

# TI-Nspire™ CX

## Guía de Referencia

## ***Información importante***

Excepto por lo que se establezca expresamente en contrario en la Licencia que se incluye con el programa, Texas Instruments no otorga ninguna garantía, ni expresa ni implícita, incluidas pero sin limitarse a cualquier garantía implícita de comerciabilidad e idoneidad con un propósito en particular, en relación con cualquier programa o material impreso, y hace dichos materiales disponibles únicamente "tal y como se encuentran". En ningún caso Texas Instruments será responsable en relación con ninguna persona de daños especiales, colaterales, incidentales o consecuenciales en conexión con o que surjan de la compra o el uso de estos materiales, y la responsabilidad única y exclusiva de Texas Instruments, independientemente de la forma de acción, no excederá la cantidad estipulada en la licencia para el programa. Asimismo, Texas Instruments no será responsable de ninguna reclamación de ningún tipo en contra del uso de estos materiales por parte de cualquier otro individuo.

© 2023 Texas Instruments Incorporated

Los productos reales pueden ser ligeramente distintos de las imágenes proporcionadas.

## Índice de contenido

<b>Plantillas de expresiones</b> .....	<b>1</b>
<b>Listado alfabético</b> .....	<b>7</b>
A .....	7
B .....	16
C .....	20
D .....	37
E .....	47
F .....	55
G .....	63
I .....	74
L .....	82
M .....	97
N .....	106
O .....	116
P .....	118
Q .....	126
R .....	129
S .....	145
T .....	166
U .....	179
V .....	179
W .....	180
X .....	183
Z .....	184
<b>Símbolos</b> .....	<b>191</b>
<b>TI-Nspire™ CX II: comandos para dibujar</b> .....	<b>215</b>
Cómo programar gráficos .....	215
Pantalla de gráficos .....	215
Vista y configuraciones predeterminadas .....	216
Mensajes de errores de la pantalla de gráficos .....	217
Comandos no válidos mientras está en modo de gráficos .....	217
C .....	219
D .....	220
F .....	223
G .....	225
P .....	226
S .....	228
U .....	230

<b>Elementos vacíos (inválidos)</b> .....	<b>231</b>
<b>Accesos directos para ingresar expresiones matemáticas</b> .....	<b>233</b>
<b>Jerarquía de EOS™ (Sistema Operativo de Ecuaciones)</b> .....	<b>235</b>
<b>Características de programación de TI-Nspire CX II - TI-Basic</b> .....	<b>237</b>
Sangría automática en el editor de programación .....	237
Mensajes de error mejorados para TI-Basic .....	237
<b>Constantes y valores</b> .....	<b>240</b>
<b>Códigos y mensajes de error</b> .....	<b>241</b>
<b>Códigos de advertencia y mensajes</b> .....	<b>250</b>
<b>Información general</b> .....	<b>252</b>
<b>Índice alfabético</b> .....	<b>253</b>

## Plantillas de expresiones

Las plantillas de expresiones ofrecen una manera fácil de ingresar expresiones matemáticas en una notación matemática estándar. Cuando se inserta una plantilla, ésta aparece en la línea de ingreso con pequeños bloques en las posiciones donde se pueden ingresar elementos. Un cursor muestra cuál elemento se puede ingresar.

Use las teclas de flechas o presione **tab** para mover el cursor a cada posición del elemento, y escriba un valor o una expresión para el elemento. Presione **enter** o **ctrl enter** para evaluar la expresión.

### Plantilla de fracciones

**ctrl** **÷** teclas



**Nota:** Vea también / (dividir), página 193.

Ejemplo:

$$\frac{12}{8 \cdot 2} \qquad \frac{3}{4}$$

### Plantilla de exponentes

**^** teclas



**Nota:** Escriba el primer valor, presione **^** y después escriba el exponente. Para regresar el cursor a la línea base, presione la flecha derecha (►).

**Nota:** Vea también ^ (potencia), página 194.

Ejemplo:

$$2^3 \qquad 8$$

### Plantilla de raíz cuadrada

**ctrl** **x<sup>2</sup>** teclas



**Nota:** Vea también  $\sqrt{()}$  (raíz cuadrada), página 203.


Ejemplo:

$$\sqrt{4} \qquad 2$$
$$\sqrt{\{9,16,4\}} \qquad \{3,4,2\}$$

## Plantilla de raíz enésima

  teclas



 **Nota:** Vea también `root()`, página 141.

Ejemplo:

$\sqrt[3]{8}$	2
$\sqrt[3]{\{8,27,15\}}$	{2,3,2.46621}

## e plantilla de exponentes

 tecla





Exponencial natural  $e$  elevado a una potencia

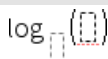
**Nota:** Vea también `e^()`, página 47.

Ejemplo:

$e^1$	2.71828182846
-------	---------------

## Plantilla de logística

  tecla



Calcula la logística para una base especificada. Para un predeterminado de base 10, omitir la base.

**Nota:** Vea también `logistic()`, página 93.

Ejemplo:

$\log_4(2)$	0.5
-------------	-----

## Plantilla de compuesto de variables (2 piezas)

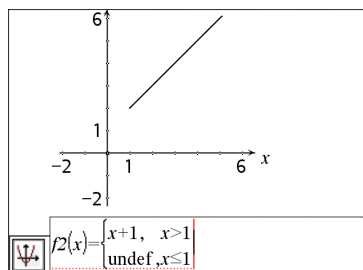
Catálogo > 



Permite crear expresiones y condiciones para una función de compuesto de variables de dos-piezas. Para agregar una pieza, haga clic en la plantilla y repita la plantilla.

**Nota:** Vea también `piecewise()`, página 120.

Ejemplo:



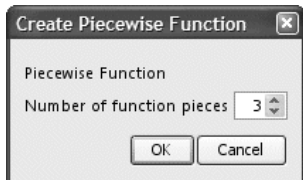
## Plantilla de compuesto de variables (N piezas)

Catálogo > 

Permite crear expresiones y condiciones para una función de compuesto de variables de  $N$ -piezas. Indicadores para  $N$ .

Ejemplo:

Vea el ejemplo de plantilla de compuesto de variables (2 piezas).



**Nota:** Vea también `piecewise()`, página 120.

## Sistema de plantilla de 2 ecuaciones

Catálogo > 



Creará un sistema de dos ecuaciones lineales. Para agregar una fila a un sistema existente, haga clic en la plantilla y repita la plantilla.

Ejemplo:

$$\text{solve} \left( \begin{cases} x+y=0 \\ x-y=5 \end{cases}, x, y \right) \quad x = \frac{5}{2} \text{ and } y = \frac{-5}{2}$$

$$\text{solve} \left( \begin{cases} y=x^2-2 \\ x+2 \cdot y=-1 \end{cases}, x, y \right) \\ x = \frac{-3}{2} \text{ and } y = \frac{1}{4} \text{ or } x=1 \text{ and } y=-1$$

**Nota:** Vea también `system()`, página 165.

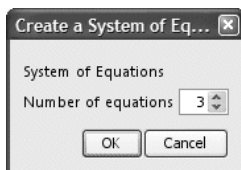
## Sistema de plantilla de N ecuaciones

Catálogo > 

Permite crear un sistema de  $N$  ecuaciones lineales. Indicadores para  $N$ .

Ejemplo:


Vea el ejemplo de Sistema de plantilla de ecuaciones (2 piezas).



**Nota:** Vea también `system()`, página 165.

## Plantilla de valor absoluto

Catálogo > 

 **Nota:** Vea también **abs()**, página 7.

Ejemplo:

$$\left| \{2, -3, 4, -4^3\} \right| \quad \{2, 3, 4, 64\}$$

## plantilla gg°mm'ss.ss''

Catálogo > 



Permite ingresar ángulos en el formato **gg°mm'ss.ss''**, donde **gg** es el número de grados decimales, **mm** es el número de minutos y **ss.ss** es el número de segundos.

Ejemplo:

$$30^{\circ}15'10'' \quad 0.528011$$

## Plantilla de matriz (2 x 2)

Catálogo > 



Crea una matriz de 2 x 2

Ejemplo:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 5 \quad \begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix}$$

## Plantilla de matriz (1 x 2)

Catálogo > 



Ejemplo:

$$\text{crossP}(\begin{bmatrix} 1 & 2 \end{bmatrix}, \begin{bmatrix} 3 & 4 \end{bmatrix}) \quad \begin{bmatrix} 0 & 0 & -2 \end{bmatrix}$$

## Plantilla de matriz (2 x 1)

Catálogo > 



Ejemplo:

$$\begin{bmatrix} 5 \\ 8 \end{bmatrix} \cdot 0.01 \quad \begin{bmatrix} 0.05 \\ 0.08 \end{bmatrix}$$

## Plantilla de matriz (m x n)

Catálogo > 

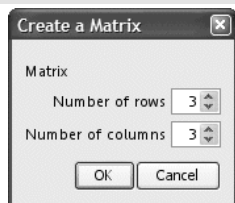
La plantilla aparece después de que se le indica especificar el número de filas y columnas.

Ejemplo:



## Plantilla de matriz (m x n)

Catálogo > 



$$\text{diag} \begin{pmatrix} 4 & 2 & 6 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{pmatrix} \quad [4 \ 2 \ 9]$$

**Nota:** Si se crea una matriz con un número grande de filas y columnas, puede llevarse unos cuantos segundos en aparecer.

## Plantilla de suma ( $\Sigma$ )

Catálogo > 

$$\sum_{i=0}^{} (i)$$

Ejemplo:

$$\sum_{n=3}^7 (n) \quad 25$$

**Nota:** Vea también  $\Sigma()$  (**sumaSec**), página 204.

## Plantilla de producto ( $\Pi$ )

Catálogo > 

$$\prod_{i=0}^{} (i)$$

Ejemplo:

$$\prod_{n=1}^5 \left( \frac{1}{n} \right) \quad \frac{1}{120}$$

**Nota:** Vea también  $\Pi()$  (**prodSec**), página 204.

## Plantilla de primera derivada

Catálogo > 

$$\frac{d}{dx} (x)$$

Ejemplo:

$$\frac{d}{dx} (|x|)_{x=0} \quad \text{undef}$$

## Plantilla de primera derivada

Catálogo > 

La plantilla de primera derivada se puede usar para calcular la primera derivada en un punto numéricamente, usando métodos de autodiferenciación.

**Nota:** Vea también **d()** (**derivada**), página 202.

## Plantilla de segunda derivada

Catálogo > 

$$\frac{d^2}{dx^2}(\square)$$

Ejemplo:

---

$$\frac{d^2}{dx^2}(x^3)|_{x=3} \quad 18$$

---

La plantilla de segunda derivada se puede usar para calcular la segunda derivada en un punto numéricamente, usando métodos de autodiferenciación.

**Nota:** Vea también **d()** (**derivada**), página 202.

## Plantilla de integral definida

Catálogo > 

$$\int_a^b \square dx$$

Ejemplo:

---

$$\int_0^{10} x^2 dx \quad 333.333$$

---

La plantilla de integral definida se puede usar para calcular la integral definida numéricamente, usando el mismo método que con `nint()`.

**Nota:** Vea también `nint()`, página 110.

## Listado alfabético

Los elementos cuyos nombres no son alfabéticos (como +, ! y >) se enumeran al final de esta sección, comenzando (página 191). A menos que se especifique lo contrario, todos los ejemplos en esta sección se realizaron en el modo de reconfiguración predeterminado, y se supone que todas las variables no están definidas.

### A

#### abs()

Catálogo > 

**abs**(*ValorI*)⇒*valor*

$$\left| \left\{ \frac{\pi}{2}, \frac{\pi}{3} \right\} \right| \quad \{1.5708, 1.0472\}$$

**abs**(*ListaI*)⇒*lista*

$$|2-3 \cdot i| \quad 3.60555$$

**abs**(*MatrizI*)⇒*matriz*

Entrega el valor absoluto del argumento.

**Nota:** Vea también **Plantilla de valor absoluto**, página 4.

Si el argumento es un número complejo, entrega el módulo del número.

#### amortTbl() (tablaAmort)

Catálogo > 

**amortTbl**(*NPgo*,*N*,*I*,*VP*, [*Pgo*], [*VF*], [*PpA*], [*CpA*], [*PgoAl*], [*valorRedondo*])⇒*matriz*

amortTbl(12,60,10,5000,,12,12)

0	0.	0.	5000.
1	-41.67	-64.57	4935.43
2	-41.13	-65.11	4870.32
3	-40.59	-65.65	4804.67
4	-40.04	-66.2	4738.47
5	-39.49	-66.75	4671.72
6	-38.93	-67.31	4604.41
7	-38.37	-67.87	4536.54
8	-37.8	-68.44	4468.1
9	-37.23	-69.01	4399.09
10	-36.66	-69.58	4329.51
11	-36.08	-70.16	4259.35
12	-35.49	-70.75	4188.6

La función de amortización que entrega una matriz como una tabla de amortización para un conjunto de argumentos de TVM.

*NPgo* es el número de pagos a incluirse en la tabla. La tabla comienza con el primer pago.

*N*, *I*, *VP*, *Pgo*, *VF*, *PpA*, *CpA*, and *PgoAl* se describen en la tabla de argumentos de VTD, página 176.

- Si se omite *Pgo*, se predetermina a  $Pgo = \text{tvmPmt}(N, I, VP, VF, PpA, CpA, PgoAl)$ .
- Si se omite *VF*, se predetermina a  $VF = 0$ .
- Los predeterminados para *PpA*, *CpA* y *PgoAl* son los mismos que para las

funciones de TVM.

*valorRedondo* especifica el número de lugares decimales para el redondeo. Predeterminado=2.

Las columnas en la matriz de resultado están en este orden: Número de pago, cantidad pagada a interés, cantidad pagada a capital y balance.

El balance desplegado en la fila  $n$  es el balance después del pago  $n$ .

Se puede usar la matriz de salida como entrada para las otras funciones de amortización  $\Sigma\text{Int}()$  y  $\Sigma\text{Prn}()$ , página 205y **bal()**, página 16.

## and (y)

*ExprBooleana1 and ExprBooleana2*  $\Rightarrow$  *expresión Booleana*

*ListaBooleana1 and ListaBooleana2*  $\Rightarrow$  *Lista Booleana*

*MatrizBooleana1 and MatrizBooleana2*  $\Rightarrow$  *Matriz Booleana*

Entrega verdadero o falso o una forma simplificada del ingreso original.

*Entero1 and Entero2*  $\Rightarrow$  *entero*

Compara dos enteros reales bit por bit usando una operación **y**. En forma interna, ambos enteros se convierten en números binarios de 64 bits firmados. Cuando se comparan los bits correspondientes, el resultado es 1 si ambos bits son 1; de otro modo, el resultado es 0. El valor producido representa los resultados de los bits, y se despliega de acuerdo con el modo de Base.

En modo de base hexadecimal:

0h7AC36 and 0h3D5F	0h2C16
--------------------	--------

Importante: Cero, no la letra O.

En modo de base binaria:

0b100101 and 0b100	0b100
--------------------	-------

En modo de base decimal:

37 and 0b100	4
--------------	---

## and (y)

Catálogo > 

Se pueden ingresar enteros en cualquier base de números. Para un ingreso binario o hexadecimal, se debe usar el prefijo 0b ó 0h, respectivamente. Sin un prefijo, los enteros se tratan como decimales (base 10).

**Nota:** Un ingreso binario puede tener hasta 64 dígitos (sin contar el prefijo 0b). Un ingreso hexadecimal puede tener hasta 16 dígitos.

## angle()

Catálogo > 

**angle(Valor1)**⇒*valor*

Entrega el ángulo del argumento, interpretando el argumento como un número complejo.

En modo de ángulo en Grados:

---

$$\text{angle}(0+2\cdot i) \qquad 90$$

---

En modo de ángulo en Gradianes:

---

$$\text{angle}(0+3\cdot i) \qquad 100$$

---

En modo de ángulo en Radianes:

---

$$\text{angle}(1+i) \qquad 0.785398$$

$$\text{angle}(\{1+2\cdot i, 3+0\cdot i, 0-4\cdot i\})$$

---

$$\{1.10715, 0., -1.5708\}$$

---

**angle(Lista1)**⇒*lista*

**angle(Matriz1)**⇒*matriz*

Entrega una lista o matriz de ángulos de los elementos en *Lista1* o *Matriz1*, interpretando cada elemento como un número complejo que representa un punto de coordenada bidimensional o rectangular.

## ANOVA

Catálogo > 

**ANOVA** *Lista1, Lista2[, Lista3, ..., Lista20]*  
*[, Bandera]*

Realiza un análisis unidireccional de la varianza para comparar las medias de dos a 20 poblaciones. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

*Bandera*=0 para Datos, *Bandera*=1 para Estadísticas

Variable de salida	Descripción
stat.F	Valor de F estadístico
stat.ValP	Nivel más bajo de significancia en el cual la hipótesis nula se puede rechazar
stat.df	Grados de libertad de los grupos
stat.SS	Suma de cuadrados de los grupos
stat.MS	Cuadrados medios de los grupos
stat.dfError	Grados de libertad de los errores
stat.SSError	Suma de cuadrados de los errores
stat.MSError	Cuadrado medio de los errores
stat.sp	Desviación estándar agrupada
stat.xbarlista	Media de la entrada de las listas
stat.ListaCBajo	95% de intervalos de confianza para la media de cada lista de entrada
stat.ListaCALto	95% de intervalos de confianza para la media de cada lista de entrada

## ANOVA2way (ANOVA2vías)

**ANOVA2way** *Lista1,Lista2*  
[*Lista3,...,Lista10*][*LevRow*]

Genera un análisis bidireccional de la varianza para comparar las medias de dos a 10 poblaciones. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

*LevRow*=0 para bloque

*LevRow*=2,3,...,*Len*-1, para factor dos, donde *Len*=largo(*Lista1*)=largo(*Lista2*) = ... = largo(*Lista10*) y *Len* / *LevRow* ∈ {2,3,...}

Salidas: Diseño de bloque

Variable de salida	Descripción
stat.F	F estadístico del factor de columna

Variable de salida	Descripción
stat.ValP	Nivel más bajo de significancia en el cual la hipótesis nula se puede rechazar
stat.df	Grados de libertad del factor de columna
stat.SS	Suma de cuadrados del factor de columna
stat.MS	Cuadrados medios para el factor de columna
stat.BloqF	F estadístico para el factor
stat.BloqValP	Probabilidad más baja a la cual la hipótesis nula se puede rechazar
stat.dfbloque	Grados de libertad del factor
stat.SSBloque	Suma de cuadrados para el factor
stat.MSBloque	Cuadrados medios para el factor
stat.dfError	Grados de libertad de los errores
stat.SSError	Suma de cuadrados de los errores
stat.MSError	Cuadrados medios para los errores
stat.s	Desviación estándar del error

#### Salidas del FACTOR DE COLUMNA

Variable de salida	Descripción
stat.Fcol	F estadístico del factor de columna
stat.ValPCol	Valor de probabilidad del factor de columna
stat.dfCol	Grados de libertad del factor de columna
stat.SSCol	Suma de cuadrados del factor de columna
stat.MSCol	Cuadrados medios para el factor de columna

#### Salidas del FACTOR DE FILAS

Variable de salida	Descripción
stat.FFila	F estadístico del factor de fila
stat.ValPFila	Valor de probabilidad del factor de fila
stat.dfFila	Grados de libertad del factor de fila
stat.SSFila	Suma de cuadrados del factor de fila
stat.MSFila	Cuadrados medios para el factor de fila



## Salidas de INTERACCIÓN

Variable de salida	Descripción
stat.FInterac	F estadístico de la interacción
stat.ValPInterac	Valor de probabilidad de la interacción
stat.dfInterac	Grados de libertad de la interacción
stat.SSInterac	Suma de cuadrados de la interacción
stat.MSInterac	Cuadrados medios para la interacción

## Salidas de ERROR

Variable de salida	Descripción
stat.dfError	Grados de libertad de los errores
stat.SSError	Suma de cuadrados de los errores
stat.MSError	Cuadrados medios para los errores
s	Desviación estándar del error

### Ans

  **teclas**

**Ans**⇒*valor*


56 56

Entrega el resultado de la expresión evaluada más recientemente.

56+4 60



60+4 64

### approx()

**Catálogo** > 

**approx**(*Valor1*)⇒*número*

Entrega la evaluación del argumento como una expresión que contiene valores decimales, cuando es posible, independientemente del modo **Auto** o **Aproximado** actual.

Esto es equivalente a ingresar el argumento y presionar  .

$\text{approx}\left(\frac{1}{3}\right)$  0.333333

$\text{approx}\left(\left\{\frac{1}{3}, \frac{1}{9}\right\}\right)$  {0.333333, 0.111111}

$\text{approx}\{\{\sin(\pi), \cos(\pi)\}\}$  {0., -1.}

$\text{approx}\left(\left[\sqrt{2}, \sqrt{3}\right]\right)$  [1.41421 1.73205]

$\text{approx}\left(\left[\frac{1}{3}, \frac{1}{9}\right]\right)$  [0.333333 0.111111]

**approx**(*Lista1*)⇒*lista*

$\text{approx}\{\{\sin(\pi), \cos(\pi)\}\}$  {0., -1.}

**approx**(*Lista1*)⇒*lista*

$\text{approx}\left(\left[\sqrt{2}, \sqrt{3}\right]\right)$  [1.41421 1.73205]



**approx()**

Catálogo &gt;

Entrega una lista o *matriz* donde cada elemento se ha evaluado a un valor decimal, cuando es posible.

**▶approxFraction()**

Catálogo &gt;

*Valor* ▶`approxFraction([Tol])⇒valor`

*Lista* ▶`approxFraction([Tol])⇒lista`

*Matriz* ▶`approxFraction([Tol])⇒matriz`

Entrega la entrada como una fracción, usando una tolerancia de *Tol*. Si *Tol* se omite, se usa una tolerancia de 5.E-14.

**Nota:** Se puede insertar esta función desde el teclado de la computadora al escribir `@>approxFraction(...)`.

$\frac{1}{2} + \frac{1}{3} + \tan(\pi)$	0.833333
<code>0.8333333333333333▶approxFraction(5.E-14)</code>	$\frac{5}{6}$
<code>{π,1.5}▶approxFraction(5.E-14)</code>	$\left\{ \frac{5419351}{1725033}, \frac{3}{2} \right\}$

**approxRational()**

Catálogo &gt;

`approxRational(Valor[, Tol])⇒valor`

`approxRational(Lista[, Tol])⇒lista`

`approxRational(Matriz[, Tol])⇒matriz`

Entrega el argumento como una fracción usando una tolerancia de *Tol*. Si *Tol* se omite, se usa una tolerancia de 5.E-14.

<code>approxRational(0.333,5·10<sup>-5</sup>)</code>	$\frac{333}{1000}$
<code>approxRational({0.2,0.33,4.125},5.E-14)</code>	$\left\{ \frac{1}{5}, \frac{33}{100}, \frac{33}{8} \right\}$

**arccos()**Vea  $\cos^{-1}()$ , página 28.**arccosh()**Vea  $\cosh^{-1}()$ , página 29.**arccot()**Vea  $\cot^{-1}()$ , página 30.

**arccoth()** Vea  $\text{coth}^{-1}()$ , página 31.

**arccsc()** Vea  $\text{csc}^{-1}()$ , página 34.

**arccsch()** Vea  $\text{csch}^{-1}()$ , página 34.

**arcsec()** Vea  $\text{sec}^{-1}()$ , página 145.


**arcsech()** Vea  $\text{sech}()$ , página 146.

**arcsin()** Vea  $\text{sin}()$ , página 154.

**arcsinh()** Vea  $\text{sinh}()$ , página 156.

**arctan()** Vea  $\text{tan}()$ , página 167.

**arctanh()** Vea  $\text{tanh}()$ , página 168.

**augment()** Catálogo > 

**augment(Lista1, Lista2)⇒lista**

$\text{augment}(\{1, -3, 2\}, \{5, 4\})$	$\{1, -3, 2, 5, 4\}$
--	----------------------

Entrega una nueva lista que es *Lista2* adjuntada al final de *Lista1*.

**augment()**Catálogo > **augment**(*Matriz1*, *Matriz2*)⇒*matriz*

Entrega una nueva matriz que es *Matriz2* adjuntada a *Matriz1*. Cuando se usa el caracter “,” las matrices deben tener dimensiones de fila iguales, y *Matriz2* se adjunta a *Matriz1* como nuevas columnas. No altera *Matriz1* o *Matriz2*.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
$\begin{bmatrix} 5 \\ 6 \end{bmatrix} \rightarrow m2$	$\begin{bmatrix} 5 \\ 6 \end{bmatrix}$
$\text{augment}(m1,m2)$	$\begin{bmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \end{bmatrix}$

**avgRC()**Catálogo > **avgRC**(*Expr1*, *Var* [=Valor] [, *Paso*])⇒*expresión***avgRC**(*Expr1*, *Var* [=Valor] [, *Lista1*])⇒*lista***avgRC**(*Lista1*, *Var* [=Valor] [, *Paso*])⇒*lista***avgRC**(*Matriz1*, *Var* [=Valor] [, *Paso*])⇒*matriz*

Entrega el cociente diferencial progresivo (tasa de cambio promedio).

*Expr1* puede ser un nombre de función definido por el usuario (vea **Func**).

Cuando se especifica el *Valor*, se eliminan todas las asignaciones anteriores de la variable o cualquier sustitución "|" para la variable.

*Paso* es el valor del paso. Si se omite *Paso* se predetermina a 0.001.

Tome en cuenta que la función similar **centralDiff()** usa el cociente diferencial central.

$x:=2$	2
$\text{avgRC}(x^2-x+2,x)$	3.001
$\text{avgRC}(x^2-x+2,x,1)$	3.1
$\text{avgRC}(x^2-x+2,x,3)$	6

**bal()**

**bal**(*NPgo*,*N*,*I*,*VP*, [*Pgo*], [*VF*], [*PpA*], [*CpA*], [*PgoAl*], [*valorRedondo*])⇒*valor*

`bal(5,6,5.75,5000,,12,12)` 833.11

**bal**(*NPgo*,*tablaAmort*)⇒*valor*

`tbl:=amortTbl(6,6,5.75,5000,,12,12)`

Función de amortización que calcula el balance del programa después de un pago especificado.

0	0.	0.	5000.
1	-23.35	-825.63	4174.37
2	-19.49	-829.49	3344.88
3	-15.62	-833.36	2511.52
4	-11.73	-837.25	1674.27
5	-7.82	-841.16	833.11
6	-3.89	-845.09	-11.98

*N*, *I*, *VP*, *Pgo*, *VF*, *PpA*, *CpA*, *PgoAl* se describen en la tabla de argumentos de VTD, página 176.

`bal(4,tbl)` 1674.27

*NPgo* especifica el número de pago después del cual usted desea que los datos se calculen.

*N*, *I*, *VP*, *Pgo*, *VF*, *PpA*, *CpA*, *PgoAl* se describen en la tabla de argumentos de VTD, página 176.

- Si se omite *Pgo*, se predetermina a *Pgo=tvmpmt* (*N*,*I*,*VP*,*VF*,*PpA*,*CpA*,*PgoAl*).
- Si se omite *VF*, se predetermina a *VF=0*.
- Los predeterminados para *PpA*, *CpA*, *PgoAl* son los mismos que para las funciones de VTD.

*valorRedondo* especifica el número de lugares decimales para el redondeo. Predeterminado=2.

**bal**(*NPgo*,*tablaAmort*) calcula el balance después del número de pago *NPgo*, basado en la tabla de amortización *tablaAmort*. El argumento *tablaAmort* debe ser una matriz en la forma descrita bajo **amortTbl()**, página 7.

**Nota:** Vea también **ΣInt()** y **ΣPrn()**, página 205.

*Entero1* ►Base2⇒*entero*

256►Base2

0b100000000

**Nota:** Se puede insertar este operador desde el teclado de la computadora al escribir @>Base2.

0h1F►Base2

0b11111

Convierte *Entero1* en un número binario. Los número binarios o hexadecimales siempre tienen un prefijo 0b ó 0h, respectivamente. Cero, no la letra O, seguida de b o de h.

0b *númeroBinario*

0h *númeroHexadecimal*

Un número binario puede tener hasta 64 dígitos. Un número hexadecimal puede tener hasta 16.

Sin un prefijo, *Entero1* se trata como decimal (base 10). El resultado se despliega en binario, independientemente del modo de la Base.

Los números negativos se despliegan en forma de "complemento de dos". Por ejemplo:

-1 se despliega como  
 0hFFFFFFFFFFFFFFFF en modo de base Hexadecimal  
 0b111...111 (64 1's) en modo de base Binaria

-2<sup>63</sup> se despliega como  
 0h8000000000000000 en modo de base Hexadecimal  
 0b100...000 (63 ceros) en modo de base Binaria

Si se ingresa un entero decimal que está fuera del rango de una forma binaria de 64 bits firmada, se usa una operación de módulo simétrico para llevar el valor al rango apropiado. Considere los siguientes ejemplos de valores fuera del rango.

$2^{63}$  se convierte en  $-2^{63}$  y se despliega como 0h8000000000000000 en modo de base Hexadecimal 0b100...000 (63 ceros) en modo de base Binaria

$2^{64}$  se convierte en 0 y se despliega como 0h0 en modo de base Hexadecimal 0b0 en modo de base Binaria

$-2^{63} - 1$  se convierte en  $2^{63} - 1$  y se despliega como 0h7FFFFFFFFFFFFFFF en modo de base Hexadecimal 0b111...111 (64 1's) en modo de base Binaria

►Base10

*Enterol* ►Base10⇒*entero*

0b10011►Base10	19
----------------	----

**Nota:** Se puede insertar este operador desde el teclado de la computadora al escribir @>Base10.

0h1F►Base10	31
-------------	----

Convierte *Integerl* en un número decimal (base 10). El ingreso binario o hexadecimal siempre debe tener un prefijo 0b ó 0h, respectivamente.

0b *númeroBinario*

0h *númeroHexadecimal*

Cero, no la letra O, seguida de b o de h.

Un número binario puede tener hasta 64 dígitos. Un número hexadecimal puede tener hasta 16.

Sin un prefijo, *Integerl* se trata como decimal. El resultado se despliega en decimal, independientemente del modo de la Base.

*Entero1* ►Base16⇒*entero*

256►Base16

0h100

**Nota:** Se puede insertar este operador desde el teclado de la computadora al escribir @>Base16.

0b111100001111►Base16

0hF0F

Convierte *Entero1* en un número hexadecimal. Los número binarios o hexadecimales siempre tienen un prefijo 0b ó 0h, respectivamente.

0b *númeroBinario*

0h *númeroHexadecimal*

Cero, no la letra O, seguida de b o de h.

Un número binario puede tener hasta 64 dígitos. Un número hexadecimal puede tener hasta 16.

Sin un prefijo, *Integer1* se trata como decimal (base 10). El resultado se despliega en hexadecimal, independientemente del modo de la Base.

Si se ingresa un entero decimal que es demasiado grande para una forma binaria de 64 bits firmada, se usa una operación de módulo simétrico para llevar el valor al rango apropiado. Para obtener más información, vea ►Base2, página 17.

## binomCdf()

**binomCdf**(*n,p*)⇒*lista*

**binomCdf**

(*n,p,límiteInferior,límiteSuperior*)⇒*número*  
si *límiteInferior* y *límiteSuperior* son  
números, *lista* si *límiteInferior* y  
*límiteSuperior* son listas

**binomCdf**(*n,p,límiteSuperior*) para  $P(0 \leq X \leq \text{límiteSuperior}) \Rightarrow \text{número}$  si *límiteSuperior* es un número, *lista* si *límiteSuperior* es una lista

**binomCdf()**

Catálogo &gt;

Genera una probabilidad acumulativa para la distribución binómica discreta con  $n$  número de pruebas y probabilidad  $p$  de éxito en cada prueba.

Para  $P(X \leq \text{limiteSuperior})$ , configure  $\text{limiteInferior}=0$

**binomPdf()**

Catálogo &gt;

**binomPdf( $n,p$ )**⇒*lista*

**binomPdf( $n,p,XVal$ )**⇒*número* si  $XVal$  es un número, *lista* si  $XVal$  es una lista

Genera una probabilidad para la distribución binómica discreta con  $n$  número de pruebas y probabilidad  $p$  de éxito en cada prueba.

**C****ceiling() (techo)**

Catálogo &gt;

**ceiling( $Valor1$ )**⇒*valor*

<code>ceiling(.456)</code>	1.
----------------------------	----

Entrega el entero más cercano que es  $\geq$  el argumento.

El argumento puede ser un número real o complejo.

**Nota:** Vea también **floor()**.

**ceiling( $Lista1$ )**⇒*lista*

<code>ceiling({-3.1,1,2.5})</code>	{-3.,1,3.}
------------------------------------	------------

**ceiling( $Matriz1$ )**⇒*matriz*

<code>ceiling(<math>\begin{bmatrix} 0 &amp; -3.2 \cdot i \\ 1.3 &amp; 4 \end{bmatrix}</math>)</code>	$\begin{bmatrix} 0 & -3 \cdot i \\ 2. & 4 \end{bmatrix}$
--	--

Entrega una lista o matriz del techo de cada elemento.

**centralDiff()**

Catálogo &gt;

**centralDiff( $Expr1,Var [=Valor]$  [,Paso])**⇒*expresión*

<code>centralDiff(cos(x),x) x=<math>\frac{\pi}{2}</math></code>	-1.
---	-----

**centralDiff( $Expr1,Var$  [,Paso])** |  $Var=Valor$ ⇒*expresión*

**centralDiff( $Expr1,Var [=Valor]$ )**



[,Lista])⇒*lista*

**centralDiff**(*Lista1*,*Var* [=*Valor*]  
[,*Paso*])⇒*lista*

**centralDiff**(*Matriz1*,*Var* [=*Valor*]  
[,*Paso*])⇒*matriz*

Entrega la derivada numérica usando la fórmula del cociente diferencial central.

Cuando se especifica el *Valor*, se eliminan todas las asignaciones anteriores de la variable o cualquier sustitución "|" para la variable.

*Paso* es el valor del paso. Si se omite *Paso*, se predetermina a 0.001.

Al usar *Lista1* o *Matriz1*, la operación se mapea a lo largo de los valores en la lista y a lo largo de los elementos de la matriz.

**Nota:** Vea también **avgRC()**.

**char**(*Entero*)⇒*caracter*

Entrega una cadena de caracteres que contiene el caracter numerado *Entero* desde el conjunto de caracteres del dispositivo portátil. El rango válido para *Entero* es 0–65535.

char(38)	"&"
char(65)	"A"

$\chi^2$ 2way *matrizObs*

**chi22way** *matrizObs*

Resuelve una prueba  $\chi^2$  para la asociación en la tabla bidireccional de conteos en la matriz observada *matrizObs*. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Para obtener información sobre el efecto de los elementos vacíos en una matriz, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat. $\chi^2$	Estadísticas cuadradas de Ji: suma (observada - esperada) <sup>2</sup> /esperada
stat.ValP	Nivel más bajo de significancia en el cual la hipótesis nula se puede rechazar
stat.df	Grados de libertad para las estadísticas cuadradas de ji
stat.ExpMat	Matriz de tabla de conteo elemental esperada, suponiendo una hipótesis nula
stat.CompMat	Matriz de contribuciones de estadísticas cuadradas de ji elementales

 $\chi^2$ Cdf() $\chi^2$ Cdf

(límiteInferior, límiteSuperior, df)  $\Rightarrow$  número si límiteInferior y límiteSuperior son números, lista si límiteInferior y límiteSuperior son listas

## chi2Cdf

(límiteInferior, límiteSuperior, df)  $\Rightarrow$  número si límiteInferior y límiteSuperior son números, lista si límiteInferior y límiteSuperior son listas

Genera la probabilidad de distribución  $\chi^2$  entre límiteInferior y límiteSuperior para grados específicos de libertad df.

Para  $P(X \leq \text{límiteSuperior})$ , configure límiteInferior = 0.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

 $\chi^2$ GOF

$\chi^2$ GOF listaObs, listaExp, df

chi2GOF listaObs, listaExp, df

Realiza una prueba para confirmar que los datos de la muestra son de una población que cumple con una distribución especificada. *listaObs* es una lista de conteos y debe contener enteros. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

Variable de salida	Descripción
stat. $\chi^2$	Estadísticas cuadradas de Ji: suma((observada - esperada) <sup>2</sup> /esperada
stat.ValP	Nivel más bajo de significancia en el cual la hipótesis nula se puede rechazar
stat.df	Grados de libertad para las estadísticas cuadradas de ji
stat.ListaComp	Contribuciones de estadísticas cuadradas de ji elementales

 $\chi^2$ Pdf()

$\chi^2$ Pdf(*XVal*,*df*) $\Rightarrow$ número si *XVal* es un número, *lista* si *XVal* es una lista

chi2Pdf(*XVal*,*df*) $\Rightarrow$ número si *XVal* es un número, *lista* si *XVal* es una lista

Genera la función de densidad de probabilidad (pdf) para la distribución  $\chi^2$  a un valor especificado *XVal* para los grados de libertad especificados *df*.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

## ClearAZ (LimpiarAZ)

## ClearAZ

Limpia todas las variables de carácter único en el espacio del problema actual.

$5 \rightarrow b$	5
<i>b</i>	5
ClearAZ	Done
<i>b</i>	"Error: Variable is not defined"

Si una o más de las variables están bloqueadas, este comando despliega un mensaje de error y borra únicamente las variables no bloqueadas. Vea **unLock**, página 179.

## ClrErr (LimpErr)

## ClrErr

Limpia el estado del error y configura *Codigerr* de la variable del sistema a cero.

La cláusula **Else** del bloque **Try...Else...EndTry** debe usar **ClrErr** o **PassErr**. Si el error se debe procesar o ignorar, use **ClrErr**. Si no se sabe qué hacer con el error, use **PassErr** para enviarlo al siguiente manipulador de errores. Si no hay ningún otro manipulador de errores **Try...Else...EndTry** pendiente, el cuadro de diálogo de error se desplegará como normal.

**Nota:** Vea también **PasErr**, página 119, y **Try**, página 172.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

Para consultar un ejemplo de **ClrErr**, vea el Ejemplo 2 bajo el comando **Try**, página 172.

## colAugment()

$\text{colAugment}(\text{Matriz1}, \text{Matriz2}) \Rightarrow \text{matriz}$

Entrega una nueva matriz que es *Matriz2* adjuntada a *Matriz1*. Las matrices deben tener dimensiones de columna iguales, y *Matriz2* se adjunta a *Matriz1* como nuevas filas. No altera *Matriz1* o *Matriz2*.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
$\begin{bmatrix} 5 & 6 \end{bmatrix} \rightarrow m2$	$\begin{bmatrix} 5 & 6 \end{bmatrix}$
$\text{colAugment}(m1, m2)$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$

## colDim()

$\text{colDim}(\text{Matriz}) \Rightarrow \text{expresión}$

$\text{colDim}\left(\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}\right)$	3
--	---

**colDim()**

Catálogo &gt;

Entrega el número de columnas contenidas en *Matriz*.

**Nota:** Vea también **rowDim()**.

**colNorm()**

Catálogo &gt;

**colNorm**(*Matriz*) $\Rightarrow$ *expresión*

Entrega el máximo de las sumas de los valores absolutos de los elementos en las columnas en *Matriz*.

**Nota:** Los elementos de matriz indefinida no están permitidos. Vea también **rowNorm()**.

$\begin{bmatrix} 1 & -2 & 3 \\ 4 & 5 & -6 \end{bmatrix} \rightarrow mat$	$\begin{bmatrix} 1 & -2 & 3 \\ 4 & 5 & -6 \end{bmatrix}$
<b>colNorm</b> ( <i>mat</i> )	9

**conj()**

Catálogo &gt;

**conj**(*Valor1*) $\Rightarrow$ *valor*

**conj**(*Lista1*) $\Rightarrow$ *lista*

**conj**(*Matriz1*) $\Rightarrow$ *matriz*

Entrega el complejo conjugado del argumento.

<b>conj</b> ( $1+2 \cdot i$ )	$1-2 \cdot i$
<b>conj</b> $\left(\begin{bmatrix} 2 & 1-3 \cdot i \\ -i & -7 \end{bmatrix}\right)$	$\begin{bmatrix} 2 & 1+3 \cdot i \\ i & -7 \end{bmatrix}$

**constructMat()**

Catálogo &gt;

**constructMat**  
(*Expr*,*Var1*,*Var2*,*numFilas*,*numCols*)  
 $\Rightarrow$ *matriz*

Entrega una matriz basada en los argumentos.

*Expr* es una expresión en las variables *Var1* y *Var2*. Los elementos en la matriz resultante se forman al evaluar *Expr* para cada valor incrementado de *Var1* y *Var2*.

*Var1* se incrementa automáticamente desde 1 a *numFilas*. Dentro de cada fila, *Var2* se incrementa desde 1 a *numCols*.

<b>constructMat</b> $\left(\frac{1}{i+j}, i, j, 3, 4\right)$	$\begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$
--	---

**CopyVar** *Var1, Var2***CopyVar** *Var1., Var2.*

**CopyVar** *Var1, Var2* copia el valor de la variable *Var1* a la variable *Var2*, creando *Var2* si es necesario. La variable *Var1* debe tener un valor.













Si *Var1* es el nombre de una función existente definida por el usuario, copia la definición de esa función a la función *Var2*. La función *Var1* se debe definir.

*Var1* debe cumplir con los requisitos de nombramiento de la variable o debe ser una expresión de indirección que se simplifica a un nombre de variable que cumple con los requisitos.

**CopyVar** *Var1., Var2.* copia todos los miembros del grupo de la variable *Var1*. al grupo *Var2.*, creando *Var2.* si es necesario.

*Var1.* debe ser el nombre de un grupo de variables existente, como los resultados de las estadísticas *stat.nn* o las variables creadas usando la función **LibShortcut()**. Si *Var2.* ya existe, este comando reemplaza todos los miembros que son comunes para ambos grupos y agrega los miembros que no existen todavía. Si uno o más miembros de *Var2.* están bloqueados, todos los miembros de *Var2.* se dejan sin cambios.

Define $a(x)=\frac{1}{x}$	Done
Define $b(x)=x^2$	Done
CopyVar a,c: c(4)	$\frac{1}{4}$
CopyVar b,c: c(4)	16

aa.a:=45	45																
aa.b:=6.78	6.78																
CopyVar aa.bb.	Done																
getVarInfo()	<table border="1"> <tr> <td>aa.a</td> <td>"NUM"</td> <td></td> <td>0</td> </tr> <tr> <td>aa.b</td> <td>"NUM"</td> <td></td> <td>0,</td> </tr> <tr> <td>bb.a</td> <td>"NUM"</td> <td></td> <td>0</td> </tr> <tr> <td>bb.b</td> <td>"NUM"</td> <td></td> <td>0</td> </tr> </table>	aa.a	"NUM"		0	aa.b	"NUM"		0,	bb.a	"NUM"		0	bb.b	"NUM"		0
aa.a	"NUM"		0														
aa.b	"NUM"		0,														
bb.a	"NUM"		0														
bb.b	"NUM"		0														

**corrMat()****corrMat**(*Lista1,Lista2[,...[,Lista20]]*)

Genera la matriz de correlación para la matriz aumentada [*Lista1, Lista2, ..., Lista20*].

**cos()****cos**(*Valor1*)⇒*valor*

En modo de ángulo en Grados:

cos(Lista1)⇒lista

cos(Valor1) entrega el coseno del argumento como un valor.

cos(Lista1) entrega una lista de cosenos de todos los elementos en Lista1.

**Nota:** El argumento se interpreta como un ángulo en grados, gradianes o radianes, de acuerdo con la configuración del modo del ángulo actual. Se puede usar °, G o r para anular el modo de ángulo en forma temporal.

**cos**

(matrizCuadrada1)⇒matrizCuadrada

Entrega el coseno de la matriz de matrizCuadrada1. Esto no es lo mismo que calcular el coseno de cada elemento.

Cuando una función escalar f(A) opera en matrizCuadrada1 (A), el resultado se calcula por medio del algoritmo:

Compute los valores propios (λ<sub>i</sub>) y los vectores propios (V<sub>i</sub>) de A.

matrizCuadrada1 debe ser diagonalizable. Asimismo, no puede tener variables simbólicas a las que no se ha asignado un valor.

Forme las matrices:

$$B = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \text{ and } X = [V_1, V_2, \dots, V_n]$$

Luego A = X B X<sup>-1</sup> y f(A) = X f(B) X<sup>-1</sup>. Por ejemplo, cos(A) = X cos(B) X<sup>-1</sup> donde:

---


$$\cos\left(\left(\frac{\pi}{4}\right)^r\right) \quad 0.707107$$


---

$$\cos(45) \quad 0.707107$$


---

$$\cos(\{0,60,90\}) \quad \{1,0.5,0\}$$


---

En modo de ángulo en Gradianes:

---


$$\cos(\{0,50,100\}) \quad \{1,0.707107,0\}$$


---

En modo de ángulo en Radianes:

---


$$\cos\left(\frac{\pi}{4}\right) \quad 0.707107$$


---

$$\cos(45^\circ) \quad 0.707107$$


---

En modo de ángulo en Radianes:

---


$$\cos\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right) \quad \begin{bmatrix} 0.212493 & 0.205064 & 0.121389 \\ 0.160871 & 0.259042 & 0.037126 \\ 0.248079 & -0.090153 & 0.218972 \end{bmatrix}$$


---

**cos()**

cos(B) =

$$\begin{bmatrix} \cos(\lambda_1) & 0 & \dots & 0 \\ 0 & \cos(\lambda_2) & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \cos(\lambda_n) \end{bmatrix}$$

Todos los cálculos se realizan usando aritmética de punto flotante.

**cos<sup>-1</sup>()**cos<sup>-1</sup>(Valor) ⇒ valor

En modo de ángulo en Grados:

cos<sup>-1</sup>(Lista) ⇒ lista
 0.

cos<sup>-1</sup>•ValorIΣ entrega el ángulo cuyo coseno es ValorI

En modo de ángulo en Gradianes:

cos<sup>-1</sup>(ListaI) entrega una lista de cosenos inversos de cada elemento de ListaI.

 100.

**Nota:** El resultado se entrega como un ángulo en grados, gradianes o radianes, de acuerdo con la configuración del modo del ángulo actual.

En modo de ángulo en Radianes:

---

 {1.5708,1.36944,1.0472}

---

**Nota:** Se puede insertar esta función desde el teclado al escribir **arccos** (...).

cos<sup>-1</sup>  
(matrizCuadradaI) ⇒ matrizCuadrada

En el modo de ángulo en Radianes y el Formato Complejo Rectangular:

Entrega el coseno inverso de la matriz de matrizCuadradaI. Esto no es lo mismo que calcular el coseno inverso de cada elemento. Para obtener información acerca del método de cálculo, consulte cos().

---




---




---

matrizCuadradaI debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

Para ver el resultado completo, presione ▲ y después use ◀ y ▶ para mover el cursor.



**cosh()****cosh**(*Valor1*)⇒*valor*

En modo de ángulo en Grados:

**cosh**(*Lista1*)⇒*lista*

---


$$\cosh\left(\left(\frac{\pi}{4}\right)_r\right) \quad 1.74671\text{E}19$$


---

**cosh**(*Valor1*) entrega el coseno hiperbólico del argumento.**cosh**(*Lista1*) entrega una lista de cosenos hiperbólicos de cada elemento de *Lista1*.**cosh**  
(*matrizCuadrada1*)⇒*matrizCuadrada*

En modo de ángulo en Radianes:

Entrega el coseno hiperbólico de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular el coseno hiperbólico de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

---


$$\cosh\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right) \begin{bmatrix} 421.255 & 253.909 & 216.905 \\ 327.635 & 255.301 & 202.958 \\ 226.297 & 216.623 & 167.628 \end{bmatrix}$$


---

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.**cosh<sup>-1</sup>()****cosh<sup>-1</sup>**(*Valor1*)⇒*valor*

---


$$\cosh^{-1}(1) \quad 0$$

**cosh<sup>-1</sup>**(*Lista1*)⇒*lista*

---


$$\cosh^{-1}(\{1,2,1,3\}) \quad \{0,1.37286,1.76275\}$$


---

**cosh<sup>-1</sup>**(*Valor1*) entrega el coseno hiperbólico inverso del argumento.**cosh<sup>-1</sup>**(*Lista1*) entrega una lista de cosenos hiperbólicos inversos de cada elemento de *Lista1*.**Nota:** Se puede insertar esta función desde el teclado al escribir **arccosh** (...).**cosh<sup>-1</sup>**  
(*matrizCuadrada1*)⇒*matrizCuadrada*

En el modo de ángulo en Radianes y en el Formato Complejo Rectangular:

## $\cosh^{-1}()$

Entrega el coseno hiperbólico inverso de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular el coseno hiperbólico inverso de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

$$\cosh^{-1}\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right)$$
$$\begin{bmatrix} 2.52503+1.73485\cdot i & -0.009241-1.49086\cdot i \\ 0.486969-0.725533\cdot i & 1.66262+0.623491\cdot i \\ -0.322354-2.08316\cdot i & 1.26707+1.79018\cdot i \end{bmatrix}$$

Para ver el resultado completo, presione  $\blacktriangle$  y después use  $\blacktriangleleft$  y  $\blacktriangleright$  para mover el cursor.

## **cot()**

**cot(Valor1)**  $\Rightarrow$  *valor*

En modo de ángulo en Grados:

**cot(Lista1)**  $\Rightarrow$  *lista*

$$\text{cot}(45) \quad 1.$$

Entrega la cotangente de *Valor1* o entrega una lista de cotangentes de todos los elementos en *Lista1*.

En modo de ángulo en Gradianes:

**Nota:** El argumento se interpreta como un ángulo en grados, gradianes o radianes, de acuerdo con la configuración del modo del ángulo actual. Se puede usar  $^{\circ}$ ,  $^{\text{G}}$  o  $^{\text{r}}$  para anular el modo de ángulo en forma temporal.

$$\text{cot}(50) \quad 1.$$

En modo de ángulo en Radianes:

$$\text{cot}(\{1,2,1,3\})$$
$$\{0.642093, -0.584848, -7.01525\}$$

## $\cot^{-1}()$

**$\cot^{-1}(\text{Valor1})$**   $\Rightarrow$  *valor*

En modo de ángulo en Grados:

**$\cot^{-1}(\text{Lista1})$**   $\Rightarrow$  *lista*

$$\cot^{-1}(1) \quad 45.$$

Entrega el ángulo cuya cotangente es *Valor1* o entrega una lista que contiene las cotangentes inversas de cada elemento de *Lista1*.

En modo de ángulo en Gradianes:


$$\cot^{-1}(1) \quad 50.$$

**Nota:** El resultado se entrega como un ángulo en grados, gradianes o radianes, de acuerdo con la configuración del modo del ángulo actual.

En modo de ángulo en Radianes:

**Nota:** Se puede insertar esta función desde el teclado al escribir **arccot (...)**.

$$\cot^{-1}(1) \quad 0.785398$$


**coth()**Catálogo > **coth(Valor1)**⇒valor

coth(1.2) 1.19954

**coth(Lista1)**⇒lista

coth({1,3,2}) {1.31304,1.00333}

Entrega la cotangente hiperbólica de *Valor1* o entrega una lista de cotangentes hiperbólicas de todos los elementos de *Listas1*.

**coth<sup>-1</sup>()**Catálogo > **coth<sup>-1</sup>(Valor1)**⇒valorcoth<sup>-1</sup>(3.5) 0.293893**coth<sup>-1</sup>(Lista1)**⇒listacoth<sup>-1</sup>({-2,2,1,6})  
{-0.549306,0.518046,0.168236}

Entrega la cotangente hiperbólica inversa de *Valor1* o entrega una lista que contiene las cotangentes hiperbólicas inversas de cada elemento de *Listas1*.

**Nota:** Se puede insertar esta función desde el teclado al escribir **arccoth** (...).

**count()**Catálogo > **count(Valor1oLista1 [,Valor2oLista2 [,...]])**⇒valor

count(2,4,6) 3

count({2,4,6}) 3

Entrega el conteo acumulado de todos los elementos en los argumentos que se evalúan a valores numéricos.

count(2,{4,6}, $\begin{bmatrix} 8 & 10 \\ 12 & 14 \end{bmatrix}$ ) 7

Cada argumento puede ser una expresión, valor, lista o matriz. Se puede mezclar tipos de datos y usar argumentos de varias dimensiones.

Para una lista, matriz o rango de celdas, cada elemento se evalúa para determinar si se debe incluir en el conteo.

Dentro de la aplicación Listas y Hoja de Cálculo, se puede usar un rango de celdas en lugar de cualquier argumento.

Los elementos vacíos (anulados) se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

**countif() (conteoSi)**

**countif(Lista,Criterios)⇒valor**

Entrega el conteo acumulado de todos los elementos en *Lista* que cumplen con los *Criterios* especificados.

Los *criterios* pueden ser:

- Un valor, una expresión o una cadena. Por ejemplo, **3** cuenta sólo aquellos elementos en *Lista* que se simplifican al valor 3.
- Una expresión Booleana que contiene el símbolo ? como un marcador de posición para cada elemento. Por ejemplo, **?<5** cuenta sólo aquellos elementos en *Lista* que son menos de 5.

Dentro de la aplicación Listas y Hoja de Cálculo, se puede usar un rango de celdas en lugar de *Lista*.

Los elementos vacíos (anulados) en la lista se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

**Nota:** Vea también **sumIf()**, página 164, y **frequency()**, página 60.

countIf({1,3,"abc",undef,3,1},3)	2
----------------------------------	---

Cuenta el número de elementos iguales a 3.

countIf({"abc","def","abc",3},"dif")	1
--------------------------------------	---

Cuenta el número de elementos iguales a "dif."

countIf({1,3,5,7,9},?<5)	2
--------------------------	---

Cuenta 1 y 3.

countIf({1,3,5,7,9},2<?<8)	3
----------------------------	---

Cuenta 3, 5 y 7.

countIf({1,3,5,7,9},?<4 or ?>6)	4
---------------------------------	---

Cuenta 1, 3, 7 y 9.

**cPolyRoots() (RaícesPoliC)**

**cPolyRoots(Poli,Var)⇒lista**

**cPolyRoots(ListaDeCoefts)⇒lista**

La primera sintaxis, **cPolyRoots(Poli,Var)**, entrega una lista de raíces complejas del polinomio *Poli* con respecto de la variable *Var*.

polyRoots(y <sup>3</sup> +1,y)	{-1}
--------------------------------	------

cPolyRoots(y <sup>3</sup> +1,y)	{-1,0.5-0.866025 <i>i</i> ,0.5+0.866025 <i>i</i> }
---------------------------------	--

polyRoots(x <sup>2</sup> +2·x+1,x)	{-1,-1}
------------------------------------	---------

cPolyRoots({1,2,1})	{-1,-1}
---------------------	---------

*Poli* debe ser un polinomio en forma expandida en una variable. No use formas expandidas como  $y^2 \cdot y + 1$  ó  $x \cdot x + 2 \cdot x + 1$

La segunda sintaxis, **cPolyRoots** (*ListaDeCoefs*), entrega una lista de raíces complejas para los coeficientes en *ListaDeCoefs*.

**Nota:** Vea también **polyRoots()**, página 122.

**crossP()**

**crossP**(*Lista1*, *Lista2*) $\Rightarrow$ *lista*

Entrega el producto cruzado de *Lista1* y *Lista2* como una lista.

*Lista1* y *Lista2* deben tener una dimensión igual, y la dimensión debe ser 2 ó 3.

**crossP**(*Vector1*, *Vector2*) $\Rightarrow$ *vector*

Entrega un vector de fila o columna (dependiendo de los argumentos) que es el producto cruzado de *Vector1* y *Vector2*.

