

TI-Nspire™ CX CAS

Manual de Referência

Informações importantes

Salvo indicação em contrário constante da Licença que acompanha o programa, a Texas Instruments renuncia a todas as garantias mencionadas, quer sejam expressas ou implícitas, incluindo mas não se limitando a qualquer garantia implícita de comercialização ou adequação a um fim específico, no que respeita aos materiais licenciados são disponibilizados numa base "como estão". A TI não se responsabiliza, em circunstância alguma, por qualquer dano indireto, especial ou acidental, relacionado ou decorrente da utilização destes materiais, e a única e exclusiva responsabilidade da Texas Instruments, independentemente da forma de Ação, não excederá o preço indicado na licença do programa. Além disso, a Texas Instruments não se responsabiliza por qualquer reclamação relacionada com a utilização destes materiais por terceiros.

© 2026 Texas Instruments Incorporated

Os produtos reais podem variar ligeiramente das imagens fornecidas.

Conteúdo

Modelos de expressão	1
Lista alfabética	8
A	8
B	18
C	22
D	49
E	63
F	73
G	84
I	95
L	104
M	121
N	130
O	140
P	143
Q	153
R	156
S	172
T	199
U	216
V	216
W	218
X	220
Z	221
Símbolos	230
TI-Nspire™ CX II - Comandos de desenho	258
Programação de gráficos	258
Ecrã de gráficos	258
Vista e definições padrão	259
Mensagens de erro no ecrã de gráficos	260
Comandos inválidos no modo de gráficos	260
C	262
D	263
F	266
G	268
P	269
S	271
U	273

Elementos (nulos) vazios	274
Atalhos para introduzir expressões matemáticas	276
Hierarquia do EOS™ (Equation Operating System)	278
TI-Nspire CX II - Funcionalidades de programação TI-Basic	280
Recuos automáticos no Editor de Programação	280
Mensagens de erro melhoradas para TI-Basic	280
Constantes e valores	283
Mensagens e códigos de erros	284
Códigos de aviso e mensagens	293
Informações gerais	295
Índice remissivo	296

Modelos de expressão

Os modelos de expressão oferecem uma forma simples para introduzir expressões matemáticas em notação matemática padronizada. Quando introduzir um modelo, aparece na linha de entrada com pequenos blocos em posições em que pode introduzir elementos. Um cursor mostra o elemento que pode introduzir.

Utilize as teclas de setas ou prima **tab** para mover o cursor para a posição de cada elemento e escreva um valor ou uma expressão para o elemento. Prima **enter** ou **ctrl** **enter** para avaliar a expressão.

Modelo de fracção

Teclas **ctrl** **÷**



Nota: Consulte também / (dividir), página 232.

Exemplo:

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

Modelo de expoente

Tecla **^**



Nota: Escreva o primeiro valor, prima **^** e, em seguida, escreva o expoente. Para colocar o cursor na base, prima a seta direita (**►**).

Exemplo:

$$2^3 \quad 8$$

Nota: Consulte também ^ (potência), página 233.

Modelo de raiz quadrada

Teclas **ctrl** **x²**



Nota: Consulte também $\sqrt{}$ (raiz quadrada), página 244.

Exemplo:

$$\sqrt{4} \quad 2$$
$$\sqrt{\{9, a, 4\}} \quad \{3, \sqrt{\{a\}}, 2\}$$

Modelo de raiz de índice N

Teclas ctrl ^



$\sqrt{\square}$ **Nota:** Consulte também **raiz()**, página 168.

Exemplo:

$$\sqrt[3]{8} \quad 2$$

$$\sqrt[3]{\{8, 27, b\}} \quad \left\{ 2, 3, b^{\frac{1}{3}} \right\}$$

Modelo de expoente e

Tecla e^x



Exponencial natural e elevado à potência

Nota: Consulte também **e^()**, página 63.

Exemplo:

$$e^1 \quad e$$

$$e^{1.} \quad 2.71828182846$$

Modelo de log

Teclas ctrl 10^x



Calcule o log para uma base especificada. Para uma predefinição de base 10, omite a base.

Nota: Consulte também **log()**, página 117.

Exemplo:

$$\log_{\square}(\square) \quad 0.5$$

Modelo de Função por ramos (2 ramos)

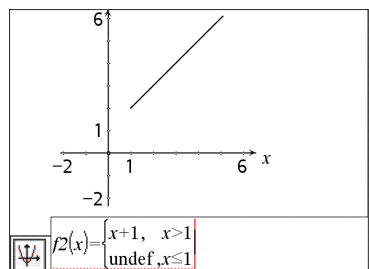
Catálogo |a|^a/_a



Permite criar expressões e condições para uma função por ramos de 2 ramos. Para adicionar um ramo, clique no modelo e repita o modelo.

Nota: Consulte também **piecewise()**, página 144.

Exemplo:



Modelo de Função por ramos (N ramos)

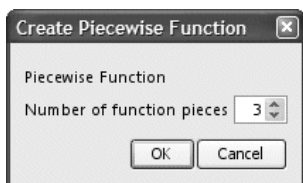
Catálogo>



Permite criar expressões e condições para uma função por ramos de N -ramos. Para adicionar um ramo, clique no modelo e repita o modelo.

Exemplo:

Consulte o exemplo para o modelo de Função por ramos (2 ramos).



Nota: Consulte também `piecewise()`, página 144.

Modelo do sistema de 2 equações

Catálogo>



Cria um sistema de duas equações. Para adicionar uma linha a um sistema existente, clique no modelo e repita o modelo.

Exemplo:

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

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

Nota: Consulte também `sistema()`, página 199.

Modelo do sistema de N equações

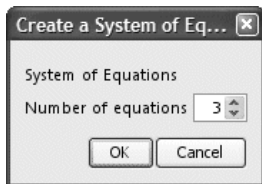
Catálogo>



Permite criar um sistema de N equações. Pede N .

Exemplo:

Consulte o exemplo do modelo do sistema de equações (2 equações).



Nota: Consulte também `sistema()`, página 199.

Modelo do valor absoluto

Catálogo >



 **Nota:** Consulte também **abs()**, página 8.

Exemplo:

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

Modelo gg°mm'ss.ss''

Catálogo >



 °  '  ''

Permite introduzir ângulos na forma **gg° mm' ss.ss''**, em que **gg** é o número de graus decimais, **mm** é o número de minutos e **ss.ss** é o número de segundos.

Exemplo:

$$30^{\circ}15'10'' \quad \frac{10891 \cdot \pi}{64800}$$

Modelo da matriz (2 x 2)

Catálogo >



Cria uma matriz 2 x 2.

Exemplo:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot a \quad \begin{bmatrix} a & 2 \cdot a \\ 3 \cdot a & 4 \cdot a \end{bmatrix}$$

Modelo da matriz (1 x 2)

Catálogo >



Exemplo:

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

Modelo da matriz (2 x 1)

Catálogo >



Exemplo:

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

Modelo da matriz (m x n)

Catálogo >

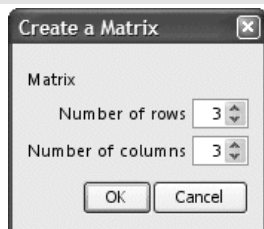


O modelo aparece depois de lhe ser pedido para especificar o número de linhas e colunas.

Exemplo:

Modelo da matriz (m x n)

Catálogo > 



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

Nota: Se criar uma matriz com um grande número de linhas e colunas, pode demorar alguns momentos a aparecer.

Modelo da soma (Σ)

Catálogo > 

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

Exemplo:

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

Nota: Consulte também $\Sigma()$ (sumSeq), página 245.

Modelo do produto (Π)

Catálogo > 

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

Exemplo:

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

Nota: Consulte também $\Pi()$ (prodSeq), página 244.

Modelo da primeira derivada

Catálogo > 

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

Exemplo:

Pode também utilizar o modelo da primeira derivada para calcular a primeira derivada num ponto.

Modelo da primeira derivada

Catálogo > 

Nota: Consulte também **d() (derivada)**, página 241.

$$\frac{d}{dx}(x^3) \quad 3 \cdot x^2$$

$$\frac{d}{dx}(x^3)|_{x=3} \quad 27$$

Modelo da segunda derivada

Catálogo > 

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

Pode também utilizar o modelo da segunda derivada para calcular a segunda derivada num ponto.

Nota: Consulte também **d() (derivada)**, página 241.

Exemplo:

$$\frac{d^2}{dx^2}(x^3) \quad 6 \cdot x$$

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

Modelo da derivada de índice N

Catálogo > 

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

Pode utilizar o modelo da n-ésima derivada para calcular a derivada de ordem n.

Nota: Consulte também **d() (derivada)**, página 241.

Exemplo:

$$\frac{d^3}{dx^3}(x^3)|_{x=3} \quad 6$$

Modelo do integral definido

Catálogo > 

$$\int_a^b \square dx$$

Nota: Consulte também **f() integral()**, página 242.

Exemplo:

$$\int_a^b x^2 dx \quad \frac{b^3}{3} - \frac{a^3}{3}$$

Modelo do integral indefinido

Catálogo > 

$$\int \square d\square$$

Nota: Consulte também `j()` **integral()**, página 242.

Exemplo:

$$\int x^2 dx \quad \frac{x^3}{3}$$

Modelo do limite

Catálogo > 

$$\lim_{\square \rightarrow \square} (\square)$$

Utilize – ou (–) para o limite esquerdo.
Utilize + para o limite direito.

Nota: Consulte também `limit()`, página 106.

Exemplo:

$$\lim_{x \rightarrow 5} (2 \cdot x + 3) \quad 13$$

Lista alfabética

Os itens cujos nomes não sejam alfabéticos (como +, !, e >) são listados no fim desta secção, começando (aqui). Salvo indicação em contrário, todos os exemplos desta secção foram efectuados no modo de reinicialização predefinido e todas as variáveis são assumidas como indefinidas.

A

abs()

Catálogo > 

abs(*ExprI*) ⇒ expressão

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

abs(*ListaI*) ⇒ lista

$$\left| 2-3 \cdot i \right|$$

abs(*MatrizI*) ⇒ matriz

$$\left| z \right|$$

Devolve o valor absoluto do argumento.

$$\left| x+y \cdot i \right|$$

Nota: Consulte também **Modelo do valor absoluto**, página 4.

Se o argumento for um número complexo, devolve o módulo do número.

Nota: Todas as variáveis indefinidas são tratadas como variáveis reais.

amortTbl()

Catálogo > 

amortTbl(*NPmt*, *N*, *I*, *PV*, [*Pmt*], [*FV*], [*PpY*], [*CpY*], [*PmtAt*], [*ValorArredondado*]) ⇒ matriz

$$\text{amortTbl}(12,60,10,5000,,,12,12)$$

Função de amortização que devolve uma matriz como uma tabela de amortização para um conjunto de argumentos TVM.

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

NPmt é o número de pagamentos a incluir na tabela. A tabela começa com o primeiro pagamento.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* e *PmtAt* são descritos na tabela de argumentos TVM, página 213.

- Se omitir *Pmt*, predefine-se para *Pmt* = **tvmPmt** (*N*, *I*, *PV*, *FV*, *PpY*, *CpY*, *PmtAt*).
- Se omitir *FV*, predefine-se para *FV* =0.
- As predefinições para *PpY*, *CpY* e

PmtAt são iguais às predefinições para as funções TVM.

ValorArredondado especifica o número de casas decimais para arredondamento. Predefinição=2.

As colunas da matriz de resultados são por esta ordem: Número de pagamentos, montante pago para juros, montante para capital e saldo.

O saldo apresentado na linha *n* é o saldo após o pagamento *n*.

Pode utilizar a matriz de saída como entrada para as outras funções de amortização Σ **Int()** e Σ **Prn()**, página 245 e **bal()**, página 18.

and

ExprBooleana1 and ExprBooleana2
 $\Rightarrow$ *Expressão booleana*

$x \geq 3$ and $x \geq 4$	$x \geq 4$
$\{x \geq 3, x \leq 0\}$ and $\{x \geq 4, x \leq 2\}$	$\{x \geq 4, x \leq 2\}$

ListaBooleana1 and ListaBooleana2
 $\Rightarrow$ *Lista booleana*

MatrizBooleana1 and MatrizBooleana2
 $\Rightarrow$ *Matriz booleana*

Devolve falso, verdadeiro ou uma forma simplificada da entrada original.

Inteiro1 and Inteiro2 $\Rightarrow$ *número inteiro*

Compara dois números inteiros reais bit a bit com uma operação **and**. Internamente, ambos os números inteiros são convertidos para números binários de 64 bits assinados. Quando os bits correspondentes forem comparados, o resultado é 1 se ambos os bits forem 1; caso contrário, o resultado é 0. O valor devolvido representa os resultados dos bits e aparece de acordo com o modo base.

No modo base Hex:

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

Importante: Zero, não a letra O.

No modo base Bin:

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

No modo base Dec:

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

Pode introduzir os números inteiros em qualquer base numérica. Para uma entrada binária ou hexadecimal, tem de utilizar o prefixo 0b ou 0h, respectivamente. Sem um prefixo, os números inteiros são tratados como decimais (base 10).

Se introduzir um número inteiro decimal muito grande para uma forma binária de 64 bits assinada, é utilizada uma operação de módulo simétrico para colocar o valor no intervalo adequado.

Nota: Uma entrada binária pode ter até 64 dígitos (não contando com o prefixo 0b). Uma entrada hexadecimal pode ter até 16 dígitos.

angle()

Catálogo >

angle(Expr1) $\Rightarrow$ expressão

Devolve o ângulo do argumento, interpretando o argumento como um número complexo.

Nota: Todas as variáveis indefinidas são tratadas como variáveis reais.

No modo de ângulo Graus:

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

No modo de ângulo Gradianos:

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

No modo de ângulo Radianos:

$\text{angle}(1+i)$	$\frac{\pi}{4}$
---------------------	-----------------

$\text{angle}(z)$	$\frac{-\pi \cdot (\text{sign}(z)-1)}{2}$
-------------------	---

$\text{angle}(x+i \cdot y)$	$\frac{\pi \cdot \text{sign}(y)}{2} - \tan^{-1}\left(\frac{x}{y}\right)$
-----------------------------	--

$\text{angle}(\{1+2 \cdot i, 3+0 \cdot i, 0-4 \cdot i\})$	$\left\{\frac{\pi}{2} - \tan^{-1}\left(\frac{1}{2}\right), 0, -\frac{\pi}{2}\right\}$
---	---

angle(Lista1) $\Rightarrow$ lista

angle(Matriz1) $\Rightarrow$ matriz

Devolve uma lista ou matriz de ângulos dos elementos em *Lista1* ou *Matriz1*, interpretando cada elemento como um número complexo que representa um ponto de coordenada rectangular bidimensional.

ANOVA

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

Efectua uma análise de variação de uma via para comparar as médias de 2 a 20 populações. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Marcador =0 para Dados, *Marcador* =1 para Estatística

Variável de saída	Descrição
stat.F	Valor da estatística F
stat.PVal	Menor nível de significância para o qual a hipótese nula pode ser rejeitada
stat.df	Graus de liberdade dos grupos
stat.SS	Soma dos quadrados dos grupos
stat.MS	Quadrados médios para os grupos
stat.dfError	Graus de liberdade dos erros
stat.SSError	Soma dos quadrados dos erros
stat.MSError	Quadrado médio para os erros
stat.sp	Desvio padrão associado
stat.xbarlist	Média da entrada das listas
stat.CLowerList	Intervalos de confiança de 95% para a média de cada lista de entrada
stat.CUpperList	Intervalos de confiança de 95% para a média de cada lista de entrada

ANOVA2way

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

Calcula uma análise de variação bidireccional através da comparação das médias de 2 a 10 populações. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

LinhaNiv=0 para Bloco

LinhaNiv=2,3,...,*Len*-1, para Dois fatores,
em que *Len*=comprimento
(*Lista1*)=comprimento(*Lista2*) = ... =
comprimento(*Lista10*) e *Len* / *LinhaNiv* ∈ {2,3,...}

Saídas: Design do bloco

Variável de saída	Descrição
stat.F	F estatística do factor da coluna
stat.PVal	Menor nível de significância para o qual a hipótese nula pode ser rejeitada
stat.df	Graus de liberdade do factor da coluna
stat.SS	Soma dos quadrados do factor da coluna
stat.MS	Quadrados médios para o factor da coluna
stat.FBloco	F estatística para o factor
stat.PValBlock	Menor probabilidade de rejeição da hipótese nula
stat.dfBlock	Graus de liberdade para factor
stat.SSBlock	Soma dos quadrados para o factor
stat.MSBlock	Quadrados médios para o factor
stat.dfError	Graus de liberdade dos erros
stat.SSError	Soma dos quadrados dos erros
stat.MSError	Quadrados médios para os erros
stat.s	Desvio padrão do erro

Saídas do factor da coluna

Variável de saída	Descrição
stat.Fcol	F estatística do factor da coluna
stat.PValCol	Valor da probabilidade do factor da coluna

Variável de saída	Descrição
stat.dfCol	Graus de liberdade do factor da coluna
stat.SSCol	Soma dos quadrados do factor da coluna
stat.MSCol	Quadrados médios para o factor da coluna

Saídas do factor da linha

Variável de saída	Descrição
stat.FLinha	F estatística do factor da linha
stat.PValRow	Valor da probabilidade do factor da linha
stat.dfRow	Graus de liberdade do factor da linha
stat.SSRow	Soma dos quadrados do factor da linha
stat.MSRow	Quadrados médios para o factor da linha

Saídas de interacção

Variável de saída	Descrição
stat.FInteragir	F estatística da interacção
stat.PValInteract	Valor da probabilidade da interacção
stat.dfInteract	Graus de liberdade da interacção
stat.SSInteract	Soma de quadrados da interacção
stat.MSInteract	Quadrados médios para interacção

Saídas de erros

Variável de saída	Descrição
stat.dfError	Graus de liberdade dos erros
stat.SSError	Soma dos quadrados dos erros
stat.MSError	Quadrados médios para os erros
s	Desvio padrão do erro

Ans	Teclas  	
Ans ⇒ <i>valor</i>	56	56
Devolve o resultado da expressão avaliada mais recentemente.	56+4	60
	60+4	64

approx()	Catálogo > 	
approx (<i>Expr1</i>) ⇒ <i>expressão</i>	$\text{approx}\left(\frac{1}{3}\right)$	0.333333
Devolve a avaliação dos argumentos como uma expressão com valores decimais, quando possível, independentemente do modo Auto ou Aproximado actual.	$\text{approx}\left(\left\{\frac{1}{3}, \frac{1}{9}\right\}\right)$	{ 0.333333, 0.111111 }
Isto é equivalente a introduzir o argumento e a introduzir   .	$\text{approx}\{\sin(\pi), \cos(\pi)\}$	{ 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 (<i>Lista1</i>) ⇒ <i>lista</i>	$\text{approx}\{\sin(\pi), \cos(\pi)\}$	{ 0, -1 }
approx (<i>Matriz1</i>) ⇒ <i>matriz</i>	$\text{approx}\left(\left[\sqrt{2}, \sqrt{3}\right]\right)$	[1.41421 1.73205]
Devolve uma lista ou uma <i>matriz</i> em que cada elemento foi avaliado para um valor decimal, quando possível.		

approxFraction()	Catálogo > 	
<i>Expr</i> ▶ approxFraction (<i>[Tol]</i>) ⇒ <i>expressão</i>	$\frac{1}{2} + \frac{1}{3} + \tan(\pi)$	0.833333
<i>Lista</i> ▶ approxFraction (<i>[Tol]</i>) ⇒ <i>lista</i>	0.8333333333333333 ▶ approxFraction (5.E-14)	$\frac{5}{6}$
<i>Matriz</i> ▶ approxFraction (<i>[Tol]</i>) ⇒ <i>matriz</i>	{ $\pi, 1.5$ } ▶ approxFraction (5.E-14)	$\left\{\frac{5419351}{1725033}, \frac{3}{2}\right\}$
Devolve a entrada como uma fracção com uma tolerância de <i>Tol</i> . Se omitir <i>Tol</i> , é utilizada uma tolerância de 5.E-14.		
Nota: Pode introduzir esta função através da escrita de @>approxFraction (...) no teclado do computador.		

approxRational()**Catálogo >** **approxRational(Expr [, Tol])** $\Rightarrow$ expressão

$$\text{approxRational}(0.333, 5 \cdot 10^{-5}) \quad \frac{333}{1000}$$

approxRational(Lista [, Tol]) $\Rightarrow$ lista

$$\text{approxRational}(\{0.2, 0.33, 4.125\}, 5 \cdot 10^{-14})$$

$$\left\{ \frac{1}{5}, \frac{33}{100}, \frac{33}{8} \right\}$$

approxRational(Matriz [, Tol]) $\Rightarrow$ matriz

Devolve o argumento como uma fracção com uma tolerância de *Tol*. Se omitir *Tol*, é utilizada uma tolerância de 5.E-14.

arccos()**Consulte $\cos^{-1}()$, página 34.****arccosh()****Consulte $\cosh^{-1}()$, página 36.****arccot()****Consulte $\cot^{-1}()$, página 37.****arccoth()****Consulte $\coth^{-1}()$, página 38.****arccsc()****Consulte $\csc^{-1}()$, página 40.****arccsch()****Consulte $\operatorname{csch}^{-1}()$, página 41.**

arcLen()Catálogo > **arcLen(*Expr1*, *Var*, *Início*, *Fim*)** $\Rightarrow$ expressão

Devolve o comprimento do arco de *Expr1* do *Início* ao *Fim* em relação à variável *Var*.

O comprimento do arco é calculado como um integral que assume uma definição do modo de função.

arcLen(*Lista1*, *Var*, *Início*, *Fim*) $\Rightarrow$ lista

Devolve uma lista dos comprimentos dos arcos de cada elemento de *Lista1* do *Início* ao *Fim* em relação a *Var*.

$$\text{arcLen}(\cos(x), x, 0, \pi) \quad 3.8202$$

$$\text{arcLen}(f(x), x, a, b) \quad \int_a^b \sqrt{\left(\frac{d}{dx}(f(x))\right)^2 + 1} dx$$

$$\text{arcLen}(\{\sin(x), \cos(x)\}, x, 0, \pi) \quad \{3.8202, 3.8202\}$$

arcsec()Consulte $\sec^{-1}()$, página 172.**arcsech()**Consulte $\text{sech}^{-1}()$, página 173.**arcsin()**Consulte $\sin^{-1}()$, página 184.**arcsinh()**Consulte $\sinh^{-1}()$, página 185.**arctan()**Consulte $\tan^{-1}()$, página 200.**arctanh()**Consulte $\tanh^{-1}()$, página 202.

augment()Catálogo > **augment(Lista1, Lista2) ⇒ lista**

Devolve uma nova lista que é a *Lista2* acrescentada ao fim da *Lista1*.

augment(Matriz1, Matriz2) ⇒ matriz

Devolve uma nova lista que é a *Matriz2* acrescentada ao fim da *Matriz1*. Quando utilizar o carácter “,” as matrizes têm de ter dimensões de colunas iguais, e a *Matriz2* é acrescentada à *Matriz1* como novas colunas. Não altere *Matriz1* ou *Matriz2*.

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

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

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

Devolve o quociente de diferença de avanço (taxa de câmbio média).

Expr1 pode ser um nome de função definido pelo utilizador (ver **Func**).

Ao especificar o *Valor*, substitui qualquer atribuição de variável anterior ou qualquer substituição atual “|” para a variável.

Passo é o valor do passo. Se omitir *Passo*, predefine-se para 0,001.

Não se esqueça de que a função similar **centralDiff()** utiliza o quociente de diferença central.

$\text{avgRC}(f(x), x, h)$	$\frac{f(x+h) - f(x)}{h}$
$\text{avgRC}(\sin(x), x, h) x=2$	$\frac{\sin(h+2) - \sin(2)}{h}$
$\text{avgRC}(x^2 - x + 2, x)$	$2 \cdot (x - 0.4995)$
$\text{avgRC}(x^2 - x + 2, x, 0.1)$	$2 \cdot (x - 0.45)$
$\text{avgRC}(x^2 - x + 2, x, 3)$	$2 \cdot (x + 1)$

bal()		Catálogo > 																													
bal (<i>NPmt</i> , <i>N</i> , <i>I</i> , <i>PV</i> , [<i>Pmt</i>], [<i>FV</i>], [<i>PpY</i>], [<i>CpY</i>], [<i>PmtAt</i>], [<i>ValorArredondado</i>]) ⇒ <i>valor</i>		bal(5,6,5.75,5000,,12,12) 833.11																													
bal (<i>NPmt</i> , <i>TabelaDeDepreciação</i>) ⇒ <i>valor</i>		<i>tbl</i> :=amortTbl(6,6,5.75,5000,,12,12)																													
Função de amortização que calcula o saldo do plano após um pagamento especificado.		<table><tr><td>0</td><td>0.</td><td>0.</td><td>5000.</td></tr><tr><td>1</td><td>-23.35</td><td>-825.63</td><td>4174.37</td></tr><tr><td>2</td><td>-19.49</td><td>-829.49</td><td>3344.88</td></tr><tr><td>3</td><td>-15.62</td><td>-833.36</td><td>2511.52</td></tr><tr><td>4</td><td>-11.73</td><td>-837.25</td><td>1674.27</td></tr><tr><td>5</td><td>-7.82</td><td>-841.16</td><td>833.11</td></tr><tr><td>6</td><td>-3.89</td><td>-845.09</td><td>-11.98</td></tr></table>		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
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																												
<i>N</i> , <i>I</i> , <i>PV</i> , <i>Pmt</i> , <i>FV</i> , <i>PpY</i> , <i>CpY</i> e <i>PmtAt</i> são descritos na tabela de argumentos TVM, aqui.		bal(4, <i>tbl</i>) 1674.27																													
<i>NPmt</i> especifica o número de pagamentos a partir dos quais quer os dados calculados.																															
<i>N</i> , <i>I</i> , <i>PV</i> , <i>Pmt</i> , <i>FV</i> , <i>PpY</i> , <i>CpY</i> e <i>PmtAt</i> são descritos na tabela de argumentos TVM, aqui.																															
- Se omitir <i>Pmt</i>, predefine-se para <i>Pmt</i> = tvmPmt(<i>N</i>, <i>I</i>, <i>PV</i>, <i>FV</i>, <i>PpY</i>, <i>CpY</i>, <i>PmtAt</i>). - Se omitir <i>FV</i>, predefine-se para <i>FV</i> =0. - As predefinições para <i>PpY</i>, <i>CpY</i> e <i>PmtAt</i> são iguais às predefinições para as funções TVM.																															
<i>ValorArredondado</i> especifica o número de casas decimais para arredondamento. Predefinição=2.																															
bal (<i>NPmt</i> , <i>TabelaDeDepreciação</i>) calcula o saldo após o número de pagamentos <i>NPmt</i> , baseado na tabela de amortização <i>TabelaDeDepreciação</i> . O argumento <i>TabelaDeDepreciação</i> tem de ser uma matriz no forma descrita em amortTbl (), aqui.																															
Nota: Consulte também Σ Int() e Σ Prn() , aqui.																															

NúmeroInteiro1 ►Base2 ⇒ *número inteiro*

256►Base2	0b100000000
0h1F►Base2	0b11111

Nota: Pode introduzir este operador através da escrita de @►Base2 no teclado do computador.

Converte *NúmeroInteiro1* para um número binário. Os números binários ou hexadecimais têm sempre um prefixo 0b ou 0h, respectivamente. Zero, não a letra O, seguido por b ou h.

0b *NúmeroBinário*

0h *NúmeroHexadecimal*

Um número binário pode ter até 64 dígitos. Um número hexadecimal pode ter até 16 dígitos.

Sem um prefixo, *NúmeroInteiro1* é tratado como decimal (base 10). O resultado aparece em binário, independentemente do modo base.

Os números negativos aparecem no formato de “complemento de dois”. Por exemplo,

–1 aparece como

0hFFFFFFFFFFFFFFFF no modo base Hex
0b111...111 (64 1's) no modo base Binário

–263 aparece como

0h8000000000000000 no modo base Hex
0b100...000 (63 zeros) no modo base Binário

Se introduzir um número inteiro na base 10 fora do intervalo de uma forma binária de 64 bits assinada, é utilizada uma operação de módulo simétrico para colocar o valor no intervalo adequado. Considere os seguintes exemplos de valores fora do intervalo.

2^{63} torna-se -2^{63} e aparece como
0h8000000000000000 no modo base
Hex

0b100...000 (63 zeros) no modo base
Binário

2^{64} torna-se 0 e aparece como 0h0 no
modo base Hex 0b0 no modo base
Binário

$-2^{63} - 1$ torna-se $2^{63} - 1$ e aparece
como 0h7FFFFFFFFFFFFFFF no modo
base Hex 0b111...111 (64 1's) no modo
base Binário

NúmeroInteiro1 ►Base10 $\Rightarrow$ número
inteiro

0b10011►Base10	19
0h1F►Base10	31

Nota: Pode introduzir este operador
através da escrita de @►Base10 no
teclado do computador.

Converte *NúmeroInteiro1* para um
número decimal (base 10). Uma entrada
binária ou hexadecimal têm de ter
sempre um prefixo 0b ou 0h,
respectivamente.

0b *NúmeroBinário*

0h *NúmeroHexadecimal*

Zero, não a letra O, seguido por b ou h.

Um número binário pode ter até 64
dígitos. Um número hexadecimal pode
ter até 16 dígitos.

Sem um prefixo, *NúmeroInteiro1* é
tratado como decimal. O resultado
aparece em decimal,
independentemente do modo base.

NúmeroInteiro1 ►Base16 ⇒ número inteiro

256►Base16	0h100
0b111100001111►Base16	0hF0F

Nota: Pode introduzir este operador através da escrita de @►Base16 no teclado do computador.

Converte *NúmeroInteiro1* para um número hexadecimal. Os números binários ou hexadecimais têm sempre um prefixo 0b ou 0h, respectivamente.

0b *NúmeroBinário*

0h *NúmeroHexadecimal*

Zero, não a letra O, seguido por b ou h.

Um número binário pode ter até 64 dígitos. Um número hexadecimal pode ter até 16 dígitos.

Sem um prefixo, *NúmeroInteiro1* é tratado como decimal (base 10). O resultado aparece em hexadecimal, independentemente do modo base.

Se introduzir um número inteiro na base 10 muito grande para uma forma binária de 64 bits assinada, é utilizada uma operação de módulo simétrico para colocar o valor no intervalo adequado. Para mais informações, consulte ►Base2, página 19.

binomCdf()

binomCdf(*n, p*) ⇒ lista

binomCdf(*n, p, LimiteInferior, LimiteSuperior*) ⇒ número se *LimiteInferior* e *LimiteSuperior* forem números, lista se *LimiteInferior* e *LimiteSuperior* forem listas

binomCdf(*n, p, LimiteSuperior*) para $P(0 \leq X \leq \text{LimiteSuperior})$ ⇒ número se *LimiteSuperior* for um número, lista se *LimiteSuperior* for uma lista

Calcula uma probabilidade cumulativa para a distribuição binomial discreta com n número de tentativas e a probabilidade p de sucesso de cada tentativa.

