuesta http (Date, Expires, LastModified, Server, ContentType, CharSet)

Si hay error salta el error salvo si retornarCodigoRespuesta es cierto, que indica que para los errores http y de socket retorne su código de error (los demás errores seguirán saltando)
- Los códigos de respuesta http tiene 3 dígitos y los más habituales son 200 (éxito), 404 (no existe) y 401/403 (sin permiso).
- Los códigos de error de socket son mayores de 10000 y algunos típicos son 11001 (host desconocido), 10061 (puerto no accesible, porque no hay nadie escuchando o un firewall impide el paso).

Sintaxis: HttpPost('url', contenido [, opciones])

Tipo de dato devuelto: registro

Ejemplo:

```
resultado = HttpPost(
  'http://localhost/test-post.php';
  #[campo: 'valor']
);
Mensaje(resultado);


resultado = HttpPost(
  'http://localhost/demos/imagen.png';
  #[];
  #[RetornarBinario: cierto]
);
no hay nadie escuchando o un firewall impide el paso) BinGraba(resultado.Respuesta; 'C:\imagen.png');


b = BinCrea;
BinCarga(b; 'C:\imagen.png');
resultado = HttpPost(
  'http://localhost/test-post.php';
  b;
  #[]
);
Mensaje(resultado);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.87. ExtraerRuta

Descripción:
Permite obtener la ruta de un fichero.

Sintaxis: ExtraerRuta(<t>)

Tipo de dato devuelto: Texto

Ejemplo:
```
x = ExtraerRuta("\\servidor\Distrito\Pyme\Pyme.ini");
%% x valdrá "\\servidor\Distrito\Pyme"
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.88. ExisteDirectorio

Descripción:
Devuelve cierto si existe el directorio pasado como ruta, sino devuelve falso. Es indiferente si se indica la "\" al final o no.

Sintaxis: ExisteDirectorio(ruta)

Tipo de dato devuelto: Lógico

Ejemplo:
```
ruta_aplicación = ObtenerInformación("RutaTrabajo");
ruta_salida_informes = ComponerRuta(ruta_aplicación; "Salida informes");
si No ExisteDirectorio(ruta_salida_informes) entonces {
  CrearDirectorio(ruta_salida_informes);
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.89. ExisteFichero

Descripción:
Devuelve cierto o falso si el fichero existe o no.

Sintaxis: ExisteFichero(fichero)

Tipo de dato devuelto: Lógico

Ejemplo:
```
x = ExisteFichero('C:\Distrito\Compart\DK_GLOBAL.FDB');
%% x valdrá cierto
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.90. CódigoCarácter

Descripción:
Retorna el código unicode del primer carácter del texto.

Los caracteres unicode del 0 al 127 corresponden a los ascii.
Los caracteres unicode del 0 al 255 corresponden a los iso-8859-1, que equivalen al win1252 típico de Windows en la mayoría de los casos excepto en los caracteres entre 128 y 159 (p.ej. '€' tiene código 128 en win1252 y 8634 en unicode).
Retornará 0 si el texto está vacío.

Sintaxis: CódigoCarácter('texto')

Tipo de dato devuelto: TNumero

Ejemplo:
```
txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
lAscii = #();
i = 1;
mientras {i <= txTamaño(txt)} haz {
  chr = txTrozo(txt; i; 1);
  lAñade(lAscii; #[Letra: chr;
                   ASCII: CodigoCaracter(chr)]);
  i = i + 1;
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.91. AbrirFichero

Descripción:

AbrirFichero('fichero'; 'acción'; 'visualización'; 'parámetros');
fichero es obligatorio, los demás opcionales
acción: abrir, imprimir, editar, explorar, buscar (si no se indica será abrir)
visualización: normal, maximizar, minimizar, ocultar (si no se indica será normal)
parámetros: si fichero es un programa serán sus parámetros, si no lo es no debe ponerse nada

Si no se especifica "acción" se hará lo mismo que Windows al hacer doble clic sobre el fichero.

Sintaxis: AbrirFichero('fichero'; 'acción'; 'visualización'; 'parámetros')

Tipo de dato devuelto:

Ejemplo:

```
AbrirFichero('C:\documento.pdf'; 'imprimir');
AbrirFichero('C:\programa.exe'; ''; 'minimizar');
AbrirFichero("notepad"; ""; ""; "c:\fichero.txt")
AbrirFichero("http:\\www.google.es");
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.92. MensajeHTML

Descripción:

MensajeHtml('html'; título)
MensajeHtml('html'; parametros)
MensajeHtml(binario; título)
MensajeHtml(binario; parametros)

Muestra en un visor html el texto o binario que se le indique
- El primer parámetro es obligatorio, el segundo es opcional
- Si el segundo parámetro es de un tipo de datos simple (nada, texto, número, lógico, fecha, hora, fechahora), indica el título de la ventana
- Si el segundo parámetro es un registro, permite especificar diversas opciones:
· Título (cualquier tipo de datos simple): título de la ventana (se convierte a texto con la misma conversión que haría EnTexto)
· Tamaño (registro): indica las dimensiones de la ventana:
Ancho (valor de tipo numérico, en pixels)
Alto (valor de tipo numérico, en pixels)
· ModoRestringido (lógico, por defecto cierto): no permite la ejecución de javascript/activex/java y los enlaces los abre en el navegador de internet externo

Sintaxis: MensajeHTML(txHTML | binario [; txTitulo [; parametros]])

Tipo de dato devuelto:

Ejemplo:

```
txHTML =
 '<html>
  <body>
   <p>
   <h1>Hola Mundo</h1>
   </p>
   <script>alert("Hola javascript");</script>
  </body>
 </html>';
MensajeHtml(txHTML;
        #[Título: 'Visor';
          ModoRestringido: falso;
          Tamaño: #[ancho: 300; alto: 200]]);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.93. CopiarFichero

Descripción:
CopiarFichero('ficheroOrigen'; 'ficheroDestino')

Copia el fichero de origen en el de destino
Da error si el fichero de destino ya existe

Sintaxis: CopiarFichero('ficheroOrigen'; 'ficheroDestino')

Tipo de dato devuelto:

Ejemplo:
CopiarFichero('C:\Distrito\Pyme\PYME.EXE'; 'C:\Distrito\Pyme\PYME-copia.EXE');

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.94. FTPDesconectar

Descripción:
Se desconecta del ftp
El parámetro es obligatorio
"ftp" es el valor retornado al llamar a FTPConecta

Sintaxis: FTPDesconectar(ftp)

Tipo de dato devuelto:

Ejemplo:
```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

FTPBajarFichero(ftp; 'prueba.txt'; 'C:\Distrito');

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.95. FTPFijarDirectorioSuperior

Descripción:
Fija como directorio actual el superior al actual
El parámetro es obligatorio
"ftp" es el valor retornado al llamar a FTPConecta

Sintaxis: FTPFijarDirectorioSuperior(ftp)

Tipo de dato devuelto:

Ejemplo:

```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

FTPFijarDirectorioSuperior(ftp);
FTPSubirFichero(ftp; 'prueba.txt');

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.96. FTPBorrarFichero

Descripción:
Borra el fichero especificado
Los dos parámetros son obligatorios
"ftp" es el valor retornado al llamar a FTPConecta

Sintaxis: FTPBorrarFichero(ftp; 'fichero')

Tipo de dato devuelto:

Ejemplo:

```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

si FTPExisteFichero(ftp; 'prueba.txt') entonces {
  FTPBorrarFichero(ftp; 'prueba.txt');
};

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.97. FTPForzarDirectorio

Descripción:

Crea los directorios y subdirectorios de la ruta especificada según sea necesario y se posiciona en ella
Los dos parámetros son obligatorios
"ftp" es el valor retornado al llamar a FTPConecta
"directorio" es una ruta en remoto que indica un directorio con posibles subdirectorios
Se asegura que existe la ruta que se le pasa, creando los directorios que no existan
Se posiciona en ella
La ruta es relativa al directorio actual

Sintaxis: FTPForzarDirectorio(ftp; directorio)

Tipo de dato devuelto:

Ejemplo:

```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

FTPForzarDirectorio(ftp; 'uploaddir/imagenes');
FTPSubirFichero(ftp; 'prueba.txt');

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.98. FTPCrearDirectorio

Descripción:
Crea el directorio especificado
Los dos parámetros son obligatorios
"ftp" es el valor retornado al llamar a FTPConecta

Sintaxis: FTPCrearDirectorio(ftp; 'directorio')

Tipo de dato devuelto:

Ejemplo:
```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

FTPCrearDirectorio(ftp; 'uploaddir');
FTPFijarDirectorioActual(ftp; 'uploaddir')
FTPSubirFichero(ftp; 'prueba.txt');

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.99. FTPBorrarDirectorio

Descripción:
Borra el directorio especificado
Los dos parámetros son obligatorios
"ftp" es el valor retornado al llamar a FTPConecta

Sintaxis: FTPBorrarDirectorio(ftp; 'directorio')

Tipo de dato devuelto:

Ejemplo:
```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

FTPBorrarDirectorio(ftp; 'uploaddir');

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.100. FTPFijarDirectorioActual

Descripción:
Fija como directorio actual el que se le indique
Los dos parámetros son obligatorios
"ftp" es el valor retornado al llamar a FTPConecta

Sintaxis: FTPFijarDirectorioActual(ftp; 'directorio')

Tipo de dato devuelto:

Ejemplo:
```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

FTPFijarDirectorioActual(ftp; 'uploaddir');
FTPSubirFichero(ftp; 'prueba.txt');

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.101. FTPRenombrarFichero

Descripción:
Renombra el fichero especificado
Los tres parámetros son obligatorios
"ftp" es el valor retornado al llamar a FTPConecta

Sintaxis: FTPRenombrarFichero(ftp; 'fichero', 'nuevoNombre')

Tipo de dato devuelto:

Ejemplo:

```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

si FTPExisteFichero(ftp; 'prueba.txt') entonces {
  FTPRenombrarFichero(ftp; 'prueba.txt'; 'prueba.old');
};

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.102. FTPSubirFichero

Descripción:

Sube un fichero o un binario al ftp
Los dos primeros parámetros son obligatorios, el tercero es opcional
"ftp" es el valor retornado al llamar a FTPConecta
"origenLocal" puede ser un binario o un fichero (puede y debe incluir ruta)
"ficheroDestinoFtp" es un parámetro opcional (si origenLocal es binario, entonces sí es obligatorio incluirlo)
Si se incluye, indica el nombre del fichero remoto donde se grabará la subida (no puede incluir ruta, utiliza la actual en el ftp)
Si no se incluye, se grabará manteniendo el nombre del fichero local

Sintaxis: FTPSubirFichero(ftp; origenLocal; ficheroDestinoFtp)

Tipo de dato devuelto:

Ejemplo:

```
ftp = FTPConectar('localhost'; 'user'; 'pwd');

FTPSubirFichero(ftp; 'c:\prueba\prueba.txt'; 'pruebasubida.txt');

FTPDesconectar(ftp);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.103. EnviarMensaje

Descripción:
Permite enviar un mensaje a otro usuario de la aplicación.

Parámetros:
usuario: Nombre del usuario al que enviar el mensaje. Puede ser una lista con los nombres a enviar. Si está en blanco el mensaje se manda al usuario actual.
mensaje: Mensaje que se enviará. No tiene límite de tamaño.