Tanto *Vector1* como *Vector2* deben ser vectores de fila, o ambos deben ser vectores de columna. Ambos vectores deben tener una dimensión igual, y la dimensión debe ser 2 ó 3.

$$\text{crossP}(\{0.1, 2.2, -5\}, \{1, -0.5, 0\})$$


---


$$\{-2.5, -5., -2.25\}$$

$$\text{crossP}(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \begin{bmatrix} -3 & 6 & -3 \end{bmatrix})$$


---


$$\text{crossP}(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 0 & 0 & -2 \end{bmatrix})$$

**csc()**

**csc**(*Valor1*) $\Rightarrow$ *valor*

En modo de ángulo en Grados:

**csc**(*Lista1*) $\Rightarrow$ *lista*

$$\text{csc}(45) \quad 1.41421$$

Entrega la cosecante de *Valor1* o entrega una lista que contiene las cosecantes de todos los elementos en *Lista1*.

En modo de ángulo en Gradianes:

$$\text{csc}(50) \quad 1.41421$$

**csc()**

En modo de ángulo en Radianes:

$$\text{csc}\left(\left\{1, \frac{\pi}{2}, \frac{\pi}{3}\right\}\right) \quad \{1.1884, 1., 1.1547\}$$

**csc<sup>-1</sup>()****csc<sup>-1</sup>(Valor1)** ⇒ *valor*

En modo de ángulo en Grados:

**csc<sup>-1</sup>(Lista1)** ⇒ *lista*

$$\text{csc}^{-1}(1) \quad 90.$$

Entrega el ángulo cuya cosecante es *Valor1* o entrega una lista que contiene las cosecantes inversas de cada elemento de *Lista1*.

En modo de ángulo en Gradianes:

$$\text{csc}^{-1}(1) \quad 100.$$

**Nota:** El resultado se entrega como un ángulo en grados, gradianes o radianes, de acuerdo con la configuración del modo del ángulo actual.

En modo de ángulo en Radianes:

$$\text{csc}^{-1}(\{1, 4, 6\}) \quad \{1.5708, 0.25268, 0.167448\}$$

**Nota:** Se puede insertar esta función desde el teclado al escribir **arccsc (...)**.

**csch()**

Catálogo &gt;

**csch(Valor1)** ⇒ *valor*

$$\text{csch}(3) \quad 0.099822$$

**csch(Lista1)** ⇒ *lista*

$$\text{csch}(\{1, 2, 1, 4\}) \quad \{0.850918, 0.248641, 0.036644\}$$

Entrega la cosecante hiperbólica de *Valor1* o entrega una lista de cosecantes hiperbólicas de todos los elementos de *Lista1*.

**csch<sup>-1</sup>()**

Catálogo &gt;

**csch<sup>-1</sup>(Valor)** ⇒ *valor*

$$\text{csch}^{-1}(1) \quad 0.881374$$

**csch<sup>-1</sup>(Lista1)** ⇒ *lista*

$$\text{csch}^{-1}(\{1, 2, 1, 3\}) \quad \{0.881374, 0.459815, 0.32745\}$$

Entrega la cosecante hiperbólica inversa de *Valor1* o entrega una lista que contiene las cosecantes hiperbólicas inversas de cada elemento de *Lista1*.

**Nota:** Se puede insertar esta función desde el teclado al escribir **arccsch** (...).

**CubicReg**

**CubicReg** *X*, *Y*, [*Frec*] [, *Categoría*, *Incluir*]]

Resuelve la regresión polinómica cúbica  $y = a \cdot x^3 + b \cdot x^2 + c \cdot x + d$  en listas *X* y *Y* con frecuencia *Frec*. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

*X* y *Y* son listas de variables independientes y dependientes.

*Frec* es una lista opcional de valores de frecuencia. Cada elemento en *Frec* especifica la frecuencia de la ocurrencia para cada punto de datos *X* y *Y* correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

*Categoría* es una lista de códigos de categoría numérica o de cadena para los datos *X* y *Y* correspondientes.


*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $a \cdot x^3 + b \cdot x^2 + c \cdot x + d$
stat.a, stat.b, stat.c, stat.d	Coefficientes de regresión
stat.R <sup>2</sup>	Coefficiente de determinación

Variable de salida	Descripción
stat.Resid	Residuales de la regresión
stat.XReg	La lista de puntos de datos en <i>Lista X</i> modificada se usa de hecho en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoriæ Incluir</i>
stat.YReg	La lista de puntos de datos en <i>Lista Y</i> modificada se usa de hecho en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoriæ Incluir</i>
stat.FrecReg	Lista de frecuencias correspondientes a <i>stat.XReg</i> y <i>stat.YReg</i>

## cumulativeSum()

Catálogo > 

**cumulativeSum(Lista1)** ⇒ lista

$\text{cumulativeSum}(\{1,2,3,4\}) \quad \{1,3,6,10\}$

Entrega una lista de sumas acumulativas de los elementos en *Lista1* comenzando en el elemento 1.

**cumulativeSum(Matriz1)** ⇒ matriz

1 2	→ m1	1 2
3 4		3 4
5 6		5 6
<b>cumulativeSum(m1)</b>		1 2
		4 6
		9 12

Entrega una matriz de sumas acumulativas de los elementos en *Matriz1*. Cada elemento está en la suma acumulativa de la columna desde la parte superior hasta la parte inferior.

Un elemento vacío (anulado) en *Lista1* o *Matriz1* produce un elemento anulado en la lista o matriz resultante. Para obtener más información sobre elementos vacíos, vea página 231.

## Cycle

Catálogo > 

### Cycle

Lista de funciones que suma los enteros desde 1 hasta 100, saltándose 50.

Transfiere el control de inmediato a la siguiente iteración del bucle actual (**For**, **While**, o **Loop**).

**Cycle** no está permitido afuera de las tres estructuras de bucles ((**For**, **While**, o **Loop**)).



**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

Define $g()$ =Func	Done
Local $temp,i$	
$0 \rightarrow temp$	
For $i,1,100,1$	
If $i=50$	
Cycle	
$temp+i \rightarrow temp$	
EndFor	
Return $temp$	
EndFunc	
$g()$	5000

## ►Cylind

*Vector* ►Cylind

**Nota:** Se puede insertar este operador desde el teclado de la computadora al escribir @>Cylind.

Despliega el vector de fila o columna en forma cilíndrica  $[r, \angle\theta, z]$ .

*Vector* debe tener exactamente tres elementos. Puede ser una fila o una columna.

$[2 \ 2 \ 3]$ ►Cylind	$[2.82843 \ \angle 0.785398 \ 3.]$
-----------------------	------------------------------------

## D

## dbd()

**dbd(*fecha1*,*fecha2*)**⇒*valor*

Entrega el número de días entre *fecha1* y *fecha2* usando el método de conteo de días reales.

*fecha1* y *fecha2* pueden ser números dentro del rango de las fechas en el calendario estándar. Si tanto *fecha1* como *fecha2* son listas, deberán tener la misma longitud.

Tanto *fecha1* como *fecha2* deben estar entre los años 1950 a 2049.

dbd(12.3103,1.0104)	1
dbd(1.0107,6.0107)	151
dbd(3112.03,101.04)	1
dbd(101.07,106.07)	151

Usted puede ingresar las fechas en uno de dos formatos. La colocación decimal se diferencia entre los formatos de fecha.

MM.DDAA (formato que se usa de manera común en los Estados Unidos)

DDMM.AA (formato que se usa de manera común en Europa)

**►DD**

*Expr1* ►DD⇒*valor*

En modo de ángulo en Grados:

*Lista1* ►DD⇒*lista*

$(1.5^\circ)$ ►DD	1.5°
$(45^\circ 22' 14.3")$ ►DD	45.3706°
$(\{45^\circ 22' 14.3", 60^\circ 0' 0"\})$ ►DD	{45.3706°, 60°}

*Matriz1* ►DD⇒*matriz*

**Nota:** Usted puede insertar este operador desde el teclado de la computadora al escribir @>DD.

Entrega el decimal equivalente del argumento expresado en grados. El argumento es un número, lista o matriz que se interpreta por medio de la configuración del modo de Ángulo en gradianes, radianes o grados.

En modo de ángulo en Gradianes:

1►DD	$\frac{9}{10}$
------	----------------

En modo de ángulo en Radianes:

$(1.5)$ ►DD	85.9437°
-------------	----------

**►Decimal**

*Número1* ►Decimal⇒*valor*

$\frac{1}{3}$ ►Decimal	0.333333
------------------------	----------

*Lista1* ►Decimal⇒*valor*

*Matriz1* ►Decimal⇒*valor*

**Nota:** Usted puede insertar este operador desde el teclado de la computadora al escribir @>Decimal.

Despliega el argumento en forma decimal. Este operador se puede usar únicamente al final de la línea de ingreso.

**Define** *Var* = *Expresión*

**Define Función**(*Param1*, *Param2*, ...) = *Expresión*

Define la variable *Var* o la función definida por el usuario *Función*.

Los parámetros, como *Param1*, proporcionan marcadores de posición para pasar argumentos a la función. Cuando llame a una función definida por el usuario, usted deberá suministrar argumentos (por ejemplo, valores o variables) que correspondan a los parámetros. Cuando se llama, la función evalúa la *Expresión* usando los argumentos provistos.

*Var* y *Función* no pueden ser el nombre de una variable de sistema o de una función o un comando integrado.

**Nota:** Esta forma de **Define** es equivalente a ejecutar la expresión: *expresión* → *Función* (*Param1*, *Param2*).

**Define Función**(*Param1*, *Param2*, ...) = **Func**  
*Bloque*  
**EndFunc**

**Define Programa**(*Param1*, *Param2*, ...) = **Prgm**  
*Bloque*  
**EndPrgm**

En esta forma, la función o el programa definido por el usuario puede ejecutar un bloque de varias sentencias.

*Bloque* puede ser una sentencia sencilla o una serie de sentencias en líneas separadas. *Bloque* también puede incluir expresiones e instrucciones (como **If**, **Then**, **Else**, y **For**).

Define $g(x,y)=2 \cdot x-3 \cdot y$	Done
$g(1,2)$	-4
$1 \rightarrow a: 2 \rightarrow b: g(a,b)$	-4
Define $h(x)=\text{when}(x<2,2 \cdot x-3,-2 \cdot x+3)$	Done
$h(-3)$	-9
$h(4)$	-5

Define $g(x,y)=\text{Func}$	Done
If $x>y$ Then	
Return $x$	
Else	
Return $y$	
EndIf	
EndFunc	
$g(3,-7)$	3

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

**Nota:** Vea también **Define LibPriv**, página 40y **Define LibPub**, página 40.

```
Define g(x,y)=Prgm
  If x>y Then
    Disp x," greater than ",y
  Else
    Disp x," not greater than ",y
  EndIf
EndPrgm

```

---

*Done*

```
g(3,-7)
-----
3 greater than -7
-----
Done
```

**Define LibPriv** *Var = Expresión*

**Define LibPriv** *Función(Param1, Param2, ...)* = *Expresión*

**Define LibPriv** *Función(Param1, Param2, ...)* = **Func**  
*Bloque*  
**EndFunc**

**Define LibPriv** *Programa(Param1, Param2, ...)* = **Prgm**  
*Bloque*  
**EndPrgm**

Opera igual que **Define**, excepto porque define una variable de librería privada, función o programa. Las funciones y los programas privados no aparecen en el Catálogo.

**Nota:** Vea también **Define**, página 39 y **Define LibPub**, página 40.

**Define LibPub** *Var = Expresión*

**Define LibPub** *Función(Param1, Param2, ...)* = *Expresión*

**Define LibPub** *Función(Param1, Param2, ...)*

...) = Func

*Bloque*

**EndFunc**

**Define LibPub** Programa(Param1, Param2,

...) = Prgm

*Bloque*

**EndPrgm**

Opera igual que **Define**, excepto porque define una variable de librería pública, función o programa. Las funciones y los programas públicos aparecen en el Catálogo después de que la librería se ha guardado y actualizado.

**Nota:** Vea también **Define**, página 39 y **Define LibPriv**, página 40.

**deltaList()**

Vea  $\Delta$ List(), página 89.

**DelVar**

Catálogo > 

**DelVar** Var1[, Var2] [, Var3] ...

$2 \rightarrow a$	2
-------------------	---

**DelVar** Var.

$(a+2)^2$	16
-----------	----

Borra la variable o el grupo de variables especificado de la memoria.

DelVar a	Done
----------	------

$(a+2)^2$	"Error: Variable is not defined"
-----------	----------------------------------

Si una o más de las variables están bloqueadas, este comando despliega un mensaje de error y borra únicamente las variables no bloqueadas. Vea **unLock**, página 179.

## DelVar

Catálogo > 

**DelVar** *Var.* borra todos los miembros del grupo de variables *Var.* (como las estadísticas *stat.nn* los resultados o las variables que se crean con el uso de **LibShortcut()** función). El punto (.) en esta forma de comando **DelVar** lo limita a borrar un grupo de variables; la variable sencilla *Var* no se ve afectada.

<code>aa.a:=45</code>	45									
<code>aa.b:=5.67</code>	5.67									
<code>aa.c:=78.9</code>	78.9									
<code>getVarInfo()</code>	<table border="1"><tr><td><code>aa.a</code></td><td>"NUM"</td><td>"0"</td></tr><tr><td><code>aa.b</code></td><td>"NUM"</td><td>"0"</td></tr><tr><td><code>aa.c</code></td><td>"NUM"</td><td>"0"</td></tr></table>	<code>aa.a</code>	"NUM"	"0"	<code>aa.b</code>	"NUM"	"0"	<code>aa.c</code>	"NUM"	"0"
<code>aa.a</code>	"NUM"	"0"								
<code>aa.b</code>	"NUM"	"0"								
<code>aa.c</code>	"NUM"	"0"								
<code>DelVar aa.</code>	<i>Done</i>									
<code>getVarInfo()</code>	"NONE"									

## delVoid() (borrInvalído)

Catálogo > 

**delVoid**(*Lista1*)⇒*lista*

<code>delVoid({1,void,3})</code>	{1,3}
----------------------------------	-------

Entrega una lista que tiene el contenido de *Lista1* con todos los elementos (nulos) vacíos eliminados.

Para obtener más información sobre elementos vacíos, vea página 231.

## det()

Catálogo > 

**det**(*matrizCuadrada*,  
*Tolerancia*)⇒*expresión*



$\det\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$	-2
--	----

Entrega la determinante de *matrizCuadrada*.

$\begin{bmatrix} 1.€20 & 1 \\ 0 & 1 \end{bmatrix} \rightarrow mat1$	$\begin{bmatrix} 1.€20 & 1 \\ 0 & 1 \end{bmatrix}$
---	--

De manera opcional, cualquier elemento de matriz se trata como cero si su valor absoluto es menor que la *Tolerancia*. Esta tolerancia se usa sólo si la matriz tiene ingresos de punto flotante y no contiene ninguna variable simbólica a la que no se le haya asignado un valor. De otro modo, la *Tolerancia* se ignora.

<code>det(mat1)</code>	0
<code>det(mat1,.1)</code>	1.€20

- Si usted usa   o configura el modo **Auto o Aproximado** para aproximar, los cálculos se realizan al usar la aritmética de punto flotante.
- Si la *Tolerancia* se omite o no se usa, la tolerancia predeterminada se calcula como:

5E-14 · **max(dim**  
*(matrizCuadrada)* · **rowNorm**  
*(matrizCuadrada)*

**diag()**

**diag(Lista)** ⇒ *matriz*

diag([2 4 6])	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix}$
---------------	---

**diag(matrizFila)** ⇒ *matriz*

**diag(matrizColumna)** ⇒ *matriz*

Entrega una matriz con los valores en la lista o matriz de argumentos en su diagonal principal.

**diag(matrizCuadrada)** ⇒ *matrizFila*

Entrega una matriz de filas que contiene los elementos de la diagonal principal de *matrizCuadrada*.

$\begin{bmatrix} 4 & 6 & 8 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{bmatrix}$	$\begin{bmatrix} 4 & 6 & 8 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{bmatrix}$
diag(Ans)	$\begin{bmatrix} 4 & 2 & 9 \end{bmatrix}$

*matrizCuadrada* debe ser cuadrada.

**dim()**

**dim(Lista)** ⇒ *entero*

dim({0,1,2})	3
--------------	---

Entrega la dimensión de *Lista*.

**dim(Matriz)** ⇒ *lista*

dim( $\begin{bmatrix} 1 & -1 \\ 2 & -2 \\ 3 & 5 \end{bmatrix}$ )	{3,2}
--	-------

Entrega las dimensiones de la matriz como una lista de dos elementos {filas, columnas}.

**dim(Cadena)** ⇒ *entero*

dim("Hello")	5
dim("Hello "&"there")	11

Entrega el número de caracteres contenidos en la cadena de caracteres *Cadena*.

**Disp** *exprOCadena1* [, *exprOCadena2*]

...

Despliega los argumentos en el historial de la *Calculadora*. Los argumentos se despliegan en sucesión, con espacios pequeños como separadores.

Es útil principalmente con programas y funciones para asegurar en despliegue de cálculos intermedios.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección *Calculadora* de la guía del producto.

```
Define chars(start,end)=Prgm
  For i,start,end
  Disp i," ",char(i)
  EndFor
EndPrgm
```

*Done*

*chars(240,243)*

240 ð

241 ñ

242 ò

243 ó

*Done*

## DispAt

**DispAt** *int,expr1* [,*expr2* ...] ...

**DispAt** permite especificar la línea en la que se mostrará en la pantalla la expresión o cadena de caracteres especificada.

El número de línea se puede especificar como una expresión.

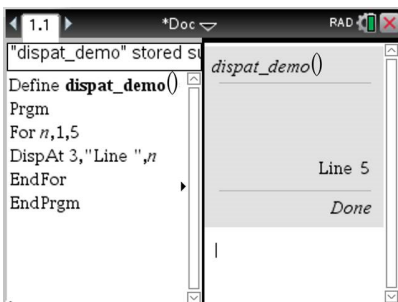
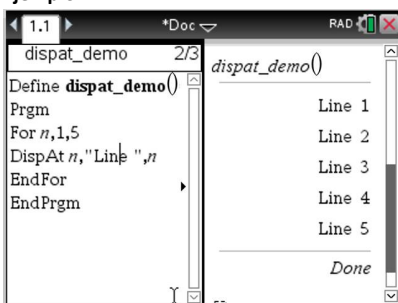
Tenga en cuenta que el número de línea no es para toda la pantalla, sino para el área inmediatamente después del comando/programa.

Este comando permite tener salidas tipo tablero de instrumentos de programas donde el valor de una expresión o de una lectura de sensor se actualiza en la misma línea.

**DispAty Disp** pueden utilizarse dentro del mismo programa.

DispAt

### Ejemplo





**Nota:** El número máximo se establece en 8 ya que coincide con una pantalla llena de líneas en la pantalla del dispositivo portátil, siempre y cuando las líneas no tengan expresiones matemáticas en 2D. El número exacto de líneas depende del contenido de la información mostrada.

**Ejemplos ilustrativos:**

Define z(=	Salida
Prgm	z()
For n,1,3	Iteration 1:
DispAt 1, "N: ", n	Line 1: N:1
Disp "Hello"	Line 2: Hello
EndFor	
EndPrgm	Iteration 2:
	Line 1: N:2
	Line 2: Hello
	Line 3: Hello
	Iteration 3:
	Line 1: N:3
	Line 2: Hello
	Line 3: Hello
	Line 4: Hello
Define z1(=	z1()
Prgm	Line 1: N:3
For n,1,3	Line 2: Hello
DispAt 1, "N: ", n	Line 3: Hello
EndFor	Line 4: Hello
	Line 5: Hello
For n,1,3	
Disp "Hello"	
EndFor	
EndPrgm	

**Condiciones de error:**

Mensaje de error	Descripción
El número de línea de DispAt debe ser entre 1 y 8	La expresión evalúa el número de línea fuera del rango 1 a 8 (inclusive)
Muy pocos argumentos	Le falta uno o más argumentos a la función o al comando.
No hay argumentos	Igual que el cuadro de diálogo actual 'error de sintaxis'
Demasiados argumentos	Limite los argumentos. Mismo error que en Disp.

<b>Mensaje de error</b> Tipo de datos no válido	<b>Descripción</b> El primer argumento debe ser un número.
Anular: anular DispAt	Un tipo de error datatype "Hello World" se produce para la anulación (si se define la devolución de llamada)

**►DMS (►GMS) Catálogo >**

*Valor* ►DMS

En modo de ángulo en Grados:

*Lista* ►DMS

$$\overline{(45.371) \blacktriangleright \text{DMS} \qquad 45^{\circ}22'15.6''}$$

*Matriz* ►DMS

$$\overline{\{\{45.371,60\}\} \blacktriangleright \text{DMS} \qquad \{45^{\circ}22'15.6'',60^{\circ}\}}$$

**Nota:** Usted puede insertar este operador desde el teclado de la computadora al escribir @►DMS.

Interpreta el argumento como un ángulo y despliega el número GMS (GGGGGG°MM'SS.ss") equivalente. Vea °, ', " (página 209) para el formato GMS (grado, minutos, segundos).

**Nota:** ►DMS se convertirá de radianes a grados cuando se use en el modo de Radián. Si la entrada va seguida de un símbolo de grados °, no ocurrirá ninguna conversión. Usted puede usar ►DMS sólo al final de una línea de ingreso.

**dotP() (pPunto) Catálogo >**

**dotP**(Lista1, Lista2)⇒expresión

$$\overline{\text{dotP}(\{1,2\},\{5,6\}) \qquad 17}$$

Entrega el producto "punto" de dos listas.

**dotP**(Vector1, Vector2)⇒expresión

$$\overline{\text{dotP}([1 \ 2 \ 3],[4 \ 5 \ 6]) \qquad 32}$$

Entrega el producto punto" de dos vectores.

Ambos deben ser vectores de fila, o ambos deben ser vectores de columna.

## E

### e^()

 tecla



$e^{(Valor1)} \Rightarrow valor$

$e^1$	2.71828
-------	---------

Entrega **e** elevado a la potencia de *Valor1*.

$e^{3^2}$	8103.08
-----------	---------

**Nota:** Vea también **plantilla de exponente e**, página 2.

**Nota:** Presionar  para desplegar  $e^()$  (es diferente de presionar el caracter  en el teclado).

Usted puede ingresar un número complejo en la forma polar  $rei^\theta$ . Sin embargo, use esta forma sólo en el modo de ángulo en Radianes; esto causa un error de Dominio en el modo de ángulo en Grados o en Gradianes.

$e^{(Lista1)} \Rightarrow lista$

$e^{\{1,1,.05\}}$	$\{2.71828,2.71828,1.64872\}$
-------------------	-------------------------------

Entrega **e** elevado a la potencia de cada elemento en *Lista1*.


$e^{\text{(matrizCuadrada1)}} \Rightarrow \text{matrizCuadrada}$

$e^{\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}}$	$\begin{bmatrix} 782.209 & 559.617 & 456.509 \\ 680.546 & 488.795 & 396.521 \\ 524.929 & 371.222 & 307.879 \end{bmatrix}$
--	---

Entrega el exponencial de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular **e** elevado a la potencia de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

### eff()

Catálogo > 

$eff(tasaNominal, CpA) \Rightarrow valor$

$eff(5.75,12)$	5.90398
----------------	---------

Función financiera que convierte la tasa de interés nominal *tasaNominal* en una tasa efectiva anual, donde *CpA* se da como el número de periodos de capitalización por año.

*tasaNominal* debe ser un número real y *CpA* debe ser un número real > 0.

**Nota:** Vea también **nom()**, página 111.

## eigVC() (vcProp)

**eigVC(matrizCuadrada)**⇒*matriz*

En Formato Complejo Rectangular:

Entrega una matriz que contiene los vectores propios para una *matrizCuadrada* real o compleja, donde cada columna en el resultado corresponde a un valor propio. Tome en cuenta que un vector propio no es único; puede escalarse por medio de cualquier factor constante. Los vectores propios se normalizan, lo que significa que si  $V = [x_1, x_2, \dots, x_n]$ , entonces:

$$x_1^2 + x_2^2 + \dots + x_n^2 = 1$$

*matrizCuadrada* se balancea primero con transformaciones de similaridad hasta que las normas de fila y columna están tan cerca del mismo valor como es posible. La *matrizCuadrada* se reduce entonces a una forma de Hessenberg superior y los vectores propios se generan o se obtienen por medio de la factorización de Schur.

$$\begin{bmatrix} -1 & 2 & 5 \\ 3 & -6 & 9 \\ 2 & -5 & 7 \end{bmatrix} \rightarrow m1 \quad \begin{bmatrix} -1 & 2 & 5 \\ 3 & -6 & 9 \\ 2 & -5 & 7 \end{bmatrix}$$

$$\text{eigVC}(m1)$$

$$\begin{bmatrix} -0.800906 & 0.767947 & ( \\ 0.484029 & 0.573804+0.052258 \cdot i & 0.5738 \\ 0.352512 & 0.262687+0.096286 \cdot i & 0.2626 \end{bmatrix}$$

Para ver el resultado completo, presione  $\blacktriangle$  y después use  $\blacktriangleleft$  y  $\blacktriangleright$  para mover el cursor.

## eigVI() (vlProp)

**eigVI(matrizCuadrada)**⇒*lista*

En modo de formato complejo Rectangular:

Entrega una lista de valores propios de una *matrizCuadrada* real o compleja.

*matrizCuadrada* se balancea primero con transformaciones de similaridad hasta que las normas de fila y columna están tan cerca del mismo valor como es posible. La *matrizCuadrada* se reduce entonces a una forma de Hessenberg superior y los vectores propios se generan o se obtienen por medio de la matriz de Hessenberg superior.


$$\begin{bmatrix} -1 & 2 & 5 \\ 3 & -6 & 9 \\ 2 & -5 & 7 \end{bmatrix} \rightarrow m1 \quad \begin{bmatrix} -1 & 2 & 5 \\ 3 & -6 & 9 \\ 2 & -5 & 7 \end{bmatrix}$$

$$\text{eigVI}(m1)$$

$$\{-4.40941, 2.20471+0.763006 \cdot i, 2.20471-0. \cdot i\}$$

Para ver el resultado completo, presione  $\blacktriangle$  y después use  $\blacktriangleleft$  y  $\blacktriangleright$  para mover el cursor.

## Elseif (MásSi)

Catálogo > 

```

If ExprBoolean1 Then
  Bloque1
Elseif ExprBoolean2 Then
  Bloque2
:
:
Elseif ExprBooleanN Then
  BloqueN
EndIf
:
:

```

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

```

Define g(x)=Func
  If x<=5 Then
    Return 5
  Elseif x>5 and x<0 Then
    Return -x
  Elseif x≥0 and x≠10 Then
    Return x
  Elseif x=10 Then
    Return 3
  EndIf
EndFunc

```

Done

## EndFor (TerminarPara)

Vea For, página 58.

## EndFunc (TerminarFunc)

Vea Func, página 62.

## EndIf (TerminarSi)

Vea If, página 74.

## EndLoop (TerminarBucle)

Vea Loop, página 96.

## EndPrgm (TerminarPrgm)

Vea Prgm, página 124.

## euler ()

Catálogo > 

**euler**(*Expr*, *Var*, *varDep*, {*Var0*, *VarMax*}, *var0Dep*, *PasoVar* [, *pasoEuler*]) *matriz* ⇒

**euler**(*SistemaDeExpr*, *Var*, *ListaDeVarsDep*, {*Var0*, *VarMax*}, *ListaDeVars0Dep*, *PasoVar* [, *pasoEuler*]) *matriz* ⇒

**euler**(*ListaDeExpr*, *Var*, *ListaDeVarsDep*, {*Var0*, *VarMax*}, *ListaDeVars0Dep*, *PasoVar* [, *pasoEuler*]) *matriz* ⇒

Use el método de Euler para resolver el sistema

$$\frac{d \text{ depVar}}{d \text{ Var}} = \text{Expr}(\text{Var}, \text{depVar})$$

con *varDep*(*Var0*)=*var0Dep* en el intervalo [*Var0*,*VarMax*]. Entregará una matriz cuya primera fila define los valores del resultado de *Var* y cuya segunda fila define el valor del primer componente de solución a los valores de *Var* correspondientes, y así sucesivamente.

*Expr* es el lado derecho que define la ecuación diferencial ordinaria (EDO).

*SistemaDeExpr* es el sistema de lados derechos que define el sistema de EDOs (corresponde al orden de variables dependientes en *ListaDeVarsDep*).

*ListaDeExpr* es una lista de lados derechos que define el sistema de EDOs (corresponde al orden de variables dependientes en *ListaDeVarsDep*).

Ecuación diferencial:

$$y' = 0.001 \cdot y \cdot (100 - y) \text{ y } y(0) = 10$$

euler(0.001·y·(100-y),t,y,{0,100},10,1)				
0.	1.	2.	3.	4.
10.	10.9	11.8712	12.9174	14.042

Para ver el resultado completo, presione ▲ y después use ◀ y ▶ para mover el cursor.

Sistema de ecuaciones:

$$\begin{cases} y1' = y1 + 0.1 \cdot y1 \cdot y2 \\ y2' = 3 \cdot y2 - y1 \cdot y2 \end{cases}$$

$$\text{con } y1(0) = 2 \text{ y } y2(0) = 5$$

euler( $\begin{cases} y1+0.1 \cdot y1 \cdot y2 \\ 3 \cdot y2 - y1 \cdot y2 \end{cases}$ ,t,{y1,y2},{0,5},{2,5},1)					
0.	1.	2.	3.	4.	5.
2.	1.	1.	3.	27.	243.
5.	10.	30.	90.	90.	-2070.

*Var* es la variable independiente.

*ListaDeVarsDep* es una lista de variables dependientes.

{*Var0*, *VarMax*} es una lista de dos elementos que le dice a la función que se integre de *Var0* a *VarMax*.

*ListaDeVars0Dep* es una lista de valores iniciales para variables dependientes.

*PasoVar* es un número distinto de cero de manera que  $\text{sign}(PasoVar) = \text{sign}(VarMax - Var0)$  y las soluciones se entregan a  $Var0 + i \cdot PasoVar$  para todos  $i=0,1,2,\dots$  de tal manera que  $Var0 + i \cdot PasoVar$  está en  $[var0, VarMax]$  (puede que no haya un valor de solución en *VarMax*).

*pasoEuler* es un entero positivo (predeterminado a 1) que define el número de pasos de Euler entre los valores de resultado. El tamaño del paso real utilizado por el método de Euler es  $PasoVar / pasoEuler$ .

## eval ()

**eval(*Expr*)**  $\Rightarrow$  *cadena*

**eval()** solo es válida en el TI-Innovator™ Hub argumento del comando de los comandos de programación **Get**, **GetStr** y **Send**. El software evalúa la expresión *Expr* y reemplaza el enunciado **eval()** con el resultado como cadena de caracteres.

El argumento *Expr* se debe simplificar a un número real.

## Menú del Concentrador

Establezca el elemento azul de LED RGB a una intensidad media.

<i>lum</i> :=127	127
Send "SET COLOR.BLUE eval( <i>lum</i> )"	Done

Restablezca el elemento azul a APAGADO.

Send "SET COLOR.BLUE OFF"	Done
---------------------------	------

El argumento eval() se debe simplificar a un número real.

Send "SET LED eval("4") TO ON"	"Error: Invalid data type"
--------------------------------	----------------------------

Programa el elemento rojo a que aparezca gradualmente

```
Define fadein()=
Prgm
For i,0,255,10
Send "SET COLOR.RED eval(i)"
Wait 0.1
EndFor
Send "SET COLOR.RED OFF"
EndPrgm
```

Ejecute el programa.

<i>fadein()</i>	<i>Done</i>
<i>n</i> :=0.25	0.25
<i>m</i> :=8	8
<i>n · m</i>	2.
Send "SET COLOR.BLUE ON TIME eval( <i>n · m</i> )"	<i>Done</i>
<i>iostr.SendAns</i>	"SET COLOR.BLUE ON TIME 2"

Aunque **eval()** no muestra el resultado, puede ver la cadena de comandos del Concentrador después de ejecutar el comando al inspeccionar cualquiera de las siguientes variables especiales.

*iostr.SendAns*  
*iostr.GetAns*  
*iostr.GetStrAns*

**Nota:** Consulte además **Get** (página 64), **GetStr** (página 71) y **Send** (página 146).

## Exit (Salir)

Catálogo >

### Exit

Sale del bloque **For**, **While**, o **Loop** .

**Exit** no está permitido afuera de las tres estructuras de bucles (**For**, **While**, o **Loop**).

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

Listado de funciones:

Define <i>g()</i> =Func	<i>Done</i>
Local <i>temp,i</i>	
0 → <i>temp</i>	
For <i>i</i> ,1,100,1	
<i>temp</i> + <i>i</i> → <i>temp</i>	
If <i>temp</i> >20 Then	
Exit	
EndIf	
EndFor	
EndFunc	
<i>g()</i>	21



**exp()****ex** **exp(Valor1)**⇒*valor*Entrega **e** elevado a la potencia de *Valor1*.

$e^1$	2.71828
$e^{3^2}$	8103.08

**Nota:** Vea también la plantilla exponencial **e**, página 2.

Usted puede ingresar un número complejo en la forma polar  $rei^\theta$ . Sin embargo, use esta forma sólo en el modo de ángulo en Radianes; esto causa un error de Dominio en el modo de ángulo en Grados o en Gradianes.

**exp(Lista1)**⇒*lista*Entrega **e** elevada a la potencia de cada elemento en *Listal*.


$e\{1,1,.05\}$	$\{2.71828,2.71828,1.64872\}$
----------------	-------------------------------

**exp****(matrizCuadrada1)**⇒*matrizCuadrada*

Entrega el exponencial de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular **e** elevado a la potencia de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.


$e \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$	$\begin{bmatrix} 782.209 & 559.617 & 456.509 \\ 680.546 & 488.795 & 396.521 \\ 524.929 & 371.222 & 307.879 \end{bmatrix}$
--	---

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

**expr()****Catálogo** > **expr(Cadena)**⇒*expresión*

Entrega la cadena de caracteres contenida en *Cadena* como una expresión y la ejecuta de inmediato.

"Define cube(x)=x^3" → <i>funcstr</i>	
"Define cube(x)=x^3"	
<i>expr(funcstr)</i>	<i>Done</i>
<i>cube(2)</i>	8

**ExpReg****Catálogo** > **ExpReg** *X*, *Y* [, [*Frec*] [, [*Categoría*, *Incluir*]]

Genera la regresión exponencial  $y = a \cdot (b)^x$  en listas  $X$  y  $Y$  con frecuencia  $Frec$ . Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

$X$  y  $Y$  son listas de variables independientes y dependientes.

$Frec$  es una lista opcional de valores de frecuencia. Cada elemento en  $Frec$  especifica la frecuencia de la ocurrencia para cada punto de datos  $X$  y  $Y$  correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

$Categoría$  es una lista de códigos de categoría numérica o de cadena para los datos  $X$  y  $Y$  correspondientes.

*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.


Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $a \cdot (b)^x$
stat.a, stat.b	Coefficientes de regresión
stat.r <sup>2</sup>	Coefficiente de determinación lineal para datos transformados
stat.r	Coefficiente de correlación para datos transformados ( $x, \ln(y)$ )
stat.Resid	Residuales asociados con el modelo exponencial
stat.TransResid	Residuales asociadas con el ajuste lineal de datos transformados
stat.XReg	La lista de puntos de datos en <i>Lista X</i> modificada se usa de hecho en la regresión con base en las restricciones de las Categorías $Frec$ , <i>Lista de Categorías</i> $Incluir$

Variable de salida	Descripción
stat.YReg	La lista de puntos de datos en <i>Lista Y</i> modificada se usa de hecho en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoriæ Includ</i>
stat.FrecReg	Lista de frecuencias correspondientes a <i>stat.XReg</i> y <i>stat.YReg</i>

## F



### factor()

Catálogo > 

**factor(númeroRacional)** entrega el número racional factorizado en primos. Para números compuestos, el tiempo de cómputo aumenta exponencialmente con el número de dígitos en el segundo factor más grande. Por ejemplo, factorizar un entero de 30 dígitos podría llevarse más de un día, y factorizar un número de 100 dígitos podría llevarse más de un siglo.


<code>factor(152417172689)</code>	<code>123457·1234577</code>
<code>isPrime(152417172689)</code>	<code>false</code>

Para detener el cálculo manualmente:

- **Dispositivo portátil:** Mantenga presionada la tecla  y presione  varias veces.
- **Windows®:** Mantenga presionada la tecla **F12** y presione **Intro** varias veces.
- **Macintosh®:** Mantenga presionada la tecla **F5** y presione **Intro** varias veces.
- **iPad®:** La aplicación muestra un indicador. Puede seguir esperando o cancelar.

Si usted simplemente desea determinar si un número es primo, use **isPrime()** en su lugar. Es mucho más rápido, en particular si *númeroRacional* no es primo y si el segundo factor más grande tiene más de cinco dígitos.

### F Cdf()

Catálogo > 

F Cdf  
(

*límiteInferior*

,  
*límiteSuperior*  
*,númerodf,denomdf*⇒*número* si  
*límiteInferior* y *límiteSuperior* son  
 números, *lista* si *límiteInferior* y  
*límiteSuperior* son listas

**FCdf**

(  
*límiteInferior*  
 ,  
*límiteSuperior*  
*,númerodf,denomdf*⇒*número* si  
*límiteInferior* y *límiteSuperior* son  
 números, *lista* si *límiteInferior* y  
*límiteSuperior* son listas

Calcula la probabilidad de la distribución F entre el *Liminte inferior* y *Limite Superior* para los grados de libertad *dfNumer* y *dfDenom* especificados.

Para  $P(X \leq \text{Limite superior})$ , establecer *Limite Inferior*=0.

**Fill (Llenar)**

**Fill** *Valor*, *varMatriz*⇒*matriz*

Reemplaza cada elemento en la variable *varMatriz* con *Valor*.  
*varMatriz* ya debe existir.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	→ <i>amatrix</i>	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
Fill 1.01, <i>amatrix</i>		<i>Done</i>
<i>amatrix</i>		$\begin{bmatrix} 1.01 & 1.01 \\ 1.01 & 1.01 \end{bmatrix}$

**Fill** *Valor*, *varLista*⇒*lista*

Reemplaza cada elemento en la variable *varLista* con *Valor*.  
*varLista* ya debe existir.

{1,2,3,4,5}	→ <i>alist</i>	{1,2,3,4,5}
Fill 1.01, <i>alist</i>		<i>Done</i>
<i>alist</i>		{1.01,1.01,1.01,1.01,1.01}

**FiveNumSummary (ResumenNúmCinco)**

**FiveNumSummary** *X*[,*Frec*]  
 [,*Categoría*,*Incluir*]

Proporciona una versión abreviada de las estadísticas de 1 variable en la lista  $X$ . Un resumen de resultados se almacena en la variable *stat.results* (página 160).

$X$  representa una lista que contiene los datos.

*Frec* es una lista opcional de valores de frecuencia. Cada elemento en *Frec* especifica la frecuencia de la ocurrencia para cada punto de datos  $X$  y  $Y$  correspondientes. El valor predeterminado es 1.

*Categoría* es una lista de códigos de categoría numérica para los datos  $X$  correspondientes.

*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

Un elemento (inválido) vacío en cualquiera de las listas  $X$ , *Frec*, o *Categoría* da como resultado un inválido para el elemento correspondiente de todas esas listas. Para obtener más información sobre elementos vacíos, vea página 231.

Variable de salida	Descripción
stat.MínX	Mínimo de valores x.
stat.C <sub>1</sub> X	1er Cuartil de x.
stat.MedianaX	Mediana de x.
stat.C <sub>3</sub> X	3er Cuartil de x.
stat.MaxX	Máximo de valores x.

## floor() (piso)

**floor**(*Valor1*) ⇒ entero

$\text{floor}(-2.14) = -3.$

Entrega el entero más grande que es  $\leq$  el argumento. Esta función es idéntica a **int** ().

El argumento puede ser un número real o complejo.

## floor() (piso)

Catálogo >

**floor**(*Lista1*) ⇒ *lista*

$\text{floor}\left(\left\{\frac{3}{2}, 0, -5, 3\right\}\right)$	$\{1, 0, -6\}$
---	----------------

**floor**(*Matriz1*) ⇒ *matriz*

$\text{floor}\left(\begin{pmatrix} 1.2 & 3.4 \\ 2.5 & 4.8 \end{pmatrix}\right)$	$\begin{pmatrix} 1. & 3. \\ 2. & 4. \end{pmatrix}$
---	--

Entrega una lista o matriz del piso de cada elemento.

**Nota:** Vea también **ceiling()** e **int()**.

## For (Para)

Catálogo >

**For** *Var*, *Bajo*, *Alto* [, *Paso*]

*Bloque*

**EndFor**

Ejecuta las sentencias en *Bloque* iterativamente para cada valor de *Var*, desde *Bajo* hasta *Alto*, en incrementos de *Paso*.

*Var* no debe ser una variable de sistema.

*Paso* puede ser positivo o negativo. El valor predeterminado es 1.

*Bloque* puede ser una sentencia sencilla o una serie de sentencias separadas con el caracter ":".

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

Define $g()$ = Func	Done
Local <i>tempsum</i> , <i>step</i> , <i>i</i>	
0 → <i>tempsum</i>	
1 → <i>step</i>	
For <i>i</i> , 1, 100, <i>step</i>	
<i>tempsum</i> + <i>i</i> → <i>tempsum</i>	
EndFor	
EndFunc	
$g()$	5050

## format()

Catálogo >

**format**(*Valor* [, *cadenaFormato*]) ⇒ *cadena*

Entrega *Valor* como una cadena de caracteres con base en la plantilla de formato.

*cadenaFormato* es una cadena y debe ser en la forma: "F[n]", "S[n]", "E[n]", "G[n][c]", donde [ ] indican porciones adicionales.

$\text{format}(1.234567, "f3")$	"1.235"
$\text{format}(1.234567, "s2")$	"1.23E0"
$\text{format}(1.234567, "e3")$	"1.235E0"
$\text{format}(1.234567, "g3")$	"1.235"
$\text{format}(1234.567, "g3")$	"1,234.567"
$\text{format}(1.234567, "g3,r:")$	"1:235"

F[n]: Formato fijo. n es el número de dígitos a desplegar después del punto decimal.

S[n]: Formato científico. n es el número de dígitos a desplegar después del punto decimal.

E[n]: Formato de ingeniería. n es el número de dígitos después del primer dígito significativo. El exponente se ajusta a un múltiplo de tres, y el punto decimal se mueve hacia la derecha por cero, uno o dos dígitos.

G[n][c]: Igual que el formato fijo, pero también separa los dígitos hacia la izquierda de la raíz en grupos de tres. c especifica el caracter del separador del grupo y se predetermina a una coma. Si c es un punto, la raíz se mostrará como una coma.

[Rc]: Cualquiera de los especificadores anteriores puede tener un sufijo con la bandera de la raíz Rc, donde c es un caracter sencillo que especifica qué sustituir para el punto de la raíz.

**fPart() (parteF)**

**fPart**(Expr1) ⇒ expresión

fPart(-1.234)	-0.234
---------------	--------

**fPart**(Lista1) ⇒ lista

fPart({1,-2.3,7.003})	{0,-0.3,0.003}
-----------------------	----------------

**fPart**(Matriz1) ⇒ matriz

Entrega la parte fraccional del argumento.

Para una lista o matriz, entrega las partes fraccionales de los elementos.

El argumento puede ser un número real o complejo.

**FPdf()**

**FPdf**(XVal,númerodf,denomdf) ⇒ número si

*XVal* es un número, *lista* si *XVal* es una lista

Resuelve la probabilidad de distribución  $F$  en *XVal* para los *numerodf* (grados de libertad) y *denomdf* especificados.

**freqTable▶list()**

**freqTable▶list**

*(Lista1, listaEnteroFrec)⇒lista*

Entrega una lista que contiene los elementos desde *Lista1* expandida de acuerdo con las frecuencias en *listaEnteroFrec*. Esta función se puede usar para construir una tabla de frecuencia para la aplicación de Datos y Estadísticas.

*Lista1* puede ser cualquier lista válida.

*listaEnteroFrec* debe tener la misma dimensión que *Lista1* y debe contener sólo elementos enteros no negativos. Cada elemento especifica el número de veces que el elemento de *Lista1* correspondiente se repetirá en la lista de resultados. Un valor de cero excluye el elemento de *Lista1* correspondiente.

**Nota:** Usted puede insertar esta función desde el teclado de la computadora al escribir **freqTable@>list(...)**.

Los elementos vacíos (anulados) se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

freqTable▶list({1,2,3,4},{1,4,3,1})	{1,2,2,2,2,3,3,3,4}
freqTable▶list({1,2,3,4},{1,4,0,1})	{1,2,2,2,4}

**frequency (frecuencia)**

**frequency***(Lista1, listaCajones)⇒lista*

Entrega una lista que contiene los conteos de los elementos en *Lista1*. Los conteos se basan en los rangos (cajones) que usted define en *listaCajones*.

datalist:={1,2,e,3,π,4,5,6,"hello",7}	
{1,2,2.71828,3,3.14159,4,5,6,"hello",7}	
frequency(datalist,{2.5,4.5})	{2,4,3}

Explicación del resultado:

2 elementos de *listaDatos* son  $\leq 2.5$



Si *listaCajones* es  $\{b(1), b(2), \dots, b(n)\}$ , los rangos especificados son  $\{? \leq b(1), b(1) < ? \leq b(2), \dots, b(n-1) < ? \leq b(n), b(n) > ?\}$ . La lista resultante es un elemento más largo que *listaCajones*.

Cada elemento del resultado corresponde al número de elementos de *Lista1* que están en el rango de ese cajón. Expresado en términos de la función **countIf()**, el resultado es  $\{\text{conteoSi}(\text{lista}, ? \leq b(1)), \text{conteoSi}(\text{lista}, b(1) < ? \leq b(2)), \dots, \text{conteoSi}(\text{lista}, b(n-1) < ? \leq b(n)), \text{conteoSi}(\text{lista}, b(n) > ?)\}$ .

Los elementos de *Lista1* que no pueden estar "colocados en un cajón" se ignoran. Los elementos (inválidos) vacíos también se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

Dentro de la aplicación Listas y Hoja de Cálculo, usted puede usar un rango de celdas en lugar de ambos argumentos.

**Nota:** Vea también **countIf()**, página 32.

4 elementos de *listaDatos* son  $>2.5$  y  $\leq 4.5$

3 elementos de *listaDatos* son  $>4.5$

El elemento "hola" es una cadena y no se puede colocar en ninguno de los cajones definidos.

## FTest\_2Samp

**FTest\_2Samp** *Lista1, Lista2[, Frec1[, Frec2[, Hipot]]]*

**FTest\_2Samp** *Lista1, Lista2[, Frec1[, Frec2[, Hipot]]]*

(Entrada de lista de datos)

**FTest\_2Samp** *sx1, n1, sx2, n2[, Hipot]*

**FTest\_2Samp** *sx1, n1, sx2, n2[, Hipot]*

(Entrada de estadísticas de resumen)

Realiza una prueba  $F$  de dos muestras. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Para  $H_a: \sigma_1 > \sigma_2$ , configurar *Hipot* > 0

Para  $H_a: \sigma_1 \neq \sigma_2$  (predeterminado),  
configurar *Hipot* =0

Para  $H_a: \sigma_1 < \sigma_2$ , configurar *Hipot*<0

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.F	Estadística $\hat{U}$ calculada para la secuencia de datos
stat.ValP	Nivel más bajo de significancia en el cual la hipótesis nula se puede rechazar
stat.numerodf	grados de libertad del numerador = n1-1
stat.denomdf	grados de libertad del denominador = n2-1
stat.sx1, stat.sx2	Desviaciones estándar de muestra de las secuencias de datos en <i>Lista 1</i> y <i>Lista 2</i>
stat.x1_bar stat.x2_bar	Muestra significa las secuencias de datos en <i>Lista 1</i> y <i>Lista 2</i>
stat.n1, stat.n2	Tamaño de las muestras

## Func

### Func

*Bloque*

### EndFunc

Plantilla para crear una función definida por el usuario.

*Bloque* puede ser una sentencia sencilla, una serie de sentencias separadas con el caracter ":" o una serie de sentencias en líneas separadas. La función puede usar la instrucción **Return** para producir un resultado específico.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

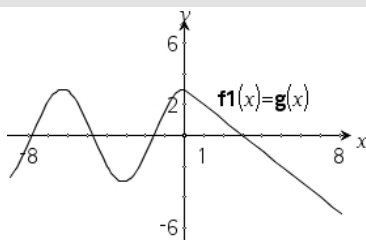
Defina una función de compuesto de variables:

---


```
Define g(x)=Func Done
    If x<0 Then
    Return 3*cos(x)
    Else
    Return 3-x
    EndIf
EndFunc
```

---

Resultado de graficar g(x)



## G

**gcd()** (mcd)Catálogo > **gcd**(Número1, Número2)⇒expresión

gcd(18,33)

3

Entrega el máximo común divisor de los dos argumentos. El **gcd** de dos fracciones es el **gcd** de sus numeradores dividido entre el **lcm** de sus denominadores.

En el modo de Auto o Aproximado, el **gcd** de los números de punto flotante es 1.0.

**gcd**(Lista1, Lista2)⇒lista


gcd({12,14,16},{9,7,5})

{3,7,1}

Entrega los máximos comunes divisores de los elementos correspondientes en *Lista1* y *Lista2*.

**gcd**(Matriz1, Matriz2)⇒matrizgcd( $\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}, \begin{pmatrix} 4 & 8 \\ 12 & 16 \end{pmatrix}$ ) $\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$ 

Entrega los máximos comunes divisores de los elementos correspondientes en *Matriz1* y *Matriz2*.

**geomCdf()**Catálogo > **geomCdf**

**geomCdf**(*p*, *límiteInferior*, *límiteSuperior*)⇒*número* si *límiteInferior* y *límiteSuperior* son números, *lista* si *límiteInferior* y *límiteSuperior* son listas

**geomCdf**(*p*, *límiteSuperior*) para  $P(1 \leq X \leq \text{límiteSuperior})$ ⇒*número* si *límiteSuperior* es un número, *lista* si *límiteSuperior* es una lista

Resuelve una probabilidad geométrica acumulativa desde *limiteInferior* hasta *limiteSuperior* con la probabilidad de éxito *pespecificada*.

Para  $P(X \leq \textit{limiteSuperior})$ , configure *limiteInferior* =1.

**geomPdf**(*p*,*XVal*)⇒*número* si *XVal* es un número, *lista* si *XVal* es una lista

Resuelve una probabilidad en *XVal*, el número de la prueba en la que ocurre el primer éxito, para la distribución geométrica discreta con la probabilidad de éxito *p*.

**Get****Menú del Concentrador**

**Get**[*promptString*,]*var*[, *statusVar*]

**Get**[*promptString*,]*func*(*arg1*, ...*argn*)  
[, *statusVar*]

Comando de programación: Recupera un valor de uno conectado TI-Innovator™ Hub y asigna el valor a *var* variable.

El valor se debe solicitar:

- Por adelantado, a través de un comando **Send "READ ..."** .  
— o bien —
- Mediante la inserción de una solicitud "**READ ...**" como argumento *promptString* opcional. Este método le permite usar un solo comando para solicitar el valor y recuperarlo.

Se lleva a cabo una simplificación implícita. Por ejemplo, una cadena recibida de "123" se interpreta como valor numérico. Para conservar la cadena, use **GetStr** en lugar de **Get**.

Ejemplo: Solicite el valor actual del sensor de nivel de luz incorporado del concentrador. Use **Get** para recuperar el valor y asignarlo a *lightval* variable.

Send "READ BRIGHTNESS"	Done
Get <i>lightval</i>	Done
<i>lightval</i>	0.347922

Inserte la solicitud READ dentro del comando **Get**.

Get "READ BRIGHTNESS", <i>lightval</i>	Done
<i>lightval</i>	0.378441

Si incluye el argumento opcional *statusVar*, se le asigna un valor que se basa en el éxito de la operación. Un valor de cero significa que no se recibieron datos.

En la segunda sintaxis, el argumento *func()* permite a un programa almacenar la cadena recibida como una definición de la función. La sintaxis opera como si el programa ejecutara el comando:


Se define  $func(arg1, \dots, argn) =$   
*received string*

Entonces el programa puede usar la función *func()* definida.

**Nota:** Puede usar el comando **Get** dentro de un programa definido por el usuario pero no dentro de una función.

**Nota:** Consulte además **GetStr**, página 71 y **Send**, página 146.

## getDenom()

Catálogo > 

**getDenom(Fracción1) ⇒ valor**

Transforma el argumento en una expresión que tiene un denominador común reducido, y después entrega su denominador.

$x:=5; y:=6$	6
$getDenom\left(\frac{x+2}{y-3}\right)$	3
$getDenom\left(\frac{2}{7}\right)$	7
$getDenom\left(\frac{1}{x} + \frac{y^2+y}{y^2}\right)$	30

## getKey()

Catálogo > 

**getKey ([0 | 1]) ⇒ returnString**

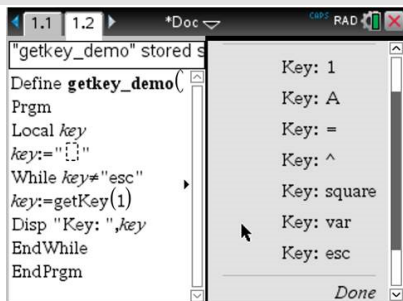
**Descripción:getKey():** permite a un programa de TI-Basic obtener entradas de teclado, dispositivo portátil, computadora y emulador en la computadora.

**Ejemplo:**

`getKey()`

**Ejemplo:**

- `keypressed:=getKey()`: devolverá una tecla o una cadena vacía si no se ha presionado ninguna tecla. Esta llamada volverá inmediatamente.
- `keypressed := getKey(1)` esperará hasta que se presione una tecla. Esta llamada hará una pausa en la ejecución del programa hasta que se presione una tecla.



### Manejo de teclas presionadas:

Tecla de dispositivo portátil/emulador	Computadora	Valor devuelto
Esc	Esc	"esc"
Tableta sensible al tacto: clic superior	n/a	"up"
Activado	n/a	"home"
Scratchapps	n/a	"scratchpad"
Tableta sensible al tacto: clic izquierdo	n/a	"left"
Tableta sensible al tacto: clic en el centro	n/a	"center"
Tableta sensible al tacto: clic derecho	n/a	"right"
Doc	n/a	"doc"
Tabulación	Tabulación	"tab"
Tableta sensible al tacto: clic inferior	Flecha hacia abajo	"down"
Menú	n/a	"menu"
Ctrl	Ctrl	sin devolución
Mayús	Mayús	sin devolución
Variable	n/a	"var"
Supr	n/a	"del"

Tecla de dispositivo portátil/emulador	Computadora	Valor devuelto
=	=	"="
trigonometría	n/a	"trig"
0 a 9	0 a 9	"0" ... "9"
Plantillas	n/a	"template"
Catálogo	n/a	"cat"
^	^	"^"
X^2	n/a	"square"
/ (tecla de división)	/	"/"
* (tecla de multiplicación)	*	"*"
e^x	n/a	"exp"
10^x	n/a	"10power"
+	+	"+"
-	-	"_"
(	(	"{"
)	)	"}"
.	.	". "
(-)	n/a	"_" (signo de resta)
Intro	Intro	"enter"
ee	n/a	"E" (notación científica E)
a - z	a-z	alfa = letra presionada (minúsculas) ("a" - "z")
mayús a-z	mayús a-z	alfa = letra presionada "A" - "Z"
		Nota: ctrl-mayús sirve para bloquear mayúsculas
?!	n/a	"?!"
pi	n/a	"pi"

Tecla de dispositivo portátil/emulador	Computadora	Valor devuelto
Bandera	n/a	sin devolución
,	,	","
Devolver	n/a	"return"
Espacio	Espacio	" " (espacio)
Inaccesible	Teclas de caracteres especiales como @, !, ^, etc.	Se devuelve el carácter
n/a	Teclas de funciones	Ningún carácter devuelto
n/a	Teclas especiales de control de la computadora	Ningún carácter devuelto
Inaccesible	Otras teclas de computadora que no están disponibles en la calculadora mientras getKey() está esperando que se presione una tecla. ({, },, ;, ...)	El mismo carácter que se obtiene en Notas (no en un cuadro de matemáticas)

**Nota:** Es importante señalar que la presencia de **getKey()** en un programa cambia cómo se manejan ciertos eventos en el sistema. Algunos de estos se describen a continuación.