Para $P(X \leq \text{LimiteSuperior})$, defina $\text{LimiteInferior}=0$

binomPdf(n, p) $\Rightarrow$ lista

binomPdf(n, p, ValX) $\Rightarrow$ número se ValX for um número, lista se ValX for uma lista

Calcula uma probabilidade para a distribuição binomial discreta com o n número de tentativas e a probabilidade p de sucesso de cada tentativa.

C

ceiling(ExprI) $\Rightarrow$ número inteiro

$\text{ceiling}(.456)$	1.
------------------------	----

Devolve o número inteiro mais próximo que é $\geq$ o argumento.

O argumento pode ser um número complexo ou real.

Nota: Consulte também **floor()**.

ceiling(ListaI) $\Rightarrow$ lista

$\text{ceiling}(\{-3.1, 1, 2.5\})$	$\{-3., 1, 3.\}$
------------------------------------	------------------

ceiling(MatrizI) $\Rightarrow$ matriz

$\text{ceiling}\left(\begin{bmatrix} 0 & -3.2 \cdot i \\ 1.3 & 4 \end{bmatrix}\right)$	$\begin{bmatrix} 0 & -3. \cdot i \\ 2. & 4 \end{bmatrix}$
--	---

Devolve uma lista ou matriz do ceiling de cada elemento.

centralDiff(*Expr1*, *Var* [= *Valor*]
[, *Passo*]) ⇒ expressão

centralDiff(*Expr1*, *Var*
[, *Passo*]) | *Var* = *Valor* ⇒ expressão

centralDiff(*Expr1*, *Var* [= *Valor*]
[, *Lista*]) ⇒ lista

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

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

Devolve a derivada numérica com a fórmula do quociente da diferença central.

Ao especificar o *Valor*, substitui qualquer atribuição de variável anterior ou qualquer substituição atual “|” para a variável.

Passo é o valor do passo. Se omitir *Passo*, preddefine-se para 0,001.

Quando utilizar *Lista1* ou *Matriz1*, a operação é mapeada através dos valores da lista ou dos elementos da matriz.

Nota: Consulte também **avgRC()** e **d()**.

$\text{centralDiff}(\cos(x), x, h)$	$\frac{-\left(\cos(x-h) - \cos(x+h)\right)}{2 \cdot h}$
$\lim_{h \rightarrow 0} \left(\text{centralDiff}(\cos(x), x, h) \right)$	$-\sin(x)$
$\text{centralDiff}(x^3, x, 0.01)$	$3 \cdot (x^2 + 0.000033)$
$\text{centralDiff}(\cos(x), x) x = \frac{\pi}{2}$	$-1.$
$\text{centralDiff}(x^2, x, \{0.01, 0.1\})$	$\{2 \cdot x, 2 \cdot x\}$

cFactor()

cFactor(*Expr1* [, *Var*]) ⇒ expressão

cFactor(*Lista1* [, *Var*]) ⇒ lista

cFactor(*Matriz1* [, *Var*]) ⇒ matriz

cFactor(*Expr1*) devolve *Expr1* decomposta em factores em relação a todas as variáveis sobre um denominador comum.

$\text{cFactor}(a^3 \cdot x^2 + a \cdot x^2 + a^3 + a \cdot x)$	$a \cdot (a^2 + 1) \cdot (x - i) \cdot (x + i)$
$\text{cFactor}\left(x^2 + \frac{4}{9}\right)$	$\frac{(3 \cdot x - 2 \cdot i) \cdot (3 \cdot x + 2 \cdot i)}{9}$
$\text{cFactor}(x^2 + 3)$	$x^2 + 3$
$\text{cFactor}(x^2 + a)$	$x^2 + a$

Expr1 é decomposta o mais possível em factores racionais lineares mesmo que isto introduza novos números não reais. Esta alternativa é adequada se quiser a factorização em relação a mais do que uma variável.

cFactor(*Expr1*, *Var*) devolve *Expr1* decomposta em factores em relação à variável *Var*.

Expr1 é decomposta o mais possível em factores que são lineares em *Var*, com talvez constantes não reais, mesmo que introduza subexpressões ou constantes irracionais que são irracionais noutras variáveis.

Os factores e os termos são ordenados com *Var* como variável principal. As potências similares de *Var* são recolhidas em cada factor. Inclua *Var* se a factorização for necessária em relação apenas a essa variável e estiver disposto a aceitar expressões irracionais em qualquer outra variável para aumentar a factorização em relação a *Var*. Pode existir alguma decomposição em factores incidental em relação a outras variáveis.

Para a definição Auto do modo **Auto ou Aproximado**, incluindo *Var*, permite também a aproximação a coeficientes de pontos flutuantes em que os coeficientes irracionais não podem ser expressos explicitamente em termos das funções integradas. Mesmo quando exista apenas uma variável, incluindo *Var*, pode produzir a factorização mais completa.

Nota: Consulte também **factor()**.

$\text{cFactor}(a^3 \cdot x^2 + a \cdot x^2 + a^3 + a \cdot x)$	$a \cdot (a^2 + 1) \cdot (x - i) \cdot (x + i)$
$\text{cFactor}(x^2 + 3, x)$	$(x + \sqrt{3} \cdot i) \cdot (x - \sqrt{3} \cdot i)$
$\text{cFactor}(x^2 + a, x)$	$(x + \sqrt{a} \cdot i) \cdot (x - \sqrt{a} \cdot i)$

$\text{cFactor}(x^5 + 4 \cdot x^4 + 5 \cdot x^3 - 6 \cdot x - 3)$	$x^5 + 4 \cdot x^4 + 5 \cdot x^3 - 6 \cdot x - 3$
$\text{cFactor}(x^5 + 4 \cdot x^4 + 5 \cdot x^3 - 6 \cdot x - 3, x)$	$(x - 0.964673) \cdot (x + 0.611649) \cdot (x + 2.12543) \cdot (x + 0.964673i) \cdot (x - 0.964673i)$

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

char(*Número inteiro*) $\Rightarrow$ *carácter*

$\text{char}(38)$	"&"
$\text{char}(65)$	"A"

Devolve uma cadeia de caracteres com o carácter numerado *Número inteiro* a partir do conjunto de caracteres da unidade portátil. O intervalo válido para o *Número inteiro* é 0–65535.

charPoly()

charPoly
(*MatrizQuadrada*, *Var*) $\Rightarrow$ expressão polinomial

charPoly
(*MatrizQuadrada*, *Expr*) $\Rightarrow$ expressão polinomial

charPoly
(*MatrizQuadrada1*, *Matriz2*) $\Rightarrow$ expressão polinomial

Devolve o polinómio característico de *MatrizQuadrada*. O polinómio característico de $n \times n$ matriz *A*, indicado por $p_A(\lambda)$, é o polinómio definido por

$$p_A(\lambda) = \det(\lambda \cdot I - A)$$

em que *I* indica a matriz identidade $n \times n$.

MatrizQuadrada1 e *MatrizQuadrada2* têm de ter as dimensões iguais.

$m := \begin{bmatrix} 1 & 3 & 0 \\ 2 & -1 & 0 \\ -2 & 2 & 5 \end{bmatrix}$	$\begin{bmatrix} 1 & 3 & 0 \\ 2 & -1 & 0 \\ -2 & 2 & 5 \end{bmatrix}$
$\text{charPoly}(m, x)$	$-x^3 + 5 \cdot x^2 + 7 \cdot x - 35$
$\text{charPoly}(m, x^2 + 1)$	$-x^6 + 2 \cdot x^4 + 14 \cdot x^2 - 24$
$\text{charPoly}(m, m)$	0

 χ^2 2way

χ^2 2way *MatrizObs*

chi22way *MatrizObs*

Calcula um teste χ^2 para associação à tabela de contagens bidireccional na matriz observada *MatrizObs*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Para mais informações sobre o efeito dos elementos vazios numa matriz, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat. χ^2	Estatística do Qui quadrado: soma (observada - prevista) ² / prevista
stat.PVal	Menor nível de significância para o qual a hipótese nula pode ser rejeitada
stat.df	Graus de liberdade para a estatística do Qui quadrado
stat.ExpMat	Matriz da tabela de contagem de elementos previsto, assumindo a hipótese nula
stat.CompMat	Matriz de contribuições da estatística do Qui quadrado dos elementos

χ^2 Cdf()

Catálogo > 

χ^2 Cdf

(LimiteInferior, LimiteSuperior, df) ⇒ número se LimiteInferior e LimiteSuperior forem números, lista se LimiteInferior e LimiteSuperior forem listas

chi2Cdf

(LimiteInferior, LimiteSuperior, df) ⇒ número se LimiteInferior e LimiteSuperior forem números, lista se LimiteInferior e LimiteSuperior forem listas

Calcula a probabilidade de distribuição χ^2 entre *LimiteInferior* e *LimiteSuperior* para os graus de liberdade especificados *df*.

Para $P(X \leq \text{LimiteSuperior})$, defina *LimiteInferior* = 0.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

χ^2 GOF

Catálogo > 

χ^2 GOF *Lista obs, Lista exp, df*

chi2GOF *Lista obs, Lista exp, df*

Efectua um teste para confirmar que os dados da amostra são de uma população que está em conformidade com uma distribuição especificada. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
$\text{stat.}\chi^2$	Estatística do Qui quadrado: $\text{soma}((\text{observada} - \text{prevista})^2 / \text{prevista})$
stat.PVal	Menor nível de significância para o qual a hipótese nula pode ser rejeitada
stat.df	Graus de liberdade para a estatística do Qui quadrado
stat.CompList	Matriz de contribuições da estatística do Qui quadrado dos elementos

 χ^2 Pdf()

$\chi^2\text{Pdf}(ValX, df) \Rightarrow$ número se *ValX* for um número, lista se *ValX* for uma lista

$\text{chi2Pdf}(ValX, df) \Rightarrow$ número se *ValX* for um número, lista se *ValX* for uma lista

Calcula a função de densidade de probabilidade (pdf) para a distribuição χ^2 num valor *ValX* especificado para os graus de liberdade especificados *df*.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

ClearAZ**ClearAZ**

Apaga todas as variáveis de um carácter no espaço do problema actual.

$5 \rightarrow b$	5
<i>b</i>	5
ClearAZ	Done
<i>b</i>	<i>b</i>

Se uma ou mais variáveis estiverem bloqueadas, este comando mostra uma mensagem de erro e só elimina as variáveis desbloqueadas. Consulte **unLock**, página 216.

ClrErr

Catálogo >

ClrErr

Para ver um exemplo de **ClrErr**, consulte o exemplo 2 no comando **Try**, página 209.

Apaga o estado de erro e define a variável do sistema *errCode* para zero.

A proposição **Else** do bloco **Try...Else...EndTry** deve utilizar **ClrErr** ou **PassErr**. Se tiver de processar ou ignorar o erro, utilize **ClrErr**. Se não souber o que fazer com o erro, utilize **PassErr** para o enviar para a rotina de tratamento de erros seguinte. Se não existirem mais rotinas de tratamento de erros **Try...Else...EndTry** pendente, a caixa de diálogo de erros aparecerá como normal.

Nota: Consulte também **PassErr**, página 144, e **Try**, página 209.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

colAugment()

Catálogo >

colAugment(*Matriz1*, *Matriz2*) $\Rightarrow$ *matriz*

Devolve uma nova lista que é a *Matriz2* acrescentada ao fim da *Matriz1*. As matrizes têm de ter dimensões de colunas iguais, e a *Matriz2* é acrescentada à *Matriz1* como novas colunas. Não altere *Matriz1* ou *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()

Catálogo >

colDim(*Matriz*) $\Rightarrow$ *expressão*

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

Devolve o número de colunas contidas em *Matriz*.

Nota: Consulte também `rowDim()`.

colNorm()

colNorm(*Matriz*) $\Rightarrow$ expressão

Devolve o máximo das somas dos valores absolutos dos elementos nas colunas em *Matriz*.

Nota: Os elementos da matriz indefinidos não são permitidos. Consulte também `rowNorm()`.

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

comDenom()

comDenom(*Expr1* [, *Var*]) $\Rightarrow$ expressão

comDenom(*Lista1* [, *Var*]) $\Rightarrow$ lista

comDenom(*Matriz1* [, *Var*]) $\Rightarrow$ matriz

$$\text{comDenom} \left(\frac{y^2+y}{(x+1)^2} + y^2+y \right)$$

$$\frac{x^2 \cdot y^2 + x^2 \cdot y + 2 \cdot x \cdot y^2 + 2 \cdot x \cdot y + 2 \cdot y^2 + 2 \cdot y}{x^2 + 2 \cdot x + 1}$$

comDenom(*Expr1*) devolve uma fracção simplificada com um numerador completamente expandido sobre um denominador completamente expandido.

comDenom(*Expr1*, *Var*) devolve um rácio reduzido do numerador e do denominador expandidos em relação a *Var*. Os termos e os factores são ordenados com *Var* como variável principal. As potências similares de *Var* são recolhidas. Pode existir alguma decomposição em factores incidental dos coeficientes recolhidos. Comparada para omitir *Var*, esta poupa tempo frequentemente, memória e espaço no ecrã, enquanto torna a expressão mais compreensível. Torna também as operações subsequentes no resultado mais rápidas e poupa a memória.

$$\text{comDenom} \left(\frac{y^2+y}{(x+1)^2} + y^2+y, x \right)$$

$$\frac{x^2 \cdot y \cdot (y+1) + 2 \cdot x \cdot y \cdot (y+1) + 2 \cdot y \cdot (y+1)}{x^2 + 2 \cdot x + 1}$$

$$\text{comDenom} \left(\frac{y^2+y}{(x+1)^2} + y^2+y, y \right)$$

$$\frac{y^2 \cdot (x^2 + 2 \cdot x + 2) + y \cdot (x^2 + 2 \cdot x + 2)}{x^2 + 2 \cdot x + 1}$$

Se *Var* não ocorrer em *Expr1*, **comDenom**(*Expr1*, *Var*) devolve uma fracção simplificada com um numerador não expandido sobre um denominador não expandido. Estes resultados poupam geralmente mais tempo, memória e espaço no ecrã. Estes resultados decompostos parcialmente tornam também as operações subsequentes no resultado mais rápidas e poupam a memória.

Mesmo quando não exista um denominador, a função **comden** é frequentemente uma forma rápida para alcançar a factorização parcial se **factor()** for muito lento ou se esgotar a memória.

Sugestão: Introduza esta definição da função **comden()** e experimente-a rotinamente como uma alternativa para **comDenom()** e **factor()**.

Define *comden*(*exprn*)=**comDenom**(*exprn*,*abc*)

Done

$$\text{comden}\left(\frac{y^2+y}{(x+1)^2}+y^2+y\right) = \frac{(x^2+2\cdot x+2)\cdot y\cdot (y+1)}{(x+1)^2}$$

$$\text{comden}(1234\cdot x^2\cdot (y^3-y)+2468\cdot x\cdot (y^2-1))$$

$$1234\cdot x\cdot (x\cdot y+2)\cdot (y^2-1)$$

completeSquare ()

completeSquare(*ExprOrEqn*, *Var*) $\Rightarrow$ expressão ou equação

completeSquare(*ExprOrEqn*, *Var*^{*Power*}) $\Rightarrow$ expressão ou equação

completeSquare(*ExprOrEqn*, *Var1*, *Var2* [...]) $\Rightarrow$ expressão ou equação

completeSquare(*ExprOrEqn*, {*Var1*, *Var2* [...]}) $\Rightarrow$ expressão ou equação

Converte uma expressão polinomial quadrática da forma $a\cdot x^2+b\cdot x+c$ para a forma $a\cdot (x-h)^2+k$

ou

Converte uma equação do 2º grau da forma $a\cdot x^2+b\cdot x+c=d$ para a forma $a\cdot (x-h)^2=k$

$$\text{completeSquare}(x^2+2\cdot x+3,x) = (x+1)^2+2$$

$$\text{completeSquare}(x^2+2\cdot x=3,x) = (x+1)^2=4$$

$$\text{completeSquare}(x^6+2\cdot x^3+3,x^3) = (x^3+1)^2+2$$

$$\text{completeSquare}(x^2+4\cdot x+y^2+6\cdot y+3=0,x,y) = (x+2)^2+(y+3)^2=10$$

$$\text{completeSquare}(3\cdot x^2+2\cdot y+7\cdot y^2+4\cdot x=3,\{x,y\}) = 3\cdot \left(x+\frac{2}{3}\right)^2+7\cdot \left(y+\frac{1}{7}\right)^2=\frac{94}{21}$$

$$\text{completeSquare}(x^2+2\cdot x\cdot y,x,y) = (x+y)^2-y^2$$

O primeiro argumento tem de ser uma expressão quadrática ou equação na forma padrão, em relação ao segundo argumento.

O segundo argumento tem de ser um único termo de uma só variável ou um único termo de uma só variável elevado a uma potência racional, por exemplo x , y^2 ou $z^{(1/3)}$.

A terceira e quarta expressões de sintaxe para concluir o quadrado nas variáveis *Var1*, *Var2* [...]).

conj()

conj(*Expr1*) $\Rightarrow$ expressão

conj(*Lista1*) $\Rightarrow$ lista

conj(*Matriz1*) $\Rightarrow$ matriz

Devolve o conjugado complexo do argumento.

Nota: Todas as variáveis indefinidas são tratadas como variáveis reais.

$\text{conj}(1+2 \cdot i)$	$1-2 \cdot i$
$\text{conj}\left(\begin{bmatrix} 2 & 1-3 \cdot i \\ -i & -7 \end{bmatrix}\right)$	$\begin{bmatrix} 2 & 1+3 \cdot i \\ i & -7 \end{bmatrix}$
$\text{conj}(z)$	$\bar{z}$
$\text{conj}(x+i \cdot y)$	$x-y \cdot i$

constructMat()

constructMat
(*Expr*, *Var1*, *Var2*, *NúmLinhas*,
NúmColunas) $\Rightarrow$ matriz

Devolve uma matriz de acordo com os argumentos.

Expr é uma expressão nas variáveis *Var1* e *Var2*. Os elementos da matriz resultante são formados através da avaliação de *Expr* para cada valor incrementado de *Var1* e *Var2*.

Var1 é incrementada automaticamente de 1 a *NúmLinhas*. Em cada linha, *Var2* é incrementada de 1 a *NúmColunas*.

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

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

CopyVar *Var1*, *Var2* copia o valor da variável *Var1* à variável *Var2*, criando *Var2*, se for necessário. A variável *Var1* tem de ter um valor.

Se *Var1* for o nome de uma função definida pelo utilizador existente, copia a definição dessa função para a função *Var2*. A função *Var1* tem de ser definida.

Var1 tem de cumprir os requisitos de nomeação de variáveis ou tem de ser uma expressão indirecta que se simplifica para um nome de variável que cumpra os requisitos.

CopyVar *Var1*., *Var2*. copia todos os membros da *Var1*. grupo de variáveis para a *Var2*. grupo, criando *Var2*. se for necessário.

Var1. tem de ser o nome de um grupo de variáveis existentes, como, por exemplo, o da estatística *stat.nn* resultados ou variáveis criados com a função **LibShortcut()**. Se *Var2*. já existe, este comando substitui todos os membros comuns a ambos os grupos e adiciona os membros que já não existam. Se um ou mais membros de *Var2*. estiverem bloqueados, todos os membros de *Var2*. ficam inalteráveis.

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

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

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

Calcula a matriz de correlação para a matriz aumentada [*Lista1*, *Lista2*, ..., *Lista20*].

pcos*Expr* ▶**cos**

Nota: Pode introduzir este operador através da escrita de `@>cos` no teclado do computador.

$$\left(\sin(x)\right)^2 \rightarrow \cos \quad 1 - \left(\cos(x)\right)^2$$

Representa *Expr* em função do co-seno. Este é um operador de conversão. Apenas pode ser utilizado no fim da linha de entrada.

cos reduz todas as potências de $\sin(\dots)$ módulo $1 - \cos(\dots)^2$ para quaisquer polinómios residuais de potências de $\cos(\dots)$ tenham expoentes no intervalo $[0, 2]$. Por conseguinte, o resultado ficará livre de $\sin(\dots)$ se e só se $\sin(\dots)$ ocorrer na expressão fornecida apenas em potências pares.

Nota: Este operador de conversão não é suportado nos modos de ângulos Graus ou Grados. Antes de o utilizar, certifique-se de que o modo Ângulo está definido para Radianos e que *Expr* não contém referências explícitas a ângulos em graus ou grados.

cos()

Tecla

cos(Expr1) ⇒ expressão

No modo de ângulo Graus:

cos(Lista1) ⇒ lista

$$\cos\left(\frac{\pi}{4}\right) \quad \frac{\sqrt{2}}{2}$$

cos(Expr1) devolve o co-seno do argumento como uma expressão.

$$\cos(45) \quad \frac{\sqrt{2}}{2}$$

cos(Lista1) devolve uma lista de co-senos de todos os elementos na *Lista1*.

$$\cos(\{0,60,90\}) \quad \left\{1, \frac{1}{2}, 0\right\}$$

Nota: O argumento é interpretado como um ângulo express em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual. Pode utilizar °, G ou r para substituir o modo de ângulo temporariamente.

No modo de ângulo Gradianos:

$$\cos(\{0,50,100\}) \quad \left\{1, \frac{\sqrt{2}}{2}, 0\right\}$$

No modo de ângulo Radianos:

$\cos\left(\frac{\pi}{4}\right)$	$\frac{\sqrt{2}}{2}$
$\cos(45^\circ)$	$\frac{\sqrt{2}}{2}$

cos(MatrizQuadradaI) $\Rightarrow$ Matriz quadrada

Devolve o co-seno da matriz da *MatrizQuadradaI*. Isto não é o mesmo que calcular o co-seno de cada elemento.

Quando uma função escalar $f(A)$ operar na *MatrizQuadradaI* (A), o resultado é calculado pelo algoritmo:

Calcule os valores próprios (λ_i) e os vectores próprios (V_i) de A.

MatrizQuadradaI tem de ser diagonalizável. Também não pode ter variáveis simbólicas sem um valor.

Forme as matrizes:

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

$A = X B X^{-1}$ e $f(A) = X f(B) X^{-1}$. Por exemplo, $\cos(A) = X \cos(B) X^{-1}$ em que:

$\cos(B) =$

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

Todos os cálculos são efectuados com a aritmética de ponto flutuante.

No modo de ângulo Radianos:

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

cos⁻¹(ExprI) $\Rightarrow$ expressão

No modo de ângulo Graus:

cos⁻¹(ListaI) $\Rightarrow$ lista

$\cos^{-1}(1)$	0
----------------	---

$\cos^{-1}(Expr1)$ devolve o ângulo cujo co-seno é *Expr1* como uma expressão.

$\cos^{-1}(Lista1)$ devolve uma lista de co-senos inversos de cada elemento de *Lista1*.

Nota: O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

Nota: Pode introduzir esta função através da escrita de **arccos (...)** no teclado.

$\cos^{-1}(MatrizQuadrada1) \Rightarrow$ *Matriz quadrada*

Devolve o co-seno inverso da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o co-seno inverso de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Gradianos:

$$\cos^{-1}(0) \quad 100$$

No modo de ângulo Radianos:

$$\cos^{-1}(\{0,0.2,0.5\}) \quad \left\{ \frac{\pi}{2}, 1.36944, 1.0472 \right\}$$

No modo de ângulo Radianos e Formato complexo rectangular:

$$\cos^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \quad \begin{bmatrix} 1.73485+0.064606 \cdot i & -1.49086+2.10514 \\ -0.725533+1.51594 \cdot i & 0.623491+0.77836 \cdot i \\ -2.08316+2.63205 \cdot i & 1.79018-1.27182 \cdot i \end{bmatrix}$$

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

$\cosh(Expr1) \Rightarrow$ *expressão*

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

$\cosh(Expr1)$ devolve o co-seno hiperbólico do argumento como uma expressão.

$\cosh(Lista1)$ devolve uma lista dos co-senos hiperbólicos de cada elemento de *Lista1*.

$\cosh(MatrizQuadrada1) \Rightarrow$ *Matriz quadrada*

No modo de ângulo Graus:

$$\cosh\left(\left(\frac{\pi}{4}\right)_r\right) \quad \cosh(45)$$

No modo de ângulo Radianos:

cosh()

Catálogo > 

Devolve o co-seno hiperbólico da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o co-seno hiperbólico de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

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

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

cosh⁻¹()

Catálogo > 

cosh⁻¹(Expr1) ⇒ expressão

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

cosh⁻¹(Lista1) ⇒ lista

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

cosh⁻¹(Expr1) devolve o co-seno hiperbólico inverso do argumento como uma expressão.

cosh⁻¹(Lista1) devolve uma lista dos co-senos hiperbólicos inversos de cada elemento de *Lista1*.

Nota: Pode introduzir esta função através da escrita de **arccosh (...)** no teclado.

cosh⁻¹(MatrizQuadrada1) ⇒ *Matriz quadrada*

Devolve o co-seno hiperbólico inverso da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o co-seno hiperbólico inverso de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Radianos e Formato complexo rectangular:

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

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

cot()

Tecla 

cot(Expr1) ⇒ expressão

No modo de ângulo Graus:

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

Devolve a co-tangente de *Expr1* ou devolve uma lista das co-tangentes de todos os elementos em *List1*.

Nota: O argumento é interpretado como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual. Pode utilizar °, G ou R para substituir o modo de ângulo temporariamente.

Nota: Pode introduzir esta função através da escrita de **arccot (...)** no teclado.

$\cot(45)$	1
------------	---

No modo de ângulo Gradianos:

$\cot(50)$	1
------------	---

No modo de ângulo Radianos:

$\cot(\{1, 2, 1, 3\})$	$\left\{ \frac{1}{\tan(1)}, 0.584848, \frac{1}{\tan(3)} \right\}$
------------------------	---

cot⁻¹()Tecla **cot⁻¹(Expr1)** $\Rightarrow$ expressão**cot⁻¹(Lista1)** $\Rightarrow$ lista

Devolve o ângulo cuja co-tangente é *Expr1* ou devolve uma lista com as co-tangentes inversas de cada elemento de *List1*.

Nota: O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

No modo de ângulo Graus:

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

No modo de ângulo Gradianos:

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

No modo de ângulo Radianos:

$\cot^{-1}(1)$	$\frac{\pi}{4}$
----------------	-----------------

coth()Catálogo > **coth(Expr1)** $\Rightarrow$ expressão**coth(Lista1)** $\Rightarrow$ lista

Devolve a co-tangente hiperbólica de *Expr1* ou devolve uma lista das co-tangentes hiperbólicas de todos os elementos de *List1*.

$\coth(1.2)$	1.19954
--------------	---------

$\coth(\{1, 3, 2\})$	$\left\{ \frac{1}{\tanh(1)}, 1.00333 \right\}$
----------------------	--

coth⁻¹()**Catálogo >** **coth⁻¹(Expr1)** ⇒ expressãocoth⁻¹(3,5) 0.293893**coth⁻¹(Lista1)** ⇒ listacoth⁻¹({-2,2,1,6})

Devolve a co-tangente hiperbólica inversa de *Expr1* ou devolve uma lista com as co-tangentes hiperbólicas inversas de cada elemento de *Lista1*.

$$\left\{ \frac{-\ln(3)}{2}, 0.518046, \frac{\ln\left(\frac{7}{5}\right)}{2} \right\}$$

Nota: Pode introduzir esta função através da escrita de **arccoth** (...) no teclado.

count()**Catálogo >** **count(Valor1ouLista1 [, Valor2ouLista2 [...]])** ⇒ valor

count(2,4,6) 3

count({2,4,6}) 3

Devolve a contagem acumulada de todos os elementos nos argumentos que se avaliam para valores numéricos.

count(2,{4,6},{8 10; 12 14}) 7

Cada argumento pode ser uma expressão, valor, lista ou matriz. Pode misturar tipos de dados e utilizar argumentos de várias dimensões.

count(1/2, 3+4*i, undef, "hello", x+5, sign(0)) 2

Para uma lista, matriz ou intervalo de dados, cada elemento é avaliado para determinar se deve ser incluído na contagem.

No último exemplo, apenas 1/2 e 3+4* i são contados. Os restantes argumentos, partindo do princípio que *x* é indefinido, não se avaliam para valores numéricos.

Na aplicação Listas e Folha de cálculo, pode utilizar um intervalo de células no lugar de qualquer argumento.

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

countif()**Catálogo >** **countif(Lista, Critérios)** ⇒ valor

countif({1,3,"abc",undef,3,1},3) 2

Devolve a contagem acumulada de todos os elementos em *Lista* que cumpram os *critérios* especificados.

Conta o número de elementos igual a 3.

Critérios podem ser:

- Um valor, uma expressão ou uma cadeia. Por exemplo, 3 conta apenas aqueles elementos em *Lista* que se simplificam para o valor 3.
- Uma expressão booleana com o símbolo ? como um identificador para cada elemento. Por exemplo, ?<5 conta apenas aqueles elementos em *Lista* inferiores a 5.

Na aplicação Listas e Folha de cálculo, pode utilizar um intervalo de células no lugar de *Lista*.

Os elementos (nulos) vazios da lista são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

Nota: Consulte também **sumif()**, página 198 e **frequency()**, página 81.

$\text{countIf}(\{ "abc", "def", "abc", 3 \}, "def")$	1
---	---

Conta o número de elementos igual a "def."

$\text{countIf}(\{ x^{-2}, x^{-1}, 1, x, x^2 \}, x)$	1
--	---

Conta o número de elementos igual a x; este exemplo assume que a variável x é indefinida.

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

Conta 1 e 3.

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

Conta 3, 5, e 7.

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

Conta 1, 3, 7 e 9.

cPolyRoots()

cPolyRoots(Poli, Var) ⇒ lista

cPolyRoots(ListaDeCoeficientes) ⇒ lista

A primeira sintaxe, **cPolyRoots(Poly, Var)**, devolve uma lista de raízes complexas do polinómio *Poly* na variável *Var*.

Poly tem de ser um polinómio numa variável.

A segunda sintaxe, **cPolyRoots(ListaDeCoeficientes)**, devolve uma lista de raízes complexas para os coeficientes em *ListaDeCoeficientes*.

Nota: Consulte também **polyRoots()**, página 149.

$\text{polyRoots}(y^3 + 1, y)$	$\{-1\}$
--------------------------------	----------

$\text{cPolyRoots}(y^3 + 1, y)$	$\left\{ -1, \frac{1}{2} - \frac{\sqrt{3}}{2}i, \frac{1}{2} + \frac{\sqrt{3}}{2}i \right\}$
---------------------------------	---

$\text{polyRoots}(x^2 + 2 \cdot x + 1, x)$	$\{-1, -1\}$
--	--------------

$\text{cPolyRoots}(\{1, 2, 1\})$	$\{-1, -1\}$
----------------------------------	--------------

crossP(Lista1, Lista2) ⇒ lista

Devolve o produto cruzado de *Lista1* e *Lista2* como uma lista.