Sintaxis: EnviarMensaje(<usuario>; <mensaje>)

Tipo de dato devuelto:

Ejemplo:
EnviarMensaje(#('usuario1'; 'usuario2'); 'Prueba mensaje');

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.104. BinAñade

Descripción:
Añade el byte al final del binario

Sintaxis: BinAñade(binario; byte)

Tipo de dato devuelto:

Ejemplo:

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.105. BinBorra

Descripción:
Borra todo o parte del contenido del binario.

El primer parámetro es obligatorio, los otros son opcionales.
Si sólo se le pasa un parámetro => borra todo el contenido (vacía el binario).
Si se le pasa inicio pero no longitud => borra el contenido a partir de inicio hasta el final.
Si se le pasan tanto inicio como longitud => borra el contenido a partir de inicio, tantos bytes como indique longitud.
La posición de inicio empieza en 1.

Sintaxis: BinBorra(binario[; inicio[; longitud]])

Tipo de dato devuelto:

Ejemplo:

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones varias

15.106. TrocearTextoImpresión

Descripción:

Trocea el texto en líneas con palabras completas, según la fuente y el ancho especificados.
Sólo funciona en los formatos de impresión y en el generador de informes.
Todos los parámetros son obligatorios;
Texto es el texto a trocear; puede contener múltiples párrafos separados por retornos de carro
Fuente es el número de fuente del formato de impresión
Ancho es el ancho máximo en cm que puede tener cada línea

Retorna una lista de registros, cada uno con los siguientes campos:
Línea (el texto de esa línea)
FinPárrafo (valor de tipo lógico que indica si es la última línea del párrafo)

Sintaxis: TrocearTextoImpresión('texto'; fuente; ancho)

Tipo de dato devuelto: TLista

Ejemplo:

```
ResultadoNada({
 cuadro = CuadroCrea;

 CuadroConfiguracion(cuadro; #[mostrartituloscolumnas: falso; mostrartitulosfilas: falso; bordesinteriores: falso;
celdafuente: 0;
 celdajustificado: 'I']);

 col = CuadroAñadeColumna(cuadro; "col"; #[ancho: 9,5]);

 textos = TrocearTextoImpresión(
  'Es imprescindible que todos los Usuarios de Windows, y en especial el usuario que realiza la instalación, que va
a utilizar la aplicación tengan
 habilitados todos los permisos de acceso a la carpeta de destino y a todas sus subcarpetas.'
  +Cáracter(13) %% cambio de párrafo
  +'De lo contrario, la aplicación no funcionará correctamente. Si tiene un sistema con permisos asignados según
los usuarios, indique esta
 información al administrador de su sistema.';
  0;
  9
 );

 LAplica(textos; {
  texto = textos[params[1]];
  fila = CuadroAñadeFila(cuadro; '');
  celda = CuadroCelda(cuadro; col; fila);
  celda.Valor = texto.Línea;
  si no (texto.FinPárrafo) entonces {
   celda.Justificado = 'J';
  };
 });
});
cuadro;
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión: SI

Tipo de instrucción:

15.107. DatoAdicional

Descripción:

Esta función permite obtener información sobre los datos adicionales de las entidades.
Los parámetros son:
<entidad>: Un texto que indica la entidad de la se desea obtener información
Documento Dato adicional del documento actual
Línea Dato adicional de la línea actual del documento
Artículo Dato adicional del artículo de la línea actual del documento
Cliente Dato adicional del cliente o proveedor del documento actual
Obra Dato adicional de la obra actual
ArtículoObra Dato adicional del artículo de la línea actual de la obra
<formato>: Un texto que indica el formato del resultado. Las macros válidas son:
%n El nombre del dato adicional
%v El valor del dato adicional
<dato>: Un texto opcional que indica el nombre del dato adicional requerido. Si este

Sintaxis: DatoAdicional(<entidad>;<formato>;<dato>)

Tipo de dato devuelto: Texto

Ejemplo:

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión: SI

Tipo de instrucción:

15.108. ArtículoUbicaciónCaracterística

Descripción:

Esta función permite obtener información sobre las características indicadas en las ubicaciones de las líneas de los documentos. Los parámetros son:

<formato>: Un texto que indica el formato del resultado. Las macros válidas son:
%n: El nombre de la característica. Ejemplo: "Lote" ó "Talla-Color"
%v: El valor de la característica. Ejemplo: "20" ó "38-Negro"
%p: El par nombre-valor. Ejemplo: "Lote 20" ó "Talla 38, Color Negro"
%c: La cantidad indicada: Ejemplo: "5,2"
%u: La unidad en la que se expresa dicha cantidad: Ejemplo: "Kg"
%k: El código de la característicadimensión (o todas), la función devolverá todos los valores existentes en esa dimensión.

%bk: Código de la ubicación
%bn: Nombre de la ubicación
%bp: El par nombre-valor.

Sintaxis: ArtículoUbicaciónCaracterística(<formato>)

Tipo de dato devuelto: Texto

Ejemplo:

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión: SI

Tipo de instrucción:

15.109. NUEVA VARIABLE

Descripción:
Variable:

ArtículoUbicaciónCaracterísticaPar

Fórmula:

ArtículoUbicaciónCaracterística(....)

%k,%n,%v,%p,%c,%u -> para la característica
%bk,%bn,%bp,%bc-> para la ubicación

Sintaxis:

Tipo de dato devuelto:

Ejemplo:

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión: SI

Tipo de instrucción:

15.110. FormateaCantidad

Descripción:

Devuelve una cadena de texto con el valor formateado.
Permite especificar si se han de separar los miles o no, devolver la cadena vacía si es cero, especificar el nº de decimales así como el texto de la unidad que ha de acompañar al valor devuelto.

Parámetros:
Valor - Numérico. Obligatorio. El valor a formatear.
Separar de Miles - Lógico. Opcional, por defecto falso. Formatear el valor con el separador de miles o no.
Mostrar Cero - Lógico. Opcional, por defecto cierto. Si es falso y el valor es cero devuelve la cadena vacía.
Nº decimales - Numérico. Opcional, por defecto 2. Indica el nº de decimales con el que se formateará el valor, redondeando si es preciso.
Unidad - Texto. Opcional, por defecto vacío. Texto que se añadirá a la cadena formateada del valor, separado por un espacio.

Sintaxis: FormateaCantidad(Valor [; Separar miles [; Mostrar cero [; Nº decimales [; Unidad]]]]);

Tipo de dato devuelto: Texto

Ejemplo:

FormateaCantidad(1) -> '1,00'
FormateaCantidad(1000; Falso) -> '1000,00'
FormateaCantidad(1000; Cierto) -> '1.000,00'
FormateaCantidad(0; Falso; Cierto) -> '0,00'
FormateaCantidad(0; Falso; Falso) -> ''
FormateaCantidad(1,235; Falso; Falso) -> '1,24'
FormateaCantidad(1,235; Falso; Falso; 3) -> '1,235'
FormateaCantidad(1,234; Falso; Falso; 2; 'm.') -> '1,23 m.'

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción:

15.111. TxCarga

Descripción:

TxCarga('fichero'; parametros)
TxCarga(binario; parametros)

Carga el contenido del fichero o binario y lo retorna como un texto
El primer parámetro es obligatorio, el segundo es opcional
Parametros es un registro que permite especificar diversas opciones:
Codificación (texto) (posibles valores 'ansi', 'utf8', 'utf16'): por defecto ansi.
Retorna el texto cargado.
Si el contenido es utf con marca BOM, detecta automáticamente la codificación sin necesidad de especificarla.

Sintaxis: TxCarga(<t> | <binario> [; parametros]);

Tipo de dato devuelto: Texto

Ejemplo:

```
texto = 'Hola €9#Ñ&Á?É Í_Ó-Ú.Z$Z!Z\Z/Z%Z<Z>Z áéíóúàèìòùäëïöüâêîôûñ-ÁÉÍÓÚÀÈÌÒÙÄËÏÖÜÂÊÎÔÛÑ-minus-MAYUS çÇ adioS.';
TxGraba(texto; 'C:\Distrito\prueba.txt');

lista = LCrea;
LCarga(lista; 'C:\Distrito\prueba.txt');
LGraba(lista; 'C:\Distrito\prueba_u8.txt'; #[codificación: 'utf8']); %% por defecto le graba el BOM

texto2 = TxCarga('C:\Distrito\prueba_u8.txt'); %% al tener BOM detecta automáticamente la codificación
texto2 = TxLimpiaDer(texto2); %% borro el retorno de carro que le añadió LGraba al final
TxGraba(texto2; 'C:\Distrito\prueba_u8.txt'; #[codificación: 'utf8']);
Mensaje(si TxIgual(texto; texto2) entonces {'texto bien'} sino {'texto mal'});
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción:

15.112. EnUPC

Descripción:

Retorna el texto codificado para poder utilizar con las fuentes UPC-A.ttf, UPC-A-Media.ttf y UPC-A-Baja.ttf (sirven para UPC-A)
si texto no es un UPC-A válido, retorna un texto vacío
el primer parámetro es obligatorio, el segundo es opcional
fuente_alta indica (por defecto cierto):
si es cierto que vamos a utilizar la fuente alta (UPC-A.ttf)
si es falso que vamos a utilizar la fuente media o la baja

Sintaxis: EnUPC('texto'; fuente_alta)

Tipo de dato devuelto:

Ejemplo:

EnUPC('644209420957')
EnUPC('644209420957'; falso)

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción:

15.113. EsEAN13

Descripción:
Retorna cierto si el texto es un código de barras EAN-13 válido (tiene 13 dígitos y el dígito de control es correcto), y falso si no lo es el parámetro es obligatorio

Sintaxis: EsEAN3('texto')

Tipo de dato devuelto:

Ejemplo:
EsEAN13('8425561010039') %% retorna cierto
EsEAN13('8425561010030') %% retorna falso

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción:

15.114. EsUPCA

Descripción:
Retorna cierto si el texto es un código de barras UPC-A válido (tiene 12 dígitos y el dígito de control es correcto), y falso si no lo es el parámetro es obligatorio.

Sintaxis: EsUPCA('texto')

Tipo de dato devuelto:

Ejemplo:
EsUPCA('644209420957') %% retorna cierto
EsUPCA('644209420950') %% retorna falso

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción:

15.115. TxGraba

Descripción:

TxGraba(texto; 'fichero'; parametros)
TxGraba(texto; binario; parametros)

Graba el contenido del texto en el fichero o binario
Los dos primeros parámetros son obligatorios, el tercero es opcional
Parametros es un registro que permite especificar diversas opciones:
Codificación (texto) (posibles valores 'ansi', 'utf8', 'utf16'): por defecto ansi
BOM (lógico): indica si se graba la marca BOM (byte order mark) al principio del fichero si la codificación lo soporta (por defecto cierto)

Sintaxis: TxGraba(texto; <t> | <binario> [; parametros]);

Tipo de dato devuelto:

Ejemplo:

```
texto = 'Hola €9#Ñ&Á?É Í_Ó-Ú.Z$Z!Z\Z/Z%Z<Z>Z áéíóúàèìòùäëïöüâêîôûñ-ÁÉÍÓÚÀÈÌÒÙÄËÏÖÜÂÊÎÔÛÑ-minus-MAYUS çÇ adioS.';
TxGraba(texto; 'C:\Distrito\prueba.txt');

lista = LCrea;
LCarga(lista; 'C:\Distrito\prueba.txt');
LGraba(lista; 'C:\Distrito\prueba_u8.txt'; #[codificación: 'utf8']); %% por defecto le graba el BOM

texto2 = TxCarga('C:\Distrito\prueba_u8.txt'); %% al tener BOM detecta automáticamente la codificación
texto2 = TxLimpiaDer(texto2); %% borro el retorno de carro que le añadió LGraba al final
TxGraba(texto2; 'C:\Distrito\prueba_u8.txt'; #[codificación: 'utf8']);
Mensaje(si TxIgual(texto; texto2) entonces {'texto bien'} sino {'texto mal'});

bin = BinCrea;
BinCarga(bin; 'C:\Distrito\prueba_u8.txt');
bin2 = BinCrea;
TxGraba(texto2; bin2; #[codificación: 'utf8']);
Mensaje(si bin==bin2 entonces {'bin bien'} sino {'bin mal'});
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción:

15.116. CUADROS CONCEPTOS RECURSOS Y MATERIALES

Descripción:

FUNCIONES GENERICAS:

CuadroTraspone(cuadro)
intercambia entre sí las filas y las columnas del cuadro (las filas pasan a ser columnas y las columnas pasan a ser filas)

-Ahora a las celdas de cuadros en el evaluador se les pueden poner enlaces (propiedades de la celda 'TipoEnlace' y 'ClaveEnlace').
-Como clave de enlace toma el valor de la celda, salvo que se le especifique otra.