**Terminar el programa y manejar el evento:** exactamente como si el usuario saliera del programa al presionar la tecla **ENCENDER**.

"**Compatibilidad**" a continuación significa que el sistema funciona como se espera y que el programa continúa ejecutándose.

Evento	Dispositivo	Computadora: TI-Nspire™ Student Software
Encuesta rápida	Terminar programa, manejar evento	Igual que en el dispositivo portátil (TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software, solamente)
Admin. de archivos remotos  (Incluye enviar el archivo 'Exit Press 2 Test' desde otro dispositivo portátil o computadora)	Terminar programa, manejar evento	Igual que en el dispositivo portátil.  (TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software solamente)
Terminar clase	Terminar programa,	Compatibilidad



Evento	Dispositivo	Computadora: TI-Nspire™ Student Software
	manejar evento	(TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software solamente)

Evento	Dispositivo	Computadora: todas las versiones de TI-Nspire™
TI-Innovator™ Hub : conectar/desconectar	Compatibilidad: puede emitir comandos correctamente al TI-Innovator™ Hub. Después de salir del programa, el TI-Innovator™ Hub sigue funcionando con el dispositivo portátil.	Igual que en el dispositivo portátil

## getLangInfo() (obtInfoIdioma)

Catálogo > 

**getLangInfo()** ⇒ *cadena*

getLangInfo()

"en"

Entrega una cadena que corresponde al nombre corto del idioma activo actualmente. Por ejemplo, usted puede usarlo en un programa o una función para determinar el idioma actual.

Inglés = "en"

Danés = "da"

Alemán = "de"

Finlandés = "fi"

Francés = "fr"

Italiano = "it"

Holandés = "nl"

Holandés belga = "nl\_BE"

Noruego = "no"

Portugués = "pt"

Español = "es"

Sueco = "sv"

## getLockInfo()

Catálogo > 

**getLockInfo**(*Var*)⇒*valor*

Entrega el estado de bloqueada/desbloqueada actual de la variable *Var*.

*valor* =0: *Var* está desbloqueada o no existe.

*valor* =1: *Var* está bloqueada y no se puede modificar ni borrar.

Vea **Lock**, página 93 y **unLock**, página 179.

<i>a</i> :=65	65
Lock <i>a</i>	Done
getLockInfo( <i>a</i> )	1
<i>a</i> :=75	"Error: Variable is locked."
DelVar <i>a</i>	"Error: Variable is locked."
Unlock <i>a</i>	Done
<i>a</i> :=75	75
DelVar <i>a</i>	Done

## getMode()

Catálogo > 

**getMode**(*EnteroNombreModo*)⇒*valor*

**getMode**(0)⇒*lista*

**getMode**(*EnteroNombreModo*) entrega un valor que representa la configuración actual del modo *EnteroNombreModo* .

**getMode**(0) entrega una lista que contiene pares de números. Cada par consiste en un entero de modo y un entero de configuración.



Para obtener un listado de modos y sus configuraciones, consulte la tabla de abajo.

Si usted guarda las configuraciones con **getMode**(0) → *var*, podrá usar **setMode** (*var*) en una función o un programa para restaurar temporalmente las configuraciones dentro de la ejecución de la función o el programa únicamente. Vea **setMode**(*var*), página 149.

getMode(0)	{ 1,7,2,1,3,1,4,1,5,1,6,1,7,1 }
getMode(1)	7
getMode(7)	1

Modo Nombre	Modo Entero	Cómo configurar enteros
Desplegar dígitos	1	1=Flotante, 2=Flotante1, 3=Flotante2, 4=Flotante3, 5=Flotante4, 6=Flotante5, 7=Flotante6, 8=Flotante7, 9=Flotante8, 10=Flotante9, 11=Flotante10, 12=Flotante11, 13=Flotante12, 14=Fijo0, 15=Fijo1, 16=Fijo2, 17=Fijo3, 18=Fijo4, 19=Fijo5, 20=Fijo6, 21=Fijo7, 22=Fijo8, 23=Fijo9, 24=Fijo10, 25=Fijo11, 26=Fijo12
Ángulo	2	1=Radián, 2=Grado, 3=Gradián
Formato exponencial	3	1=Normal, 2=Científico, 3=Ingeniería
Real o Complejo	4	1=Real, 2=Rectangular, 3=Polar
Auto o Aprox.	5	1=Auto, 2=Aproximado
Formato de Vector	6	1=Rectangular, 2=Cilíndrico, 3=Esférico
Base	7	1=Decimal, 2=Hexagonal, 3=Binario

## getNum()

Catálogo >  

**getNum(FracciónI)** ⇒ *valor*

Transforma el argumento en una expresión que tiene un denominador común reducido, y después entrega su numerador.

$x:=5; y:=6$  6

$\text{getNum}\left(\frac{x+2}{y-3}\right)$  7

$\text{getNum}\left(\frac{2}{7}\right)$  2

$\text{getNum}\left(\frac{1}{x} + \frac{1}{y}\right)$  11

## GetStr

## Menú del Concentrador

**GetStr**[*promptString*,] *var*[, *statusVar*]

Para ver ejemplos, consulte **Get**.

**GetStr**[*promptString*,] *func*(*arg1*, ...*argn*)  
[, *statusVar*]

Comando de programación: Opera de forma idéntica que el comando **Get**, excepto que el valor recuperado siempre se interpreta como una cadena. En contraste, el comando **Get** interpreta la respuesta como una expresión a menos que esté entre comillas ("").

**Nota:** Consulte además **Get**, página 64 y **Send**, página 146.

**getType()**Catálogo > 

**getType(var)** *cadena* ⇒

Entrega una cadena que indica el tipo de datos de la variable *var*.

Si *var* no se ha definido, entrega la cadena "NINGUNA".

$\{1,2,3\} \rightarrow temp$	$\{1,2,3\}$
getType( <i>temp</i> )	"LIST"
$3 \cdot i \rightarrow temp$	$3 \cdot i$
getType( <i>temp</i> )	"EXPR"
DelVar <i>temp</i>	Done
getType( <i>temp</i> )	"NONE"

**getVarInfo()**Catálogo > 

**getVarInfo()** ⇒ *matriz* o *cadena*

**getVarInfo(CadenaNombreLib)** ⇒ *matriz* o *cadena*

**getVarInfo()** entrega una matriz de información (nombre de variable, tipo, accesibilidad de librería y estado de bloqueada/desbloqueada) para todas las variables y los objetos de librería definidos en el problema actual.

Si no hay ninguna variable definida,

**getVarInfo()** entrega la cadena "NINGUNA".

**getVarInfo(CadenaNombreLib)** entrega una matriz de información para todos los objetos de librería definidos en la librería *CadenaNombreLib*. *CadenaNombreLib* debe ser una cadena (texto encerrado entre comillas) o una variable de cadena.

Si la librería *CadenaNombreLib* no existe, ocurrirá un error.

getVarInfo()	"NONE"												
Define $x=5$	Done												
Lock $x$	Done												
Define LibPriv $y=\{1,2,3\}$	Done												
Define LibPub $z(x)=3 \cdot x^2 - x$	Done												
getVarInfo()	<table border="1"> <tr> <td><math>x</math></td> <td>"NUM"</td> <td>"{ }"</td> <td>1</td> </tr> <tr> <td><math>y</math></td> <td>"LIST"</td> <td>"LibPriv"</td> <td>0</td> </tr> <tr> <td><math>z</math></td> <td>"FUNC"</td> <td>"LibPub"</td> <td>0</td> </tr> </table>	$x$	"NUM"	"{ }"	1	$y$	"LIST"	"LibPriv"	0	$z$	"FUNC"	"LibPub"	0
$x$	"NUM"	"{ }"	1										
$y$	"LIST"	"LibPriv"	0										
$z$	"FUNC"	"LibPub"	0										
getVarInfo( <i>tmp3</i> )	"Error: Argument must be a string"												
getVarInfo("tmp3")	[volcy12 "NONE" "LibPub" 0]												

## getVarInfo()

Catálogo > 

Tome en cuenta el ejemplo de la izquierda, en el cual el resultado de **getVarInfo()** se asigna a la variable *vs*. Intentar desplegar la fila 2 ó la fila 3 de *vs* entrega un error de "Lista o matriz inválida" porque al menos uno de los elementos en esas filas (variable *b*, por ejemplo) se reevalúa a una matriz.

Este error también podría ocurrir cuando se usa *Ans* para reevaluar un resultado de **getVarInfo()**.

El sistema arroja el error anterior porque la versión actual del software no soporta una estructura de matriz generalizada donde un elemento de una matriz puede ser una matriz o una lista.

$a:=1$	1												
$b:=\begin{bmatrix} 1 & 2 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 \end{bmatrix}$												
$c:=\begin{bmatrix} 1 & 3 & 7 \end{bmatrix}$	$\begin{bmatrix} 1 & 3 & 7 \end{bmatrix}$												
$vs:=\text{getVarInfo}()$	<table border="1"> <tr> <td><i>a</i></td> <td>"NUM"</td> <td>"<math>\begin{bmatrix} \end{bmatrix}</math>"</td> <td>0</td> </tr> <tr> <td><i>b</i></td> <td>"MAT"</td> <td>"<math>\begin{bmatrix} \end{bmatrix}</math>"</td> <td>0</td> </tr> <tr> <td><i>c</i></td> <td>"MAT"</td> <td>"<math>\begin{bmatrix} \end{bmatrix}</math>"</td> <td>0</td> </tr> </table>	<i>a</i>	"NUM"	" $\begin{bmatrix} \end{bmatrix}$ "	0	<i>b</i>	"MAT"	" $\begin{bmatrix} \end{bmatrix}$ "	0	<i>c</i>	"MAT"	" $\begin{bmatrix} \end{bmatrix}$ "	0
<i>a</i>	"NUM"	" $\begin{bmatrix} \end{bmatrix}$ "	0										
<i>b</i>	"MAT"	" $\begin{bmatrix} \end{bmatrix}$ "	0										
<i>c</i>	"MAT"	" $\begin{bmatrix} \end{bmatrix}$ "	0										
$vs\begin{bmatrix} 1 \end{bmatrix}$	$\begin{bmatrix} 1 & \text{"NUM"} & \text{"\begin{bmatrix} \end{bmatrix}" } & 0 \end{bmatrix}$												
$vs\begin{bmatrix} 1,1 \end{bmatrix}$	1												
$vs\begin{bmatrix} 2 \end{bmatrix}$	"Error: Invalid list or matrix"												
$vs\begin{bmatrix} 2,1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 \end{bmatrix}$												

## Goto (IrA)

Catálogo > 

### Goto nombreEtiqueta

Transfiere el control a la etiqueta *nombreEtiqueta*.

*nombreEtiqueta* se debe definir en la misma función al usar una instrucción **Lbl**.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

Define $g()$ =Func	Done
Local <i>temp,i</i>	
$0 \rightarrow temp$	
$1 \rightarrow i$	
Lbl <i>top</i>	
$temp+i \rightarrow temp$	
If $i < 10$ Then	
$i+1 \rightarrow i$	
Goto <i>top</i>	
EndIf	
Return <i>temp</i>	
EndFunc	
$g()$	55

## ►Grad

Catálogo > 

### Expr1 ►Grad⇒expresión

Convierte *Expr1* para la medida de ángulo en gradianes.

**Nota:** Usted puede insertar este operador desde el teclado de la computadora al escribir **@>Grad**.

En modo de ángulo en Grados:

$$\begin{bmatrix} 1.5 \end{bmatrix} \blacktriangleright \text{Grad} \quad \begin{bmatrix} 1.66667 \end{bmatrix}^g$$

En modo de ángulo en Radianes:

$$\begin{bmatrix} 1.5 \end{bmatrix} \blacktriangleright \text{Grad} \quad \begin{bmatrix} 95.493 \end{bmatrix}^g$$

**identity()**Catálogo > **identity(Entero)** ⇒ *matriz*

Produce la matriz de identidad con una dimensión de *Entero*.

*Entero* debe ser un entero positivo.

identity(4)	1	0	0	0
	0	1	0	0
	0	0	1	0
	0	0	0	1

**Si**Catálogo > 

**Si BooleanExpr**  
*Enunciado*

**Si BooleanExpr Entonces**  
*Bloque*

**EndIf**

Si *BooleanExpr* evalúa si es verdadero, ejecuta el enunciado simple *Enunciado* o el bloque de enunciados *Bloque* antes de proceder a ejecutar.

Si *BooleanExpr* evalúa si es falso, procede a ejecutar sin ejecutar el enunciado o bloque de enunciados.

El *Bloque* puede ser un solo enunciado o una secuencia de enunciados separados por el carácter ":".

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

**Si BooleanExpr Entonces**  
*Bloque1*

**Else**  
*Bloque2*

**EndIf**

Si *BooleanExpr* evalúa si es verdadero, ejecuta *Bloque1* y pasa al *Bloque2*.

Si *BooleanExpr* evalúa si es falso, pasa a *Bloque1* pero ejecuta *Bloque2*.

Define $g(x)=Func$	<i>Done</i>
If $x<0$ Then	
Return $x^2$	
EndIf	
EndFunc	
$g(-2)$	4

Define $g(x)=Func$	<i>Done</i>
If $x<0$ Then	
Return $-x$	
Else	
Return $x$	
EndIf	
EndFunc	
$g(12)$	12
$g(-12)$	12

*Bloque1* y *Bloque2* pueden ser un solo enunciado.

**Si** *BooleanExpr1* **Entonces**  
*Bloque1*

**Elseif** *BooleanExpr2* **Entonces**  
*Bloque2*

:

**Elseif** *BooleanExprN* **Entonces**  
*BlockN*

**EndIf**

Permite ramificar. Si *BooleanExpr1* evalúa si es verdadero, ejecuta *Block1*. Si *BooleanExpr1* evalúa si es falso, evalúa *BooleanExpr2*, y así sucesivamente.

---

```
Define g(x)=Func
  If x<5 Then
    Return 5
  ElseIf x>5 and x<0 Then
    Return -x
  ElseIf x≥0 and x≠10 Then
    Return x
  ElseIf x=10 Then
    Return 3
  EndIf
EndFunc
```

*Done*

$g(-4)$	4
$g(10)$	3

## ifFn()

**ifFn**(*BooleanExpr*, *Value* *If true* [, *Value* *If false* [, *Value* *If unknown*]])  
⇒ *expresión, lista, o matriz*

Evalúa la expresión booleana *BooleanExpr* (o cada elemento de *BooleanExpr*) y genera un resultado en base a las reglas siguientes:

- *BooleanExpr* puede probar un solo valor, una lista, o una matriz.
- Si un elemento de *BooleanExpr* evalúa si es verdadero, produce el elemento correspondiente de *Value\_If\_true*.
- Si un elemento de *BooleanExpr* evalúa si es falso, produce el elemento correspondiente de *Value\_If\_false*. Si omite *Value\_If\_false*, produce indef.
- Si un elemento de *BooleanExpr* no es ni verdadero ni falso, produce el elemento correspondiente *Value\_If\_unknown*. Si omite *Value\_If\_unknown*, produce indef.
- Si el segundo, tercero, o cuarto argumento de la función **ifFn()** es expresión sencilla, la prueba booleana

---

```
ifFn({1,2,3}<2.5,{5,6,7},{8,9,10})
      {5,6,10}
```

El valor de prueba de **1** es menor a 2,5; por lo que el correspondiente

El elemento *Value\_If\_True* de **5** se copia a la lista de resultados.

El valor de prueba de **2** es menor a 2,5; por lo que el correspondiente

El elemento *Value\_If\_True* de **6** se copia a la lista de resultados.

El valor de prueba de **3** no es menor a 2,5; por que su elemento *Value\_If\_False* correspondiente de **10** se copia a la lista de resultados.

---

```
ifFn({1,2,3}<2.5,4,{8,9,10})      {4,4,10}
```

*Value\_If\_true* es un valor sencillo y corresponde a cualquier posición seleccionada.

**ifFn()**Catálogo > 

se aplica a cada posición en *BooleanExpr*.

**Nota:** Si el enunciado simplificado *BooleanExpr* involucra una lista o matriz, todos los demás argumentos de la lista o matriz deben tener las mismas dimensiones, y el resultado tendrá también las mismas dimensiones.

$$\text{ifFn}(\{1,2,3\} < 2.5, \{5,6,7\}) \quad \{5,6,\text{undef}\}$$

*Value\_If\_false* no está especificado. Se utiliza *Indef*.

$$\text{ifFn}(\{2, "a" \} < 2.5, \{6,7\}, \{9,10\}, "err") \quad \{6, "err" \}$$

Se selecciona un elemento de *Value\_If\_true*.  
Se selecciona un elemento de *Value\_If\_undefined*.

**imag()**Catálogo > 

**imag**(*ValueI*) ⇒ *valor*

Produce la parte imaginaria del argumento.

$$\text{imag}(1+2 \cdot i) \quad 2$$

**imag**(*ListI*) ⇒ *lista*

Produce una lista de las partes imaginarias de los elementos.

$$\text{imag}(\{-3,4-i,i\}) \quad \{0,-1,1\}$$

**imag**(*MatrixI*) ⇒ *matriz*

Produce una matriz de las partes imaginarias de los elementos.

$$\text{imag} \begin{pmatrix} 1 & 2 \\ i \cdot 3 & i \cdot 4 \end{pmatrix} \quad \begin{bmatrix} 0 & 0 \\ 3 & 4 \end{bmatrix}$$
**Indirección**

Consulte #(), página 207.

**inString()**Catálogo > 

**inString**(*srcString*, *subString* [, *Arrancar*]) ⇒ *entero*


Produce la posición del carácter en la serie *srcString* en la cual inicia la primera ocurrencia de la serie *subString*.

$$\text{inString}(\text{"Hello there"}, \text{"the"}) \quad 7$$

$$\text{inString}(\text{"ABCEFG"}, \text{"D"}) \quad 0$$



## inString()


Catálogo > 

*Arrancar*, si se incluye, especifica la posición del caracter dentro de *srcString* en dónde inicia la búsqueda.

Predeterminado = 1 (el primer caracter de *srcString*).

Si *srcString* no contiene *subString* o *Arrancar* es > la longitud de *srcString*, produce cero.

## int()

Catálogo > 

**int(Valor)** ⇒ entero

**int(List1)** ⇒ lista

**int(Matrix1)** ⇒ matriz


Produce el mayor entero que sea menor o igual al argumento. Esta función es idéntica a **floor()**.

El argumento puede ser un número real o uno complejo.

Para una lista o matriz, produce el mayor entero de cada uno de los elementos.

$\text{int}(-2.5)$	-3.
$\text{int}([-1.234 \ 0 \ 0.37])$	$[-2. \ 0 \ 0.]$

## intDiv()

Catálogo > 

**intDiv(Number1, Number2)** ⇒ entero

**intDiv(List1, List2)** ⇒ lista


**intDiv(Matrix1, Matrix2)** ⇒ matriz

Produce la parte entera con signo de (*Number1* ÷ *Number2*).

Para las listas y matrices, produce la parte entera con signos de (argumento 1 ÷ argumento 2) para cada par del elemento.

$\text{intDiv}(-7,2)$	-3
$\text{intDiv}(4,5)$	0
$\text{intDiv}(\{12,-14,-16\},\{5,4,-3\})$	$\{2,-3,5\}$

## interpolar ()


Catálogo > 

**interpolar(xValue, xList, yList, yPrimeList)** ⇒ lista

Ecuación diferencial:  
 $y' = -3 \cdot y + 6 \cdot t + 5$  y  $y(0) = 5$

Esta función hace lo siguiente:

## interpolat ()

Catálogo > 

Dadas  $xList$ ,  $yList=f(xList)$ , y  $yPrimeList=f'(xList)$  para cierta función desconocida  $f$ , se usa una interpolación cúbica para aproximar la función  $f$  al  $xValue$ . Se supone que  $xList$  es una lista de números monótonicamente crecientes o decrecientes, aunque esta función puede entregar un valor incluso cuando no lo es. Esta función avanza a través de  $xList$  en busca de un intervalo [ $xList[i]$ ,  $xList[i+1]$ ] que contenga un  $xValue$ . Si encuentra dicho intervalo, produce un valor interpolado para  $f$  ( $xValue$ ); de otro modo, produce **undef**.

$xList$ ,  $yList$ , y  $yPrimeList$  deben tener la misma dimensión  $\geq 2$  y contener expresiones que se simplifiquen a números.

$xValue$  puede ser un número o una lista de números.

```
rK:=rk23(-3*y+6*t+5,I,y,{0,10},5,1)
{0. 1. 2. 3. 4. }
{5. 3.19499 5.00394 6.99957 9.00593 10.}
```

Para ver el resultado completo, presione  $\blacktriangle$  y después use  $\blacktriangleleft$  y  $\blacktriangleright$  para mover el cursor.

Use la función `interpolat()` para calcular los valores de la función para la lista `valox`:

```
xvaluelist:=seq(i,i,0,10,0.5)
{0,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9,9.5,10}
xlist:=mat▶list(rk[1])
{0.,1.,2.,3.,4.,5.,6.,7.,8.,9.,10.}
ylist:=mat▶list(rk[2])
{5.,3.19499,5.00394,6.99957,9.00593,10.99706,12.99957,15.00394,17.00957,19.01593,21.02282}
yprimelist:=-3*y+6*t+5|y=ylist and t=xlist
{-10.,1.41503,1.98819,2.00129,1.98221,2.00617,2.00129,1.98819,1.98221,2.00617,2.00129}
interpolate(xvaluelist,xlist,ylist,yprimelist)
{5.,2.67062,3.19499,4.02782,5.00394,6.00011,7.00011,8.00011,9.00011,10.00011}
```

## inv $\chi^2$ ()


Catálogo > 

`inv $\chi^2$ (Area,df)`

`invChi2(Área,df)`

Calcula la función de probabilidad acumulada inversa  $\chi^2$  (chi-cuadrada) que se especifica a partir de los grados de libertad  $df$  para una determinada  $Área$  bajo la curva.

## invF()

Catálogo > 

`invF(Area,dfNumer,dfDenom)`

`invF(Area,dfNumer,dfDenom)`

Calcula la función de probabilidad de distribución acumulada inversa  $F$  que se especifica a partir de  $dfNumer$  y  $dfDenom$  para una determinada  $Área$  bajo la curva.

## invBinom()

Catálogo > 

### invBinom

(CumulativeProb, NumTrials, Prob, OutputForm) ⇒ *escalar* o *matriz*

Dado el número de intentos (*Numintentos*) y la probabilidad de éxito de cada intento (*Prob*), esta función produce el número mínimo de éxitos, *k*, de tal forma que la probabilidad acumulativa de éxitos *k* es mayor que o igual a la probabilidad acumulativa dada (*CumulativeProb*).

*OutputForm=0*, muestra el resultado como un escalar (predeterminado).

*OutputForm=1*, muestra el resultado como una matriz.

Ejemplo: Mary y Kevin están jugando a los dados. Mary debe adivinar el número máximo de veces que aparece 6 en 30 lanzamientos. Si el número 6 sale ese número de veces o menos, Mary gana. Además, entre menor sea el número que ella adivine, mayores sus ganancias. ¿Cuál es el número más pequeño que Mary puede adivinar si desea que la probabilidad de ganar sea mayor al 77%?

$\text{invBinom}\left(0.77, 30, \frac{1}{6}\right)$	6
$\text{invBinom}\left(0.77, 30, \frac{1}{6}, 1\right)$	$\begin{bmatrix} 5 & 0.616447 \\ 6 & 0.776537 \end{bmatrix}$

## invBinomN()

Catálogo > 

invBinomN(CumulativeProb, Prob, NumSuccess, OutputForm) ⇒ *escalar* o *matriz*

Dada la probabilidad de éxito de cada intento (*Prob*), y el número de éxitos (*NumSuccess*), esta función produce el número mínimo de intentos, *N*, de tal forma que la probabilidad acumulativa de éxitos *x* sea menor que o igual a la probabilidad acumulativa dada (*CumulativeProb*).

*OutputForm=0*, muestra el resultado como un escalar (predeterminado).

*OutputForm=1*, muestra el resultado como una matriz.

Ejemplo: Monique está practicando tiros a gol. Ella sabe por su experiencia que su probabilidad de anotar un gol es del 70%. Ella planea practicar hasta anotar 50 goles. ¿Cuántos tiros debe intentar para asegurarse que la probabilidad de anotar por lo menos 50 goles sea de más de 0,99?

$\text{invBinomN}(0.01, 0.7, 49)$	86
$\text{invBinomN}(0.01, 0.7, 49, 1)$	$\begin{bmatrix} 85 & 0.010451 \\ 86 & 0.00709 \end{bmatrix}$

## invNorm()

Catálogo > 

invNorm(Área[,μ[,σ]])

Calcula la función de distribución normal acumulada inversa para un *Área* determinada bajo la curva de distribución normal especificada por la media,  $\mu$ , y por  $\sigma$ .

inv<sub>t</sub>(*Area*,*df*)

Calcula el valor acumulado de la función de probabilidad inversa *t* de Student que se especifica a partir de los grados de libertad *df* para una determinada *Area* bajo la curva.

## iPart()

iPart(*Número*) ⇒ enteroiPart(*List1*) ⇒ listaiPart(*Matrix1*) ⇒ matriz

iPart(-1.234)	-1.
iPart( $\left\{\frac{3}{2}, -2.3, 7.003\right\}$ )	{1, -2., 7.}

Produce la parte entera del argumento.

Para listas y matrices, produce la parte entera de cada elemento.

El argumento puede ser un número real o uno complejo.

## irr()

irr(*CF0*,*CFList* [,*CFFreq*]) ⇒ valor

La función financiera calcula la tasa interna de retorno de una inversión.

*CF0* es el flujo de caja inicial en la hora 0; que debe ser un número real.

*CFList* es una lista de cantidades de flujo de cada después del flujo de caja inicial *CF0*.

*CFFreq* es una lista opcional en la cual cada elemento especifica la frecuencia de ocurrencia para una cantidad agrupada (consecutiva) de flujo de caja, la cual el el elemento correspondiente de *CFList*. El valor predeterminado es 1; si usted ingresa valores, estos deben ser enteros positivos < 10.000.

**Nota:** Consulte también mirr(), página 102.

list1:={6000,-8000,2000,-3000}	{6000,-8000,2000,-3000}
list2:={2,2,2,1}	{2,2,2,1}
irr(5000,list1,list2)	-4.64484

## isPrime()

Catálogo > 

**isPrime(Número)** ⇒ *Expresión booleana constante*

Produce verdadero o falso para indicar si el *número* es un entero  $\geq 2$  que se puede dividir solamente por sí mismo y 1.

Si el *Número* excede en unos 306 dígitos y no tiene factores  $\leq 1021$ , **isPrime(Número)** muestra un mensaje de error.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

isPrime(5)	true
isPrime(6)	false

Función para encontrar el siguiente número primo después de un número especificado:

Define nextprim( <i>n</i> )=Func	Done
Loop	
$n+1 \rightarrow n$	
If isPrime( <i>n</i> )	
Return <i>n</i>	
EndLoop	
EndFunc	
nextprim(7)	11

## isVoid()

Catálogo > 

**isVoid(Var)** ⇒ *Expresión booleana constante*

**isVoid(Expr)** ⇒ *Expresión booleana constante*

**isVoid(List)** ⇒ *lista de expresiones booleanas constantes*

Produce verdadero o falso para indicar si el argumento es un tipo de datos vacío.

Para obtener mayor información sobre los elementos vacíos, consulte página 231.

$a := \_$	$\_$
isVoid( <i>a</i> )	true
isVoid({ 1, $\_$ , 3 })	{ false, true, false }

**Lbl (Etiqu)**Catálogo > **Lbl nombreEtiqueta**

Define una etiqueta con el nombre *nombreEtiqueta* dentro de una función.

Usted puede usar una instrucción **Goto** *nombreEtiqueta* para transferir el control a la instrucción que sigue inmediatamente a la etiqueta.

*nombreEtiqueta* debe cumplir con los mismos requisitos de nombrado que un nombre de variable.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

Define $g()$ =Func	Done
Local $temp,i$	
$0 \rightarrow temp$	
$1 \rightarrow i$	
Lbl $top$	
$temp+i \rightarrow temp$	
If $i < 10$ Then	
$i+1 \rightarrow i$	
Goto $top$	
EndIf	
Return $temp$	
EndFunc	

$g()$	55
-------	----

**lcm() (mínimo común múltiplo)**Catálogo > 

**lcm(Número1, Número2)**⇒expresión


**lcm(Lista1, Lista2)**⇒lista

**lcm(Matriz1, Matriz2)**⇒matriz

Entrega el mínimo común múltiplo de los dos argumentos. El **lcm** de dos fracciones es el **lcm** de sus numeradores dividido entre el **gcd** de sus denominadores. El **lcm** de los números de punto flotante fraccional es su producto.

Para dos listas o matrices, entrega los mínimos comunes múltiplos de los elementos correspondientes.

lcm(6,9)	18
lcm( $\left\{\frac{1}{3}, -14, 16\right\}, \left\{\frac{2}{15}, 7, 5\right\}$ )	$\left\{\frac{2}{3}, 14, 80\right\}$

**left() (izquierda)**Catálogo > 

**left(cadenaFuente[, Num])**⇒cadena

left("Hello",2)	"He"
-----------------	------

Entrega los caracteres de *Num* del extremo izquierdo contenidos en una cadena de caracteres *cadenaFuente*.

Si usted omite *Num*, entrega toda la *cadenaFuente*.

**left(Lista1[, Num])**⇒*lista*

---

```
left({1,3,-2,4},3)      {1,3,-2}
```

---

Entrega los elementos de *Num* del extremo izquierdo contenidos en *Lista1*.

Si usted omite *Num*, entrega toda la *Lista1*.

**left(Comparación)**⇒*expresión*

Entrega el lado del extremo izquierdo de una ecuación o desigualdad.

**libShortcut() (accesoDirectoLib)**

**libShortcut(CadenaNombreLib, CadenaNombreAccesoDirecto [, BanderaLibPriv])**⇒*lista de variables*

Crea un grupo de variables en el problema actual que contiene referencias para todos los objetos en el documento de librería especificado *cadenaNombreLib*. También agrega los miembros del grupo al menú de Variables. Entonces usted puede referirse a cada objeto al usar su *CadenaNombreAccesoDirecto*.

Configure *BanderaLibPriv=0* para excluir objetos de librería privada (predeterminado)

Configure *BanderaLibPriv=1* para incluir objetos de librería privada

Para copiar un grupo de variables, vea **CopyVar** (página 26).

Para borrar un grupo de variables, vea **DelVar** (página 41).

Este ejemplo supone un documento de librería almacenado y actualizado en forma apropiada nombrado **linalg2** que contiene objetos definidos como *limpmat*, *gauss1* y *gauss2*.

---

```
getVarInfo("linalg2")
  [clearmat "FUNC" "LibPub "
   gauss1   "PRGM" "LibPriv "
   gauss2   "FUNC" "LibPub " ]
```

---

```
libShortcut("linalg2", "la")
  {la.clearmat, la.gauss2}
```

---

```
libShortcut("linalg2", "la", 1)
  {la.clearmat, la.gauss1, la.gauss2}
```

---

**LinRegBx**  $X, Y, [Frec], [Categoría, Incluir]$

Resuelve la regresión lineal  $y = a + b \cdot x$  en las listas  $X$  y  $Y$  con frecuencia  $Frec$ . Un resumen de resultados se almacena en la variable *resultados.estad* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

$X$  y  $Y$  son listas de variables independientes y dependientes.

$Frec$  es una lista opcional de valores de frecuencia. Cada elemento en  $Frec$  especifica la frecuencia de la ocurrencia para cada punto de datos  $X$  y  $Y$  correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

$Categoría$  es una lista de códigos de categoría numérica o de cadena para los datos  $X$  y  $Y$  correspondientes.

$Incluir$  es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.


Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $a + b \cdot x$
stat.a, stat.b	Coefficientes de regresión
stat.r <sup>2</sup>	Coefficiente de determinación
stat.r	Coefficiente de correlación
stat.Resid	Residuales de la regresión
stat.XReg	La lista de puntos de datos en <i>Lista X</i> modificada se usa de hecho en la regresión con base en las restricciones de las Categorías $Frec$ , <i>Lista de Categorías</i> e <i>Incluir</i>



Variable de salida	Descripción
stat.YReg	La lista de puntos de datos en <i>Lista Y</i> modificada se usa de hecho en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categorias</i> e <i>Incluir</i>
stat.FrecReg	Lista de frecuencias correspondientes a <i>stat.XReg</i> y <i>stat.YReg</i>

## LinRegMx

Catálogo > 

**LinRegMx** *X*,*Y*, [*Frec*], [*Categoría*, *Incluir*]

Resuelve la regresión lineal  $y = m \cdot x + b$  en las listas *X* y *Y* con frecuencia *Frec*. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

*X* y *Y* son listas de variables independientes y dependientes.

*Frec* es una lista opcional de valores de frecuencia. Cada elemento en *Frec* especifica la frecuencia de la ocurrencia para cada punto de datos *X* y *Y* correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

*Categoría* es una lista de códigos de categoría numérica o de cadena para los datos *X* y *Y* correspondientes.


*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $y = m \cdot x + b$
stat.m, stat.b	Coefficientes de regresión
stat.r <sup>2</sup>	Coefficiente de determinación

Variable de salida	Descripción
stat.r	Coefficiente de correlación
stat.Resid	Residuales de la regresión
stat.XReg	La lista de puntos de datos en <i>Lista X</i> modificada se usa de hecho en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoríae Incluir</i>
stat.YReg	La lista de puntos de datos en <i>Lista Y</i> modificada se usa de hecho en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoríae Incluir</i>
stat.FrecReg	Lista de frecuencias correspondientes a <i>stat.XReg</i> y <i>stat.YReg</i>

## LinRegtIntervals

Catálogo > 

### LinRegtIntervals $X, Y[, F[, 0[, NivC]]]$

Para Pendiente. Resuelve en un intervalo de confianza de nivel C para la pendiente.

### LinRegtIntervals $X, Y[, F[, 1[, valX[, nivC]]]$

Para Respuesta. Resuelve un valor "y" previsto en un intervalo de predicción de nivel C para una observación sencilla, así como un intervalo de confianza de nivel C para la respuesta promedio.

Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual.

$X$  y  $Y$  son listas de variables independientes y dependientes.

$F$  es una lista opcional de valores de frecuencia. Cada elemento en  $F$  especifica la frecuencia de la ocurrencia para cada punto de datos  $X$  y  $Y$  correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $a+b \cdot x$
stat.a, stat.b	Coefficientes de regresión
stat.df	Grados de libertad
stat.r <sup>2</sup>	Coefficiente de determinación
stat.r	Coefficiente de correlación
stat.Resid	Residuales de la regresión


Únicamente para un tipo de pendiente

Variable de salida	Descripción
[stat.CBajo, stat.CAlto]	Intervalo de confianza para la pendiente.
stat.ME	Margen de error del intervalo de confianza
stat.EEPendiente	Error estándar de pendiente
stat.s	Error estándar sobre la línea

Para tipo de Respuesta únicamente

Variable de salida	Descripción
[stat.CBajo, stat.CAlto]	Intervalo de confianza para la respuesta promedio
stat.ME	Margen de error del intervalo de confianza
stat.EE	Error estándar de respuesta promedio
[stat.PredBaja, stat.PredAlta]	Intervalo de predicción para una observación sencilla
stat.MEPred	Margen de error del intervalo de predicción
stat.EEPred	Error estándar para la predicción
stat.ŷ	$a + b \cdot \text{val}X$

## LinRegtTest

Catálogo > 

LinRegtTest  $X, Y, \text{Frec}, [\text{Hipot}]$

Resuelve una regresión lineal en las listas  $X$  y  $Y$  y una prueba  $t$  en el valor de la pendiente  $\beta$  y el coeficiente de correlación  $\rho$  para la ecuación  $y=\alpha+\beta x$ . Prueba la hipótesis nula  $H_0:\beta=0$  (equivalentemente,  $\rho=0$ ) contra una de las tres hipótesis alternativas.

Todas las listas deben tener una dimensión igual.

$X$  y  $Y$  son listas de variables independientes y dependientes.

*Frec* es una lista opcional de valores de frecuencia. Cada elemento en *Frec* especifica la frecuencia de la ocurrencia para cada punto de datos  $X$  y  $Y$  correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

*Hipot* es un valor opcional que especifica una de las tres hipótesis alternativas contra la cual se probará la hipótesis nula ( $H_0:\beta=\rho=0$ ).

Para  $H_a: \beta \neq 0$  y  $\rho \neq 0$  (predeterminada), configuran *Hipot*=0

Para  $H_a: \beta < 0$  y  $\rho < 0$ , configuran *Hipot*<0

Para  $H_a: \beta > 0$  y  $\rho > 0$ , configuran *Hipot*>0


Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $a + b \cdot x$
stat.t	$t$ -Estadística para prueba de significancia
stat.ValP	Nivel más bajo de significancia en el cual la hipótesis nula se puede rechazar
stat.df	Grados de libertad
stat.a, stat.b	Coefficientes de regresión

Variable de salida	Descripción
stat.s	Error estándar sobre la línea
stat.EEPendiente	Error estándar de pendiente
stat.r <sup>2</sup>	Coefficiente de determinación
stat.r	Coefficiente de correlación
stat.Resid	Residuales de la regresión

## linSolve()

Catálogo > 

**linSolve**(*SistemaDeEcnsLineales*,  
*Var1*, *Var2*, ...)⇒*lista*

$$\text{linSolve}\left(\left\{\begin{array}{l} 2 \cdot x + 4 \cdot y = 3 \\ 5 \cdot x - 3 \cdot y = 7 \end{array}, \{x, y\}\right\}, \left\{\begin{array}{l} 37 \\ 26 \end{array}, \frac{1}{26}\right\}\right)$$

**linSolve**(*EcnLineal1* and *EcnLineal2*  
and ..., *Var1*, *Var2*, ...)⇒*lista*

$$\text{linSolve}\left(\left\{\begin{array}{l} 2 \cdot x = 3 \\ 5 \cdot x - 3 \cdot y = 7 \end{array}, \{x, y\}\right\}, \left\{\begin{array}{l} 3 \\ 2 \end{array}, \frac{1}{6}\right\}\right)$$

**linSolve**{*EcnLineal1*, *EcnLineal2*, ...},  
*Var1*, *Var2*, ...)⇒*lista*

$$\text{linSolve}\left(\left\{\begin{array}{l} \text{apple} + 4 \cdot \text{pear} = 23 \\ 5 \cdot \text{apple} - \text{pear} = 17 \end{array}, \{\text{apple}, \text{pear}\}\right\}, \left\{\begin{array}{l} 13 \\ 3 \end{array}, \frac{14}{3}\right\}\right)$$

**linSolve**(*SistemaDeEcnsLineales*,  
{*Var1*, *Var2*, ...}) ⇒*lista*

$$\text{linSolve}\left(\left\{\begin{array}{l} \text{apple} \cdot 4 + \frac{\text{pear}}{3} = 14 \\ -\text{apple} + \text{pear} = 6 \end{array}, \{\text{apple}, \text{pear}\}\right\}, \left\{\begin{array}{l} 36 \\ 13 \end{array}, \frac{114}{13}\right\}\right)$$

**linSolve**(*EcnLineal1* and *EcnLineal2*  
and ..., {*Var1*, *Var2*, ...})⇒*lista*


**linSolve**{*EcnLineal1*, *EcnLineal2*, ...},  
{*Var1*, *Var2*, ...}) ⇒*lista*

Entrega una lista de soluciones para las variables *Var1*, *Var2*, ...

El primer argumento se debe evaluar para un sistema de ecuaciones lineales o una ecuación lineal sencilla. De otro modo, ocurrirá un error de argumento.

Por ejemplo, evaluar **linSolve(x=1 y x=2,x)** produce un resultado de "Error de Argumento".

## ΔList()

Catálogo > 

**ΔList**(*Lista1*)⇒*lista*

$$\Delta\text{List}(\{20, 30, 45, 70\}) \quad \left\{\begin{array}{l} 10, 15, 25 \end{array}\right\}$$

**Nota:** Usted puede insertar esta función desde el teclado al escribir **deltaList** (...).

Entrega una lista que contiene las diferencias entre los elementos consecutivos en *Lista1*. Cada elemento de *Lista1* se sustrae del siguiente elemento de *Lista1*. La lista resultante siempre es un elemento más corto que la *Lista1* original.

## list▶mat()

list▶mat(*Lista* [, *elementosPorFila*])⇒matriz

Entrega una matriz llenada fila por fila con los elementos de *Lista1*.

*elementosPorFila*, si están incluidos, especifica el número de elementos por fila. El predeterminado es el número de elementos en *Lista* (una fila).

Si *Lista* no llena la matriz resultante, se agregan ceros.

**Nota:** Usted puede insertar esta función desde el teclado de la computadora al escribir `list@>mat(...)`.

list▶mat({1,2,3})	[ 1 2 3 ]						
list▶mat({1,2,3,4,5},2)	<table border="1"> <tr><td>1</td><td>2</td></tr> <tr><td>3</td><td>4</td></tr> <tr><td>5</td><td>0</td></tr> </table>	1	2	3	4	5	0
1	2						
3	4						
5	0						

## ln()

ln(*Valor1*)⇒valor

ln(2.)	0.693147
--------	----------

ln(*Lista*)⇒lista

Entrega el logaritmo natural del argumento.

Si el modo de formato complejo es Real:

Para una lista, entrega los logaritmos naturales de los elementos.

ln({-3,1.2,5})	"Error: Non-real calculation"
----------------	-------------------------------

Si el modo de formato complejo es Rectangular:

ln({-3,1.2,5})	{1.09861+3.14159·i,0.182322,1.60944}
----------------	--------------------------------------

ln

(*matrizCuadrada1*)⇒matrizCuadrada

En el modo de ángulo en Radianes y el formato complejo Rectangular:

Entrega el logaritmo natural de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular el logaritmo natural de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()** en.

$$\ln \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \begin{matrix} 1.83145+1.73485 \cdot i & 0.009193-1.49086 \\ 0.448761-0.725533 \cdot i & 1.06491+0.623491 \cdot i \\ -0.266891-2.08316 \cdot i & 1.12436+1.79018 \cdot i \end{matrix}$$

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

Para ver el resultado completo, presione  $\blacktriangle$  y después use  $\blacktriangleleft$  y  $\blacktriangleright$  para mover el cursor.

## LnReg

**LnReg** *X*, *Y*, [*Frec*] [, *Categoría*, *Incluir*]]

Resuelve la regresión logarítmica  $y = a + b \cdot \ln(x)$  en las listas *X* y *Y* con frecuencia *Frec*. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

*X* y *Y* son listas de variables independientes y dependientes.

*Frec* es una lista opcional de valores de frecuencia. Cada elemento en *Frec* especifica la frecuencia de la ocurrencia para cada punto de datos *X* y *Y* correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

*Categoría* es una lista de códigos de categoría numérica o de cadena para los datos *X* y *Y* correspondientes.

*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $a + b \cdot \ln(x)$

Variable de salida	Descripción
stat.a, stat.b	Coefficientes de regresión
stat.r <sup>2</sup>	Coefficiente de determinación lineal para datos transformados
stat.r	Coefficiente de correlación para datos transformados (ln(x), y)
stat.Resid	Residuales asociados con el modelo logarítmico
stat.TransResid	Residuales asociadas con el ajuste lineal de datos transformados
stat.XReg	La lista de puntos de datos en <i>Lista X</i> modificada se usa de hecho en la regresión con base en las restricciones de las Categorías <i>Frec, Lista de Categoriae Incluir</i>
stat.YReg	La lista de puntos de datos en <i>Lista Y</i> modificada se usa de hecho en la regresión con base en las restricciones de las Categorías <i>Frec, Lista de Categoriae Incluir</i>
stat.FrecReg	Lista de frecuencias correspondientes a <i>stat.XReg</i> y <i>stat.YReg</i>

## Local

Catálogo > 

**Local** *Var1* [, *Var2*] [, *Var3*] ...

Declara las *vars* especificadas como variables locales. Esas variables existen sólo durante la evaluación de una función y se borran cuando la función termina la ejecución.

**Nota:** Las variables locales ahorran memoria porque sólo existen en forma temporal. Asimismo, no alteran ninguno de los valores de variable global existentes. Las variables locales se deben usar para los bucles y para guardar temporalmente los valores en una función de líneas múltiples, ya que las modificaciones en las variables globales no están permitidas en una función.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

```

Define rollcount()=Func
  Local i
  1 → i
  Loop
  If randInt(1,6)=randInt(1,6)
  Goto end
  i+1 → i
  EndLoop
  Lbl end
  Return i
EndFunc

```

	Done
rollcount()	16
rollcount()	3



**LockVar1**[, *Var2*] [, *Var3*] ...

**LockVar.**

Bloquea las variables o el grupo de variables especificado. Las variables bloqueadas no se pueden modificar ni borrar.



Usted no puede bloquear o desbloquear la variable de sistema *Ans*, y no puede bloquear los grupos de variables de sistema *stat.* o *tvm.*

**Nota:** El comando **Lock** limpia el historial de Deshacer/Rehacer cuando se aplica a variables no bloqueadas.

Vea **unLock**, página 179 y **getLockInfo()**, página 70.

<i>a</i> :=65	65
Lock <i>a</i>	Done
getLockInfo( <i>a</i> )	1
<i>a</i> :=75	"Error: Variable is locked."
DelVar <i>a</i>	"Error: Variable is locked."
Unlock <i>a</i>	Done
<i>a</i> :=75	75
DelVar <i>a</i>	Done

**log()**

  **teclas**

**log**(*Valor1*[,*Valor2*])⇒*valor*

**log**(*Lista1*[,*Valor2*])⇒*lista*

Entrega el logaritmo *Valor2* base del primer argumento.

**Nota:** Vea también **Plantilla de logaritmos**, página 2.

Para una lista, entrega el logaritmo *Valor2* base de los elementos.

Si el segundo argumento se omite, se usa 10 como la base.

$\log_{10} (2.)$	0.30103
$\log_4 (2.)$	0.5
$\log_3 (10) - \log_3 (5)$	0.63093

Si el modo de formato complejo es Real:

$\log_{10} (\{-3,1.2,5\})$	"Error: Non-real calculation"
----------------------------	-------------------------------

Si el modo de formato complejo es Rectangular:

$\log_{10} (\{-3,1.2,5\})$	$\{0.477121+1.36438 \cdot i, 0.079181, 0.69897\}$
----------------------------	---

**log**(*matrizCuadrada* [,*Valor*])⇒*matrizCuadrada*

En el modo de ángulo en Radianes y el formato complejo Rectangular:

Entrega el logaritmo *Valor* base de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular el logaritmo *Valor* base de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

Si el argumento base se omite, se usa 10 como la base.

$$\log_{10} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{pmatrix} 0.795387+0.753438 \cdot i & 0.003993-0.6474 \cdot i \\ 0.194895-0.315095 \cdot i & 0.462485+0.2707 \cdot i \\ -0.115909-0.904706 \cdot i & 0.488304+0.7774 \cdot i \end{pmatrix}$$

Para ver el resultado completo, presione  $\blacktriangle$  y después use  $\blacktriangleleft$  y  $\blacktriangleright$  para mover el cursor.

## Logístic

**Logístic** *X*, *Y*, [*Frec*] [, *Categoría*, *Incluir*]

Resuelve la regresión logística  $y = (c / (1 + a \cdot e^{bx}) + d)$  en las listas *X* y *Y* con frecuencia *Frec*. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

*X* y *Y* son listas de variables independientes y dependientes.

*Frec* es una lista opcional de valores de frecuencia. Cada elemento en *Frec* especifica la frecuencia de la ocurrencia para cada punto de datos *X* y *Y* correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .


*Categoría* es una lista de códigos de categoría numérica o de cadena para los datos *X* y *Y* correspondientes.

*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $c/(1+a \cdot e^{bx+d})$
stat.a, stat.b, stat.c	Coefficientes de regresión
stat.Resid	Residuales de la regresión
stat.XReg	La lista de puntos de datos en la <i>Lista X</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoría</i> e <i>Incluir</i>
stat.YReg	La lista de puntos de datos en la <i>Lista Y</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoría</i> e <i>Incluir</i>
stat.FrecReg	Lista de frecuencias correspondientes a <i>stat.XReg</i> y <i>stat.YReg</i>

## LogísticD

Catálogo > 

**LogísticD** *X*, *Y* [, [*Iteraciones*] , [*Frec*] [, *Categoría*, *Incluir*] ]

Resuelve la regresión logística  $y = (c / (1 + a \cdot e^{bx}))$  en las listas *X* y *Y* con frecuencia *Frec*, utilizando un número específico de *Iteraciones*. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

*X* y *Y* son listas de variables independientes y dependientes.

*Frec* es una lista opcional de valores de frecuencia. Cada elemento en *Frec* especifica la frecuencia de la ocurrencia para cada punto de datos *X* y *Y* correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

*Categoría* es una lista de códigos de categoría numérica o de cadena para los datos *X* y *Y* correspondientes.

*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $c/(1+a \cdot e^{bx})$
stat.a, stat.b, stat.c, stat.d	Coefficientes de regresión
stat.Resid	Residuales de la regresión
stat.XReg	La lista de puntos de datos en la <i>Lista X</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoríae Incluir</i>
stat.YReg	La lista de puntos de datos en la <i>Lista Y</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoríae Incluir</i>
stat.FrecReg	Lista de frecuencias correspondientes a <i>stat.XReg</i> y <i>stat.YReg</i>

## Loop (Bucle)

### Loop

*Bloque*

### EndLoop

Ejecuta en forma repetida las sentencias en el *Bloque*. Tome en cuenta que el bucle se ejecutará sin parar, a menos que se ejecute una instrucción **Goto** o **Exit** dentro del *Bloque*.

*Bloque* es una secuencia de sentencias separadas con el caracter ":".

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

Define <i>rollcount()</i> =Func	
Local <i>i</i>	
$1 \rightarrow i$	
Loop	
If $\text{randInt}(1,6)=\text{randInt}(1,6)$	
Goto <i>end</i>	
$i+1 \rightarrow i$	
EndLoop	
Lbl <i>end</i>	
Return <i>i</i>	
EndFunc	
	<i>Done</i>
<i>rollcount()</i>	16
<i>rollcount()</i>	3

**LU** *Matriz*, *matrizB*, *matrizA*, *matrizP* [*Tol*]

Calcula la descomposición BA (baja-alta) de Doolittle de una matriz real o compleja. La matriz triangular baja se almacena en *matriz B*, la matriz triangular alta en *matriz A* y la matriz de permutación (que describe los cambios de fila realizados durante el cálculo) en *matriz P*.

$$matrizB \cdot matrizA = matrizP \cdot matriz$$

De manera opcional, cualquier elemento de matriz se trata como cero si su valor absoluto es menor que la *Tolerancia*. Esta tolerancia se usa sólo si la matriz tiene ingresos de punto flotante y no contiene ninguna variable simbólica a la que no se le haya asignado un valor. De otro modo, la *Tolerancia* se ignora.

- Si usted usa   o configura el modo **Auto** o **Aproximado** para aproximar, los cálculos se realizan al usar la aritmética de punto flotante.
- Si la *Tolerancia* se omite o no se usa, la tolerancia predeterminada se calcula como:  
 $5E-14 \cdot \text{máx}(\text{dim}(\text{Matriz})) \cdot \text{normaFila}(\text{Matriz})$

El algoritmo de factorización **LU** usa un pivoteo parcial con intercambios de filas.

$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix}$	$\rightarrow m1$	$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix}$
LU <i>m1</i> , <i>lower</i> , <i>upper</i> , <i>perm</i>		Done
<i>lower</i>		$\begin{bmatrix} 1 & 0 & 0 \\ \frac{5}{6} & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}$
<i>upper</i>		$\begin{bmatrix} 6 & 12 & 18 \\ 0 & 4 & 16 \\ 0 & 0 & 1 \end{bmatrix}$
<i>perm</i>		$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

## M


### mat▶list()

**mat▶list**(*Matriz*) ⇒ *lista*

Entrega una lista completada con los elementos de *Matriz*. Los elementos se copian desde *Matriz* fila por fila.

**Nota:** Usted puede insertar esta función desde el teclado de la computadora al escribir **mat@>list (...)**.

<b>mat▶list</b> ( $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ )	$\{1,2,3\}$
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ → <i>m1</i>	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
<b>mat▶list</b> ( <i>m1</i> )	$\{1,2,3,4,5,6\}$


**max()**Catálogo > **max(Valor1, Valor2)**⇒expresión $\max\{2.3, 1.4\}$  2.3**max(Lista1, Lista2)**⇒lista $\max\{\{1, 2\}, \{-4, 3\}\}$   $\{1, 3\}$ **max(Matriz1, Matriz2)**⇒matriz

Entrega el máximo de los dos argumentos. Si los argumentos son dos listas de matrices, entrega una lista de matriz que contiene el valor máximo de cada par de elementos correspondientes.

**max(Lista)**⇒expresión $\max\{\{0, 1, -7, 1.3, 0.5\}\}$  1.3Entrega el elemento máximo en *lista*.**max(Matriz1)**⇒matriz $\max\left(\begin{bmatrix} 1 & -3 & 7 \\ -4 & 0 & 0.3 \end{bmatrix}\right)$   $[1 \ 0 \ 7]$ 

Entrega un vector de fila que contiene el elemento máximo de cada columna en *Matriz1*.

Los elementos vacíos (anulados) se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

**Nota:** Vea también **min()**.**mean() (media)**Catálogo > **mean(Lista[, listaFrec])**⇒expresión $\text{mean}\{\{0.2, 0.1, -0.3, 0.4\}\}$  0.26Entrega la media de los elementos en *Lista*. $\text{mean}\{\{1, 2, 3\}, \{3, 2, 1\}\}$   $\frac{5}{3}$ 

Cada elemento de *listaFrec* cuenta el número de ocurrencias consecutivas del elemento correspondiente en *Lista*.

**mean(Matriz1[, matrizFrec])**⇒matriz

En formato de vector Rectangular:

Entrega un vector de fila de las medias de todas las columnas en *Matriz1*.

Cada elemento de *matrizFrec* cuenta el número de ocurrencias consecutivas del elemento correspondiente en *Matriz1*.

Los elementos vacíos (anulados) se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

mean	$\begin{pmatrix} 0.2 & 0 \\ -1 & 3 \\ 0.4 & -0.5 \end{pmatrix}$	$[-0.133333 \quad 0.833333]$
mean	$\begin{pmatrix} \frac{1}{5} & 0 \\ -1 & 3 \\ \frac{2}{5} & -\frac{1}{2} \end{pmatrix}$	$\begin{bmatrix} -\frac{2}{15} & \frac{5}{6} \end{bmatrix}$
mean	$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}, \begin{pmatrix} 5 & 3 \\ 4 & 1 \\ 6 & 2 \end{pmatrix}$	$\begin{bmatrix} \frac{47}{15} & \frac{11}{3} \end{bmatrix}$

## median() (mediana)

**median(Lista[, listaFrec])** ⇒ expresión

median({0.2,0.1,-0.3,0.4}) 0.2

Entrega la mediana de los elementos en *Lista*.

Cada elemento de *listaFrec* cuenta el número de ocurrencias consecutivas del elemento correspondiente en *Lista*.

**median(Matriz1[, matrizFrec])** ⇒ matriz

median( $\begin{pmatrix} 0.2 & 0 \\ 1 & -0.3 \\ 0.4 & -0.5 \end{pmatrix}$ )  $[0.4 \quad -0.3]$

Entrega un vector de fila que contiene las medianas de las columnas en *Matriz1*.

Cada elemento de *matrizFrec* cuenta el número de ocurrencias consecutivas del elemento correspondiente en *Matriz1*.

**Notas:**

- Todos los ingresos en la lista o matriz se deben simplificar a números.
- Los elementos vacíos (inválidos) en la lista o matriz se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

## MedMed

**MedMed X,Y [, Frec] [, Categoria, Incluir]**

Genera la línea media-mediana  $y = (m \cdot x + b)$  en las listas  $X$  y  $Y$  con frecuencia  $Frec$ . Un resumen de resultados se almacena en la variable *stat.results*. (Vea página 160.)