Lista1 e *Lista2* têm de ter dimensões iguais e a dimensão tem de ser 2 ou 3.

crossP(Vector1, Vector2) ⇒ vector

Devolve um vector da linha ou coluna (dependendo dos argumentos) que é o produto cruzado de *Vector1* e *Vector2*.

Vector1 e *Vector2* têm de ser vectores de linhas ou ambos têm de ser vectores de colunas. Ambos os vectores têm de ter dimensões iguais e a dimensão tem de ser 2 ou 3.

$$\text{crossP}(\{a1, b1\}, \{a2, b2\})$$

$$\{0, 0, a1 \cdot b2 - a2 \cdot b1\}$$

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

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

$$\text{crossP}(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \begin{bmatrix} -3 & 6 & -3 \\ 0 & 0 & -2 \end{bmatrix})$$

$$\begin{bmatrix} -3 & 6 & -3 \\ 0 & 0 & -2 \end{bmatrix}$$

csc()Tecla **csc(Expr1) ⇒ expressão**

No modo de ângulo Graus:

csc(Lista1) ⇒ lista

Devolve a co-secante de *Expr1* ou devolve uma lista com as co-secantes de todos os elementos em *Lista1*.

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

$$\sqrt{2}$$

No modo de ângulo Gradianos:

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

$$\sqrt{2}$$

No modo de ângulo Radianos:

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

$$\left\{\frac{1}{\sin(1)}, 1, \frac{2\sqrt{3}}{3}\right\}$$

csc⁻¹()Tecla **csc⁻¹(Expr1) ⇒ expressão**

No modo de ângulo Graus:

csc⁻¹(Lista1) ⇒ lista

$$\text{csc}^{-1}(1)$$

$$90.$$

Devolve o ângulo cuja co-secante é *Expr1* ou devolve uma lista com as co-secantes inversas de cada elemento de *Lista1*.

No modo de ângulo Gradianos:

csc⁻¹()**Tecla** 

Nota: O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

Nota: Pode introduzir esta função através da escrita de **arccsc (...)** no teclado.

csc⁻¹(1)

100.

No modo de ângulo Radianos:

$$\text{csc}^{-1}\left(\left\{1, 4, 6\right\}\right) \quad \left\{\frac{\pi}{2}, \sin^{-1}\left(\frac{1}{4}\right), \sin^{-1}\left(\frac{1}{6}\right)\right\}$$

csch()**Catálogo >** **csch(Expr1)** ⇒ expressão**csch(Lista1)** ⇒ lista

Devolve a co-secante hiperbólica de *Expr1* ou devolve uma lista das co-secantes hiperbólicas de todos os elementos de *Lista1*.

csch(3)

$$\frac{1}{\sinh(3)}$$

csch({1,2,1,4})

$$\left\{\frac{1}{\sinh(1)}, 0.248641, \frac{1}{\sinh(4)}\right\}$$

csch⁻¹()**Catálogo >** **csch⁻¹(Expr1)** ⇒ expressão**csch⁻¹(Lista1)** ⇒ lista

Devolve a co-secante hiperbólica inversa de *Expr1* ou devolve uma lista com as co-secantes hiperbólicas inversas de cada elemento de *Lista1*.

Nota: Pode introduzir esta função através da escrita de **arccsch (...)** no teclado.

csch⁻¹(1)sinh⁻¹(1)csch⁻¹({1,2,1,3})

$$\left\{\sinh^{-1}(1), 0.459815, \sinh^{-1}\left(\frac{1}{3}\right)\right\}$$

cSolve()**Catálogo >** **cSolve(Equação, Var)** ⇒ Expressão booleana**cSolve(Equação, Var=Tentativa)** ⇒ Expressão booleana**cSolve(Desigualdade, Var)** ⇒ Expressão booleanacSolve(x³=-1,x)

$$x = \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i \text{ or } x = \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i \text{ or } x = -1$$

solve(x³=-1,x)

x=-1

Devolve as soluções complexas candidatas de uma equação ou desigualdade para *Var*. O objectivo é produzir candidatos para todas as soluções reais e não reais. Mesmo que *Equação* seja real, **cSolve()** permite resultados não reais no Formato complexo de resultados reais.

cSolve() define temporariamente o domínio para complexo durante a resolução mesmo que o domínio actual seja real. No domínio complexo, as potências fraccionárias que tenham denominadores ímpares utilizam o principal em vez da derivação real. Consequentemente, as soluções de **solve()** para equações que envolvam essas potências fraccionárias não são necessariamente um subconjunto dessas do **cSolve()**.

cSolve() começa com os métodos simbólicos exactos. **cSolve()** utiliza também a decomposição polinomial complexa iterativa, se for necessária.

Nota: Consulte também **cZeros()**, **solve()** e **zeros()**.

cSolve(*Eqn1* and *Eqn2* [and...],
VarOuTentativa1, *VarOuTentativa2* [,
...]) ⇒ *Expressão booleana*

cSolve(*SistemaDeEquações*,
VarOuTentativa1, *VarOuTentativa2* [,
...]) ⇒ *Expressão booleana*

Devolve soluções complexas candidatas para as equações algébricas simultâneas, em que cada *VarOuTentativa* especifica uma variável que quer resolver.

$\text{cSolve}\left(x^{\frac{1}{3}}=1,x\right)$	false
$\text{solve}\left(x^{\frac{1}{3}}=1,x\right)$	$x=-1$

No modo de visualização de dígitos de Fix 2:

$\text{exact}\left(\text{cSolve}\left(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3=0,x\right)\right)$
$x\cdot\left(x^4+4\cdot x^3+5\cdot x^2-6\right)=3$
$\text{cSolve}\left(\text{Ans},x\right)$
$x=-1.11+1.07\cdot i$ or $x=-1.11-1.07\cdot i$ or $x=2.18$

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

Opcionalmente, pode especificar uma tentativa inicial para uma variável. Cada *varOuTentativa* tem de ter a forma:

variável

– ou –

variável = *número real ou não real*

Por exemplo, x é válido e logo é $x=3+i$.

Se todas as equações forem polinomiais e se não especificar qualquer tentativa inicial, **cSolve()** utiliza o método de eliminação lexical Gröbner/Buchberger para tentar determinar **todas as** soluções complexas.

As soluções complexas podem incluir soluções reais e não reais, como no exemplo à direita.

$$\text{cSolve}\left(u \cdot v - u = v \text{ and } v^2 = -u, \{u, v\}\right) \\ u = \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i \text{ and } v = \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i \text{ or } u = \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i$$

Para ver o resultado completo, prima **►** e, de seguida, utilize **◀** e **►** para mover o cursor.

As equações polinomiais simultâneas podem ter variáveis adicionais que não tenham valores, mas representam os valores numéricos dados que possam ser substituídos posteriormente.

$$\text{cSolve}\left(u \cdot v - u = c \cdot v \text{ and } v^2 = -u, \{u, v\}\right) \\ u = \frac{-(\sqrt{4 \cdot c - 1} \cdot i + 1)^2}{4} \text{ and } v = \frac{\sqrt{4 \cdot c - 1} \cdot i + 1}{2}$$

Pode também incluir variáveis de soluções que não aparecem nas equações. Estas soluções mostram como as famílias de soluções podem conter constantes arbitrárias da forma $c \cdot k$, em que k é um sufixo com valor inteiro de 1 a 255.

$$\text{cSolve}\left(u \cdot v - u = v \text{ and } v^2 = -u, \{u, v, w\}\right) \\ u = \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i \text{ and } v = \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i \text{ and } w = c \cdot k \text{ or } \dots$$

Para sistemas polinomiais, o tempo de cálculo ou o esgotamento da memória podem depender fortemente da ordem em que liste as variáveis das soluções. Se a escolha inicial esgotar a memória ou a sua paciência, tente reorganizar as variáveis nas equações e/ou na lista *varOuTentativa*.

Se não incluir nenhuma tentativa e se a equação for não polinomial em qualquer variável, mas todas as equações forem lineares em todas as variáveis da solução, **cSolve()** utiliza a eliminação Gaussiana para tentar determinar todas as soluções.

$$\text{cSolve}\left(u+v=e^w \text{ and } u-v=i, \{u,v\}\right)$$

$$u=\frac{e^w+i}{2} \text{ and } v=\frac{e^w-i}{2}$$

Se um sistema não for polinomial em todas as variáveis nem linear nas variáveis das soluções, **cSolve()** determina no máximo uma solução com um método iterativo aproximado. Para o fazer, o número de variáveis de soluções tem de ser igual ao número de equações e todas as outras variáveis nas equações têm de ser simplificadas para números.

$$\text{cSolve}\left(e^z=w \text{ and } w=z^2, \{w,z\}\right)$$

$$w=0.494866 \text{ and } z=0.703467$$

Uma tentativa não real é frequentemente necessária para determinar uma solução não real. Para convergência, uma tentativa pode ter de ficar próxima a uma solução.

$$\text{cSolve}\left(e^z=w \text{ and } w=z^2, \{w,z=1+i\}\right)$$

$$w=0.149606+4.8919 \cdot i \text{ and } z=1.58805+1.5402 \cdot i$$

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

CubicReg

CubicReg *X*, *Y*, [*Freq*] [, *Categoria*, *Incluir*]

Calcula a regressão polinomial cúbica $y = a \cdot x^3 + b \cdot x^2 + c \cdot x + d$ a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

X e *Y* são listas de variáveis independentes e dependentes.

Freq é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros ≥ 0 .

Categoria é uma lista de códigos de categorias para os dados X e Y correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $a \cdot x^3 + b \cdot x^2 + c \cdot x + d$
stat.a, stat.b, stat.c, stat.d	Coeficientes de regressão
stat.R ²	Coeficiente de determinação
stat.Resid	Resíduos da regressão
stat.XReg	Lista de pontos de dados na <i>Lista X</i> modificada utilizada na regressão com base em restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.YReg	Lista de pontos de dados na <i>Lista Y</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.FreqReg	Lista de frequências correspondentes a <i>stat.XReg</i> e <i>stat.YReg</i>

cumulativeSum()

cumulativeSum(Lista1) ⇒ lista

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

Devolve uma lista das somas acumuladas dos elementos em *Lista1*, começando no elemento 1.

cumulativeSum(Matriz1) ⇒ matriz

Devolve uma matriz das somas cumulativas dos elementos em *Matriz1*. Cada elemento é a soma cumulativa da coluna de cima a baixo.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
$\text{cumulativeSum}(m1)$	$\begin{bmatrix} 1 & 2 \\ 4 & 6 \\ 9 & 12 \end{bmatrix}$

Um elemento (nulo) vazio em *Lista1* ou em *Matriz1* produz um elemento nulo na matriz ou lista resultante. Para mais informações sobre os elementos vazios, consulte página 274.

Cycle

Catálogo >

Cycle

Lista de funções que soma os números inteiros de 1 a 100 ignorando 50.

Transfere o controlo imediatamente para a iteração seguinte do ciclo actual (**For**, **While** ou **Loop**).

Cycle não é permitido fora das três estruturas em espiral (**For**, **While** ou **Loop**).

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

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

Catálogo >

Vector ►Cylind

$\begin{bmatrix} 2 & 2 & 3 \end{bmatrix}$ ►Cylind	$\begin{bmatrix} 2\sqrt{2} & \angle \frac{\pi}{4} & 3 \end{bmatrix}$
---	--

Nota: Pode introduzir este operador através da escrita de **@>Cylind** no teclado do computador.

Apresenta o vector da linha ou coluna em forma cilíndrica $[r, \angle\theta, z]$.

Vector tem de ter exactamente três elementos. Pode ser uma linha ou coluna.

cZeros()

Catálogo >

cZeros(*Expr*, *Var*) $\Rightarrow$ lista

No modo de visualização de dígitos de Fix 3:

$cZeros(x^5+4 \cdot x^4+5 \cdot x^3-6 \cdot x-3,x)$
$\{-1.114+1.073 \cdot i, -1.114-1.073 \cdot i, -2.125, -0.612, 0\}$

Devolve uma lista de valores reais ou não reais candidatos de *Var* que torna *Expr* =0. **cZeros()** faz isto, calculando **explist(cSolve(Expr =0, Var), Var)**. Caso contrário, **cZeros()** é similar a **zeros()**.

Nota: Consulte também **cSolve()**, **solve()** e **zeros()**.

cZeros{*Expr1*, *Expr2* [, ...] }, {*VarOuTentativa1*, *VarOuTentativa2* [, ...] } ⇒ *matriz*

Devolve posições candidatas em que as expressões são zero simultaneamente. Cada *VarOuTentativa* especifica um desconhecido cujo valor procura.

Opcionalmente, pode especificar uma tentativa inicial para uma variável. Cada *VarOuTentativa* tem de ter a forma:

variável

– ou –

variável = número real ou não real

Por exemplo, *x* é válido e logo é *x=3+ i*.

Se todas as expressões forem polinomiais e não especificar qualquer tentativa inicial, **cZeros()** utiliza o método de eliminação Gröbner/Buchberger lexical para tentar para determinar **todos** os zeros complexos.

Os zeros complexos podem incluir os zeros reais e não reais, como no exemplo à direita.

Cada linha da matriz resultante representa um zero alternativo com os componentes ordenados da mesma forma que na lista *VarOuTentativa*. Para extrair uma linha, indexe a matriz por [*linha*].

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

$$\text{cZeros}\left(\left\{u \cdot v - u - v, v^2 + u\right\}, \left\{u, v\right\}\right) = \begin{bmatrix} \frac{0}{2} - \frac{\sqrt{3}}{2} \cdot i & \frac{0}{2} + \frac{\sqrt{3}}{2} \cdot i \\ \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i & \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i \end{bmatrix}$$

Extrair linha 2:

$$\text{Ans}[2] = \left[\frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i, \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i \right]$$

Os polinomiais simultâneos podem ter variáveis adicionais sem valores, mas representam valores numéricos dados que podem ser substituídos posteriormente.

Pode também incluir variáveis desconhecidas que não aparecem nas expressões. Estes zeros mostram como as famílias de zeros podem conter constantes arbitrárias da forma $c \cdot k$, em que k é um sufixo com valor inteiro de 1 a 255.

Para sistemas polinomiais, o tempo de cálculo ou o esgotamento da memória podem depender fortemente da ordem em que liste os desconhecidos. Se a escolha inicial esgotar a memória ou a sua paciência, tente reorganizar as variáveis nas expressões e/ou na lista *VarOuTentativa*.

Se não incluir qualquer tentativa ou se qualquer expressão for não polinomial em qualquer variável, mas todas as expressões forem lineares em todos os desconhecidos, **cZeros()** utiliza a eliminação Gaussiana para tentar determinar todos os zeros.

Se um sistema não for polinomial em todas as variáveis nem linear nos desconhecidos, **cZeros()** determina no máximo um zero com um método iterativo aproximado. Para o fazer, o número de valores desconhecidos tem de ser igual ao número de expressões, e todas as outras variáveis nas expressões têm de ser simplificadas para números.

Uma tentativa não real é frequentemente necessária para determinar um zero não real. Para convergência, uma tentativa pode ter de ficar próxima a um zero.

$$\left| \begin{array}{c} \text{cZeros}\left(\left\{u \cdot v - u - c \cdot v^2, v^2 + u\right\}, \{u, v\}\right) \\ \left[\begin{array}{cc} 0 & 0 \\ -(c-1)^2 & -(c-1) \end{array} \right] \end{array} \right|$$

$$\left| \begin{array}{c} \text{cZeros}\left(\left\{u \cdot v - u - v \cdot v^2 + u\right\}, \{u, v, w\}\right) \\ \text{cZero}\left(\left\{u \cdot (v-1) - v \cdot u + v^2\right\}, \{u, v, w\}\right) \\ \left[\begin{array}{ccc} 0 & 0 & c4 \\ \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i & \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i & c4 \\ \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i & \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i & c4 \end{array} \right] \end{array} \right|$$

$$\left| \begin{array}{c} \text{cZeros}\left(\left\{u + v - e^{1w}, u - v - i\right\}, \{u, v\}\right) \\ \left[\begin{array}{cc} \frac{e^{1w} + i}{2} & \frac{e^{1w} - i}{2} \end{array} \right] \end{array} \right|$$

$$\left| \begin{array}{c} \text{cZeros}\left(\left\{e^z - w, w - z^2\right\}, \{w, z\}\right) \\ [0.494866 \quad -0.703467] \end{array} \right|$$

$$\left| \begin{array}{c} \text{cZeros}\left(\left\{e^{-z} - w, w - z^2\right\}, \{w, z = 1 + i\}\right) \\ [0.149606 + 4.8919 \cdot i \quad 1.58805 + 1.54022 \cdot i] \end{array} \right|$$

dbd()

Catálogo > 

dbd(*data1*,*data2*) ⇒ *valor*

Devolve o número de dias entre *data1* e *data2* com o método de contagem de dias actual.

data1 e *data2* podem ser números ou listas de números no intervalo das datas no calendário padrão. Se *data1* e *data2* forem listas, têm de ter o mesmo comprimento.

data1 e *data2* têm de estar entre os anos 1950 e 2049.

Pode introduzir as datas num de dois formatos. A colocação decimal diferencia-se entre os formatos de data.

MM.AAAA (formato utilizado nos Estados Unidos)

DDMM.AA (formato utilizado na Europa)

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

►DD

Catálogo > 

Expr1 ►DD ⇒ valor

Lista1 ►DD ⇒ lista

Matriz1 ►DD ⇒ matriz

Nota: Pode introduzir este operador através da escrita de @►DD no teclado do computador.

Devolve o decimal equivalente do argumento expresso em graus. O argumento é um número, uma lista ou uma matriz que é interpretada pela definição do modo ângulo em gradianos, radianos ou graus.

No modo de ângulo Graus:

{1.5°}►DD	1.5°
{45°22'14.3"}►DD	45.3706°
{ { 45°22'14.3",60°0'0" } }►DD	{ 45.3706°,60° }

No modo de ângulo Gradianos:

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

No modo de ângulo Radianos:

{1.5}►DD	85.9437°
----------	----------

Expressão1 ►Decimal $\Rightarrow$ expressão

Lista1 ►Decimal $\Rightarrow$ expressão

Matriz1 ►Decimal $\Rightarrow$ expressão

Nota: Pode introduzir este operador através da escrita de @>Decimal no teclado do computador.

Mostra o argumento em forma decimal. Este operador só pode ser utilizado no fim da linha de entrada.

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

Define

Define *Var* = *Expressão*

Define *Função*(*Parâml*, *Parâm2*, ...) = *Expressão*

Define a variável *Var* ou a função *Função* definida pelo utilizador.

Os parâmetros como, por exemplo, *Parâml*, fornecem marcadores para argumentos de passagem para a função. Quando chamar uma função definida pelo utilizador, tem de fornecer os argumentos (por exemplo, valores ou variáveis) correspondentes aos parâmetros. Quando chamada, a função avalia a *Expressão* com os argumentos fornecidos.

Var e *Função* não podem ter o nome de uma variável do sistema, um comando ou uma função integrada.

Nota: Esta forma de **Define** é equivalente à execução da expressão: *expressão* $\rightarrow$ *Função*(*Parâml*, *Parâm2*).

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 Função(*Parâ1, Parâ2, ...*) =
Func

Bloco

EndFunc

Define Programa(*Parâ1, Parâ2, ...*) =
Prgm

Bloco

EndPrgm

Desta forma, o programa ou a função definida pelo utilizador pode executar um bloco de várias afirmações.

Bloco pode ser uma afirmação ou uma série de afirmações em linhas separadas. *O bloco* pode também incluir expressões e instruções (como, por exemplo, **If**, **Then**, **Else** e **For**).

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

Nota: Consulte também **Define LibPriv**, página 51, e **Define LibPub**, página 52.

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

$g(3,-7)$ 3

Define $g(x,y)=\text{Prgm}$
 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

Define LibPriv *Var = Expressão*

Define LibPriv Função(*Parâ1, Parâ2, ...*) = *Expressão*

Define LibPriv Função(*Parâ1, Parâ2, ...*) = **Func**

Bloco

EndFunc

Define LibPriv Programa(*Parâ1, Parâ2,*

...) = Prgm

Bloco

EndPrgm

Funciona da mesma forma que **Define**, excepto com um programa, uma função ou uma variável da biblioteca privada. As funções e os programas privados não aparecem no Catálogo.

Nota: Consulte também **Define**, página 50, e **Define LibPub**, página 52.

Define LibPub *Var = Expressão*

Define LibPub *Função(Parâm1, Parâm2, ...)* = *Expressão*

Define LibPub *Função(Parâm1, Parâm2, ...)* = **Func**

Bloco

EndFunc

Define LibPub *Programa(Parâm1, Parâm2, ...)* = **Prgm**

Bloco

EndPrgm

Funciona da mesma forma que **Define**, excepto com um programa, uma função ou uma variável da biblioteca pública. As funções e os programas públicos aparecem no Catálogo depois de guardar e actualizar a biblioteca.

Nota: Consulte também **Define**, página 50, e **Define LibPriv**, página 51.

DelVar

Catálogo >

DelVar Var1[, Var2] [, Var3] ...	$2 \rightarrow a$	2
DelVar Var.	$(a+2)^2$	16
Elimina a variável ou o grupo de variáveis especificado da memória.	DelVar a	Done
	$(a+2)^2$	$(a+2)^2$

Se uma ou mais variáveis estiverem bloqueadas, este comando mostra uma mensagem de erro e só elimina as variáveis desbloqueadas. Consulte **unLock**, página 216.

DelVar Var. elimina todos os membros da Var. grupo de variáveis (como, por exemplo, as estatísticas *stat.nn* resultados ou variáveis criados com a função **LibShortcut()**). O ponto (.) nesta forma do comando **DelVar** limita-o à eliminação do grupo de variáveis; a variável simples Var não é afectada.

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

delVoid()

Catálogo >

delVoid(Lista1)⇒lista	delVoid({1,void,3})	{1,3}
-----------------------	---------------------	-------

Devolve uma lista com o conteúdo de Lista1 com todos os elementos (nulos) vazios removidos.

Para mais informações sobre os elementos vazios, consulte página 274.

derivative()

Consulte d(), página 241.

deSolve(1^{a} ou 2^{a} *OrdemODE*, *Var*, *depVar*)
 $\Rightarrow$ uma solução geral

Devolve uma equação que especifica explicita ou implicitamente uma solução geral para a equação diferencial ordinária (ODE) de 1^{a} ou 2^{a} ordem. Na ODE:

- Utilize um símbolo de apóstrofo (prima () para indicar a 1^{a} derivada da variável dependente em relação à variável independente.
- Utilize dois símbolos de apóstrofo para indicar a segunda derivada correspondente.

O símbolo de apóstrofo é utilizado para derivadas apenas em deSolve(). Noutros casos, utilize **d()**.

A solução geral de uma equação de 1^{a} ordem contém uma constante arbitrária da forma $c\ k$, em que k é um sufixo com valor inteiro de 1 a 255. A solução de uma equação de 2^{a} ordem contém duas constantes.

Aplique **solve()** numa solução implícita se a quiser tentar converter para uma ou mais soluções explícitas equivalentes.

Quando comparar os resultados com as soluções dos manuais, não se esqueça de que diferentes métodos introduzem constantes arbitrárias em diferentes pontos no cálculo, que pode produzir diferentes soluções gerais.

deSolve(1^{a} *OrdemODE* and *Condinic*, *Var*, *depVar*) $\Rightarrow$ uma solução específica

Devolve uma solução específica que satisfaz 1^{a} *OrdemODE* e *Condinic*. Esta é geralmente mais simples do que determinar uma solução geral, substituir valores iniciais, resolver com constante arbitrária e , em seguida, substituir esse valor na solução geral.

Condinic é uma equação da forma:

$$\begin{aligned} \text{deSolve}(y'' + 2 \cdot y' + y &= x^2, x, y) \\ y &= (c3 \cdot x + c4) \cdot e^{-x} + x^2 - 4 \cdot x + 6 \\ \text{right}(Ans) \rightarrow \text{temp} & \quad (c3 \cdot x + c4) \cdot e^{-x} + x^2 - 4 \cdot x + 6 \\ \frac{d^2}{dx^2}(\text{temp}) + 2 \cdot \frac{d}{dx}(\text{temp}) + \text{temp} &= x^2 \quad 0 \\ \text{DelVar temp} & \quad Done \end{aligned}$$

$$\begin{aligned} \text{deSolve}(y' &= (\cos(y))^2 \cdot x, x, y) \quad \tan(y) = \frac{x^2}{2} + c4 \end{aligned}$$

$$\begin{aligned} \text{solve}(Ans, y) \\ y &= \tan^{-1}\left(\frac{x^2 + 2 \cdot c4}{2}\right) + n3 \cdot \pi \\ Ans | c4 = c - 1 \text{ and } n3 = 0 \\ y &= \tan^{-1}\left(\frac{x^2 + 2 \cdot (c - 1)}{2}\right) \end{aligned}$$

$$\begin{aligned} \sin(y) &= (v \cdot e^x + \cos(y)) \cdot y' \rightarrow ode \\ \sin(y) &= (e^x \cdot y + \cos(y)) \cdot y' \\ \text{deSolve}(ode \text{ and } y(0) &= 0, x, y) \rightarrow soln \\ \frac{-(2 \cdot \sin(y) + y^2)}{2} &= (e^x - 1) \cdot e^{-x} \cdot \sin(y) \\ soln | x = 0 \text{ and } y = 0 & \quad true \\ ode | y' = \text{impDif}(soln, x, y) & \quad true \\ \text{DelVar ode, soln} & \quad Done \end{aligned}$$

$depVar$ (*ValorIndependenteInicial*) =
ValorDependenteInicial

ValorIndependenteInicial e
ValorDependenteInicial podem ser
variáveis como, por exemplo, x_0 e y_0 que
não tenham valores guardados. A
diferenciação implícita pode ajudar a
verificar as soluções implícitas.

deSolve

($2^{\text{a}}\text{OrdemODEandCondicin1and}$
Condicini2, Var, depVar) $\Rightarrow$ uma solução
específica

Devolve uma solução específica que satisfaz
 $2^{\text{a}}\text{Ordem ODE}$ e tem um valor
especificado da variável dependente e da
primeira derivada num ponto.

Para *Condicin1*, utilize a forma:

$depVar$ (*ValorIndependenteInicial*) =
ValorDependenteInicial

Para *Condicin2*, utilize a forma:

$depVar$ (*ValorIndependenteInicial*) =
Valor1ªDerivadaInicial

deSolve

($2^{\text{a}}\text{OrdemODEandCondbnd1andCondbnd2}$
, *Var, depVar*) $\Rightarrow$ uma solução específica

Apresenta uma solução particular
 $2^{\text{a}}\text{OrdemODE}$ e tem valores especificados
em dois pontos diferentes.

$$\text{deSolve}\left(w'' - \frac{2 \cdot w'}{x} + \left(9 + \frac{2}{x^2}\right) \cdot w = x \cdot e^x \text{ and } w\left(\frac{\pi}{6}\right) = 0 \text{ and } w\left(\frac{\pi}{3}\right) = 0, x, w\right)$$

$$w = \frac{x \cdot e^x}{(\ln(e))^2 + 9} + \frac{\frac{\pi}{3} \cdot x \cdot \cos(3 \cdot x)}{(\ln(e))^2 + 9} - \frac{\frac{\pi}{6} \cdot x \cdot \sin(3 \cdot x)}{(\ln(e))^2 + 9}$$

$$\text{deSolve}\left(y' = y^{\frac{-1}{2}} \text{ and } y(0) = 0 \text{ and } y'(0) = 0, t, y\right)$$

$$\frac{\frac{2 \cdot y^{\frac{4}{3}}}{3} = t}{3}$$

$$\text{solve}\left(\frac{\frac{2 \cdot y^{\frac{4}{3}}}{3} = t, y\right)$$

$$y = \frac{\frac{1}{3 \cdot 3^{\frac{3}{4}}} \cdot \frac{2}{2^{\frac{3}{4}}} \cdot t^{\frac{4}{3}}}{4} \text{ and } t \geq 0$$

$$\text{deSolve}(y'' = x \text{ and } y(0) = 1 \text{ and } y'(2) = 3, x, y)$$

$$y = \frac{x^3}{6} + x + 1$$

$$\text{deSolve}(y'' = 2 \cdot y' \text{ and } y(3) = 1 \text{ and } y'(4) = 2, x, y)$$

$$y = e^{2 \cdot x - 8} - e^{-2} + 1$$

det()

Catálogo > 

det(*MatrizQuadrada* [, *Tolerância*]) $\Rightarrow$ expressão

Apresenta o determinante de *MatrizQuadrada*.

Opcionalmente, qualquer elemento da matriz é tratado como zero se o valor absoluto for inferior à *Tolerância*. Esta tolerância é utilizada apenas se a matriz tiver entradas de ponto flutuante e não contiver nenhuma variável simbólica sem nenhum valor atribuído. Caso contrário, *Tolerância* é ignorada.

- Se utilizar   ou definir o modo **Auto ou Aproximado** para Aproximado, os cálculos são efectuados com a aritmética de ponto flutuante.
- Se *Tolerância* for omitida ou não utilizada, a tolerância predefinida é calculada da seguinte forma:

$$5E-14 \cdot \max(\dim(\text{MatrizQuadrada})) \cdot \text{rowNorm}(\text{MatrizQuadrada})$$

$\det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right)$	$a \cdot d - b \cdot c$
$\det\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$	-2
$\det\left(\text{identity}(3) - x \cdot \begin{bmatrix} 1 & -2 & 3 \\ -2 & 4 & 1 \\ -6 & -2 & 7 \end{bmatrix}\right)$	$-(98 \cdot x^3 - 55 \cdot x^2 + 12 \cdot x - 1)$
$\begin{bmatrix} 1.E20 & 1 \\ 0 & 1 \end{bmatrix} \rightarrow \text{mat1}$	$\begin{bmatrix} 1.E20 & 1 \\ 0 & 1 \end{bmatrix}$
$\det(\text{mat1})$	0
$\det(\text{mat1}, 1)$	1.E20

diag()

Catálogo > 

diag(*Lista*) $\Rightarrow$ matriz

diag(*MatrizLinha*) $\Rightarrow$ matriz

diag(*MatrizColuna*) $\Rightarrow$ matriz

Devolve uma matriz com os valores da matriz ou da lista de argumentos na diagonal principal.

diag(*MatrizQuadrada*) $\Rightarrow$ *MatrizLinha*

Devolve uma matriz da linha com elementos da diagonal principal de *MatrizQuadrada*.