```
celda = CuadroCelda(cuadro; columna; fila);
celda.Valor = 'ALFOMBRILLA';
celda.TipoEnlace = 'Articulo';
%%celda.ClaveEnlace = 'otra cosa';
```

-Ahora a las columnas de los cuadros se les pueden especificar nuevas propiedades, que se utilizarán como valor por defecto para las celdas:
'CeldaFuente', 'CeldaFuenteColor', 'CeldaFondoColor' y 'CeldaJustificado'

FUNCIONES DE PYME:

CalcularDescuentos('descuentos')
retorna el descuento numérico equivalente a los descuentos en texto separados por ';' que se le pasen como parámetro
p.ej. CalcularDescuentos('10;5') retorna 14.5

-Ahora se puede leer CantidadReal en los Leer actual de las líneas de materiales.
-Ahora se puede leer NombreRecurso en los Leer actual de las líneas de recursos.

Nuevas funciones para obtener los conceptos en los formatos de impresión de documentos: ArtículoMaterial/ArtículoRecurso y ArtículoCuadroMateriales/ArtículoCuadroRecursos.

ArtículoMaterial('formato')
ArtículoRecurso('formato')
retornan un texto formateado sustituyendo variables las variables se especifican como %varible% y se pueden usar las mismas que en el leer actual de materiales y recursos de obras

ArtículoCuadroMateriales(campos; configuracion))
ArtículoCuadroRecursos(campos; configuracion))
retornan un cuadro el primer parámetro es la lista de expresiones de columnas y serán expresiones del evaluador que pueden usar las mismas variables que en el leer actual de materiales y recursos de obras el segundo parámetro es opcional y es una lista de registros que especifica la configuración de las columnas de la tabla

Ejemplos:

```
---------

ArtículoRecurso('Recurso: %codrecurso%: %totalhoras%h %precio% %importe%');
ArtículoMaterial('Material: %codarticulo%: %cantidadreal% %precio% %importe%');

ResultadoNada({
 cuadroRec =
  ArtículoCuadroRecursos(
   #(
    'codrecurso';
```

```
        'nombrerecurso';
        'EnTexto(cantidadrealdoclin * (si totalhoras=="" entonces {0} sino {HoraEnNúmero(totalhoras)/3600})) + "h"';
        'Formatea("0,.00"; precio)';
        'dto = CalcularDescuentos(descuentos); si dto==0 entonces {""} sino {EnTexto(dto)+"%"}';
        'Formatea("0,.00"; cantidadrealdoclin*importe)'
       );
      #(#[ancho: 0,5; celdajustificado: 'I']; #[ancho: 1,5; celdajustificado: 'I']; #[ancho: 1,5]; #[ancho: 1,5]; #[ancho: 1];
#[ancho: 1,5])
     );
   CuadroConfiguracion(cuadroRec; #[celdafuente: 0]);

   cuadroMat =
    ArtículoCuadroMateriales(
     #(
      'codarticulo';
      'nombrearticulo';
      'cantidadrealdoclin*cantidadreal';
      'Formatea("0,.00"; precio)';
      'dto = CalcularDescuentos(descuentos); si dto==0 entonces {""} sino {EnTexto(dto)+"%"}';
      'Formatea("0,.00"; cantidadrealdoclin*importe)'
     );
     #(#[ancho: 0,5; celdajustificado: 'I']; #[ancho: 1,5; celdajustificado: 'I']; #[ancho: 1,5]; #[ancho: 1,5]; #[ancho: 1];
#[ancho: 1,5])
     );
   CuadroConfiguracion(cuadroMat; #[celdafuente: 0]);
  });
  cuadroRec;
  cuadroMat;
```

TIPOS DE ENLACE DE PYME:

- FicheroExterno
- Articulo
- Cliente
- Proveedor
- Agente
- Documento
- PresupuestoProveedor
- PedidoProveedor
- AlbaranProveedor
- FacturaProveedor
- PresupuestoCliente
- PedidoCliente
- AlbaranCliente
- FacturaCliente
- TicketCliente
- MovimientosAlmacen
- RecuentoAlmacen
- Recurso
- Proyecto
- Certificacion
- Fabrica
- EstiloFabrica
- Reparacion
- EstiloReparacion
- ObjetoReparar
- Contrato
- Recibos
- Remesa
- Almacen
- Estacion
- Cuenta

Zona
Familia
GrupoCategoria
Categoria
Tarifa
FormaPago
FormaEnvio
TipoFormaPago
TipoCliente
TipoDireccion
TipoCatalogo
TipoContacto
TipoEstadoSolicitud
TipoContrato

Sintaxis:

Tipo de dato devuelto:

TipoTrabajo

Ejemplo:

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción:

15.117. Operadores

Descripción:

1. Operadores de cálculo

+ (más)
Suma de dos números
Ejemplo: 3 + 4 devuelve 7
Concatenación de dos textos
Ejemplo: "uno" + "dos" devuelve "unodos"

- (menos)
Resta de dos números
Ejemplo: 3 - 4 devuelve -1
Cambio de signo de un número
Ejemplo: -(3 + 4) devuelve -7

** (asterisco)*
Multiplicación de dos números
*Ejemplo: 3 * 4 devuelve 12*

/ (barra de división)
División de dos números
Ejemplo: 3 / 4 devuelve 0,75

modulo
Resto de la división entera de dos números
Ejemplo 7 modulo 4 devuelve 3

^ (Potencia)
Eleva el primer término a la potencia expresada por el segundo.
Ejemplo 3^2 devuelve 9

no
negación de un valor lógico
Ejemplo: no (3 > 4) devuelve cierto

y
Combina dos valores lógicos con el operador lógico AND
Ejemplo: (1 > 2) y (3 < 4) devuelve falso

o
Combina dos valores lógicos con el operador lógico OR
Ejemplo: (1 > 2) o (3 < 4) devuelve cierto

2. Operador de asignación

= (igual)
Asigna un valor (y un tipo) a una variable, definiéndola si no existe
Ejemplo: a = 7
Nota: No utilizar este operador para comparar dos elementos (un error bastante frecuente)

3. Operadores de comparación

== (doble igual)
Devuelve cierto si los dos elementos comparados son iguales. Es aplicable a todos los tipos, binarios incluidos, pero si los elementos son de

diferente tipo, se produce un "error de tipo".
Si se comparan cadenas alfanuméricas, la comparación es case insensitive: "HOLA" == "hola" devuelve CIERTO.
Ejemplo: 3 == 4 devuelve falso

<> (menor-mayor)
Devuelve cierto si los dos elementos comparados son diferentes. Es aplicable a todos los tipos, binarios incluidos, pero si los elementos son de
diferente tipo, se produce un "error de tipo".
Si se comparan cadenas alfanuméricas, la comparación es case insensitive: "HOLA" <> "hola" devuelve FALSO.
Ejemplo: 3 <> 4 devuelve cierto

> , >= , < , <= (mayor, mayor-igual, menor, menor-igual)
Si los operandos son números, evalúa las comparaciones según el valor numérico de los operandos
Ejemplo: 3 > 4 devuelve falso
Si los operandos son textos, evalúa las comparaciones según el orden alfabético de los operandos, sin tener en cuenta las mayúsculas /
minúsculas
Ejemplo: "abeja" < "mosca" devuelve cierto
Si los operandos son lógicos, se considera que cierto es mayor a falso
Ejemplo: cierto > falso devuelve cierto
Si los operandos son de diferente tipo entre sí, o de cualquier otro tipo que los arriba indicados, se produce un "error de tipo"

4. Otros operadores

( ) (paréntesis)
Agrupan operaciones, alterando su precedencia. Se permiten ilimitados niveles
Ejemplo: (a + b)(c + (d - e) / 2)*
Indican los argumentos o parámetros de las funciones
Ejemplo: EnLetras(a)
Precedidos por el símbolo #, definen una lista de elementos
Ejemplo: #("uno";"dos";"tres")

[ ] (corchetes)
Permiten obtener un caracter de un texto
Ejemplo: a = "distrito-k"; a[3] devuelve "s"
Permiten obtener un elemento de una lista (array)
Ejemplo: a = #("uno";"dos";"tres"); a[2] devuelve "dos"
Permite leer el byte de un binario
Ejemplo: binData[1] devuelve el valor del primer byte el binario binData
Evalúan una función predefinida (depende de la aplicación)
Ejemplo (sólo en SQL Conta): [s '430'] devuelve el saldo de la cuenta 430

{ } (llaves)
Delimitan un bloque de código
Ejemplo: si a == b entonces {a = a + 1} sino {a = a - 1}

@ (arroba)
Permite obtener el valor de una variable cuyo nombre se obtiene de una expresión. De esta forma se puede conseguir acceder a una variable que tenga espacios en blanco en el nombre. Puede usarse para acceder al valor de los datos adiciones que tiene espacios en blanco desde un formato de impresión. También permite evaluar un bloque de código.
Ejemplos:
a = 25; nombre = "a"; @nombre devuelve 25
a = 25; @"a" devuelve 25
Dato adicional artículo = 'X1'
@"Dato adicional artículo" devuelve 'X1'

& (ampersand)
Permite obtener la variable cuyo nombre se obtiene a partir de una expresión
Ejemplo: a = 25; &a devuelve la variable a, no el valor 25

; (punto y coma)
Permite separar los elementos de una lista (array).
Ejemplo: a = #("uno";"dos";"tres")
Permite separar los argumentos o parámetros de una función.
Ejemplo: Min(a; b)
Permite ejecutar más de una orden, separándolas entre sí. El valor devuelto por el conjunto de órdenes es el último evaluado.
Ejemplo: a = 3; b = 4; a + b; Se ejecutan las tres órdenes, y devuelve 7

Sintaxis:

Tipo de dato devuelto:

Ejemplo:

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción:

16. FUNCIONES ESPECÍFICAS FORMATOS DE IMPRESIÓN

16.1. DimensionesTextoImpresión

Descripción:
DimensionesTextoImpresión('texto'; fuente)

Calcula las dimensiones del texto, según la fuente especificada.
Sólo funciona en los formatos de impresión y en el generador de informes
Todos los parámetros son obligatorios
· Texto: Literal cuyas dimensiones se desea obtener
· Fuente: Número de fuente del formato de impresión

Retorna un registro con los siguientes campos:
Ancho (valor de tipo numérico, en cm)
Alto (valor de tipo numérico, en cm)

Sintaxis: DimensionesTextoImpresión('texto'; fuente)

Tipo de dato devuelto: TRegistro

Ejemplo:
DimensionesTextoImpresión('Texto de prueba'; 0);
%% si la fuente 0 es Arial 10, retorna: #[Alto: 0,38947 Ancho: 2,56117]
%% si la fuente 0 es Arial 10 negrita, retorna: #[Alto: 0,4064 Ancho: 2,72203]

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión: SI

Tipo de instrucción: Funciones específicas formatos de impresión

16.2. ArtículoCaracterística

Descripción:

Esta función permite obtener información sobre las características indicadas en las líneas de los documentos. Los parámetros son:
<formato>: Un texto que indica el formato del resultado. Las macros válidas son:
%n: El nombre de la característica. Ejemplo: "Lote" ó "Talla-Color"
%v: El valor de la característica. Ejemplo: "20" ó "38-Negro"
%p: El par nombre-valor. Ejemplo: "Lote 20" ó "Talla 38, Color Negro"
%c: La cantidad indicada: Ejemplo: "5,2"
%u: La unidad en la que se expresa dicha cantidad: Ejemplo: "Kg"
%k: El código de la característica
<dato>: Un texto opcional que indica el nombre de la característica requerida. Si este parámetro no se indica, la función devuelve una lista con todas las características existentes en la línea del documento.
<dimensión1> ... <dimensiónN>: Textos opcionales que indican el valor requerido de cada una de las dimensiones de la característica. Si se omite alguna dimensión (o todas), la función devolverá todos los valores existentes en esa dimensión.
<orden>: Texto opcional que indica el nombre de la dimensión que devolverá y por la que se ordenará la lista resultante.

Sintaxis: ArtículoCaracterística(<formato>; <dato>; <dimensión1>; ...; <dimensiónN>; <orden>)

Tipo de dato devuelto: Texto

Ejemplo:

ArtículoCaracterística("%p: %c");

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión: SI

Tipo de instrucción: Funciones específicas formatos de impresión

17. FUNCIONES DE CUADROS

17.1. ArtículoCuadroCaracterística

Descripción:

Cuadro de dos dimensiones con los valores de características de la línea.

Si se añade el campo directamente al formato de impresión pueden configurarse varias de sus propiedades. Si se utiliza dentro de los scripts se tiene acceso a todas las propiedades del cuadro.

todos_valores: Booleano. Indica si se muestran todos los valores permitidos de la característica (cierto) o sólo aquellos que tienen contenido (falso).

Puede formularse con el campo. Devuelve una lista de cuadros, un cuadro por cada característica que tenga el artículo.

Como títulos de columnas se utiliza la última dimensión de la característica y como títulos de filas todas las demás. Si es una característica de una única dimensión sólo tendrá columnas.

Sintaxis: ArtículoCuadroCaracterística([todos_valores])

Tipo de dato devuelto: Lista cuadros

Ejemplo:

```
ResultadoNada({
 %% Configuración de si imprimir todos los valores permitidos de las características o sólo los que tengan cantidad
 todos_valores = falso;

 lista_cuadros = ArtículoCuadroCaracterísticas(todos_valores);
 i = 1; mientras {i <= LTamaño(lista_cuadros)} haz {
  CuadroConfiguración(lista_cuadros[i]; #[
    MostrarTítulosColumnas: cierto
    TítuloColumnaAncho: 1,2
    CeldaFuente: 2
    CeldaFuenteColor: "#000000"
    CeldaJustificado: "I"
    TítuloFuente: 3
    TítuloFuenteColor: "#000000"
    TítuloFondoColor: "#CCCCCC"
    TítuloJustificado: "C"
    BordesInteriores: cierto
    BordesExteriores: cierto
    BordesColor: "#000000"
   ]
  );
  i = i + 1;
 };
});
ArtículoNombre;
lista_cuadros;
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión: SI

Tipo de instrucción: Funciones cuadros

17.2. CuadroCuantasFilas

Descripción:
Devuelve el número de filas que tiene un cuadro. No se cuenta la fila de títulos.

Sintaxis: CuadroCuantasFilas(<cuadro>)

Tipo de dato devuelto: Número

Ejemplo:

```
cuadro = CuadroCrea;
i = 1; mientras {i <= 4} haz {
  CuadroAñadeFila(cuadro; "Fila " + Formatea("0"; i));
  i = i + 1;
};
x = CuadroCuantasFilas(cuadro); %% x valdrá 4
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.3. CuadroConfiguración

Descripción:

Permite configurar las propiedades del cuado. Se pasa un registro con los campos de la configuración. No es obligatorio pasar todo los campos posibles, los que no se pasan mantienen sus valores. Los valores que se fijan aquí serán los que utilizarán todas las celdas que no tengan indicado un valor distinto en las propiedades de la celda.

Campos:
MostrarTítulosColumnas: Booleano. Indica si se mostrarán los títulos de las columnas. Por defecto: cierto.
TítuloColumnaAncho: Número. Ancho de las columnas del cuadro en centímetros. Por defecto 1. Puede modificarse independientemente cada columna con la instrucción CuadroColumna.
TítuloColumnaFuente: Número. Fuente de los títulos de columna. Es el número de la fuente en el formato de impresión o informe. Las fuentes empiezan a contar en cero. Por defecto: la que tenga el campo fórmula donde se imprime. Puede modificarse independientemente cada columna con la instrucción CuadroColumna.
TítuloColumnaFuenteColor: Texto. Código color html en hexadecimal de la fuente. Por defecto: #000000. Puede modificarse independientemente cada columna con la instrucción CuadroColumna.
TítuloColumnaFondoColor: Texto. Código color html en hexadecimal de fondo de las celdas. Por defecto: <vacío> (no pinta nada). Puede fijarse por celda.
TítuloColumnaJustificado: Texto. Indica el alineado del valor de las celdas. Valores posibles: I (Izquierda), D (Derecha), C (Centrado), J
(Justificado). Por defecto: 'C'.
MostrarTítulosColumnaTodasPaginas: Booleano. Indica si se mostrarán los títulos de las columnas cuando el cuadro salte a la siguiente página.
Por defecto: falso. Sólo válido en cuadros impresos con el editor de informes.

MostrarTítulosFilas: Booleano. Indica si se mostarán los títulos de las filas. Por defecto: cierto.
TítuloFilaAncho: Número. Ancho de la columna con el título de las filas en centímetros. Por defecto 1. Puede modificarse con la instrucción CuadroColumnaTítuloFila.
TítuloFilaFuente: Número. Fuente de los títulos de fila. Es el número de la fuente en el formato de impresión o informe. Las fuentes empiezan a contar en cero. Por defecto: la que tenga el campo fórmula donde se imprime. Puede modificarse con la instrucción CuadroColumnaTítuloFila.
TítuloFilaFuenteColor: Texto. Código color html en hexadecimal de la fuente. Por defecto: #000000. Puede modificarse con la instrucciónCuadroColumnaTítuloFila.
TítuloFilaFondoColor: Texto. Código color html en hexadecimal de fondo de las celdas. Por defecto: <vacío> (no pinta nada). Puede modificarse con la instrucción CuadroColumnaTítuloFila.
TítuloFilaJustificado: Texto. Indica el alineado del valor de las celdas. Valores posibles: I (Izquierda), D (Derecha), C (Centrado). Por defecto: 'I'.

CeldaFuente: Número. Índice de la fuente en el formato de impresión o informe. Las fuentes empiezan a contar en cero. Por defecto: la que tenga el campo fórmula donde se imprime. Puede fijarse por celda.
CeldaFuenteColor: Texto. Código color html en hexadecimal de la fuente. Por defecto: #000000. Puede fijarse por celda.
CeldaFondoColor: Texto. Código color html en hexadecimal de fondo de las celdas. Por defecto: <vacío> (no pinta nada). Puede fijarse por celda.
CeldaJustificado: Texto. Indica el alineado del valor de las celdas. Valores posibles: I (Izquierda), D (Derecha), C (Centrado). Por defecto: 'D'. Puede fijarse por celda.

BordesInteriores: Booleano. Indica si se imprimirán los bordes interiores del cuadro. Por defecto: cierto.
BordesExteriores: Booleano. Indica si se imprimirán los bordes exteriores del cuadro. Por defecto: cierto.
BordesColor: Textol. Código color html en hexadecimal para las bordes del cuadro, tanto interiores como exteriores. Por defecto: #808080.

FilaAlto: Número. Alto de las filas en cm. En los informes del generador se utilizará este alto, en los formatos de impresión de documentos se usará el menor de los altos de los campos de artículo, como en el resto de campos. Por defecto 0,4 cm.

Sintaxis: CuadroConfiguración(<cuadro>; <registro_configuración>)

Tipo de dato devuelto: Nada

Ejemplo:

```
ResultadoNada({
  %% Impresión cuadro características
  %% Obtención características línea
  lista_cuadros_características = ArtículoCuadroCaracterísticas(falso);
  %% La variable ArtículoCuadroCaracterísticas devuelve una lista donde cada característica del artículo es un
elemento
  %% Fijar propiedades cuadros
  i = 1; mientras {i <= LTamaño(lista_cuadros_características)} haz {
    %% Asignación propiedades cuadro
    %% No es obligatorio indicarlas todas cuando se configuran, las propiedades no indicadas mantienen su valor
    CuadroConfiguración(
      lista_cuadros_características[i];
      #[
        TítuloColumnaAncho: 1
        TítuloColumnaFondoColor: "#C0C0C0"
        TítuloFilaFondoColor: "#C0C0C0"
        TítuloFilaAncho: 2,5
        BordesColor: "#000000"
      ]
    );
    i = i + 1;
  };
});
lista_cuadros_características;
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión: NO

Tipo de instrucción: Funciones cuadros

17.4. CuadroCrea

Descripción:
Crea un cuadro vacío.