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

$X$  y  $Y$  son listas de variables independientes y dependientes.

$Frec$  es una lista opcional de valores de frecuencia. Cada elemento en  $Frec$  especifica la frecuencia de la ocurrencia para cada punto de datos  $X$  y  $Y$  correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

*Categoría* es una lista de códigos de categoría numérica o de cadena para los datos  $X$  y  $Y$  correspondientes.

*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de la recta mediana-mediana: $m \cdot x + b$
stat.m, stat.b	Coeficientes del modelo
stat.Resid	Residuales desde la recta mediana-mediana
stat.XReg	La lista de puntos de datos en la <i>Lista X</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoría</i> e <i>Incluir</i>
stat.YReg	La lista de puntos de datos en la <i>Lista Y</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoría</i> e <i>Incluir</i>
stat.FrecReg	Lista de frecuencias correspondientes a <i>stat.XReg</i> y <i>stat.YReg</i>



**mid()**Catálogo > **mid(cadenaFuente, Iniciar[, Contar])**⇒cadena

Entrega caracteres de *Conteo* de la cadena de caracteres *cadenaFuente*, comenzando con el número de caracteres *Iniciar*.

Si se omite *Conteo* o es mayor que la dimensión de *cadenaFuente*, entrega todos los caracteres de *cadenaFuente*, comenzando con el número de caracteres *Iniciar*.

El *Conteo* debe ser  $\geq 0$ . Si *Conteo* = 0, entrega una cadena vacía.

**mid(listaFuente, Iniciar [, Conteo])**⇒lista

Entrega elementos de *Conteo* de *listaFuente*, comenzando con el número de elementos del *Inicio*.

Si se omite *Conteo* o es mayor que la dimensión de *listaFuente*, entrega todos los elementos de *listaFuente*, comenzando con el número de elementos del *Inicio*.

El *Conteo* debe ser  $\geq 0$ . Si *Conteo* = 0, entrega una lista vacía.

**mid(listaCadenaFuente, Iniciar[, Conteo])**⇒lista

Entrega cadenas de *Conteo* de la lista de cadenas *listaCadenaFuente*, comenzando con el número de elementos del *Inicio*.

mid("Hello there",2)	"ello there"
mid("Hello there",7,3)	"the"
mid("Hello there",1,5)	"Hello"
mid("Hello there",1,0)	"[]"

mid({9,8,7,6},3)	{7,6}
mid({9,8,7,6},2,2)	{8,7}
mid({9,8,7,6},1,2)	{9,8}
mid({9,8,7,6},1,0)	{[]}

mid({"A","B","C","D"},2,2)	{"B","C"}
----------------------------	-----------

**mín()**Catálogo > **mín(Valor1, Valor2)**⇒expresión**mín(Lista1, Lista2)**⇒lista**mín(Matriz1, Matriz2)**⇒matriz

min(2,3,1,4)	1.4
min({1,2},{-4,3})	{-4,2}

Entrega el mínimo de los dos argumentos. Si los argumentos son dos listas o matrices, entrega una lista o matriz que contiene el valor mínimo de cada par de elementos correspondientes.

**mín(Lista)** ⇒ *expresión*

Entrega el elemento mínimo de *Lista*.

**mín(Matriz1)** ⇒ *matriz*

Entrega un vector de fila que contiene el elemento mínimo de cada columna en *Matriz1*.

**Nota:** Vea también **max()**.

$$\min(\{0,1,-7,1,3,0,5\}) \quad -7$$

$$\min\left(\begin{pmatrix} 1 & -3 & 7 \\ -4 & 0 & 0.3 \end{pmatrix}\right) \quad [-4 \quad -3 \quad 0.3]$$

**mirr**

(*tasaFinanciación*, *tasaReinversión*, *FE0*, *listaFE*, [*frecFE*])

La función financiera que entrega la tasa interna de rendimiento modificada de una inversión.

*tasaFinanciación* es la tasa de interés que usted paga sobre las cantidades de flujo de efectivo.

*tasaReinversión* es la tasa de interés a la que se reinvierten los flujos de efectivo.

*FE0* es el flujo de efectivo inicial en tiempo 0; debe ser un número real.

*ListaFE* es una lista de cantidades de flujo de efectivo después del flujo de efectivo inicial *FE0*.

*FrecFE* es una lista opcional en la cual cada elemento especifica la frecuencia de ocurrencia para una cantidad de flujo de efectivo (consecutivo) agrupado, que es el elemento correspondiente de la *ListaFE*. La predeterminada es 1; si usted ingresa valores, éstos deben ser enteros positivos < 10,000.

$$\begin{array}{l} list1 := \{6000, -8000, 2000, -3000\} \\ \qquad \qquad \qquad \qquad \qquad \qquad \{6000, -8000, 2000, -3000\} \\ list2 := \{2, 2, 2, 1\} \qquad \qquad \qquad \{2, 2, 2, 1\} \\ mirr(4.65, 12, 5000, list1, list2) \quad 13.41608607 \end{array}$$

**Nota:** Vea también irr(), página 80.

## mod()

**mod(Valor1, Valor2)** ⇒ expresión

mod(7,0)	7
----------	---

**mod(Lista1, Lista2)** ⇒ lista

mod(7,3)	1
----------	---

**mod(Matriz1, Matriz2)** ⇒ matriz

mod(-7,3)	2
-----------	---

Entrega el segundo argumento del módulo del primer argumento conforme se define por medio de las identidades:

mod(7,-3)	-2
-----------	----

$$\text{mod}(x,0) = x$$

mod(-7,-3)	-1
------------	----

$$\text{mod}(x,y) = x - y \text{ piso}(x/y)$$

mod({12,-14,16},{9,7,-5})	{3,0,-4}
---------------------------	----------

Cuando el segundo argumento no es cero, el resultado es periódico en ese argumento. El resultado es cero o tiene el mismo signo que el segundo argumento.

Si los argumentos son dos listas o dos matrices, entrega una lista o matriz que contiene el módulo de cada par de elementos correspondientes.

**Nota:** Vea también remain(), . página 136

## mRow() (filaM)

**mRow(Valor, Matriz1, Índice)** ⇒ matriz

Entrega una copia de *Matriz1* con cada elemento en la fila *Índice* de *Matriz1* multiplicado por *Valor*.

mRow( $\frac{-1}{3}$ , $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ , 2)	$\begin{bmatrix} 1 & 2 \\ -1 & -4 \\ 3 & 3 \end{bmatrix}$
--	---

## mRowAdd() (agrFilaM)

**mRowAdd(Valor, Matriz1, Índice1, Índice2)** ⇒ matriz

Entrega una copia de *Matriz1* con cada elemento en la fila *Índice2* de *Matriz1* reemplazado por:

mRowAdd(-3, $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ , 1, 2)	$\begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix}$
--	---

$$\text{Valor} \cdot \text{fila } \text{Índice1} + \text{fila } \text{Índice2}$$

**MultReg**  $Y, X1[,X2[,X3,...[,X10]]]$ 

Calcula la regresión lineal múltiple de la lista  $Y$  en las listas  $X1, X2, \dots, X10$ . Un resumen de resultados se almacena en la variable *resultados.estad* (página 160).

Todas las listas deben tener una dimensión igual.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \dots$
stat.b0, stat.b1, ...	Coefficientes de regresión
stat.R <sup>2</sup>	Coefficiente de determinación múltiple
stat.ŷLista	$\hat{y}_{Lista} = b_0 + b_1 \cdot x_1 + \dots$
stat.Resid	Residuales de la regresión

**MultRegIntervals****MultRegIntervals**  $Y, X1[,X2[,X3,...[,X10]]], listaValX[,nivelC]$ 

Computa un valor y previsto, un intervalo de predicción de nivel  $C$  para una observación sencilla, así como un intervalo de confianza de nivel  $C$  para la respuesta media.

Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \dots$
stat.ŷ	Un estimado de punto: $\hat{y} = b_0 + b_1 \cdot x_1 + \dots$ para <i>listaValX</i>

Variable de salida	Descripción
stat.dfError	Grados de libertad de error
stat.CBajo, stat.CAlto	Intervalo de confianza para una respuesta media
stat.ME	Margen de error del intervalo de confianza
stat.EE	Error estándar de respuesta media
stat.PredBaja, stat.PredAlta	Intervalo de predicción para una observación sencilla
stat.MEPred	Margen de error del intervalo de predicción
stat.EEPred	Error estándar para la predicción
stat.ListaB	Lista de coeficientes de regresión, {b0,b1,b2,...}
stat.Resid	Residuales de la regresión

## MultRegTests (PruebasRegMult)

Catálogo > 

### MultRegTests $Y, X1[,X2[,X3,...[,X10]]]$

La prueba de regresión lineal múltiple resuelve una regresión lineal múltiple sobre los datos dados y proporciona la estadística de la prueba  $F$  global y las estadísticas de la prueba  $t$  para los coeficientes.

Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

### Salidas

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $b_0+b_1 \cdot x_1+b_2 \cdot x_2+ \dots$
stat.F	Estadística de la prueba $F$ global
stat.ValP	Valor P asociado con la estadística de $F$ global
stat.R <sup>2</sup>	Coefficiente de determinación múltiple
stat.AjustR <sup>2</sup>	Coefficiente de determinación múltiple ajustado
stat.s	Desviación estándar del error

Variable de salida	Descripción
stat.DW	Estadística de Durbin-Watson; se usa para determinar si la autocorrelación de primer grado está presente en el modelo
stat.dfReg	Grados de libertad de la regresión
stat.SCReg	Suma de cuadrados de la regresión
stat.CMReg	Cuadrado medio de la regresión
stat.dfError	Grados de libertad de error
stat.SSError	Suma de cuadrados del error
stat.CMError	Cuadrado medio del error
stat.ListaB	{b0,b1,...} Lista de coeficientes
stat.ListaT	Lista de estadísticas t, una para cada coeficiente en la ListaB
stat.ListaP	Valores P de la lista para cada estadística t
stat.ListaEE	Lista de errores estándar para los coeficientes en la ListaB
stat.ŷLista	$\hat{y} \text{Lista} = b_0 + b_1 \cdot x_1 + \dots$
stat.Resid	Residuales de la regresión
stat.ResidE	Residuales estandarizados; se obtienen al dividir un residual entre su desviación estándar
stat.DistCook	Distancia de Cook; medida de la influencia de una observación con base en el residual y el apalancamiento
stat.Apalancamiento	Medida de cuán lejos están los valores de la variable independiente de sus valores medios

## N

### nand

teclas  

*BooleanExpr1 nand BooleanExpr2*  
devuelve *expresión booleana*

*BooleanList1 nand BooleanList2*  
devuelve *lista booleana*

*BooleanMatrix1 nand BooleanMatrix2*  
devuelve *matriz booleana*

Devuelve la negación de una operación **and** lógica en los dos argumentos. Devuelve verdadero, falso o una forma simplificada de la ecuación.

Para listas y matrices, devuelve comparaciones elemento por elemento.

*Entero1 nand Entero2* ⇒ *entero*

Compara dos números reales enteros bit a bit utilizando una operación **nand**. Internamente, ambos números enteros se convierten en números binarios de 64 bit con signos. Cuando se comparan bits correspondientes, el resultado es 0 si ambos bits son 1; de lo contrario el resultado es 1. El valor devuelto representa los resultados bit, y se muestran según el modelo Base.

Puede ingresar los números enteros en cualquier base numérica. Para una entrada binaria o hexadecimal, debe utilizar el prefijo 0b o 0h respectivamente. Sin un prefijo, se trata a los números enteros como decimales (base 10).

3 and 4	0
3 nand 4	-1
{1,2,3} and {3,2,1}	{1,2,1}
{1,2,3} nand {3,2,1}	{-2,-3,-2}

## nCr()

*nCr(Valor1, Valor2)* ⇒ *expresión*

Para entero *Valor1* y *Valor2* con *Valor1* ≥ *Valor2* ≥ 0, **nCr()** es el número de combinaciones de los elementos del *Valor1* tomadas del *Valor2* a la vez. (Esto también se conoce como un coeficiente binomial).

*nCr(Valor, 0)* ⇒ 1

*nCr(Valor, enteroNeg)* ⇒ 0

*nCr(Valor, enteroPos)* ⇒ *Valor · (Valor-1) · ... · (Valor-enteroPos+1) / enteroPos!*

*nCr(Valor, noEntero)* ⇒ *expresión!* / *((Valor-noEntero)! · noEntero!)*

<i>nCr(z,3)</i>   <i>z=5</i>	10
<i>nCr(z,3)</i>   <i>z=6</i>	20

**nCr()**

Catálogo &gt;

**nCr(Lista1, Lista2)⇒lista** $nCr(\{5,4,3\},\{2,4,2\})$   $\{10,1,3\}$ 

Entrega una lista de combinaciones con base en los pares de elementos correspondientes en las dos listas. Los argumentos deben tener el mismo tamaño que la lista.

**nCr(Matriz1, Matriz2)⇒matriz** $nCr\left(\begin{bmatrix} 6 & 5 \\ 4 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}\right)$   $\begin{bmatrix} 15 & 10 \\ 6 & 3 \end{bmatrix}$ 

Entrega una matriz de combinaciones con base en los pares de elementos correspondientes en las dos matrices. Los argumentos deben tener el mismo tamaño que la matriz.

**nDerivative()**

Catálogo &gt;

**nDerivative(Expr1, Var=Valor [,Orden])⇒valor** $nDerivative(|x|,x=1)$  1**nDerivative(Expr1, Var[,Orden]) | Var=Valor⇒valor** $nDerivative(|x|,x)|_{x=0}$  undef $nDerivative(\sqrt{x-1},x)|_{x=1}$  undef

Entrega la derivada numérica calculada con el uso de métodos de autodiferenciación.

Cuando se especifica el *Valor*, se eliminan todas las asignaciones anteriores de la variable o cualquier sustitución "|" para la variable.

Si la variable *Var* no contiene un valor numérico, usted debe proporcionar el *Valor*.

El *Orden* de la derivada debe ser 1 ó 2.

**Nota:** El algoritmo de la **nDerivative()** tiene una limitación: funciona recursivamente a través de la expresión no simplificada, determinando el valor numérico de la primera derivada (y de la segunda, si aplica) y la evaluación de cada subexpresión, lo que puede conllevar a un resultado inesperado.

 $nDerivative\left(x \cdot \left(x^2+x\right)^{\frac{1}{3}}, x, 1\right)|_{x=0}$  undef $centralDiff\left(x \cdot \left(x^2+x\right)^{\frac{1}{3}}, x\right)|_{x=0}$  0.000033



Tome en consideración el ejemplo de la derecha. La primera derivada de  $x \cdot (x^2+x)^{1/3}$  en  $x=0$  es igual a 0. Sin embargo, dado que la primera derivada de la subexpresión  $(x^2+x)^{1/3}$  es indefinida en  $x=0$ , y este valor se usa para calcular la derivada de la expresión total, **nDerivative()** reporta el resultado como indefinido y despliega un mensaje de advertencia.

Si usted encuentra esta limitación, verifique la solución en forma gráfica. Usted también puede tratar de usar **centralDiff()**.

**newList() (nuevaLista)**

**newList**(*elementosNum*) $\Rightarrow$ *lista*

newList(4)	{0,0,0,0}
------------	-----------

Entrega una lista con una dimensión de *elementosNum*. Cada elemento es cero.

**newMat()**

**newMat**(*filasNum*,  
*columnasNum*) $\Rightarrow$ *matriz*

newMat(2,3)	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
-------------	--

Entrega una matriz de ceros con la dimensión *filasNum* por *columnasNum*.

**nfMax()**

**nfMax**(*Expr*, *Var*) $\Rightarrow$ *valor*

$\text{nfMax}(x^2 - 2 \cdot x - 1, x)$	-1.
--	-----

**nfMax**(*Expr*, *Var*, *limiteInferior*) $\Rightarrow$ *valor*

$\text{nfMax}(0.5 \cdot x^3 - x - 2, x, -5, 5)$	5.
---	----

**nfMax**(*Expr*, *Var*, *limiteInferior*,  
*limiteSuperior*) $\Rightarrow$ *valor*

**nfMax**(*Expr*, *Var*) | *limiteInferior* $\leq$ *Var*  
 $\leq$ *limiteSuperior* $\Rightarrow$ *valor*

Entrega un valor numérico candidato de la variable *Var* donde ocurre el local máximo de *Expr*.

Si proporciona el *límite inferior* y el *límite superior*, la función buscará en el intervalo cerrado [*límite Inferior*,*límite superior*] el valor del máximo local en la función.

**nfMín()**

**nfMin**(*Expr*, *Var*)⇒*valor*

$\text{nfMin}(x^2+2 \cdot x+5,x)$	-1.
-----------------------------------	-----

**nfMin**(*Expr*, *Var*, *límiteInferior*)⇒*valor*

$\text{nfMin}(0.5 \cdot x^3-x-2,x,-5,5)$	-5.
--	-----

**nfMin**(*Expr*, *Var*, *límiteInferior*, *límiteSuperior*)⇒*valor*

**nfMin**(*Expr*, *Var*) | *límiteInferior* ≤ *Var* ≤ *límiteSuperior* ⇒ *valor*

Entrega un valor numérico candidato de la *Var* donde ocurre el local mínimo de *Expr* .

Si proporciona el *límite inferior* y el *límite superior*, la función buscará en el intervalo cerrado [*límite Inferior*,*límite superior*] el valor del minimo local en la función.

**nInt()**

**nInt**(*Expr1*, *Var*, *Inferior*, *Superior*)⇒*expresión*

$\text{nInt}(e^{-x^2},x,-1,1)$	1.49365
--------------------------------	---------

Si el integrando *Expr1* no contiene ninguna variable que no sea *Var*, y si *Inferior* y *Superior* son constantes, positiva  $\infty$  o negativa  $\infty$ , entonces **nInt()** entrega una aproximación de  $\int(\text{Expr1}, \text{Var}, \text{Inferior}, \text{Superior})$ . Esta aproximación es un promedio ponderado de algunos valores muestra del integrando en el intervalo  $\text{Inferior} < \text{Var} < \text{Superior}$ .

**nInt()**Catálogo > 

La meta es seis dígitos significativos. El logaritmo adaptable termina cuando parece probable que la meta se ha alcanzado, o bien cuando parece improbable que las muestras adicionales producirán una mejora importante.

Se desplegará una advertencia ("Exactitud cuestionable") cuando parece que la meta no se ha alcanzado.

Anide **nInt()** para hacer una integración numérica múltiple. Los límites de la integración pueden depender de las variables de integración afuera de los mismos.

$$\text{nInt}(\cos(x), x, \pi, \pi + 1.E-12) \quad -1.04144E-12$$

$$\text{nInt}\left(\text{nInt}\left(\frac{e^{-x \cdot y}}{\sqrt{x^2 - y^2}}, y, -x, x\right), x, 0, 1\right) \quad 3.30423$$

**nom()**Catálogo > 

**nom(tasaEfectiva, CpA) ⇒ valor**

Función financiera que convierte la tasa de interés efectiva anual *tasaEfectiva* en una tasa nominal, con *CpA* dado como el número de periodos compuestos por año.

*tasaEfectiva* debe ser un número real y *CpA* debe ser un número real > 0.

**Nota:** Vea también **eff()**, página 47.

$$\text{nom}(5.90398, 12) \quad 5.75$$

**nor**teclas  

*BooleanoExpr1* **nor** *BooleanoExpr2*  
devuelve *expresión booleana*

*BooleanaLista1* **nor** *BooleanaLista2*  
devuelve *lista booleana*

*BooleanaMatriz1* **nor** *BooleanaMatriz2*  
devuelve *matriz booleana*

Devuelve la negación de una operación **or** lógica en los dos argumentos. Devuelve verdadero, falso o una forma simplificada de la ecuación.

Para listas y matrices, devuelve comparaciones elemento por elemento.

*Entero1* nor *Entero2* ⇒ *entero*

Compara dos números reales enteros bit a bit utilizando una operación **nor**. Internamente, ambos números enteros se convierten en números binarios de 64 bit y con signos. Cuando se comparan bits correspondientes, el resultado es 1 si ambos bits son 1; de lo contrario el resultado es 0. El valor devuelto representa los resultados bit, y se muestran según el modelo Base.

Puede ingresar los números enteros en cualquier base numérica. Para una entrada binaria o hexadecimal, debe utilizar el prefijo 0b o 0h respectivamente. Sin un prefijo, se trata a los números enteros como decimales (base 10).

3 or 4	7
3 nor 4	-8
{1,2,3} or {3,2,1}	{3,2,3}
{1,2,3} nor {3,2,1}	{-4,-3,-4}

## norm()

Catálogo > 

**norm**(*Matriz*) ⇒ *expresión*

**norm**(*Vector*) ⇒ *expresión*

Entrega la norma Frobenius.

$\text{norm}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$	5.47723
$\text{norm}\left(\begin{bmatrix} 1 & 2 \end{bmatrix}\right)$	2.23607
$\text{norm}\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$	2.23607

## normCdf() (CdfNormal)

Catálogo > 

**normCdf**(*límiteInferior*, *límiteSuperior*,  $\mu$ ,  $[\sigma]$ ) ⇒ *número* si *límiteInferior* y *límiteSuperior* son números, *lista* si *límiteInferior* y *límiteSuperior* son listas

Resuelve la probabilidad de distribución normal entre *límiteInferior* y *límiteSuperior* para  $\mu$  (predeterminado=0) y  $\sigma$  (predeterminado=1) especificados.

Para  $P(X \leq \text{límiteSuperior})$ , configure *límiteInferior* = -9E999.

**normPdf(ValX[,μ[,σ]])** ⇒ número si *ValX* es un número, *lista* si *ValX* es una lista

Resuelve la función de densidad de probabilidad para la distribución normal en un valor *ValX* especificado para  $\mu$  y  $\sigma$  especificados.

**not**

**not** *Booleana* ⇒ *expresión Booleana*

Entrega verdadero, falso o una forma simplificada del argumento.

**not** *Entero I* ⇒ *entero*

Entrega el complemento de uno de un entero real. En forma interna, *Entero I* se convierte en un número binario de 64 bits signado. El valor de cada bit se invierte (0 se convierte en 1, y viceversa) para el complemento de uno. Los resultados se despliegan de acuerdo con el modo de la Base.

Usted puede ingresar el entero en cualquier base de números. Para un ingreso binario o hexadecimal, se debe usar el prefijo 0b ó 0h, respectivamente. Sin un prefijo, el entero se trata como decimal (base 10).

Si se ingresa un entero decimal que es demasiado grande para una forma binaria de 64 bits firmada, se usa una operación de módulo simétrico para llevar el valor al rango apropiado. Para obtener más información, vea **►Base2**, página 17.

not (2≥3)	true
not 0hB0►Base16	0hFFFFFFFFFFFFFF4F
not not 2	2

En modo de base hexadecimal:

**Importante:** Cero, no la letra O.

not 0h7AC36	0hFFFFFFFFFFFF853C9
-------------	---------------------

En modo de base binaria:

0b100101►Base10	37
not 0b100101	0b11111111111111111111111111111111►
not 0b100101►Base10	-38

Para ver el resultado completo, presione **▲** y después use **◀** y **▶** para mover el cursor.

**Nota:** Un ingreso binario puede tener hasta 64 dígitos (sin contar el prefijo 0b). Un ingreso hexadecimal puede tener hasta 16 dígitos.

**nPr() (prN)**

Catálogo &gt;

**nPr(Valor1, Valor2)⇒expresión**

Para entero *Valor1* y *Valor2* con  $Valor1 \geq Valor2 \geq 0$ , **nPr()** es el número de permutaciones de los elementos del *Valor1* tomadas del *Valor2* a la vez.

$nPr(z,3) z=5$	60
$nPr(z,3) z=6$	120
$nPr(\{5,4,3\},\{2,4,2\})$	{20,24,6}
$nPr\left(\begin{bmatrix} 6 & 5 \\ 4 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}\right)$	$\begin{bmatrix} 30 & 20 \\ 12 & 6 \end{bmatrix}$

**nPr(Valor, 0)⇒1****nPr(Valor, enteroNeg)⇒ 1/((Valor+1) · (Valor+2)... (Valor-enteroNeg))****nPr(Valor, enteroPos)⇒ Valor · (Valor-1)... (Valor-enteroPos+1)****nPr(Valor, noEntero)⇒Valor! / (Valor-noEntero)!****nPr(Lista1, Lista2)⇒lista**

Entrega una lista de permutaciones con base en los pares de elementos correspondientes en las dos listas. Los argumentos deben tener el mismo tamaño que la lista.

$nPr(\{5,4,3\},\{2,4,2\})$	{20,24,6}
----------------------------	-----------

**nPr(Matriz1, Matriz2)⇒matriz**

Entrega una matriz de permutaciones con base en los pares de elementos correspondientes en las dos matrices. Los argumentos deben tener el mismo tamaño que la matriz.

$nPr\left(\begin{bmatrix} 6 & 5 \\ 4 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}\right)$	$\begin{bmatrix} 30 & 20 \\ 12 & 6 \end{bmatrix}$
--	---

**npv() (vpn)**

Catálogo &gt;

**npv(TasaInterés,FEO,ListaFE [,FrecFE])**

Función financiera que calcula el valor presente neto; la suma de los valores presentes para las entradas y salidas de efectivo. Un resultado positivo para el vpn indica una inversión rentable.

*tasaInterés* es la tasa por la que se descuentan los flujos de efectivo (el costo del dinero) durante un periodo.

*FEO* es el flujo de efectivo inicial en tiempo 0; debe ser un número real.

$list1:=\{6000,-8000,2000,-3000\}$	{6000,-8000,2000,-3000}
$list2:=\{2,2,2,1\}$	{2,2,2,1}
$npv(10,5000,list1,list2)$	4769.91

*ListaFE* es una lista de cantidades de flujo de efectivo después del flujo de efectivo inicial *FE0*.

*FrecFE* es una lista en la cual cada elemento especifica la frecuencia de ocurrencia para una cantidad de flujo de efectivo (consecutivo) agrupado, que es el elemento correspondiente de la *ListaFE*. La predeterminada es 1; si usted ingresa valores, éstos deben ser enteros positivos < 10,000.

**nSolve() (solucionN)**

**nSolve**(*Ecuación*,*Var*  
[=*Cálculo*]) ⇒ número de error\_cadena

**nSolve**(*Ecuación*,*Var*  
[=*Cálculo*],*límiteInferior*) ⇒ número de error\_cadena

**nSolve**(*Ecuación*,*Var*  
[=*Cálculo*],*límiteInferior*,*límiteSuperior*)  
⇒ número de error\_cadena

**nSolve**(*Ecuación*,*Var*[=*Cálculo*]) |  
*límiteInferior* ≤ *Var* ≤ *límiteSuperior*  
⇒ número de error\_cadena

Busca iterativamente una solución numérica real aproximada para *Ecuación* para su variable uno. Especifique la variable como:

*variable*

– 0 –

*variable* = número real

Por ejemplo, x es válida y también lo es x=3.

$\text{nSolve}(x^2 + 5 \cdot x - 25 = 9, x)$	3.84429
$\text{nSolve}(x^2 = 4, x = -1)$	-2.
$\text{nSolve}(x^2 = 4, x = 1)$	2.

**Nota:** Si hay varias soluciones, usted puede usar un cálculo para ayudar a encontrar una solución particular.

**nSolve()** intenta determinar un punto donde la residual es cero o dos puntos relativamente cercanos donde la residual tiene signos opuestos y la magnitud de la residual no es excesiva. Si no puede lograr esto al usar un número modesto de puntos de muestra, entrega la cadena "ninguna solución encontrada".

$\text{nSolve}(x^2+5\cdot x-25=9,x) x<0$	-8.84429
$\text{nSolve}\left(\frac{(1+r)^{24}-1}{r}=26,r\right) r>0 \text{ and } r<0.25$	0.006886
$\text{nSolve}(x^2=-1,x)$	"No solution found"

## O

**OneVar** [1,]X[,][Frec][,Categoría,Incluir]]

**OneVar** [n,]X1,X2[X3[,...[,X20]]]

Calcula estadísticas de 1 variable en hasta 20 listas. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

*Frec* es una lista opcional de valores de frecuencia. Cada elemento en *Frec* especifica la frecuencia de la ocurrencia para cada punto de datos *X* y *Y* correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

*Categoría* es una lista de códigos de categoría numérica para los valores *X* correspondientes.


*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

Un elemento (inválido) vacío en cualquiera de las listas *X*, *Frec* o *Categoría* da como resultado un inválido para el elemento correspondiente de todas esas listas. Un elemento vacío en cualquiera de las listas *X1* a *X20* da como resultado vacío para el elemento correspondiente de todas esas listas. Para obtener más información sobre elementos vacíos, vea página 231.



Variable de salida	Descripción
stat. $\bar{x}$	Media de valores x
stat. $\Sigma x$	Suma de valores x
stat. $\Sigma x^2$	Suma de valores $x^2$
stat.ex	Desviación estándar muestra de x
stat. $\sigma x$	Desviación estándar de población de x
stat.n	Número de puntos de datos
stat.MinX	Mínimo de valores x
stat.C <sub>1</sub> X	1er Cuartil de x
stat.MedianaX	Mediana de x
stat.C <sub>3</sub> X	3er Cuartil de x
stat.MaxX	Máximo de valores x
stat.SCX	Suma de cuadrados de desviaciones de la media de x

**or**

Catálogo > 

*BooleanaExpr1* **or** *BooleanaExpr2*  
devuelve *expresión booleana*

*BooleanaLista1* **or** *BooleanaLista2*  
devuelve *lista booleana*

*BooleanaMatriz1* **or** *BooleanaMatriz2*  
devuelve *matriz booleana*

Entrega verdadero o falso o una forma simplificada del ingreso original.

Entrega verdadero si cualquiera de las expresiones o ambas se simplifican a verdadero. Entrega falso si ambas expresiones se evalúan a falso.

**Nota:** Vea **xor**.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

*Entero1* **or** *Entero2*  $\Rightarrow$  *entero*

Define  $g(x)=$ Func

Done

If  $x \leq 0$  or  $x \geq 5$

Goto end

Return  $x \cdot 3$

Lbl end

EndFunc

$g(3)$  9

$g(0)$  A function did not return a value

En modo de base hexadecimal:

Compara dos enteros reales bit por bit usando una or operación. En forma interna, ambos enteros se convierten en números binarios de 64 bits firmados. Cuando se comparan los bits correspondientes, el resultado es 1 si cualquiera de los bits es 1; el resultado es 0 sólo si ambos bits son 0. El valor producido representa los resultados de los bits, y se despliega de acuerdo con el modo de Base.

Se pueden ingresar enteros en cualquier base de números. Para un ingreso binario o hexadecimal, se debe usar el prefijo 0b or 0h, respectivamente. Sin un prefijo, los enteros se tratan como decimales (base 10).

Si se ingresa un entero decimal que es demasiado grande para una forma binaria de 64 bits firmada, se usa una operación de módulo simétrico para llevar el valor al rango apropiado. Para obtener más información, vea **Base2**, página 17.

**Nota:** Vea **xor**.

0h7AC36 or 0h3D5F	0h7BD7F
-------------------	---------

**Importante:** Cero, no la letra O.

En modo de base binaria:

0b100101 or 0b100	0b100101
-------------------	----------

**Nota:** Un ingreso binario puede tener hasta 64 dígitos (sin contar el prefijo 0b). Un ingreso hexadecimal puede tener hasta 16 dígitos.

## ord()

**ord**(*Cadena*) ⇒ *entero*

**ord**(*Lista1*) ⇒ *lista*

Entrega el código numérico del primer carácter en la cadena de caracteres *Cadena*, o bien una lista de los primeros caracteres de cada elemento de la lista.

ord("hello")	104
char(104)	"h"
ord(char(24))	24
ord({"alpha", "beta"})	{97,98}

## P


### P►Rx()

**P►Rx**(*rExpr*, *θExpr*) ⇒ *expresión*

En modo de ángulo en Radianes:

**P►Rx**(*rLista*, *θLista*) ⇒ *lista*

**P►Rx**(*rMatriz*, *θMatriz*) ⇒ *matriz*


**P►Rx()**Catálogo > 

Entrega la coordenada x equivalente del par  $(r, \theta)$ .

**Nota:** El argumento  $\theta$  se interpreta como un ángulo en grados, gradianes o radianes, de acuerdo con el modo de ángulo actual. Si el argumento es una expresión, usted puede usar  $^\circ$ ,  $G$  o  $r$  para anular la configuración del modo de ángulo en forma temporal.

**Nota:** Usted puede insertar esta función desde el teclado de la computadora al escribir **P@>Rx (...)**.

$P►Rx(4,60^\circ)$	2.
$P►Rx\left(\{-3,10,1.3\},\left\{\frac{\pi}{3},\frac{\pi}{4},0\right\}\right)$	$\{-1.5,7.07107,1.3\}$

**P►Ry()**Catálogo > 

**P►Ry(rValor,  $\theta$ Valor)⇒valor**

**P►Ry(rLista,  $\theta$ Lista)⇒lista**

**P►Ry(rMatriz,  $\theta$ Matriz)⇒matriz**


Entrega la coordenada y equivalente del par  $(r, \theta)$ .

**Nota:** El argumento  $\theta$  se interpreta como un ángulo en grados, radianes o gradianes, de acuerdo con el modo de ángulo actual.

**Nota:** Usted puede insertar esta función desde el teclado de la computadora al escribir **P@>Ry (...)**.

En modo de ángulo en Radianes:

$P►Ry(4,60^\circ)$	3.4641
$P►Ry\left(\{-3,10,1.3\},\left\{\frac{\pi}{3},\frac{\pi}{4},0\right\}\right)$	$\{-2.59808,-7.07107,0\}$

**PassErr (PasarErr)**Catálogo > **PassErr**

Pasa un error al siguiente nivel.

Si la variable de sistema *códigoErr* es cero, **PassErr** no hace nada.

Para ver un ejemplo de **PasarErr**, vea el Ejemplo 2 bajo el comando **Intentar**, página 172.

La cláusula **Else** del bloque **Try...Else...EndTry** debe usar **ClrErr** o **PassErr**. Si el error se debe procesar o ignorar, use **ClrErr**. Si no se sabe qué hacer con el error, use **PassErr** para enviarlo al siguiente manipulador de errores. Si no hay ningún otro manipulador de errores **Try...Else...EndTry** pendiente, el cuadro de diálogo de error se desplegará como normal.

**Nota:** Ver también **BorrarErr**, página 24 e **intento**, page página 172.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

## piecewise() (compuestoDeVariables)

**piecewise**(*Expr1* [, *Cond1* [, *Expr2* [, *Cond2* [, ... ]]])

Define  $p(x) = \begin{cases} x, & x > 0 \\ \text{undef}, & x \leq 0 \end{cases}$  *Done*

Entrega definiciones para una función de compuesto de variables en la forma de una lista. Usted también puede crear definiciones de compuesto de variables al usar una plantilla.

$p(1)$	1
$p(-1)$	undef

**Nota:** Vea también **Plantilla de compuesto de variables**, página 3.

## poissCdf()

**poissCdf**

( $\lambda$ , *límiteInferior*, *límiteSuperior*)  $\Rightarrow$  número si *límiteInferior* y *límiteSuperior* son números, lista si *límiteInferior* y *límiteSuperior* son listas

**poissCdf**( $\lambda$ , *límiteSuperior*) para  $P(0 \leq X \leq \text{límiteSuperior}) \Rightarrow$  número si *límiteSuperior* es un número, lista si *límiteSuperior* es una lista

Resuelve una probabilidad acumulativa para la distribución de Poisson discreta con una media especificada  $\lambda$ .

Para  $P(X \leq \text{límiteSuperior})$ , configure  $\text{límiteInferior}=0$

$\text{poissPdf}(\lambda, \text{ValX}) \Rightarrow \text{número}$  si  $\text{ValX}$  es un número, *lista* si  $\text{ValX}$  es una lista

Resuelve una probabilidad para la distribución de Poisson discreta con la media especificada  $\lambda$ .

*Vector* ►Polar

[1 3.]►Polar [3.16228 71.5651]

**Nota:** Usted puede insertar este operador desde el teclado de la computadora al escribir `@>Polar`.

Despliega el *vector* en forma polar [ $r \angle \theta$ ]. El vector debe ser de dimensión 2 y puede ser una fila o una columna.

**Nota:** ►Polar es una instrucción de formato de despliegue, no una función de conversión. Usted puede usarla sólo al final de una línea de ingreso, y no actualiza *ans*.

**Nota:** Vea también ►Rect, página 133.

*valorComplejo* ►Polar

Despliega el *vectorComplejo* en forma polar.

- El modo de ángulo en grados entrega ( $r \angle \theta$ ).
- El modo de ángulo en radianes entrega  $re^{i\theta}$ .

*valorComplejo* puede tener cualquier forma compleja. Sin embargo, un ingreso de  $re^{i\theta}$  causa un error en el modo de ángulo en grados.

En modo de ángulo en Radianes:

$(3+4i)$ ►Polar	$e^{0.927295 \cdot i \cdot 5}$
$\left(4 \angle \frac{\pi}{3}\right)$ ►Polar	$e^{1.0472 \cdot i \cdot 4}$

En modo de ángulo en Gradianes:

$(4i)$ ►Polar ( $4 \angle 100.$ )

**►Polar**

Catálogo &gt;

**Nota:** Usted debe usar los paréntesis para un ingreso polar ( $r \angle \theta$ ).

En modo de ángulo en Grados:

$(3+4i) \blacktriangleright$ Polar	$(5 \angle 53.1301)$
------------------------------------	----------------------

**polyEval() (evalPoli)**

Catálogo &gt;

**polyEval(Lista1, Expr1)**  $\Rightarrow$  expresión

$\text{polyEval}(\{1,2,3,4\},2)$	26
----------------------------------	----

**polyEval(Lista1, Lista2)**  $\Rightarrow$  expresión

$\text{polyEval}(\{1,2,3,4\},\{2,-7\})$	$\{26,-262\}$
---	---------------

Interpreta el primer argumento como el coeficiente de un polinomio de grado descendente, y entrega el polinomio evaluado para el valor del segundo argumento.

**polyRoots() (raícesPoli)**

Catálogo &gt;

**polyRoots(Poli,Var)**  $\Rightarrow$  lista

$\text{polyRoots}(y^3+1,y)$	$\{-1\}$
-----------------------------	----------

**polyRoots(ListaDeCoefs)**  $\Rightarrow$  lista

$\text{cPolyRoots}(y^3+1,y)$	$\{-1,0.5-0.866025i,0.5+0.866025i\}$
------------------------------	--------------------------------------

La primera sintaxis, **polyRoots(Poli,Var)**, entrega una lista de raíces reales del polinomio *Poli* con respecto de la variable *Var*. Si no existe ninguna raíz real, entrega una lista vacía:  $\{ \}$ .

$\text{polyRoots}(x^2+2 \cdot x+1,x)$	$\{-1,-1\}$
---------------------------------------	-------------

*Poli* debe ser un polinomio en forma expandida en una variable. No use formas expandidas como  $y^2 \cdot y+1$  ó  $x \cdot x+2 \cdot x+1$

$\text{polyRoots}(\{1,2,1\})$	$\{-1,-1\}$
-------------------------------	-------------

La segunda sintaxis, **polyRoots(ListaDeCoefs)**, entrega una lista de raíces reales para los coeficientes en *ListadeCoefs*.

**Nota:** Vea también **cPolyRoots()**, página 32.

**PowerReg (RegPot)**

Catálogo &gt;

**PowerReg X,Y [, Frec] [, Categoría, Incluir]**

Resuelve la regresión de potencia  $y = a \cdot (x)^b$  en listas  $X$  y  $Y$  con frecuencia  $Frec$ . Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

$X$  y  $Y$  son listas de variables independientes y dependientes.

$Frec$  es una lista opcional de valores de frecuencia. Cada elemento en  $Frec$  especifica la frecuencia de la ocurrencia para cada punto de datos  $X$  y  $Y$  correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

$Categoría$  es una lista de códigos de categoría numérica o de cadena para los datos  $X$  y  $Y$  correspondientes.


*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $a \cdot (x)^b$
stat.a, stat.b	Coefficientes de regresión
stat.r <sup>2</sup>	Coefficiente de determinación lineal para datos transformados
stat.r	Coefficiente de correlación para datos transformados ( $\ln(x)$ , $\ln(y)$ )
stat.Resid	Residuales asociados con el modelo de potencia
stat.TransResid	Residuales asociadas con el ajuste lineal de datos transformados
stat.XReg	La lista de puntos de datos en la <i>Lista X</i> modificada que se usa en realidad en la regresión con base en las restricciones de las <i>Categorías Frec</i> , <i>Lista de Categorías Incluir</i>

Variable de salida	Descripción
stat.YReg	La lista de puntos de datos en la <i>Lista Y</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoríae Incluir</i>
stat.FrecReg	Lista de frecuencias correspondientes a <i>stat.XReg</i> y <i>stat.YReg</i>

## Prgm

Catálogo > 

### Prgm

*Bloque*

### EndPrgm

Plantilla para crear un programa definido por el usuario. Se debe usar con el comando **Define**, **Define LibPub**, o **Define LibPriv**.

*Bloque* puede ser una sentencia sencilla, una serie de sentencias separadas con el caracter ";" o una serie de sentencias en líneas separadas.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

Calcular MCD y desplegar los resultados intermedios.

---

```
Define proggcd(a,b)=Prgm
  Local d
  While b≠0
    d:=mod(a,b)
    a:=b
    b:=d
  Disp a," ",b
  EndWhile
  Disp "GCD=",a
  EndPrgm
  Done
```

---

```
proggcd(4560,450)
-----
450 60
60 30
30 0
GCD=30
-----
Done
```

---


## prodSeq()

Vea  $\Pi()$ , página 204.

## Product (PI)

Vea  $\Pi()$ , página 204.



**product()**Catálogo > **product**(*Lista*[, *Iniciar*[, *Terminar*]]) $\Rightarrow$ *expresión*

Entrega el producto de los elementos contenidos en *Lista*. *Inicio* y *Término* son opcionales. Especifican un rango de elementos.

$\text{product}(\{1,2,3,4\})$	24
$\text{product}(\{4,5,8,9\},2,3)$	40

**product**(*Matriz1*[, *Iniciar*[, *Terminar*]]) $\Rightarrow$ *matriz*

Entrega un vector de fila que contiene los productos de los elementos en las columnas de *Matriz1*. *Inicio* y *término* son opcionales. Especifican un rango de filas.

$\text{product}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}\right)$	[28 80 162]
$\text{product}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},1,2\right)$	[4 10 18]

Los elementos vacíos (inválidos) se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

**propFrac()**Catálogo > **propFrac**(*Valor1*[, *Var*]) $\Rightarrow$ *valor*

**propFrac**(*número\_racional*) entrega *número\_racional* como la suma de un entero y una fracción que tiene el mismo signo y una magnitud de denominador mayor que la magnitud del numerador.

$\text{propFrac}\left(\frac{4}{3}\right)$	$1 + \frac{1}{3}$
$\text{propFrac}\left(\frac{-4}{3}\right)$	$-1 - \frac{1}{3}$

**propFrac**(*expresión\_racional*,*Var*) entrega la suma de las proporciones apropiadas y un polinomio con respecto de *Var*. El grado de *Var* en el denominador excede el grado de *Var* en el numerador en cada proporción apropiada. Se recopilan potencias similares de *Var*. Los términos y sus factores se ordenan con *Var* como la variable principal.

Si se omite *Var*, se realiza una expansión de la fracción apropiada con respecto de la variable más principal. Entonces los coeficientes de la parte polinómica se tornan apropiados con respecto de su variable más principal primero y así sucesivamente.

**propFrac()**

Catálogo &gt;

Usted puede usar la función **propFrac()** para representar fracciones mezcladas y demostrar la suma y la resta de fracciones mezcladas.

$\text{propFrac}\left(\frac{11}{7}\right)$	$1+\frac{4}{7}$
$\text{propFrac}\left(3+\frac{1}{11}+\frac{3}{4}\right)$	$8+\frac{37}{44}$
$\text{propFrac}\left(3+\frac{1}{11}-\left(5+\frac{3}{4}\right)\right)$	$-2-\frac{29}{44}$

**Q****QR**

Catálogo &gt;

**QR** *Matriz, matrizQ, matrizR[, Tol]*

Calcula la factorización de QR de Householder de una matriz real o una matriz compleja. Las matrices Q y R resultantes se almacenan en la *Matriz* especificada. La matriz Q es unitaria. La matriz R es triangular superior.

De manera opcional, cualquier elemento de matriz se trata como cero si su valor absoluto es menor que la *Tolerancia*. Esta tolerancia se usa sólo si la matriz tiene ingresos de punto flotante y no contiene ninguna variable simbólica a la que no se le haya asignado un valor. De otro modo, la *Tolerancia* se ignora.

- Si usted usa o configura el modo **Auto** o **Aproximado** para aproximar, los cálculos se realizan al usar la aritmética de punto flotante.
- Si la *Tolerancia* se omite o no se usa, la tolerancia predeterminada se calcula como:  
 $5E-14 \cdot \text{máx}(\text{dim}(\text{Matriz})) \cdot \text{normaFila}(\text{Matriz})$

El número de punto flotante (9.) en m1 causa que los resultados se calculen en forma de punto flotante.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9. \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9. \end{bmatrix}$
QR m1,qm,rm	Done
qm	$\begin{bmatrix} 0.123091 & 0.904534 & 0.408248 \\ 0.492366 & 0.301511 & -0.816497 \\ 0.86164 & -0.301511 & 0.408248 \end{bmatrix}$
rm	$\begin{bmatrix} 8.12404 & 9.60114 & 11.0782 \\ 0. & 0.904534 & 1.80907 \\ 0. & 0. & 0. \end{bmatrix}$

La factorización de QR se resuelve numéricamente al usar transformaciones de Householder. La solución simbólica se resuelve al usar Gram-Schmidt. Las columnas en *nombreMatQ* son los vectores de base ortonormal que extienden el espacio definido por la *matriz*.

## QuadReg (RegCuad)

**QuadReg**  $X, Y$  [, *Frec*] [, *Categoría*, *Incluir*]]

Resuelve la regresión polinómica cuadrática  $= a \cdot x^2 + b \cdot x + c$  en las listas  $X$  y  $Y$  con frecuencia *Frec*. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

$X$  y  $Y$  son listas de variables independientes y dependientes.

*Frec* es una lista opcional de valores de frecuencia. Cada elemento en *Frec* especifica la frecuencia de la ocurrencia para cada punto de datos  $X$  y  $Y$  correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

*Categoría* es una lista de códigos de categoría numérica o de cadena para los datos  $X$  y  $Y$  correspondientes.

*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $a \cdot x^2 + b \cdot x + c$

stat.a, stat.b, stat.c	Coefficientes de regresión
stat.R <sup>2</sup>	Coefficiente de determinación
stat.Resid	Residuales de la regresión
stat.XReg	La lista de puntos de datos en la <i>Lista X</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoríaae Incluir</i>
stat.YReg	La lista de puntos de datos en la <i>Lista Y</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoríaae Incluir</i>
stat.FrecReg	Lista de frecuencias correspondientes a <i>stat.XReg</i> y <i>stat.YReg</i>

## QuartReg (RegCuart)

Catálogo > 

**QuartReg** *X,Y* [, *Frec*] [, *Categoría*, *Incluir*]

Resuelve la regresión polinómica cuártica  $y = a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$  en las listas *X* y *Y* con frecuencia *Frec*. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

*X* y *Y* son listas de variables independientes y dependientes.

*Frec* es una lista opcional de valores de frecuencia. Cada elemento en *Frec* especifica la frecuencia de la ocurrencia para cada punto de datos *X* y *Y* correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

*Categoría* es una lista de códigos de categoría numérica o de cadena para los datos *X* y *Y* correspondientes.


*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de regresión: $a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$
stat.a, stat.b, stat.c, stat.d, stat.e	Coefficientes de regresión
stat.R <sup>2</sup>	Coefficiente de determinación
stat.Resid	Residuales de la regresión
stat.XReg	La lista de puntos de datos en la <i>Lista X</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoríae Incluir</i>
stat.YReg	La lista de puntos de datos en la <i>Lista Y</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoríae Incluir</i>
stat.FrecReg	Lista de frecuencias correspondientes a <i>stat.XReg</i> y <i>stat.YReg</i>

## R

### R ► Pθ()

Catálogo > 

R ► Pθ (*xValue*, *yValue*) ⇒ *valor*

R ► Pθ (*xList*, *yList*) ⇒ *lista*

R ► Pθ (*xMatrix*, *yMatrix*) ⇒ *matriz*

Produce la coordenada θ equivalente de los argumentos pares (*x*,*y*).

**Nota:** El resultado se obtiene como un grado, gradián, o ángulo radián, de acuerdo con la configuración actual del modo del ángulo.

**Nota:** Puede insertar esta función con el teclado de la computadora escribiendo **R@>Ptheta (...)**.

En modo de ángulo en grados:

R ► Pθ(2,2) 45.

En modo de ángulo en gradianes:


R ► Pθ(2,2) 50.

En modo de ángulo en radianes:

R ► Pθ(3,2) 0.588003

R ► Pθ  $\left( \begin{bmatrix} 3 & -4 & 2 \end{bmatrix}, \begin{bmatrix} 0 & \frac{\pi}{4} & 1.5 \end{bmatrix} \right)$   
 [0. 2.94771 0.643501]

### R ► Pr()

Catálogo > 

R ► Pr (*xValue*, *yValue*) ⇒ *valor*

R ► Pr (*xList*, *yList*) ⇒ *lista*

R ► Pr (*xMatrix*, *yMatrix*) ⇒ *matriz*

En modo de ángulo en radianes:

**R►Pr()**

Catálogo &gt;

Produce la coordenada-r equivalente de los argumentos pares  $(x,y)$ .

**Nota:** Puede insertar esta función con el teclado de la computadora escribiendo **R@>Pr (...)**.

R►Pr(3,2)	3.60555
R►Pr( $\left[ \begin{matrix} 3 & -4 & 2 \\ 0 & \frac{\pi}{4} & 1.5 \end{matrix} \right]$ )	$\begin{bmatrix} 3 & 4.07638 & \frac{5}{2} \end{bmatrix}$

**►Rad**

Catálogo &gt;

*Value*►Rad ⇒ *valor*

Convierte el argumento en una medida en ángulo radián.

**Nota:** Puede insertar esta función con el teclado de la computadora escribiendo **@>Rad**.

En modo de ángulo en grados:

(1.5)►Rad	(0.02618) <sup>r</sup>
-----------	------------------------

En modo de ángulo en gradianes:

(1.5)►Rad	(0.023562) <sup>r</sup>
-----------	-------------------------

**rand()**

Catálogo &gt;

**rand()** ⇒ *expresión*

**rand(#Trials)** ⇒ *lista*

**rand()** entrega un valor aleatorio entre 0 y 1.

**rand(#Trials)** produce una lista que contiene *#Trials* valores aleatorios de entre 0 y 1.

Ajusta la semilla de número aleatorio.

RandSeed 1147	Done
rand(2)	{0.158206,0.717917}

**randBin()**

Catálogo &gt;

**randBin(*n, p*)** ⇒ *expresión*

**randBin(*n, p, #Trials*)** ⇒ *lista*

**randBin(*n, p*)** produce un número aleatorio real de una distribución binomial especificada.

**randBin(*n, p, #Trials*)** produce una lista que contiene *#Trials* números aleatorios reales de una distribución binomial especificada.

randBin(80,0.5)	46.
randBin(80,0.5,3)	{43.,39.,41.}

**randInt()**Catálogo > **randInt**

(  
*lowBound,upBound*)  
 ⇒ expresión

<code>randInt(3,10)</code>	3.
<code>randInt(3,10,4)</code>	{9.,3.,4.,7.}

**randInt**

(*lowBound,upBound*  
 ,#Trials) ⇒ lista

**randInt**

(  
*lowBound,upBound*)  
 produce un entero  
 aleatorio dentro del  
 rango especificado  
 por los límites  
 enteros *lowBound*  
 y *upBound*.

**randInt**

(*lowBound,upBound*  
 ,#Trials) produce  
 una lista que  
 contiene #Trials de  
 enteros aleatorios  
 dentro del rango  
 especificado.

**randMat()**Catálogo > 

**randMat**(*numRows, numColumns*) ⇒  
 matriz

Produce una matriz de enteros de entre -  
 9 y 9 de la dimensión especificada.

Ambos argumentos deben simplificarse a  
 enteros.

RandSeed 1147	Done									
<code>randMat(3,3)</code>	<table border="1"> <tr><td>8</td><td>-3</td><td>6</td></tr> <tr><td>-2</td><td>3</td><td>-6</td></tr> <tr><td>0</td><td>4</td><td>-6</td></tr> </table>	8	-3	6	-2	3	-6	0	4	-6
8	-3	6								
-2	3	-6								
0	4	-6								

**Nota:** Los valores en esta matriz cambiarán  
 cada vez que presione .

**randNorm()**Catálogo > 

**randNorm**( $\mu, \sigma$ ) ⇒ expresión  
**randNorm**( $\mu, \sigma, \#Trials$ ) ⇒ lista

RandSeed 1147	Done
<code>randNorm(0,1)</code>	0.492541
<code>randNorm(3,4.5)</code>	-3.54356

## randNorm()

Catálogo > 

**randNorm**( $\mu$ ,  $\sigma$ ) produce un número decimal de la distribución normal especificada. Este puede ser cualquier número real pero altamente concentrado en el intervalo  $[\mu-3\cdot\sigma, \mu+3\cdot\sigma]$ .

**randNorm**( $\mu$ ,  $\sigma$ , #Trials) produce una lista que contiene #Trials de números decimales de la distribución normal especificada.

## randPoly()

Catálogo > 

**randPoly**(Var, Order)  $\Rightarrow$  expresión

Entrega un polinomio en el Var del Orden especificado. Los coeficientes son enteros aleatorios en el rango de -9 a 9. El coeficiente inicial no será cero.

Orden debe ser 0 a 99.

RandSeed 1147	Done
randPoly(x,5)	$-2\cdot x^5+3\cdot x^4-6\cdot x^3+4\cdot x-6$

## randSamp()

Catálogo > 

**randSamp**(List,#Trials[,noRepl])  $\Rightarrow$  lista

Produce una lista que contiene una muestra aleatoria de #Trials intentos desde la Lista con una opción para reemplazo de muestra (noRepl=0), o no reemplazo de muestra (noRepl=1). El valor predeterminado es con reemplazo de muestra.

Define list3={1,2,3,4,5}	Done
Define list4=randSamp(list3,6)	Done
list4	{1.,3.,3.,1.,3.,1.}

## RandSeed

Catálogo > 

**RandSeed** Número

Si el Número = 0, ajusta las semillas a los valores predeterminados de fábrica para el generador de números aleatorios. Si el Número  $\neq$  0, se usa para generar dos semillas, las cuales se almacenan en las variables del sistema seed1 y seed2.

RandSeed 1147	Done
rand()	0.158206



**real()**

Catálogo &gt;

**real(Value1)** ⇒ *valor*

$$\text{real}(2+3\cdot i) \quad 2$$

Produce la parte real del argumento.

**real(List1)** ⇒ *lista*

$$\text{real}(\{1+3\cdot i, 3, i\}) \quad \{1, 3, 0\}$$

Produce las partes reales de todos los elementos.

**real(Matrix1)** ⇒ *matriz*

$$\text{real}\left(\begin{pmatrix} 1+3\cdot i & 3 \\ 2 & i \end{pmatrix}\right) \quad \begin{bmatrix} 1 & 3 \\ 2 & 0 \end{bmatrix}$$

Produce las partes reales de todos los elementos.

**► Recta**

Catálogo &gt;

*Vector* ► **Recta****Nota:** Puede insertar esta función con el teclado de la computadora escribiendo @>Rect.

$$\left(3 \angle \frac{\pi}{4} \quad \angle \frac{\pi}{6}\right) \blacktriangleright \text{Rect} \\ [1.06066 \quad 1.06066 \quad 2.59808]$$

Muestra el *Vector* en forma rectangular [x, y, z]. El vector debe ser de dimensión 2 o 3 y puede ser una fila o una columna.**Nota:** ► **Recta** es una instrucción de mostrar formato, no una función de conversión. Puede utilizarla solamente al final de la línea de ingreso y no actualiza a *ans*.**Nota:** Consulte también ► **Polar**, página 121.*complexValue* ► **Recta**Muestra *complexValue* en forma rectangular a+bi. *complexValue* puede tener cualquier forma compleja. Sin embargo, una entrada  $re^{i\theta}$  causa un error en el modo de ángulo en grados.**Nota:** Debe usar paréntesis para una entrada polar ( $r \angle \theta$ ).