MatrizQuadrada tem de ser quadrada.

$\text{diag}([2 \ 4 \ 6])$	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix}$
$\begin{bmatrix} 4 & 6 & 8 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{bmatrix}$	$\begin{bmatrix} 4 & 6 & 8 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{bmatrix}$
$\text{diag}(\text{Ans})$	$\begin{bmatrix} 4 & 2 & 9 \end{bmatrix}$

dim()Catálogo > **dim(Lista)** ⇒ número inteiro

dim({ 0,1,2 }) 3

Devolve a dimensão de *Lista*.**dim(Matriz)** ⇒ listadim($\begin{pmatrix} 1 & -1 \\ 2 & -2 \\ 3 & 5 \end{pmatrix}$) { 3,2 }

Devolve as dimensões da matriz como uma lista de dois elementos {linhas, colunas}.

dim(Cadeia) ⇒ número inteiro

dim("Hello") 5

Devolve o número de caracteres contidos na cadeia de caracteres *Cadeia*.

dim("Hello "&"there") 11

DispCatálogo > **Disp *exprOuCadeia1* [, *exprOuCadeia2*] ...**

```

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

```

Mostra os argumentos no histórico da *Calculadora*. Os argumentos são apresentados em sucessão com espaços pequenos como separadores.

Útil principalmente em programas e funções para garantir a visualização de cálculos intermédios.

```

chars(240,243)
-----
240 ð
241 ñ
242 ò
243 ó
-----

```

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção *Calculadora do manual do utilizador do produto*.**DispAt**Catálogo > **DispAt *int,expr1* [,*expr2* ...] ...**

DispAt

DispAt permite-lhe especificar a linha onde a expressão ou cadeia será apresentada no ecrã.**Exemplo**

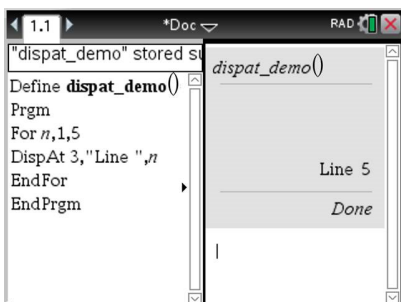
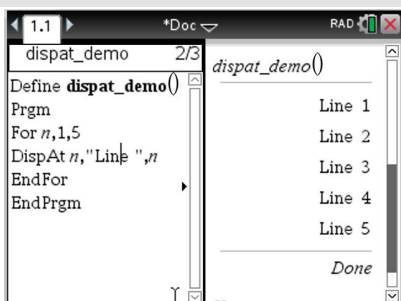
O número da linha pode ser especificado como uma expressão.

Tenha em atenção que o número da linha não se destina ao ecrã inteiro, mas à área imediatamente a seguir ao comando/programa.

Este comando permite uma apresentação de dados semelhante a um painel em que o valor de uma expressão ou de uma leitura de sensor é atualizado na mesma linha.

DispAte **Disp** podem ser utilizados no mesmo programa.

Nota: o número máximo está definido para 8, uma vez que esse número corresponde a um ecrã cheio de linhas no ecrã da unidade portátil - desde que as linhas não contenham expressões matemáticas 2D. O número exato de linhas depende do conteúdo da informação apresentada.



Exemplos ilustrativos:

Define z()=	Output
Prgm	z()
For n,1,3	Iteração 1:
DispAt 1,"N: ",n	Linha 1: N:1
Disp "Olá"	Linha 2: Olá
EndFor	
EndPrgm	Iteração 2:
	Linha 1: N:2
	Linha 2: Olá
	Linha 3: Olá
	Iteração 3:
	Linha 1: N:3
	Linha 2: Olá
	Linha 3: Olá

	Linha 4: Olá
Define z1()=	z1()
Prgm	Linha 1: N:3
For n,1,3	Linha 2: Olá
DispAt 1,"N: ",n	Linha 3: Olá
EndFor	Linha 4: Olá
	Linha 5: Olá
For n,1,4	
Disp "Olá"	
EndFor	
EndPrgm	

Condições de erro:

Mensagem de erro	Descrição
O número de linha DispAt deve situar-se entre 1 e 8	A expressão avalia o número de linha fora do intervalo 1-8 (inclusive)
Poucos argumentos	A função ou o comando não tem um ou mais argumentos.
Nenhum argumento	Igual à caixa de diálogo atual 'erro de sintaxe'
Demasiados argumentos	Limitar argumento. Mesmo erro que Disp.
Tipo de dados inválido	O primeiro argumento tem de ser um número.
Nulo: DispAt nulo	O erro de tipo de dados "Olá mundo" é projetado para o nulo (se o callback estiver definido)
Operador de conversão: DispAt 2_ft @> _m, "Olá mundo"	CAS: O erro de tipo de dados é projetado (se o callback estiver definido) Numérico: A conversão será avaliada e, se o resultado for um argumento válido, DispAt imprime a cadeia na linha de resultados.

Expr ►DMS

No modo de ângulo Graus:

Lista ►DMS $\{45.371\}$ ►DMS 45°22'15.6"*Matriz* ►DMS $\{\{45.371,60\}\}$ ►DMS $\{45^{\circ}22'15.6",60^{\circ}\}$

Nota: Pode introduzir este operador através da escrita de @>DMS no teclado do computador.

Interpreta o argumento como um ângulo e mostra o número DMS equivalente (DDDDDD °MM 'SS.ss "). Consulte °, ', " (página 250) para o formato DMS (grau, minutos, segundos).

Nota: ►DMS converterá de radianos para graus quando utilizado em modo de radianos. Se a entrada for seguida por um símbolo de grau °, não ocorrerá nenhuma conversão. Pode utilizar o ►DMS apenas no fim de uma linha de entrada.

domain() (domínio)

domain(Expr1, Var)⇒expressão

Devolve o domínio de *Expr1* em relação à *Var*.

domain() pode ser utilizado para examinar domínios e funções. Está limitado ao domínio real e finito.

Esta funcionalidade tem limitações devido a deficiências de simplificação algébrica computacional e a algoritmos de resolução.

Certas funções não podem ser utilizadas como argumentos para **domain()**, independentemente de aparecerem explicitamente ou em variáveis e funções definidas pelo utilizador. No exemplo seguinte, a expressão não pode ser simplificada porque $f()$ é uma função não permitida.

$$\text{domain}\left(\begin{pmatrix} x \\ \frac{1}{t} \\ 1 \end{pmatrix} dt, x\right) \rightarrow \text{domain}\left(\begin{pmatrix} x \\ \frac{1}{t} \\ 1 \end{pmatrix} dt, x\right)$$

$\text{domain}\left(\frac{1}{x+y}, y\right)$	$-\infty < y < -x \text{ or } -x < y < \infty$
$\text{domain}\left(\frac{x+1}{x^2+2 \cdot x}, x\right)$	$x \neq -2 \text{ and } x \neq 0$
$\text{domain}\left((\sqrt{x})^2, x\right)$	$0 \leq x < \infty$
$\text{domain}\left(\frac{1}{x+y}, y\right)$	$-\infty < y < -x \text{ or } -x < y < \infty$

dominantTerm(Expr1, Var [, Ponto]) ⇒ expressão

dominantTerm(Expr1, Var [, Ponto]) | Var > Ponto ⇒ expressão

dominantTerm(Expr1, Var [, Ponto]) | Var < Ponto ⇒ expressão

Devolve o termo dominante de uma representação da série de potência de Expr1 aberta sobre Ponto. O termo dominante é aquele cuja magnitude cresce mais rapidamente junto a Var = Ponto. A potência resultante de (Var - Ponto) pode ter um expoente fraccionário e/ou negativo. O coeficiente desta potência pode incluir logaritmos de (Var - Ponto) e outras funções de Var que são dominadas por todas as potências de (Var - Ponto) com o mesmo sinal de expoente.

O Ponto predefine-se para 0. O Ponto pode ser ∞ ou $-\infty$, nestes casos, o termo dominante será o termo com o expoente maior de Var em vez do expoente menor de Var.

dominantTerm(...) devolve “dominantTerm(...)” se não for capaz de determinar essa representação, como para singularidades essenciais, como, por exemplo, $\sin(1/z)$ a $z=0$, $e^{-1/z}$ a $z=0$, ou e^z a $z = \infty$ ou $-\infty$.

$\text{dominantTerm}(\tan(\sin(x)) - \sin(\tan(x)), x)$	$\frac{x^7}{30}$
$\text{dominantTerm}\left(\frac{1 - \cos(x-1)}{(x-1)^3}, x, 1\right)$	$\frac{1}{2 \cdot (x-1)}$
$\text{dominantTerm}\left(x^{-2} \cdot \tan\left(\frac{1}{x}\right), x\right)$	$\frac{1}{x^3}$
$\text{dominantTerm}(\ln(x^x - 1) \cdot x^{-2}, x)$	$\frac{\ln(x \cdot \ln(x))}{x^2}$

$\text{dominantTerm}\left(e^{\frac{-1}{z}}, z\right)$	$\text{dominantTerm}\left(e^{\frac{-1}{z}}, z, 0\right)$
$\text{dominantTerm}\left(\left(1 + \frac{1}{n}\right)^n, n, \infty\right)$	e
$\text{dominantTerm}\left(\tan^{-1}\left(\frac{1}{x}\right), x, 0\right)$	$\frac{\pi \cdot \text{sign}(x)}{2}$
$\text{dominantTerm}\left(\tan^{-1}\left(\frac{1}{x}\right), x, x > 0\right)$	$\frac{\pi}{2}$

Se a série ou um das derivadas tiver uma descontinuidade em *Ponto*, o resultado contém provavelmente subexpressões do sinal(...) ou abs(...) da forma para uma variável de expansão real ou $(-1)^{\text{floor}(\dots\text{ângulo}(\dots))}$ para uma variável de expansão complexa, que é uma que termina com “_”. Se quiser utilizar o termo dominante apenas para os valores num lado de *Ponto*, adicione ao **dominantTerm(...)**, um valor adequado de “| *Var* > *Ponto*”, “| *Var* < *Ponto*”, “| *Var* ≥ *Ponto*” ou “| *Var* ≤ *Ponto*” para obter um resultado mais simples.

dominantTerm() distribui-se pelas listas e matrizes do 1º argumento.

dominantTerm() é útil quando quiser saber a expressão mais simples possível que é assintótica para outra expressão como *Var* → *Ponto*. **dominantTerm()** é também útil quando não for óbvio qual é o grau do primeiro termo não zero de uma série, e não quiser descobrir iterativamente de forma interactiva ou através de um ciclo do programa.

Nota: Consulte também **série()**, página 176.

dotP()

dotP(Lista1, Lista2) ⇒ expressão

Devolve o produto do “ponto” de duas listas.

$\text{dotP}(\{a,b,c\},\{d,e,f\})$	$a \cdot d + b \cdot e + c \cdot f$
$\text{dotP}(\{1,2\},\{5,6\})$	17

dotP(Vector1, Vector2) ⇒ expressão

Devolve o produto do “ponto” de dois vectores.

$\text{dotP}([a \ b \ c],[d \ e \ f])$	$a \cdot d + b \cdot e + c \cdot f$
$\text{dotP}([1 \ 2 \ 3],[4 \ 5 \ 6])$	32

Ambos têm de ser vectores da linha ou da coluna.

e^()	Tecla 	
e^(Expr1) ⇒ expressão	e^1	e
Devolve e elevado à potência Expr1.	$e^1.$	2.71828
Nota: Consulte também e modelo do expoente, página 2.	e^{3^2}	e^9
Nota: Premir  para ver e ^() é diferente de premir o carácter  no teclado.		
Pode introduzir um número complexo na forma polar $re^{i\theta}$. No entanto, utilize esta forma apenas no modo de ângulo Radianos; causa um erro de domínio no modo de ângulo Graus ou Gradianos.		
e^(Lista1) ⇒ lista	$e^{\{1,1.,0.5\}}$	$\{e,2.71828,1.64872\}$
Devolve e elevado à potência de cada elemento em Lista1.		
e^(MatrizQuadrada1) ⇒ MatrizQuadrada	$e^{\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}}$	$\begin{bmatrix} 782.209 & 559.617 & 456.509 \\ 680.546 & 488.795 & 396.521 \\ 524.929 & 371.222 & 307.879 \end{bmatrix}$
Devolve a matriz exponencial de MatrizQuadrada1. Isto não é o mesmo que calcular e elevado à potência de cada elemento. Para mais informações sobre o método de cálculo, consulte cos().		
MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.		

eff()	Catálogo > 	
eff(TaxaNominal,CpY) ⇒ valor	$eff(5.75,12)$	5.90398
Função financeira que converte a taxa de juro nominal TaxaNominal para uma taxa efectiva anual, dando CpY como o número de período compostos por ano.		
TaxaNominal tem de ser um número real e CpY tem de ser um número real > 0.		

Nota: Consulte também `nom()`, página 134.

eigVc()

eigVc(MatrizQuadrada) ⇒ matriz

No Formato complexo rectangular:

Devolve uma matriz com os vectores próprios para uma *MatrizQuadrada* real ou complexa, em que cada coluna do resultado corresponde a um valor próprio. Não se esqueça de que um vector próprio não é único; pode ser dimensionado por qualquer factor constante. Os vectores próprios são normalizados, significando que se $V = [x_1, x_2, \dots, x_n]$:

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

MatrizQuadrada é primeiro equilibrada com tranformações de similaridade até as normas das colunas e linhas estarem o mais perto possível do mesmo valor. A *MatrizQuadrada* é reduzida para a forma Hessenberg superior e os vectores próprios são calculados através de uma factorização Schur.

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

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

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

eigVI()

eigVI(MatrizQuadrada) ⇒ lista

No modo de formato complexo rectangular:

Devolve uma lista dos valores próprios de uma *MatrizQuadrada* real ou complexa.

MatrizQuadrada é primeiro equilibrada com tranformações de similaridade até as normas das colunas e linhas estarem o mais perto possível do mesmo valor. A *MatrizQuadrada* é reduzida para a forma Hessenberg superior e os valores próprios são calculados a partir da matriz Hessenberg superior.

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

$$\text{eigVI}(mI) \{ -4.40941, 2.20471 + 0.763006 \cdot i, 2.20471 - 0.763006 \cdot i \}$$

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

ElseIf

Catálogo > **Se** *ExprBooleana1**Block1***ElseIf** *BooleanExpr2**Block2*

⋮

ElseIf *ExprBooleanaN**BlockN***EndIf**

⋮

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

Define $g(x)$ = FuncIf $x \leq -5$ Then

Return 5

ElseIf $x > -5$ and $x < 0$ ThenReturn $-x$ ElseIf $x \geq 0$ and $x \neq 10$ ThenReturn x ElseIf $x = 10$ Then

Return 3

EndIf

EndFunc

Done

EndFor

Consulte For, página 79.

EndFunc

Consulte Func, página 83.

EndIf

Consulte If, página 96.

EndLoop

Consulte Loop, página 120.

euler ()

Catálogo > 

euler(Expr, Var, depVar, {Var0, VarMax}, depVar0, VarStep [, eulerStep]) $\Rightarrow$ matriz

euler(SystemOfExpr, Var, ListOfDepVars, {Var0, VarMax}, ListOfDepVars0, VarStep [, eulerStep]) $\Rightarrow$ matriz

euler(ListOfExpr, Var, ListOfDepVars, {Var0, VarMax}, ListOfDepVars0, VarStep [, eulerStep]) $\Rightarrow$ matriz

Utiliza o método de Euler para resolver o sistema

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

com $\text{depVar}(\text{Var0}) = \text{depVar0}$ no intervalo $[\text{Var0}, \text{VarMax}]$. Apresenta uma matriz cuja primeira linha define os valores de saída Var e cuja segunda linha define o valor da primeira componente da solução nos valores Var correspondentes, e assim por diante.

Expr é o lado direito que define a equação diferencial ordinária (EDO).

SystemOfExpr é o sistema de lados direitos que definem o sistema de EDOs (corresponde à ordem de variáveis dependentes em ListOfDepVars).

Equação diferencial:

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

$$\text{euler}\left(0.001 \cdot y \cdot (100 - y), t, y, \{0, 100\}, 10, 1\right)$$

0.	1.	2.	3.	4.
10.	10.9	11.8712	12.9174	14.042

Para ver o resultado completo, prima $\blacktriangle$ e, de seguida, utilize $\blacktriangleleft$ e $\blacktriangleright$ para mover o cursor.

Compare o resultado acima com a solução exacta CAS obtida através de deSolve() e seqGen():

$$\text{deSolve}(y' = 0.001 \cdot y \cdot (100 - y) \text{ and } y(0) = 10, t, y)$$

$$y = \frac{100 \cdot (1.10517)^t}{(1.10517)^t + 9.}$$

$$\text{seqGen}\left(\frac{100 \cdot (1.10517)^t}{(1.10517)^t + 9.}, t, y, \{0, 100\}\right)$$

$$\{10., 10.9367, 11.9494, 13.0423, 14.2189\}$$

Sistema de equações:

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

com $y1(0) = 2$ e $y2(0) = 5$

ListOfExpr é uma lista de segundos membros que definem o sistema de EDOs (corresponde à ordem de variáveis dependentes em *ListOfDepVars*).

Var é a variável independente.

ListOfDepVars é uma lista de variáveis dependentes.

{*Var0*, *VarMax*} é uma lista de dois elementos que informa a função para integrar de *Var0* a *VarMax*.

ListOfDepVars0 é uma lista de valores iniciais para variáveis dependentes.

VarStep é um número diferente de zero tal como **sign(*VarStep*) = sign(*VarMax-Var0*)** e as soluções regressam a *Var0*+*i·VarStep* para todos os *i*=0,1,2,... tal como *Var0*+*i·VarStep* está em [*var0*,*VarMax*] (pode não existir um valor de solução em *VarMax*).

eulerStep é um número inteiro positivo (passa para 1) que define o número de passos Euler entre os valores de saída. O tamanho de passo real utilizado pelo método Euler é *VarStep/eulerStep*.

$$\text{euler} \left(\begin{array}{l} \left\{ \begin{array}{l} -y1+0.1 \cdot y1 \cdot y2 \\ 3 \cdot y2 - y1 \cdot y2 \end{array} \right\}, \{y1, y2\}, \{0, 5\}, \{2, 5\}, 1 \end{array} \right)$$

0.	1.	2.	3.	4.	5.
2.	1.	1.	3.	27.	243.
5.	10.	30.	90.	90.	-2070.

eval ()

Menu Hub

eval(*Expr*) ⇒ cadeia

eval() só é válida no TI-Innovator™ Hub argumento Comando dos comandos programados **Get**, **GetStr** e **Send**. O software avalia a expressão *Expr* e substitui a instrução **eval()** pelo resultado como cadeia de caracteres.

O argumento *Expr* tem de ser simplificado para um número real.

Definir o elemento azul do LED RGB para metade da intensidade.

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

Repor o elemento azul para DESLIGADO.

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

O argumento **eval()** tem de ser simplificado para um número real.

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

Programar para aparecimento gradual do elemento vermelho.

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

Executar o programa.

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

Embora **eval()** não apresente o resultado, pode ver a cadeia de comando resultante do Hub após executar o comando inspecionando qualquer uma das variáveis especiais seguintes.

iostr.SendAns
iostr.GetAns
iostr.GetStrAns

Nota: Ver também **Get** (página 85), **GetStr** (página 93) e **Send** (página 173).

exact()

Catálogo > 

exact(Expr1 [, Tolerância]) ⇒ expressão

exact(Lista1 [, Tolerância]) ⇒ lista

exact(Matriz1 [, Tolerância]) ⇒ matriz

Utiliza o modo aritmético Exacto para apresentar, quando possível, o número racional equivalente do argumento.

Tolerância especifica a tolerância para a conversão; a predefinição é 0 (zero).

exact(0.25)	$\frac{1}{4}$
exact(0.333333)	$\frac{333333}{1000000}$
exact(0.333333,0.001)	$\frac{1}{3}$
exact(3.5·x+y)	$\frac{7 \cdot x}{2} + y$
exact({ 0.2,0.33,4.125 })	$\left\{ \frac{1}{5}, \frac{33}{100}, \frac{33}{8} \right\}$

Exit

Catálogo > 

Exit

Listagem de funções:

Sai do bloco **For**, **While** ou **Loop** actual.

Exit não é permitido fora das três estruturas circulares (**For**, **While** ou **Loop**).

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

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

►exp

Expr ►exp

Representa *Expr* em função do expoente natural *e*. Este é um operador de conversão. Apenas pode ser utilizado no fim da linha de entrada.

Nota: Pode introduzir este operador através da escrita de **@>exp** no teclado do computador.

$\frac{d}{dx}(e^x + e^{-x})$	$2 \cdot \sinh(x)$
$2 \cdot \sinh(x) \blacktriangleright \exp$	$e^x - e^{-x}$

exp()

exp(Expr1) ⇒ expressão

Devolve **e** elevado à potência *Expr1*.

Nota: Consulte também **e** modelo do expoente, página 2.

Pode introduzir um número complexo na forma polar $re^{i\theta}$. No entanto, utilize esta forma apenas no modo de ângulo Radianos; causa um erro de domínio no modo de ângulo Graus ou Gradianos.

exp(Lista1) ⇒ lista

Devolve **e** elevado à potência de cada elemento em *Lista1*.

exp(MatrizQuadrada1) ⇒ MatrizQuadrada

e^1	e
$e^{1.}$	2.71828
e^{3^2}	e^9

$e^{\{1,1.,0.5\}}$	$\{e,2.71828,1.64872\}$
--------------------	-------------------------

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

Devolve a matriz exponencial de *MatrizQuadrada1*. Isto não é o mesmo que calcular **e** elevado à potência de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

exp▶lista()Catálogo > 

exp▶lista(Expr, Var) ⇒ lista

Examina *Expr* para equações separadas pela palavra “ou,” e devolve uma lista com os lados direitos das equações da forma *Var=Expr*. Isto fornece uma forma simples para extrair alguns valores das soluções embebidos nos resultados das funções **solve()**, **cSolve()**, **fMin()** e **fMax()**.

Nota: **exp▶list()** não é necessário com os **zeros** e as funções **cZeros()** porque devolvem uma lista dos valores das soluções directamente.

Pode introduzir esta função através da escrita de **exp@>list(...)** no teclado.

$\text{solve}(x^2 - x - 2 = 0, x)$	$x = 1 \text{ or } x = 2$
$\text{exp▶list}(\text{solve}(x^2 - x - 2 = 0, x), x)$	$\{-1, 2\}$

expand()Catálogo > 

expand(Expr1 [, Var]) ⇒ expressão

expand(Lista1 [, Var]) ⇒ lista

expand(Matriz1 [, Var]) ⇒ matriz

expand(Expr1) devolve *Expr1* expandido em relação a todas as variáveis. A expansão é uma expansão polinomial para polinómios e a expansão de fracção parcial para expressões racionais.

$\text{expand}((x+y+1)^2)$
$x^2 + 2 \cdot x \cdot y + 2 \cdot x \cdot y^2 + 2 \cdot y + 1$
$\text{expand}\left(\frac{x^2 - x + y^2 - y}{x^2 \cdot y^2 - x^2 \cdot y - x \cdot y^2 + x \cdot y}\right)$
$\frac{1}{x-1} + \frac{1}{x} + \frac{1}{y-1} + \frac{1}{y}$

O objectivo de **expand()** é transformar *Expr1* numa soma e/ou diferença de termos simples. Pelo contrário, o objectivo de **factor()** é transformar *Expr1* num produto e/ou quociente de factores simples.

expand (Expr1, Var) devolve *Expr1* expandido em relação a *Var*. As potências similares de *Var* são recolhidas. Os termos e os factores são ordenados com *Var* como variável principal. Pode existir alguma decomposição de factores incidental ou a expansão dos coeficientes recolhidos. Comparada para omitir *Var*, esta poupa tempo frequentemente, memória e espaço no ecrã, enquanto torna a expressão mais compreensível.

Mesmo quando exista apenas uma variável, a utilização de *Var* pode tornar a factorização do denominador utilizada para a expansão da fracção parcial mais completa.

Sugestão: Para expressões racionais, **propFrac()** é mais rápida, mas uma alternativa menos extrema para **expand()**.

Nota: Consulte também **comDenom()** para um numerador expandido sobre um denominador expandido.

expand (Expr1, [Var]) também distribui potências fraccionárias e logaritmos, independentemente de *Var*. Para uma distribuição aumentada de potências fraccionárias e logaritmos, os limites das desigualdades podem ser necessários para garantir que alguns factores são não negativos.

expand (Expr1, [Var]) também distribui valores absolutos, **sign()**, e exponenciais, independentemente de *Var*.

$$\begin{array}{l} \text{expand}\left((x+y+1)^2, y\right) \quad y^2+2 \cdot y \cdot (x+1)+(x+1)^2 \\ \text{expand}\left((x+y+1)^2, x\right) \quad x^2+2 \cdot x \cdot (y+1)+(y+1)^2 \\ \text{expand}\left(\frac{x^2-x \cdot y^2-y}{x^2 \cdot y^2-x^2 \cdot y-x \cdot y^2+x \cdot y}, y\right) \\ \frac{1}{y-1}-\frac{1}{y}+\frac{1}{x \cdot (x-1)} \\ \text{expand}(Ans, x) \quad \frac{1}{x-1}-\frac{1}{x}+\frac{1}{y \cdot (y-1)} \end{array}$$

$$\begin{array}{l} \text{expand}\left(\frac{x^3+x^2-2}{x^2-2}\right) \quad \frac{2 \cdot x}{x^2-2}+x+1 \\ \text{expand}(Ans, x) \quad \frac{1}{x-\sqrt{2}}+\frac{1}{x+\sqrt{2}}+x+1 \end{array}$$

$$\begin{array}{l} \ln(2 \cdot x \cdot y)+\sqrt{2 \cdot x \cdot y} \quad \ln(2 \cdot x \cdot y)+\sqrt{2 \cdot x \cdot y} \\ \text{expand}(Ans) \quad \ln(x \cdot y)+\sqrt{2 \cdot x \cdot y}+\ln(2) \\ \text{expand}(Ans)|y \geq 0 \quad \ln(x)+\sqrt{2 \cdot x} \cdot \sqrt{y}+\ln(y)+\ln(2) \\ \text{sign}(x \cdot y)+|x \cdot y| \cdot e^{2 \cdot x+y} \quad \text{sign}(x \cdot y)+|x \cdot y| \cdot e^{2 \cdot x+y} \\ \text{expand}(Ans) \quad \text{sign}(x) \cdot \text{sign}(y)+|x| \cdot |y|+(e^x)^2 \cdot e^y \end{array}$$

Nota: Consulte também **tExpand()** para a soma de ângulos trigonométricos e a expansão de ângulos múltiplos.

expr()

expr(*Cadeia*) $\Rightarrow$ expressão

Devolve a cadeia de caracteres contidos em *Cadeia* como uma expressão e executa-a imediatamente.

<code>expr("1+2+x^2+x")</code>	x^2+x+3
<code>expr("expand((1+x)^2)")</code>	$x^2+2\cdot x+1$
<code>"Define cube(x)=x^3" $\rightarrow$ funcstr</code>	
<code>"Define cube(x)=x^3"</code>	
<code>expr(funcstr)</code>	<i>Done</i>
<code>cube(2)</code>	8

ExpReg

ExpReg *X*, *Y* [, [*Freq*] [, [*Categoria*, *Incluir*]]

Calcula a regressão exponencial $y = a \cdot (b)^x$ a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

X e *Y* são listas de variáveis independentes e dependentes.

Freq é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

Categoria é uma lista de códigos de categorias para os dados *X* e *Y* correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $a \cdot (b)^x$
stat.a, stat.b	Parâmetros da regressão
stat.r ²	Coefficiente de determinação linear para dados transformados
stat.r	Coefficiente de correlação para dados transformados (x, ln(y))
stat.Resid	Resíduos associados ao modelo exponencial
stat.ResidTrans	Residuais associados ao ajuste linear de dados transformados
stat.XReg	Lista de pontos de dados na <i>Lista X</i> modificada utilizada na regressão com base em restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.YReg	Lista de pontos de dados na <i>Lista Y</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.FreqReg	Lista de frequências correspondentes a <i>stat.XReg</i> e <i>stat.YReg</i>

F

factor()

Catálogo > 

factor(*Expr1* [, *Var*]) $\Rightarrow$ expressão

factor(*Lista1* [, *Var*]) $\Rightarrow$ lista

factor(*Matriz1* [, *Var*]) $\Rightarrow$ matriz

factor(*Expr1*) devolve *Expr1* decomposta em relação a todas as variáveis sobre um denominador comum.

$$\begin{aligned} \text{factor}(a^3 \cdot x^2 - a \cdot x^2 - a^3 + a) &= a \cdot (a-1) \cdot (a+1) \cdot (x-1) \cdot (x+1) \\ \text{factor}(x^2+1) &= x^2+1 \\ \text{factor}(x^2-4) &= (x-2) \cdot (x+2) \\ \text{factor}(x^2-3) &= x^2-3 \\ \text{factor}(x^2-a) &= x^2-a \end{aligned}$$

Expr1 é decomposta o mais possível em factores racionais lineares sem introduzir novas subexpressões não reais. Esta alternativa é adequada se quiser a factorização em relação a mais de uma variável.

factor(*Expr1*, *Var*) devolve *Expr1* decomposta em relação à variável *Var*.

Expr1 é decomposta o mais possível em factores reais lineares em *Var*, mesmo que introduza constantes irracionais ou subexpressões irracionais noutras variáveis.

$$\begin{aligned} \text{factor}(a^3 \cdot x^2 - a \cdot x^2 - a^3 + a, x) &= a \cdot (a^2-1) \cdot (x-1) \cdot (x+1) \\ \text{factor}(x^2-3, x) &= (x+\sqrt{3}) \cdot (x-\sqrt{3}) \\ \text{factor}(x^2-a, x) &= (x+\sqrt{a}) \cdot (x-\sqrt{a}) \end{aligned}$$

Os factores e os termos são ordenados com *Var* como variável principal. As potências similares de *Var* são recolhidas em cada factor. Inclua *Var* se a factorização for necessária em relação apenas a essa variável e estiver disposto a aceitar expressões irracionais em qualquer outra variável para aumentar a factorização em relação a *Var*. Pode existir alguma decomposição de factores incidental em relação a outras variáveis.

Para a definição Auto do modo **Auto ou Aproximado**, incluindo *Var*, permite também a aproximação a coeficientes de pontos flutuantes em que os coeficientes irracionais não podem ser expressos explicitamente em termos das funções integradas. Mesmo quando exista apenas uma variável, incluindo *Var*, pode produzir a factorização mais completa.

Nota: Consulte também **comDenom()** para uma forma mais rápida para obter a decomposição de factores parcial quando **factor()** não for suficientemente rápido ou se a memória ficar esgotada.