Sintaxis: CuadroCrea

Tipo de dato devuelto: TCuadro

Ejemplo:
cuadro = CuadroCrea;

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.5. CuadroTraspone

Descripción:
Intercambia entre sí las filas y las columnas del cuadro (las filas pasan a ser columnas y las columnas pasan a ser filas)

Sintaxis: CuadroTraspone(cuadro)

Tipo de dato devuelto: Nada

Ejemplo:

```
cuadro = cuadroCrea;
cuadroañadefila(cuadro; "Fila 1");
cuadroañadefila(cuadro; "Fila 2");
CuadroAñadeColumna(cuadro; "Col 1");
CuadroAñadeColumna(cuadro; "Col 2");

CuadroCelda(cuadro; 1;1).valor = 11;
CuadroCelda(cuadro; 1;2).valor = 12;
CuadroCelda(cuadro; 2;1).valor = 21;
CuadroCelda(cuadro; 2;2).valor = 22;

Cuadrotraspone(cuadro);
cuadro;
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.6. CuadroFila

Descripción:

Permite fijar las propiedades de una fila concreta. Devuelve un registro con las propiedades de la fila.
Sólo afecta a la columna de títulos, el resto de columnas se cambian con las propiedades de la celda, columna o cuadro (la primera que esté fijada).
Campos del registro:
Fuente: Índice de la fuente de la fila
FuenteColor: Código color html en hexadecimal de la fuente. Ejemplo: #FF00FF.
FondoColor: Código color html en hexadecimal de fondo de la celda. Ejemplo: #FFFFF.
Título: Título de la fila
Justificado: Indica el alineado del valor. Valores posibles: I (Izquierda), D (Derecha), C (Centrado), J (Justificado).

Sintaxis: CuadroFila(cuadro; fila)

Tipo de dato devuelto: registro

Ejemplo:

```
ResultadoNada({
 %% Creación cuadro
 cuadro = CuadroCrea;
 %% Creación columnas
 i = 1; mientras {i <= 4} haz {
  CuadroAñadeColumna(cuadro; "Col. " + Formatea("0"; i));
  i = i + 1;
 };
 %% Creación filas
 i = 1; mientras {i <= 10} haz {
  CuadroAñadeFila(cuadro; "Fila " + Formatea("0"; i));
  i = i + 1;
 };
 %% Rellenado celdas
 i = 0;
 f = 1; mientras {f <= CuadroCuantasFilas(cuadro)} haz {
  c = 1; mientras {c <= CuadroCuantasColumnas(cuadro)} haz {
   i = i + 1;
   CuadroCelda(cuadro; c; f).valor = i;
   c = c + 1;
  };
  f = f + 1;
 };
 %% Fijado fondo filas pares
 f = 1; mientras {f <= CuadroCuantasFilas(cuadro)} haz {
  si (f modulo 2 == 0) entonces {
   CuadroFila(cuadro; f).FondoColor = "#D7D7D7";
  };
  f = f + 1;
 };
 %% Fijado fondo columnas pares
 c = 1; mientras {c <= CuadroCuantasColumnas(cuadro)} haz {
  si (c modulo 2 == 0) entonces {
   CuadroColumna(cuadro; c).FondoColor = "#D7D7D7";
  };
  c = c + 1;
 };
});
cuadro;
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.7. CuadroIdentificación

Descripción:

Devuelve un registro con la identificación del cuadro. No pueden modificarse estos datos en el cuadro. Cuando el cuadro lo ha generado el programa automáticamente a partir de una variable que se carga con características tendrá la información indicada en cada campo más adelante, si se ha generado por scripts estos campos estarán vacíos.

Campos:
Identificador: Texto. Código de la característica
DescripciónColumnas: Texto. Nombre de la dimensión de la característica que se utiliza para los títulos de columna. Es la última dimensión de la característica.
DescripciónFilas: Texto. Nombre de la dimensión de la característica que se utiliza para los títulos de fila. Son todas las dimensiones de la característica menos la última.

Sintaxis: CuadroIdentificación(cuadro)

Tipo de dato devuelto: registro

Ejemplo:

```
i = 1; mientras {i <= LTamaño(ArtículoCuadroCaracterísticas)} haz {
 x = CuadroIdentificacion(ArtículoCuadroCaracterísticas[i]);
 %
 Con la característica 4 de 3 dimensiones (Caducidad-Lote-Marca), x valdrá:
 #[
  DescripcionColumas: 'Marca'
   DescripcionFilas: 'Caducidad - Lote'
  Identificador: '4'
 ]
 %
 i = i + 1;
};
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.8. CuadroCuantasColumnas

Descripción:
Devuelve el número de columnas que tiene un cuadro. No se cuenta la columna con los títulos de las filas.

Sintaxis: CuadroCuantasColumnas(<cuadro>)

Tipo de dato devuelto: Número

Ejemplo:

```
cuadro = CuadroCrea;
i = 1; mientras {i <= 5} haz {
  CuadroAñadeColumna(cuadro; "Columna " + Formatea("0"; i));
  i = i + 1;
};
x = CuadroCuantasColumnas(cuadro); %% x valdrá 5
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.9. CuadroColumnaTítuloFila

Descripción:

Permite fijar las propiedades de la primera columna, la que tiene los títulos de las filas. Devuelve un registro con las propiedades de la columna.
Para fijar las propiedades se crea una variable con el resultado de la función y se manipula, al manipular la misma se modifica el cuadro al que está vinculada.
Otra opción es usar la función como un registro y manipular el campo deseado.
Campos del registro:
Fuente: Índice de la fuente de la columna
FuenteColor: Código color html en hexadecimal de la fuente. Ejemplo: #FF00FF.
FondoColor: Código color html en hexadecimal de fondo de la celda. Ejemplo: #FFFFF.
Título: Título de la columna
Justificado: Indica el alineado del valor. Valores posibles: I (Izquierda), D (Derecha), C (Centrado), J (Justificado).
Ancho: Ancho de la columna en centímetros (cm).

Sintaxis: CuadroColumnaTítuloFila(<cuadro>)

Tipo de dato devuelto: registro

Ejemplo:

```
ResultadoNada({
  %% Impresión cuadro características

  %% Obtención características línea
  lista_cuadros_características = ArtículoCuadroCaracterísticas;
  %% La variable ArtículoCuadroCaracterísticas devuelve una lista donde cada característica del artículo es un
elemento

  %% Fijar propiedades cuadros
  i = 1; mientras {i <= LTamaño(lista_cuadros_características)} haz {

    %% Asignación propiedades cuadro
    %% No es obligatorio indicarlas todas cuando se configuran, las propiedades no indicadas mantienen su valor
    CuadroConfiguración(
     lista_cuadros_características[i];
     #[
       AnchoColumna: 0,65
       TítuloColumnaFondoColor: "#C0C0C0"
       TítuloFilaFondoColor: "#C0C0C0"
       BordesColor: "#000000"
     ]
    );

    %% Fijar el ancho de la primera columna (títulos)
    CuadroColumnaTítuloFila(lista_cuadros_características[i]).ancho = 2;
    %% Fijar alineación columna títulos fila con variable
    columna_títulos = CuadroColumnaTituloFila(lista_cuadros_características[i]);
    columna_títulos.título = 'Talla - C.';
    columna_títulos.justificado = 'C';
    columna_títulos.fondocolor = "#C0C0C0";
    i = i + 1;
   };
  });
  lista_cuadros_características;
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.10. CuadroBorrarFila

Descripción:
Elimina la fila indicada del cuadro.
Las filas empiezan a contar en 1.

Sintaxis: CuadroBorrarFila(<cuadro>; <fila>)

Tipo de dato devuelto: Nada

Ejemplo:

```
cuadro = CuadroCrea;
i = 1; mientras {i <= 4} haz {
  CuadroAñadeFila(cuadro; "Fila " + Formatea("0"; i));
  i = i + 1;
};
x = CuadroCuantasFilas(cuadro); %% x valdrá 4
CuadroBorrarFila(cuadro; 2);
x = CuadroCuantasFilas(cuadro); %% x valdrá 3
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.11. CuadroColumna

Descripción:

Permite fijar las propiedades de una columna concreta. Devuelve un registro con las propiedades de la columna. Las celdas usarán esta configuración salvo que tengan fijada una distinta.

Campos del registro:
Fuente: Índice de la fuente de la columna
FuenteColor: Código color html en hexadecimal de la fuente. Ejemplo: #FF00FF.
FondoColor: Código color html en hexadecimal de fondo de la celda. Ejemplo: #FFFFF.
Título: Título de la columna
Justificado: Indica el alineado del valor. Valores posibles: I (Izquierda), D (Derecha), C (Centrado), J (Justificado)
Ancho: Ancho de la columna en centímetros (cm).

Valores por defecto para las celdas de datos la columna.
CeldaFuente: Fuente por defecto para las celdas de la columna.
CeldaFuenteColor: Color del texto por defecto para las celdas de la columna
CeldaFondoColor: Color del fordo por defecto para las celdas de la columna
CeldaJustificado: Justificación por defecto para las celdas de la columna.

Sintaxis: CuadroColumna(<cuadro>; <columna>)

Tipo de dato devuelto: registro

Ejemplo:

```
ResultadoNada({
 %% Creación cuadro
 cuadro = CuadroCrea;
 %% Creación columnas
 i = 1; mientras {i <= 4} haz {
  CuadroAñadeColumna(cuadro; "Col. " + Formatea("0"; i));
  i = i + 1;
 };
 %% Creación filas
 i = 1; mientras {i <= 10} haz {
  CuadroAñadeFila(cuadro; "Fila " + Formatea("0"; i));
  i = i + 1;
 };
 %% Rellenado celdas
 i = 0;
 f = 1; mientras {f <= CuadroCuantasFilas(cuadro)} haz {
  c = 1; mientras {c <= CuadroCuantasColumnas(cuadro)} haz {
   i = i + 1;
   CuadroCelda(cuadro; c; f).valor = i;
   c = c + 1;
  };
  f = f + 1;
 };
 %% Fijado fondo filas pares
 f = 1; mientras {f <= CuadroCuantasFilas(cuadro)} haz {
  si (f modulo 2 == 0) entonces {
   CuadroFila(cuadro; f).FondoColor = "#D7D7D7";
  };
  f = f + 1;
 };
 %% Fijado fondo columnas pares
 c = 1; mientras {c <= CuadroCuantasColumnas(cuadro)} haz {
  si (c modulo 2 == 0) entonces {
   CuadroColumna(cuadro; c).FondoColor = "#D7D7D7";
  };
  c = c + 1;
```

```
  };
});
cuadro;
```

Instrucción específica de contabilidad:
Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.12. CuadroCelda

Descripción:

Permite fijar las propiedades de una celda concreta. Devuelve un registro con las propiedades de la celda. Las celdas de títulos (filas y columnas) no están accesibles con esta función, algunas se manipulan con "CuadroConfiguración". Si alguna propiedad no se fija se utiliza de la columna y si tampoco está fijada ahí se usa la del cuadro.

Campos del registro:

Fuente: Índice de la fuente de la columna

FuenteColor: Código color html en hexadecimal de la fuente. Ejemplo: #FF00FF.

FondoColor: Código color html en hexadecimal de fondo de la celda. Ejemplo: #FFFFF.

Valor: Contenido de la celda.

Justificado: Indica el alineado del valor. Valores posibles: I (Izquierda), D (Derecha), C (Centrado), J (Justificado).

Máscara: Indica la máscara que se utilizará para formatear los valores de las celdas. Depende del tipo de dato de la celda.

TipoEnlace: Texto identificativo del tipo de enlace de la celda.

ClaveEnlace: Valor de la clave para identificar el registro del tipo definido en TipoEnlace. No puede pasarse un dato de tipo numérico, debe de ser un texto para que no se añada el punto de los miles.