En modo de ángulo en radianes:

$$\left(4 \cdot e^{\frac{\pi}{3}}\right) \blacktriangleright \text{Rect} \quad 11.3986 \\ \left(4 \angle \frac{\pi}{3}\right) \blacktriangleright \text{Rect} \quad 2.+3.4641 \cdot i$$

En modo de ángulo en gradianes:

$$\left((1 \angle 100)\right) \blacktriangleright \text{Rect} \quad i$$

En modo de ángulo en grados:

$$\left(4 \angle 60\right) \blacktriangleright \text{Rect} \quad 2.+3.4641 \cdot i$$



**Nota:** Para escribir  $\angle$ , seleccione de la lista de símbolos en el catálogo.

## ref()

**ref**(*Matrix1*[, *Tol*]) ⇒ *matriz*

Produce la forma escalonada por filas de *Matrix1*.

Opcionalmente, cualquier elemento de la matriz es tratado como cero si su valor absoluto es menor a *Tol*. Esta tolerancia solamente se utiliza si la matriz tiene entradas de punto flotante y no contiene ninguna variable simbólica a la que no se haya asignado un valor. De otra forma, *Tol* se ignora.

- Si usa   o si ajusta el modo **Auto** o **Aproximado** para que sea Aproximado, los cálculos se hacen usando aritmética de punto flotante.
- Si *Tol* se omite o no se utiliza, la tolerancia predeterminada se calcula como:  
 $5E-14 \cdot \max(\dim(\text{Matrix1})) \cdot \text{rowNorm}(\text{Matrix1})$

Evite los elementos indefinidos en la *Matrix1*. Estos pueden dar lugar a resultados inesperados.

Por ejemplo, si *a* es indefinida en la siguiente expresión, se muestra un mensaje de advertencia y el resultado se muestra como:

$$\text{ref} \left( \begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \quad \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & a & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La advertencia aparece debido a que el elemento generalizado  $1/a$  no sería válido para  $a=0$ .

$$\text{ref} \left( \begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix} \right) \quad \begin{bmatrix} 1 & \frac{-2}{5} & \frac{-4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$$

Puede evitar esto almacenando un valor a *ade* antemano o utilizando el operador restrictivo "|" para sustituir un valor, tal como se muestra en el siguiente ejemplo.


$$\text{ref} \left( \begin{array}{ccc} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right) | a=0 \quad \begin{array}{ccc} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{array}$$

**Nota:** Consulte también `rref()`, page 144.

## RefreshProbeVars

### RefreshProbeVars

Le permite el acceso a los datos del sensor desde todas las sondas de sensor conectadas en su programa TI-Basic.

Valor de StatusVar	Estado
<i>statusVar</i> =0	Normal (continuar con el programa) La aplicación Vernier DataQuest™ se encuentra en el modo de recolección de datos.
<i>statusVar</i> =1	<b>Nota:</b> La aplicación Vernier DataQuest™ debe estar en el modo medidor para que este comando funcione. 
<i>statusVar</i> =2	La aplicación Vernier DataQuest™ no se ha iniciado.
<i>statusVar</i> =3	La aplicación Vernier DataQuest™ se ha iniciado, pero usted no ha conectado ningún sensor.

### Ejemplo

```
Definir temp()=
Prgm
© Verifica si el sistema está
listo
Estado RefreshProbeVars
Si el estado=0 entonces
Disp "listo"
Para n,1,50
Estado RefreshProbeVars
temperatura:=meter.temperature
Disp "Temperatura: ",temperatura
Si la temperatura>30 Entonces
Disp "Muy caliente"
EndIf
© Espere 1 segundo entre
muestras
Espere 1
EndFor
Else
```

Disp "No listo. Intente de nuevo más tarde"

EndIf

Terminar Prgm

Nota: Esto también se puede utilizar con TI-Innovator™ Hub.

## remain()

**remain**(*Value1*, *Value2*) ⇒ *valor*

**remain**(*List1*, *List2*) ⇒ *lista*

**remain**(*Matrix1*, *Matrix2*) ⇒ *matriz*

Produce el residuo del primer argumento con respecto al segundo argumento tal como se define por las identidades:

$\text{remain}(x,0) = x$

$\text{remain}(x,y) = x - y \cdot \text{iPart}(x/y)$

Como consecuencia, note que **remain**(-*x*,*y*) = **remain**(*x*,*y*). El resultado es o bien cero o tiene el mismo signo que el primer argumento.

**Nota:** Consulte también **mod**(), página 103.

remain(7,0)	7
remain(7,3)	1
remain(-7,3)	-1
remain(7,-3)	1
remain(-7,-3)	-1
remain({12,-14,16},{9,7,-5})	{3,0,1}

remain( $\begin{pmatrix} 9 & -7 \\ 6 & 4 \end{pmatrix}, \begin{pmatrix} 4 & 3 \\ 4 & -3 \end{pmatrix}$ )	$\begin{pmatrix} 1 & -1 \\ 2 & 1 \end{pmatrix}$
--	---

## Solicitar

**Solicitar** *promptString*, *var* [, *DispFlag* [, *statusVar*]]

**Solicitar** *promptString*, *func*(*arg1*, ...*argn*) [, *DispFlag* [, *statusVar*]]

Comando de programación: Pausa el programa y muestra un cuadro de diálogo que contiene el mensaje *promptString* y un cuadro de ingreso para respuesta del usuario.

Definir un Programa:

```
Definir request_demo()=Prgm
  Solicitar "Radio: ",r
  Disp "Área = ",pi*r^2
Terminar Prgm
```

Ejecutar el programa e ingresar una respuesta:

```
request_demo()
```

Cuando el usuario ingresa una respuesta y hace clic en **Aceptar** (OK), el contenido del cuadro de ingreso se asigna a la variable *var*.

Si el usuario hace clic en **Cancelar** (Cancel), el programa procede sin aceptar ninguna entrada. El programa usa el valor previo de *var* si *var* ya estaba definido.

El argumento opcional *DispFlag* puede ser cualquier expresión.

- Si *DispFlag* se omite o se evalúa como **1**, el mensaje de pregunta y la respuesta del usuario se muestran en el historial de la calculadora.
- Si *DispFlag* evalúa a **0**, la pregunta y la respuesta no se muestran en el historial.

El argumento opcional *statusVar* le da al programa una manera de determinar cómo el usuario descartó el cuadro de diálogo. Tome en cuenta que *statusVar* requiere el argumento *DispFlag*.

- Si el usuario hizo clic en **OK** o presionó **Intro** o **Ctrl+Intro**, la variable *statusVar* se configura a un valor de **1**.
- De otra manera, la variable *statusVar* se configura a un valor de **0**.

El argumento *func()* le permite a un programa almacenar la respuesta del usuario como una definición de función. La sintaxis opera como si el usuario ejecutara el comando:

Definir *func(arg1, ...argn) = respuesta del usuario*

Entonces el programa puede usar la función *func()* definida. La *promptString* debería guiar al usuario a ingresar una *respuesta de usuario* apropiada que complete la definición de la función.



Resultado después de seleccionar **OK**:

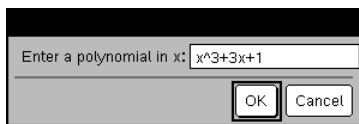
Radio: 6/2  
Área= 28,2743

Definir un Programa:

```
Definir polynomial()=Prgm
  Solicitar "Ingrese un polinomio en
  x:",p(x)
  Disp "Raíces reales son:",polyRoots
  (p(x),x)
Terminar Prgm
```

Ejecutar el programa e ingresar una respuesta:

polynomial()





Resultado después de ingresar  $x^3+3x+1$  y seleccionar **OK**:

Las raíces reales son:  $\{-0.322185\}$

**Nota:** Usted puede utilizar el comando Request dentro de un programa definido por el usuario, pero no dentro de una función.

Para detener un programa que contiene un comando **Request** dentro de un bucle infinito:

- **Dispositivo portátil:** Mantenga presionada la tecla  **on** y presione  varias veces.
- **Windows®:** Mantenga presionada la tecla **F12** y presione **Intro** varias veces.
- **Macintosh®:** Mantenga presionada la tecla **F5** y presione **Intro** varias veces.
- **iPad®:** La aplicación muestra un indicador. Puede seguir esperando o cancelar.

**Nota:** Consulte también **RequestStr**, page 138.


## RequestStr

**RequestStr** *promptString*, *var*[, *DispFlag*]

Comando de programación: Opera de forma idéntica a la primera sintaxis del comando **Solicitar**, excepto que la respuesta del usuario siempre es interpretada como una cadena. En contraste, el comando **Solicitar** interpreta la respuesta como una expresión a menos que el usuario la coloque entre comillas ("").

**Nota:** Puede usar el comando **RequestStr** dentro de un programa definido por el usuario, pero no dentro de una función.

Para detener un programa que contiene un comando **RequestStr** dentro de un bucle infinito:

- **Dispositivo portátil:** Mantenga presionada la tecla  **on** y presione


Definir un Programa:

```
Definir requestStr_demo()=Prgm
  RequestStr "Su nombre: ",name,0
  Disp "La respuesta tiene ",dim
(nombre)," caracteres."
EndPrgm
```

Ejecutar el programa e ingresar una respuesta:

```
requestStr_demo()
```



 varias veces.

- **Windows®:** Mantenga presionada la tecla **F12** y presione **Intro** varias veces.
- **Macintosh®:** Mantenga presionada la tecla **F5** y presione **Intro** varias veces.
- **iPad®:** La aplicación muestra un indicador. Puede seguir esperando o cancelar.

**Nota:** Consulte también **Request**, page 136.

Resultado después de seleccionar **OK** (Tenga en cuenta que el argumento *DispFlag* de **0** omite la pregunta y la respuesta del historial:

```
requestStr_demo()
```

La respuesta tiene 5 caracteres.

## Return

### Return [*Expr*]

Return *Expr* como el resultado de la función. Usar dentro del bloque

**Func...EndFunc**.

**Nota:** Usar **Return** sin un argumento dentro de un bloque **Prgm...EndPrgm** para salir de un programa.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

```
Define factorial (nn)=
Func
Local answer,counter
1 → answer
For counter,1,nn
answer·counter → answer
EndFor
Return answer}
EndFunc
```

```
factorial (3)
```

```
6
```

## right()

**right**(*List1*[, *Num*]) ⇒ *lista*

Produce los elementos *Num* más a la derecha que se incluyen en *List1*.

Si omite *Num*, produce todos los de *List1*.

**right**(*sourceString*[, *Num*]) ⇒ *serie*

Produce los caracteres *Num* que se incluyen en la serie de caracteres *sourceString*.

Si omite *Num*, produce todos los de *sourceString*.

```
right({1,3,-2,4},3)           {3,-2,4}
```

```
right("Hello",2)             "lo"
```

**right**(*Comparación*) ⇒ *expresión*

Produce el lado derecho de una ecuación o desigualdad.

**rk23 ()**

**rk23**(*Expr*, *Var*, *depVar*, {*Var0*, *VarMax*}, *depVar0*, *VarStep* [, *difto1*]) ⇒ *matriz*

**rk23**(*SystemOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *difto1*]) ⇒ *matriz*

**rk23**(*ListOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *difto1*]) ⇒ *matriz*

Use el método de Runge-Kutta para resolver el sistema

$$\frac{d \text{ depVar}}{d \text{ Var}} = \text{Expr}(\text{Var}, \text{depVar})$$

con  $\text{depVar}(\text{Var0}) = \text{depVar0}$  en el intervalo  $[\text{Var0}, \text{VarMax}]$ . Entrega una matriz cuya primera fila define los valores de resultado de *Var* conforme se definen por medio de *VarStep*. La segunda fila define el valor del primer componente de solución a los valores de *Var* correspondientes, y así sucesivamente.

*Expr* es el lado derecho que define la ecuación diferencial ordinaria (EDO).

*SystemOfExpr* es un sistema de lados derechos que define el sistema de EDOs (corresponde al orden de variables dependientes en *ListOfDepVars*).

*ListOfExpr* es una lista de lados derechos que define el sistema de EDOs (corresponde al orden de variables dependientes en *ListOfDepVars*).

*Var* es la variable independiente.

Ecuación diferencial:

$$y' = 0.001 * y * (100 - y) \text{ y } y(0) = 10$$

$$\text{rk23}\left\{0.001 \cdot y \cdot (100 - y), t, y, \{0, 100\}, 10, 1\right\}$$

0.	1.	2.	3.	4.
10.	10.9367	11.9493	13.042	14.2

Para ver el resultado completo, presione ▲ y después use ◀ y ▶ para mover el cursor.

La misma ecuación con *difto1* configurada a 1.E-6

$$\text{rk23}\left\{0.001 \cdot y \cdot (100 - y), t, y, \{0, 100\}, 10, 1, 1.E-6\right\}$$

0.	1.	2.	3.	4.
10.	10.9367	11.9495	13.0423	14.2189

Sistema de ecuaciones:

$$\begin{cases} y1' = -y1 + 0.1 \cdot y1 \cdot y2 \\ y2' = 3 \cdot y2 - y1 \cdot y2 \end{cases}$$

con  $y1(0) = 2$  y  $y2(0) = 5$

$$\text{rk23}\left\{\begin{cases} y1 + 0.1 \cdot y1 \cdot y2 \\ 3 \cdot y2 - y1 \cdot y2 \end{cases}, t, \{y1, y2\}, \{0, 5\}, \{2, 5\}, 1\right\}$$

0.	1.	2.	3.	4.
2.	1.94103	4.78694	3.25253	1.82848
5.	16.8311	12.3133	3.51112	6.27245



*ListOfDepVars* es una lista de variables dependientes.

{*Var0*, *VarMax*} es una lista de dos elementos que le dice a la función que se integre de *Var0* a *VarMax*.

*ListOfDepVars0* es una lista de valores iniciales para variables dependientes.

Si *VarStep* se evalúa a un número distinto de cero:  $\text{signo}(\text{VarStep}) = \text{signo}(\text{VarMax} - \text{Var0})$  y las soluciones se entregan a  $\text{Var0} + i * \text{VarStep}$  para todos  $i=0,1,2,\dots$  de tal manera que  $\text{Var0} + i * \text{VarStep}$  esté en  $[\text{var0}, \text{VarMax}]$  (pudiera no tener un valor de solución en *VarMax*).

Si *VarMax* se evalúa a cero, las soluciones se entregan a los valores *Var* de "Runge-Kutta".

*diftol* es la tolerancia de error (predeterminado a 0.001).

## root()

**root(Value)** ⇒ raíz

**root(Value1, Value2)** ⇒ raíz

$$\sqrt[3]{8}$$

2

**root(Valor)** entrega la raíz cuadrada de *Valor*.

$$\sqrt[3]{3}$$

1.44225

**root(Value1, Value2)** entrega la raíz *Value2* de *Value1*. *Value1* puede ser una constante real o compleja de punto flotante, o una constante racional entera o compleja.

**Nota:** Consulte también **plantilla de rootNth**, página 2.

## rotate()

**rotate(Integer1[, #ofRotations])** ⇒ entero

En modo base binaria:



## rotate()

Catálogo > 

Si *#ofRotations* es positiva, la rotación es a la izquierda. Si *#ofRotations* es negativa, la rotación es a la derecha. El valor predeterminado es  $-1$  (rota un carácter a la derecha).

## round()

Catálogo > 

**round**(*Value1* [, *digitos*])  $\Rightarrow$  *valor*

$\text{round}(1.234567,3)$  1.235

Produce el argumento redondeado al número de dígitos especificado después del punto decimal.

los *digitos* deben ser un entero en el rango de 0 a 12. Si no se incluyen los *digitos*; produce el argumento redondeado a 12 dígitos significativos.

**Nota:** El modo Mostrar dígitos pudiera afectar la forma en que esto se muestra.

**round**(*List1* [, *digits*])  $\Rightarrow$  *lista*

$\text{round}(\{\{\pi, \sqrt{2}, \ln(2)\}, 4\})$   
 $\{3.1416, 1.4142, 0.6931\}$

Produce una lista de los elementos redondeados al número de dígitos especificado.

**round**(*Matrix1* [, *digits*])  $\Rightarrow$  *matriz*

$\text{round}\left(\begin{bmatrix} \ln(5) & \ln(3) \\ \pi & e^1 \end{bmatrix}, 1\right)$   $\begin{bmatrix} 1.6 & 1.1 \\ 3.1 & 2.7 \end{bmatrix}$

Produce una matriz de los elementos redondeados al número de dígitos especificado.

## rowAdd()

Catálogo > 

**rowAdd**(*Matrix1*, *rIndex1*, *rIndex2*)  $\Rightarrow$  *matriz*

$\text{rowAdd}\left(\begin{bmatrix} 3 & 4 \\ -3 & -2 \end{bmatrix}, 1, 2\right)$   $\begin{bmatrix} 3 & 4 \\ 0 & 2 \end{bmatrix}$

Produce una copia de *Matrix1* con el *rIndex2* de filas reemplazado por la suma de las filas *rIndex1* y por *rIndex2*.

**rowDim()**

Catálogo &gt;

**rowDim**(*Matrix*) ⇒ *expresión*Produce el número de filas en *Matrix*.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
<b>rowDim</b> ( <i>m1</i> )	3

**Nota:** Consulte también **colDim()**, página 24.**rowNorm()**

Catálogo &gt;

**rowNorm**(*Matrix*) ⇒ *expresión*Produce el máximo de sumas de los valores absolutos de los elementos en las filas en *Matrix*.

<b>rowNorm</b> $\left(\begin{bmatrix} -5 & 6 & -7 \\ 3 & 4 & 9 \\ 9 & -9 & -7 \end{bmatrix}\right)$	25
---	----

**Nota:** Todos los elementos de la matriz deben simplificarse a números. Consulte también **colNorm()**, página 25.**rowSwap()**

Catálogo &gt;

**rowSwap**(*Matrix1*, *rIndex1*, *rIndex2*) ⇒ *matriz*Produce *Matrix1* con los *rIndex1* y *rIndex2* de las filas intercambiados.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow mat$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
<b>rowSwap</b> ( <i>mat</i> ,1,3)	$\begin{bmatrix} 5 & 6 \\ 3 & 4 \\ 1 & 2 \end{bmatrix}$



**rref()**

Catálogo &gt;

**rref**(*Matrix1*[, *Tol*]) ⇒ *matriz*Produce la forma escalonada reducida por filas de *Matrix1*.

<b>rref</b> $\left(\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}\right)$	$\begin{bmatrix} 1 & 0 & 0 & \frac{66}{71} \\ 0 & 1 & 0 & \frac{147}{71} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$
---	---

Opcionalmente, cualquier elemento de la matriz es tratado como cero si su valor absoluto es menor a *Tol*. Esta tolerancia solamente se utiliza si la matriz tiene entradas de punto flotante y no contiene ninguna variable simbólica a la que no se haya asignado un valor. De otra forma, *Tol* se ignora.

- Si usa   o si ajusta el modo **Auto o Aproximado** para que sea Aproximado, los cálculos se hacen usando aritmética de punto flotante.
- Si *Tol* se omite o no se utiliza, la tolerancia predeterminada se calcula como:  
 $5E-14 \cdot \max(\dim(\text{Matrix } I)) \cdot \text{rowNorm}(\text{Matrix } I)$

**Nota:** Consulte también **ref()**, page 134.

## S

## sec()

 tecla

**sec**(*Valor1*)  $\Rightarrow$  *valor*

En modo de ángulo en Grados:

**sec**(*Lista1*)  $\Rightarrow$  *lista*

$\text{sec}(45)$  1.41421

Entrega la secante de *Valor1* o entrega una lista que contiene las secantes de todos los elementos en *Lista1*.

$\text{sec}\{1,2,3,4\}$  {1.00015,1.00081,1.00244}

**Nota:** El argumento se interpreta como un ángulo en grados, gradianes o radianes, de acuerdo con la configuración del modo del ángulo actual. Se puede usar °, G, o R para anular el modo de ángulo en forma temporal.

sec<sup>-1</sup>() tecla

**sec<sup>-1</sup>**(*Valor1*)  $\Rightarrow$  *valor*

En modo de ángulo en Grados:

**sec<sup>-1</sup>**(*Lista1*)  $\Rightarrow$  *lista*

$\text{sec}^{-1}(1)$  0.

Entrega el ángulo cuya secante es *Valor1* o entrega una lista que contiene las secantes inversas de cada elemento de *Lista1*.

En modo de ángulo en Gradianes:

$\text{sec}^{-1}(\sqrt{2})$  50.

**Nota:** El resultado se entrega como un ángulo en grados, gradianes o radianes, de acuerdo con la configuración del modo del ángulo actual.

En modo de ángulo en Radianes:


## sec<sup>-1</sup>()

 **tecla**

**Nota:** Usted puede insertar esta función desde el teclado al escribir **arcsec (...)** .

sec<sup>-1</sup>({1,2,5})    {0,1.0472,1.36944}

## sech()

**Catálogo >** 

sech(*Valor1*) ⇒ *valor*

sech(3)    0.099328

sech(*Lista1*) ⇒ *lista*

sech({1,2,3,4})  
{0.648054,0.198522,0.036619}

Entrega la secante hiperbólica de *Valor1* o entrega una lista que contiene las secantes hiperbólicas de los elementos de *Lista1* .

## sech<sup>-1</sup>()

**Catálogo >** 

sech<sup>-1</sup>(*Valor1*) ⇒ *valor*

En el modo de ángulo en Radianes y el modo complejo Rectangular:

sech<sup>-1</sup>(*Lista1*) ⇒ *lista*

sech<sup>-1</sup>(1)    0

Entrega la secante hiperbólica inversa de *Valor1* o entrega una lista que contiene las secantes hiperbólicas inversas de cada elemento de *Lista1* .

sech<sup>-1</sup>({1,-2,2.1})  
{0,2.0944·i,8.E-15+1.07448·i}

**Nota:** Usted puede insertar esta función desde el teclado al escribir **arcsech (...)** .

## Send

## Menú del Concentrador

**Send** *exprOrString1* [, *exprOrString2*] ...

Ejemplo: Encienda el elemento azul del LED RGB incorporado durante 0.5 segundos.

Comando de programación: Envía uno o más TI-Innovator™ Hub comandos a un concentrador conectado.

Send "SET COLOR.BLUE ON TIME .5"  
*Done*

*exprOrString* debe ser un comando válido TI-Innovator™ Hub . En general, *exprOrString* contiene un comando "SET ..." para controlar un dispositivo o un comando "READ ..." para solicitar datos.

Ejemplo: Solicite el valor actual del sensor de nivel de luz incorporado del concentrador. Un comando **Get** recupera el valor y lo asigna a *lightval* variable.

Los argumentos se envían al concentrador sucesivamente.

Send "READ BRIGHTNESS"    *Done*  
Get *lightval*    *Done*  
*lightval*    0.347922

## Send

**Nota:** Puede usar el comando **Send** dentro de un programa definido por el usuario pero no dentro de una función.

**Nota:** Consulte además **Get** (página 64), **GetStr** (página 71) y **eval()** (página 51).

## Menú del Concentrador

Ejemplo: Envíe una frecuencia calculada a la bocina incorporada del concentrador. Use la variable especial *iostr.SendAns* para mostrar el comando del concentrador con la expresión evaluada.

$n:=50$	50
$m:=4$	4
Send "SET SOUND eval(m·n)"	Done
<i>iostr.SendAns</i>	"SET SOUND 200"

## seq() (secuen)

**seq**(*Expr*, *Var*, *Bajo*, *Alto*, *Paso*) ⇒ lista

Incrementa *Var* desde *Bajo* hasta *Alto* por un incremento de *Paso*, evalúa *Expr* y entrega los resultados como una lista. Los contenidos originales de *Var* están ahí todavía después de que se completa **seq()**.

El valor predeterminado para *Paso* = 1.

## Catálogo >

$\text{seq}(n^2, n, 1, 6)$	$\{1, 4, 9, 16, 25, 36\}$
$\text{seq}\left(\frac{1}{n}, n, 1, 10, 2\right)$	$\left\{1, \frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{9}\right\}$
$\text{sum}\left(\text{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right)$	$\frac{1968329}{1270080}$

**Nota:** Para forzar un resultado aproximado,

**Dispositivo portátil:** Presione  .

**Windows®:** Presione **Ctrl+Intro**.

**Macintosh®:** Presione **⌘+Intro**.

**iPad®:** Sostenga **Intro** y seleccione .

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right)$	1.54977
--	---------

## seqGen()

**seqGen**(*Expr*, *Var*, *varDep*, {*Var0*, *VarMax*}, [*ListaDeTérminosInic* [, *PasoVar* [, *ValorMax*]]) lista ⇒

Genera una lista de términos para la secuencia  $\text{varDep}(Var)=Expr$  como sigue: Incrementa la variable independiente *Var* desde *Var0* hasta *VarMax* por medio de *PasoVar*, evalúa  $\text{varDep}(Var)$  para los valores correspondientes de *Var* usando la fórmula *Expr* y *ListaDeTérminosInic*, y entrega los resultados como una lista.

## Catálogo >

Genera los 5 primeros términos de la secuencia  $u(n) = u(n-1)^2/2$ , con  $u(1)=2$  y *PasoVar*=1.

$\text{seqGen}\left(\frac{(u(n-1))^2}{n}, n, u, \{1, 5\}, \{2\}\right)$	$\left\{2, 2, \frac{4}{3}, \frac{4}{9}, \frac{16}{405}\right\}$
---	---

Ejemplo en el que *Var0*=2:

**seqGen**(ListaOSistemaDeExpr, Var, ListaDeVarsDep, {Var0, VarMax}, [, MatrizDeTérminosInic [, PasoVar [, ValorMax]]) matriz  $\Rightarrow$

Genera una matriz de términos para un sistema (o una lista) de secuencias *ListaDeVarsDep* (*Var*)=*ListaOSistemaDeExpr* como sigue: Incrementa la variable independiente *Var* desde *Var0* hasta *VarMax* por medio de *PasoVar*, evalúa *ListaDeVarsDep*(*Var*) para los valores correspondientes de *Var* usando la fórmula *ListaOSistemaDeExpr* y *MatrizDeTérminosInic*, y entrega los resultados como una matriz.

Los contenidos originales de *Var* no cambian después de que se completa **seqGen()**.

El valor predeterminado para *PasoVar* = 1.

$$\text{seqGen}\left(\frac{u(n-1)+1}{n}, n, u, \{2,5\}, \{3\}\right) \\ \left\{3, \frac{4}{3}, \frac{7}{12}, \frac{19}{60}\right\}$$

Sistema de dos secuencias:

$$\text{seqGen}\left(\left\{\frac{1}{n}, \frac{u_1(n-1)}{2} + u_1(n-1)\right\}, n, \{u_1, u_2\}, \{1,5\}, \left[\begin{array}{c} \_ \\ 2 \end{array}\right]\right) \\ \left[\begin{array}{ccccc} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24} \end{array}\right]$$

Nota: El Vacío ( $\_$ ) en la matriz de términos iniciales anterior se usa para indicar que el término inicial para  $u_1(n)$  se calcula utilizando la fórmula de secuencia explícita  $u_1(n)=1/n$ .

## seqn()

**seqn**(Expr(*u*, *n* [, *ListaDeTérminosInic* [, *nMax* [, *ValorMax*]])) lista  $\Rightarrow$

Genera una lista de términos para una secuencia  $u(n)=\text{Expr}(u, n)$  como sigue: Incrementa *n* desde 1 hasta *nMax* por 1, evalúa  $u(n)$  para los valores correspondientes de *n* usando la fórmula  $\text{Expr}(u, n)$  y *ListaDeTérminosInic*, y entrega los resultados como una lista.

**seqn**(Expr(*n* [, *nMax* [, *ValorMax*]]) lista  $\Rightarrow$

Genera una lista de términos para una secuencia no recursiva  $u(n)=\text{Expr}(n)$  como sigue: Incrementa *n* desde 1 hasta *nMax* por 1, evalúa  $u(n)$  para los valores correspondientes de *n* usando la fórmula  $\text{Expr}(n)$  y entrega los resultados como una lista.

Si *nMax* falta, *nMax* se configura a 2500

Genera los 6 primeros términos de la secuencia  $u(n) = u(n-1)/2$ , con  $u(1)=2$ .

$$\text{seqn}\left(\frac{u(n-1)}{n}, \{2\}, 6\right) \\ \left\{2, 1, \frac{1}{3}, \frac{1}{12}, \frac{1}{60}, \frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2}, 6\right) \\ \left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25}, \frac{1}{36}\right\}$$



Si  $nMax=0$ ,  $nMax$  se configura a 2500

**Nota:** `seqn()` llama `seqGen()` con  $n0=1$  y  $n paso =1$

### setMode() (configModo)

`setMode(enteroNombreModo, enteroConfig) ⇒ entero`

`setMode(lista) ⇒ lista de enteros`

Sólo es válido dentro de una función o un programa.

`setMode(enteroNombreModo, enteroConfig)` configura en forma temporal el modo *enteroNombreModo* a la nueva configuración *enteroConfigy* entrega un entero correspondiente a la configuración original de ese modo. El cambio está limitado a la duración de la ejecución del programa/la función.

*enteroNombreModo* especifica cuál modo usted desea configurar. Debe ser uno de los enteros de modo de la tabla de abajo.

*enteroConfig* especifica la nueva configuración para el modo. Debe ser uno de los enteros de configuración que se enumeran abajo para el modo específico que usted está configurando.

`setMode(lista)` le permite cambiar varias configuraciones. *lista* contiene pares de enteros de modo y enteros de configuración. `setMode(lista)` entrega una lista similar cuyos pares de enteros representan los modos y las configuraciones originales.

Si usted ha guardado todas las configuraciones de modo con `getMode(0) → var`, podrá usar `setMode(var)` para restaurar esas configuraciones hasta que la función o el programa exista. Vea `getMode()`, página 70.

Despliegue el valor aproximado de  $\pi$  usando la configuración predeterminada para Desplegar Dígitos, y luego despliegue  $\pi$  con una configuración de Fijo2. Revise para ver que el predeterminado esté restaurado después de que se ejecute el programa.

Define <i>prog1()</i> =Prgm	Done
Disp $\pi$	
setMode(1,16)	
Disp $\pi$	
EndPrgm	
<hr/>	
<i>prog1()</i>	
	3.14159
	3.14
	Done

**Nota:** Las configuraciones del modo actual se pasan a las subrutinas llamadas. Si cualquier subrutina cambia una configuración del modo, el cambio de modo se perderá cuando el control regrese a la rutina de llamada.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

Modo Nombre	Modo Entero	Cómo configurar enteros
Desplegar dígitos	1	1=Flotante, 2=Flotante1, 3=Flotante2, 4=Flotante3, 5=Flotante4, 6=Flotante5, 7=Flotante6, 8=Flotante7, 9=Flotante8, 10=Flotante9, 11=Flotante10, 12=Flotante11, 13=Flotante12, 14=Fijo0, 15=Fijo1, 16=Fijo2, 17=Fijo3, 18=Fijo4, 19=Fijo5, 20=Fijo6, 21=Fijo7, 22=Fijo8, 23=Fijo9, 24=Fijo10, 25=Fijo11, 26=Fijo12
Ángulo	2	1=Radián, 2=Grado, 3=Gradián
Formato exponencial	3	1=Normal, 2=Científico, 3=Ingeniería
Real o Complejo	4	1=Real, 2=Rectangular, 3=Polar
Auto o Aprox.	5	1=Auto, 2=Aproximado
Formato de Vector	6	1=Rectangular, 2=Cilíndrico, 3=Esférico
Base	7	1=Decimal, 2=Hexagonal, 3=Binario

## shift() (cambiar)

**shift(Entero1[,#deCambios])**⇒entero

En modo de base binaria:

```

shift(0b1111010110000110101)
                                0b111101011000011010
shift(256,1)                      0b1000000000

```

En modo de base hexadecimal:

Cambia los bits en un entero binario. Usted puede ingresar *Entero1* en cualquier base de números; se convierte automáticamente en una forma binaria de 64 bits signada. Si la magnitud de *Entero1* es demasiado grande para esta forma, una operación de módulo simétrico lo lleva dentro del rango. Para obtener más información, vea ▶**Base2**, página 17.

shift(0h78E)	0h3C7
shift(0h78E,-2)	0h1E3
shift(0h78E,2)	0h1E38

**Importante:** Para ingresar un número binario o hexadecimal, use siempre el prefijo 0b ó 0h (cero, no la letra O).

Si *#deCambios* es positivo, el cambio es hacia la izquierda. Si *#deCambios* es negativo, el cambio es hacia la derecha. El predeterminado es -1 (cambiar a la derecha un bit).

En un cambio a la derecha, el bit del extremo derecho se elimina y 0 ó 1 se inserta para coincidir con el bit del extremo izquierdo. En un cambio a la izquierda, el bit del extremo izquierdo se elimina y 0 ó 1 se inserta como el bit del extremo derecho.

Por ejemplo, en un cambio a la derecha:

Cada bit cambia a la derecha.

0b0000000000000111101011000011010

Inserta 0 si el bit del extremo izquierdo es 0, ó 1 si el bit del extremo izquierdo es 1.

produce:

0b00000000000000111101011000011010

El resultado se despliega de acuerdo con el modo de la Base. Los ceros líderes no se muestran.

**shift(Lista1 [,#deCambios])⇒lista**

Entrega una copia de *Listas1* cambiada a la derecha o a la izquierda por elementos de *#de Cambios* . No altera *Listas1* .

En modo de base decimal:

shift({1,2,3,4})	{undef,1,2,3}
shift({1,2,3,4},-2)	{undef,undef,1,2}
shift({1,2,3,4},2)	{3,4,undef,undef}

Si *#deCambios* es positivo, el cambio es hacia la izquierda. Si *#deCambios* es negativo, el cambio es hacia la derecha. El predeterminado es -1 (cambiar a la derecha un elemento).

Los elementos introducidos al principio o al final de *lista* por medio del cambio están configurados al símbolo "indef".

**shift(*Cadena1* [,*#deCambios*])⇒*cadena***

Entrega una copia de *Cadena1* cambiada a la derecha o a la izquierda por caracteres de *#de Cambios* . No altera *Cadena1*.

shift("abcd")	" abc"
shift("abcd",-2)	" ab"
shift("abcd",1)	"bcd "

Si *#deCambios* es positivo, el cambio es hacia la izquierda. Si *#deCambios* es negativo, el cambio es hacia la derecha. El predeterminado es -1 (cambiar a la derecha un caracter).

Los caracteres introducidos al principio o al final de *cadena* por medio del cambio están configurados a un espacio.

**sign()**

**sign(*Valor1*)⇒*valor***

sign(-3.2)	-1
------------	----

**sign(*Lista1*)⇒*lista***

sign({2,3,4,-5})	{1,1,1,-1}
------------------	------------

**sign(*Matriz1*)⇒*matriz***

Para *Valor1* real o complejo, entrega *Valor1* / **abs(*Valor1*)** cuando *Valor1* ≠ 0.

Si el modo de formato complejo es Real:

sign([-3 0 3])	[-1 undef 1]
----------------	--------------

**Entrega 1 si *Valor1* es positivo.**

**Entrega -1 si *Valor1* es negativo.**

**sign(0)** entrega ±1 si el modo de formato complejo es Real; de otro modo, se entrega a sí mismo.

**sign(0)** representa el círculo de unidad en el dominio complejo.

Para una lista o matriz, entrega los signos de todos los elementos.

**simult(matrizCoef, vectorConst[, Tol])** ⇒ matriz

Entrega un vector de columna que contiene las soluciones para un sistema de ecuaciones lineales.

Nota: Vea también **linSolve()**, página 89.

*matrizCoef* debe ser una matriz cuadrada que contiene los coeficientes de las ecuaciones.

*vectorConst* debe tener el mismo número de filas (misma dimensión) que *matrizCoef* y contener las constantes.

De manera opcional, cualquier elemento de matriz se trata como cero si su valor absoluto es menor que la *Tolerancia*. Esta tolerancia se usa sólo si la matriz tiene ingresos de punto flotante y no contiene ninguna variable simbólica a la que no se le haya asignado un valor. De otro modo, la *Tolerancia* se ignora.

- Si usted configura el modo **Auto o Aproximado** en Aproximado, los cálculos se hacen usando aritmética de punto flotante.
- Si la *Tolerancia* se omite o no se usa, la tolerancia predeterminada se calcula como:  
 $5E-14 \cdot \max(\dim(\text{matrizCoef}), \text{normaFila}(\text{matrizCoef}))$

**simult(matrizCoef, matrizConst[, Tol])** ⇒ matriz

Soluciona varios sistemas de ecuaciones lineales, donde cada sistema tiene los mismos coeficientes de ecuaciones pero constantes diferentes.

Cada columna en *matrizConst* debe contener las constantes para un sistema de ecuaciones. Cada columna en la matriz resultante contiene la solución para el sistema correspondiente.

Solucione para x y y:

$$x + 2y = 1$$

$$3x + 4y = -1$$

$$\text{simult}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) \quad \begin{bmatrix} -3 \\ 2 \end{bmatrix}$$

La solución es  $x=-3$  y  $y=2$ .

Solución:

$$ax + by = 1$$

$$cx + dy = 2$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow \text{matx1} \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$\text{simult}\left(\text{matx1}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) \quad \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

Solucionar:

$$x + 2y = 1$$

$$3x + 4y = -1$$

$$x + 2y = 2$$

$$3x + 4y = -3$$

$$\text{simult}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ -1 & -3 \end{bmatrix}\right) \quad \begin{bmatrix} -3 & -7 \\ 2 & \frac{9}{2} \end{bmatrix}$$

Para el primer sistema,  $x=-3$  y  $y=2$ . Para el segundo sistema,  $x=-7$  y  $y=9/2$ .

**sin()** (sen)

 tecla

$\sin(\text{Valor1}) \Rightarrow \text{valor}$

En modo de ángulo en Grados:

$\sin(\text{Lista1}) \Rightarrow \text{lista}$

$$\sin\left(\left[\frac{\pi}{4}\right]r\right) \quad 0.707107$$

$\sin(\text{Valor1})$  entrega el seno del argumento.

$$\sin(45) \quad 0.707107$$

$\sin(\text{Lista1})$  entrega una lista de senos de todos los elementos en *Lista1*.

$$\sin(\{0,60,90\}) \quad \{0,0.866025,1\}$$

**Nota:** El argumento se interpreta como un ángulo en grados, gradianes o radianes, de acuerdo con el modo del ángulo actual. Usted puede usar °, G o r para anular la configuración del modo de ángulo en forma temporal.

En modo de ángulo en Gradianes:

$$\sin(50) \quad 0.707107$$

En modo de ángulo en Radianes:

$$\sin\left(\frac{\pi}{4}\right) \quad 0.707107$$

$$\sin(45^\circ) \quad 0.707107$$

**sin**

$(\text{matrizCuadrada1}) \Rightarrow \text{matrizCuadrada}$

Entrega el seno de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular el seno de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

En modo de ángulo en Radianes:

$$\sin\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right) \quad \begin{bmatrix} 0.9424 & -0.04542 & -0.031999 \\ -0.045492 & 0.949254 & -0.020274 \\ -0.048739 & -0.00523 & 0.961051 \end{bmatrix}$$

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

**sin<sup>-1</sup>()** (sen<sup>-1</sup>)

 tecla

$\sin^{-1}(\text{Valor1}) \Rightarrow \text{valor}$

En modo de ángulo en Grados:

$\sin^{-1}(\text{Lista1}) \Rightarrow \text{lista}$

$$\sin^{-1}(1) \quad 90.$$

$\sin^{-1}(\text{Valor1})$  entrega el ángulo cuyo seno es *Valor1*.

En modo de ángulo en Gradianes:

## $\sin^{-1}()$ ( $\text{sen}^{-1}$ )

 **tecla**

$\sin^{-1}(\text{Lista1})$  entrega una lista de senos inversos de cada elemento de *Lista1*.

**Nota:** El resultado se entrega como un ángulo en grados, gradianes o radianes, de acuerdo con la configuración del modo del ángulo actual.

**Nota:** Usted puede insertar esta función desde el teclado al escribir **arcosen** (...).

$\sin^{-1}$   
(*matrizCuadrada1*) $\Rightarrow$ *matrizCuadrada*

Entrega el seno inverso de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular el seno inverso de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

$\sin^{-1}(1)$  100.


En modo de ángulo en Radianes:

$\sin^{-1}(\{0,0.2,0.5\})$   $\{0.,0.201358,0.523599\}$

En el modo de ángulo en Radianes y el modo de formato complejo Rectangular:

$\sin^{-1}\left(\begin{bmatrix} 1 & 5 \\ 4 & 2 \end{bmatrix}\right)$   
 $\begin{bmatrix} -0.174533-0.12198 \cdot i & 1.74533-2.35591 \cdot i \\ 1.39626-1.88473 \cdot i & 0.174533-0.593162 \cdot i \end{bmatrix}$

## $\sinh()$ ( $\text{senh}$ )

**Catálogo** > 

$\sinh(\text{verNúm1}) \Rightarrow \text{valor}$

$\sinh(1.2)$  1.50946

$\sinh(\text{Lista1}) \Rightarrow \text{lista}$

$\sinh(\{0,1,2,3\})$   $\{0,1.50946,10.0179\}$

$\sinh(\text{Valor1})$  entrega el seno hiperbólico del argumento.

$\sinh(\text{Lista1})$  entrega una lista de senos hiperbólicos de cada elemento de *Lista1*.

$\sinh$   
(*matrizCuadrada1*) $\Rightarrow$ *matrizCuadrada*

Entrega el seno hiperbólico de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular el seno hiperbólico de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

En modo de ángulo en Radianes:

$\sinh\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right)$   
 $\begin{bmatrix} 360.954 & 305.708 & 239.604 \\ 352.912 & 233.495 & 193.564 \\ 298.632 & 154.599 & 140.251 \end{bmatrix}$

## sinh() (senh)

Catálogo > 

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

## sinh<sup>-1</sup>() (senh<sup>-1</sup>)

Catálogo > 

sinh<sup>-1</sup>(*Valor1*) ⇒ *valor*

$$\sinh^{-1}(0) \quad 0$$

sinh<sup>-1</sup>(*Lista1*) ⇒ *lista*

$$\sinh^{-1}\{0,2,1,3\} \quad \{0,1.48748,1.81845\}$$

sinh<sup>-1</sup>(*Valor1*) entrega el seno hiperbólico inverso del argumento.

sinh<sup>-1</sup>(*Lista1*) entrega una lista de los senos hiperbólicos inversos de cada elemento de *Lista1*.

**Nota:** Usted puede insertar esta función desde el teclado al escribir **arcosenh** (...).

sinh<sup>-1</sup>  
(*matrizCuadrada1*) ⇒ *matrizCuadrada*

Entrega el seno hiperbólico inverso de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular el seno hiperbólico inverso de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

En modo de ángulo en Radianes:

$$\sinh^{-1}\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \begin{bmatrix} 0.041751 & 2.15557 & 1.1582 \\ 1.46382 & 0.926568 & 0.112557 \\ 2.75079 & -1.5283 & 0.57268 \end{bmatrix}$$

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

## SinReg

Catálogo > 

SinReg *X*, *Y* [, [*Iteraciones*] , [*Periodo*] [, *Categoría*, *Incluir*] ]

Resuelve la regresión sinusoidal en las listas *X* y *Y*. Se almacena un resumen de resultados en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

*X* y *Y* son listas de variables independientes y dependientes.



*Iteraciones* es un valor que especifica el número máximo de veces (1 a 16) que se intentará una solución. Si se omite, se usa 8. Por lo general, los valores más grandes dan como resultado una mejor exactitud, pero tiempos de ejecución más largos, y viceversa.

*Periodo* especifica un periodo estimado. Si se omite, la diferencia entre los valores en  $X$  deberán ser iguales y estar en orden secuencial. Si usted especifica el *Periodo*, las diferencias entre los valores  $x$  pueden ser desiguales.

*Categoría* es una lista de códigos de categoría numérica o de cadena para los datos  $X$  y  $Y$  correspondientes.

*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

El resultado de **SinReg** siempre es en radianes, independientemente de la configuración del modo de ángulo.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

Variable de salida	Descripción
stat.EcnReg	Ecuación de Regresión: $a \cdot \sin(bx+c)+d$
stat.a, stat.b, stat.c, stat.d	Coefficientes de regresión
stat.Resid	Residuales de la regresión
stat.XReg	La lista de puntos de datos en la <i>Lista X</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoríaae</i> <i>Incluir</i>
stat.YReg	La lista de puntos de datos en la <i>Lista Y</i> modificada que se usa en realidad en la regresión con base en las restricciones de las Categorías <i>Frec</i> , <i>Lista de Categoríaae</i> <i>Incluir</i>
stat.FrecReg	Lista de frecuencias correspondientes a <i>stat.XReg</i> y <i>stat.YReg</i>

## SortA (OrdenarA)

Catálogo > 

**SortA** *Lista1* [, *Lista2*] [, *Lista3*] ...

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
---------------------------------	---------------

**SortA** *Vector1* [, *Vector2*] [, *Vector3*] ...

SortA <i>list1</i>	Done
--------------------	------

Ordena los elementos del primer argumento en orden ascendente.

<i>list1</i>	$\{1,2,3,4\}$
--------------	---------------

Si usted incluye argumentos adicionales, ordena los elementos de cada uno, de manera que sus nuevas posiciones coinciden con las nuevas posiciones de los elementos en el primer argumento.

$\{4,3,2,1\} \rightarrow list2$	$\{4,3,2,1\}$
---------------------------------	---------------

Todos los argumentos deben ser nombres de listas o vectores. Todos los argumentos deben tener dimensiones iguales.

SortA <i>list2,list1</i>	Done
--------------------------	------

Los elementos vacíos (inválidos) dentro del primer argumento se mueven a la parte inferior. Para obtener más información sobre elementos vacíos, vea página 231.

<i>list2</i>	$\{1,2,3,4\}$
--------------	---------------

<i>list1</i>	$\{4,3,2,1\}$
--------------	---------------

## SortD (OrdenarD)

Catálogo > 

**SortD** *Lista1* [, *Lista2*] [, *Lista3*] ...

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
---------------------------------	---------------

**SortD** *Vector1* [, *Vector2*] [, *Vector3*] ...

$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
---------------------------------	---------------

Idéntico a **SortA**, excepto que **SortD** ordena los elementos en orden descendente.

SortD <i>list1,list2</i>	Done
--------------------------	------

Los elementos vacíos (inválidos) dentro del primer argumento se mueven a la parte inferior. Para obtener más información sobre elementos vacíos, vea página 231.

<i>list1</i>	$\{4,3,2,1\}$
--------------	---------------

<i>list2</i>	$\{3,4,1,2\}$
--------------	---------------

## ►Sphere (►Esfera)

Catálogo > 

*Vector* ►Sphere

$[1 \ 2 \ 3] \text{►Sphere}$
$[3.74166 \ \angle 1.10715 \ \angle 0.640522]$

**Nota:** Usted puede insertar este operador desde el teclado de la computadora al escribir **e>Sphere**.

## ►Sphere (►Esfera)

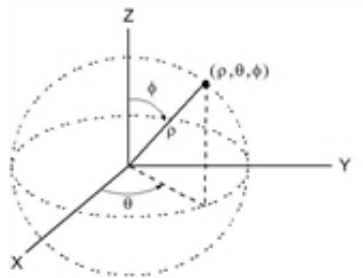
Catálogo > 

Despliega el vector de fila o columna en forma esférica [ $\rho$   $\angle$   $\theta$   $\angle$   $\phi$ ].

*Vector* debe ser de dimensión 3 y puede ser un vector de fila o de columna.

**Nota:** ►Sphere es una instrucción de formato de despliegue, no una función de conversión. Usted puede usarla sólo al final de una línea de ingreso.

$$\left( \begin{matrix} 2 & \angle & \frac{\pi}{4} & 3 \end{matrix} \right) \text{►Sphere}$$
$$\left[ 3.60555 \quad \angle 0.785398 \quad \angle 0.588003 \right]$$



## sqrt()

Catálogo > 

$\text{sqrt}(Valor1) \Rightarrow valor$

$$\sqrt{4} \quad 2$$

$\text{sqrt}(Lista) \Rightarrow lista$

$$\sqrt{\{9,2,4\}} \quad \{3,1.41421,2\}$$

Entrega la raíz cuadrada del argumento.

Para una lista, entrega las raíces cuadradas de todos los elementos en *Lista1*.

**Nota:** Vea también **Plantilla de raíz cuadrada**, página 1.

**stat.results**

Despliega los resultados de un cálculo de estadísticas.

Los resultados se despliegan como un conjunto de pares de valores de nombres. Los nombres específicos que se muestran dependen de la función o del comando de estadísticas evaluado de manera más reciente.

Usted puede copiar un nombre o valor y pegarlo en otras ubicaciones.

**Nota:** Evite definir variables que usan los mismos nombres que aquellos que se usan para análisis estadístico. En algunos casos, podría ocurrir una condición de error. Los nombres de variable que se usan para análisis estadístico se enumeran en la tabla de abajo.

 $xlist:=\{1,2,3,4,5\}$        $\{1,2,3,4,5\}$ 
 $ylist:=\{4,8,11,14,17\}$        $\{4,8,11,14,17\}$ 
LinRegMx  $xlist,ylist,1$ : stat.results

"Title"	"Linear Regression (mx+b)"
"RegEqn"	"m*x+b"
"m"	3.2
"b"	1.2
"r <sup>2</sup> "	0.996109
"r"	0.998053
"Resid"	"{...}"


stat.values	"Linear Regression (mx+b)"
	"m*x+b"
	3.2
	1.2
	0.996109
	0.998053
	"{-0.4,0.4,0.2,0,-0.2}"

stat.a	stat.dfDenom	stat.MedianY	stat.Q3X	stat.SSBlock
stat.AdjR <sup>2</sup>	stat.dfBlock	stat.MEPred	stat.Q3Y	stat.SSCol
stat.b	stat.dfCol	stat.MinX	stat.r	stat.SSX
stat.b0	stat.dfError	stat.MinY	stat.r <sup>2</sup>	stat.SSY
stat.b1	stat.dfInteract	stat.MS	stat.RegEqn	stat.SSError
stat.b2	stat.dfReg	stat.MSBlock	stat.Resid	stat.SSInteract
stat.b3	stat.dfNumer	stat.MSCol	stat.ResidTrans	stat.SSReg
stat.b4	stat.dfRow	stat.MSError	stat.ox	stat.SSRow
stat.b5	stat.DW	stat.MSInteract	stat.oy	stat.tList
stat.b6	stat.e	stat.MSReg	stat.ox1	stat.UpperPred
stat.b7	stat.ExpMatrix	stat.MSRow	stat.ox2	stat.UpperVal
stat.b8	stat.F	stat.n	stat.Σx	stat.̄x
stat.b9	stat.FBlock	stat.̂p	stat.Σx <sup>2</sup>	stat.̄x1
stat.b10	stat.Fcol	stat.̂p1	stat.Σxy	stat.̄x2
stat.bList	stat.FInteract	stat.̂p2	stat.Σy	stat.̄xDiff
stat.χ <sup>2</sup>	stat.FreqReg	stat.̂pDiff	stat.Σy <sup>2</sup>	stat.̄xList

stat.c	stat.Frow	stat.PList	stat.s	stat.XReg
stat.CLower	stat.Leverage	stat.PVal	stat.SE	stat.XVal
stat.CLowerList	stat.LowerPred	stat.PValBlock	stat.SEList	stat.XValList
stat.CompList	stat.LowerVal	stat.PValCol	stat.SEPred	stat. $\bar{y}$
stat.CompMatrix	stat.m	stat.PValInteract	stat.sResid	stat. $\hat{y}$
stat.CookDist	stat.MaxX	stat.PValRow	stat.SEslope	stat. $\hat{y}$ List
stat.CUpper	stat.MaxY	stat.Q1X	stat.sp	stat.YReg
stat.CUpperList	stat.ME	stat.Q1Y	stat.SS	
stat.d	stat.MedianX			

**Nota:** Cada vez que la aplicación de Listas y Hoja de Cálculo calcula resultados estadísticos, copia las variables del grupo “stat.” a un grupo “stat.#.”, donde # es un número que se incrementa en forma automática. Esto le permite mantener los resultados anteriores mientras realiza varios cálculos.

## stat.values

Catálogo > 

stat.values


Vea el ejemplo de **stat.results**.

Despliega una matriz de los valores calculados para la función o el comando de estadísticas evaluado de manera más reciente.

A diferencia de **stat.results**, **stat.values** omite los nombres asociados con los valores.

Usted puede copiar un valor y pegarlo en otras ubicaciones.

## stDevPop() (desvEstPob)

Catálogo > 

**stDevPop**(*Lista* [, *listaFrec*]) ⇒ *expresión*

En modos de ángulo en Radianes y auto:

Entrega la desviación estándar de población de los elementos en *Lista*.

stDevPop({ 1,2,5,-6,3,-2 })	3.59398
-----------------------------	---------

stDevPop({ 1.3,2.5,-6.4 }, { 3,2,5 })	4.11107
---------------------------------------	---------

Cada elemento de *listaFrec* cuenta el número de ocurrencias consecutivas del elemento correspondiente en *Lista*.

**Nota:** *Lista* debe tener al menos dos elementos. Los elementos vacíos (inválidos) se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

**stDevPop**(*Matriz1* [, *matrizFrec*]) ⇒ *matriz*

Entrega un vector de fila de las desviaciones estándar de población las columnas en *Matriz1*.

Cada elemento de *matrizFrec* cuenta el número de ocurrencias consecutivas del elemento correspondiente en *Matriz1*.

**Nota:** *Matriz1* debe tener al menos dos filas. Los elementos vacíos (inválidos) se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

$$\text{stDevPop} \left( \begin{array}{ccc} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{array} \right) \\ \left[ 3.26599 \quad 2.94392 \quad 1.63299 \right]$$

$$\text{stDevPop} \left( \begin{array}{cc} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{array} , \begin{array}{cc} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{array} \right) \\ \left[ 2.52608 \quad 5.21506 \right]$$

## stDevSamp() (desvEstMuest)

**stDevSamp**(*Lista* [, *listaFrec*]) ⇒ *expresión*

Entrega la desviación estándar muestra de los elementos en *Lista*.

Cada elemento de *listaFrec* cuenta el número de ocurrencias consecutivas del elemento correspondiente en *Lista*.

**Nota:** *Lista* debe tener al menos dos elementos. Los elementos vacíos (inválidos) se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

**stDevSamp**(*Matriz1* [, *matrizFrec*]) ⇒ *matriz*

Entrega un vector de fila de las desviaciones estándar muestra de las columnas en *Matriz1*.

Cada elemento de *matrizFrec* cuenta el número de ocurrencias consecutivas del elemento correspondiente en *Matriz1*.

$$\text{stDevSamp}(\{1, 2, 5, -6, 3, -2\}) \quad 3.937$$

$$\text{stDevSamp}(\{1.3, 2.5, -6.4\}, \{3, 2, 5\}) \\ 4.33345$$

$$\text{stDevSamp} \left( \begin{array}{ccc} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{array} \right) \\ \left[ 4. \quad 3.60555 \quad 2. \right]$$

$$\text{stDevSamp} \left( \begin{array}{cc} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{array} , \begin{array}{cc} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{array} \right) \\ \left[ 2.7005 \quad 5.44695 \right]$$

**Nota:** *Matriz1* debe tener al menos dos filas. Los elementos vacíos (inválidos) se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

**Stop (Detener)****Stop**

Comando de programación: Termina el programa.

**Stop** no está permitido en las funciones.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

<i>i</i> :=0	0
Define <i>prog1</i> ()=Prgm	Done
For <i>i</i> ,1,10,1	
If <i>i</i> =5	
Stop	
EndFor	
EndPrgm	
<i>prog1</i> ()	Done
<i>i</i>	5

**Almacenar**

Vea → (almacenar), página 213.