Nota: Consulte também **cFactor()** para decompor tudo para coeficientes complexos em busca de factores lineares.

factor(NúmeroRacional) devolve o número racional em primos. Para números compostos, o tempo de cálculo cresce exponencialmente com o número de dígitos no segundo maior factor. Por exemplo, a decomposição em factores de um número inteiro de 30 dígitos pode demorar mais de um dia e a decomposição em factores de um número de 100 dígitos pode demorar mais de um século.

Para parar um cálculo manualmente,

- **Dispositivo portátil:** Manter

$\text{factor}(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3)$
$x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3$
$\text{factor}(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3,x)$
$(x-0.964673)\cdot (x+0.611649)\cdot (x+2.12543)\cdot (x^2+1.141678x+0.999999)$

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

pressionada a tecla  **on** e pressionar  repetidamente.

- **Windows®:** Manter pressionada a tecla **F12** e pressionar **Enter** repetidamente.
- **Macintosh®:** Manter pressionada a tecla **F5** e pressionar **Enter** repetidamente.
- **iPad®:** A aplicação apresenta um pedido. Pode continuar a aguardar ou pode cancelar.

Se quiser apenas determinar se um número é primo, utilize **isPrime()**. É muito mais rápido, em especial, se o *NúmeroRacional* não for primo e o segundo maior factor tiver mais de cinco dígitos.

FCdf()

FCdf(*LimiteInferior*, *LimiteSuperior*, *dfNumer*, *dfDenom*) $\Rightarrow$ número se *LimiteInferior* e *LimiteSuperior* forem números, lista se *LimiteInferior* e *LimiteSuperior* forem listas

FCdf(*LimiteInferior*, *LimiteSuperior*, *dfNumer*, *dfDenom*) $\Rightarrow$ número se *LimiteInferior* e *LimiteSuperior* forem números, lista se *LimiteInferior* e *LimiteSuperior* forem listas

Calcula a probabilidade da distribuição **F** entre *LimiteInferior* e *LimiteSuperior* para o *dfNumer* (graus de liberdade) e *dfDenom* especificados.

Para $P(X \leq \text{LimiteSuperior})$, definir *LimiteInferior* = 0.

Fill *Expr*, *VarMatriz* $\Rightarrow$ *matriz*

Substitui cada elemento na variável *VarMatriz* por *Expr*.

matrixVar já tem de existir.

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

Fill *Expr*, *VarLista* $\Rightarrow$ *lista*

Substitui cada elemento na variável *VarLista* por *Expr*.

VarLista já tem de existir.

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

FiveNumSummary

FiveNumSummary *X*[,*Freq*]
[,*Categoria*,*Incluir*]

Fornecer uma versão abreviada da estatística de 1 variável na lista *X*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

X representa uma lista de dados.

Freq é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada valor *X* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

Categoria é uma lista de códigos de categorias numéricas para os valores *X* correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Um elemento (nulo) vazio em qualquer das listas *X*, *Freq* ou *Category* resulta num nulo para o elemento correspondente de todas essas listas. Para mais informações sobre os elementos vazios, consulte página 274.

Variável de saída	Descrição
stat.MinX	Mínimo dos valores x
stat.Q ₁ X	1º quartil de x
stat.MedianX	Mediana de x
stat.Q ₃ X	3º quartil de x
stat.MaxX	Máximo dos valores x

floor()

Catálogo > 

floor(Expr1) ⇒ número inteiro

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

Devolve o maior número inteiro que é ≤ o argumento. Esta função é idêntica a **int** ().

O argumento pode ser um número complexo ou real.

floor(Lista1) ⇒ lista

$$\text{floor}\left(\left\{\frac{3}{2}, 0, -5.3\right\}\right) \quad \{1, 0, -6\}$$

floor(Matriz1) ⇒ matriz

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

Devolve uma lista ou matriz do floor de cada elemento.

Nota: Consulte também **ceiling()** e **int()**.

fMax()

Catálogo > 

fMax(Expr, Var) ⇒ Expressão booleana

$$\text{fMax}\left(1-(x-a)^2-(x-b)^2, x\right) \quad x=\frac{a+b}{2}$$

fMax(Expr, Var, LimiteInferior)

$$\text{fMax}\left(.5 \cdot x^3 - x - 2, x\right) \quad x=\infty$$

fMax(Expr, Var, LimiteInferior, LimiteSuperior)

fMax(Expr, Var) | LimiteInferior ≤ Var ≤ LimiteSuperior

Devolve uma expressão booleana que especifica os valores candidatos de Var que maximiza Expr ou localiza o menor limite superior.

Pode utilizar o operador de limite ("|") para limitar o intervalo da solução e/ou especificar outras restrições.

$$\text{fMax}\left(0.5 \cdot x^3 - x - 2, x\right) | x \leq 1 \quad x=-0.816497$$

Para a definição Aproximado do modo **Auto ou Aproximado**, **fMax()** procura iterativamente um máximo local aproximado. Isto é frequentemente mais rápido, em especial, se utilizar o operador “|” para limitar a procura a um intervalo relativamente pequeno que contenha exactamente um máximo local.

Nota: Consulte também **fMin()** e **max()**.

fMin(*Expr*, *Var*) $\Rightarrow$ Expressão booleana

fMin(*Expr*, *Var*, *LimiteInferior*)

fMin(*Expr*, *Var*, *LimiteInferior*, *LimiteSuperior*)

fMin(*Expr*, *Var*) | *LimiteInferior* $\leq$ *Var* $\leq$ *LimiteSuperior*

Devolve uma expressão booleana que especifica os valores candidatos de *Var* que minimiza *Expr* ou localiza o maior limite inferior.

Pode utilizar o operador de limite (“|”) para limitar o intervalo da solução e/ou especificar outras restrições.

Para a definição Aproximado do modo **Auto ou Aproximado**, **fMin()** procura iterativamente um mínimo local aproximado. Isto é frequentemente mais rápido, em especial, se utilizar o operador “” para limitar a procura a um intervalo relativamente pequeno que contenha exactamente um mínimo local.

Nota: Consulte também **fMax()** e **min()**.

$\text{fMin}\left(1-(x-a)^2-(x-b)^2, x\right)$	$x=-\infty \text{ or } x=\infty$
$\text{fMin}\left(0.5 \cdot x^3 - x - 2, x\right), x \geq 1$	$x=1.$

For *Var, Baixo, Alto* [, *Passo*]*Bloco***EndFor**

Executa as declarações em *Bloco* iterativamente para cada valor de *Var*, de *Baixo* para *Alto*, em incrementos de *Passo*.

Var não tem de ser uma variável do sistema.

Passo pode ser positivo ou negativo. O valor predefinido é 1.

Bloco pode ser uma declaração ou uma série de declarações separadas pelo carácter ":".

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

Define g()	Done
Local tempsum, step, i	
0 → tempsum	
1 → step	
For i, 1, 100, step	
tempsum + i → tempsum	
EndFor	
EndFunc	

g()	5050
-----	------

format()**format**(*Expr* [, *CadeiaFormato*])
 ⇒ *cadeia*

Devolve *Expr* como uma cadeia de caracteres com base no modelo do formato.

Expr tem de ser simplificada para um número.

CadeiaFormato é uma cadeia e tem de estar na forma: "F[n]", "S[n]", "E[n]", "G[n][c]", em que [] indica porções opcionais.

F[n]: Formato fixo. n é o número de dígitos para visualizar o ponto decimal.

S[n]: Formato científico. n é o número de dígitos para visualizar o ponto decimal.

format(1.234567, "f3")	"1.235"
format(1.234567, "s2")	"1.23E0"
format(1.234567, "e3")	"1.235E0"
format(1.234567, "g3")	"1.235"
format(1234.567, "g3")	"1,234.567"
format(1.234567, "g3,r:")	"1:235"

E[n]: Formato de engenharia. n é o número de dígitos após o primeiro dígito significativo. O expoente é ajustado para um múltiplo de três e o ponto decimal é movido para a direita zero, um ou dois dígitos.

G[n][c]: Igual ao formato fixo mas também separa os dígitos à esquerda da raiz em grupos de três. c especifica o carácter do separador de grupos e predefine para uma vírgula. Se c for um ponto, a raiz será apresentada como uma vírgula.

[Rc]: Qualquer um dos especificadores acima pode ser sufixado com o marcador de raiz Rc, em que c é um carácter que especifica o que substituir pelo ponto da raiz.

fPart()

fPart(*Expr1*) $\Rightarrow$ expressão

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

fPart(*Lista1*) $\Rightarrow$ lista

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

fPart(*Matriz1*) $\Rightarrow$ matriz

Devolve a parte fraccionária do argumento.

Para uma lista ou matriz, devolve as partes fraccionárias dos elementos.

O argumento pode ser um número complexo ou real.

FPdf()

FPdf(*ValX*, *dfNumer*, *dfDenom*) $\Rightarrow$ número se *ValX* for um número, lista se *ValX* for uma lista

Calcula a probabilidade da distribuição F no *ValX* para o *dfNumer* (graus de liberdade) e o *dfDenom* especificados.

freqTable►list

(*Lista1*,*ListaNúmerosInteirosFreq*)⇒
lista

Apresenta uma lista com os elementos de *Lista1* expandida de acordo com as frequências em *ListaNúmerosInteirosFreq*. Esta função pode ser utilizada para construir uma tabela de frequência para a aplicação Dados e Estatística.

Lista1 pode ser qualquer lista válida.

ListaNúmerosInteirosFreq tem de ter a mesma dimensão da *Lista1* e só deve conter elementos de números inteiros não negativos. Cada elemento especifica o número de vezes que o elemento de *Lista1* correspondente é repetido na lista de resultados. Um valor de zero exclui o elemento de *Lista1* correspondente.

Nota: Pode introduzir esta função através da escrita de **freqTable@>list (...)** no teclado do computador.

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

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})	{1,2,2,2,2,4}

frequency()

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

Devolve uma lista que contém as contagens dos elementos em *Lista1*. As contagens são baseadas em intervalos (bins) definidos em *Listabins*.

Se *Listabins* for {b(1), b(2), ..., b(n)}, os intervalos especificados são {? ≤ b(1), b(1) < ? ≤ b(2), ..., b(n-1) < ? ≤ b(n), b(n) > ?}. A lista resultante é um elemento maior que *Listabins*.

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}

Explicação do resultado:

2 elementos da *Lista de dados* são ≤ 2.5

4 elementos da *Lista de dados* são >2.5 e ≤ 4.5

3 elementos da *Lista de dados* são >4.5

Cada elemento do resultado corresponde ao número de elementos de *Lista1* que estão no intervalo desse lote. Expresso em termos da função **countIf()**, o resultado é { countIf(list, ?≤ b(1)), countIf(lista, b(1)<?≤ b(2)), ..., countIf(lista, b(n-1)<?≤ b(n)), countIf(lista, b(n)>?) }.

O elemento "hello" é uma cadeia e não pode ser colocado em nenhum lote definido.

Elementos de *Lista1* que não podem ser "colocados num lote" são ignorados.

Elementos de *Lista1* que não podem ser "colocados num lote" são ignorados. Os elementos (nulos) vazios também são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

Na aplicação Listas e Folha de cálculo, pode utilizar um intervalo de células no lugar de ambos os argumentos.

Nota: Consulte também **countIf()**, página 38.

FTest_2Samp

FTest_2Samp *Lista1*, *Lista2* [, *Freq1* [, *Freq2* [, *Hipótese*]]]

FTest_2Samp *Lista1*, *Lista2* [, *Freq1* [, *Freq2* [, *Hipótese*]]]

(Entrada da lista de dados)

FTest_2Samp *sx1*, *n1*, *sx2*, *n2* [, *Hipótese*]

FTest_2Samp *sx1*, *n1*, *sx2*, *n2* [, *Hipótese*]

(Entrada estatística do resumo)

Efectua um teste F de duas amostras. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

ou $H_a: \sigma_1 > \sigma_2$, defina *Hipótese*>0

Para $H_a: \sigma_1 \neq \sigma_2$ (predefinição), defina *Hipótese*=0

Para $H_a: \sigma_1 < \sigma_2$, defina *Hipótese* < 0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.F	Estatística $\hat{U}$ calculada para a sequência de dados
stat.PVal	Menor nível de significância para o qual a hipótese nula pode ser rejeitada
stat.dfNumer	graus de liberdade do “numerador” = $n_1 - 1$
stat.dfDenom	graus de liberdade do “denominador” = $n_2 - 1$
stat.sx1, stat.sx2	Desvios padrão da amostra das sequências de dados em <i>Lista 1</i> e <i>Lista 2</i>
stat.x1_bar stat.x2_bar	Médias da amostra das sequência de dados em <i>Lista 1</i> e <i>Lista 2</i>
stat.n1, stat.n2	Tamanho das amostras

Func

Func

Definir uma função por ramos:

Bloco

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

Done

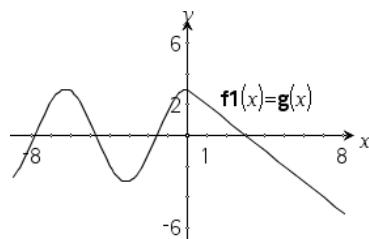
EndFunc

Modelo para criar uma função definida pelo utilizador.

Bloco pode ser uma declaração, uma série de declarações separadas pelo carácter “:” ou uma série de declarações em linhas separadas. A função pode utilizar a função **Return** para devolver um resultado específicos.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

Resultado do gráfico $g(x)$



gcd()	Catálogo > 
gcd (<i>Valor1</i> , <i>Valor2</i>) $\Rightarrow$ expressão	$\text{gcd}(18,33)$ 3
Devolve o máximo divisor comum dos dois argumentos. O gcd de duas fracções é o gcd dos numeradores divididos pelo lcm dos denominadores.	
No modo Auto ou Aproximado, o gcd dos números do ponto flutuante fraccionária é 1.0.	
gcd (<i>Lista1</i> , <i>Lista2</i>) $\Rightarrow$ lista	$\text{gcd}(\{12,14,16\},\{9,7,5\})$ { 3,7,1 }
Devolve os máximos divisores comuns dos elementos correspondentes em <i>Lista1</i> e <i>Lista2</i> .	
gcd (<i>Matriz1</i> , <i>Matriz2</i>) $\Rightarrow$ matriz	$\text{gcd}\left(\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}, \begin{pmatrix} 4 & 8 \\ 12 & 16 \end{pmatrix}\right)$ $\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$
Devolve os máximos divisores comuns dos elementos correspondentes em <i>Matriz1</i> e <i>Matriz2</i> .	

geomCdf()	Catálogo > 
geomCdf	
(p, LimiteInferior, LimiteSuperior) $\Rightarrow$ número se <i>LimiteInferior</i> e <i>LimiteSuperior</i> forem números, lista se <i>LimiteInferior</i> e <i>LimiteSuperior</i> forem listas	
geomCdf(p, LimiteSuperior) para $P(1 \leq X \leq \text{LimiteSuperior}) \Rightarrow$ número se <i>LimiteSuperior</i> for um número, lista se <i>LimiteSuperior</i> for uma lista	
Calcula uma probabilidade geométrica cumulativa do <i>LimiteInferior</i> ao <i>LimiteSuperior</i> com a probabilidade de sucesso especificada <i>p</i> .	
Para $P(X \leq \text{LimiteSuperior})$, defina <i>LimiteInferior</i> = 1.	

geomPdf()	Catálogo > 
geomPdf(p, ValX) $\Rightarrow$ número se <i>ValX</i> for um	

número, *lista* se *ValX* for uma lista

Calcula uma probabilidade em *ValX*, o número da tentativa em que ocorre o primeiro sucesso, para a distribuição geométrica discreta com a probabilidade de sucesso especificada *p*.

Get

Menu Hub

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

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

Programar comando: Recupera um valor de um conectado TI-Innovator™ Hub e atribui o valor à variável *var*.

O valor tem de ser pedido:

- Com antecedência, através de um comando **Send "READ ..."** .
— ou —
- Incorporando um pedido **"READ ..."** como o argumento *promptString* opcional. Este método permite-lhe utilizar um único comando para pedir e recuperar o valor.

Ocorre uma simplificação implícita. Por exemplo, uma cadeia recebida como "123" é interpretada como um valor numérico. Para preservar a cadeia, usar **GetStr** em vez de **Get**.

Se incluir o argumento opcional *statusVar*, é atribuído um valor com base no êxito da operação. Um valor de zero significa que não foram recebidos dados.

Na segunda sintaxe, o argumento *func*() permite que o programa armazene a cadeia recebida como uma definição de função. Esta sintaxe funciona como se o programa executasse o comando:

Exemplo: pedir o valor atual do sensor de nível de luz incorporado no hub. Usar **Get** para recuperar o valor e atribuí-lo à variável *lightval*.

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

Incorporar o pedido READ no comando **Get**.

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

Define $func(arg1, \dots, argn) = cadeia\ recebida$

O programa pode então usar a função definida $func()$.

Nota: pode usar o comando **Get** dentro de um programa definido pelo utilizador mas não dentro de uma função.

Nota: ver também **GetStr**, página 93 e **Send**, página 173.

getDenom()

Catálogo >

getDenom(Expr1) $\Rightarrow$ expressão

Transforma o argumento numa expressão que tem um denominador comum simplificado e, em seguida, devolve o denominador.

$getDenom\left(\frac{x+2}{y-3}\right)$	$y-3$
$getDenom\left(\frac{2}{7}\right)$	7
$getDenom\left(\frac{1}{x} + \frac{y^2+y}{y^2}\right)$	$x \cdot y$

getKey()

Catálogo >

**codeTouch([0|1]) $\Rightarrow$
Cadeiadevolvida**

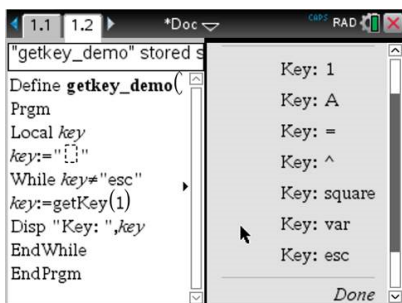
Descrição: **codeTouch()** - permite a um programa em TI-Basic obter introduções com o teclado - portátil, computador de secretária e emulador no computador de secretária.

Exemplo:

- teclapremida := **codeTouch()**
devolverá uma chave ou uma cadeia vazia se não tiver sido premida qualquer tecla. Esta chamada será devolvida de imediato.
- tecla premida := **codeTouch(1)** irá aguardar até ser premida uma tecla. Esta chamada irá colocar a execução do programa em pausa até ser premida uma tecla.

getKey()

Exemplo:



Processar batimentos de teclas:

Dispositivo portátil/tecla do emulador	Ambiente de trabalho	Valor devolvido
Esc	Esc	"esc"
Touchpad - Clique superior	N/D	"cima"
Ligar	N/D	"nome"
Scratch apps	N/D	"rascunho"
Touchpad - Clique do lado esquerdo	N/D	"esquerda"
Touchpad - Clique central	N/D	"centro"
Touchpad - Clique do lado direito	N/D	"direita"
Doc	N/D	"doc"
Tab	Tab	"tab"
Touchpad - Clique inferior	Seta para baixo	"baixo"
Menu	N/D	"menu"
Ctrl	Ctrl	sem devolução
Deslocar	Deslocar	sem devolução
Var	N/D	"var"
Eliminar	N/D	"eliminar"
=	=	"="
trig	N/D	"trig"
0 a 9	0-9	"0" ... "9"
Modelos	N/D	"modelo"
Catálogo	N/D	"cat"
^	^	"^"
X^2	N/D	"quadrado"
/ (tecla de divisão)	/	"/"
* (tecla de multiplicação)	*	"*"
e^x	N/D	"exp"

Dispositivo portátil/tecla do emulador	Ambiente de trabalho	Valor devolvido
10^x	N/D	"à potência de 10"
+	+	"+"
-	-	"_"
(	(	"("
)	)	")"
.	.	". "
(-)	N/D	"-" (sinal de negação)
Enter	Enter	"enter"
ee	N/D	"E" (notação científica E)
a - z	a-z	alfa = letra premida (minúsculas) ("a" - "z")
shift a-z	shift a-z	alfa = letra premida "A" - "Z"
		Nota: ctrl-shift ativa as maiúsculas
?!	N/D	"?!"
pi	N/D	"pi"
Marcador	N/D	sem devolução
,	,	" , "
Return	N/D	"return"
Espaço	Espaço	" " (espaço)
Inacessível	Caracteres especiais como @,!,^, etc.	O carácter é devolvido
N/D	Teclas de função	Nenhum carácter devolvido
N/D	Teclas de controlo do ambiente de trabalho especiais	Nenhum carácter devolvido
Inacessível	As restantes teclas do ambiente de trabalho que	O mesmo carácter que obtém em Notas (e não

Dispositivo portátil/tecla do emulador	Ambiente de trabalho	Valor devolvido
	não estão disponíveis na calculadora durante codeTouch() aguardam uma tecla pressionada. {, },,, ;, ...)	numa caixa matemática)

Nota: é importante salientar que a presença de **codeTouch()** num programa altera a forma como alguns eventos são tratados pelo sistema. Alguns destes eventos são descritos em seguida.

Terminar programa e processar evento - Exatamente como se o utilizador abrisse o programa premindo a tecla **ON**

"**Suporte**" abaixo significa - O sistema funciona como previsto - o programa continua a ser executado.

Evento	Dispositivo	Ambiente de trabalho - Software TI-Nspire™ do aluno
Consulta rápida	Terminar programa, processar evento	Da mesma forma que no portátil (TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software apenas)
Gestão de ficheiros remota (Incl. o envio do ficheiro 'Exit Press 2 Test' de outro portátil ou computador de secretária-portátil)	Terminar programa, processar evento	Da mesma forma que no portátil. (TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software apenas)
Terminar aula	Terminar programa, processar evento	Suporte (TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software apenas)

Evento	Dispositivo	Ambiente de trabalho - TI-Nspire™ Todas as versões
TI-Innovator™ Hub ligar/desligar	Suporte - Pode gerar comandos com êxito para TI-Innovator™ Hub. Depois de sair do programa, TI-Innovator™ Hub ainda está a funcionar com o portátil.	Da mesma forma que no portátil.

getLangInfo()⇒*abreviatura*

Apresenta uma abreviatura do nome do idioma activo. Por exemplo, pode utilizá-lo num programa ou função para determinar o idioma actual.

- Inglês = “en”
- Dinamarquês = “da”
- Alemão = “de”
- Finlandês = “fi”
- Francês = “fr”
- Italiano = “it”
- Holandês = “nl”
- Flamengo = “nl_BE”
- Norueguês = “no”
- Português = “pt”
- Espanhol = “es”
- Sueco = “sv”

getLangInfo()	"en"
---------------	------

getLockInfo(Var)⇒*valor*

Devolve o estado de bloqueio/desbloqueio actual da variável *Var*.

valor =0: *Var* está desbloqueada ou não existe.

valor =1: *Var* está bloqueada e não pode ser modificada nem eliminada.

Consulte **Lock**, página 116, **eunLock**, página 216.

<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(NúmeroInteiroNomeModo)
⇒ *valor*

```
getMode(0)
{ 1,7,2,1,3,1,4,1,5,1,6,1,7,1,8,1 }
```

getMode(0) ⇒ *lista*

```
getMode(1) 7
```

getMode(NúmeroInteiroNomeModo)
devolve um valor que representa a
definição actual do modo
NúmeroInteiroNomeModo.

```
getMode(8) 1
```

getMode(0) devolve uma lista com os
pares de números. Cada par é composto
por um número inteiro do modo e um
número inteiro da definição.

Para uma listagem dos modos e das
definições, consulte a tabela abaixo.

Se guardar as definições com **getMode(0)**
→ *var*, pode utilizar **setMode(var)** num
programa ou função para restaurar
temporariamente as definições na
execução da função ou do programa.
Consulte **setMode()**, página 177.

Nome do modo	Número inteiro do modo	Números inteiros da definição
Ver dígitos	1	1 =Flutuante, 2 =Flutuante1, 3 =Flutuante2, 4 =Flutuante3, 5 =Flutuante4, 6 =Flutuante5, 7 =Flutuante6, 8 =Flutuante7, 9 =Flutuante8, 10 =Flutuante9, 11 =Flutuante10, 12 =Flutuante11, 13 =Flutuante12, 14 =Fixo0, 15 =Fixo1, 16 =Fixo2, 17 =Fixo3, 18 =Fixo4, 19 =Fixo5, 20 =Fixo6, 21 =Fixo7, 22 =Fixo8, 23 =Fixo9, 24 =Fixo10, 25 =Fixo11, 26 =Fixo12
Ângulo	2	1 =Radianos, 2 =Graus, 3 =Gradianos
Formato exponencial	3	1 =Normal, 2 =Científica, 3 =Engenharia
Real ou Complexo	4	1 =Real, 2 =Rectangular, 3 =Polar
Auto or Aprox.	5	1 =Auto, 2 =Aproximado, 3 =Exacto
Formato vectorial	6	1 =Rectangular, 2 =Cilíndrico, 3 =Esférico
Base	7	1 =Decimal, 2 =Hex, 3 =Binário
Sistema de unidades	8	1 =SI, 2 =Eng/EUA

getNum()	Catálogo > 	
getNum(<i>Expr1</i>) ⇒ <i>expressão</i>		
Transforma o argumento numa expressão que tem um denominador comum simplificado e, em seguida, devolve o numerador.		
	$\text{getNum}\left(\frac{x+2}{y-3}\right)$	$x+2$
	$\text{getNum}\left(\frac{2}{7}\right)$	2
	$\text{getNum}\left(\frac{1}{x} + \frac{1}{y}\right)$	$x+y$

GetStr	Hub Menu	
GetStr [<i>promptString</i> ,] <i>var</i> [, <i>statusVar</i>]	Para exemplos, ver Get .	
GetStr [<i>promptString</i> ,] <i>func</i> (<i>arg1</i> , ... <i>argn</i>) [, <i>statusVar</i>]		
Programar comando: funciona de forma idêntica ao comando Get , mas o valor recuperado é sempre interpretado como uma cadeia. Em contraste, o comando Get interpreta a resposta como uma expressão a não ser que esteja entre aspas ("").		
Nota: ver também Get , página 85 e Send , página 173.		

getType()	Catálogo > 	
getType(<i>var</i>) ⇒ <i>cadeia de texto</i>		
Apresenta uma cadeia de texto que indica o tipo de dados da variável <i>var</i> .		
Se <i>var</i> não tiver sido definido, apresenta a cadeia de texto "NENHUM".		
	$\{1,2,3\} \rightarrow temp$	$\{1,2,3\}$
	$\text{getType}(temp)$	"LIST"
	$3 \cdot i \rightarrow temp$	$3 \cdot i$
	$\text{getType}(temp)$	"EXPR"
	$\text{DelVar } temp$	<i>Done</i>
	$\text{getType}(temp)$	"NONE"

getVarInfo()⇒matriz ou palavra

getVarInfo
(*CadeiaDoNomeDaBiblioteca*)⇒matriz
ou palavra

getVarInfo() devolve uma matriz de informações (nome da variável, tipo, acessibilidade da biblioteca e estado de bloqueio/desbloqueio) para todas as variáveis e os objectos da biblioteca definidos no problema actual.

Se não definir nenhuma variável,
getVarInfo() apresenta a palavra

getVarInfo
(*NomeDaBiblioteca*)apresenta uma
matriz com informações para todos os
objectos da biblioteca definidos na
biblioteca
CadeiaDoNomeDaBiblioteca.
CadeiaDoNomeDaBiblioteca tem de
ser uma palavra (texto entre aspas) ou
uma variável da frase.

Se a biblioteca
CadeiaDoNomeDaBiblioteca não
existir, ocorre um erro.

Veja o exemplo do lado esquerdo, em
que o resultado de **getVarInfo()** é
atribuído à variável *vs*. A tentar de
apresentação da linha 2 ou da linha 3 de
vs apresenta uma mensagem de erro de
“Matriz ou lista inválida” porque pelo
menos um dos elementos nessas linhas
(variável *b*, por exemplo) reavalia-se
para uma matriz.

Este erro pode também ocorrer quando
utilizar *Ans* para reavaliar um resultado
getVarInfo().

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><tr><td>x</td><td>"NUM"</td><td>"{"}</td><td>1</td></tr><tr><td>y</td><td>"LIST"</td><td>"LibPriv "</td><td>0</td></tr><tr><td>z</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($tmp3$)	"Error: Argument must be a string"												
getVarInfo("tmp3")	<table><tr><td>$volcyl2$</td><td>"NONE"</td><td>"LibPub "</td><td>0</td></tr></table>	$volcyl2$	"NONE"	"LibPub "	0								
$volcyl2$	"NONE"	"LibPub "	0										

$a:=1$	1												
$b:=[1\ 2]$	$[1\ 2]$												
$c:=[1\ 3\ 7]$	$[1\ 3\ 7]$												
$vs:=\text{getVarInfo}()$	<table><tr><td>a</td><td>"NUM"</td><td>"{"</td><td>0</td></tr><tr><td>b</td><td>"MAT"</td><td>"{"</td><td>0</td></tr><tr><td>c</td><td>"MAT"</td><td>"{"</td><td>0</td></tr></table>	a	"NUM"	"{"	0	b	"MAT"	"{"	0	c	"MAT"	"{"	0
a	"NUM"	"{"	0										
b	"MAT"	"{"	0										
c	"MAT"	"{"	0										
$vs[1]$	$[1\ \text{"NUM"}\ \text{"{"}\ 0]$												
$vs[1,1]$	1												
$vs[2]$	"Error: Invalid list or matrix"												
$vs[2,1]$	$[1\ 2]$												

O sistema apresenta o erro acima porque a versão actual do software não suporta uma estrutura de matriz generalizada em que um elemento de uma matriz pode ser uma matriz ou uma lista.

Goto

Goto *NomeDefinição*

Transfere o controlo para a definição *NomeDefinição*.

NomeDefinição tem de ser definido na mesma função com uma instrução **Lbl**.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

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

►Grad

Expr1 ►Grad $\Rightarrow$ expressão

Converte *Expr1* para medição do ângulo de gradianos.

Nota: Pode introduzir este operador através da escrita de @►Grad no teclado do computador.

No modo de ângulo Graus:

$(1.5) \blacktriangleright \text{Grad}$	$(1.66667)^g$
---	---------------

No modo de ângulo Radianos:

$(1.5) \blacktriangleright \text{Grad}$	$(95.493)^g$
---	--------------

I

identity ()

identity(*Número inteiro*) $\Rightarrow$ matriz

Devolve a matriz identidade com uma dimensão de *Número inteiro*.

Número inteiro tem de ser um número natural.

identity(4)	<table border="1"> <tr><td>1</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>1</td></tr> </table>	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
1	0	0	0														
0	1	0	0														
0	0	1	0														
0	0	0	1														

If *BooleanExpr* *Declaração*

If *ExprBooleana* Then *Bloco*

EndIf

Se a *ExprBooleana* for avaliada como verdadeira, executa a declaração individual *Declaração* ou o bloco de declarações *Bloco* antes de continuar a execução.