TIPOS DE ENLACE DE PYME:

FicheroExterno
Articulo
Cliente
Proveedor
Agente
Documento
PresupuestoProveedor
PedidoProveedor
AlbaranProveedor
FacturaProveedor
PresupuestoCliente
PedidoCliente
AlbaranCliente
FacturaCliente
TicketCliente
MovimientosAlmacen
RecuentoAlmacen
Recurso
Proyecto
Certificacion
Fabrica
EstiloFabrica
Reparacion
EstiloReparacion
ObjetoReparar
Contrato
Recibos
Remesa
Almacen
Estacion
Cuenta
Zona
Familia
GrupoCategoria
Categoria
Tarifa
FormaPago
FormaEnvio
TipoFormaPago
TipoCliente
TipoDireccion
TipoCatalogo

TipoContacto
TipoEstadoSolicitud
TipoContrato
TipoTrabajo

Sintaxis: CuadroCelda(<cuadro>; <columna>; <fila>)
Tipo de dato devuelto: registro

Ejemplo:

```
cuadro = CuadroCrea;
i = 1; mientras {i <= 5} haz {
  CuadroAñadeColumna(cuadro; "Columna " + Formatea("0"; i));
  i = i + 1;
};
CuadroAñadeFila(cuadro; "Fila 1");
CuadroCelda(cuadro; 1; 1).valor = 3;  %% La celda 1,1 tendrá el valor 3
x = CuadroCelda(cuadro; 1; 1).valor; %% x valdrá 3
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.13. CuadroBorrarColumna

Descripción:
Elimina la columna indicada del cuadro.
Las columnas empiezan a contar en 1.

Sintaxis: CuadroBorrarColumna(<cuadro>; <columna>)

Tipo de dato devuelto: Nada

Ejemplo:

```
cuadro = CuadroCrea;
i = 1; mientras {i <= 5} haz {
  CuadroAñadeColumna(cuadro; "Columna " + Formatea("0"; i));
  i = i + 1;
};
x = CuadroCuantasColumnas(cuadro); %% x valdrá 5
CuadroBorrarColumna(cuadro; 3);
x = CuadroCuantasColumnas(cuadro); %% x valdrá 4
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.14. CuadroAñadeFila

Descripción:

Permite añadir filas a un cuadro existente.
Devuelve el índice de la fila. Empiezan a contar en 1.
El alto será el menor de todos los altos de la línea en la impresión.

Se pasan los siguientes parámetros:
cuadro: Obligatorio. Cuadro al que se añadirá la columna.
título_fila: Obligatorio. Título de la fila.
registro_fila: Opcional. Registro con las propiedades de la fila. Tiene la misma estructura que el registro devuelto por CuadroFila.

Sintaxis: CuadroAñadeFila(<cuadro>; <título_fila> [;<registro_fila>])

Tipo de dato devuelto: Número

Ejemplo:

```
cuadro = CuadroCrea;
i = 1; mientras {i <= 4} haz {
  CuadroAñadeFila(cuadro; "Fila " + Formatea("0"; i));
  i = i + 1;
};
x = CuadroCuantasFilas(cuadro); %% x valdrá 4
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.15. CuadroAñadeColumna

Descripción:

Permite añadir columnas a un cuadro existente.
Devuelve el índice de la columna. Empiezan a contar en 1.

Se pasan los siguientes parámetros:
cuadro: Obligatorio. Cuadro al que se añadirá la columna.
título_columna: Obligatorio. Título de la columna.
registro_columna: Opcional. Registro con las propiedades de la columna. Tiene la misma estructura que el registro devuelto por CuadroColumna.

Sintaxis: CuadroAñadeColumna(<cuadro>; <título_columna> [;<registro_columna>])

Tipo de dato devuelto: Número

Ejemplo:

```
cuadro = CuadroCrea;
i = 1; mientras {i <= 5} haz {
  CuadroAñadeColumna(cuadro; "Columna " + Formatea("0"; i));
  i = i + 1;
};
x = CuadroCuantasColumnas(cuadro); %% x valdrá 5
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

17.16. CuadroExporta

Descripción:

Exporta un cuadro a Excel.
Según la extensión del fichero especificado (XLS o XLSX) genera un fichero para una versión de excel u otra.

Fuentes:
Por defecto, cada celda asumirá la fuente especificada para la celda o columna. En los formatos de impresión e informes siempre se asume que hay una fuente especificada, pero desde un script no siempre es asi, por lo que en estos casos es necesario enviar una lista con los datos de las fuentes, se trata de una lista de registros donde cada registro contiene los datos de una fuente. El primer registro de la lista contiene los datos de la fuente 0.
Cada registro se ajusta al siguiente formato:

```
#[Nombre: "Calibri";
    Tamaño: 20;
    Negrita: Cierto;
    Cursiva: Cierto;
    Subrayado: Cierto;
```

Sintaxis: CuadroExporta(cuadro; fichero; #(#[Datos de las Fuentes]; ...))

Tipo de dato devuelto:

Ejemplo:

```
%%
=============================================================================================
%% Obtiene una lista y la exporta directamente a excel
%%
=============================================================================================
%% Funciones auxiliares
%% ---------------------------------------------------------------------------------------

%% Copia los valores de un registro sobre otro, sobreescribiendo los valores de campos existentes y añadiendo los campos que no existan
%% RCopy(ROrigen; RDestino);
RCopy = {
  RC_ROrigen = Params[1];
  RC_RDestino = Params[2];

  RC_IDCampo = 1;
  mientras {RC_IDCampo <= RTamaño(RC_ROrigen)} haz {
    RC_NombreCampo = RNombreCampo(RC_ROrigen; RC_IDCampo);
    si no RExisteCampo(RC_RDestino; RC_NombreCampo) entonces {
      RAñade(RC_RDestino; RC_NombreCampo);
    };
    RC_RDestino[RC_NombreCampo] = RC_ROrigen[RC_NombreCampo];

    RC_IDCampo = RC_IDCampo + 1;
  };
}; %% RCopy


%% Crea un cuadro.
%% Devuelve un cuadro con las columnas y los títulos especificados
%% CrearCuadro(Lista de titulos de columna; Lista de registros de formato de columna [; Registro de configuración de cuadro]);
CrearCuadro = {
  CC_ColTitles = Params[1];
  CC_ColFormat = Params[2];
  si lTamaño(Params) >= 3 entonces {
```

```
    CC_CuadroCFG = Params[3];
  } sino {
    %% Valores por defecto
    CC_CuadroCFG = #[celdafuente: 0
                MostrarTítulosFilas: Falso;
                BordesInteriores: Cierto
                Bordesexteriores: Cierto];
  };

  CC_Cuadro = cuadroCrea;
  CuadroConfiguracion(CC_Cuadro; CC_CuadroCFG);
    Tachado: Cierto];        CC_i = 1;
  Mientras {CC_i <= lTamaño(CC_ColTitles)} haz {
    si CC_i <= ltamaño(CC_ColFormat) entonces {
      CC_ColCfg = CC_ColFormat[CC_i];
    } sino {
      CC_ColCfg = #[];
    };
    cuadroAñadeColumna(CC_Cuadro; CC_ColTitles[CC_i]; CC_ColCfg);
    CC_i = CC_i + 1;
  };

  CC_Cuadro;
 }; %% CreaCuadro

 %% Añade una línea a un cuadro, cubierta ya con los valores especificados
 %% Devuelve el índice a la línea añadida
 %% CrearLineaCuadro(Cuadro; Lista de los datos de las celdas [; Registro de configuración de fila [; Lista de
registros de configuracion de
 celda]]);
 CrearLineaCuadro = {
  CLC_Cuadro = Params[1];
  CLC_Datos = Params[2];
  si lTamaño(Params) >= 3 entonces {
    CLC_CeldasCFG = Params[3];
  } sino {
    %% Valores por defecto
    CLC_CeldasCFG = #();
  };
  si lTamaño(Params) >= 4 entonces {
    CLC_CuadroCFG = Params[4];
  } sino {
    %% Valores por defecto
    CLC_CuadroCFG = #[];
  };

  CLC_nColumnas = CuadroCuantasColumnas(CLC_Cuadro);
  CLC_IDFila = CuadroAñadeFila(CLC_Cuadro; ""; CLC_CuadroCFG);

  CLC_IDCol = 1;
  Mientras {CLC_IDCol <= Min(lTamaño(CLC_Datos); CLC_nColumnas)} haz {
    CLC_Celda = CuadroCelda(CLC_Cuadro; CLC_IDCol; CLC_IDFila);
    CLC_Celda.Valor = CLC_Datos[CLC_IDCol];
    si CLC_IDCol <= lTamaño(CLC_CeldasCFG) entonces {
      @RCopy(CLC_CeldasCFG[CLC_IDCol]; CLC_Celda);
```

Instrucción específica de contabilidad:

Instrucción específica de formatos de edición:

Instrucción específica de formatos de impresión:

Tipo de instrucción: Funciones cuadros

18. FORMATO DE MÁSCARAS PARA NÚMEROS

Una máscara para números es un texto que indica la forma de presentar un número, y puede estar formada por los siguientes especificadores:

"0" Indica la posición de un dígito. Si el valor que se está formateando tiene un dígito en la posición donde el "0" aparece en la máscara, entonces se muestra ese dígito. Si no, se muestra un "0" en esa posición.

"#" Indica la posición de un dígito. Si el valor que se está formateando tiene un dígito en la posición donde el "0" aparece en la máscara, entonces se muestra ese dígito. Si no, no se muestra nada en esa posición.

"." Separador decimal. El primer caracter "." en la máscara determina la posición del separador decimal en el valor presentado. Cualquier "." adicional se ignora.

"," Separador de miles. Si la máscara contiene algún caracter ",", el valor presentado tendrá separadores de miles cada tres dígitos a la izquierda del separador decimal. La ubicación y la cantidad de "," en la máscara no afecta al resultado.

"e+" Notación científica. Si cualquiera de los textos "e+", "e-" aparecen en la máscara, el número es presentado utilizando notación científica. Puede agregarse a estos símbolos un grupo de hasta cuatro "0", para indicar el mínimo número de dígitos del exponente. El especificador "e+" muestra siempre el signo del exponente, ya sea positivo o negativo, pero el especificador "e-" sólo muestra el signo si es negativo.

'xx' ó "xx" Los caracteres encerrados por comillas simples o dobles se muestran tal como se indican, y no afectan al formato. Lo mismo ocurre si utilizamos cualquier caracter que no sea un especificador de la presente lista.

";" Separa las secciones para números positivos, negativos o ceros en la máscara.

Notas

- El número se presenta siempre redondeado a la máxima cantidad de decimales indicados por los "0" ó "#" a la derecha del separador decimal. Si la máscara no contiene el separador decimal, el número se redondeará al entero más cercano.

- Si el número tuviera más dígitos que los indicados en la máscara, estos dígitos se presentarán a la izquierda del primer "0" ó "#".

- Para permitir diferentes máscaras para valores positivos, negativos y ceros, la máscara puede contener entre una y tres secciones separadas por ";".
  - Una sección:
    La máscara se aplica a todos los valores
  - Dos secciones:
    La primera se aplica a valores positivos y ceros, y la segunda a negativos
  - Tres secciones:
    La primera se aplica a valores positivos, la segunda a negativos, y la tercera a ceros

- Si la máscara está vacía, o no se especifica máscara, el valor se presenta utilizando la siguiente máscara por defecto: ",0.#####". Es decir, con separador de miles, y con hasta cinco decimales, presentando siempre el cero pava valores menores a uno.

- En cualquier caso, si hubiera más de 18 dígitos a la izquierda del separador decimal, se presentará el número en notación científica, aunque no se haya especificado.