**string() (cadena)**

**string**(*Expr*)⇒*cadena*

Simplifica *Expr* y entrega el resultado de una cadena de caracteres.

string(1.2345)	"1.2345"
string(1+2)	"3"

**subMat()**

**subMat**(*Matriz1* [, *iniciarFila*] [, *iniciarCol*] [, *terminarFila*] [, *terminarCol*]) ⇒*matriz*

Entrega la submatriz especificada de *Matriz1*.

Predeterminados: *iniciarFila*=1, *iniciarCol*=1, *terminarFila*=última fila, *terminarCol*=última columna.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
subMat( <i>m1</i> ,2,1,3,2)	$\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}$
subMat( <i>m1</i> ,2,2)	$\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$

**sum()**

Catálogo >

**sum(Lista[, Iniciar[, Terminar]])** ⇒ expresión

Entrega la suma de todos los elementos en *Lista*.

*Inicio* y *Término* son opcionales. Especifican un rango de elementos.

Cualquier argumento inválido produce un resultado inválido. Los elementos vacíos (inválidos) en *Lista* se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

**sum(MatrizI[, Iniciar[, Terminar]])** ⇒ matriz

Entrega un vector de fila que contiene las sumas de todos los elementos en las columnas de *MatrizI*.

*Inicio* y *Término* son opcionales. Especifican un rango de filas.

Cualquier argumento inválido produce un resultado inválido. Los elementos vacíos (inválidos) en *MatrizI* se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

$\text{sum}(\{1,2,3,4,5\})$	15
$\text{sum}(\{a,2 \cdot a,3 \cdot a\})$	"Error: Variable is not defined"
$\text{sum}(\text{seq}(n,n,1,10))$	55
$\text{sum}(\{1,3,5,7,9\},3)$	21

$\text{sum}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}\right)$	$[5 \ 7 \ 9]$
$\text{sum}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}\right)$	$[12 \ 15 \ 18]$
$\text{sum}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},2,3\right)$	$[11 \ 13 \ 15]$

**sumIf() (sumaSi)**

Catálogo >

**sumIf(Lista,Criterios[, ListaSuma])** ⇒ valor

Entrega la suma acumulada de todos los elementos en *Lista* que cumplen con los *Criterios* especificados. De manera opcional, usted puede especificar una lista alterna, *listaSuma*, para proporcionar los elementos a acumular.

$\text{sumIf}(\{1,2,e,3,\pi,4,5,6\},2.5 < ? < 4.5)$	12.859874482
$\text{sumIf}(\{1,2,3,4\},2 < ? < 5, \{10,20,30,40\})$	70



*Lista* puede ser una expresión, lista o matriz. *ListaSuma*, si se especifica, debe tener la(s) misma(s) dimensión(es) que *Lista*.

Los criterios pueden ser:

- Un valor, una expresión o una cadena. Por ejemplo, **34** acumula sólo aquellos elementos en *Lista* que se simplifican al valor 34.
- Una expresión Booleana que contiene el símbolo ? como un marcador de posición para cada elemento. Por ejemplo, **?<10** acumula sólo aquellos elementos en *Lista* que son menos de 10.

Cuando un elemento de *Lista* cumple con los *Criterios*, el elemento se agrega a la suma acumulativa. Si usted incluye *listaSuma*, el elemento correspondiente de *listaSuma* se agrega a la suma en su lugar.

Dentro de la aplicación de Listas y Hoja de Cálculo, usted puede usar un rango de celdas en lugar de *Lista* y *listaSuma*.


Los elementos vacíos (inválidos) se ignoran. Para obtener más información sobre elementos vacíos, vea página 231.

**Nota:** Vea también **countIf()**, página 32.

**system(Valor1 [, Valor2 [, Valor3 [, ...]])**

Entrega un sistema de ecuaciones, formateado como una lista. Usted también puede crear un sistema al usar una plantilla.

## T (trasponer)

Catálogo >  $Matriz1\mathbf{T}\Rightarrow$ matriz

Entrega el traspuesto conjugado complejo de  $Matriz1$ .

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}^{\mathbf{T}} \qquad \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

**Nota:** Usted puede insertar este operador desde el teclado de la computadora al escribir @t.

## tan()

 tecla $\tan(Valor1)\Rightarrow$ valor

En modo de ángulo en Grados:

 $\tan(Lista1)\Rightarrow$ lista

$\tan(Valor1)$  entrega la tangente del argumento.

$\tan(Lista1)$  entrega una lista de las tangentes de todos los elementos en  $Listal$ .

**Nota:** El argumento se interpreta como un ángulo en grados, gradianes o radianes, de acuerdo con el modo del ángulo actual. Usted puede usar °, G o R para anular la configuración del modo de ángulo en forma temporal.

$\tan\left(\frac{\pi}{4}\right)$	1.
$\tan(45)$	1.
$\tan(\{0,60,90\})$	$\{0.,1.73205,\text{undef}\}$

En modo de ángulo en Gradianes:

$\tan\left(\frac{\pi}{4}\right)$	1.
$\tan(50)$	1.
$\tan(\{0,50,100\})$	$\{0.,1.,\text{undef}\}$

En modo de ángulo en Radianes:

$\tan\left(\frac{\pi}{4}\right)$	1.
$\tan(45^\circ)$	1.
$\tan\left(\left\{\pi, \frac{\pi}{3}, -\pi, \frac{\pi}{4}\right\}\right)$	$\{0.,1.73205,0.,1.\}$

tan

 $(matrizCuadrada1)\Rightarrow$ matrizCuadrada

En modo de ángulo en Radianes:

## tan()

 tecla

Entrega la tangente de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular la tangente de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

$$\tan \left( \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix} \right) = \begin{bmatrix} -28.2912 & 26.0887 & 11.1142 \\ 12.1171 & -7.83536 & -5.48138 \\ 36.8181 & -32.8063 & -10.4594 \end{bmatrix}$$

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

## tan<sup>-1</sup>()

 tecla

tan<sup>-1</sup>(*Valor1*) ⇒ *valor*

En modo de ángulo en Grados:

tan<sup>-1</sup>(*Lista1*) ⇒ *lista*

$$\tan^{-1}(1) = 45$$

tan<sup>-1</sup>(*Valor1*) entrega el ángulo cuya tangente es *Valor1*.

En modo de ángulo en Gradianes:

tan<sup>-1</sup>(*Lista1*) entrega una lista de las tangentes inversas de cada elemento de *Lista1*.

$$\tan^{-1}(1) = 50$$

**Nota:** El resultado se entrega como un ángulo en grados, gradianes o radianes, de acuerdo con la configuración del modo del ángulo actual.

En modo de ángulo en Radianes:

$$\tan^{-1}(\{0,0,2,0,5\}) = \{0,0.197396,0.463648\}$$

**Nota:** Usted puede insertar esta función desde el teclado al escribir **arcotan** (...).

## tan<sup>-1</sup>

(*matrizCuadrada1*) ⇒ *matrizCuadrada*

En modo de ángulo en Radianes:

Entrega la tangente inversa de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular la tangente inversa de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

$$\tan^{-1} \left( \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix} \right) = \begin{bmatrix} -0.083658 & 1.26629 & 0.62263 \\ 0.748539 & 0.630015 & -0.070012 \\ 1.68608 & -1.18244 & 0.455126 \end{bmatrix}$$

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

**tanh**(*Valor1*)⇒*valor*

tanh(1.2) 0.833655

**tanh**(*Lista1*)⇒*lista*

tanh({0,1}) {0.,0.761594}

**tanh**(*Valor1*) entrega la tangente hiperbólica del argumento.**tanh**(*Lista1*) entrega una lista de las tangentes hiperbólicas de cada elemento de *Lista1*.**tanh****(matrizCuadrada1)**⇒*matrizCuadrada*

Entrega la tangente hiperbólica de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular la tangente hiperbólica de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

En modo de ángulo en Radianes:

$$\text{tanh} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} -0.097966 & 0.933436 & 0.425972 \\ 0.488147 & 0.538881 & -0.129382 \\ 1.28295 & -1.03425 & 0.428817 \end{bmatrix}$$

**tanh<sup>-1</sup>()****tanh<sup>-1</sup>**(*Valor1*)⇒*valor*

En formato complejo Rectangular:


**tanh<sup>-1</sup>**(*Lista1*)⇒*lista*tanh<sup>-1</sup>(0) 0.**tanh<sup>-1</sup>**(*Valor1*) entrega la tangente hiperbólica inversa del argumento.tanh<sup>-1</sup>({1,2,1,3})  
{undef,0.518046-1.5708·i,0.346574-1.5708·i}**tanh<sup>-1</sup>**(*Lista1*) entrega una lista de las tangentes hiperbólicas inversas de cada elemento de *Lista1*.Para ver el resultado completo, presione **▲** y después use **◀** y **▶** para mover el cursor.**Nota:** Usted puede insertar esta función desde el teclado al escribir **arctanh** (...).**tanh<sup>-1</sup>****(matrizCuadrada1)**⇒*matrizCuadrada*

En el modo de ángulo en Radianes y el formato complejo Rectangular:

Entrega la tangente hiperbólica inversa de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular la tangente hiperbólica inversa de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

$$\text{tanh}^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} -0.099353+0.164058\cdot i & 0.267834-1.4908 \\ -0.087596-0.725533\cdot i & 0.479679-0.94730 \\ 0.511463-2.08316\cdot i & -0.878563+1.7901 \end{bmatrix}$$


## $\tanh^{-1}()$

Catálogo > 

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

Para ver el resultado completo, presione **▲** y después use **◀** y **▶** para mover el cursor.

## tCdf()

Catálogo > 


### tCdf

$(\text{límiteInferior}, \text{límiteSuperior}, df) \Rightarrow$  número si el *límiteInferior* y el *límiteSuperior* son números, *lista* si el *límiteInferior* y el *límiteSuperior* son listas

Resuelve la probabilidad de distribución de Student-*t* entre el *límiteInferior* y el *límiteSuperior* para los grados de libertad especificados *df*.

Para  $P(X \leq \text{límiteSuperior})$ , configure *límiteInferior* = -9E999.

## Text

Catálogo > 

### TextindicarCad[, DespBandera]

Comando de programación: Pausa el programa y despliega la cadena de caracteres *indicarCad* en un cuadro de diálogo.

Cuando el usuario selecciona **OK**, la ejecución del programa continúa.



El argumento *bandera* opcional puede ser cualquier expresión.

- Si *DespBandera* se omite o se evalúa a **1**, el mensaje de texto se agrega al historial de la Calculadora.
- Si *DespBandera* se evalúa a **0**, el mensaje de texto no se agrega al historial.

Si el programa necesita una respuesta escrita del usuario, consulte **Request**, página 136 o **RequestStr**, página 138.

**Nota:** Usted puede usar este comando dentro de un programa definido por el usuario, pero no dentro de una función.

Defina un programa que pause para desplegar cada uno de cinco números aleatorios en un cuadro de diálogo.

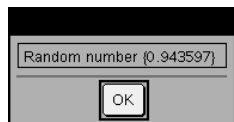
Dentro de la plantilla Prgm...TerminarPrgm, llene cada línea al presionar  en lugar de . En el teclado de la computadora, presione y sostenga **Alt** y presione **Ingresar**.

```
Define text_demo()=Prgm
  For i,1,5
    string:= "Random number " &
string(rand(i))
    Text string
  EndFor
EndPrgm
```

Ejecute el programa:

```
text_demo()
```

Muestra de un cuadro de diálogo:



## Then (Entonces)

Vea If, página 74.

## tInterval (intervaloT)

Catálogo > 

**tInterval** *Lista* [, *Frec* [, *nivelC*]]

(Entrada de lista de datos)

**tInterval**  $\bar{x}$ , *sx*, *n* [, *nivelC*]


(Entrada de estadísticas de resumen)

Resuelve un intervalo de confianza  $t$ . Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.CBajo, stat.CAlto	Intervalo de confianza para una media de población desconocida
stat. $\bar{x}$	Media de la muestra de la secuencia de datos de la distribución aleatoria normal
stat.ME	Margen de error
stat.df	Grados de libertad
stat. $\sigma_x$	Desviación estándar muestra
stat.n	Longitud de la secuencia de datos con media de la muestra muestra

## tInterval\_2Samp (intervaloT\_2Muest)

Catálogo > 

**tInterval\_2Samp** *Lista1*, *Lista2* [, *Frec1* [, *Frec2* [, *nivelC* [, *Agrupado*]]]]

(Entrada de lista de datos)

**tInterval\_2Samp**  $\bar{x}1, sx1, n1, \bar{x}2, sx2, n2$   
[*nivelC*, *Agrupado*]

(Entrada de estadísticas de resumen)

Resuelve un intervalo de confianza  $t$  de dos muestras. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

*Agrupado*=1 agrupa las varianzas;  
*Agrupado*=0 no agrupa las varianzas.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.CBajo, stat.CAlto	Intervalo de confianza que contiene la probabilidad de distribución del nivel de confianza
stat. $\bar{x}1$ - $\bar{x}2$	Medias de las muestras de las secuencias de datos de la distribución aleatoria normal
stat.ME	Margen de error
stat.df	Grados de libertad
stat. $\bar{x}1$ , stat. $\bar{x}2$	Medias muestra de las secuencias de datos de la distribución aleatoria normal
stat. $\sigma x1$ , stat. $\sigma x2$	Desviaciones estándar muestra para <i>Lista 1</i> y <i>Lista 2</i>
stat.n1, stat.n2	Número de muestras en las secuencias de datos
stat.sp	La desviación estándar agrupada. Calculada cuando <i>Agrupado</i> = Sí

### tPdf() (PdfT)

**tPdf**(*ValX*,*df*) $\Rightarrow$ número si *ValX* es un número, *lista* si *ValX* es una lista

Resuelve la función de densidad de probabilidad (pdf) para la distribución de Student- $t$  a un valor  $x$  especificado con grados de libertad  $df$  especificados.

trace(matrizCuadrada)⇒valor

Entrega el trazado (suma de todos los elementos de la diagonal principal) de matrizCuadrada.

trace	$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$	15
a:=12		12
trace	$\begin{pmatrix} a & 0 \\ 1 & a \end{pmatrix}$	24

Try (Intentar)

Try  
 bloque1  
 Else  
 bloque2  
 EndTry

Ejecuta el *bloque1* a menos que ocurra un error. La ejecución del programa se transfiere al *bloque2* si ocurre un error en el *bloque1*. La variable de sistema *códigoErr* contiene el código del error para permitir que el programa ejecute la recuperación del error. Para obtener una lista de códigos de error, vea "Códigos y mensajes de error", página 241.

*bloque1* y *bloque2* pueden ser una sentencia sencilla o una serie de sentencias separadas por el caracter ":".

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

Ejemplo 2

Para ver los comandos **Try**, **ClrErr**, y **PassErr** en operación, ingrese el programa `valspropios()` que se muestra a la derecha. Ejecute el programa al ejecutar cada una de las siguientes expresiones.

---


$$\text{eigenvals} \left( \begin{pmatrix} -3 \\ -41 \\ 5 \end{pmatrix}, \begin{bmatrix} -1 & 2 & -3.1 \end{bmatrix} \right)$$


---

```
Define prog1()=Prgm
    Try
        z:=z+1
        Disp "z incremented."
    Else
        Disp "Sorry, z undefined."
    EndTry
EndPrgm
Done
z:=1:prog1()
z incremented.
Done
DelVar z:prog1()
Sorry, z undefined.
Done
```

Defina `valspropios(a,b)=Prgm`

© El programa `valspropios(A,B)` despliega los valores propios de

```
Try
    Disp "A= ",a
    Disp "B= ",b
    Disp ""
```



**Nota:** Vea también **ClrErr**, página 24 y **PassErr**, página 119.

```

Disp "Los valores propios de A·B son:",eigVl
(a*b)

Else

  If errCode=230 Then

    Disp "Error: El producto de A·B debe ser
una matriz cuadrada"

    ClrErr

  Else

    PassErr

  EndIf

EndTry

EndPrgm

```

## tTest (pruebaT)

**tTest**  $\mu_0$ ,Lista[,Frec[,Hipot]]

(Entrada de lista de datos)

**tTest**  $\mu_0$ , $\bar{x}$ ,sx,n,[Hipot]

(Entrada de estadísticas de resumen)

Realiza una prueba de hipótesis para una sola media de población desconocida  $\mu$  cuando la desviación estándar de población,  $\sigma$  se desconoce. Un resumen de resultados se almacena en la variable *stat.results*. (página 160).

Pruebe  $H_0: \mu = \mu_0$ , contra uno de los siguientes:

Para  $H_a: \mu < \mu_0$ , configure *Hipot*<0


Para  $H_a: \mu \neq \mu_0$  (predeterminado), configure *Hipot*=0

Para  $H_a: \mu > \mu_0$ , configure *Hipot*>0

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

Variable de salida	Descripción
stat.t	$(\bar{x} - \mu_0) / (\text{desvest} / \text{sqrt}(n))$
stat.ValP	Nivel más bajo de significancia en el cual la hipótesis nula se puede rechazar
stat.df	Grados de libertad
stat. $\bar{x}$	Media de muestra de la secuencia de datos en <i>Lista</i>
stat.ex	Desviación estándar muestra de la secuencia de datos
stat.n	Tamaño de la muestra

## tTest\_2Samp (pruebaT\_2Muest)

Catálogo > 

**tTest\_2Samp** *Lista1,Lista2[,Frec1[,Frec2[,Hipot[,Agrupado]]]]*

(Entrada de lista de datos)

**tTest\_2Samp**  $\bar{x}1, sx1, n1, \bar{x}2, sx2, n2[,Hipot[,Agrupado]]$

(Entrada de estadísticas de resumen)

Resuelve una prueba *T* de dos muestras. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Pruebe  $H_0: \mu_1 = \mu_2$ , contra uno de los siguientes:

Para  $H_a: \mu_1 < \mu_2$ , configure *Hipot*<0

Para  $H_a: \mu_1 \neq \mu_2$  (predeterminado), configure *Hipot*=0

Para  $H_a: \mu_1 > \mu_2$ , configure *Hipot*>0

*Agrupado*=1 agrupa las varianzas


*Agrupado*=0 no agrupa las varianzas

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

Variable de salida	Descripción
stat.t	Valor normal estándar resuelto para la diferencia de las medias

Variable de salida	Descripción
stat.ValP	Nivel más bajo de significancia en el cual la hipótesis nula se puede rechazar
stat.df	Grados de libertad para la estadística T
stat. $\bar{x}1$ , stat. $\bar{x}2$	Medias muestra de las secuencias de datos en <i>Lista 1</i> y <i>Lista 2</i>
stat.sx1, stat.sx2	Desviaciones estándar de muestras de las secuencias de datos en <i>Lista 1</i> y <i>Lista 2</i>
stat.n1, stat.n2	Tamaño de las muestras
stat.sp	La desviación estándar agrupada. Calculada cuando <i>Agrupado</i> =1.

### tvfV()

Catálogo > 


**tvfV**(*N,I,VP,Pgo,[PpA],[CpA],[PgoA]*)⇒valor

tvfV(120,5,0, 500,12,12)      77641.1

La función financiera que calcula el valor futuro del dinero.

**Nota:** Los argumentos que se usan en las funciones del VTD se describen en la tabla de argumentos del VTD, página 176. Vea también **amortTbI()**, página 7.

### tvml()

Catálogo > 


**tvml**(*N,VP,Pgo,[PpA],[CpA],[PgoA]*)⇒valor

tvml(240,100000,-1000,0,12,12)      10.5241

La función financiera que calcula la tasa de interés por año.

**Nota:** Los argumentos que se usan en las funciones del VTD se describen en la tabla de argumentos del VTD, página 176. Vea también **amortTbI()**, página 7.

### tvnN()

Catálogo > 

**tvnN**(*N,I,VP,Pgo,[PpA],[CpA],[PgoA]*)⇒valor

tvnN(5,0,-500,77641,12,12)      120.

La función financiera que calcula el número de periodos de pago.

**Nota:** Los argumentos que se usan en las funciones del VTD se describen en la tabla de argumentos del VTD, página 176. Vea también **amortTbl()**, página 7.

## tvmPmt

**tvmPmt**( $N, I, VP, VF, [PpA], [CpA], [PgoAl]$ ) $\Rightarrow$ valor

tvmPmt(60,4,30000,0,12,12)      -552.496

La función financiera que calcula la cantidad de cada pago.

**Nota:** Los argumentos que se usan en las funciones del VTD se describen en la tabla de argumentos del VTD, página 176. Vea también **amortTbl()**, página 7.

## tvmPV()

**tvmPV**( $N, I, Pgo, VF, [PpA], [CpA], [PgoAl]$ ) $\Rightarrow$ valor

tvmPV(48,4,-500,30000,12,12)      -3426.7


La función financiera que calcula el valor presente.

**Nota:** Los argumentos que se usan en las funciones del VTD se describen en la tabla de argumentos del VTD, página 176. Vea también **amortTbl()**, página 7.

argumento del VTD*	Descripción	Tipo de datos
$N$	Número de periodos de pago	número real
$I$	tasa de interés anual	número real
$VP$	Valor presente	número real
$Pgo$	cantidad de pago	número real
$VF$	Valor futuro	número real
$PpA$	Pagos por año, predeterminado=1	entero > 0
$CpA$	Periodos de capitalización por año, predeterminado=1	entero > 0
$PgoAl$	Pago vencido al final o al principio de cada periodo, predeterminado=final	entero (0=final, 1=principio)

\* Estos nombres de argumento de valor tiempo del dinero son similares a los nombres de variable del VTD (como **vtd.vp** y **vtd.pgo**) que se usan en el solucionador financiero de la aplicación de la *Calculadora*. Sin embargo, las funciones financieras no almacenan sus valores o resultados de argumento para las variables del VTD.

## TwoVar (DosVar)

Catálogo > 

**TwoVar**  $X, Y, [Frec] [, Categoría, Incluir]$

Calcula las estadísticas de DosVar. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Todas las listas deben tener una dimensión igual, excepto por *Incluir*.

$X$  y  $Y$  son listas de variables independientes y dependientes.

*Frec* es una lista opcional de valores de frecuencia. Cada elemento en *Frec* especifica la frecuencia de la ocurrencia para cada punto de datos  $X$  y  $Y$  correspondientes. El valor predeterminado es 1. Todos los elementos deben ser enteros  $\geq 0$ .

*Categoría* es una lista de códigos de categoría numérica para los datos de  $X$  y  $Y$  correspondientes.

*Incluir* es una lista de uno o más códigos de categoría. Sólo aquellos elementos de datos cuyo código de categoría está incluido en esta lista están incluidos en el cálculo.

Un elemento (inválido) vacío en cualquiera de las listas  $X$ , *Frec* o *Categoría* da como resultado un inválido para el elemento correspondiente de todas esas listas. Un elemento vacío en cualquiera de las listas  $X1$  a  $X20$  da como resultado un inválido para el elemento correspondiente de todas esas listas. Para obtener más información sobre elementos vacíos, vea página 231.

Variable de salida	Descripción
stat. $\bar{x}$	Media de valores $x$
stat. $x$	Suma de valores $x$
stat. $x^2$	Suma de valores $x^2$

<b>Variable de salida</b>	<b>Descripción</b>
stat.ex	Desviación estándar de muestra de x
stat. x	Desviación estándar de población de x
stat.n	Número de puntos de datos
stat. $\bar{y}$	Media de valores y
stat. y	Suma de valores y
stat. $y^2$	Suma de valores $y^2$
stat.sy	Desviación estándar de muestra de y
stat. y	Desviación estándar de población de y
stat. xy	Suma de los valores $x \cdot y$
stat.r	Coefficiente de correlación
stat.MínX	Mínimo de valores x
stat.C <sub>1</sub> X	1er Cuartil de x
stat.MedianaX	Mediana de x
stat.C <sub>3</sub> X	3er Cuartil de x
stat.MaxX	Máximo de valores x
stat.MínY	Mínimo de valores y
stat.C <sub>1</sub> Y	1er Cuartil de y
stat.MedY	Mediana de y
stat.C <sub>3</sub> Y	3er Cuartil de y
stat.MaxY	Máximo de valores y
stat. $(x - )^2$	Suma de cuadrados de desviaciones de la media de x
stat. $(y - )^2$	Suma de cuadrados de desviaciones de la media de y

## U

### unitV()

Catálogo > 

**unitV**(*Vector1*) ⇒ *vector*

Entrega un vector de unidad de fila o de columna, dependiendo de la forma de *Vector1*.

*Vector1* debe ser una matriz de fila sencilla o una matriz de columna sencilla.

unitV([1 2 1])	[0.408248 0.816497 0.408248]
unitV( $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ )	$\begin{pmatrix} 0.267261 \\ 0.534522 \\ 0.801784 \end{pmatrix}$

### unlock (desbloquear)

Catálogo > 

**unlock** *Var1*[, *Var2*] [, *Var3*] ...

**unlock** *Var*.

Desbloquea las variables o el grupo de variables especificado. Las variables bloqueadas no se pueden modificar ni borrar.

Vea **lock**, página 93 y **getLockInfo()**, página 70.

<i>a</i> :=65	65
lock <i>a</i>	Done
getLockInfo( <i>a</i> )	1
<i>a</i> :=75	"Error: Variable is locked."
DelVar <i>a</i>	"Error: Variable is locked."
unlock <i>a</i>	Done
<i>a</i> :=75	75
DelVar <i>a</i>	Done

## V

### varPop()

Catálogo > 

**varPop**(*Lista*[, *listaFrec*]) ⇒ *expresión*


Entrega la varianza de población de *Lista*.

Cada elemento de *listaFrec* cuenta el número de ocurrencias consecutivas del elemento correspondiente en *Lista*.

**Nota:** *Lista* debe contener al menos dos elementos.

Si un elemento en cualquiera de las listas está vacío (inválido), ese elemento se ignora, y el elemento correspondiente en la otra lista también se ignora. Para obtener más información sobre elementos vacíos, vea página 231.

varPop({5,10,15,20,25,30})	72.9167
----------------------------	---------

**varSamp() (varMuest)**Catálogo > **varSamp(Lista[, listaFrec])**⇒expresiónEntrega la varianza muestra de *Lista*.Cada elemento de *listaFrec* cuenta el número de ocurrencias consecutivas del elemento correspondiente en *Lista*.**Nota:** *Lista* debe contener al menos dos elementos.

Si un elemento en cualquiera de las listas está vacío (inválido), ese elemento se ignora, y el elemento correspondiente en la otra lista también se ignora. Para obtener más información sobre elementos vacíos, vea página 231.

**varSamp(Matriz1[, matrizFrec])**⇒matrizEntrega un vector de fila que contiene la varianza muestra de cada columna en *Matriz1*.Cada elemento de *matrizFrec* cuenta el número de ocurrencias consecutivas del elemento correspondiente en *Matriz1*.

Si un elemento en cualquiera de las matrices está vacío (inválido), ese elemento se ignora, y el elemento correspondiente en la otra matriz también se ignora. Para obtener más información sobre elementos vacíos, vea página 231.

**Nota:** *Matriz1* debe contener al menos dos filas.

$\text{varSamp}(\{1,2,5,-6,3,-2\})$	$\frac{31}{2}$
$\text{varSamp}(\{1,3,5\},\{4,6,2\})$	$\frac{68}{33}$

$\text{varSamp}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{bmatrix}\right)$	$[4.75 \ 1.03 \ 4]$
$\text{varSamp}\left(\begin{bmatrix} -1.1 & 2.2 \\ 3.4 & 5.1 \\ -2.3 & 4.3 \end{bmatrix}, \begin{bmatrix} 6 & 3 \\ 2 & 4 \\ 5 & 1 \end{bmatrix}\right)$	$[3.91731 \ 2.08411]$

**W****Wait**Catálogo > **Wait tiempoEnSegundos**Suspende la ejecución por un periodo de *tiempoEnSegundos* segundos.

Para esperar 4 segundos:

**Wait 4**

Para esperar 1/2 segundo:



**Wait 0.5**



**Wait** es especialmente útil en un programa que necesite una demora breve para permitir que los datos solicitados estén disponibles.

El argumento *tiempoEnSegundos* debe ser una expresión que se simplifica a un valor decimal en el rango de 0 a 100. El comando redondea este valor al 0.1 segundo más cercano.

Para cancelar un **Wait** que se encuentra en proceso,

- **Dispositivo portátil:** Mantenga presionada la tecla  y presione  varias veces.
- **Windows®:** Mantenga presionada la tecla **F12** y presione **Intro** varias veces.
- **Macintosh®:** Mantenga presionada la tecla **F5** y presione **Intro** varias veces.
- **iPad®:** La aplicación muestra un indicador. Puede seguir esperando o cancelar.

**Nota:** Puede usar el comando **Wait** dentro de un programa definido por el usuario, pero no dentro de una función.

Para esperar 1.3 segundos usando la variable *seccount*:

```
seccount:=1.3
Wait seccount
```

Este ejemplo enciende un LED verde durante 0.5 segundos y luego lo apaga.

```
Send "SET GREEN 1 ON"
Wait 0.5
Send "SET GREEN 1 OFF"
```


## warnCodes ()

**warnCodes**(*Expr1*, *VarEstado*)  
*expresión* ⇒

Evalúa la expresión *Expr1*, entrega el resultado y almacena los códigos de cualquier advertencia generada en la variable de lista *varEstado*. Si no se genera ninguna advertencia, esta función asigna a *varEstado* una lista vacía.

*Expr1* puede ser cualquier expresión matemática de TI-Nspire™ o de CAS de TI-Nspire™. Usted no puede usar un comando o asignación como *Expr1*.

*VarEstado* debe ser un nombre de variable válido.

	warnCodes(det([1.23456E-999]),warn)
	1.23456E-999
warn	{ 10029 }

Para obtener una lista de códigos de advertencia y mensajes asociados, vea página 250.

## when() (cuando)

**when**(*Condición*, *resultadoVerdadero* [, *resultadoFalso*][, *resultadoDesconocido*])  $\Rightarrow$  *expresión*

Entrega un *resultadoVerdadero*, *resultadoFalso* o *resultadoDesconocido*, dependiendo de si la *Condición* es verdadera, falsa o desconocida. Entrega la entrada si hay muy pocos argumentos para especificar el resultado apropiado.

Omita tanto el *resultadoFalso* como el *resultadoDesconocido* para hacer una expresión definida sólo en la región donde la *Condición* es verdadera.

Use un **undef** *resultadoFalso* para definir una expresión que se grafique sólo en un intervalo.

**when()** es útil para definir funciones recursivas.

$\text{when}(x < 0, x + 3)   x = 5$	undef
-------------------------------------	-------

$\text{when}(n > 0, n \cdot \text{factorial}(n-1), 1) \rightarrow \text{factorial}(n)$	Done
--	------

$\text{factorial}(3)$	6
3!	6

## While (Mientras)

**While** *Condición*

*Bloque*

**EndWhile**

Ejecuta las sentencias en *Bloque* siempre y cuando la *Condición* sea verdadera.

*Bloque* puede ser una sentencia sencilla o una secuencia de sentencias separadas con el caracter ":".

Define $\text{sum\_of\_recip}(n) = \text{Func}$	
Local $i, \text{tempsum}$	
$1 \rightarrow i$	
$0 \rightarrow \text{tempsum}$	
While $i \leq n$	
$\text{tempsum} + \frac{1}{i} \rightarrow \text{tempsum}$	
$i + 1 \rightarrow i$	
EndWhile	
Return $\text{tempsum}$	
EndFunc	
	Done

$\text{sum\_of\_recip}(3)$	$\frac{11}{6}$
	6

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

## X

## xor

*BooleanaExpr1* **xor** *BooleanaExpr2*  
devuelve *expresión booleana*

true xor true	false
5>3 xor 3>5	true

*BooleanaLista1* **xor** *BooleanaLista2*  
devuelve *lista booleana*

*BooleanaMatriz1* **xor** *BooleanaMatriz2*  
devuelve *matriz booleana*

Entrega verdadero si *ExprBooleana1* es verdadera y *ExprBooleana2* es falsa, o viceversa.

Entrega falso si ambos argumentos son verdaderos o si ambos son falsos. Entrega una expresión Booleana simplificada si cualquiera de los argumentos no se puede resolver a verdadero o falso.

**Nota:** Vea **or**, página 117.

*Entero1* **xor** *Entero2*  $\Rightarrow$  *entero*

Compara dos enteros reales bit por bit usando una operación **xor**. En forma interna, ambos enteros se convierten en números binarios de 64 bits firmados. Cuando se comparan los bits correspondientes, el resultado es 1 si cualquiera de los bits (pero no ambos) es 1; el resultado es 0 si ambos bits son 0 ó ambos bits son 1. El valor producido representa los resultados de los bits, y se despliega de acuerdo con el modo de Base.

En modo de base hexadecimal:

**Importante:** Utilice el número cero, no la letra "0".

0h7AC36 xor 0h3D5F	0h79169
--------------------	---------

En modo de base binaria:

0b100101 xor 0b100	0b100001
--------------------	----------

**Nota:** Un ingreso binario puede tener hasta 64 dígitos (sin contar el prefijo 0b). Un ingreso hexadecimal puede tener hasta 16 dígitos.

Se pueden ingresar enteros en cualquier base de números. Para un ingreso binario o hexadecimal, se debe usar el prefijo 0b ó 0h, respectivamente. Sin un prefijo, los enteros se tratan como decimales (base 10).

Si se ingresa un entero decimal que es demasiado grande para una forma binaria de 64 bits firmada, se usa una operación de módulo simétrico para llevar el valor al rango apropiado. Para obtener más información, vea **►Base2**, página 17.

**Nota:** Vea **or**, página 117.

## Z

### zInterval (intervaloZ)

**zInterval**  $\sigma$ ,*Lista* [,*Frec* [,*nivelC*]]

(Entrada de lista de datos)

**zInterval**  $\sigma$ , $\bar{x}$ ,*n* [,*nivelC*]

(Entrada de estadísticas de resumen)


Resuelve un intervalo de confianza  $Z$ . Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.CBajo, stat.CAlto	Intervalo de confianza para una media de población desconocida
stat. $\bar{x}$	Media muestra de la secuencia de datos de la distribución aleatoria normal
stat.ME	Margen de error
stat.ex	Desviación estándar muestra
stat.n	Longitud de la secuencia de datos con media muestra

Variable de salida	Descripción
stat.σ	Desviación estándar de población conocida para la secuencia de datos <i>Lista</i>

### zInterval\_1Prop (intervaloZ\_1Prop)

Catálogo > 

**zInterval\_1Prop**  $x, n$  [*nivelC*]


Resuelve un intervalo de confianza  $Z$  de una proporción. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

$x$  es un entero no negativo.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.CBajo, stat.CAlto	Intervalo de confianza que contiene la probabilidad de distribución del nivel de confianza
stat. $\hat{p}$	La proporción de éxitos calculada
stat.ME	Margen de error
stat.n	Número de muestras en la secuencia de datos

### zInterval\_2Prop (intervaloZ\_2Prop)

Catálogo > 

**zInterval\_2Prop**  $x1, n1, x2, n2$  [*nivelC*]

Resuelve un intervalo de confianza  $Z$  de dos proporciones. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

$x1$  y  $x2$  son enteros no negativos.

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.CBajo, stat.CAlto	Intervalo de confianza que contiene la probabilidad de distribución del nivel de confianza
stat. $\hat{p}$ Dif	La diferencia entre proporciones calculada
stat.ME	Margen de error
stat. $\hat{p}1$	Estimación de proporción de primera muestra
stat. $\hat{p}2$	Estimación de proporción de segunda muestra
stat.n1	Tamaño de la muestra en una secuencia de datos
stat.n2	Tamaño de la muestra en la secuencia de datos de dos

### zInterval\_2Samp (intervaloZ\_2Muest)

Catálogo > 

**zInterval\_2Samp**  $\sigma_1, \sigma_2, Lista1, Lista2$   
[,Frec1[,Frec2[,nivelC]]]

(Entrada de lista de datos)

**zInterval\_2Samp**  $\sigma_1, \sigma_2, \bar{x}1, n1, \bar{x}2, n2$  [,nivelC]

(Entrada de estadísticas de resumen)


Resuelve un intervalo de confianza  $Z$  de dos muestras. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.CBajo, stat.CAlto	Intervalo de confianza que contiene la probabilidad de distribución del nivel de confianza
stat. $\bar{x}1 - \bar{x}2$	Medias muestra de las secuencias de datos de la distribución aleatoria normal
stat.ME	Margen de error
stat. $\bar{x}1$ , stat. $\bar{x}2$	Medias muestra de las secuencias de datos de la distribución aleatoria normal
stat. $\sigma x1$ , stat. $\sigma x2$	Desviaciones estándar muestras para <i>Lista 1</i> y <i>Lista 2</i>

Variable de salida	Descripción
stat.n1, stat.n2	Número de muestras en las secuencias de datos
stat.r1, stat.r2	Desviaciones estándar de población conocidas para <i>Lista 1</i> y <i>Lista 2</i>

## zTest (pruebaZ)

Catálogo > 

**zTest**  $\mu_0, \sigma, Lista, [Frec[, Hipot]]$

(Entrada de lista de datos)

**zTest**  $\mu_0, \sigma, \bar{x}, n[, Hipot]$

(Entrada de estadísticas de resumen)

Realiza una prueba  $z$  con frecuencia *listaFrec*. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Pruebe  $H_0: \mu = \mu_0$ , contra uno de los siguientes:

Para  $H_a: \mu < \mu_0$ , configure *Hipot*<0


Para  $H_a: \mu \neq \mu_0$  (predeterminado), configure *Hipot*=0

Para  $H_a: \mu > \mu_0$ , configure *Hipot*>0

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea "Elementos vacíos (inválidos)" (página 231).

Variable de salida	Descripción
stat.z	$(\bar{x} - \mu_0) / (\sigma / \text{sqrt}(n))$
stat.Valor P	Probabilidad más baja a la cual la hipótesis nula se puede rechazar
stat. $\bar{x}$	Media de muestra de la secuencia de datos en <i>Lista</i>
stat.ex	Desviación estándar de muestras de la secuencia de datos. Sólo se entrega para la entrada de <i>Datos</i> .
stat.n	Tamaño de la muestra

## zTest\_1Prop (pruebaZ\_1Prop)

Catálogo > 

**zTest\_1Prop**  $p_0, x, n[, Hipot]$

Resuelve una prueba  $Z$  de una proporción.  
Un resumen de resultados se almacena en la variable *stat.results* (página 160).

$x$  es un entero no negativo.

Pruebe  $H_0: p = p_0$  contra uno de los siguientes:

Para  $H_a: p > p_0$ , configure *Hipot*>0

Para  $H_a: p \neq p_0$  (predeterminado), configure *Hipot*=0

Para  $H_a: p < p_0$ , configure *Hipot*<0

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.p0	Proporción poblacional de la hipótesis
stat.z	Valor normal estándar calculado para la proporción
stat.ValP	Nivel más bajo de significancia en el cual la hipótesis nula se puede rechazar
stat. $\hat{p}$	Proporción muestral estimada
stat.n	Tamaño de la muestra

**zTest\_2Prop**  $x1, n1, x2, n2[, Hipot]$

Resuelve una prueba  $Z$  de dos proporciones.  
Un resumen de resultados se almacena en la variable *stat.results* (página 160).

$x1$  y  $x2$  son enteros no negativos.

Pruebe  $H_0: p1 = p2$ , contra uno de los siguientes:

Para  $H_a: p1 > p2$ , configure *Hipot*>0

Para  $H_a: p1 \neq p2$  (predeterminado), configure *Hipot*=0

Para  $H_a: p < p_0$ , configure *Hipot*<0



Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

Variable de salida	Descripción
stat.z	Valor normal estándar calculado para la diferencia de las proporciones
stat.ValP	Nivel más bajo de significancia en el cual la hipótesis nula se puede rechazar
stat. $\hat{p}1$	Estimación de proporción de primera muestra
stat. $\hat{p}2$	Estimación de proporción de segunda muestra
stat. $\hat{p}$	Estimación de proporción de muestras agrupadas
stat.n1, stat.n2	Número de muestras tomadas en las pruebas 1 y 2

## zTest\_2Samp (pruebaZ\_2Muest)

**zTest\_2Samp**  $\sigma_1, \sigma_2, Lista1, Lista2[, Frec1 [, Frec2[, Hipot]]]$

(Entrada de lista de datos)

**zTest\_2Samp**  $\sigma_1, \sigma_2, \bar{x}1, n1, \bar{x}2, n2[, Hipot]$

(Entrada de estadísticas de resumen)

Resuelve una prueba  $Z$  de dos muestras. Un resumen de resultados se almacena en la variable *stat.results* (página 160).

Pruebe  $H_0: \mu_1 = \mu_2$ , contra uno de los siguientes:

Para  $H_a: \mu_1 < \mu_2$ , configure *Hipot*<0

Para  $H_a: \mu_1 \neq \mu_2$  (predeterminado), configure *Hipot*=0

Para  $H_a: \mu_1 > \mu_2$ , *Hipot*>0

Para obtener información sobre el efecto de los elementos vacíos en una lista, vea “Elementos vacíos (inválidos)” (página 231).

<b>Variable de salida</b>	<b>Descripción</b>
stat.z	Valor normal estándar computado para la diferencia de las medias
stat.ValP	Nivel más bajo de significancia en el cual la hipótesis nula se puede rechazar
stat. $\bar{x}1$ , stat. $\bar{x}2$	Muestras de las medias de las secuencias de datos en <i>Lista1</i> y <i>Lista2</i>
stat.sx1, stat.sx2	Desviaciones estándar de muestras de las secuencias de datos en <i>Lista1</i> y <i>Lista2</i>
stat.n1, stat.n2	Tamaño de las muestras

# Símbolos

## + (agregar)

**+** tecla

$Valor1 + Valor2 \Rightarrow valor$

56	56
----	----

Entrega la suma de los dos argumentos.

$56+4$	60
--------	----

$60+4$	64
--------	----

$64+4$	68
--------	----

$68+4$	72
--------	----

$Lista1 + Lista2 \Rightarrow lista$

$\left\{22, \pi, \frac{\pi}{2}\right\} \rightarrow l1$	$\{22, 3.14159, 1.5708\}$
--	---------------------------

$Matriz1 + Matriz2 \Rightarrow matriz$

$\left\{10, 5, \frac{\pi}{2}\right\} \rightarrow l2$	$\{10, 5, 1.5708\}$
--	---------------------

Entrega una lista (o matriz) que contiene las sumas de los elementos correspondientes en *Lista1* y *Lista2* (o *Matriz1* y *Matriz2*).

$l1+l2$	$\{32, 8.14159, 3.14159\}$
---------	----------------------------

Las dimensiones de los argumentos deben ser iguales.

$Valor + Lista1 \Rightarrow lista$

$15 + \{10, 15, 20\}$	$\{25, 30, 35\}$
-----------------------	------------------

$Lista1 + Valor \Rightarrow lista$

$\{10, 15, 20\} + 15$	$\{25, 30, 35\}$
-----------------------	------------------

Entrega una lista que contiene las sumas de *Valor* y cada elemento en *Lista1*.

$Valor + Matriz1 \Rightarrow matriz$

$20 + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 21 & 2 \\ 3 & 24 \end{bmatrix}$
---	--

$Matriz1 + Valor \Rightarrow matriz$

Entrega una matriz con *Valor* agregado a cada elemento en la diagonal de *Matriz1*. *Matriz1* debe ser cuadrada.

**Nota:** Use .+ (punto más) para agregar una expresión a cada elemento.

## -(sustraer)

**-** tecla

$Valor1 - Valor2 \Rightarrow valor$

$6-2$	4
-------	---

Entrega *Valor1* menos *Valor2*.

$\pi - \frac{\pi}{6}$	2.61799
-----------------------	---------

**- (sustraer)** $Lista1 - Lista2 \Rightarrow lista$ 

$$\left\{ 22, \pi, \frac{\pi}{2} \right\} - \left\{ 10, 5, \frac{\pi}{2} \right\} \quad \left\{ 12, 1.85841, 0 \right\}$$

 $Matriz1 - Matriz2 \Rightarrow matriz$ 

$$\begin{bmatrix} 3 & 4 \end{bmatrix} - \begin{bmatrix} 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 2 & 2 \end{bmatrix}$$

Sustraer a cada elemento en *Lista2* (o *Matriz2*) del elemento correspondiente en *Lista1* (o *Matriz1*) y entrega los resultados.

Las dimensiones de los argumentos deben ser iguales.

 $Valor - Lista1 \Rightarrow lista$ 

$$15 - \{10, 15, 20\} \quad \{5, 0, -5\}$$

 $Lista1 - Valor \Rightarrow lista$ 

$$\{10, 15, 20\} - 15 \quad \{-5, 0, 5\}$$

Sustraer a cada elemento de *Lista1* de *Valor* o sustraer *Valor* de cada elemento de *Lista1* y entrega una lista con los resultados.

 $Valor - Matriz1 \Rightarrow matriz$ 

$$20 - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \begin{bmatrix} 19 & -2 \\ -3 & 16 \end{bmatrix}$$

 $Matriz1 - Valor \Rightarrow matriz$ 

$Valor - Matriz1$  entrega una matriz de *Valor* veces la matriz de identidad menos *Matriz1*. La *Matriz1* debe ser cuadrada.

$Matriz1 - Valor$  entrega una matriz de *Valor* veces la matriz de identidad sustraída de *Matriz1*. La *Matriz1* debe ser cuadrada.

**Nota:** Use .- (punto menos) para sustraer una expresión de cada elemento.

**· (multiplicar)** $Valor1 \cdot Valor2 \Rightarrow valor$ 

$$2 \cdot 3.45 \quad 6.9$$

Entrega el producto de los dos argumentos.

 $Lista1 \cdot Lista2 \Rightarrow lista$ 

$$\{1., 2, 3\} \cdot \{4, 5, 6\} \quad \{4, 10, 18\}$$

Entrega una lista que contiene los productos de los elementos correspondientes en *Lista1* y *Lista2*.

**· (multiplicar)** **tecla**

Las dimensiones de las listas deben ser iguales.

$Matriz1 \cdot Matriz2 \Rightarrow matriz$

Entrega el producto de la matriz de  $Matriz1$  y  $Matriz2$ .

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 7 & 8 \\ 7 & 8 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 42 & 48 \\ 105 & 120 \end{bmatrix}$$

El número de columnas en  $Matriz1$  debe igualar el número de filas en  $Matriz2$ .

$Valor \cdot Lista1 \Rightarrow lista$

$$\pi \cdot \{4,5,6\} = \{12.5664, 15.708, 18.8496\}$$

$Lista1 \cdot Valor \Rightarrow lista$

Entrega una lista que contiene los productos de  $Valor$  y cada elemento en  $Lista1$ .

$Valor \cdot Matriz1 \Rightarrow matriz$

$Matriz1 \cdot Valor \Rightarrow matriz$

Entrega una matriz que contiene los productos de  $Valor$  y cada elemento en  $Matriz1$ .

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 0.01 = \begin{bmatrix} 0.01 & 0.02 \\ 0.03 & 0.04 \end{bmatrix}$$

$$6 \cdot \text{identity}(3) = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

**Nota:** Use  $\cdot$  (punto multiplicar) para multiplicar una expresión por cada elemento.

**/ (dividir)** **tecla**

$Valor1 / Valor2 \Rightarrow valor$

Entrega el cociente de  $Valor1$  dividido entre  $Valor2$ .

$$\frac{2}{3.45} = 0.57971$$

**Nota:** Vea también **Plantilla de fracciones**, página 1.

$Lista1 / Lista2 \Rightarrow lista$

Entrega una lista que contiene los cocientes de  $Lista1$  divididos entre  $Lista2$ .

$$\frac{\{1., 2, 3\}}{\{4, 5, 6\}} = \left\{ 0.25, \frac{2}{5}, \frac{1}{2} \right\}$$

Las dimensiones de las listas deben ser iguales.

## / (dividir)



tecla

 $Valor / Lista1 \Rightarrow lista$ 

$$\frac{6}{\{3,6,\sqrt{6}\}} \quad \{2,1,2,44949\}$$

 $Lista1 / Valor \Rightarrow lista$ 

$$\frac{\{7,9,2\}}{7 \cdot 9 \cdot 2} \quad \left\{ \frac{1}{18}, \frac{1}{14}, \frac{1}{63} \right\}$$

Entrega una lista que contiene los cocientes de *Valor* divididos entre *Lista1* o de *Lista1* divididos entre *Valor*.

 $Valor / Matriz1 \Rightarrow matriz$ 

$$\frac{\begin{bmatrix} 7 & 9 & 2 \end{bmatrix}}{7 \cdot 9 \cdot 2} \quad \begin{bmatrix} \frac{1}{18} & \frac{1}{14} & \frac{1}{63} \end{bmatrix}$$

 $Matriz1 / Valor \Rightarrow matriz$ 

Entrega una matriz que contiene los cocientes de *Matriz1/Valor*.

**Nota:** Use . / (punto dividir) para dividir una expresión entre cada elemento.

## ^ (potencia)



tecla

 $Valor1 \wedge Valor2 \Rightarrow valor$ 

$$4^2 \quad 16$$

 $Lista1 \wedge Lista2 \Rightarrow lista$ 

$$\{2,4,6\} \{1,2,3\} \quad \{2,16,216\}$$

Entrega el primer argumento elevado a la potencia del segundo argumento.

**Nota:** Vea también **Plantilla de exponentes**, página 1.

Para una lista, entrega los elementos en *Lista1* elevados a la potencia de los elementos correspondientes en *Lista2*.

En el dominio real, las potencias fraccionarias que han reducido los exponentes con denominadores impares usan la rama real contra la rama principal para el modo complejo.

 $Valor \wedge Lista1 \Rightarrow lista$ 

$$\pi \{1,2,3\} \quad \{3.14159,9.8696,0.032252\}$$

Entrega *Valor* elevado a la potencia de los elementos en *Lista1*.

 $Lista1 \wedge Valor \Rightarrow lista$ 

$$\{1,2,3,4\}^{-2} \quad \left\{ 1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16} \right\}$$

Entrega los elementos en *Lista1* elevados a la potencia de *Valor*.

**^ (potencia)** **tecla**

*matrizCuadrada1* ^ entero  $\Rightarrow$  *matriz*

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^2$	$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$
--	---

Entrega *matrizCuadrada1* elevada a la potencia del entero .

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1}$	$\begin{bmatrix} -2 & 1 \\ 3 & -1 \\ 2 & 2 \end{bmatrix}$
---	---

*matrizCuadrada1* debe ser una matriz cuadrada.

Si entero = -1, resuelve la matriz inversa.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-2}$	$\begin{bmatrix} 11 & -5 \\ 2 & 2 \\ -15 & 7 \\ 4 & 4 \end{bmatrix}$
---	--

Si entero < -1, resuelve la matriz inversa a una potencia positiva apropiada.

**x2 (cuadrado)** **tecla**

*Valor1*<sup>2</sup>  $\Rightarrow$  *valor*

$4^2$	16
-------	----

Entrega el cuadrado del argumento.

$\{2,4,6\}^2$	$\{4,16,36\}$
---------------	---------------

*Lista1*<sup>2</sup>  $\Rightarrow$  *lista*

$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}^2$	$\begin{bmatrix} 40 & 64 & 88 \\ 49 & 79 & 109 \\ 58 & 94 & 130 \end{bmatrix}$
---	--

Entrega una lista que contiene los cuadrados de los elementos en la *Lista1*.

*matrizCuadrada1*<sup>2</sup>  $\Rightarrow$  *matriz*

$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}^{\wedge 2}$	$\begin{bmatrix} 4 & 16 & 36 \\ 9 & 25 & 49 \\ 16 & 36 & 64 \end{bmatrix}$
--	--

Entrega el cuadrado de la matriz de *matrizCuadrada1*. Esto no es lo mismo que calcular el cuadrado de cada elemento. Use .^2 para calcular el cuadrado de cada elemento.

**.+ (punto agregar)** **teclas**

*Matriz1* .+ *Matriz2*  $\Rightarrow$  *matriz*


$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} .+ \begin{bmatrix} 10 & 30 \\ 20 & 40 \end{bmatrix}$	$\begin{bmatrix} 11 & 32 \\ 23 & 44 \end{bmatrix}$
--	--

*Valor* .+ *Matriz1*  $\Rightarrow$  *matriz*

$5 .+ \begin{bmatrix} 10 & 30 \\ 20 & 40 \end{bmatrix}$	$\begin{bmatrix} 15 & 35 \\ 25 & 45 \end{bmatrix}$
---	--

*Matriz1* .+ *Matriz2* entrega una matriz que es la suma de cada par de elementos correspondientes en *Matriz1* y *Matriz2*.

*Valor* .+ *Matriz1* entrega una matriz que es la suma de *Valor* y cada elemento en *Matriz1*.

**.- (punto sust.)**  **teclas***Matriz1* .- *Matriz2*  $\Rightarrow$  *matriz*

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} .- \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} \quad \begin{bmatrix} -9 & -18 \\ -27 & -36 \end{bmatrix}$$

*Valor* .- *Matriz1*  $\Rightarrow$  *matriz*

$$5 .- \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} \quad \begin{bmatrix} -5 & -15 \\ -25 & -35 \end{bmatrix}$$

*Matriz1* .- *Matriz2* entrega una matriz que es la diferencia entre cada para de elementos correspondientes en *Matriz1* y *Matriz2*.

Valor .- *Matriz1* entrega una matriz que es la diferencia de *Valor* y cada elemento en *Matriz1*.

**.· (punto mult.)**  **teclas***Matriz1* .· *Matriz2*  $\Rightarrow$  *matriz*

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} .· \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} \quad \begin{bmatrix} 10 & 40 \\ 90 & 160 \end{bmatrix}$$

*Valor* .· *Matriz1*  $\Rightarrow$  *matriz*

$$5 .· \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} \quad \begin{bmatrix} 50 & 100 \\ 150 & 200 \end{bmatrix}$$

*Matriz1* .· *Matriz2* entrega una matriz que es el producto de cada par de elementos correspondientes en *Matriz1* y *Matriz2*.

Valor .· *Matriz1* entrega una matriz que contiene los productos de *Valor* y cada elemento en *Matriz1*.

**./ (punto dividir)**  **teclas***Matriz1* ./ *Matriz2*  $\Rightarrow$  *matriz*

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} ./ \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} \quad \begin{bmatrix} \frac{1}{10} & \frac{1}{10} \\ \frac{1}{10} & \frac{1}{10} \end{bmatrix}$$

*Valor* ./ *Matriz1*  $\Rightarrow$  *matriz*

$$5 ./ \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} \quad \begin{bmatrix} \frac{1}{10} & \frac{1}{10} \\ \frac{1}{2} & \frac{1}{4} \\ \frac{1}{6} & \frac{1}{8} \end{bmatrix}$$

*Matriz1* ./ *Matriz2* entrega una matriz que es el cociente de cada par de elementos correspondientes en *Matriz1* y *Matriz2*.

Valor ./ *Matriz1* entrega una matriz que es el cociente de *Valor* y cada elemento en *Matriz1*.





**= (igual)****= teclas** $Expr1 = Expr2 \Rightarrow$  expresión Booleana $Lista1 = Lista2 \Rightarrow$  lista Booleana $Matriz1 = Matriz2 \Rightarrow$  matriz BooleanaEntrega verdadero si  $Expr1$  se determina como igual a  $Expr2$ .Entrega falso si  $Expr1$  se determina como no igual a  $Expr2$ .

Cualquier otra cosa entrega una forma simplificada de la ecuación.

Para listas y matrices, entrega comparaciones elemento por elemento.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

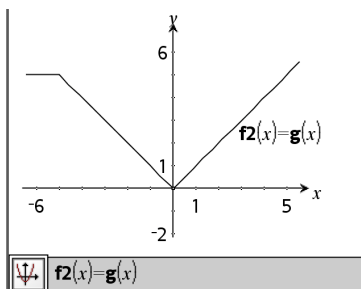
Ejemplo de función que usa símbolos de prueba matemática: =, ≠, &lt;, &gt;, ≤, ≥

```

Define g(x)=Func
  If x≤-5 Then
    Return 5
  ElseIf x>-5 and x<0 Then
    Return -x
  ElseIf x≥0 and x≠10 Then
    Return x
  ElseIf x=10 Then
    Return 3
  EndIf
EndFunc

```


Done

Resultado de graficar  $g(x)$ **≠ (no igual)****ctrl = teclas** $Expr1 \neq Expr2 \Rightarrow$  expresión Booleana $Lista1 \neq Lista2 \Rightarrow$  lista Booleana $Matriz1 \neq Matriz2 \Rightarrow$  matriz BooleanaEntrega verdadero si  $Expr1$  se determina como no igual a  $Expr2$ .Entrega si  $Expr1$  se determina como igual a  $Expr2$ .