Se a *ExprBooleana* for avaliada como falsa, continua a execução sem executar a declaração ou o bloco de declarações.

Bloco pode ser uma declaração ou uma sequência de declarações separadas pelo carácter ":" .

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

If *ExprBooleana* Then *Bloco1*

Else *Bloco2*

EndIf

Se a *ExprBooleana* for avaliada como verdadeira, executa o *Bloco1* e ignora o *Bloco2*.

Se a *ExprBooleana* for avaliada como falsa, ignora o *Bloco1* , mas executa o *Bloco2*.

Bloco1 e *Bloco2* podem ser uma declaração única.

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

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

```

If ExprBooleana1 Then
    Bloco1
ElseIf ExprBooleana2 Then
    Bloco2
:
ElseIf ExprBooleanaN Then
    BlocoN
EndIf

```

Permite a derivação. Se a *ExprBooleana1* for avaliada como verdadeira, executa o *Bloco1*. Se a *ExprBooleana1* for avaliada como falsa, avalia a *ExprBooleana2*, etc.

```

Define  $g(x)$  = Func
    If  $x < 5$  Then
        Return 5
    ElseIf  $x > 5$  and  $x < 0$  Then
        Return  $\neg x$ 
    ElseIf  $x \geq 0$  and  $x \neq 10$  Then
        Return  $x$ 
    ElseIf  $x = 10$  Then
        Return 3
    EndIf
EndFunc

```

Done

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

ifFn ()

```

ifFn(ExprBooleana, Value_If_true
[, Value_If_false [, Value_If_unknown]])
⇒ expressão, lista ou matriz

```

Avalia a expressão booleana *ExprBooleana* (ou cada elemento da *ExprBooleana*) e produz um resultado com base nas seguintes regras:

- *ExprBooleana* pode testar um valor individual, uma lista ou uma matriz.
- Se um elemento da *ExprBooleana* for avaliado como verdadeiro, devolve o elemento correspondente de *Value_If_true*.
- Se um elemento da *ExprBooleana* for avaliado como falsa, devolve o elemento correspondente de *Value_If_false*. Se omitir *Value_If_false*, devolve undef.
- Se um elemento da *ExprBooleana* não for verdadeiro nem falso, devolve o elemento correspondente *Value_If_unknown*. Se omitir *Value_If_unknown*, devolve undef.
- Se o segundo, o terceiro ou o quarto argumento da função **ifFn()** for uma expressão individual, o teste booleano é aplicado a todas as posições da

```

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

```

O valor do teste de **1** é inferior a 2.5, por esta razão, o elemento

Value_If_True correspondente de **5** é copiado para a lista de resultados.

O valor do teste de **2** é inferior a 2.5, por esta razão, o elemento

Value_If_True correspondente de **6** é copiado para a lista de resultados.

O valor do teste de **3** não é inferior a 2.5, por esta razão, o elemento *Value_If_False* correspondente de **10** é copiado para a lista de resultados.

```

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

```

Value_If_true é um valor individual e corresponde a qualquer posição seleccionada.

```

ifFn({1,2,3}<2.5,{5,6,7})
{5,6,undef}

```

ExprBooleana.

Nota: Se a declaração *ExprBooleana* simplificada envolver uma lista ou matriz, todos os outros argumentos da lista ou matriz têm de ter as mesmas dimensões e o resultado terá as mesmas dimensões.

Value_If_false não é especificado. Undef é utilizado.

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

$$\{6, "err"\}$$

Um elemento seleccionado de *Value_If_true*.
Um elemento seleccionado de *Value_If_*
unknown.

imag()

imag(*ExprI*) $\Rightarrow$ expressão

Devolve a parte imaginária do argumento.

Nota: Todas as variáveis indefinidas são tratadas como variáveis reais. Consulte também *real()*, página 159

imag(*ListaI*) $\Rightarrow$ lista

Devolve uma lista de partes imaginárias dos elementos.

imag(*MatrizI*) $\Rightarrow$ matriz

Devolve uma matriz das partes imaginárias dos elementos.

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

$$\text{imag}(z)$$

$$\text{imag}(x+i \cdot y)$$

$$2$$

$$0$$

$$y$$

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

$$\{0, -1, 1\}$$

$$\text{imag}\left(\begin{bmatrix} a & b \\ i \cdot c & i \cdot d \end{bmatrix}\right)$$

$$\begin{bmatrix} 0 & 0 \\ c & d \end{bmatrix}$$

impDif()

impDif(*Equação*, *Var*, *VarDependente* [, *Ord*]) $\Rightarrow$ expressão

em que a ordem *Ord* predefine-se para 1.

Calcula a derivada implícita para equações em que uma variável é definida implicitamente nos termos de outra.

$$\text{impDif}(x^2+y^2=100, x, y)$$

$$\frac{-x}{y}$$

indirecta

Consultar #(), página 247.

inString ()**Catálogo >** 

inString(*CadeiaDeOrigem*,
CadeiaSecundária[, *Início*]) ⇒ *número inteiro*

inString ("Hello there", "the")	7
inString ("ABCEFG", "D")	0

Devolve a posição do carácter na cadeia *CadeiaDeOrigem* em que começa a primeira ocorrência da cadeia *CadeiaSecundária*.

Início, se incluído, especifica a posição do carácter na *CadeiaDeOrigem* em que começa a procura. Predefinição = 1 (o primeiro carácter de *CadeiaDeOrigem*).

Se *CadeiaDeOrigem* não contiver *CadeiaSecundária* ou *Início* for > o comprimento de *CadeiaDeOrigem*, devolve zero.

int ()**Catálogo >** 

int(*Expr*) ⇒ *número inteiro*

int(*List1*) ⇒ *lista*

int(*Matrix1*) ⇒ *matriz*

int (-2.5)	-3.
int ([-1.234 0 0.37])	[-2. 0 0.]

Devolve o maior número inteiro que é igual ou inferior ao argumento. Esta função é idêntica a **floor()**.

O argumento pode ser um número complexo ou real.

Para uma lista ou matriz, devolve o maior número inteiro de cada elemento.

intDiv ()**Catálogo >** 

intDiv(*Number1*, *Number2*) ⇒ *número inteiro*

intDiv(*List1*, *List2*) ⇒ *lista*

intDiv(*Matrix1*, *Matrix2*) ⇒ *matriz*

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

Devolve a parte do número inteiro assinada de (*Número1* ÷ *Número2*).

Para listas e matrizes, devolve a parte do número inteiro assinada de (argumento 1 ÷ argumento 2) para cada par de elementos.

interpolar(*xValue*, *xList*, *yList*, *yPrimeList*) $\Rightarrow$ *lista*

Esta função efectua o seguinte:

Dado *xList*, *yList*=**f**(*xList*) e *yPrimeList*=**f'**(*xList*) para alguma função **f** desconhecida, é utilizada uma interpolante cúbica para aproximar a função **f** em *xValue*. Presume-se que *xList* é uma lista de números estritamente crescentes ou decrescentes, mas esta função pode apresentar um valor mesmo quando não o seja. Esta função percorre *xList* procurando por um intervalo [*xList*[*i*], *xList*[*i*+1]] que contenha *xValue*. Se encontrar tal intervalo, apresenta um valor interpolado para **f**(*xValue*); caso contrário, apresenta **.undef**.

xList, *yList* e *yPrimeList* têm de ter a mesma dimensão ≥ 2 e conter expressões que simplificam para números.

xValue pode ser uma variável indefinida, um número ou uma lista de números.

Equação diferencial:

$$y' = -3 \cdot y + 6 \cdot t + 5 \text{ e } y(0) = 5$$

$$rk := rk23(-3 \cdot y + 6 \cdot t + 5, t, y, \{0, 10\}, 5, 1)$$

0.	1.	2.	3.	4.
5.	3.19499	5.00394	6.99957	9.00593

Para ver o resultado completo, prima $\blacktriangle$ e, de seguida, utilize $\blacktriangleleft$ e $\blacktriangleright$ para mover o cursor.

Utilize a função de interpolação() para calcular os valores de função para *xvalueList*:

$$xvalueList := seq(i, i, 0, 10, 0.5)$$

$$\{0, 0.5, 1., 1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5., 5.5, 6., 6.5, \blacktriangleright\}$$

$$xlist := mat \blacktriangleright list(rk[1])$$

$$\{0., 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.\}$$

$$ylist := mat \blacktriangleright list(rk[2])$$

$$\{5., 3.19499, 5.00394, 6.99957, 9.00593, 10.9978\}$$

$$yprimelist := -3 \cdot y + 6 \cdot t + 5 | y = ylist \text{ and } t = xlist$$

$$\{-10., 1.41503, 1.98819, 2.00129, 1.98221, 2.006\}$$

$$interpolate(xvalueList, xlist, ylist, yprimelist)$$

$$\{5., 2.67062, 3.19499, 4.02782, 5.00394, 6.00011, \blacktriangleright\}$$

invχ²(*Área*, *df*)

invChi2(*Área*, *df*)

Calcula a função de probabilidade cumulativa inversa χ^2 (Qui quadrado) especificada pelo grau de liberdade, df para uma determinada *Area* debaixo da curva

invF(*Área*,*Numerdf*,*Denomdf*)

invF(*Área*,*Numerdf*,*Denomdf*)

calcula a função de distribuição cumulativa inversa F especificada pelo *dfNumer* e o *dfDenom* para uma determinada *Area* debaixo da curva.

invBinom
(*CumulativeProb*,*NumTrials*,*Prob*,
OutputForm) $\Rightarrow$ *escalar* ou *matriz*

Dado o número de tentativas (*NumTrials*) e a probabilidade de sucesso de cada tentativa (*Prob*), esta função devolve o número mínimo de sucessos, k , de forma a que a probabilidade cumulativa de k sucessos seja igual ou superior à probabilidade cumulativa dada (*CumulativeProb*).

OutputForm=0, apresenta o resultado como uma *escalar* (predefinição).

OutputForm=1, apresenta o resultado como uma *matriz*.

Exemplo: A Maria e o Carlos estão a jogar aos dados. A Maria tem de adivinhar o número máximo de vezes que o 6 aparece em 30 jogadas. Se o número 6 aparecer esse número de vezes ou menos, a Maria ganha. Além disso, quanto menor for o número que ela adivinhar, maiores serão os seus ganhos. Qual é o número mais pequeno que a Maria consegue adivinhar se ela quiser que a probabilidade de ganhar seja superior a 77%?

invBinom $\left(0.77, 30, \frac{1}{6}\right)$	6
invBinom $\left(0.77, 30, \frac{1}{6}, 1\right)$	$\begin{bmatrix} 5 & 0.616447 \\ 6 & 0.776537 \end{bmatrix}$

invBinomN(*CumulativeProb*,*Prob*,
NumSuccess,*OutputForm*) $\Rightarrow$ *scalar* ou *matriz*

Exemplo: A Mónica está a praticar lançamentos para netball. Sabe por experiência própria que as suas hipóteses de acertar um lançamento são de 70%. Ela pretende praticar até conseguir 50 acertos. Quantos lançamentos tem de tentar para garantir que a probabilidade de obter pelo menos 50 acertos seja superior a 0,99?

invBinomN()Catálogo > 

Dada a probabilidade de sucesso de cada tentativa (*Prob*) e o número de sucessos (*NumSuccess*), esta função devolve o número mínimo de tentativas, N , de forma a que a probabilidade cumulativa de x sucessos é inferior ou igual à probabilidade cumulativa dada (*CumulativeProb*).

$\text{invBinomN}(0.01, 0.7, 49)$	86
$\text{invBinomN}(0.01, 0.7, 49, 1)$	$\begin{bmatrix} 85 & 0.010451 \\ 86 & 0.00709 \end{bmatrix}$

OutputForm=0, apresenta o resultado como uma escalar (predefinição).

OutputForm=1, apresenta o resultado como uma matriz.

invNorm()Catálogo > 

invNorm(*Área*[, μ],[σ]])

Calcula a função de distribuição normal cumulativa inversa para uma determinada *Área* mediante a curva de distribuição normal especificada por μ e σ .

invT()Catálogo > 

invT(*Área*,*df*)

Calcula a função de probabilidade inversa cumulativa da t-student especificada pelo grau de liberdade, *df* para uma determinada *Área* sob a curva.

iPart ()Catálogo > 

iPart(*Number*) $\Rightarrow$ *número inteiro*

iPart(*List1*) $\Rightarrow$ *lista*

iPart(*Matrix1*) $\Rightarrow$ *matriz*

$\text{iPart}(-1.234)$	-1.
$\text{iPart}\left(\left\{\frac{3}{2}, -2.3, 7.003\right\}\right)$	$\{1, -2., 7.\}$

Devolve a parte do número inteiro do argumento.

Para listas e matrizes, devolve a parte do número inteiro de cada elemento.

O argumento pode ser um número complexo ou real.

irr(*CF0*,*ListaCF* [,*FreqCF*]) $\Rightarrow$ *valor*

Função financeira que calcula a taxa de retorno interna de um investimento.

CF0 é o cash flow inicial no momento 0; tem de ser um número real.

ListaCF é uma lista de montantes de cash flow após o cash flow inicial *CF0*.

FreqCF é uma lista opcional em que cada elemento especifica a frequência da ocorrência para um montante de cash flow agrupado (consecutivo), que é o elemento correspondente de *ListaCF*. A predefinição é 1; se introduzir valores, têm de ser números inteiros positivos < 10.000.

Nota: Consulte também **mirr()**, página 126.

<i>list1</i> := { 6000, -8000, 2000, -3000 }	
	{ 6000, -8000, 2000, -3000 }
<i>list2</i> := { 2, 2, 2, 1 }	{ 2, 2, 2, 1 }
irr (5000, <i>list1</i> , <i>list2</i>)	-4.64484

isPrime()

isPrime(*Número*) $\Rightarrow$ *Expressão constante booleana*

Devolve verdadeiro ou falso para indicar se o *número* é um número inteiro ≥ 2 que é divisível apenas por si e 1.

Se o *Número* exceder cerca de 306 dígitos e não tiver factores ≤ 1021 , **isPrime**(*Número*) mostra uma mensagem de erro.

Se quiser apenas determinar se o *Número* é primo, utilize **isPrime()** em vez de **factor()**. É muito mais rápido, em especial, se o *Número* não for primo e tiver um segundo factor maior que exceda cerca de cinco dígitos.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

isPrime (5)	true
isPrime (6)	false

Função para localizar o número primeiro seguinte após um número especificado:

Define <i>nextprim</i> (<i>n</i>) = Func	<i>Done</i>
Loop	
$n + 1 \rightarrow n$	
If isPrime (<i>n</i>)	
Return <i>n</i>	
EndLoop	
EndFunc	
<i>nextprim</i> (7)	11

isVoid(Var) $\Rightarrow$ Expressão constante
booleana

isVoid(Expr) $\Rightarrow$ Expressão constante
booleana

isVoid(List) $\Rightarrow$ lista de expressões
constantes booleanas

Devolve verdadeiro ou falso para indicar se o argumento é um tipo de dados nulos.

Para mais informações sobre elementos nulos, consulte página 274.

$a := _$	$_$
$\text{isVoid}(a)$	true
$\text{isVoid}(\{1, _, 3\})$	$\{ \text{false}, \text{true}, \text{false} \}$

L

Lbl NomeDefinição

Define uma definição com o nome *NomeDefinição* numa função.

Pode utilizar uma instrução **Goto** *NomeDefinição* para transferir o controlo para a instrução imediatamente a seguir à definição.

NomeDefinição tem de cumprir os mesmos requisitos de nomeação do nome de uma variável.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

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(Número1, Número2) $\Rightarrow$ expressão

lcm(Lista1, Lista2) $\Rightarrow$ lista

lcm(Matriz1, Matriz2) $\Rightarrow$ matriz

$\text{lcm}(6, 9)$	18
$\text{lcm}\left(\left\{\frac{1}{3}, -14, 16\right\}, \left\{\frac{2}{15}, 7, 5\right\}\right)$	$\left\{\frac{2}{3}, 14, 80\right\}$

Devolve o mínimo múltiplo comum dos dois argumentos. O **lcm** de duas fracções é o **lcm** dos numeradores divididos pelo **gcd** dos denominadores. O **lcm** dos números de ponto flutuante fraccionários é o produto.

Para duas listas ou matrizes, devolve os mínimos múltiplos comuns dos elementos correspondentes.

left()

left(*CadeiaDeOrigem* [, *Num*]) $\Rightarrow$ *cadeia*

$\text{left}(\text{"Hello"}, 2)$ "He"

Devolve os caracteres *Num* mais à esquerda contidos na cadeia de caracteres *CadeiaDeOrigem*.

Se omitir *Num*, devolve todos os caracteres de *CadeiaDeOrigem*.

left(*Lista1* [, *Num*]) $\Rightarrow$ *lista*

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

Devolve os elementos *Num* mais à esquerda em *Lista1*.

Se omitir *Num*, devolve todos os elementos de *Lista1*.

left(*Comparação*) $\Rightarrow$ *expressão*

$\text{left}(x < 3)$ *x*

Devolve o lado esquerdo de uma equação ou desigualdade.

libShortcut()

libShortcut
(*CadeiaDoNomeDaBiblioteca*,
CadeiaDoNomeDoAtalho [,
MarcadorDeBibPriv]) $\Rightarrow$ *lista de*
variáveis

Este exemplo assume um documento de biblioteca actualizado e guardado adequadamente denominado **linalg2** que contém objectos definidos como *clearmat*, *gauss1* e *gauss2*.

Cria um grupo de variáveis no problema actual que contém referências a todos os objectos no documento da biblioteca especificado

CadeiaDoNomeDaBiblioteca. Adiciona também os membros do grupo ao menu Variáveis. Pode referir-se a cada objecto com a *CadeiaDoNomeDoAtalho*.

Definir

MarcadorDeBibliotecaPrivada=0 para excluir objectos da biblioteca privada (predefinição)

Definir

MarcadorDeBibliotecaPrivada=1 para incluir objectos da biblioteca privada

Para copiar um grupo de variáveis, consulte **CopyVar**, página 32.

Para eliminar um grupo de variáveis, consulte **DelVar**, página 53.

```
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}
```

limit() ou lim()

limit(*Expr1*, *Var*, *Ponto* [, *Direcção*])
 $\Rightarrow$ expressão

limit(*Lista1*, *Var*, *Ponto* [, *Direcção*])
 $\Rightarrow$ lista

limit(*Matriz1*, *Var*, *Ponto* [, *Direcção*])
 $\Rightarrow$ matriz

Devolve o limite requerido.

Nota: Consulte também **Modelo do limite**, página 7.

Direcção: negativa=da esquerda, positiva=da direita, caso contrário=ambos. (Se omitida, a *Direcção* predefine-se para ambos.)

Os limites no ∞ positivo e no ∞ negativo são sempre convertidos para limites de um lado do lado finito.

$\lim_{x \rightarrow 5} (2 \cdot x + 3)$ 13

$\lim_{x \rightarrow 0^+} \left(\frac{1}{x} \right)$ ∞

$\lim_{x \rightarrow 0} \left(\frac{\sin(x)}{x} \right)$ 1

$\lim_{h \rightarrow 0} \left(\frac{\sin(x+h) - \sin(x)}{h} \right)$ $\cos(x)$

$\lim_{n \rightarrow \infty} \left(\left(1 + \frac{1}{n} \right)^n \right)$ e

Dependendo das circunstâncias, **limit()** devolve-se ou **undef** quando não consegue determinar um limite único. Isto não significa necessariamente que não existe um limite único. **undef** significa que o resultado é um número desconhecido com a magnitude finita ou infinita, ou é um conjunto completo desses números.

limit() utiliza método como a regra L'Hopital's, logo existem limites únicos que não consegue determinar. Se a *Expr1* contiver variáveis indefinidas diferentes de *Var*, pode ter de as limitar para obter um resultado mais conciso.

Os limites podem ser muito sensíveis ao erro de arredondamento. Quando possível, evite a definição Aproximado do modo **Auto** ou **Aproximado** e os números aproximados quando calcular os limites. Caso contrário, os limites que devem ser zero ou ter magnitude infinita provavelmente não terão, e os limites que devem ter magnitude diferente de zero finita podem não ter.

$\lim_{x \rightarrow \infty} (a^x)$	undef
$\lim_{x \rightarrow \infty} (a^x) a > 1$	∞
$\lim_{x \rightarrow \infty} (a^x) a > 0 \text{ and } a < 1$	0

LinRegBx

LinRegBx *X,Y,[Freq],[Categoria,Incluir]*

Calcula a regressão lineary = a+b · xa partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

X e *Y* são listas de variáveis independentes e dependentes.

Freq é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

Categoria é uma lista de códigos de categorias para os dados X e Y correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $a+b \cdot x$
stat.a, stat.b	Parâmetros de regressão
stat.r ²	Coeficiente de determinação
stat.r	Coeficiente de correlação
stat.Resid	Resíduos da regressão
stat.XReg	Lista de dados na <i>Lista X</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.YReg	Lista de dados na <i>Lista Y</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.FreqReg	Lista de frequências correspondentes a <i>stat.XReg</i> e <i>stat.YReg</i>

LinRegMx $X,Y,[Freq],[Categoria,Incluir]$

Calcula a regressão linear $y = m \cdot x + b$ a partir das listas X e Y com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

X e Y são listas de variáveis independentes e dependentes.

Freq é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

Categoria é uma lista de códigos de categorias para os dados *X* e *Y* correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $m \cdot x + b$
stat.m, stat.b	Parâmetros de regressão
stat.r ²	Coefficiente de determinação
stat.r	Coefficiente de correlação
stat.Resid	Resíduos da regressão
stat.XReg	Lista de dados na <i>Lista X</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.YReg	Lista de dados na <i>Lista Y</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.FreqReg	Lista de frequências correspondentes a <i>stat.XReg</i> e <i>stat.YReg</i>

LinRegtIntervals *X*,*Y* [,*F* [,0 [,*NivC*]]]

Para declive. Calcula o intervalo de confiança de nível *C* do declive.

LinRegtIntervals *X*,*Y* [,*F* [,1 [,*ValX* [,*NivC*]]]]

Para resposta. Calcula um valor y previsto, um intervalo de previsão de nível C para uma observação, e um intervalo de confiança de nível C para a resposta média.

Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão.

X e Y são listas de variáveis independentes e dependentes.

F é uma lista opcional de valores de frequência. Cada elemento em F especifica a frequência de ocorrência para cada ponto de dados X e Y correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros ≥ 0 .

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $a+b \cdot x$
stat.a, stat.b	Parâmetros de regressão
stat.df	Graus de liberdade
stat.r ²	Coeficiente de determinação
stat.r	Coeficiente de correlação
stat.Resid	Resíduos da regressão

Apenas para o tipo de declive

Variável de saída	Descrição
[stat.CLower, stat.CUpper]	Intervalo de confiança para o declive
stat.ME	Margem de erro do intervalo de confiança
stat.SESlope	Erro padrão do declive
stat.s	Erro padrão sobre a linha

Apenas para o tipo de resposta

Variável de saída	Descrição
[stat.CLower, stat.CUpper]	Intervalo de confiança para a resposta média
stat.ME	Margem de erro do intervalo de confiança
stat.SE	Erro padrão da resposta média
[stat.LowerPred, stat.UpperPred]	Intervalo de previsão para uma observação
stat.MEPred	Margem de erro do intervalo de previsão
stat.SEPred	Erro padrão para previsão
stat. $\hat{y}$	$a + b \cdot X_{Val}$

LinRegtTest

Catálogo > 

LinRegtTest $X, Y[, Freq[, Hipótese]]$

Calcula uma regressão linear a partir das listas X e Y e um teste t no valor do declive β e o coeficiente de correlação ρ para a equação $y = \alpha + \beta x$. Testa a hipótese nula $H_0: \beta = 0$ (equivalentemente, $\rho = 0$) em relação a uma das três hipóteses alternativas.

Todas as listas têm de ter a mesma dimensão.

X e Y são listas de variáveis independentes e dependentes.

$Freq$ é uma lista opcional de valores de frequência. Cada elemento em $Freq$ especifica a frequência de ocorrência para cada ponto de dados X e Y correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

$Hipótese$ é um valor opcional que especifica uma de três hipóteses alternativas em relação à qual a hipótese nula ($H_0: \beta = \rho = 0$) será testada.

Para $H_a: \beta \neq 0$ e $\rho \neq 0$ (predefinição), defina $Hipótese = 0$

Para $H_a: \beta < 0$ e $\rho < 0$, defina $Hipótese < 0$

Para $H_a: \beta > 0$ e $\rho > 0$, defina $Hipótese > 0$

Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $a + b \cdot x$
stat.t	<i>t</i> -Estatística para teste de importância
stat.PVal	Menor nível de significância para o qual a hipótese nula pode ser rejeitada
stat.df	Graus de liberdade
stat.a, stat.b	Parâmetros de regressão
stat.s	Erro padrão sobre a linha
stat.SESlope	Erro padrão do declive
stat.r ²	Coefficiente de determinação
stat.r	Coefficiente de correlação
stat.Resid	Resíduos da regressão

linSolve()

linSolve(*SistemaDeEquaçõesLineares*, *Var1*, *Var2*, ...) ⇒ lista

$$\text{linSolve}\left(\begin{cases} 2 \cdot x + 4 \cdot y = 3 \\ 5 \cdot x - 3 \cdot y = 7 \end{cases}, \{x, y\}\right) \quad \left\{ \frac{37}{26}, \frac{1}{26} \right\}$$

linSolve(*EquaçãoLinear1* and *EquaçãoLinear2* e ..., *Var1*, *Var2*, ...) ⇒ lista

$$\text{linSolve}\left(\begin{cases} 2 \cdot x = 3 \\ 5 \cdot x - 3 \cdot y = 7 \end{cases}, \{x, y\}\right) \quad \left\{ \frac{3}{2}, \frac{1}{6} \right\}$$

linSolve({*EquaçãoLinear1*, *EquaçãoLinear2*, ...}, *Var1*, *Var2*, ...) ⇒ lista

$$\text{linSolve}\left(\begin{cases} \text{apple} + 4 \cdot \text{pear} = 23 \\ 5 \cdot \text{apple} - \text{pear} = 17 \end{cases}, \{\text{apple}, \text{pear}\}\right) \quad \left\{ \frac{13}{3}, \frac{14}{3} \right\}$$

linSolve(*SistemaDeEquaçõesLineares*, {*Var1*, *Var2*, ...}) ⇒ lista

$$\text{linSolve}\left(\begin{cases} \text{apple} \cdot 4 + \frac{\text{pear}}{3} = 14 \\ -\text{apple} + \text{pear} = 6 \end{cases}, \{\text{apple}, \text{pear}\}\right) \quad \left\{ \frac{36}{13}, \frac{114}{13} \right\}$$

linSolve(*EquaçãoLinear1* and *EquaçãoLinear2* e ..., {*Var1*, *Var2*, ...}) ⇒ lista

linSolve({*EquaçãoLinear1*, *EquaçãoLinear2*, ...}, {*Var1*, *Var2*, ...}) ⇒ lista

Devolve uma lista de soluções para as variáveis *Var1*, *Var2*, ...

O primeiro argumento tem de avaliar um sistema de equações do 1º grau ou uma equação individual do 1º grau. Caso contrário, ocorre um erro de argumento.

Por exemplo, a avaliação de **linSolve** (**x=1 and x=2,x**) produz um resultado de “Erro de argumento”.

ΔList()

ΔList(*Lista1*) ⇒ *lista*

$\Delta\text{List}(\{20,30,45,70\})$	$\{10,15,25\}$
--------------------------------------	----------------

Nota: Pode introduzir esta função através da escrita de **deltaList** (...) no teclado.

Devolve uma lista com as diferenças entre os elementos consecutivos em *Lista1*. Cada elemento de *Lista1* é subtraído do elemento seguinte de *Lista1*. A lista resultante é sempre um elemento mais pequeno que a *Lista1* original.

list►mat()

list►mat(*Lista* [, *elementosPorLinha*]) ⇒ *matriz*

$\text{list}\blacktriangleright\text{mat}(\{1,2,3\})$	$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$
$\text{list}\blacktriangleright\text{mat}(\{1,2,3,4,5\},2)$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 0 \end{bmatrix}$

Devolve uma matriz preenchida linha por linha com os elementos da *Lista*.

elementosPorLinha, se incluído, especifica o número de elementos por linha. A predefinição é o número de elementos em *Lista* (uma linha).

Se a *Lista* não preencher a matriz resultante, são adicionados zeros.

Nota: Pode introduzir esta função através da escrita de **list@>mat** (...) no teclado do computador.

Expr ►ln $\Rightarrow$ expressão

$$\left(\log_{10} (x) \right) \text{►ln} \quad \frac{\ln(x)}{\ln(10)}$$

Faz com que a entrada *Expr* seja convertida para uma expressão apenas com logaritmos naturais (ln).

Nota: Pode introduzir este operador através da escrita de @>1n no teclado do computador.

ln()

Teclas ctrl e^x

ln(*Expr1*) $\Rightarrow$ expressão

$$\ln(2.) \quad 0.693147$$

ln(*Lista1*) $\Rightarrow$ lista

Devolve o logaritmo natural do argumento.

Se o modo do formato complexo for Real:

Para uma lista, devolve os logaritmos naturais dos elementos.

$$\ln\{-3, 1.2, 5\}$$

"Error: Non-real calculation"

Se o modo do formato complexo for Rectangular:

$$\ln\{-3, 1.2, 5\} \quad \{\ln(3) + \pi \cdot i, 0.182322, \ln(5)\}$$

ln(*MatrizQuadrada1*)
 $\Rightarrow$ *MatrizQuadrada*

No modo de ângulo Radianos e Formato complexo rectangular:

Devolve o logaritmo natural da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o logaritmo natural de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()** em.

$$\ln \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1.83145 + 1.73485 \cdot i & 0.009193 - 1.49086 \\ 0.448761 - 0.725533 \cdot i & 1.06491 + 0.623491 \cdot i \\ -0.266891 - 2.08316 \cdot i & 1.12436 + 1.79018 \cdot i \end{bmatrix}$$

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

Para ver o resultado completo, prima $\blacktriangle$ e, de seguida, utilize $\blacktriangleleft$ e $\blacktriangleright$ para mover o cursor.

LnReg

Catálogo > 

LnReg *X*, *Y* [, [Freq] [, *Categoria*, *Incluir*]]

Calcula a regressão logarítmica $y = a + b \cdot \ln(x)$ a partir das listas X e Y com a frequência $Freq$. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

X e Y são listas de variáveis independentes e dependentes.

$Freq$ é uma lista opcional de valores de frequência. Cada elemento em $Freq$ especifica a frequência de ocorrência para cada ponto de dados X e Y correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

$Categoria$ é uma lista de códigos de categorias para os dados X e Y correspondentes.