Ejemplos

Valores	**1234**	**-1234**	**0,5**	**0**
""	1234	-1234	0,5	0
"0"	1234	-1234	1	0
"0.00"	1234,00	-1234,00	0,50	0,00
"#.##"	1234	-1234	,5	
"#,##0.00"	1.234,00	-1.234,00	0,50	0,00
"#,##0.00;(#,##0.00)"	1.234,00	(1.234,00)	0,50	0,00
"#,##0.00;;Cero"	1.234,00	-1.234,00	0,50	Cero
"0.000E+00"	1,234E+03	-1,234E+03	5,000E-01	0,000E+00
"#.###E-0"	1,234E3	-1,234E3	5E-1	0E0

19. FORMATO DE MÁSCARAS PARA FECHAS

Una máscara para fechas es un texto que indica la forma de presentar una fecha, y puede estar formada por los siguientes especificadores:

d	Muestra el día como un número de uno o dos dígitos (1-31)
dd	Muestra el día como un número de dos dígitos (01-31)
ddd	Muestra el nombre abreviado del día de la semana (lun-dom)
dddd	Muestra el nombre completo del día de la semana (lunes-domingo)
ddddd	Muestra la fecha con formato dd/mm/yyyy
dddddd	Muestra la fecha usando el formato de fecha larga definido en Windows Normalmente: dddd, dd' de 'mmmm' de 'yyyy
m	Muestra el mes como un número de uno o dos dígitos (1-12)
mm	Muestra el mes como un número de dos dígitos (01-12)
mmm	Muestra el nombre abreviado del mes (ene-dic)
mmmm	Muestra el nombre completo del mes (enero-diciembre)
yy	Muestra el año como un número de dos dígitos (00-99)
yyyy	Muestra el año como un número de cuatro dígitos (0000-9999)
/	Muestra el separador de fecha que se haya definido en Windows (normalmente la barra, "/")
'xx' ó "xx"	Los caracteres encerrados por comillas simples o dobles se muestran tal como se indican, y no afectan al formato.

20. FORMATO DE MÁSCARAS PARA HORAS

Una máscara para horas es un texto que indica la forma de presentar una hora, y puede estar formada por los siguientes especificadores:

h	Muestra la hora del día como un número de uno o dos dígitos (0-23)
hh	Muestra la hora del día como un número de dos dígitos (00-23)
n	Muestra el minuto como un número de uno o dos dígitos (0-59)
nn	Muestra el minuto como un número de dos dígitos (00-59)
s	Muestra el segundo como un número de uno o dos dígitos (0-59)
ss	Muestra el segundo como un número de dos dígitos (00-59)
t	Muestra la hora con formato "hh:nn"
tt	Muestra la hora con el formato definido en Windows (normalmente "h:nn:ss")
am/pm	Usa formato de 12 horas con el especificador h o hh precedente, y muestra 'am' o 'pm' según corresponda. El am ó pm del resultado se mostrará en mayúsculas, minúsculas o ambas según estén en el especificador.
a/p	Usa formato de 12 horas con el especificador h o hh precedente, y muestra 'a' o 'p' según corresponda. El a ó p del resultado se mostrará en mayúsculas o minúsculas según esté el especificador.
:	Muestra el separador de hora que se haya definido en Windows (normalmente los dos puntos, ":")
'xx' ó "xx"	Los caracteres encerrados por comillas simples o dobles se muestran tal como se indican, y no afectan al formato.

21. FUNCIONES ESPECÍFICAS DE SQLCONTA

Función	Resultado	Descripción:
PreguntaMeses	Texto	Muestra un cuadro de diálogo solicitando los períodos, y devuelve una lista de períodos, que puede utilizarse en la función [ ] (ver más abajo)
DatoEmpresa(<t>)	Varía	Devuelve el valor del campo <t> de la tabla de configuración de la empresa
Formula(<n>)	Varía	Localiza la fórmula con el código <n> en la tabla de fórmulas, y la evalúa
Formula(<t>)	Varía	Localiza la fórmula con el nombre <t> en la tabla de fórmulas, y la evalúa
[<c>]	Número	Saldo de la lista de cuentas <c>
[D|H|S <c> <p>]	Número	Debe, haber o saldo de la lista de cuentas <c> en la lista de períodos <p>. Si no se especifica <p>, se asume el total del ejercicio
[N <c>]	Texto	Nombre de la cuenta <c>
[C <c>]	Número	Valor de la lista de casillas <c> (en impresos)

Listas de cuentas

Los elementos que conforman una lista de cuentas, son los siguientes:

<n>	Cuenta	Ejemplo: "400"
<n>-<n>	Rango de cuentas	Ejemplo: "400-430"

Puede anteponerse a una cuenta o rango uno de los siguientes modificadores:

-	Devuelve el resultado, cambiado de signo
P	Devuelve el resultado, sólo si es positivo
N	Devuelve el resultado, sólo si es negativo

La lista de cuentas estará formada por una o más cuentas o rangos, separados por ";".

Ejemplos

"430"	Cuenta 430
"400-430"	Cuentas de la 400 a la 430
"400-430;-472"	Cuentas de la 400 a la 430, más la cuenta 472 cambiada de signo
"P400-430;472"	Cuentas de la 400 a la 430 sólo si son positivas, más la cuenta 472

Listas de períodos

Los elementos que conforman una lista de períodos, son los siguientes:

ANT	: Resultado del ejercicio anterior
AP	: Asiento de apertura
1 a 12	: Meses del año, 1=Enero, 2=Febrero, ..., 12=Diciembre

REX : Asiento de regularización de existencias
REG : Asiento de regularización
CIE : Asiento de cierre

La lista de períodos puede estar formada por uno o más elementos, separados por ";", o por un rango de elementos, separados por "-".

Ejemplos

"AP;1;3"	: Asiento de apertura, enero, marzo
"AP-REG"	: Desde asiento de apertura hasta regularización
""	: Todos los períodos
";"	: Ningún período

22. FUNCIONES ESPECÍFICAS DE SQLPYME PARA IMPRESIÓN DE DOCUMENTOS

El sistema de impresión de documentos de Pymesoft proporciona dos objetos que permiten indicar fórmulas, definiendo todas las variables indicadas en la parte derecha del editor de formularios. Estos objetos son:

DocumentoFórmula

La fórmula se evaluará una vez para cada objeto y página. Si existen varias órdenes, separadas por ";", se procederá normalmente, devolviendo el resultado de la última orden. Las funciones adicionales disponibles en este contexto son:

DatoAdicional	Se explica más adelante

ArtículoFórmula

La fórmula se evaluará una vez para cada línea del documento. Si existen varias órdenes, separadas por ";", se evaluará cada orden por separado, agregando el resultado de cada una al cuerpo del documento. Las funciones adicionales disponibles en este contexto son:

DatoAdicional	Se explica más adelante
ArtículoCaracterística	Se explica más adelante
ArtículoDocOrigenDatoAdicional	Funciona como DatoAdicional, pero obtiene los datos del documento de origen de la línea actual
CambióOrigen	Si el origen de la línea actual ha cambiado con respecto a la anterior, la función devuelve Cierto, y si no, devuelve Falso
# F<n>	Cambia la fuente actual a la de índice <n>
# I<n>	Cambia la indentación actual a <n> centímetros
# S<n>	Inserta <n> líneas a esta columna, e inserta líneas en blanco a todas las demás columnas hasta quedar a la altura de ésta.
#ENTER	Inserta una línea en blanco
#ORIGEN	Si el origen de esta línea ha cambiado con respecto a la anterior, devuelve: "De <tipo > Nº <serie >-<número> del <fecha >:"

DatoAdicional(<entidad>;<formato>;<dato>)

Esta función permite obtener información sobre los datos adicionales de las entidades. Los parámetros son:

<entidad>: Un texto que indica la entidad de la se desea obtener información

Documento	Dato adicional del documento actual
Línea	Dato adicional de la línea actual del documento
Artículo	Dato adicional del artículo de la línea actual del documento
Cliente	Dato adicional del cliente o proveedor del documento actual
Obra	Dato adicional de la obra actual
ArtículoObra	Dato adicional del artículo de la línea actual de la obra

<formato>: Un texto que indica el formato del resultado. Las macros válidas son:

%n	El nombre del dato adicional
%v	El valor del dato adicional

<dato>: Un texto opcional que indica el nombre del dato adicional requerido. Si este parámetro no se indica, la función devuelve una lista con todos los datos existentes.

Ejemplos

DatoAdicional("Cliente"; "El valor de %n es %v") podría devolver la siguiente lista
- El valor de Ultima visita es 12/01/2001
- El valor de Garantía es Sí
- El valor de Importe de la póliza es 12.500

DatoAdicional("Artículo"; "%v"; "Garantía") podría devolver el siguiente valor
- 6 meses

Funciones equivalentes

Para una mayor comodidad de uso, se han definido las siguientes funciones equivalentes:

En documentos

DocumentoDatoAdicionalNombre(<dato>)	= DatoAdicional("Documento"; "%n"; <dato>)
DocumentoDatoAdicionalValor(<dato>)	= DatoAdicional("Documento"; "%v"; <dato>)
ClienteDatoAdicionalNombre(<dato>)	= DatoAdicional("Cliente"; "%n"; <dato>)
ClienteDatoAdicionalValor(<dato>)	= DatoAdicional("Cliente"; "%v"; <dato>)
ArtículoDatoAdicionalNombre(<dato>)	= DatoAdicional("Línea"; "%n"; <dato>)
ArtículoDatoAdicionalValor(<dato>)	= DatoAdicional("Línea"; "%v"; <dato>)

En órdenes de fabricación

DatoAdicionalNombre(<dato>)	= DatoAdicional("Obra"; "%n"; <dato>)
DatoAdicionalValor(<dato>)	= DatoAdicional("Obra"; "%v"; <dato>)
ArtículoDatoAdicionalNombre(<dato>)	= DatoAdicional("ArtículoObra"; "%n"; <dato>)
ArtículoDatoAdicionalValor(<dato>)	= DatoAdicional("ArtículoObra"; "%v"; <dato>)

ArtículoCaracterística(<formato>; <dato>; <dimensión1>; ...; <dimensiónN>; <orden>)

Esta función permite obtener información sobre las características indicadas en las líneas de los documentos. Los parámetros son:

<formato>: Un texto que indica el formato del resultado. Las macros válidas son:
- %n: El nombre de la característica. Ejemplo: "Lote" ó "Talla-Color"
- %v: El valor de la característica. Ejemplo: "20" ó "38-Negro"
- %p: El par nombre-valor. Ejemplo: "Lote 20" ó "Talla 38, Color Negro"
- %c: La cantidad indicada: Ejemplo: "5,2"
- %u: La unidad en la que se expresa dicha cantidad: Ejemplo: "Kg"

<dato>: Un texto opcional que indica el nombre de la característica requerida. Si este parámetro no se indica, la función devuelve una lista con todas las caracteristicas existentes en la línea del documento.

<dimensión1> ... <dimensiónN>: Textos opcionales que indican el valor requerido de cada una de las dimensiones de la característica. Si se omite alguna dimensión (o todas), la función devolverá todos los valores existentes en esa dimensión.

<orden>: Texto opcional que indica el nombre de la dimensión que devolverá y por la que se ordenará la lista resultante.