Cualquier otra cosa entrega una forma simplificada de la ecuación.

Vea "=" (igual) ejemplo.

## $\neq$ (no igual)

  teclas

Para listas y matrices, entrega comparaciones elemento por elemento.

**Nota:** Usted puede insertar este operador desde el teclado al escribir  $\neq$

## $<$ (menor que)

  teclas

$Expr1 < Expr2 \Rightarrow \text{expresión Booleana}$

Vea "=" (igual) ejemplo.

$Lista1 < Lista2 \Rightarrow \text{lista Booleana}$

$Matriz1 < Matriz2 \Rightarrow \text{matriz Booleana}$



Entrega verdadero si  $Expr1$  se determina como menor que  $Expr2$ .

Entrega falso si  $Expr1$  se determina como mayor que o igual a  $Expr2$ .

Cualquier otra cosa entrega una forma simplificada de la ecuación.

Para listas y matrices, entrega comparaciones elemento por elemento.

## $\leq$ (menor o igual)

  teclas

$Expr1 \leq Expr2 \Rightarrow \text{expresión Booleana}$

Vea "=" (igual) ejemplo.

$Lista1 \leq Lista2 \Rightarrow \text{lista Booleana}$

$Matriz1 \leq Matriz2 \Rightarrow \text{matriz Booleana}$

Entrega verdadero si  $Expr1$  se determina como menor que o igual a  $Expr2$ .

Entrega falso si  $Expr1$  se determina como mayor que  $Expr2$ .

Cualquier otra cosa entrega una forma simplificada de la ecuación.

Para listas y matrices, entrega comparaciones elemento por elemento.

**Nota:** Usted puede insertar este operador desde el teclado al escribir  $\leq$

## > (mayor que)

ctrl = teclas

$Expr1 > Expr2 \Rightarrow$  expresión Booleana

Vea "=" (igual) ejemplo.

$Lista1 > Lista2 \Rightarrow$  lista Booleana

$Matriz1 > Matriz2 \Rightarrow$  matriz Booleana

Entrega verdadero si  $Expr1$  se determina como mayor que  $Expr2$ .

Entrega falso si  $Expr1$  se determina como menor que o igual a  $Expr2$ .

Cualquier otra cosa entrega una forma simplificada de la ecuación.

Para listas y matrices, entrega comparaciones elemento por elemento.

## $\geq$ (mayor o igual)

ctrl = teclas

$Expr1 \geq Expr2 \Rightarrow$  expresión Booleana

Vea "=" (igual) ejemplo.

$Lista1 \geq Lista2 \Rightarrow$  lista Booleana

$Matriz1 \geq Matriz2 \Rightarrow$  matriz Booleana

Entrega verdadero si  $Expr1$  se determina como mayor que o igual a  $Expr2$ .

Entrega falso si  $Expr1$  se determina como menor que  $Expr2$ .

Cualquier otra cosa entrega una forma simplificada de la ecuación.

Para listas y matrices, entrega comparaciones elemento por elemento.

**Nota:** Usted puede insertar este operador desde el teclado al escribir  $\geq$

**⇒ (implicación lógica)**teclas **ctrl** **=***BooleanaExpr1 ⇒ BooleanaExpr2*  
devuelve *expresión booleana*

5&gt;3 or 3&gt;5 true

*BooleanaLista1 ⇒ BooleanaLista2*  
devuelve *lista booleana*

5&gt;3 ⇒ 3&gt;5 false

*BooleanaMatriz1 ⇒ BooleanaMatriz2*  
devuelve *matriz booleana*

3 or 4 7

3 ⇒ 4 -4

 $\{1,2,3\}$  or  $\{3,2,1\}$   $\{3,2,3\}$  $\{1,2,3\}$  ⇒  $\{3,2,1\}$   $\{-1,-1,-3\}$ *Entero1 ⇒ Entero2* devuelve *Entero*

Evalúa la expresión **not <argumento1> or <argumento2>** y devuelve verdadero, falso o una forma simplificada de la ecuación.

Para listas y matrices, devuelve comparaciones elemento por elemento.

**Nota:** Puede insertar este operador con el teclado al escribir =>

**⇔ (implicación doble lógica, XNOR)**teclas **=** **ctrl***BooleanaExpr1 ⇔ BooleanaExpr2*  
devuelve *expresión booleana*

5&gt;3 xor 3&gt;5 true

*BooleanaLista1 ⇔ BooleanaLista2*  
devuelve *lista booleana*

5&gt;3 ⇔ 3&gt;5 false

*BooleanaMatriz1 ⇔ BooleanaMatriz2*  
devuelve *matriz booleana*

3 xor 4 7

3 ⇔ 4 -8

 $\{1,2,3\}$  xor  $\{3,2,1\}$   $\{2,0,2\}$  $\{1,2,3\}$  ⇔  $\{3,2,1\}$   $\{-3,-1,-3\}$ *Entero1 ⇔ Entero2* devuelve *Entero*

Devuelve la negación de una **XOR** operación booleana en los dos argumentos. Devuelve verdadero, falso o una forma simplificada de la ecuación.

Para listas y matrices, devuelve comparaciones elemento por elemento.

**Nota:** Puede insertar este operador con el teclado al escribir <=>

## ! (factorial)

 **tecla**

*Valor1!* ⇒ *valor*

5! 120

*Lista1!* ⇒ *lista*

$\{\{5,4,3\}\}!$   $\{120,24,6\}$

*Matriz1!* ⇒ *matriz*

$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}!$   $\begin{bmatrix} 1 & 2 \\ 6 & 24 \end{bmatrix}$

Entrega el factorial del argumento.

Para una lista o matriz, entrega una lista o una matriz de factoriales de los elementos.

## & (adjuntar)

  **teclas**

*Cadena1 & Cadena2* ⇒ *cadena*

"Hello "&"Nick"

"Hello Nick"

Entrega una cadena de texto que es *Cadena2* adjuntada a *Cadena1*.

## d() (derivada)

**Catálogo** > 

**d**(*Expr1*, *Var*[, *Orden*]) |  
*Var*=*Valor*⇒*valor*

$\frac{d}{dx}(|x|)|_{x=0}$  undef

**d**(*Expr1*, *Var*[, *Orden*])⇒*valor*

$x:=0: \frac{d}{dx}(|x|)$  undef

**d**(*Lista1*, *Var*[, *Orden*])⇒*lista*

$x:=3: \frac{d}{dx}(\{x^2, x^3, x^4\})$   $\{6, 27, 108\}$

**d**(*Matriz1*, *Var*[, *Orden*])⇒*matriz*

Excepto cuando se usa la primera sintaxis, usted debe almacenar un valor numérico en la variable *Var* antes de evaluar **d()**. Consulte los ejemplos.


**d()** se puede usar para calcular la derivada de primer y segundo orden numéricamente en un punto, usando métodos de autodiferenciación.

*Orden*, si se incluye, debe ser **1** ó **2**. El predeterminado es **1**.

**Nota:** Usted puede insertar esta función desde el teclado al escribir **derivative** (...).

**Nota:** Vea también **Primera derivada**, página 5 o **Segunda derivada**, página 6.

## d() (derivada)

Catálogo > 


**Nota:** El algoritmo d() tiene una limitación: funciona recursivamente a través de la expresión no simplificada, determinando el valor numérico de la primera derivada (y de la segunda, si aplica) y la evaluación de cada subexpresión, lo que puede conllevar a un resultado inesperado.

Tome en consideración el ejemplo de la derecha. La primera derivada de  $x \cdot (x^2+x)^{1/3}$  en  $x=0$  es igual a 0. Sin embargo, dado que la primera derivada de la subexpresión  $(x^2+x)^{1/3}$  es indefinida en  $x=0$ , y este valor se usa para calcular la derivada de la expresión total, **d()** reporta el resultado como indefinido y despliega un mensaje de advertencia.

Si usted encuentra esta limitación, verifique la solución en forma gráfica. Usted también puede tratar de usar **centralDiff()**.

$\frac{d}{dx} \left( x \cdot (x^2+x)^{\frac{1}{3}} \right) \Big _{x=0}$	undef
$\text{centralDiff} \left( x \cdot (x^2+x)^{\frac{1}{3}}, x \right) \Big _{x=0}$	0.000033

## f() (integral)

Catálogo > 

$f(\text{Expr1}, \text{Var}, \text{Baja}, \text{Alta}) \Rightarrow \text{valor}$



Entrega la integral de *Expr1* con respecto de la variable *Var* de *Baja* a *Alta*. Se puede usar para calcular la integral definida numéricamente, usando el mismo método que con **nInt()**.

**Nota:** Usted puede insertar esta función desde el teclado al escribir **integral (...)**.

**Nota:** Vea también **nInt()**, página 110y **Plantilla de integral definida**, página 6.

$\int_0^1 x^2 dx$	0.333333
-------------------	----------

## √() (raíz cuadrada)

  **teclas**

$\sqrt{(\text{Valor1})} \Rightarrow \text{valor}$

$\sqrt{(\text{Lista1})} \Rightarrow \text{lista}$

Entrega la raíz cuadrada del argumento.

$\sqrt{4}$	2
$\sqrt{\{9,2,4\}}$	{3,1.41421,2}

## $\sqrt{()}$ (raíz cuadrada)

ctrl x<sup>2</sup> teclas

Para una lista, entrega las raíces cuadradas de todos los elementos en *Lista1*.

**Nota:** Usted puede insertar esta función desde el teclado al escribir `sqrt (...)`.

**Nota:** Vea también **Plantilla de raíz cuadrada**, página 1.

## $\prod()$ (secProd)

Catálogo >

$\prod(\text{Expr1}, \text{Var}, \text{Baja}, \text{Alta}) \Rightarrow \text{expresión}$

**Nota:** Usted puede insertar esta función desde el teclado al escribir `prodSeq (...)`.

Evalúa *Expr1* para cada valor de *Var* de *Baja* a *Alta* y entrega el producto de los resultados.

**Nota:** Vea también **Plantilla de producto** ( $\prod$ ), página 5.

$\prod(\text{Expr1}, \text{Var}, \text{Baja}, \text{Baja}-1) \Rightarrow 1$

$\prod(\text{Expr1}, \text{Var}, \text{Baja}, \text{Alta}) \Rightarrow 1/\prod(\text{Expr1}, \text{Var}, \text{Alta}+1, \text{Baja}-1)$  if  $\text{Alta} < \text{Baja}-1$

Las fórmulas del producto utilizadas se derivan de la siguiente referencia:

Ronald L. Graham, Donald E. Knuth y Oren Patashnik. *Matemáticas Concretas: Una Fundación para las Ciencias de la Computación*. Lectura, Massachusetts: Addison-Wesley, 1994.

$$\prod_{n=1}^5 \left(\frac{1}{n}\right) = \frac{1}{120}$$

$$\prod_{n=1}^5 \left(\left\{\frac{1}{n}, n, 2\right\}\right) = \left\{\frac{1}{120}, 120, 32\right\}$$

$$\prod_{k=4}^3 (k) = 1$$

$$\prod_{k=4}^1 \left(\frac{1}{k}\right) = 6$$

$$\prod_{k=4}^1 \left(\frac{1}{k}\right) \cdot \prod_{k=2}^4 \left(\frac{1}{k}\right) = \frac{1}{4}$$

## $\Sigma()$ (secSuma)

Catálogo >

$\Sigma(\text{Expr1}, \text{Var}, \text{Baja}, \text{Alta}) \Rightarrow \text{expresión}$

**Nota:** Usted puede insertar esta función desde el teclado al escribir `secSuma (...)`.

$$\sum_{n=1}^5 \left(\frac{1}{n}\right) = \frac{137}{60}$$



Evalúa *Expr1* para cada valor de *Var* de *Baja* a *Alta* y entrega la suma de los resultados.

**Nota:** Vea también **Plantilla de suma**, página 5.

$\Sigma(\text{Expr1}, \text{Var}, \text{Baja}, \text{Alta}-1) \Rightarrow 0$

$\Sigma(\text{Expr1}, \text{Var}, \text{Baja}, \text{Alta}) \Rightarrow -\Sigma(\text{Expr1}, \text{Var}, \text{Alta}+1, \text{Baja}-1)$  si  $\text{Alta} < \text{Baja}-1$

$$\sum_{k=4}^3 (k) \quad 0$$

Las fórmulas de la sumatoria utilizadas se derivan de la siguiente referencia:

Ronald L. Graham, Donald E. Knuth y Oren Patashnik. *Matemáticas Concretas: Una Fundación para las Ciencias de la Computación*. Lectura, Massachusetts: Addison-Wesley, 1994.

$$\sum_{k=4}^1 (k) \quad -5$$

$$\sum_{k=4}^1 (k) + \sum_{k=2}^4 (k) \quad 4$$

 $\Sigma\text{Int}()$ 

$\Sigma\text{Int}(\text{NPgo1}, \text{NPgo2}, \text{N}, \text{I}, \text{VP}, [\text{Pgo}], [\text{VF}], [\text{PpA}], [\text{CpA}], [\text{PgoAl}], [\text{valorRedondo}]) \Rightarrow \text{valor}$

$$\Sigma\text{Int}(1,3,12,4,75,20000,,12,12) \quad -213.48$$

$\Sigma\text{Int}(\text{NPgo1}, \text{NPgo2}, \text{tablaAmort}) \Rightarrow \text{valor}$

$\text{tbl} := \text{amortTbl}(12,12,4,75,20000,,12,12)$

La función de amortización que calcula la suma del interés durante un rango de pagos específico.

*NPgo1* y *NPgo2* definen los límites iniciales y finales del rango de pagos.

*N*, *I*, *VP*, *Pgo*, *VF*, *PpA*, *CpA* y *PgoAl* se describen en la tabla de argumentos de VTD, página 176.

- Si se omite *Pgo*, se predetermina a  $Pgo = \text{tvmPmt}(N, I, VP, VF, PpA, CpA, PgoAl)$ .
- Si se omite *VF*, se predetermina a  $VF = 0$ .
- Los predeterminados para *PpA*, *CpA* y *PgoAl* son los mismos que para las funciones de VTD.

0	0.	0.	20000.
1	-77.49	-1632.43	18367.6
2	-71.17	-1638.75	16728.8
3	-64.82	-1645.1	15083.7
4	-58.44	-1651.48	13432.2
5	-52.05	-1657.87	11774.4
6	-45.62	-1664.3	10110.1
7	-39.17	-1670.75	8439.32
8	-32.7	-1677.22	6762.1
9	-26.2	-1683.72	5078.38
10	-19.68	-1690.24	3388.14
11	-13.13	-1696.79	1691.35
12	-6.55	-1703.37	-12.02

$$\Sigma\text{Int}(1,3,\text{tbl}) \quad -213.48$$

*valorRedondo* especifica el número de lugares decimales para el redondeo. Predeterminado=2.

$\Sigma\text{Int}(NPgo1, NPgo2, tablaAmort)$  calcula la suma del interés con base en la tabla de amortización *tablaAmort*. El argumento *tablaAmort* debe ser una matriz en la forma descrita bajo **amortTbl()**, página 7.

**Nota:** Vea también  $\Sigma\text{Prn}()$ , abajo y **Bal()**, página 16.

 $\Sigma\text{Prn}()$  ( $\Sigma\text{Cap}$ )

$\Sigma\text{Prn}(NPgo1, NPgo2, N, I, VP, [Pgo], [VF], [PpA], [CpA], [PgoAl], [valorRedondo]) \Rightarrow valor$

$\Sigma\text{Prn}(1,3,12,4,75,20000,,12,12)$	-4916.28
--	----------

 $\Sigma\text{Prn}$ 

$(NPgo1, NPgo2, tablaAmort) \Rightarrow valor$

La función de amortización que calcula la suma del capital durante un rango de pagos específico.

*NPgo1* y *NPgo2* definen los límites iniciales y finales del rango de pagos.

*N, I, VP, Pgo, VF, PpA, CpA, PgoAl* se describen en la tabla de argumentos de VTD, página 176.

- Si se omite *Pgo*, se predetermina a  $Pgo = \text{tvmPmt}(N, I, VP, VF, PpA, CpA, PgoAl)$ .
- Si se omite *VF*, se predetermina a  $VF = 0$ .
- Los predeterminados para *PpA, CpA, PgoAl* son los mismos que para las funciones de VTD.

*valorRedondo* especifica el número de lugares decimales para el redondeo. Predeterminado=2.

$tbl := \text{amortTbl}(12, 12, 4, 75, 20000, , 12, 12)$			
	0.	0.	20000.
1	-77.49	-1632.43	18367.57
2	-71.17	-1638.75	16728.82
3	-64.82	-1645.1	15083.72
4	-58.44	-1651.48	13432.24
5	-52.05	-1657.87	11774.37
6	-45.62	-1664.3	10110.07
7	-39.17	-1670.75	8439.32
8	-32.7	-1677.22	6762.1
9	-26.2	-1683.72	5078.38
10	-19.68	-1690.24	3388.14
11	-13.13	-1696.79	1691.35
12	-6.55	-1703.37	-12.02
$\Sigma\text{Prn}(1,3,tbl)$			
			-4916.28

ΣPrn(*NPgo1*,*NPgo2*,*tablaAmort*) calcula la suma del interés con base en la tabla de amortización *tablaAmort*. El argumento *tablaAmort* debe ser una matriz en la forma descrita bajo **amortTbl()**, página 7.

**Nota:** Vea también ΣInt(), arriba y Bal(), página 16.

## # (indirección)

ctrl  teclas# *cadenaNomVar*

Se refiere a la variable cuyo nombre es *cadenaNomVar*. Esto le permite usar cadenas para crear nombres de variable dentro de una función.

<i>xyz:=12</i>	12
<i>#("x"&amp;"y"&amp;"z")</i>	12

Crea o se refiere a la variable *xyz*.

<i>10→r</i>	10
<i>"r"→s1</i>	"r"
<i>#s1</i>	10

Entrega el valor de la variable (*r*) cuyo nombre se almacena en la variable *s1*.

## E (notación científica)

 tecla*mantisaEexponente*

Ingresa un número en la notación científica. El número se interpreta como *mantisa* × 10<sup>*exponente*</sup>.

23000.	23000.
2300000000.+4.1E15	4.1E15
3·10 <sup>4</sup>	30000

Sugerencia: Si usted desea ingresar una potencia de 10 sin causar un resultado de valor decimal, use 10<sup>entero</sup>.

**Nota:** Usted puede insertar este operador desde el teclado de la computadora al escribir @E. Por ejemplo, escriba 2.3@E4 para ingresar 2.3E4.

## g (gradián)

1 tecla

*Expr1g⇒expresión*

En modo de Grados, Gradianes o Radianes:

**g (gradián)****1** tecla*Lista*  $1\text{g} \Rightarrow$  *lista*

$\cos(50^{\text{g}})$	0.707107
-----------------------	----------

*Matriz*  $1\text{g} \Rightarrow$  *matriz*

$\cos(\{0,100^{\text{g}},200^{\text{g}}\})$	$\{1,0.,-1.\}$
---	----------------

Esta función le proporciona una manera de especificar un ángulo en gradianes mientras está en el modo de Grados o Radianes.

En el modo de ángulo en Radianes, multiplica *Expr1* por  $\pi/200$ .

En el modo de ángulo en Grados, multiplica *Expr1* por  $g/100$ .

En el modo de Gradianes, entrega *Expr1* sin cambios.

**Nota:** Usted puede insertar este símbolo desde el teclado de la computadora al escribir @g.

**r (radián)****1** tecla*Valor*  $1^{\text{r}} \Rightarrow$  *valor*

En modo de ángulo en Grados, Gradianes o Radianes:

*Lista*  $1^{\text{r}} \Rightarrow$  *lista*

$\cos\left(\frac{\pi}{4^{\text{r}}}\right)$	0.707107
---	----------

*Matriz*  $1^{\text{r}} \Rightarrow$  *matriz*

$\cos\left(\left\{0^{\text{r}},\left(\frac{\pi}{12}\right)^{\text{r}},-(\pi)^{\text{r}}\right\}\right)$	$\{1,0.965926,-1.\}$
---	----------------------

Esta función le proporciona una manera de especificar un ángulo en radianes mientras está en el modo de Grados o Gradianes.

En el modo de ángulo en Grados, multiplica el argumento por  $180/\pi$ .

En el modo de ángulo en Radianes, entrega el argumento sin cambios.

En el modo de Gradianes, multiplica el argumento por  $200/\pi$ .

Sugerencia: Use  $r$  si usted desea forzar los radianes en una definición de función independientemente del modo que prevalece cuando se usa la función.

## r (radián)

1 tecla

**Nota:** Usted puede insertar este símbolo desde el teclado de la computadora al escribir @r.

## ° (grado)

1 tecla

*Valor*1°⇒*valor*

*Lista*1°⇒*lista*

*Matriz*1°⇒*matriz*

Esta función le proporciona una manera de especificar un ángulo en grados mientras está en el modo de Gradianes o Radianes.

En el modo de ángulo en Radianes, multiplica el argumento por  $\pi/180$ .

En el modo de ángulo en Grados, entrega el argumento sin cambios.

En el modo de ángulo en Gradianes, multiplica el argumento por 10/9.

**Nota:** Usted puede insertar este símbolo desde el teclado de la computadora al escribir @d.

En modo de ángulo en Grados, Gradianes o Radianes:

$\cos(45^\circ)$	0.707107
------------------	----------

En modo de ángulo en Radianes:

## °, ', " (grado/minuto/segundo)

ctrl teclas

*gg*°*mm*'*ss*.*ss*"⇒*expresión*

*gg*Un número positivo o negativo

*mm*Un número no negativo

*ss.ss*Un número no negativo

Entrega  $gg+(mm/60)+(ss.ss/3600)$ .

Este formato de ingreso de base-60 le permite:

- Ingresar un ángulo en grados/minutos/segundos sin importar le modo de ángulo actual.
- Ingrese el tiempo como

En modo de ángulo en Grados:

25°13'17.5"	25.2215
25°30'	$\frac{51}{2}$

## ° , ' , " (grado/minuto/segundo)

  **teclas**

horas/minutos/segundos.

**Nota:** Siga ss.ss con dos apóstrofes ("), no con el símbolo de comillas (").

## ∠ (ángulo)

  **teclas**

[Radio,∠θ\_Ángulo]⇒vector

(entrada polar)

[Radio,∠θ\_Ángulo,Z\_  
Coordenada]⇒vector

(entrada cilíndrica)

[Radio,∠θ\_Ángulo,∠θ\_  
Ángulo]⇒vector

(entrada esférica)

Entrega las coordenadas como un vector dependiendo de la configuración del modo del Formato del Vector: rectangular, cilíndrica o esférica.

**Nota:** Usted puede insertar este símbolo desde el teclado de la computadora al escribir @<.

(Magnitud ∠ Ángulo)⇒valorComplejo

(entrada polar)

Ingresa un valor complejo en la forma polar ( $r∠\theta$ ). El *Ángulo* se interpreta de acuerdo con la configuración del modo del *Ángulo* actual.

En el modo de Radianes y en el formato del vector configure a:

rectangular

[5 ∠60° ∠45°]  
[1.76777 3.06186 3.53553]

cilíndrico

[5 ∠60° ∠45°]  
[3.53553 1.0472 3.53553]

esférico

[5 ∠60° ∠45°]  
[5. 1.0472 0.785398]


En el modo de ángulo en Radianes y el formato complejo Rectangular:

$5+3 \cdot i \left( 10 \angle \frac{\pi}{4} \right) \quad -2.07107-4.07107 \cdot i$

\_ (guión bajo como un elemento vacío)

**Vea "Elementos vacíos (inválidos)", página 231.**

## 10^()

**Catálogo >** 

10^ (Valor I)⇒valor

10<sup>1.5</sup>

31.6228

$10^{\wedge}(Lista1) \Rightarrow lista$

Entrega 10 elevado a la potencia del argumento.

Para una lista, entrega 10 elevado a la potencia de los elementos en *Lista1*.

$10^{\wedge}$

$(matrizCuadrada1) \Rightarrow matrizCuadrada$

Entrega 10 elevado a la potencia de *matrizCuadrada1*. Esto no es lo mismo que calcular 10 elevado a la potencia de cada elemento. Para obtener información acerca del método de cálculo, consulte **cos()**.

*matrizCuadrada1* debe ser diagonalizable. El resultado siempre contiene números de punto flotante.

$$10^{\left[ \begin{array}{ccc} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{array} \right]}$$

$$\left[ \begin{array}{ccc} 1.14336\text{E}7 & 8.17155\text{E}6 & 6.67589\text{E}6 \\ 9.95651\text{E}6 & 7.11587\text{E}6 & 5.81342\text{E}6 \\ 7.65298\text{E}6 & 5.46952\text{E}6 & 4.46845\text{E}6 \end{array} \right]$$

## $\wedge^{-1}$ (recíproco)

$Valor1 \wedge^{-1} \Rightarrow valor$

$$(3.1)^{-1} \quad 0.322581$$

$Lista1 \wedge^{-1} \Rightarrow lista$

Entrega el recíproco del argumento.

Para una lista, entrega los recíprocos de los elementos en *Lista1*.

$matrizCuadrada1 \wedge^{-1} \Rightarrow matrizCuadrada$

Entrega el inverso de *matrizCuadrada*.

*matrizCuadrada1* debe ser una matriz cuadrada no singular.

$$\left[ \begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \right]^{-1} \quad \left[ \begin{array}{cc} -2 & 1 \\ 3 & -1 \\ 2 & 2 \end{array} \right]$$

## | (operador restrictivo)

$Expr | BooleanaExpr1$   
 $[andBooleanaExpr2]...$

$$x+1|x=3 \quad 4$$

$Expr | BooleanaExpr1$   
 $[orBooleanaExpr2]...$

$$x+55|x=\sin(55) \quad 54.0002$$

El símbolo de restricción ("|") funciona como un operador binario. El operando a la izquierda de | es una expresión. El operando a la derecha de | especifica una o más relaciones que deben afectar la simplificación de la expresión. Las relaciones múltiples luego de | deben estar unidas por "and" lógica u operadores "or".

El operador restrictivo proporciona tres funciones básicas:

- Sustituciones
- Restricciones de intervalos
- Exclusiones

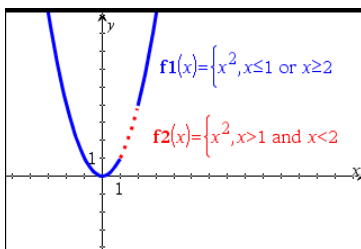
Las sustituciones tienen la forma de una igualdad, tal como  $x=3$  o  $y=\sin(x)$ . Para ser más efectiva, el lado izquierdo debe ser una variable simple. *Expr | Variable = el valor* sustituirá el valor para cada ocurrencia de la Variable en la Expr.

Las restricciones de intervalo tienen la forma de una o más desigualdades unidas por "and" lógica u operadores "or". Las restricciones de intervalo también permite la simplificación que de otro modo sería inválida o no computable.

Las exclusiones utilizan el operador relacional "distinto" ( $\neq$  o  $\neq$ ) para no tener en cuenta un valor específico.

$x^3-2\cdot x+7\rightarrow f(x)$	Done
$f(x) x=\sqrt{3}$	8.73205

$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0,x)$	0.
$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0,x) x>0 \text{ and } x<5$	3.





**→ (almacenar)**

ctrl var tecla

*Valor* → *Var**Lista* → *Var**Matriz* → *Var**Expr* → *Función(Parám1,...)**Lista* → *Función(Parám1,...)**Matriz* → *Función(Parám1,...)*

Si la variable *Var* no existe, la crea y la inicializa para *Valor*, *Listao Matriz*.

Si la variable *Var* ya existe y no está bloqueada o protegida, reemplaza sus contenidos con *Valor*, *Listao Matriz*.

**Nota:** Usted puede insertar este operador desde el teclado al escribir =: como un acceso directo. Por ejemplo, escriba `pi/4=: myvar`.

$\frac{\pi}{4} \rightarrow myvar$	0.785398
$2 \cdot \cos(x) \rightarrow yI(x)$	Done
$\{1,2,3,4\} \rightarrow lst5$	$\{1,2,3,4\}$
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow matg$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
"Hello" → <i>str1</i>	"Hello"

**:= (asignar)**

ctrl | teclas

*Var* := *Valor**Var* := *Lista**Var* := *Matriz**Función(Parám1,...)* := *Expr**Función(Parám1,...)* := *Lista**Función(Parám1,...)* := *Matriz*

Si la variable *Var* no existe, crea *Var* y la inicializa para *Valor*, *Listao Matriz*.

Si *Var* ya existe y no está bloqueada o protegida, reemplaza sus contenidos con *Valor*, *Listao Matriz*.

$myvar := \frac{\pi}{4}$	.785398
$yI(x) := 2 \cdot \cos(x)$	Done
$lst5 := \{1,2,3,4\}$	$\{1,2,3,4\}$
$matg := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
$str1 := "Hello"$	"Hello"

## © [texto]

© procesa *texto* como una línea de comentario, lo que le permite anotar funciones y programas que usted crea.

© puede estar al comienzo y en cualquier parte en la línea. Todo a la derecha de ©, al final de la línea, es el comentario.

**Nota para introducir el ejemplo:** Para obtener instrucciones sobre cómo introducir las definiciones de programas y funciones en varias líneas, consulte la sección Calculadora de la guía del producto.

Define  $g(n)$ =Func

© Declare variables

Local  $i,result$

$result:=0$

For  $i,1,n,1$  ©Loop  $n$  times

$result:=result+i^2$

EndFor

Return  $result$

EndFunc

Done

$g(3)$

14

## 0b, 0h

 teclas,  teclas

## 0b número Binario

En modo de base decimal:

0b10+0hF+10

27

## 0h número Hexadecimal

En modo de base binaria:

0b10+0hF+10

0b11011

Denota un número binario o hexadecimal, respectivamente. Para ingresar un número binario o hexadecimal, usted debe ingresar el prefijo 0b ó 0h independientemente del modo de la Base. Sin un prefijo, un número se trata como decimal (base 10).

Los resultados se despliegan de acuerdo con el modo de la Base.

En modo de base hexadecimal:

0b10+0hF+10

0h1B

## TI-Nspire™ CX II: comandos para dibujar

Este es un documento suplementario de la Guía de referencia de TI-Nspire™ y de la Guía de referencia de TI-Nspire™ CAS. Todos los comandos de TI-Nspire™ CX II se incorporarán y publicarán en la versión 5.1 de la Guía de referencia de TI-Nspire™ y de la Guía de referencia de TI-Nspire™ CAS.

### ***Cómo programar gráficos***

Se han agregado nuevos comandos en los dispositivos portátiles TI-Nspire™ CX II y en las aplicaciones de escritorio TI-Nspire™ para la programación de gráficos.

Los dispositivos portátiles TI-Nspire™ CX II cambiarán a este modo de gráficos mientras ejecutan los comandos de gráficos y volverán al contexto en donde se ejecutó el programa después de que se complete el programa.

La pantalla mostrará "Running..." en la barra superior mientras se ejecuta el programa. Mostrará "Finished" cuando se complete el programa. La presión de cualquier tecla hará que el sistema haga una transición fuera del modo de gráficos.

- La transición al modo de gráficos se activa automáticamente cuando se detecta uno de los comandos de Dibujar (gráficos) durante la ejecución del programa TI-Basic.
- Esta transición solo sucede al ejecutar un programa desde la calculadora, en un documento o una calculadora en el Bloc de Notas.
- La transición fuera del modo de gráficos sucede cuando termina el programa.
- El modo de gráficos solo se está disponible en dispositivos portátiles TI-Nspire™ CX II y en la vista de dispositivos portátiles TI-Nspire™ CX II. Significa que no se está disponible en la vista de documentos de computadora en el escritorio ni en iOS.
  - Si se detecta un comando de gráficos mientras se ejecuta un programa TI-Basic desde el contexto incorrecto, se muestra un mensaje de error y el programa TI-Basic termina.

### ***Pantalla de gráficos***

La pantalla de gráficos tendrá un encabezado en la parte superior de la pantalla en donde los comandos de gráficos no pueden escribir.

El área para dibujar de la pantalla de gráficos se borrará (color = 255,255,255) cuando se inicie la pantalla de gráficos.

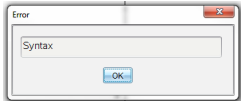
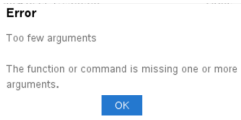
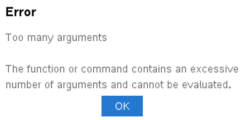
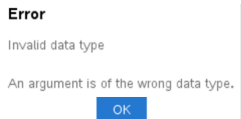
<b>Pantalla de gráficos</b>	<b>Predeterminado</b>
Altura	212
Anchura	318
Color	blanco: 255,255,255

## ***Vista y configuraciones predeterminadas***

- Los iconos de estado en la barra superior (estado de batería, estado de modo de evaluación, indicador de la red, etc.) no estarán visibles mientras se ejecute un programa de gráficos.
- Color predeterminado para dibujar: Negro (0,0,0)
- Estilo de pluma predeterminado: normal, liso
  - Espesor: 1 (delgado), 2 (normal), 3 (más grueso)
  - Estilo: 1 (liso), 2 (punteado), 3 (línea discontinua)
- Todos los comandos para dibujar utilizarán el color actual y las configuraciones de pluma; ya sea los valores predeterminados o aquellos que se establecen con los comandos de TI-Basic.
- La fuente del texto es fija y no se puede cambiar.
- Cualquier salida a la pantalla de gráficos se dibujará dentro de una ventana de recorte que es del tamaño del área para dibujar de la pantalla de gráficos. No se dibujará ninguna salida dibujada que se extienda fuera del área para dibujar de la pantalla de gráficos recortada. No se mostrará ningún mensaje de error.
- Todas las coordenadas x, y especificadas para los comandos de dibujo se definen para que 0,0 se encuentre en la parte superior del área para dibujar de la pantalla de gráficos.
  - **Excepciones:**
    - **DrawText** usa las coordenadas en la esquina inferior izquierda de la caja vinculante del texto.
    - **SetWindow** usa la esquina inferior izquierda de la pantalla
- Todos los parámetros de los comandos se pueden proporcionar como expresiones que evalúan un número, el cual se redondea al número entero más cercano.

## Mensajes de errores de la pantalla de gráficos

Si falla la validación, se muestra un mensaje de error.

Mensaje de error	Descripción	Vista
Error Sintaxis	Si el verificador de sintaxis detecta cualquier error de sintaxis, se muestra un mensaje de error e intenta colocar el cursor cerca del primer error para que usted lo pueda corregir.	
Error Muy pocos argumentos	A la función o al comando le falta un argumento o más	
Error Demasiados argumentos	La función o el comando contiene una cantidad excesiva de argumentos y no se puede evaluar.	
Error Tipo de datos no válido	Un argumento es del tipo de datos incorrecto.	

## Comandos no válidos mientras está en modo de gráficos

No se permiten algunos comandos una vez que el programa cambia al modo de gráficos. Si estos comandos se detectan mientras está en modo de gráficos, se mostrará un error y se terminará el programa.

Comando rechazado	Mensaje de error
Solicitar	No se puede ejecutar la solicitud en modo gráfico
Solicitar cadena	No se puede ejecutar RequestStr en modo gráfico
Texto	No se puede ejecutar texto en modo gráfico

Los comandos que imprimen texto en la calculadora, **disp** y **dispAt**, serán comandos compatibles en el contexto de gráficos. El texto de estos comandos se enviará a la pantalla de la calculadora (no a gráficos) y estará visible después de que el programa salga y el sistema vuelva a la aplicación de Calculadora.



**Borra**  $x$ ,  $y$ , *ancho*, *alto*

Borra toda la pantalla si no se especifican parámetros.

Si se especifican  $x$ ,  $y$ , *ancho* y *alto*, se borrará el rectángulo definido por los parámetros.

Borrar

Borra toda la pantalla

Borrar 10,10,100,50

Borra un área de rectángulo con la esquina superior izquierda en (10, 10), ancho de 100 y alto de 50

**DrawArc**Catálogo >   
CXII**DrawArc** *x, y, ancho, alto, startAngle, arcAngle*

Dibuja un arco dentro del rectángulo vinculante definido con los ángulos iniciales y de arco proporcionados.

*x, y*: coordenada superior izquierda del rectángulo vinculante

*ancho, alto*: dimensiones del rectángulo vinculante

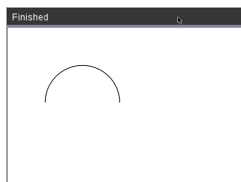
El "ángulo arco" define el barrido del arco.

Estos parámetros se pueden suministrar como expresiones que evalúan un número que se redondea al próximo número entero.

DrawArc 20,20,100,100,0,90



DrawArc 50,50,100,100,0,180



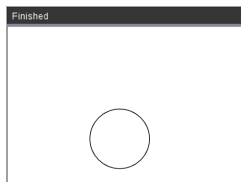
Consulte también: [FillArc](#)

**DrawCircle**Catálogo >   
CXII**DrawCircle** *x, y, radio*

*x, y*: coordenada del centro

*radio*: radio del círculo

DrawCircle 150,150,40



Consulte también: [FillCircle](#)



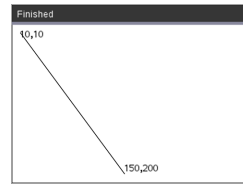
**DrawLine**  $x1, y1, x2, y2$

Dibuja una línea de  $x1, y1, x2, y2$ .

Expresiones que evalúan un número, el cual se redondea al número entero más cercano.

**Límites de pantalla:** Si las coordenadas especificadas provocan que cualquier parte de la línea se dibuje fuera de la pantalla de gráficos, se recortará esa parte de la línea y no se mostrará un mensaje de error.

DrawLine 10,10,150,200



**DrawPoly**

Los comandos tienen dos variantes:

**DrawPoly**  $xlist, ylist$

o

**DrawPoly**  $x1, y1, x2, y2, x3, y3...xn, yn$

**Nota:** DrawPoly  $xlist, ylist$

La forma conectará  $x1, y1$  a  $x2, y2, x2, y2$  a  $x3, y3$  etc.

**Nota:** DrawPoly  $x1, y1, x2, y2, x3, y3...xn, yn$

$xn, yn$  **NO** se conectará automáticamente a  $x1, y1$ .

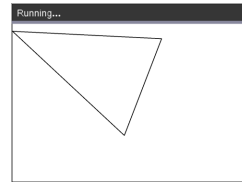
Expresiones que evalúan una lista de flotantes reales  
 $xlist, ylist$

Expresiones que evalúan una sola flotación real  
 $x1, y1...xn, yn$  = coordenadas para vértices de polígono

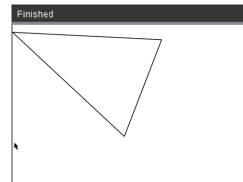
$xlist:=\{0,200,150,0\}$

$ylist:=\{10,20,150,10\}$

DrawPoly  $xlist,ylist$



DrawPoly 0,10,200,20,150,150,0,10



**Nota: DrawPoly:** Dimensiones de tamaño de entrada (ancho/alto) relacionadas con las líneas dibujadas.

Las líneas se dibujan en una caja vinculante alrededor de la coordenada especificada y las dimensiones como el tamaño real del polígono dibujado serán más grandes que el ancho y alto.

Consulte también: [FillPoly](#)

## DrawRect

**DrawRect** *x, y, ancho, alto*

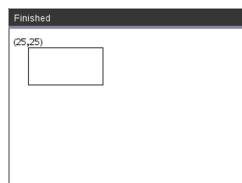
*x, y*: coordenada superior izquierda de rectángulo

*ancho, alto*: ancho y alto del rectángulo (rectángulo dibujado hacia abajo y a la derecha desde la coordenada inicial).

**Nota:** Las líneas se dibujan en una caja vinculante alrededor de la coordenada especificada y las dimensiones como el tamaño real del rectángulo dibujado serán más grandes que el ancho y alto indicados.

Consulte también: [FillRect](#)

DrawRect 25,25,100,50



## DrawText

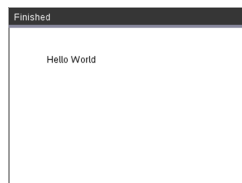
**DrawText** *x, y, exprOrString1*  
*[,exprOrString2]...*

*x, y*: coordenada de salida de texto

Dibuja el texto en *exprOrString* en la ubicación de coordenadas *x, y* especificadas.

Las reglas de *exprOrString* son las mismas que para **Disp: DrawText** puede tomar varios argumentos.

DrawText 50,50,"Hello World"



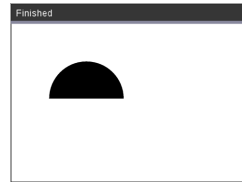
**FillArc**Catálogo >   
CXII**FillArc** *x, y, ancho, alto de startAngle, arcAngle**x, y*: coordenada superior izquierda del rectángulo vinculante

Dibuja y llena un arco dentro del rectángulo vinculante definido con los ángulos iniciales y de arco proporcionados.

El color de relleno predeterminado es negro. El comando [SetColor](#) puede establecer el color de relleno

El "ángulo arco" define el barrido del arco

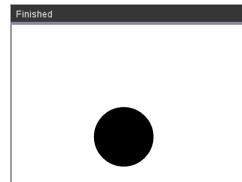
FillArc 50,50,100,100,0,180

**FillCircle**Catálogo >   
CXII**FillCircle** *x, y, radio**x, y*: coordenada del centro

Dibuja y llena un círculo en el centro especificado con el radio especificado.

El color de relleno predeterminado es negro. El comando [SetColor](#) puede establecer el color de relleno.

FillCircle 150,150,40



¡Aquí!

**FillPoly**Catálogo >   
CXII**FillPoly** *xlist, ylist*

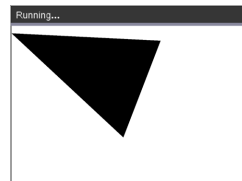
o

**FillPoly** *x1, y1, x2, y2, x3, y3...xn, yn***Nota:** La línea y el color se especifican con [SetColor](#) y [SetPen](#)

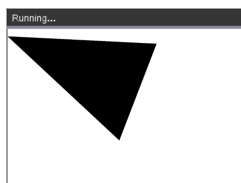
xlist:={0,200,150,0}

ylist:={10,20,150,10}

FillPoly xlist, ylist



FillPoly 0,10,200,20,150,150,0,10

**FillRect****FillRect** *x, y, ancho, alto*

*x, y*: coordenada superior izquierda de rectángulo

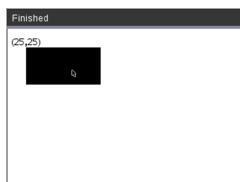
*ancho, alto*: ancho y alto del rectángulo

Dibuja y llena un rectángulo con la esquina superior izquierda en las coordenadas especificadas en (*x,y*)

El color de relleno predeterminado es negro.  
El comando [SetColor](#) puede establecer el color de relleno

**Nota:** La línea y el color se especifican con [SetColor](#) y [SetPen](#)

FillRect 25,25,100,50



## G

### getPlatform()

Catálogo >   
CXII

#### getPlatform()

getPlatform()

"dt"

Devuelve:

"dt" en las aplicaciones de software de escritorio

"hh" en los dispositivos portátiles TI-Nspire™ CX

"ios" en la aplicación TI-Nspire™ CX iPad®

---

**PaintBuffer**

Pinta el búfer de gráficos en la pantalla

Este comando se utiliza con UseBuffer para aumentar la velocidad de visualización en pantalla cuando el programa genere varios objetos gráficos.

UseBuffer

```
For n,1,10
```

```
x:=randInt(0,300)
```

```
y:=randInt(0,200)
```

```
radio:=randInt(10,50)
```

```
Wait 0.5
```

```
DrawCircle x,y,radio
```

```
EndFor
```

PaintBuffer

Este programa muestra los 10 círculos al mismo tiempo.

Si se elimina el comando "UseBuffer", se muestra cada círculo como se dibuja.

Consulte también: [UseBuffer](#)

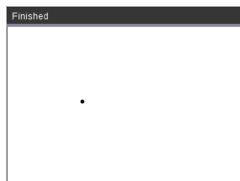
**PlotXY**  $x, y, forma$ 

$x, y$ : coordenada para graficar la forma

*forma*: número entre 1 y 13 para especificar la forma

- 1 - Círculo llenado
- 2 - Círculo vacío
- 3 - Cuadrado llenado
- 4 - Cuadrado vacío
- 5 - Cruz
- 6 - Más
- 7 - Delgado
- 8 - punto medio, sólido
- 9 - punto medio, vacío
- 10 - punto grande, sólido
- 11 - punto grande, vacío
- 12 - punto más grande, sólido
- 13 - punto más grande, vacío

PlotXY 100,100,1

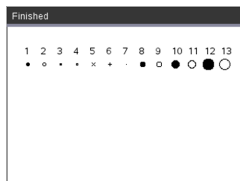


For n,1,13

DrawText 1+22\*n,40,n

PlotXY 5+22\*n,50,n

EndFor



**SetColor**Catálogo >   
CXII**SetColor**

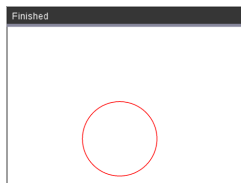
Valor rojo, valor verde, valor azul

Los valores válidos para rojo, verde y azul están entre 0 y 255

Establece el color para los comandos de dibujo subsecuentes

SetColor 255,0,0

DrawCircle 150,150,100

**SetPen**Catálogo >   
CXII**SetPen**

espesor, estilo

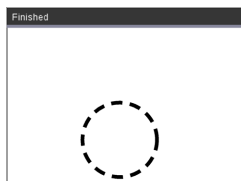
espesor: 1 &lt;= espesor &lt;= 3 | 1 es el más delgado, 3 es el más grueso

estilo: 1 = Suave, 2 = Punteado, 3 = Línea discontinua

Establece el estilo de la pluma para comandos de dibujo subsecuentes

SetPen 3,3

DrawCircle 150,150,50

**SetWindow**Catálogo >   
CXII**SetWindow**

xMin, xMax, yMin, yMax

Establece una ventana lógica que se asigna al área de dibujo de gráficos. Todos los parámetros son obligatorios.

Si la parte del objeto dibujado se encuentra fuera de la ventana, se recortará la salida (no se muestra) y no aparecerá ningún mensaje de error.

SetWindow 0,160,0,120

establecerá la ventana de salida en 0,0 en la esquina inferior izquierda con ancho de 160 y alto de 120

DrawLine 0,0,100,100

SetWindow 0,160,0,120

SetPen 3,3

DrawLine 0,0,100,100

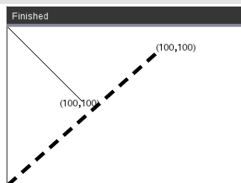


Si  $x_{min}$  es mayor o igual a  $x_{max}$ , o  $y_{min}$  es mayor o igual a  $y_{max}$ , se muestra un mensaje de error.

Cualquier objeto dibujado antes de un comando `SetWindow` no se volverá a dibujar en la nueva configuración.

Para restablecer los parámetros de la ventana a los valores predeterminados, utilice:

`SetWindow 0,0,0,0`



**UseBuffer**

Dibuja a un búfer de gráficos fuera de pantalla en vez de la pantalla (para aumentar el rendimiento)

Este comando se utiliza con `PaintBuffer` para aumentar la velocidad de visualización en pantalla cuando el programa genere varios objetos gráficos.

Con `UseBuffer`, se muestran todos los gráficos solo después de que se ejecuta el siguiente comando `PaintBuffer`.

Solo se necesita usar `UseBuffer` una vez, por ejemplo, cada uso de `PaintBuffer` no necesita un `UseBuffer` correspondiente

`UseBuffer`

```
For n,1,10  
x:=randInt(0,300)  
y:=randInt(0,200)  
radio:=randInt(10,50)  
Wait 0.5  
DrawCircle x,y,radio  
EndFor  
PaintBuffer
```

Este programa muestra los 10 círculos al mismo tiempo.

Si se elimina el comando "`UseBuffer`", se muestra cada círculo como se dibuja.

Consulte también: [PaintBuffer](#)



---

## Elementos vacíos (inválidos)

Cuando analice datos del mundo real, usted quizá no siempre tenga un conjunto de datos completo. El software TI-Nspire™ permite elementos de datos vacíos, o inválidos, de manera que usted podrá proceder con los datos cercanamente completos en lugar de tener que comenzar de nuevo o descartar los casos incompletos.


Usted puede encontrar un ejemplo de datos que incluye elementos vacíos en el capítulo de Listas y Hoja de Cálculo, bajo “Cómo graficar datos de hoja de cálculo”.

La función **delVoid()** le permite eliminar elementos vacíos de una lista. La función **isVoid()** le permite probar un elemento vacío. Para obtener detalles, vea **delVoid()**, página 42 y **isVoid()**, página 81.

**Nota:** Para ingresar un elemento vacío manualmente en una expresión matemática, escriba “\_” o la palabra clave **inválido**. La palabra clave **inválido** se convierte automáticamente en un símbolo “\_” cuando se evalúa la expresión. Para escribir “\_” en el dispositivo portátil, presione  .

### Cálculos que incluyen elementos inválidos

La mayoría de los cálculos que incluyen una entrada inválida producirán un resultado inválido. Vea los casos especiales abajo.

	_
$\gcd(100, \_)$	-
$3 + \_$	-
$\{5, \_, 10\} - \{3, 6, 9\}$	$\{2, \_, 1\}$

### Argumentos de lista que contienen elementos inválidos

Las siguientes funciones y comandos ignoran (se saltan) los elementos inválidos encontrados en argumentos de lista.

**count**, **countIf**, **cumulativeSum**, **freqTable**►**list**, **frequency**, **max**, **mean**, **median**, **product**, **stDevPop**, **stDevSamp**, **sum**, **sumIf**, **varPop**, y **varSamp**, así como cálculos de regresión, **OneVar**, **TwoVar** estadísticas de **FiveNumSummary**, intervalos de confianza y pruebas estadísticas

$\text{sum}\{2, \_, 3, 5, 6, 6\}$	16.6
$\text{median}\{1, 2, \_, \_, 3\}$	2
$\text{cumulativeSum}\{1, 2, \_, 4, 5\}$	$\{1, 3, \_, 7, 12\}$
$\text{cumulativeSum}\left(\begin{bmatrix} 1 & 2 \\ 3 & - \\ 5 & 6 \end{bmatrix}\right)$	$\begin{bmatrix} 1 & 2 \\ 4 & - \\ 9 & 8 \end{bmatrix}$

## Argumentos de lista que contienen elementos inválidos

**SortA** y **SortD** mueven todos los elementos vacíos dentro del primer argumento a la parte inferior.

$\{5,4,3,_,1\} \rightarrow list1$	$\{5,4,3,_,1\}$
$\{5,4,3,2,1\} \rightarrow list2$	$\{5,4,3,2,1\}$
SortA list1,list2	Done
list1	$\{1,3,4,5,_\}$
list2	$\{1,3,4,5,2\}$

$\{1,2,3,_,5\} \rightarrow list1$	$\{1,2,3,_,5\}$
$\{1,2,3,4,5\} \rightarrow list2$	$\{1,2,3,4,5\}$
SortD list1,list2	Done
list1	$\{5,3,2,1,_\}$
list2	$\{5,3,2,1,4\}$

En las regresiones, un vacío en una lista X o Y introduce un vacío para el elemento correspondiente del residual.

$ll:=\{1,2,3,4,5\}; l2:=\{2,_,3,5,6,6\}$	$\{2,_,3,5,6,6\}$
LinRegMx ll,l2	Done
stat.Resid	$\{0.434286,_, -0.862857, -0.011429, 0.44\}$
stat.XReg	$\{1,_,3,4,5\}$
stat.YReg	$\{2,_,3,5,6,6\}$
stat.FreqReg	$\{1,_,1,1,1,1\}$

Una categoría omitida en las regresiones introduce un vacío para el elemento correspondiente del residual.

$ll:=\{1,3,4,5\}; l2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
cat:={"M","M","F","F"}; incl:={"F"}	$\{ "F" \}$
LinRegMx ll,l2,1,cat,incl	Done
stat.Resid	$\{_,_,0,0\}$
stat.XReg	$\{_,_,4,5\}$
stat.YReg	$\{_,_,5,6,6\}$
stat.FreqReg	$\{_,_,1,1,1\}$

Una frecuencia de 0 en las regresiones introduce un vacío para el elemento correspondiente del residual.

$ll:=\{1,3,4,5\}; l2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
LinRegMx ll,l2,{1,0,1,1}	Done
stat.Resid	$\{0.069231,_, -0.276923, 0.207692\}$
stat.XReg	$\{1,_,4,5\}$
stat.YReg	$\{2,_,5,6,6\}$
stat.FreqReg	$\{1,_,1,1,1\}$

## Accesos directos para ingresar expresiones matemáticas

Los accesos directos le permiten ingresar elementos de expresiones matemáticas al escribir en lugar de usar el Catálogo o la Paleta de Símbolos. Por ejemplo, para ingresar la expresión  $\sqrt{6}$ , usted puede escribir `sqrt(6)` en la línea de ingreso. Cuando usted presiona `enter`, la expresión `sqrt(6)` se cambia a  $\sqrt{6}$ . Algunos accesos directos son útiles tanto desde el dispositivo portátil como desde el teclado de la computadora. Otros son útiles principalmente desde el teclado de la computadora.

### Desde el dispositivo portátil o el teclado de la computadora

Para ingresar esto:	Escriba este acceso directo:
$\pi$	<code>pi</code>
$\theta$	<code>theta</code>
$\infty$	<code>infinity</code>
$\leq$	<code>&lt;=</code>
$\geq$	<code>&gt;=</code>
$\neq$	<code>/=</code>
$\Rightarrow$ (implicación lógica)	<code>=&gt;</code>
$\Leftrightarrow$ (implicación doble lógica, XNOR)	<code>&lt;=&gt;</code>
$\rightarrow$ (almacenar operador)	<code>=:</code>
$   $ (valor absoluto)	<code>abs (...)</code>
$\sqrt{()}$	<code>sqrt (...)</code>
$\Sigma()$ (Plantilla de sumas)	<code>sumSeq (...)</code>
$\Pi()$ (Plantilla de productos)	<code>prodSeq (...)</code>
$\sin^{-1}()$ , $\cos^{-1}()$ , ...	<code>arcsin (...)</code> , <code>arccos (...)</code> , ...
$\Delta$ List()	<code>deltaList (...)</code>

### Desde el teclado de la computadora

Para ingresar esto:	Escriba este acceso directo:
$i$ (constante imaginaria)	<code>@i</code>
$e$ (base de logaritmo natural $e$ )	<code>@e</code>
$E$ (notación científica)	<code>@E</code>
$\top$ (trasponer)	<code>@t</code>

Para ingresar esto:	Escriba este acceso directo:
$\text{r}$ (radianes)	@r
$^\circ$ (grados)	@d
g (gradianes)	@g
$\sphericalangle$ (ángulo)	@<
► (conversión)	@>
►Decimal, ►approxFraction ( ), y así sucesivamente.	@>Decimal, @>approxFraction ( ), y así sucesivamente.

# Jerarquía de EOS™ (Sistema Operativo de Ecuaciones)

Esta sección describe el Sistema Operativo de Ecuaciones (EOS™) que se usa en la tecnología de aprendizaje de matemáticas y ciencias de TI-Nspire™. Los números, las variables y las funciones se ingresan en una secuencia directa sencilla. El software EOS™ evalúa las expresiones y ecuaciones mediante la agrupación entre paréntesis, y de acuerdo con las prioridades descritas a continuación.