$Incluir$ é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $a + b \cdot \ln(x)$
stat.a, stat.b	Parâmetros de regressão
stat.r ²	Coefficiente de determinação linear para dados transformados
stat.r	Coefficiente de correlação para dados transformados ($\ln(x)$, y)
stat.Resid	Resíduos associados ao modelo logarítmico
stat.ResidTrans	Resíduos associados ao ajuste linear dos dados transformados
stat.XReg	Lista de dados na <i>Lista X</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.YReg	Lista de dados na <i>Lista Y</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.FreqReg	Lista de frequências correspondentes a <i>stat.XReg</i> e <i>stat.YReg</i>

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

Declara as *vars* especificadas como variáveis locais. Essas variáveis só existem durante a avaliação de uma função e são eliminadas quando a função terminar a execução.

Nota: As variáveis locais poupam memória porque só existem temporariamente. Também não perturbam nenhum valor da variável global existente. As variáveis locais têm de ser utilizadas para ciclos **For** e guardar temporariamente os valores numa função multilinhas visto que as modificações nas variáveis globais não são permitidas numa função.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

```
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

Lock*Var1* [, *Var2*] [, *Var3*] ...**Lock***Var*.

Bloqueia as variáveis ou o grupo de variáveis especificadas. Não pode eliminar ou modificar as variáveis bloqueadas.

Não pode bloquear ou desbloquear a variável do sistema *Ans*, e não pode bloquear os grupos de variáveis do sistema *stat*. ou *tvm*.

Nota: O comando **Bloquear (Lock)** apaga o histórico de Anular/Repetir quando aplicado a variáveis desbloqueadas.

Consulte **unLock**, página 216, e **getLockInfo()**, página 91.

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

log()

Teclas  

log (*Expr1* [, *Expr2*]) $\Rightarrow$ expressão

log (*Lista1* [, *Expr2*]) $\Rightarrow$ lista

Devolve o logaritmo -*Expr2* base do primeiro argumento.

Nota: Consulte também **Modelo do logaritmo**, página 2.

Para uma lista, devolve o logaritmo -*Expr2* base dos elementos.

Se omitir o segundo argumento, 10 é utilizado como a base.

log (*MatrizQuadrada1* [, *Expr*]) $\Rightarrow$ *MatrizQuadrada*

Devolve o logaritmo *Expr* base da matriz de *MatrizQuadrada1*. Isto não é mesmo que calcular o logaritmo *Expr* base de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos** ().

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

Se omitir o argumento base, 10 é utilizado como a base.

$$\log_{10} (2.) \quad 0.30103$$

$$\log_4 (2.) \quad 0.5$$

$$\log_3 (10) - \log_3 (5) \quad \log_3 (2)$$

Se o modo do formato complexo for Real:

$$\log_{10} (\{-3, 1.2, 5\}) \quad \text{Error: Non-real result}$$

Se o modo do formato complexo for Rectangular:

$$\log_{10} (\{-3, 1.2, 5\}) \\ \left\{ \log_{10} (3) + 1.36438 \cdot i, 0.079181, \log_{10} (5) \right\}$$

No modo de ângulo Radianos e Formato complexo rectangular:

$$\log_{10} \left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix} \right) \\ \begin{bmatrix} 0.795387 + 0.753438 \cdot i & 0.003993 - 0.6474 \cdot i \\ 0.194895 - 0.315095 \cdot i & 0.462485 + 0.2707 \cdot i \\ -0.115909 - 0.904706 \cdot i & 0.488304 + 0.7774 \cdot i \end{bmatrix}$$

Para ver o resultado completo, prima $\blacktriangle$ e, de seguida, utilize $\blacktriangleleft$ e $\blacktriangleright$ para mover o cursor.

►logbase

Catálogo > 

Expr ►**logbase**(*Expr1*) $\Rightarrow$ expressão

Faz com que a entrada Expressão seja simplificada para uma expressão com a base *Expr1*.

$$\log_3 (10) - \log_5 (5) \blacktriangleright \logbase(5) \quad \frac{\log_5 \left(\frac{10}{3} \right)}{\log_5 (3)}$$

Nota: Pode introduzir este operador através da escrita de @>Logbase (...) no teclado do computador.

Logistic

Logistic *X*, *Y*, [*Freq*] [, *Categoria*, *Incluir*]

Calcula a regressão logística $y = (c/(1+a \cdot e^{-bx}))$ a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

X e *Y* são listas de variáveis independentes e dependentes.

Freq é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

Categoria é uma lista de códigos de categorias para os dados *X* e *Y* correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $c/(1+a \cdot e^{-bx})$
stat.a, stat.b, stat.c	Parâmetros de regressão
stat.Resid	Resíduos da regressão

Variável de saída	Descrição
stat.XReg	Lista de dados na <i>Lista X</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.YReg	Lista de dados na <i>Lista Y</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.FreqReg	Lista de frequências correspondentes a <i>stat.XReg</i> e <i>stat.YReg</i>

LogisticD

Catálogo > 

LogisticD *X*, *Y* [, [*Repetições*], [*Freq*] [, *Categoria*, *Incluir*]]

Calcula a regressão logística $y = (c/(1+a \cdot e^{-bx})+d)$ a partir das listas *X* e *Y* com a frequência *Freq*, utilizando um número especificado de *repetições*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

X e *Y* são listas de variáveis independentes e dependentes.

Iterações é um valor opcional que especifica o número máximo de vezes que uma solução será tentada. Se for omitido, 64 é utilizado. Em geral, valores maiores resultam numa melhor precisão, mas maiores tempos de execução, e vice-versa.

Freq é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

Categoria é uma lista de códigos de categorias para os dados *X* e *Y* correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $c/(1+a \cdot e^{-bx})+d$
stat.a, stat.b, stat.c, stat.d	Parâmetros de regressão
stat.Resid	Resíduos da regressão
stat.XReg	Lista de dados na <i>Lista X</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.YReg	Lista de dados na <i>Lista Y</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.FreqReg	Lista de frequências correspondentes a <i>stat.XReg</i> e <i>stat.YReg</i>

Loop

Ciclo
Bloco
EndLoop

Executa repetidamente as declarações em *Bloco*. Não se esqueça de que o ciclo será executado continuamente, excepto se executar a instrução **Ir para** ou **Sair** no *Bloco*.

Bloco é uma sequência de declarações separadas pelo carácter “:”.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

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	
	Done
<i>rollcount()</i>	16
<i>rollcount()</i>	3

LU *Matriz, MatrizI, Matrizu, Matrizp* [,Tol]

Calcula a decomposição LU (inferior-superior) Doolittle LU de uma matriz complexa ou real. A matriz triangular inferior é guardada em *MatrizI*, a matriz triangular superior em *Matrizu* e a matriz de permutações (que descreve as trocas de linhas durante o cálculo) em *Matrizp*.

$$\text{MatrizI} \cdot \text{Matrizu} = \text{Matrizp} \cdot \text{matriz}$$

Opcionalmente, qualquer elemento da matriz é tratado como zero se o valor absoluto for inferior a *Tol*. Esta tolerância só é utilizada se a matriz tiver entradas de ponto flutuante e não contiver variáveis simbólicas sem um valor atribuído. Caso contrário, *Tol* é ignorado.

- Se utilizar ou definir o modo **Auto ou Aproximado** para Aproximado, os cálculos são efectuados com a aritmética do ponto flutuante.
- Se *Tol* for omitido ou não utilizado, a tolerância predefinida for calculada como:
 $5E-14 \cdot \max(\dim(\text{Matriz})) \cdot \text{rowNorm}(\text{Matriz})$

O algoritmo de factorização **LU** utiliza a articulação parcial com as trocas de linhas.

$$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix} \rightarrow mI \quad \begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix}$$

LU mI,lower,upper,perm Done

$$\text{lower} \quad \begin{bmatrix} 1 & 0 & 0 \\ \frac{5}{6} & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}$$

$$\text{upper} \quad \begin{bmatrix} 6 & 12 & 18 \\ 0 & 4 & 16 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{perm} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} m & n \\ o & p \end{bmatrix} \rightarrow mI \quad \begin{bmatrix} m & n \\ o & p \end{bmatrix}$$

LU mI,lower,upper,perm Done

$$\text{lower} \quad \begin{bmatrix} 1 & 0 \\ \frac{m}{o} & 1 \\ o & \end{bmatrix}$$

$$\text{upper} \quad \begin{bmatrix} o & p \\ 0 & n - \frac{m \cdot p}{o} \\ o & \end{bmatrix}$$

$$\text{perm} \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

M

mat▶list()

mat▶list t(*Matriz*) ⇒*lista*

Devolve uma lista preenchida com os elementos em *Matriz*. Os elementos são copiados de *Matriz* linha por linha.

$$\text{mat▶list}([1 \ 2 \ 3]) \quad \{1,2,3\}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow mI \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$\text{mat▶list}(mI) \quad \{1,2,3,4,5,6\}$$

Nota: Pode introduzir esta função através da escrita de **mat@>list(...)** no teclado do computador.

max(*Expr1*, *Expr2*) $\Rightarrow$ expressão $\max\{2.3, 1.4\}$ 2.3**max**(*Lista1*, *Lista2*) $\Rightarrow$ lista $\max\{\{1, 2\}, \{-4, 3\}\}$ $\{1, 3\}$ **max**(*Matriz1*, *Matriz2*) $\Rightarrow$ matriz

Devolve o máximo dos dois argumentos. Se os argumentos forem duas listas ou matrizes, devolve uma lista ou matriz com o valor máximo de cada par dos elementos correspondentes.

max(*Lista*) $\Rightarrow$ expressão $\max\{\{0, 1, -7, 1.3, 0.5\}\}$ 1.3Devolve o elemento máximo em *lista*.**max**(*Matriz1*) $\Rightarrow$ matriz $\max\left(\begin{bmatrix} 1 & -3 & 7 \\ -4 & 0 & 0.3 \end{bmatrix}\right)$ $\begin{bmatrix} 1 & 0 & 7 \end{bmatrix}$

Devolve um vector da linha com o elemento máximo de cada coluna em *Matriz1*.

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

Nota: Consulte também **fMax()** e **min()**.**mean**(*Lista* [, *freList*]) $\Rightarrow$ expressão $\text{mean}(\{0.2, 0.1, -0.3, 0.4\})$ 0.26

Devolve a média dos elementos em *Lista*.

 $\text{mean}(\{1, 2, 3\}, \{3, 2, 1\})$ $\frac{5}{3}$

Cada elemento de *ListaFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

mean(*Matriz1* [, *MatrizFreq*]) $\Rightarrow$ matriz

No Formato de vector rectangular:

Devolve um vector da linha da média de todas as colunas em *Matriz1*.

Cada elemento de *MatrizFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Matriz1*.

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

mean	$\begin{pmatrix} 0.2 & 0 \\ -1 & 3 \\ 0.4 & -0.5 \end{pmatrix}$	$\begin{bmatrix} -0.133333 & 0.833333 \end{bmatrix}$
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()

median(Lista[, ListaFreq]) ⇒ expressão

median	$\{ \{ 0.2, 0.1, -0.3, 0.4 \} \}$	0.2
--------	-----------------------------------	-----

Devolve a mediana dos elementos em *Lista*.

Cada elemento de *ListaFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

median(Matriz1[, MatrizFreq]) ⇒ matriz

Devolve um vector em linha com as medianas das colunas da *Matriz1*.

median	$\begin{pmatrix} 0.2 & 0 \\ 1 & -0.3 \\ 0.4 & -0.5 \end{pmatrix}$	$\begin{bmatrix} 0.4 & -0.3 \end{bmatrix}$
--------	---	--

Cada elemento de *MatrizFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Matriz1*.

Notas:

- Todas as entradas da lista ou matriz têm de ser simplificadas para números.
- Os elementos (nulos) vazios da lista ou matriz são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

MedMed

MedMed X,Y[, Freq] [, Categoria, Incluir]

Calcula a recta média-médiai = $(m \cdot x + b)$ a partir das listas X e Y com a frequência $Freq$. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

X e Y são listas de variáveis independentes e dependentes.

$Freq$ é uma lista opcional de valores de frequência. Cada elemento em $Freq$ especifica a frequência de ocorrência para cada ponto de dados X e Y correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

$Categoria$ é uma lista de códigos de categorias para os dados X e Y correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação da recta mediana-mediana: $m \cdot x + b$
stat.m, stat.b	Parâmetros do modelo
stat.Resid	Resíduos da recta mediana-mediana
stat.XReg	Lista de dados na <i>Lista X</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.YReg	Lista de dados na <i>Lista Y</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.FreqReg	Lista de frequências correspondentes a <i>stat.XReg</i> e <i>stat.YReg</i>

mid()Catálogo > 

mid(CadeiaDeOrigem, Início [, Contagem]) ⇒ cadeia

Devolve os caracteres *Contagem* a partir da cadeia de caracteres *CadeiaDeOrigem*, começando pelo número de caracteres *Início*.

Se *Contagem* for omitida ou maior que a dimensão de *CadeiaDeOrigem*, devolve todos os caracteres de *CadeiaDeOrigem*, começando pelo número de caracteres *Início*.

Contagem tem de ser ≥ 0 . Se *Contagem* = 0, devolve uma cadeia vazia.

mid(ListaDeOrigem, Início [, Contagem]) ⇒ lista

Devolve os elementos *Contagem* de *ListaDeOrigem*, começando pelo número de elementos *Início*.

Se *Contagem* for omitida ou maior que a dimensão de *ListaDeOrigem*, devolve todos os elementos de *ListaDeOrigem*, começando pelo número de elementos *Início*.

Contagem tem de ser ≥ 0 . Se *Contagem* = 0, devolve uma lista vazia.

mid(ListaDaCadeiaDeOrigem, Início [, Contagem]) ⇒ lista

Devolve as cadeias *Contagem* da lista de cadeias *ListaDaCadeiaDeOrigem*, começando pelo número de elementos *Início*.

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"}
----------------------------	-----------

min()Catálogo > 

min(Expr1, Expr2) ⇒ expressão

min(Lista1, Lista2) ⇒ lista

min(Matriz1, Matriz2) ⇒ matriz

min(2,3,1,4)	1.4
min({1,2},{-4,3})	{-4,2}

Devolve o mínimo dos dois argumentos.
Se os argumentos forem duas listas ou matrizes, devolve uma lista ou matriz com o valor mínimo de cada par dos elementos correspondentes.

min(Lista) $\Rightarrow$ expressão

Devolve o elemento mínimo de *Listas*.

min(Matriz1) $\Rightarrow$ matriz

Devolve um vector da linha com o elemento mínimo de cada coluna em *Matriz1*.

Nota: Consulte também **fMin()** e **max()**.

$$\min(\{0,1,-7,1.3,0.5\})$$

-7

$$\min\left(\begin{bmatrix} 1 & -3 & 7 \\ -4 & 0 & 0.3 \end{bmatrix}\right)$$

$\begin{bmatrix} -4 & -3 & 0.3 \end{bmatrix}$

mirr()

mirr(TaxaDeFinanciamento, TaxaDeReinvestimento, CF0, ListaCF [, FreqCF])

Função financeira que devolve a taxa de retorno interna modificada de um investimento.

TaxaDeFinanciamento é a taxa de juro que é paga sobre os montantes de cash flow.

TaxaDeReinvestimento é a taxa de juro em que os cash flows são reinvestidos.

CF0 é o cash flow inicial no momento 0; tem de ser um número real.

ListaCF é uma lista de montantes de cash flow após o cash flow inicial *CF0*.

FreqCF é uma lista opcional em que cada elemento especifica a frequência da ocorrência para um montante de cash flow agrupado (consecutivo), que é o elemento correspondente de *ListaCF*. A predefinição é 1; se introduzir valores, têm de ser números inteiros positivos < 10,000.

Nota: Consulte também **irr()**, página 103.

$$\begin{aligned} list1 &:= \{6000, -8000, 2000, -3000\} \\ &\qquad \qquad \qquad \{6000, -8000, 2000, -3000\} \\ list2 &:= \{2, 2, 2, 1\} \\ &\qquad \qquad \qquad \{2, 2, 2, 1\} \\ mirr(4.65, 12, 5000, list1, list2) &\quad 13.41608607 \end{aligned}$$

mod()Catálogo > **mod**(*Expr1*, *Expr2*) $\Rightarrow$ expressão**mod**(*Lista1*, *Lista2*) $\Rightarrow$ lista**mod**(*Matriz1*, *Matriz2*) $\Rightarrow$ matriz

Devolve o primeiro módulo de argumentos do segundo argumento conforme definido pelas identidades:

$$\text{mod}(x, 0) = x$$

$$\text{mod}(x, y) = x - y \text{ floor}(x/y)$$

Quando o segundo argumento for diferente de zero, o resultado é periódico nesse argumento. O resultado é zero ou tem o mesmo sinal do segundo argumento.

Se os argumentos forem duas listas ou matrizes, devolve uma lista ou matriz com o módulo de cada par de elementos correspondentes.

Nota: Consulte também **remain()**, página 163

$\text{mod}(7, 0)$	7
$\text{mod}(7, 3)$	1
$\text{mod}(-7, 3)$	2
$\text{mod}(7, -3)$	-2
$\text{mod}(-7, -3)$	-1
$\text{mod}(\{12, -14, 16\}, \{9, 7, -5\})$	$\{3, 0, -4\}$

mRow()Catálogo > **mRow**(*Expr*, *Matriz1*, *Índice*) $\Rightarrow$ matriz

Devolve uma cópia de *Matriz1* com cada elemento na linha *Índice* de *Matriz1* multiplicado por *Expr*.

$$\text{mRow}\left(\frac{-1}{3}, \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, 2\right) = \begin{bmatrix} 1 & 2 \\ -1 & -4 \\ 3 & 3 \end{bmatrix}$$

mRowAdd()Catálogo > **mRowAdd**(*Expr*, *Matriz1*, *Índice1*, *Índice2*) $\Rightarrow$ matriz

Devolve uma cópia de *Matriz1* com cada elemento na linha *Índice2* de *Matriz1* substituído por:

$$\begin{aligned} \text{mRowAdd}\left(-3, \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, 1, 2\right) &= \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix} \\ \text{mRowAdd}\left(n, \begin{bmatrix} a & b \\ c & d \end{bmatrix}, 1, 2\right) &= \begin{bmatrix} a & b \\ a \cdot n + c & b \cdot n + d \end{bmatrix} \end{aligned}$$

$$\text{Expr} \cdot \text{linha } \text{Índice1} + \text{linha } \text{Índice2}$$

MultReg $Y, X1[,X2[,X3,...[,X10]]]$

Calcula a regressão linear múltipla da lista Y nas listas $X1, X2, ..., X10$. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter dimensões iguais.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $b_0+b_1 \cdot x_1+b_2 \cdot x_2+ \dots$
stat.b0, stat.b1, ...	Parâmetros de regressão
stat.R ²	Coefficiente de determinação múltipla
stat.ŷLista	$\hat{y}_{\text{Lista}} = b_0+b_1 \cdot x_1+ \dots$
stat.Resid	Resíduos da regressão

MultRegIntervals**MultRegIntervals** $Y, X1[,X2[,X3,...[,X10]]], ListaValX[,NívelC]$

Calcula um valor y previsto, um intervalo de previsão de nível C para uma observação, e um intervalo de confiança de nível C para a resposta média.

Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter dimensões iguais.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $b_0+b_1 \cdot x_1+b_2 \cdot x_2+ \dots$
stat.ŷ	Um ponto prevê: $\hat{y} = b_0 + b_1 \cdot x_1 + \dots$ para <i>ListaDeValoresX</i>
stat.dfError	Erro dos graus de liberdade

Variável de saída	Descrição
stat.CLower, stat.CUpper	Intervalo de confiança para uma resposta média
stat.ME	Margem de erro do intervalo de confiança
stat.SE	Erro padrão da resposta média
stat.LowerPred, stat.UpperrPred	Intervalo de previsão para uma observação
stat.MEPred	Margem de erro do intervalo de previsão
stat.SEPred	Erro padrão para previsão
stat.bList	Lista de parâmetros de regressão, {b0,b1,b2,...}
stat.Resid	Residuais da regressão

MultRegTests

Catálogo > 

MultRegTests $Y, X1[,X2[,X3,...[,XI0]]]$

O teste de regressão linear calcula uma regressão linear múltipla a partir dos dados fornecidos e fornece a estatística do teste F global e estatística do teste t para os coeficientes.

Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Saídas

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $b_0+b_1 \cdot x_1+b_2 \cdot x_2+ \dots$
stat.F	Estatística do teste F global
stat.PVal	Valor P associado à estatística F global
stat.R ²	Coeficiente de determinação múltipla
stat.AdjR ²	Coeficiente ajustado de determinação múltipla
stat.s	Desvio padrão do erro

Variável de saída	Descrição
stat.DW	Estatística Durbin-Watson; utilizada para determinar se a correlação automática de primeira ordem está presente no modelo
stat.dfReg	Graus de liberdade da regressão
stat.SSReg	Soma de quadrados da regressão
stat.MSReg	Quadrado médio da regressão
stat.dfError	Erro dos graus de liberdade
stat.SSError	Erro da soma de quadrados
stat.MSError	Erro do quadrado médio
stat.bList	{b0,b1,...} Lista de parâmetros
stat.tList	Lista da estatística t, um para cada coeficiente na bList
stat.PList	Lista de valores P para cada estatística t
stat.SEList	Lista de erros padrão para coeficientes na bList
stat.ŷList	$\hat{y}List = b_0 + b_1 \cdot x_1 + \dots$
stat.Resid	Resíduos da regressão
stat.sResid	Resíduos normalizados; obtido através da divisão de um resíduo pelo desvio padrão
stat.CookDist	Distância de Cook; medição da influência de uma observação com base no residual e optimização
stat.Leverage	Medição da distância entre os valores independentes e os valores médios

N

nand

Teclas ctrl =

ExprBooleana1 nand ExprBooleana2

devolve expressão booleana

ListaBooleana1 nand ListaBooleana2

devolve lista booleana

MatrizBooleana1 nand MatrizBooleana2

devolve matriz booleana

$x \geq 3$ and $x \geq 4$	$x \geq 4$
$x \geq 3$ nand $x \geq 4$	$x < 4$

Devolve a negação de uma operação **and** lógica nos dois argumentos. Devolve falso, verdadeiro ou uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

NúmeroInteiro1 **nand** *NúmeroInteiro2*
 $\Rightarrow$ número inteiro

Compara dois números inteiros reais bit a bit com uma operação **nand**. Internamente, ambos os números inteiros são convertidos para números binários de 64 bits assinados. Quando os bits correspondentes forem comparados, o resultado é 0 se ambos os bits forem 1; caso contrário, o resultado é 1. O valor devolvido representa os resultados dos bits e aparece de acordo com o modo base.

Pode introduzir os números inteiros em qualquer base numérica. Para uma entrada binária ou hexadecimal, tem de utilizar o prefixo 0b ou 0h, respetivamente. Sem um prefixo, os números inteiros são tratados como decimais (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()

Catálogo > 

nCr(*Expr1*, *Expr2*) $\Rightarrow$ expressão

Para o número inteiro *Expr1* e *Expr2* com $Expr1 \geq Expr2 \geq 0$, **nCr()** é o número de combinações de coisas de *Expr1* retiradas de *Expr2* de uma vez. (Isto também é conhecido como um coeficiente binomial.) Ambos os argumentos pode ser números inteiros ou expressões simbólicas.

nCr (<i>z</i> ,3)	$\frac{z \cdot (z-2) \cdot (z-1)}{6}$
<i>Ans</i> <i>z</i> =5	10
nCr (<i>z</i> , <i>c</i>)	$\frac{z!}{c! \cdot (z-c)!}$
<i>Ans</i>	$\frac{1}{c!}$
nPr (<i>z</i> , <i>c</i>)	$\frac{1}{c!}$

nCr(*Expr*, 0) $\Rightarrow$ 1

nCr(*Expr*, *NúmeroInteiroNeg*) $\Rightarrow$ 0

nCr(*Expr*, *NúmeroInteiroPos*) $\Rightarrow Expr \cdot (Expr - 1) \dots$

$(Expr - \text{NúmeroInteiroPos} + 1) /$
 $\text{NúmeroInteiroPos}!$

$\text{nCr}(Expr, \text{NúmeroNãoInteiro})$
 $\Rightarrow \text{expressão !}$

$((Expr - \text{NúmeroNãoInteiro})! \cdot$
 $\text{NúmeroNãoInteiro} !)$

$\text{nCr}(\text{Lista1}, \text{Lista2}) \Rightarrow \text{lista}$

$\text{nCr}(\{5,4,3\}, \{2,4,2\}) \quad \{10,1,3\}$

Devolve uma lista de combinações com base nos pares de elementos correspondentes nas duas listas. Os argumentos têm de ter o mesmo tamanho de listas.

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

$\text{nCr}\left(\begin{bmatrix} 6 & 5 \\ 4 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}\right) \quad \begin{bmatrix} 15 & 10 \\ 6 & 3 \end{bmatrix}$

Devolve uma matriz de combinações com base nos pares de elementos correspondentes nas duas matrizes. Os argumentos têm de ter o mesmo tamanho de matrizes.

nDerivative()

$\text{nDerivative}(Expr1, Var = Valor$
 $[, Ordem]) \Rightarrow valor$

$\text{nDerivative}(|x|, x=1) \quad 1$

$\text{nDerivative}(Expr1, Var[, Ordem]) \mid$
 $Var = Valor \Rightarrow valor$

$\text{nDerivative}(|x|, x)|_{x=0} \quad \text{undef}$

$\text{nDerivative}(\sqrt{x-1}, x)|_{x=1} \quad \text{undef}$

Devolve a derivada numérica calculada com os métodos de diferenciação automáticos.

Ao especificar o *Valor*, substitui qualquer atribuição de variável anterior ou qualquer substituição atual "|" para a variável.

Ordem da derivada tem de ser 1 ou 2.

newList()

$\text{newLis t}(\text{ElementosNum}) \Rightarrow \text{lista}$

$\text{newList}(4) \quad \{0,0,0,0\}$

Devolve uma lista com uma dimensão de *ElementosNum*. Cada elemento é zero.

newMat()Catálogo > 

newMat(*LinhaNum*, *ColunasNum*)
 $\Rightarrow$ matriz

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

Devolve uma matriz de zeros com a dimensão *LinhaNum* por *ColunasNum*.

nfMax()Catálogo > 

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

Devolve um valor numérico candidato da variável *Var* em que ocorre o máximo local de *Expr*.

Se fornecer um *LimiteInferior* e um *LimiteSuperior*, a função procura o máximo local no intervalo fechado [*LimiteInferior*, *LimiteSuperior*].

Nota: Consulte também **fMax()** e **d()**.

nfMin()Catálogo > 

nfMin(*Expr*, *Var*) $\Rightarrow$ valor

$\text{nfMin}(x^2 + 2 \cdot x + 5, x)$	-1.
--	-----

nfMin(*Expr*, *Var*, *LimiteInferior*)
 $\Rightarrow$ valor

$\text{nfMin}(0.5 \cdot x^3 - x - 2, x, -5, 5)$	-5.
---	-----

nfMin(*Expr*, *Var*, *LimiteInferior*,
LimiteSuperior) $\Rightarrow$ valor

nfMin(*Expr*, *Var*) | *LimiteInferior* $\leq$ *Var*
 $\leq$ *LimiteSuperior* $\Rightarrow$ valor

Devolve um valor numérico candidato da variável *Var* em que ocorre o mínimo local de *Expr*.

Se fornecer um *LimiteInferior* e um *LimiteSuperior*, a função procura o mínimo local no intervalo fechado [*LimiteInferior*,*LimiteSuperior*].

Nota: Consulte também **fMin()** e **d()**.

nInt()

nInt(*Expr1*, *Var*, *Inferior*, *Superior*)
⇒ expressão

$$\text{nInt}(e^{-x^2}, x, -1, 1) \quad 1.49365$$

Se a expressão a integrar *Expr1* não contiver nenhuma variável para além de *Var* e se *Inferior* e *Superior* forem constantes, ∞ positivo ou ∞ negativo, **nInt()** devolve uma aproximação de $\int$ (*Expr1*, *Var*, *Inferior*, *Superior*). Esta aproximação é uma média ponderada de alguns valores de amostra da expressão a integrar no intervalo *Inferior* < *Var* < *Superior*.

O objectivo é obter seis dígitos significativos. O algoritmo adaptável termina quando parecer que o objectivo foi alcançado ou quando parecer improvável que as amostras adicionais produzam uma melhoria acentuada.

$$\text{nInt}(\cos(x), x, \pi, \pi + 1. \text{E}^{-12}) \quad -1.04144 \text{E}^{-12}$$

$$\int_{-\pi}^{\pi + 10^{-12}} \cos(x) dx \quad -\sin\left(\frac{1}{1000000000000}\right)$$

Aparece um aviso (“Precisão questionável”) quando parecer que o objectivo não foi alcançado.

Nest **nInt()** para fazer integração numérica múltipla. Os limites da integração podem depender das variáveis de integração fora dos limites.

$$\text{nInt}\left(\text{nInt}\left(\frac{e^{-x \cdot y}}{\sqrt{x^2 - y^2}}, y, -x, x\right), x, 0, 1\right) \quad 3.30423$$

Nota: Consulte também **j()**, página 242.

nom()

nom(*TaxaEfectiva*, *CpY*) ⇒ valor

$$\text{nom}(5.90398, 12) \quad 5.75$$

Função financeira que converte a taxa de juro efectiva anual *TaxaEfectiva* para uma taxa nominal, dando *CpY* como o número de períodos compostos por ano.

TaxaEfectiva tem de ser um número real e *CpY* tem de ser um número real > 0.

Nota: Consulte também **eff()**, página 63.

norTeclas  

ExprBooleana1 **nor** *ExprBooleana2*
devolve expressão booleana

$x \geq 3$ or $x \geq 4$	$x \geq 3$
--------------------------	------------

ListaBooleana1 **nor** *ListaBooleana2*
devolve lista booleana

$x \geq 3$ nor $x \geq 4$	$x < 3$
---------------------------	---------

MatrizBooleana1 **nor** *MatrizBooleana2*
devolve matriz booleana

Devolve a negação de uma operação **or** lógica nos dois argumentos. Devolve falso, verdadeiro ou uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

NúmeroInteiro1 **nor** *NúmeroInteiro2* $\Rightarrow$ número inteiro

3 or 4	7
--------	---

3 nor 4	-8
---------	----

Compara dois números inteiros reais bit a bit com uma operação **nor**.

Internamente, ambos os números inteiros são convertidos para números binários de 64 bits assinados. Quando os bits correspondentes forem comparados, o resultado é 1 se ambos os bits forem 1; caso contrário, o resultado é 0. O valor devolvido representa os resultados dos bits e aparece de acordo com o modo base.

$\{1, 2, 3\}$ or $\{3, 2, 1\}$	$\{3, 2, 3\}$
--------------------------------	---------------

$\{1, 2, 3\}$ nor $\{3, 2, 1\}$	$\{-4, -3, -4\}$
---------------------------------	------------------

Pode introduzir os números inteiros em qualquer base numérica. Para uma entrada binária ou hexadecimal, tem de utilizar o prefixo 0b ou 0h, respetivamente. Sem um prefixo, os números inteiros são tratados como decimais (base 10).

norm()Catálogo > **norm**(*Matriz*) $\Rightarrow$ *expressão***norm**(*Vector*) $\Rightarrow$ *expressão*

Apresenta a norma Frobenius.

$\text{norm}\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right)$	$\sqrt{a^2+b^2+c^2+d^2}$
$\text{norm}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$	$\sqrt{30}$
$\text{norm}\left(\begin{bmatrix} 1 & 2 \end{bmatrix}\right)$	$\sqrt{5}$
$\text{norm}\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$	$\sqrt{5}$

normalLine()Catálogo > **normalLine***(Expr1,Var,Ponto)* $\Rightarrow$ *expressão***normalLine***(Expr1,Var=Ponto)* $\Rightarrow$ *expressão*

Apresenta a recta normal à curva representada por *Expr1* no ponto especificado na *Var=Ponto*.

$\text{normalLine}(x^2,x,1)$	$\frac{3}{2} \frac{x}{2}$
$\text{normalLine}((x-3)^2-4,x,3)$	$x=3$
$\text{normalLine}\left(\frac{1}{x^3},x=0\right)$	0
$\text{normalLine}(\sqrt{ x },x=0)$	undef

Certifique-se de que a variável independente não está definida. Por exemplo, se $f1(x):=5$ e $x:=3$, então **normalLine**($f1(x),x,2$) apresenta “falso.”

normCdf()Catálogo > 

normCdf(*LimiteInferior*,*LimiteSuperior*[, μ [σ]]) $\Rightarrow$ *número* se *LimiteInferior* e *LimiteSuperior* forem números, *lista* se *LimiteInferior* e *LimiteSuperior* forem listas

Calcula a probabilidade de distribuição normal entre *LimiteInferior* e *LimiteSuperior* para os μ (predefinição=0) e σ (predefinição=1) especificados.

Para $P(X \leq \text{LimiteSuperior})$, defina *LimiteInferior* = $-\infty$.

normPdf()Catálogo > 

normPdf(*ValX* [μ [σ]]) $\Rightarrow$ *número* se *ValX* for um número, *lista* se *ValX* for uma lista

Calcula a função de densidade de probabilidade para a distribuição normal num valor *ValX* especificado para μ e σ especificados.

not

Catálogo >

no t *ExprBooleana* $\Rightarrow$ *Expressão booleana*

Devolve falso, verdadeiro ou uma forma simplificada do argumento.

não *NúmeroInteiro1* $\Rightarrow$ *número inteiro*

Devolve um complemento de um número inteiro real. Internalmente, *NúmeroInteiro1* é convertido para um número de binário de 64 bits. O valor de cada bit é mudado (0 torna-se 1 e vice-versa) para um complemento. Os resultados aparecem de acordo com o modo base.

Pode introduzir o número em qualquer base numérica. Para uma entrada binária ou hexadecimal, tem de utilizar o prefixo 0b ou 0h, respectivamente. Sem um prefixo, o número inteiro é tratado como decimal (base 10).

Se introduzir um número inteiro na base 10 muito grande para uma forma binária de 64 bits assinada, é utilizada uma operação de módulo simétrico para colocar o valor no intervalo adequado. Para mais informações, consulte **►Base2**, página 19.

not(2≥3)	true
not(x<2)	x≥2
not not innocent	innocent

No modo base Hex:

Importante: Zero, não a letra O.

not 0h7AC36	0hFFFFFFFFF853C9
-------------	------------------

No modo base Bin:

0b100101►Base10	37
not 0b100101	
0b11111111111111111111111111111111►	
not 0b100101►Base10	-38

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **►** para mover o cursor.

Nota: Uma entrada binária pode ter até 64 dígitos (não contando com o prefixo 0b). Uma entrada hexadecimal pode ter até 16 dígitos.

nPr()

Catálogo >

nPr(*Expr1*, *Expr2*) $\Rightarrow$ expressão

Para o número inteiro *Expr1* e *Expr2* com *Expr1* $\geq$ *Expr2* $\geq$ 0, **nPr()** é o número de permutações de coisas de *Expr1* retiradas de *Expr2* de uma vez. Ambos os argumentos podem ser números inteiros ou expressões simbólicas.

nPr (<i>z</i> , 3)	$z \cdot (z-2) \cdot (z-1)$
Ans <i>z</i> =5	60
nPr (<i>z</i> , -3)	$\frac{1}{(z+1) \cdot (z+2) \cdot (z+3)}$
nPr (<i>z</i> , <i>c</i>)	$\frac{z!}{(z-c)!}$
Ans · nPr (<i>z</i> - <i>c</i> , - <i>c</i>)	1

nPr(*Expr*, 0) $\Rightarrow$ 1

nPr(*Expr*, *NúmeroInteiroNeg*) $\Rightarrow$ 1/
((*Expr* + 1) · (*Expr* + 2) ... (expressão
-*NúmeroInteiroNeg*))

nPr(*Expr*, *NúmeroInteiroPos*)

$\Rightarrow$ *Expr* · (*Expr* - 1) ... (*Expr*
- *NúmeroInteiroPos* + 1)

nPr(*Expr*, *NúmeroNãoInteiro*) $\Rightarrow$ *Expr*! / (*Expr* - *NúmeroNãoInteiro*)!**nPr**(*Lista1*, *Lista2*) $\Rightarrow$ lista

Devolve uma lista de permutações com base nos pares de elementos correspondentes nas duas listas. Os argumentos têm de ter o mesmo tamanho de listas.

nPr ($\{5, 4, 3\}, \{2, 4, 2\}$)	$\{20, 24, 6\}$
---	-----------------

nPr(*Matriz1*, *Matriz2*) $\Rightarrow$ matriz

Devolve uma matriz de permutações com base nos pares de elementos correspondentes nas duas matrizes. Os argumentos têm de ter a a mesma matriz de tamanhos.

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()

Catálogo >

npv(*TaxaDeJuro*, *CFO*, *ListaCF* [, *FreqCF*])

Função financeira que calcula o valor líquido actual; a soma dos valores actuais de entradas e saídas do cash flow. Um resultado positivo para npv indica um investimento lucrativo.

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

TaxaDeJuro é a taxa a descontar dos cash flows (o custo do dinheiro) durante um período.

CF0 é o cash flow inicial no momento 0; tem de ser um número real.

ListaCF é uma lista de montantes de cash flow após o cash flow inicial *CF0*.

FreqCF é uma lista em que cada elemento especifica a frequência da ocorrência para um montante de cash flow agrupado (consecutivo), que é o elemento correspondente de *ListaCF*. A predefinição é 1; se introduzir valores, têm de ser números inteiros positivos < 10,000.

nSolve()

nSolve(Equação, Var [= Tentativa])
⇒ número ou erro da cadeia

nSolve(Equação, Var [= Tentativa], LimiteInferior) ⇒ número ou erro da cadeia

nSolve(Equação, Var [= Tentativa], LimiteInferior, LimiteSuperior)
⇒ número ou erro da cadeia

nSolve(Equação, Var [= Tentativa] | LimiteInferior ≤ Var ≤ LimiteSuperior)
⇒ número ou erro da cadeia

Procura iterativamente uma solução numérica real aproximada para *Equação* para uma variável. Especifique a variável como:

variável

– ou –

variável = número real

Por exemplo, *x* é válido e logo é *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: Se existirem várias soluções, pode utilizar uma tentativa para ajudar a encontrar uma solução particular.

nSolve() é frequentemente mais rápido que **solve()** ou **zeros()**, em especial, se o operador “|” for utilizado para restringir a procura a um pequeno intervalo exactamente com uma solução simples.

nSolve() tenta determinar se um ponto em que o residual é zero ou dois pontos relativamente próximos em que o residual tem sinais opostos e a magnitude do residual não é excessiva. Se não conseguir atingir isto com um número modesto de pontos de amostra, devolve a cadeia “nenhuma solução encontrada.”

Nota: Consulte também **cSolve()**, **cZeros()**, **solve()** e **zeros()**.

$\text{nSolve}(x^2 + 5 \cdot x - 25 = 9, x) x < 0$	-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

OneVar [1,] *X* [, [*Freq*][, *Categoria*, *Incluir*]]

OneVar [*n*,] *X1*, *X2* [, *X3* [, ...[, *X20*]]]

Calcula a estatística de 1 variável até 20 listas. Um resumo dos resultados é guardado na variável *stat.results* (página 193.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

Os *argumentos X* são listas de dados.

Freq é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada valor *X* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

Categoria é uma lista de códigos de categorias numéricos para os valores *X* correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Um elemento (nulo) vazio em qualquer das listas *X*, *Freq* ou *Category* resulta num nulo para o elemento correspondente de todas essas listas. Um elemento vazio em qualquer uma das listas de *X1* a *X20* resulta num vazio para o elemento correspondente de todas essas listas. Para mais informações sobre os elementos vazios, consulte página 274.

Variável de saída	Descrição
stat. $\bar{x}$	Média dos valores x
stat. Σx	Soma dos valores x
stat. Σx^2	Soma dos valores x^2
stat.sx	Desvio padrão da amostra de x
stat. x	Desvio padrão da população de x
stat.n	Número de pontos de dados
stat.MinX	Mínimo dos valores x
stat.Q ₁ X	1º quartil de x
stat.MedianX	Mediana de x
stat.Q ₃ X	3º quartil de x
stat.MaxX	Máximo de valores x
stat.SSX	Soma de quadrados de desvios da média de x

or (ou)

ExprBooleana1 **or** *ExprBooleana2*
devolve expressão booleana

$$x \geq 3 \text{ or } x \geq 4$$

$$x \geq 3$$

ListaBooleana1 **or** *ListaBooleana2*
devolve lista booleana

MatrizBooleana1 **or** *MatrizBooleana2*
devolve matriz booleana

Devolve falso, verdadeiro ou uma forma simplificada da entrada original.

Devolve verdadeiro se uma ou ambas as expressões forem simplificadas para verdadeiro. Devolve falso apenas se ambas as expressões forem avaliadas para falso.

Nota: Consulte **xor**.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

NúmeroInterior1 or *NúmeroInterior2*
⇒ *número inteiro*

Compara dois números inteiros reais bit a bit com uma operação **or**. Internamente, ambos os números inteiros são convertidos para números binários de 64 bits assinados. Quando os bits correspondentes forem comparados, o resultado é 1 se ambos os bits forem 1; caso contrário, o resultado é 0. O valor devolvido representa os resultados dos bits e aparece de acordo com o modo base.

Pode introduzir os números inteiros em qualquer base numérica. Para uma entrada binária ou hexadecimal, tem de utilizar o prefixo 0b ou 0h, respectivamente. Sem um prefixo, os números inteiros são tratados como decimais (base 10).

Se introduzir um número inteiro na base 10 muito grande para uma forma binária de 64 bits assinada, é utilizada uma operação de módulo simétrico para colocar o valor no intervalo adequado. Para mais informações, consulte ►**Base2**, página 19.

Nota: Consulte **xor**.

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

No modo base Hex:

0h7AC36 or 0h3D5F	0h7BD7F
-------------------	---------

Importante: Zero, não a letra O.

No modo base Bin:

0b100101 or 0b100	0b100101
-------------------	----------

Nota: Uma entrada binária pode ter até 64 dígitos (não contando com o prefixo 0b). Uma entrada hexadecimal pode ter até 16 dígitos.

ord()Catálogo > **ord(Cadeia)** $\Rightarrow$ número inteiro**ord(Lista1)** $\Rightarrow$ lista

Devolve o código numérico do primeiro carácter na cadeia de caracteres *Cadeia* ou uma lista dos primeiros caracteres de cada elemento da lista.

<code>ord("hello")</code>	104
<code>char(104)</code>	"h"
<code>ord(char(24))</code>	24
<code>ord({"alpha","beta"})</code>	{ 97,98 }

P**P►Rx()**Catálogo > **P►Rx(rExpr, θExpr)** $\Rightarrow$ expressão**P►Rx(rList, θList)** $\Rightarrow$ lista**P►Rx(rMatrix, θMatrix)** $\Rightarrow$ matriz

Devolve a coordenada x equivalente do par (r, θ).

Nota: O argumento θ é interpretado como um ângulo expresso em graus, gradianos ou radianos de acordo com o modo de ângulo actual. Se o argumento for uma expressão, pode utilizar °, G ou ^r para substituir a definição do modo de ângulo temporariamente.

Nota: Pode introduzir esta função através da escrita de **P@►Rx (...)** no teclado do computador.

No modo de ângulo Radianos:

<code>P►Rx(r,θ)</code>	$\cos(\theta) \cdot r$
<code>P►Rx(4,60°)</code>	2
<code>P►Rx({-3,10,1.3},{\frac{\pi}{3},\frac{\pi}{4},0})</code>	$\left\{ \frac{-3}{2}, 5 \cdot \sqrt{2}, 1.3 \right\}$

P►Ry()Catálogo > **P►Ry(rExpr, θExpr)** $\Rightarrow$ expressão**P►Ry(rList, θList)** $\Rightarrow$ lista**P►Ry(rMatrix, θMatrix)** $\Rightarrow$ matriz

Devolve a coordenada y equivalente do par (r, θ).

No modo de ângulo Radianos:

<code>P►Ry(r,θ)</code>	$\sin(\theta) \cdot r$
<code>P►Ry(4,60°)</code>	$2 \cdot \sqrt{3}$
<code>P►Ry({-3,10,1.3},{\frac{\pi}{3},\frac{\pi}{4},0})</code>	$\left\{ \frac{-3 \cdot \sqrt{3}}{2}, -5 \cdot \sqrt{2}, 0 \right\}$

Nota: O argumento θ é interpretado como um ângulo expresso em graus, gradianos ou radianos de acordo com o modo de ângulo actual. Se o argumento for uma expressão, pode utilizar °, G ou r para substituir a definição do modo de ângulo temporariamente.

Nota: Pode introduzir esta função através da escrita de **P@>Ry (...)** no teclado do computador.

PassErr

PassErr

Para ver um exemplo de **PassErr**, consulte o exemplo 2 no comando **Try**, página 209.

Passa um erro para o nível seguinte.

Se a variável do sistema *errCode* for zero, **PassErr** não faz nada.

A proposição **Else** do bloco **Try...Else...EndTry** deve utilizar **ClrErr** ou **PassErr**. Se tiver de processar ou ignorar o erro, utilize **ClrErr**. Se não souber o que fazer com o erro, utilize **PassErr** para o enviar para a rotina de tratamento de erros seguinte. Se não existirem mais rotinas de tratamento de erros **Try...Else...EndTry** pendente, a caixa de diálogo de erros aparecerá como normal.

Nota: Consulte também **ClrErr**, página 28, e **Try**, página 209.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

piecewise()

piecewise(*Expr1* [, *Condição1* [, *Expr2* [, *Condição2* [, ...]]]])

Devolve as definições para uma função **piecewise** na forma de uma lista. Pode também criar definições **piecewise** com um modelo.

Define $p(x) = \begin{cases} x, & x > 0 \\ \text{undef}, & x \leq 0 \end{cases}$	<i>Done</i>
$p(1)$	1
$p(-1)$	undef

Nota: Consulte também **Modelo de Função por ramos**, página 3.

poissCdf()

poissCdf

$(\lambda, \text{LimiteInferior}, \text{LimiteSuperior}) \Rightarrow$ número se LimiteInferior e LimiteSuperior forem números, lista se LimiteInferior e LimiteSuperior forem listas

$\text{poissCdf}(\lambda, \text{LimiteSuperior})$ (para $P(0 \leq X \leq \text{LimiteSuperior}) \Rightarrow$ número se LimiteSuperior for um número, lista se LimiteSuperior for uma lista

Calcula uma probabilidade cumulativa para a distribuição Poisson discreta com a média especificada λ .

Para $P(X \leq \text{LimiteSuperior})$, defina $\text{LimiteInferior}=0$

poissPdf()

$\text{poissPdf}(\lambda, \text{ValX}) \Rightarrow$ número se ValX for um número, lista se ValX for uma lista

Calcula uma probabilidade para a distribuição Poisson discreta com a média especificada λ .

► Polar

Vector ► **Polar**

Nota: Pode introduzir este operador através da escrita de **@>Polar** no teclado do computador.

Apresenta o *vector* em forma polar $[r \angle \theta]$. O *vector* tem de ser de dimensão 2 e pode ser uma linha ou uma coluna.

$$\begin{array}{l} \begin{bmatrix} 1 & 3 \end{bmatrix} \text{► Polar} \quad \begin{bmatrix} 3.16228 & \angle 1.24905 \end{bmatrix} \\ \begin{bmatrix} x & y \end{bmatrix} \text{► Polar} \quad \left[\sqrt{x^2 + y^2} \quad \angle \frac{\pi \cdot \text{sign}(y)}{2} - \tan^{-1} \left(\frac{x}{y} \right) \right] \end{array}$$

Nota: ► **Polar** é uma instrução de formato de visualização, não uma função de conversão. Só pode utilizá-la no fim de uma linha de entrada e não actualiza *ans*.

Nota: Consulte também ► **Rect**, página 160.

ValorComplexo ►Polar

Apresenta *VectorComplexo* em forma polar.

- O modo de ângulo Graus devolve $(r \angle \theta)$.
- O modo de ângulo Radianos devolve $re^{i\theta}$.

ValorComplexo pode ter qualquer forma complexa. No entanto, uma entrada $re^{i\theta}$ provoca um erro no modo de ângulo Graus.

Nota: Tem de utilizar os parêntesis para uma entrada polar $(r \angle \theta)$.

No modo de ângulo Radianos:

$$\begin{array}{l} (3+4 \cdot i) \text{►Polar} \qquad e^{i \cdot \left(\frac{\pi}{2} - \tan^{-1} \left(\frac{3}{4} \right) \right)} \cdot 5 \\ \left(4 \angle \frac{\pi}{3} \right) \text{►Polar} \qquad e^{\frac{i \cdot \pi}{3}} \cdot 4 \end{array}$$

No modo de ângulo Gradianos:

$$(4 \cdot i) \text{►Polar} \qquad (4 \angle 100.)$$

No modo de ângulo Graus:

$$(3+4 \cdot i) \text{►Polar} \qquad \left(5 \angle 90 - \tan^{-1} \left(\frac{3}{4} \right) \right)$$

polyCoeffs()

polyCoeffs(Poly [, Var]) ⇒ lista

Devolve uma lista dos coeficientes do polinómio *Poly* em relação à variável *Var*.

Poly tem de ser uma expressão polinomial em *Var*. Recomendamos que não omita *Var*, excepto se *Poly* for uma expressão numa variável.

$$\text{polyCoeffs}(4 \cdot x^2 - 3 \cdot x + 2, x) \qquad \{4, -3, 2\}$$

$$\text{polyCoeffs}((x-1)^2 \cdot (x+2)^3) \qquad \{1, 4, 1, -10, -4, 8\}$$

Expande o polinómio e selecciona *x* para a *Var* omitida.

$\text{polyCoeffs}\left(\left(x+y+z\right)^2,x\right)$	$\left\{1,2\cdot\left(y+z\right),\left(y+z\right)^2\right\}$
$\text{polyCoeffs}\left(\left(x+y+z\right)^2,y\right)$	$\left\{1,2\cdot\left(x+z\right),\left(x+z\right)^2\right\}$
$\text{polyCoeffs}\left(\left(x+y+z\right)^2,z\right)$	$\left\{1,2\cdot\left(x+y\right),\left(x+y\right)^2\right\}$

polyDegree()

polyDegree(*Poli* [, *Var*]) $\Rightarrow$ *valor*

Devolve o grau da expressão polinomial *Poly* em relação à variável *Var*. Se omitir *Var*, a função **polyDegree()** selecciona uma predefinição das variáveis contidas no polinómio *Poly*.

Poly tem de ser uma expressão polinomial em *Var*. Recomendamos que não omita *Var*, excepto se *Poly* for uma expressão numa variável.

$\text{polyDegree}(5)$	0
$\text{polyDegree}(\ln(2)+\pi,x)$	0

Polinómios constantes

$\text{polyDegree}\left(4\cdot x^2-3\cdot x+2,x\right)$	2
$\text{polyDegree}\left(\left(x-1\right)^2\cdot\left(x+2\right)^3\right)$	5

$\text{polyDegree}\left(\left(x+y^2+z^3\right)^2,x\right)$	2
$\text{polyDegree}\left(\left(x+y^2+z^3\right)^2,y\right)$	4
$\text{polyDegree}\left(\left(x-1\right)^{10000},x\right)$	10000

O grau pode ser extraído mesmo que os coeficientes não possam. Isto porque o grau pode ser extraído sem expandir o polinómio.

polyEval()

polyEval(*Lista1*, *Expr1*) $\Rightarrow$ *expressão*

polyEval(*Lista1*, *Lista2*) $\Rightarrow$ *expressão*

$\text{polyEval}\left(\left\{a,b,c\right\},x\right)$	$a\cdot x^2+b\cdot x+c$
$\text{polyEval}\left(\left\{1,2,3,4\right\},2\right)$	26
$\text{polyEval}\left(\left\{1,2,3,4\right\},\left\{2,-7\right\}\right)$	$\left\{26,-262\right\}$

Interpreta o primeiro argumento como o coeficiente de um polinómio de grau descendente e devolve o polinómio avaliado para o valor do segundo argumento.

polyGcd()

polyGcd(*Expr1*, *Expr2*) $\Rightarrow$ expressão

Devolve o máximo divisor comum dos dois argumentos.

Expr1 e *Expr2* têm de ser expressões polinomiais.

Argumentos booleanos, lista e matriz não são permitidos.

$\text{polyGcd}(100, 30)$	10
$\text{polyGcd}(x^2 - 1, x - 1)$	$x - 1$
$\text{polyGcd}(x^3 - 6 \cdot x^2 + 11 \cdot x - 6, x^2 - 6 \cdot x + 8)$	$x - 2$

polyQuotient()

polyQuotient(*Poli1*, *Poli2* [, *Var*]) $\Rightarrow$ expressão

Devolve o quociente do polinómio *Poli1* dividido pelo polinómio *Poli2* em relação à variável especificada *Var*.

Poli1 e *Poli2* têm de ser expressões polinomiais em *Var*. Recomendamos que não omita *Var*, excepto se *Poli1* e *Poli2* forem expressões na mesma variável.

$\text{polyQuotient}(x - 1, x - 3)$	1
$\text{polyQuotient}(x - 1, x^2 - 1)$	0
$\text{polyQuotient}(x^2 - 1, x - 1)$	$x + 1$
$\text{polyQuotient}(x^3 - 6 \cdot x^2 + 11 \cdot x - 6, x^2 - 6 \cdot x + 8)$	x
$\text{polyQuotient}((x - y) \cdot (y - z), x + y + z, x)$	$y - z$
$\text{polyQuotient}((x - y) \cdot (y - z), x + y + z, y)$	$2 \cdot x - y + 2 \cdot z$
$\text{polyQuotient}((x - y) \cdot (y - z), x + y + z, z)$	$-(x - y)$

polyRemainder()

polyRemainder(*Poli1*, *Poli2* [, *Var*]) $\Rightarrow$ expressão

$\text{polyRemainder}(x - 1, x - 3)$	2
$\text{polyRemainder}(x - 1, x^2 - 1)$	$x - 1$
$\text{polyRemainder}(x^2 - 1, x - 1)$	0

polyRemainder()Catálogo > 

Devolve o resto do polinómio *Poli1* dividido pelo polinómio *Poly2* em relação à variável especificada *Var*.

Poli1 e *Poli2* têm de ser expressões polinomiais em *Var*. Recomendamos que não omita *Var*, excepto se *Poli1* e *Poli2* forem expressões na mesma variável.

$$\begin{array}{l} \text{polyRemainder}((x-y) \cdot (y-z), x+y+z, x) \\ \quad \quad \quad -(y-z) \cdot (2 \cdot y+z) \\ \hline \text{polyRemainder}((x-y) \cdot (y-z), x+y+z, y) \\ \quad \quad \quad -2 \cdot x^2 - 5 \cdot x \cdot z - 2 \cdot z^2 \\ \hline \text{polyRemainder}((x-y) \cdot (y-z), x+y+z, z) \\ \quad \quad \quad (x-y) \cdot (x+2 \cdot y) \end{array}$$

polyRoots()Catálogo > 

polyRoots(*Poli*, *Var*) $\Rightarrow$ lista

polyRoots(*ListaDeCoeficientes*) $\Rightarrow$ lista

A primeira sintaxe, **polyRoots(*Poli*, *Var*)**, devolve uma lista de raízes reais do polinómio *Poly* em relação à variável *Var*. Se não existirem raízes reais, devolve uma lista vazia: { }.

Poly tem de ser um polinómio numa variável.

A segunda sintaxe, **polyRoots(*ListaDeCoeficientes*)**, devolve uma lista de raízes reais para os coeficientes em *ListaDeCoeficientes*.

Nota: Consulte também **cPolyRoots()**, página 39.

$$\begin{array}{l} \text{polyRoots}(y^3+1, y) \quad \quad \quad \{-1\} \\ \hline \text{cPolyRoots}(y^3+1, y) \\ \quad \quad \quad \left\{ -1, \frac{1}{2} - \frac{\sqrt{3}}{2}i, \frac{1}{2} + \frac{\sqrt{3}}{2}i \right\} \\ \hline \text{polyRoots}(x^2+2 \cdot x+1, x) \quad \quad \quad \{-1, -1\} \\ \hline \text{polyRoots}(\{1, 2, 1\}) \quad \quad \quad \{-1, -1\} \end{array}$$

PowerRegCatálogo > 

PowerReg *X*, *Y* [, *Freq*] [, *Categoria*, *Incluir*]]

Calcula a regressão de potência $y = (a \cdot (x)^b)$ nas listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

X e Y são listas de variáveis independentes e dependentes.

Freq é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados X e Y correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

Categoria é uma lista de códigos de categorias para os dados X e Y correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $a \cdot (x)^b$
stat.a, stat.b	Parâmetros de regressão
stat.r ²	Coeficiente de determinação linear para dados transformados
stat.r	Coeficiente de correlação para dados transformados ($\ln(x)$, $\ln(y)$)
stat.Resid	Resíduos associados ao modelo de potência
stat.ResidTrans	Resíduos associados ao ajuste linear dos dados transformados
stat.XReg	Lista de dados na <i>Lista X</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.YReg	Lista de dados na <i>Lista Y</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.FreqReg	Lista de frequências correspondentes a <i>stat.XReg</i> e <i>stat.YReg</i>

Prgm

Calcule o GCD e visualize os resultados intermédios.

Bloco

EndPrgm

Modelo para criar um programa definido pelo utilizador. Tem de ser utilizado o comando **Define**, **Define BibPub** ou **Define BibPriv**.

Bloco pode ser uma afirmação, uma série de afirmações separadas pelo carácter ":" ou uma série de afirmações em linhas separadas.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