Ejemplos

ArtículoCaracterística("%p") podría devolver la siguiente lista
```
Talla 36, Color Rojo
Talla 36, Color Negro
Talla 38, Color Negro
Lote 18
Lote 20
Lote 22
```
ArtículoCaracterística("%p : %c", "Talla-Color") podría devolver la siguiente lista
```
Talla 36, Color Rojo : 12,5
Talla 36, Color Negro : 27,0
Talla 38, Color Negro : 11,4
```
ArtículoCaracterística("%n %v = %c %u", "Lotes") podría devolver la siguiente lista
```
Lote 18 = 5,2 Kg
Lote 20 = 12,4 Kg
Lote 22 = 0,7 Kg
```
ArtículoCaracterística("%p", "Talla-Color", "36") podría devolver la siguiente lista
```
Talla 36, Color Rojo
Talla 36, Color Negro
```
ArtículoCaracterística("%p", "Talla-Color", "", "Negro") podría devolver la siguiente lista
```
Talla 36, Color Negro
Talla 38, Color Negro
```
ArtículoCaracterística("%p", "Talla-Color", "", "", "Color") podría devolver la siguiente lista
```
Talla 36, Color Negro
Talla 38, Color Negro
Talla 36, Color Rojo
```

Funciones equivalentes

Para una mayor comodidad de uso, se han definido las siguientes funciones equivalentes:

```
ArtículoCaracterísticaNombre(...)      = ArtículoCaracterística("%n"; ...)
ArtículoCaracterísticaValor(...)       = ArtículoCaracterística("%v"; ...)
ArtículoCaracterísticaPar(...)         = ArtículoCaracterística("%p"; ...)
ArtículoCaracterísticaCantidad(...)    = ArtículoCaracterística("%c"; ...)
ArtículoCaracterísticaUnidad(...)      = ArtículoCaracterística("%u"; ...)
```

23. EJEMPLOS PRÁCTICOS DE SCRIPTS

En la siguiente relación hacemos mención a algunos ejemplos que se pueden realizar con los scripts para poder visualizar su funcionamiento y la potencia que alcanzan.

MOSTRAR LA UBICACIÓN DEL ALMACEN EN LOS FORMATOS DE IMPRESIÓN.

Dentro de los formatos de impresión tenemos una serie de variables para imprimir. Puede ocurrir que alguno de los campos que quiero imprimir no esté dentro de esas variables. Ahora con los scripts nos permite leer cualquier campo de cualquier tabla por lo que ya no necesitaré que exista como variable.
Un ejemplo sería sacar la ubicación del almacén de los artículos. Esa variable no existe en el formato por lo que haríamos el siguiente script:

```
Leer("Existenc"; "Ubicacion"; "CodArticulo='" + ArtículoCódigo +
"' and CodAlmacen = '"+ EnTexto(ArtículoAlmacénCódigo) + "'");
```

Este script lo tendríamos que poner dentro de la variable ArtículoFórmula del formato de impresión y en la posición que quisiéramos.

En el indicamos que tabla queremos leer en este caso Existenc que es la tabla de almacén, el campo que queremos leer la Ubicación, y la restricción que hacemos que será que codarticulo que es de la tabla de existencias es igual al campo artículocodigo del formato de impresión y lo mismo con el codalmacen que es igual a la variable artículoalmacencódigo.

RELLENAR UN DATO ADICIONAL DE CABECERA CON LA PERSONA DE CONTACTO DEL CLIENTE

Podemos crear un dato adicional de cabecera, por ejemplo le llamamos Contacto y que ese campo se rellene automáticamente con un dato de otra tabla, por ejemplo el campo persona de contacto de la tabla de clientes.
Este script lo colocaríamos en la casilla del cliente del formato de edición en salida para que cuando salga de esa casilla me rellene el dato adicional. El script sería el siguiente:

```
Cod_Cliente=Leer("DocCab";"Actual.CodCliente";"");
PersonaContacto=Leer("Domicili";"PersonaContacto";"Tipo=0 and Codtipo=1 and
codigo=" + Formatea("#";Cod_Cliente));
Escribir("CarValor";"Actual.Contacto";PersonaContacto;"");
```

En este script lo que hacemos es dividir en dos, creando 1º las variable y luego lo que sería el propiamente el script con las variables que hemos creado.

Creamos la variable codcliente donde le decimos que lea el codigo actual del cliente de la tabla de cabecera de documentos.
A continuación creamos persona de contacto y también que lo lea pero de la tabla de domicilios que es donde se encuentra la persona de contacto. Por tanto leemos domicili, lo siguiente el campo que queremos leer PersonaContacto y a continuación ponemos la restricción que este caso le decimos que tipo=0 que significa que es de clientes el domicilio(si fuera 1 sería para proveedores) el codtipo=1 que es la 1ª dirección que hemos creado y le decimos que el codigo de la dirección es igual al codigo del cliente que fue la 1ª variable que creamos.
Por último queremos que ese dato que estamos leyendo lo escriba en el dato adicional que hemos creado, por lo que hacemos la sentencia escribir, ponemos en que tabla, en este caso carvalor, como se llama el campo(contacto) y le ponemos actual porque queremos que nos rellene el campo del documento que estamos haciendo no el de todos. Y por último el valor del dato que queremos escribir que sería la variable anteriormente calculada. Las últimas comillas serían par si queremos hacer un filtro que en este caso no es necesario.

CREAR UNA SERIE DISTINTA POR CADA PROVEEDOR

Otra opción que tenemos es que en función del código del proveedor me asigne automáticamente una serie para controlar la facturación o numeración de las facturas de mis proveedores.
El script lo colocaría a la salida de la casilla del proveedor.En este caso el script sería el siguiente:

```
Codcliente=leer("Doccab";"Actual.Codcliente";"");
Escribir("DocCab";"Actual.Serie";EnTexto(codcliente);"");
```

Sería un script muy sencillo donde sólo leemos el código del proveedor y lo escribimos en la serie actual.Este script se colocaría en la salida del campo del proveedor para que al salir del campo rellene la serie.

MOSTRAR PRECIO STANDARD EN LOS DOCUMENTOS

Otro ejemplo que podríamos tener es en los documentos de venta. Cuando metemos un cliente nos sale el precio que le tenemos estipulados y nos puede interesar además visualizar el precio Standard del artículo. Para ello podemos crear un dato adicional que sea Precio Standard donde ponga automáticamente el precio de la ficha del artículo cada vez que venda uno.

El script sería el siguiente:

```
cod_articulo=Leer("DocLin";"Actual.Codarticulo";"");
precio=Leer("articulo";"precioventa";"codigo='" + cod_articulo+"'");
Escribir("DocCar";"Actual.Precio Standard";Formatea(",0.00";precio);"");
```

Este script lo pondríamos a la salida del campo de artículo, nombre y código de barras para que si busco un artículo por cualquier de esos campos al salir me rellene el dato adicional.

CONSERVAR EL PRECIO QUE SE MODIFICA

Otra opción que nos da los scripts se refiere a conservar el precio en un documento cuando se modifica. Si se le cambia manualmente el precio en un documento, cuando vuelvo a modificar la cantidad vuelve a recalcularlo para ver si tiene condiciones. Para evitar que una vez modificado el precio lo cambie cuando le introduzca la cantidad podría realizar los siguiente:
En la casilla de cantidad del formato de edición en entrada pondría lo siguiente:

```
Valor_anterior=Leer("Doclin";"Actual.Precio";"");
```

Y en la casilla de salida:

```
Escribir("Doclin";"Actual.Precio";EnTexto(valor_anterior);"");
```

De esa manera conservaría el precio que hubiese variado.

ETIQUETAS CON PRECIOS POR CARACTERÍSTICAS

Este script esta en un formato de impresión de etiquetas. Sirve para sacar el precio por caracteristica.

```
cod_tipo_iva=Leer("Articulo";"TipoIva";"codigo='" + Código+"'");
campo_iva= "IVA" + EnTexto(cod_tipo_iva+1);
porcentaje_iva=Leer("Empresa";campo_iva;"");
si Característica<>""  y CuantosRegistros("venta"; "codarticulo='" + Código + "' and codcaract=2 and valorcaract='"
+ Característica + "')<>0 entonces {Formatea(",0.00";Leer("venta";"precio";"codarticulo='" + Código + "' and codcaract<>" and valorcaract='"
+ Característica + "')*((porcentaje_iva/100)+1))} sino {Formatea(",0.00";PrecioConIva)};
```

ESTADO DE UN DOCUMENTO

El siguiente script pone a SI un dato adicional de albarán (**FACTURADO**), para poder luego realizar busquedas por dato adicional de aquellos documentos que tengan el valor del dato adicional a NO.

En Archivo / Tablas Auxiliares creamos un dato adicional de documento que se llama ESTADO y de tipo texto, indicamos que lo solicite en albaranes.

En la configuración del formato de edicion de albaranes (con el botón derecho sobre el albaran vamos a la opcion de configurar), ponemos el dato adicional ESTADO a visible y como valor por omision NO. Con esto conseguiremos que cada vez que creemos un albarán este se visualize en la lista.

Una vez configurado el formato de edición del albarán pasamos a la configuración del formato de impresión de la factura, añadiendo el script correspondiente, para que a la hora de **reemitir la factura**, **pasar albaranes a factura** o realizar el **proceso de facturación de albaranes** este se ejecute y por lo tanto se marque el albarán. Deberemos de tener en cuenta que si disponemos de varios formatos de factura este script deberá añadirse en todos.

Deberemos tener en cuenta que si realizamos un proceso de **anulación de emisión de facturas**, el dato adicional ESTADO seguira teniendo el valor SI, pues el script no se ejecutará.

El siguiente script se introducirá dentro de un campo artículo formula, en principio el usado para imprimir la descripción del articulo. Podremos dar de alta si asi lo deseamos un nuevo campo ArtículoFormula:

```
Escribir("CarValor";"Valor";"'SI'";"codclase=1 and codcaract=5 and codobjeto='" +
EnTexto(Formatea("0";Leer("DocCab";"Codigo";"serie=" +
"'" + ArtículoDocOrigenSerie + "'" +
" and numero=" + EnTexto(ArtículoDocOrigenNúmero) + " and tipo=2")))+"'");
```

Donde:

CodCaract; Es el codigo del dato adicional ESTADO.

Tipo; Especifica el tipo del documentos (este script se creo para aquellos albaranes que se pasen a pedido, si deseamos ampliar esta configuación a otro tipo de documentos deberemos de tener en cuenta este campo).

SUMA DE UN DATO ADICIONAL DE LÍNEA

Este script va acumulando el valor de un dato adicional de linea en un dato adicional de documento
El script lo introducimos en el dato adicional de línea.
Tendremos un dato adicional de cabecera que es peso total y un dato adicional de línea que será peso. De esta manera el peso que vayamos introduciendo en cada línea lo irá sumando y lo pondrá en la casilla de la cabecera como peso total.

En la entrada del dato adicional de línea pondremos :

```
peso_anterior=Leer("DocCar";"Actual.PESO";"");
peso_total=Leer("CarValor";"Actual.PESO TOTAL";"");
```

Y en la salida:

```
peso_articulo=Leer("DocCar";"Actual.PESO";"");
peso_total=peso_total-peso_anterior+peso_articulo;
Escribir("CarValor";"Actual.PESO TOTAL"; Formatea("0.00";peso_total);"");
```

OBTENER DATOS CON LEER USANDO POSICIONAMIENTO RELATIVO

Para obtener el número de líneas del documento actual sería así:
Leer(“DOCLIN”;”Actual.CuantasLíneas”;“”)

Para obtener la posición de la línea en la que estamos:
Leer(“DOCLIN”;”Actual.NúmeroLínea”;””)

Para leer el nombre de la línea 5 (siempre se empieza a contar en 1):
Leer(“DOCLIN”;”Actual.Nombre”;”5”) %% Debe pasarse la línea como tipo texto

Para leer el nombre de 5 líneas a partir de la actual:
Leer(“DOCLIN”;”Actual.Nombre”;”+5”) %% Debe pasarse la posición como tipo texto