## Orden de la evaluación

Nivel	Operador
1	Paréntesis ( ), paréntesis rectangulares [ ], corchetes { }
2	Indirección (#)
3	Llamadas de función
4	Operadores posteriores: grados-minutos-segundos ( $^{\circ}$ , $'$ , $''$ ), factorial (!), porcentaje (%), radián ( $^{\circ}$ ), subíndice ([ ]), trasponer ( $^{\top}$ )
5	Exponenciación, operador de potencia (^)
6	Negación (-)
7	Concatenación de cadenas, (&)
8	Multiplicación (*), división (/)
9	Adición (+), sustracción (-)
10	Relaciones de igualdad: igual (=), no igual ( $\neq$ o $\neq$ ), menor que (<), menor que o igual ( $\leq$ o $\leq$ ), mayor que (>), mayor que o igual ( $\geq$ o $\geq$ )
11	Lógico <b>not</b>
12	Lógico <b>and</b>
13	Lógico <b>or</b>
14	<b>xor, nor, nand</b>
15	Implicación lógica ( $\Rightarrow$ )
16	Implicación doble lógica, XNOR ( $\Leftrightarrow$ )
17	Operador restrictivo (" ")
18	Almacenar ( $\rightarrow$ )

## Paréntesis, paréntesis rectangulares y corchetes

Todos los cálculos dentro de un par de paréntesis, paréntesis rectangulares o corchetes se evalúan primero. Por ejemplo, en la expresión  $4(1+2)$ , el software EOS™ evalúa primero la parte de la expresión dentro del paréntesis,  $1+2$ , y luego multiplica el resultado, 3, por 4.

El número de paréntesis, paréntesis rectangulares y corchetes iniciales y finales debe ser el mismo dentro de una expresión o ecuación. Si no es así, se despliega un mensaje de error que indica el elemento faltante. Por ejemplo,  $(1+2)/(3+4)$  desplegará el mensaje de error “) Faltante”.

**Nota:** Debido a que el software TI-Nspire™ le permite definir sus propias funciones, un nombre de variable seguido de una expresión entre paréntesis se considera como una “llamada de función” en lugar de una multiplicación implícita. Por ejemplo  $a(b+c)$  es la función  $a$  evaluada por  $b+c$ . Para multiplicar la expresión  $b+c$  por la variable  $a$ , use la multiplicación explícita:  $a*(b+c)$ .

### Indirección

El operador de indirección (#) convierte una cadena en un nombre de variable o función. Por ejemplo, #("x"&"y"&"z") crea un nombre de variable xyz. La indirección también permite la creación y modificación de variables desde dentro de un programa. Por ejemplo, si  $10 \rightarrow r$  y  $"r" \rightarrow s1$ , entonces  $\#s1=10$ .

### Operadores posteriores

Los operadores posteriores son operadores que vienen directamente después de un argumento, como  $5!$ ,  $25\%$  ó  $60^\circ 15' 45''$ . Los argumentos seguidos de un operador posterior se evalúan en el cuarto nivel de prioridad. Por ejemplo, en la expresión  $4^3!$ ,  $3!$  se evalúa primero. El resultado, 6, entonces se convierte en el exponente de 4 para producir 4096.

### Exponenciación

La exponenciación (^) y la exponenciación elemento por elemento (.^) se evalúan de derecha a izquierda. Por ejemplo, la expresión  $2^3^2$  se evalúa igual que  $2^(3^2)$  para producir 512. Esto es diferente de  $(2^3)^2$ , que es 64.

### Negación

Para ingresar un número negativo, presione  $\boxed{-}$  seguido del número. Las operaciones posteriores y la exponenciación se realizan antes de la negación. Por ejemplo, el resultado de  $-x^2$  es un número negativo, y  $-9^2 = -81$ . Use paréntesis para cuadrar un número negativo como  $(-9)^2$  para producir 81.

### Restricción ("|")

El argumento que sigue el operador restrictivo ("|") proporciona una serie de restricciones que afectan la evaluación del argumento que precede al operador.



# Características de programación de TI-Nspire CX II - TI-Basic

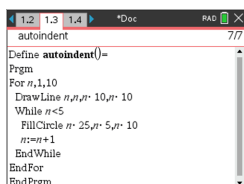
## Sangría automática en el editor de programación

Ahora el editor de programas TI-Nspire™ hace sangrías automáticas de enunciados dentro de un comando de bloque.

Los comandos de bloque son If/EndIf, For/EndFor, While/EndWhile, Loop/EndLoop, Try/EndTry

El editor automáticamente define espacios en los comandos del programa dentro de un comando de bloque. El comando de cierre del bloque se alineará con el comando de abertura.

El siguiente ejemplo muestra la sangría automática en los comandos de bloque anidados.



```
autoindent 7/7
Define autoindent()=
Prgm
For n,1,10
  DrawLine n,n,n-10,n-10
  While n<5
    FillCircle n-25,n-5,n-10
    n:=n+1
  EndWhile
EndFor
EndPrgm
```

Los fragmentos de código que se copian y pegan mantendrán la sangría original.

Si se abre un programa creado en una versión anterior del software, se mantendrá la sangría original.

---

## Mensajes de error mejorados para TI-Basic

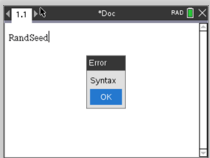
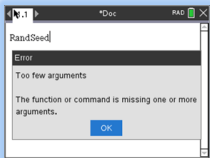
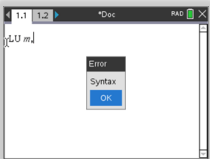
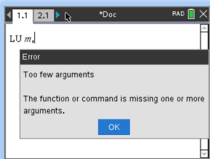
### Errores

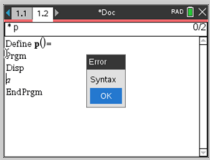
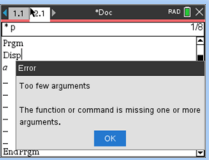
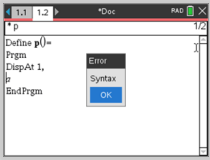
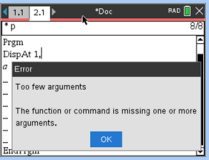
Condición de error	Nuevo mensaje
Error en la declaración condicional (If/While)	Una declaración condicional no se resolvió a <b>TRUE</b> o <b>FALSE</b> <b>NOTA:</b> Con el cambio para colocar el cursor en la línea con el error, ya no tenemos que especificar si el error es un enunciado con "If" o con "While".
Falta EndIf	Se esperaba <b>EndIf</b> pero se encontró una declaración End diferente
Falta EndFor	Se esperaba <b>EndFor</b> pero se encontró una declaración End diferente
Falta EndWhile	Se esperaba <b>EndWhile</b> pero se encontró una

Condición de error	Nuevo mensaje
	declaración End diferente
FaltaEndLoop	Se esperaba <b>EndLoop</b> pero se encontró una declaración End diferente
Falta EndTry	Se esperaba <b>EndTry</b> pero se encontró una declaración End diferente
Se omitió "Then" después de If <condition>	Falta <b>If..Then</b>
Se omitió "Then" después de Elseif <condition>	Falta <b>Then</b> en el bloque: <b>Elseif</b> .
Cuando "Then", "Else" y "Elseif" se detectaron fuera de los bloques de control	<b>Else</b> no es válido fuera de bloques: <b>If..Then..Endif</b> o <b>Try..EndTry</b>
"Elseif" aparece fuera del bloque "If..Then..Endif"	<b>Elseif</b> no es válido fuera del bloque: <b>If..Then..Endif</b>
"Then" aparece fuera del bloque "If....Endif"	<b>Then</b> no es válido fuera del bloque: <b>If..Endif</b>

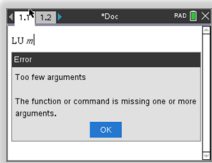
## Errores de sintaxis

En caso de que se usen comandos que esperan uno o más argumentos con una lista incompleta de argumentos, se emitirá **"Too few argument error"** en lugar del error **"syntax"**

Comportamiento actual	Nuevo comportamiento de CX II
 <p>A screenshot of a TI-84 Plus calculator interface. The command 'RandSeed' is entered. An error dialog box is displayed with the text 'Error Syntax' and an 'OK' button.</p>	 <p>A screenshot of a TI-84 Plus calculator interface. The command 'RandSeed' is entered. An error dialog box is displayed with the text 'Error Too few arguments' and 'The function or command is missing one or more arguments.' and an 'OK' button.</p>
 <p>A screenshot of a TI-84 Plus calculator interface. The command 'L1 U m' is entered. An error dialog box is displayed with the text 'Error Syntax' and an 'OK' button.</p>	 <p>A screenshot of a TI-84 Plus calculator interface. The command 'L1 U m' is entered. An error dialog box is displayed with the text 'Error Too few arguments' and 'The function or command is missing one or more arguments.' and an 'OK' button.</p>

Comportamiento actual	Nuevo comportamiento de CX II
 <p>The screenshot shows a TI-84 Plus CE II calculator screen with the following code:  <pre>* p Define p()= Prgm Disp p EndPrgm</pre> An error dialog box is displayed with the message "Syntax" and an "OK" button.</p>	 <p>The screenshot shows a TI-84 Plus CE II calculator screen with the following code:  <pre>* p Prgm Disp EndPrgm</pre> An error dialog box is displayed with the message "Too few arguments" and the text "The function or command is missing one or more arguments." and an "OK" button.</p>
 <p>The screenshot shows a TI-84 Plus CE II calculator screen with the following code:  <pre>* p Define p()= Prgm DispAt 1, p EndPrgm</pre> An error dialog box is displayed with the message "Syntax" and an "OK" button.</p>	 <p>The screenshot shows a TI-84 Plus CE II calculator screen with the following code:  <pre>* p Prgm DispAt 1, p EndPrgm</pre> An error dialog box is displayed with the message "Too few arguments" and the text "The function or command is missing one or more arguments." and an "OK" button.</p>

**Nota:** Cuando una lista incompleta de argumentos no está seguida de una coma, el mensaje de error es: “too few arguments”. Esto es igual que en las versiones anteriores.



## Constantes y valores

La siguiente tabla muestra las constantes y sus valores que están disponibles al realizar conversiones de unidades. Se pueden ingresar manualmente o seleccionarlos de la lista de **Constantes en Utilidades > Conversiones de unidades** (dispositivo portátil: presione

 3).

Constante	Nombre	Valor
_c	Velocidad de la luz	299792458 _m/_s
_Cc	Constante de Coulomb	8987551787.3682 _m/_F
_Fc	Constante de Faraday	96485.33289 _coul/_mol
_g	Aceleración de gravedad	9.80665 _m/_s <sup>2</sup>
_Gc	Constante gravitacional	6.67408E-11 _m <sup>3</sup> /_kg/_s <sup>2</sup>
_h	Constante de Planck	6.626070040E-34 _J_s
_k	Constante de Boltzmann	1.38064852E-23 _J/_°K
_μ0	Permeabilidad de un vacío	1.2566370614359E-6 _N/_A <sup>2</sup>
_μb	Magnetón de Bohr	9.274009994E-24 _J_m <sup>2</sup> /_Wb
_Me	Masa en reposo del electrón	9.10938356E-31 _kg
_Mμ	Masa del muon	1.883531594E-28 _kg
_Mn	Masa en reposo del neutrón	1.674927471E-27 _kg
_Mp	Masa en reposo del protón	1.672621898E-27 _kg
_Na	Número de Avogadro	6.022140857E23 /_mol
_q	Carga del electrón	1.6021766208E-19 _coul
_Rb	Radio de Bohr	5.2917721067E-11 _m
_Rc	Constante molar de gas	8.3144598 _J/_mol/_°K
_Rdb	Constante de Rydberg	10973731.568508/_m
_Re	Radio del electrón	2.8179403227E-15 _m
_u	Masa atómica	1.660539040E-27 _kg
_Vm	Volumen molar	2.2413962E-2 _m <sup>3</sup> /_mol
_ε0	Permeabilidad de un vacío	8.8541878176204E-12 _F/_m
_σ	Constante de Stefan-Boltzmann	5.670367E-8 _W/_m <sup>2</sup> /_°K <sup>4</sup>
_φ0	Cuantificación del flujo magnético	2.067833831E-15 _Wb

## Códigos y mensajes de error

Cuando ocurre un error, su código se asigna a la variable *códigoErr*. Los programas y funciones definidos por el usuario pueden examinar *códigoErr* para determinar la causa de un error. Para ver un ejemplo del uso de *códigoErr*, vea el Ejemplo 2 bajo el comando **Try**, página 172.

**Nota:** Algunas condiciones de error aplican sólo a los productos TI-Nspire™ CAS, y algunos aplican sólo a los productos TI-Nspire™.

Código de error	Descripción
10	Una función no produjo un valor
20	Una prueba no resolvió para VERDADERO o FALSO.  Por lo general, las variables indefinidas no se pueden comparar. Por ejemplo, la prueba Si $a < b$ causará este error si $a$ o $b$ es indefinido cuando se ejecuta la sentencia Si.
30	El argumento no puede ser un nombre de carpeta.
40	Error de argumento
50	Incongruencia de argumento  Dos o más argumentos deben ser del mismo tipo.
60	El argumento debe ser una expresión Booleana o un entero
70	El argumento debe ser un número decimal
90	El argumento debe ser una lista
100	El argumento debe ser una matriz
130	El argumento debe ser una cadena
140	El argumento debe ser un nombre de variable.  Asegúrese de que el nombre: <ul style="list-style-type: none"><li>• no comience con un dígito</li><li>• no contenga espacios o caracteres especiales</li><li>• no use guión bajo o punto en una manera inválida</li><li>• no exceda las limitaciones de longitud</li></ul> Ve la sección de la Calculadora en la documentación para obtener más detalles.
160	El argumento debe ser una expresión
165	Las baterías están demasiado bajas para enviar o recibir  Instale baterías nuevas antes de enviar o recibir.
170	Límite

Código de error	Descripción
	El límite inferior debe ser menor que el límite superior para definir el intervalo de búsqueda.
180	Salto La tecla <code>esc</code> o <code>on</code> se presionó durante un cálculo largo o durante la ejecución del programa.
190	Definición circular Este mensaje se despliega para evitar que la memoria se agote durante el reemplazo infinito de valores de variable durante la simplificación. Por ejemplo, $a+1 \rightarrow a$ , donde $a$ es una variable indefinida, causará este error.
200	Expresión de restricción inválida Por ejemplo, $\text{solve}(3x^2-4=0,x) \mid x < 0 \text{ or } x > 5$ produciría este error porque la restricción está separada por "or" en lugar de "and".
210	Tipo de datos inválido Un argumento es del tipo de datos incorrecto.
220	Límite dependiente
230	Dimensión Un índice de lista o matriz no es válido. Por ejemplo, si la lista $\{1,2,3,4\}$ está almacenada en $L1$ , entonces $L1[5]$ es un error de dimensión porque $L1$ sólo contiene cuatro elementos.
235	Error de Dimensión No hay elementos suficientes en las listas.
240	Incongruencia de dimensión Dos o más argumentos deben ser de la misma dimensión. Por ejemplo, $[1,2]+[1,2,3]$ es una incongruencia de dimensión porque las matrices contienen un número de elementos distinto.
250	Dividir por cero
260	Error de dominio Un argumento debe estar en un dominio especificado. Por ejemplo, <code>rand(0)</code> no es válido.
270	Duplicar nombre de variable
280	Else y Elseif son inválidos afuera del bloque <code>if...Endif</code>
290	A TerminarIntentar le falta la sentencia Else congruente
295	Iteración excesiva

Código de error	Descripción
300	Lista o matriz de 2 ó 3 elementos esperada
310	El primer argumento de nSolve debe ser una ecuación en una variable sencilla. No puede contener una variable no valorada que no sea la variable de interés.
320	El primer argumento de solve o cSolve debe ser una ecuación o desigualdad Por ejemplo, solve( $3x^2-4,x$ ) es vacío porque el primer argumento no es una ecuación.
345	Unidades inconsistentes
350	Índice fuera de rango
360	La cadena de indirección no es un nombre de variable válido
380	Ans indefinido O bien el cálculo anterior no creó Ans o no se ingresó ningún cálculo anterior
390	Asignación inválida
400	Valor de asignación inválido
410	Comando inválido
430	Inválido para las configuraciones del modo actual
435	Cálculo inválido
440	multiplicación implícita inválida Por ejemplo, $x(x+1)$ es inválido; mientras que, $x*(x+1)$ es la sintaxis correcta. Esto es para evitar una confusión entre la multiplicación implícita y la definición de la función.
450	Inválido en una función o expresión actual Sólo ciertos comandos son válidos en una función definida por el usuario
490	Inválido en el bloque Try..EndTry
510	Lista o matriz inválida
550	Inválido afuera de la función o el programa Un número de comandos no es válido afuera de una función o un programa. Por ejemplo, Local no se puede usar, a menos que sea una función o un programa.
560	Inválido afuera de los bloques Loop..EndLoop, For...EndFor, o While...EndWhile. Por ejemplo, el comando Exit es válido sólo adentro de estos bloques de bucle.
565	Inválido afuera del programa
570	nombre de ruta inválido

Código de error	Descripción
	Por ejemplo, \var es inválida.
575	Complejo polar inválido
580	Referencia de programa inválida Los programas no se pueden referenciar dentro de funciones o expresiones como $1+p(x)$ donde p es un programa.
600	Tabla inválida
605	uso de unidades inválido
610	Nombre de variable inválido en una sentencia Local
620	Nombre de variable o función inválido
630	Referencia de variable inválida
640	Sintaxis de vector inválida
650	Transmisión de enlace Una transmisión entre dos unidades no se completó. Verifique que el cable de conexión esté bien conectado en ambos extremos.
665	Matriz no diagonalizable
670	Memoria Baja 1. Borre algunos datos en este documento 2. Guarde y cierre este documento Si 1 y 2 fallan, extraiga y reinserte las baterías
672	Agotamiento de recursos
673	Agotamiento de recursos
680	( Faltante
690	) Faltante
700	" Faltantes
710	] Faltante
720	} Faltante
730	Sintaxis del bloque inicio o final faltante
740	Entonces faltante en el bloque If..EndIf
750	El nombre no es una función o un programa



Código de error	Descripción
765	Ninguna función seleccionada
780	No se encontró ninguna solución
800	Resultado no real Por ejemplo, si el software está en la configuración Real, $\sqrt{-1}$ es inválido. Para permitir resultados complejos, cambie la Configuración del Modo "Real o Complejo" a RECTANGULAR O POLAR.
830	Desbordamiento
850	Programa no encontrado No se pudo encontrar una referencia de programa adentro de otro programa en la ruta provista durante la ejecución.
855	Las funciones de tipo aleatorio no se permiten en la representación gráfica
860	Recurción demasiado profunda
870	variable de nombre o sistema reservada
900	Error de argumento El modelo mediana-mediana no se pudo aplicar al conjunto de datos.
910	Error de sintaxis
920	Texto no encontrado
930	Muy pocos argumentos Uno o más argumentos faltantes en la función o el comando.
940	Demasiados argumentos La expresión o ecuación contiene un número de argumentos excesivo y no se puede evaluar.
950	Demasiados subíndices
955	Demasiadas variables indefinidas
960	La variable no está definida No hay ningún valor asignado a la variable. Use uno de los siguientes comandos: <ul style="list-style-type: none"> <li>• alm →</li> <li>• :=</li> <li>• <b>Define</b></li> </ul> para asignar valores a las variables

Código de error	Descripción
965	SO sin licencia
970	Variable en uso, así que las referencias o los cambios no se permiten
980	La variable está protegida
990	Nombre de variable inválido Asegúrese de que el nombre no exceda las limitaciones de longitud
1000	Dominio de variables de ventana
1010	Zoom
1020	Error interno
1030	Violación de memoria protegida
1040	Función no soportada. Esta función requiere del Sistema de Álgebra de Computadora. Pruebe TI-Nspire™ CAS.
1045	Operador no soportado. Este operador requiere del Sistema de Álgebra de Computadora. Pruebe TI-Nspire™ CAS.
1050	Característica no soportada. Este operador requiere del Sistema de Álgebra de Computadora. Pruebe TI-Nspire™ CAS.
1060	El argumento de entrada debe ser numérico. Sólo las entradas que contienen valores numéricos están permitidos.
1070	Argumento de función trigonométrica demasiado grande para una reducción exacta
1080	Uso de Ans no soportado. Esta aplicación no soporta Ans.
1090	La función no está definida. Use uno de los siguientes comandos: <ul style="list-style-type: none"> <li>• <b>Define</b></li> <li>• <b>:=</b></li> <li>• <b>alm →</b></li> </ul> para definir una función.
1100	Cálculo no real Por ejemplo, si el software está en la configuración Real, $\sqrt{-1}$ es inválido. Para permitir resultados complejos, cambie la Configuración del Modo "Real o Complejo" a RECTANGULAR O POLAR.
1110	Límites inválidos
1120	Ningún cambio de signo

Código de error	Descripción
1130	El argumento no puede ser una lista o matriz
1140	Error de argumento El primer argumento debe ser una expresión polinómica en el segundo argumento. Si el segundo argumento se omite, el software intenta seleccionar un predeterminado.
1150	Error de argumento Los primeros dos argumento deben ser expresiones polinómicas en el tercer argumento. Si el tercer argumento se omite, el software intenta seleccionar un predeterminado.
1160	nombre de ruta de librería inválido Un nombre de ruta debe ser en la forma <code>xxx\yyy</code> , donde: <ul style="list-style-type: none"> <li>• La parte <code>xxx</code> puede tener de 1 a 16 caracteres.</li> <li>• La parte <code>yyy</code> puede tener de 1 a 15 caracteres.</li> </ul> Vea la sección de Librería en la documentación para obtener más detalles.
1170	Uso de nombre de ruta de librería inválido <ul style="list-style-type: none"> <li>• No se puede asignar un valor a un nombre de ruta al usar <b>Define</b>, <code>:=o alm →</code>.</li> <li>• Un nombre de ruta no se puede declarar como una variable Local o usarse como un parámetro en una definición de función o de programa.</li> </ul>
1180	Nombre de variable de librería inválido. Asegúrese de que el nombre: <ul style="list-style-type: none"> <li>• No contenga un punto</li> <li>• No comience con un guión bajo</li> <li>• No exceda de 15 caracteres</li> </ul> Vea la sección de Librería en la documentación para obtener más detalles.
1190	Documento de librería no encontrado: <ul style="list-style-type: none"> <li>• Verifique que la librería esté en la carpeta MiLib.</li> <li>• Actualice Librerías.</li> </ul> Vea la sección de Librería en la documentación para obtener más detalles.
1200	Variable de librería no encontrada: <ul style="list-style-type: none"> <li>• Verifique que la variable de librería existe en el primer problema en la librería.</li> <li>• Asegúrese de que la variable de librería se ha definido como LibPub o LibPriv.</li> </ul>

Código de error	Descripción
	<ul style="list-style-type: none"> <li>Actualice Librerías.</li> </ul> Vea la sección de Librería en la documentación para obtener más detalles.
1210	Nombre de acceso directo de librería inválido. Asegúrese de que el nombre: <ul style="list-style-type: none"> <li>No contenga un punto</li> <li>No comience con un guión bajo</li> <li>No exceda de 16 caracteres</li> <li>No es un nombre reservado</li> </ul> Vea la sección de Librería en la documentación para obtener más detalles.
1220	Error de dominio: Las funciones tangentLine y normalLine sólo soportan funciones valoradas reales.
1230	Error de dominio. Los operadores de conversión trigonométrica no están soportados en los modos de ángulo en Grados o Gradianes.
1250	Error de Argumento Use un sistema de ecuaciones lineales. Ejemplo de un sistema de dos ecuaciones lineales con variables x y y: $3x+7y=5$ $2y-5x=-1$
1260	Error de Argumento: El primer argumento de nfMín o nfMax debe ser una expresión en una variable sencilla. No puede contener una variable no valorada que no sea la variable de interés.
1270	Error de Argumento El Orden de la derivada debe ser igual a 1 ó 2.
1280	Error de Argumento Use un polinomio en forma expandida en una variable.
1290	Error de Argumento Use un polinomio en una variable.
1300	Error de Argumento Los coeficientes del polinomio se deben evaluar a valores numéricos.

<b>Código de error</b>	<b>Descripción</b>
1310	Error de argumento: Una función no se pudo evaluar para uno o más de sus argumentos.
1380	Error de argumento: No se permiten llamadas anidadas en la función del dominio().

## Códigos de advertencia y mensajes

Puede usar la función `warnCodes()` para almacenar los códigos de las advertencias generadas al evaluar una expresión. Esta tabla enumera cada código de advertencia numérico y su mensaje asociado. Para obtener un ejemplo de cómo almacenar códigos de advertencia, consulte `warnCodes()`, página 181.

Código de advertencia	Mensaje
10000	La operación podría introducir soluciones falsas. Cuando corresponda, intente utilizar métodos gráficos para verificar los resultados.
10001	Diferenciar una ecuación puede producir una ecuación falsa.
10002	Solución cuestionable Cuando corresponda, intente utilizar métodos gráficos para verificar los resultados.
10003	Exactitud cuestionable Cuando corresponda, intente utilizar métodos gráficos para verificar los resultados.
10004	La operación podría perder las soluciones. Cuando corresponda, intente utilizar métodos gráficos para verificar los resultados.
10005	<code>cResolver</code> podría especificar más ceros.
10006	<code>Resolver</code> puede especificar más ceros. Cuando corresponda, intente utilizar métodos gráficos para verificar los resultados.
10007	Es posible que existan más soluciones. Intente especificar límites superiores o inferiores correctos y/o un punto inicial. Ejemplos utilizando la función <code>solución()</code> : <ul style="list-style-type: none"><li><code>solución(Ecuación, Var=Estimar)   limiteInferior&lt;Var&lt;limiteSuperior</code></li><li><code>solución(Ecuación, Var)   limiteInferior&lt;Var&lt;limiteSuperior</code></li><li><code>solución(Ecuación, Var=Estimar)</code></li></ul> Cuando corresponda, intente utilizar métodos gráficos para verificar los resultados.
10008	El dominio del resultado podría ser más pequeño que el dominio de la entrada.
10009	El dominio del resultado podría ser más GRANDE que el dominio de la entrada.
10012	Cálculo no real

<b>Código de advertencia</b>	<b>Mensaje</b>
10013	$\infty^0$ o $\text{undef}^0$ reemplazado por 1
10014	$\text{undef}^0$ reemplazado por 1
10015	$1^\infty$ o $1^\text{undef}$ reemplazado por 1
10016	$1^\text{undef}$ reemplazado por 1
10017	Desbordamiento reemplazado por $\infty$ o $-\infty$
10018	La operación requiere y entrega un valor de 64 bits.
10019	Agotamiento del recurso, la simplificación podría estar incompleta.
10020	Argumento de función de trigonometría demasiado grande para una reducción exacta.
10021	La entrada contiene un parámetro indefinido. El resultado podría no ser válido para todos los posibles valores de parámetro.
10022	Especificar los límites inferior y superior apropiados podría producir una solución.
10023	El escalar se ha multiplicado por la matriz identidad.
10024	Resultado obtenido con aritmética aproximada.
10025	La equivalencia no se puede verificar en el modo EXACTO.
10026	Puede ignorarse la limitación. Especifique la limitación en la forma " <code>\</code> " 'Variable MathTestSymbol Constant' o un conjunto de estas formas, por ejemplo ' <code>x&lt;3 and x&gt;-12</code> '

## Información general

### ***Ayuda en línea***

[education.ti.com/eguide](http://education.ti.com/eguide)

Seleccione su país para obtener más información del producto.

### ***Comuníquese con Asistencia de TI***

[education.ti.com/ti-cares](http://education.ti.com/ti-cares)

Seleccione su país para obtener recursos técnicos y otro tipo de ayuda.

### ***Información sobre el servicio y la garantía***

[education.ti.com/warranty](http://education.ti.com/warranty)

Seleccione su país para obtener información acerca de la duración de los términos de la garantía o sobre el servicio para productos.

Garantía limitada. Esta garantía no afecta a sus derechos legales.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243



# Índice alfabético

		<b>^</b>	
		$\wedge^{-1}$ , recíproco .....	211
		$\wedge$ , potencia .....	194
		<b> </b>	
- , negar (-);negar (-) .....	197	, operador restrictivo .....	211
-		<b>+</b>	
- , sustraer[*] .....	191	+ , agregar .....	191
<b>!</b>		<b>/</b>	
! , factorial .....	202	/ , dividir[*] .....	193
"		<b>=</b>	
" , notación en segundo .....	209	= , igual .....	198
<b>#</b>		$\neq$ , no igual[*] .....	198
# , indirección .....	207	<b>&gt;</b>	
# , operador de indirección .....	236	> , mayor que .....	200
<b>%</b>		<b><math>\Pi</math></b>	
% , porcentaje .....	197	$\Pi$ , producto[*] .....	204
<b>&amp;</b>		<b><math>\Sigma</math></b>	
& , adjuntar .....	202	$\Sigma$ ( ) , suma[*] .....	204
<b>*</b>		$\Sigma$ Cap ( ) .....	206
* , multiplicar .....	192	$\Sigma$ Int ( ) .....	205
<b>,</b>		<b><math>\sqrt{\quad}</math></b>	
, , notación en minuto .....	209	$\sqrt{\quad}$ , raíz cuadrada[*] .....	203
<b>.</b>		<b><math>\int</math></b>	
.- , punto sustracción .....	196	$\int$ , integral[*] .....	203
.* , punto multiplicación .....	196	<b><math>\leq</math></b>	
./ , punto división .....	196	$\leq$ , menor que o igual .....	199
.^ , punto potencia .....	197	<b><math>\geq</math></b>	
.+ , punto agregar .....	195	$\geq$ , mayor que o igual .....	200
<b>:</b>			
:= , asignar .....	213		

►, convertir a ángulo en gradianes [Grad] .....	73	0b, indicador binario .....	214
►Base10, se despliega como entero decimal[Base10] .....	18	0h, indicador hexadecimal .....	214
►Base16, se despliega como hexadecimal[Base16] .....	19	<b>1</b>	
►Base2, se despliega como binario [Base2] .....	17	10^( ), potencia de diez .....	210
►Cilind, se despliega como vector cilíndrico[Cilind] .....	37	<b>A</b>	
►DD, se despliega como ángulo decimal[DD] .....	38	abs( ), valor absoluto .....	7
►Decimal, despliega el resultado como decimal[Decimal] .....	38	accesoDirectoLib( ), crear accesos directos para objetos de librería .....	83
►Esfera, se despliega como vector esférico[Esfera] .....	158	adjuntar, & .....	202
►Fracciónaprox( ) .....	13	agregar, + .....	191
►GMS, se despliega como grado/minuto/segundo [GMS] .....	46	agrFilaM( ), multiplicación y suma de fila de matriz .....	103
►Polar, se despliega como vector polar[Polar] .....	121	aleatoria matriz, randMat( ) .....	131
►Rad, convertir a ángulo radián .....	130	aleatorio polinomio, randPoly( ) .....	132
►Rect, se muestra como vector rectangular .....	133	semilla de número, RandSeed ..	132
		and, Boolean operator .....	8
→		angle( ), ángulo .....	9
→, almacenar .....	213	angle, ángulo( ) .....	9
		ANOVA, análisis de varianza unidireccional .....	9
⇒		ANOVA2vías, análisis de varianza bidireccional .....	10
⇒, implicación lógica[*] .....	201, 233	Ans, última respuesta .....	12
		aprox( ), aproximado .....	12
⇔		aproximado, aprox( ) .....	12
⇔, implicación lógica doble[*] .....	201	arccos() .....	13
		arccosh() .....	13
©		arccot() .....	13
©, comentario .....	214	arcoth() .....	14
		arccsc() .....	14
° , grados/minutos/segundos[*] .....	209	arccsch() .....	14
° , notación en grados[*] .....	209	arcoseno, cos <sup>-1</sup> ( ) .....	28
		arcoseno, sin <sup>-1</sup> ( ) .....	154
		arcotangente, tan <sup>-1</sup> ( ) .....	167
		arcsec() .....	14
		arcsech() .....	14
		arcsin() .....	14
		arcsinh() .....	14
		arctan() .....	14
		arctanh() .....	14
		argumentos del VTD .....	176

argumentos en funciones del VTD ..	176	de variable .....	76
aumentar( ), aumentar/concatenar ..	14	dentro, inString .....	76
aumentar/concatenar, aumentar( ) ..	14	derecha, right( ) .....	77, 139-140
augmentCol .....	24	expresión para cadena, cadena(	
		) .....	163
		formato, formato( ) .....	58
		indirección, # .....	207
		izquierda, izquierda( ) .....	82
		rotar, rotate( ) .....	141
		cambiar( ), cambiar .....	150
		cambiar, cambiar( ) .....	150
		car( ), cadena de caracteres .....	21
		caracteres	
		cadena, car( ) .....	21
		código numérico, ord( ) .....	118
		Cdf( ) .....	55
		Cdfgeom( ) .....	63
		CdfNormal( ) .....	112
		CdfT( ), probabilidad de distribución	
		de student-t .....	169
		ciclo, Ciclo .....	36
		Ciclo, ciclo .....	36
		clear	
		error, ClrErr .....	24
		ClrErr, clear error .....	24
		códigos de error y mensajes .....	250
		códigos y mensajes de advertencia ..	250
		códigos y mensajes de error .....	241
		comando de Texto .....	169
		Comando Detener .....	163
		Comando Wait .....	180
		combinaciones, nCr( ) .....	107
		comentario, © .....	214
		cómo almacenar	
		símbolo, & .....	213
		cómo borrar	
		variable, BorrVar .....	41
		cómo definir	
		función o programa privado ....	40
		función o programa público ....	40
		cómo desbloquear variables y	
		grupos de variables .....	179
		cómo ordenar	
		ascendente, OrdenarA .....	158
		descendente, OrdenarD .....	158
		cómo programar	
		definir programa, Prgm .....	124
		desplegar datos, Desp .....	44
		pasar error, PasarErr .....	119
<b>B</b>			
BA, descomposición baja-alta de			
matriz .....	97		
binario			
indicador, Ob .....	214		
se despliega, Base2 .....	17		
binomCdf( ) .....	19, 79		
binomPdf( ) .....	20		
bloquear variables y grupos de			
variables .....	93		
Bloquear, bloquear variable o grupo			
de variables .....	93		
Boolean operators			
and .....	8		
borrar			
elementos inválidos de la lista ..	42		
Borrar .....	219		
borrInval( ), eliminar los elementos			
inválidos .....	42		
BorrVar, borrar variable .....	41		
bucle, Bucle .....	96		
Bucle, bucle .....	96		
BxRegLin, regresión lineal .....	84		
<b>C</b>			
c22vías .....	21		
cadena			
dimensión, dim( ) .....	43		
longitud .....	43		
cadena de caracteres, car( ) .....	21		
cadena de formato, formato( ) .....	58		
cadena med, med( ) .....	101		
cadena( ), expresión para cadena ...	163		
cadena			
adjuntar, & .....	202		
cadena de caracteres, car( ) ....	21		
cadena med, med( ) .....	101		
cadena para expresión, expr( ) ..	53		
cambiar, cambiar( ) .....	150		
código de caracter, ord( ) .....	118		
cómo formatear .....	58		
cómo usar para crear nombres	236		





CdfNormal( )	112	idioma	
CdfT( )	169	obtener información del idioma	69
invNorm( )	79	if, If	74
invt( )	80	If, if	74
Inv $\chi^2$ ( )	78	ifFn( )	75
PdfNorm( )	113	igual, =	198
Pdfpoiss( )	121	imag( ), parte imaginaria	76
PdFT( )	171	implicación lógica doble, $\Leftrightarrow$	201
$\chi^2$ Cdf( )	22	implicación lógica, $\Rightarrow$	201, 233
$\chi^2$ GOF( )	22	ln( ), logaritmo natural	90
$\chi^2$ Pdf( )	23	indirección, #	207
funciones definidas por el usuario	39	inString( ), dentro de la cadena	76
funciones financieras, vtdI( )	175	int( ), entero	77
funciones financieras, vtdN( )	175	intDiv( ), dividir entero	77
funciones financieras, vtdPgo( )	176	integral definida	
funciones financieras, vtdVF( )	175	plantilla para	6
funciones financieras, vtdVP( )	176	integral, $\int$	203
funciones y programas definidos por el usuario	40	Intentar, comando de manejo de error	172
funciones y variables		interpolat( ), interpolar	77
cómo copiar	26	IntervalosRegLin, regresión lineal	86
		IntervalosRegMult( )	104
		intervaloT, intervalo de confianza t	170
		intervaloT_2Muest, intervalo de confianza tde dos muestras	170
		intervaloZ, intervalo de confianza Z	184
		intervaloZ_1Prop, intervalo de confianza Z de una proporción	185
		intervaloZ_2Muest, intervalo de confianza Z de dos muestras	186
		intervaloZ_2Prop, intervalo de confianza Z de dos proporciones	185
		intN( ), integral numérica	110
		inverso, $\wedge^{-1}$	211
		invF( )	78
		invNorm( ), distribución normal acumulada inversa)	79
		invt( )	80
		Inv $\chi^2$ ( )	78
		iPart( ), parte entera	80
		ir a, IrA	73
		IrA, ir a	73
		irr( ), tasa interna de retorno, tasa interna de retorno, irr( )	80
		isPrime( ), prueba de primos	81
<b>G</b>			
g, gradianes	207		
Get	64, 225		
getKey()	65		
GetStr	71		
getType( ), get type of variable	72		
grupos, cómo bloquear y desbloquear	93, 179		
grupos, cómo probar el estado de bloqueo	70		
<b>H</b>			
hexadecimal			
indicador, Oh	214		
se despliega, #Base16	19		
hiperbólico			
arcoseno, $\cosh^{-1}()$	29		
arcoseno, $\sinh^{-1}()$	156		
arcotangente, $\tanh^{-1}()$	168		
coseno, $\cosh()$	29		
seno, $\sinh()$	155		
tangente, $\tanh()$	168		
<b>I</b>			
identity( ), matriz de identidad	74		

isVoid ( ), prueba para elemento vacío, prueba para elemento vacío, isVoid ( ) ..	81	longitud de cadena .....	43
izquierda ( ), izquierda .....	82	<b>M</b>	
izquierda, izquierda ( ) .....	82	más sí, MásSi .....	49
<b>L</b>		MásSi, más sí .....	49
LibPriv .....	40	mat $\Rightarrow$ lista ( ), matriz para lista .....	97
LibPub .....	40	matCorr ( ), matriz de correlación ...	26
librería		matrices	
crear accesos directos para objetos .....	83	aleatorias, randMat ( ) .....	131
LimpiarAZ .....	23	aumentar/concatenar, aumentar ( ) .....	14
lista para matriz, lista4mat ( ) .....	90	cambio de fila, rowSwap ( ) .....	144
lista, conteo condicional de elementos en .....	32	cómo llenar, Llenar .....	56
lista, conteo de elementos en .....	31	descomposición baja-alta, BA ..	97
lista4mat ( ), lista para matriz .....	90	determinante, det ( ) .....	42
listaDelta ( ) .....	41	diagonal, diag ( ) .....	43
listas		dimensión de columna, dimCol ( ) .....	24
aumentar/concatenar, aumentar ( ) .....	14	dimensión de fila, rowDim ( ) ...	144
cadena med, med ( ) .....	101	dimensión, dim ( ) .....	43
diferencia, @lista ( ) .....	89	factorización de QR, QR .....	126
diferencias en una lista, @lista ( ) ..	89	forma escalonada por filas, ref ( ) ..	134
elementos vacíos en .....	231	forma escalonada reducida por filas, rref ( ) .....	144
lista para matriz, lista4mat ( ) ...	90	identidad, identity ( ) .....	74
lista, nuevaLista ( ) .....	109	lista para matriz, lista4mat ( ) ...	90
matriz para lista, mat $\Rightarrow$ lista ( ) ...	97	matriz para lista, mat $\Rightarrow$ lista ( ) ...	97
mínimo, mín ( ) .....	101	mínimo, mín ( ) .....	101
ordenar ascendente, OrdenarA .....	158	multiplicación y suma de fila, agrFilaM ( ) .....	103
ordenar descendente, OrdenarD .....	158	norma de columna, normaCol ( ) ..	25
producto cruzado, pCruz ( ) .....	33	norma de fila, rowNorm ( ) .....	144
producto punto, pPunto ( ) .....	46	nueva, nuevaMat ( ) .....	109
producto, producto ( ) .....	125	operación de fila, filaM ( ) .....	103
suma acumulativa, sumaAcumulativa ( ) ...	36	producto, producto ( ) .....	125
sumatoria, suma ( ) .....	164	punto agregar, .+ .....	195
llenar .....	223-224	punto división, .P .....	196
Llenar, llenar matriz .....	56	punto multiplicación, .* .....	196
local, Local .....	92	punto potencia, .^ .....	197
Local, variable local .....	92	punto sustracción, .N .....	196
logaritmo natural, En ( ) .....	90	submatriz, subMat ( ) .....	163, 165
logaritmos .....	90	suma acumulativa, sumaAcumulativa ( ) ...	36
Logística		suma de fila, rowAdd ( ) .....	143
plantilla para .....	2	sumatoria, suma ( ) .....	164
Logística, regresión logística .....	94	trasponer, T .....	166
LogísticaD, regresión logística .....	95	valorPropio, vlProp ( ) .....	48
		vectorPropio, vcProp ( ) .....	48

matriz (1 × 2)		norma aleatoria, randNorm ( )	131
plantilla para	4	norma Frobenius, norma( )	112
matriz (2 × 1)		norma( ), norma Frobenius	112
plantilla para	4	normaCol( ), norma de columna de	
matriz (2 × 2)		matriz	25
plantilla para	4	not, operador booleano	113
matriz (m × n)		notación en gradián, g	207
plantilla para	4	notación en grado/minuto/segundo	209
matriz de correlación, matCorr( )	26	notación en grados, -	209
matriz de identidad, identity( )	74	notación en minuto,	209
matriz para lista, matLista( )	97	notación en segundo, "	209
máximo común divisor, mcd( )	63	nueva	
mayor que o igual,	200	lista, nuevaLista( )	109
mayor que, >	200	matriz, nuevaMat( )	109
mcd( ), máximo común divisor	63	nuevaLista( ), nueva lista	109
mcm, mínimo común múltiplo	82	nuevaMat( ), nueva matriz	109
med( ), cadena med	101	numérica	
media( ), media	98	derivada, derivadaN( )	108
media, media( )	98	derivada, derivN( )	109-110
mediana( ), mediana	99	integral, intN( )	110
mediana, mediana( )	99	solución, solucionN( )	115
MedMed, regresión de línea media-			
media	99	<b>O</b>	
menor que o igual, {	199	objetos	
mientras, Mientras	182	crear accesos directos para	
Mientras, mientras	182	librería	83
mín( ), mínimo	101	obtDenom( ), obtener/producir	
mínimo común múltiplo, mcm	82	denominador	65
mínimo, mín( )	101	obtener/producir	
mod( ), módulo	103	denominador, obtDenom( )	65
modes		información de variables,	
setting, setMode( )	149	obtInfoVar( )	69, 72
módulo, mod( )	103	número, obtNúm( )	71
mostrar datos, Mostrar	146	obtInfoBloq( ), prueba el estado de	
Mostrar, mostrar datos	146	bloqueo de la variable o del	
muestra aleatoria	132	grupo de variables	70
multiplicar, *	192	obtInfoldioma( ), obtener/producir	
MxRegLin, regresión lineal	85	información del idioma	69
		obtInfoVar( ), obtener/producir	
<b>N</b>		información de variables	72
nand, operador booleano	106	obtModo( ), obtener configuraciones	
nCr( ), combinaciones	107	de modo	70
negación, cómo ingresar números		obtNúm( ), obtener/producir	
negativos	236	número	71
no igual, ≠	198	operador de indirección (#)	236
nom ), convertir efectiva a tasa		operador restrictivo " "	211
nominal	111	operador restrictivo, orden de la	
nor, operador booleano	111	evaluación	235



operadores			
orden de evaluación	235		
Operadores booleanos			
$\Rightarrow$	201, 233		
$\Leftrightarrow$	201		
nand	106		
nor	111		
not	113		
or	117		
xor	183		
or (booleano), or	117		
or, operador booleano	117		
ord( ), código de carácter numérico	118		
OrdenarA, ordenar ascendente	158		
OrdenarD, ordenar descendente	158		
<b>P</b>			
P►Rx( ), coordenada x rectangular	118		
P►Ry( ), coordenada y rectangular	119		
Para	58		
para, Para	58		
Para, para	58		
parte entera, iPart( )	80		
parte imaginaria, imag( )	76		
parteF( ), parte de función	59		
pasar error, PasarErr	119		
PasarErr, pasar error	119		
pCruz( ), producto cruzado	33		
Pdf( )	59		
Pdfgeom( )	64		
PdfNorm( )	113		
Pdfpoiss( )	121		
PdfT( ), densidad de probabilidad de student-t	171		
permutaciones, prN( )	114		
Pgrm, definir programa	124		
piecewise( )	120		
piso( ), piso	57		
piso, piso( )	57		
plantillas			
e exponente	2		
exponente	1		
fracción	1		
función de compuesto de variables (2 piezas)	2		
función de compuesto de variables (N piezas)	3		
integral definida	6		
Logística	2		
matriz (1 x 2)	4		
matriz (2 x 1)	4		
matriz (2 x 2)	4		
matriz (m x n)	4		
primera derivada	5		
producto (P)	5		
raíz cuadrada	1		
raíz enésima	2		
segunda derivada	6		
sistema de ecuaciones (2 ecuaciones)	3		
sistema de ecuaciones (N ecuaciones)	3		
suma (G)	5		
valor absoluto	4		
poissCdf( )	120		
polar			
coordenada, R►Pr( )	129		
coordenada, R►Pθ( )	129		
despliegue de vector, ►Polar	121		
polinomios			
aleatorios, randPoly( )	132		
evaluar, evalPoli( )	122		
porcentaje, %	197		
potencia de diez, 10^( )	210		
potencia, ^	194		
pPunto( ), producto punto	46		
primera derivada			
plantilla para	5		
prN( ), permutaciones	114		
probabilidad de distribución de student-t, CdfT( )	169		
probabilidad de distribución normal, CdfNormal( )	112		
prodSec( )	124		
producir, Return	139		
producto (P)			
plantilla para	5		
producto cruzado, pCruz( )	33		
producto( ), producto	125		
producto, P( )	204		
producto, producto( )	125		
programación			
mostrar datos, Mostrar	146		
programas			
cómo definir una librería privada	40		
cómo definir una librería pública	40		

programas y cómo programar			
desplegar pantalla I/O, Desp	44		
intentar, Intentar	172		
terminar intentar,			
TerminarIntentar	172		
programas y programación			
mostrar pantalla de E/S,			
Mostrar	146		
programs and programming			
clear error, ClrErr	24		
prueba de número primo, isPrime( )	81		
Prueba F de 2 muestras	61		
Prueba t de regresión lineal múltiple	105		
prueba T, pruebaT	173		
Prueba_2M, prueba F de 2 muestras	61		
PruebasRegMult( )	105		
pruebaT, prueba T	173		
pruebaT_2Muest, prueba T de dos			
muestras	174		
PruebaTRegLin	87		
pruebaZ	187		
pruebaZ_1Prop, prueba Z de una			
proporción	187		
pruebaZ_2Muest, prueba Z de dos			
muestras	189		
pruebaZ_2Prop, prueba Z de dos			
proporciones	188		
punto			
agregar, .+	195		
división, .P	196		
multiplicación, .*	196		
potencia, .^	197		
producto, pPunto( )	46		
sustracción, .N	196		
<b>Q</b>			
QR, factorización de QR	126		
<b>R</b>			
R, radián	208		
R→Pr( ), coordenada polar	129		
R→Pθ( ), coordenada polar	129		
Racionalaprox( )	13		
radián, R	208		
RaícesPoli( )	122		
RaícesPoliC( )	32		
raíz cuadrada			
plantilla para		1	
raíz cuadrada, √( )		159, 203	
raíz enésima			
plantilla para		2	
rand( ), número aleatorio	130		
randBin, número aleatorio	130		
randInt( ), entero aleatorio	131		
randMat( ), matriz aleatoria	131		
randNorm( ), norma aleatoria	131		
randPoly( ), polinomio aleatorio	132		
randSamp( )	132		
RandSeed, semilla de número			
aleatorio	132		
real( ), real	133		
real, real( )	133		
recíproco, ^-1	211		
redondeo, round( )	143		
ref( ), forma escalonada por filas	134		
RefreshProbeVars	135		
RegCuad, regresión cuadrática	127		
RegCuart, regresión cuártica	128		
RegCúbica, regresión cúbica	35		
RegExp, regresión exponencial	53		
RegLn, regresión logarítmica	91		
RegMult	104		
RegPot, regresión de potencia	122		
regresión cuadrática, RegCuad	127		
regresión cuártica, RegCuart	128		
regresión cúbica, RegCúbica	35		
regresión de línea media-media			
(MedMed)	99		
regresión de potencia, RegPot	122, 169		
regresión exponencial, RegExp	53		
regresión lineal, AxRegLin	85		
regresión lineal, BxRegLin	84, 86		
regresión logarítmica, RegLn	91		
regresión logística, Logística	94		
regresión logística, LogísticaD	95		
regresión potencia, PowerReg	136, 138		
regresión sinusoidal, RegSin	156		
regresiones			
cuadrática, RegCuad	127		
cuártica, RegCuart	128		
cúbica, RegCúbica	35		
exponencial, RegExp	53		
línea media-media (MedMed)	99		
logarítmica, RegLn	91		
Logística	94		

logística, Logística .....	95	se muestra vector rectangular, ►Rect	133
RegMult .....	104	sec <sup>-1</sup> ( ), secante inversa .....	145
regresión de potencia, RegPot .....	122, 169	sec( ), secante .....	145
regresión lineal, AxRegLin .....	85	sech <sup>-1</sup> ( ), secante hiperbólica inversa	146
regresión lineal, BxRegLin .....	84, 86	sech( ), secante hiperbólica .....	146
regresión potencia, PowerReg .....	136, 138	secSuma() .....	165
sinusoidal, RegSin .....	156	secuen( ), secuencia .....	147
RegSin, regresión sinusoidal .....	156	secuencia, secuen( ) .....	147
remain( ), residuo .....	136	segunda derivada	
RequestStr .....	138	plantilla para .....	6
residuo, remain( ) .....	136	sen( ), seno .....	154
respuesta (última), Ans .....	12	sen/( ), arcoseno .....	154
resultados de dos variables, DosVar	177	senh( ), seno hiperbólico .....	155
resultados, estadísticas .....	160	senh/( ), arcoseno hiperbólico .....	156
ResumenNúmCinco .....	56	seno, sen( ) .....	154
Return, producir .....	139	seqGen( ) .....	147
right( ), derecha .....	139	seqn( ) .....	148
right, right( ) .....	50, 181	sequence, seq( ) .....	147-148
rk23( ), función Runge Kutta .....	140	set	
rotar, rotate( ) .....	141	mode, setMode( ) .....	149
rotate( ), rotar .....	141	setMode( ), set mode .....	149
round( ), redondeo .....	143	signo( ), signo .....	152
rowAdd( ), suma de fila de matriz .....	143	signo, signo( ) .....	152
rowDim( ), dimensión de fila de		simult( ), ecuaciones simultáneas ...	153
matriz .....	144	sistema de ecuaciones (2	
rowNorm( ), norma de fila de matriz	144	ecuaciones)	
rowSwap( ), cambio de fila de matriz	144	plantilla para .....	3
rref( ), forma escalonada reducida		sistema de ecuaciones (N	
por filas .....	144	ecuaciones)	
rzcuad( ), raíz cuadrada .....	159	plantilla para .....	3
<b>S</b>			
salir, Salir .....	52	Sistema Operativo de Ecuaciones	
Salir, salir .....	52	(EOS) .....	235
se despliega como		Solicitar .....	136
ángulo decimal, ►DD .....	38	solucionLin() .....	89
binario, ►Base2 .....	17	solucionN( ), solución numérica ...	115
grado/minuto/segundo, 4GMS .....	46	strings	
hexadecimal, ►Base16 .....	19	right, right( ) .....	50, 181
se despliega como decimal,		subMat( ), submatriz .....	163, 165
►Base10 .....	18	submatriz, subMat( ) .....	163, 165
vector cilíndrico, 4Cilind .....	37	suma (G)	
vector esférico, 4Esfera .....	158	plantilla para .....	5
vector polar, ►Polar .....	121	suma acumulativa,	
se despliega como vector cilíndrico,		sumaAcumulativa( ) .....	36
4Cilind .....	37	suma de pagos de capital .....	206
se muestra como		suma de pagos de interés .....	205
vector rectangular, ►Rect .....	133	suma( ), sumatoria .....	164
		suma, S( ) .....	204
		sumaAcumulativa( ), suma	
		acumulativa .....	36

sumaSi( ) .....	164	valor tiempo del dinero, cantidad de pago .....	176
sumatoria, suma( ) .....	164	valor tiempo del dinero, Interés .....	175
sustitución con el operador " " .....	211	valor tiempo del dinero, número de pagos .....	175
sustraer, N .....	191	valor tiempo del dinero, Valor Futuro .....	175
<b>T</b>			
T, trasponer .....	166	valor tiempo del dinero, valor presente .....	176
tabla de amortización, tablaAmort( ) .....	7, 16	valores de resultados, estadísticos ..	161
tablaAmort( ), tabla de amortización .....	7, 16	valorPropio, vlProp( ) .....	48
tablaFrec( ) .....	60	variable	
$\tan^{-1}()$ , arcotangente .....	167	cómo crear un nombre desde una cadena de caracteres .....	236
$\tan()$ , tangente .....	166	variable local, Local .....	92
tangente, $\tan()$ .....	166	variables	
$\tanh^{-1}()$ , arcotangente hiperbólica ..	168	borrar, BorrVar .....	41
$\tanh()$ , tangente hiperbólica .....	168	limpie todas las letras únicas .....	23
tasa de cambio promedio, TCprom( ) .....	15	local, Local .....	92
tasa efectiva, ef( ) .....	47	variables y funciones	
tasa interna de rendimiento, tirm( ) .....	102	cómo copiar .....	26
tasa nominal, nom( ) .....	111	variables, cómo bloquear y desbloquear .....	70, 93, 179
TCprom( ), tasa de cambio promedio .....	15	varianza, varianza( ) .....	180
techo, techo( ) .....	20, 32	varMuest( ), varianza muestra .....	180
terminar		varPob( ) .....	179
bucle, TerminarBucle .....	96	vcProp( ), vector propio .....	48
función, TerminarFunc .....	62	vcUnid( ), vector de unidad .....	179
intentar, TerminarIntentar .....	172	vector de unidad, vcUnid( ) .....	179
mientras, TerminarMientras .....	182	vectores	
para, TerminarPara .....	58	producto cruzado, pCruz( ) .....	33
terminar bucle, TerminarBucle .....	96	producto de punto, pPunto( ) ..	46
terminar función, TerminarFunc .....	62	se despliega como vector	
terminar mientras,		cilíndrico, 4Cilind .....	37
TerminarMientras .....	182	unidad, vcUnid( ) .....	179
TerminarIntentar, terminar intentar .....	172	vectorPropio, vcProp( ) .....	48
TerminarMientras, terminar		vlProp( ), valorPropio .....	48
mientras .....	182	vpn( ), valor presente neto .....	114
tirm( ), tasa interna de rendimiento		vtdl( ) .....	175
modificada .....	102	vtdN( ) .....	175
trasponer, T .....	166	vtdPgo( ) .....	176
trazado( ) .....	172	vtdVF( ) .....	175
		vtdVP( ) .....	176
<b>U</b>			
UnaVar, estadísticas de una variable .....	116	<b>W</b>	
<b>V</b>			
valor absoluto		warnCodes( ), Warning codes .....	181
plantilla para .....	4		
valor presente neto, vpn( ) .....	114		

**X**

$x^2$ , cuadrado .....	195
XNOR .....	201
xor, exclusivo booleano o .....	183

 **$\Delta$** 

$\Delta$ lista( ), diferencia de lista .....	89
--	----

**X**

$\chi^2$ Cdf( ) .....	22
$\chi^2$ GOF .....	22
$\chi^2$ Pdf( ) .....	23