```

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()

Consulte [II \(\)](#), página 244.

Produto (PI)

Consulte [II \(\)](#), página 244.

product()

product(*Lista* [, *Início* [, *fim*]])
⇒ *expressão*

Apresenta o produto dos elementos contidos na *Lista*. *Início* e *Fim* são opcionais. Especificam um intervalo de elementos.

product(*Matriz1* [, *Início* [, *fim*]])
⇒ *matriz*

Devolve um vector da linha com os produtos dos elementos nas colunas de *Matriz1*. *Início* e *Fim* são opcionais. Especificam um intervalo de linhas.

product({ 1,2,3,4 })	24
product({ 2,x,y })	2 · x · y
product({ 4,5,8,9 },2,3)	40

product($\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$)	$\begin{bmatrix} 28 & 80 & 162 \end{bmatrix}$
product($\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix},1,2$)	$\begin{bmatrix} 4 & 10 & 18 \end{bmatrix}$

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

propFrac()

propFrac(*Expr1* [, *Var*]) $\Rightarrow$ expressão

propFrac(*rational_number*) devolve *rational_number* como a soma de um número inteiro e uma fracção com o mesmo sinal e uma magnitude do denominador maior que a magnitude do numerador.

propFrac(*rational_expression*, *Var*) devolve a soma das fracções adequadas e um polinómio em relação a *Var*. O grau de *Var* no denominador excede o grau de *Var* no numerador em cada fracção adequada. As potências similares de *Var* são recolhidas. Os termos e os factores são ordenados com *Var* como variável principal.

Se omitir *Var*, uma expansão da fracção adequada é efectuada em relação à variável principal. Os coeficientes da parte polinomial são efectuados adequadamente em relação à primeira variável principal, etc.

Para expressões racionais, **propFrac()** é mais rápida, mas uma alternativa menos extrema para **expand()**.

Pode utilizar a função **propFrac()** para representar as fracções mistas e demonstrar a adição e a subtracção de fracções mistas.

$$\text{propFrac}\left(\frac{4}{3}\right) \quad 1 + \frac{1}{3}$$

$$\text{propFrac}\left(\frac{-4}{3}\right) \quad -1 - \frac{1}{3}$$

$$\text{propFrac}\left(\frac{x^2+x+1}{x+1} + \frac{y^2+y+1}{y+1}, x\right) \quad \frac{1}{x+1} + x + \frac{y^2+y+1}{y+1}$$

$$\text{propFrac}(\text{Ans}) \quad \frac{1}{x+1} + x + \frac{1}{y+1} + y$$

$$\text{propFrac}\left(\frac{11}{7}\right) \quad 1 + \frac{4}{7}$$

$$\text{propFrac}\left(3 + \frac{1}{11} + 5 + \frac{3}{4}\right) \quad 8 + \frac{37}{44}$$

$$\text{propFrac}\left(3 + \frac{1}{11} - \left(5 + \frac{3}{4}\right)\right) \quad -2 - \frac{29}{44}$$

QR

Catálogo >

QR *Matriz*, *MatrizQ*, *MatrizR* [, *Tol*]

Calcula a factorização QR Householder de uma matriz complexa ou real. As matrizes Q e R resultantes são guardados nos *Matriz* especificados. A matriz Q é unitária. A matriz R é triangular superior.

Opcionalmente, qualquer elemento da matriz é tratado como zero se o valor absoluto for inferior a *Tol*. Esta tolerância só é utilizada se a matriz tiver entradas de ponto flutuante e não contiver variáveis simbólicas sem um valor atribuído. Caso contrário, *Tol* é ignorado.

- Se utilizar   ou definir o modo **Auto ou Aproximado** para Aproximado, os cálculos são efectuados com a aritmética do ponto flutuante.
- Se *Tol* for omitido ou não utilizado, a tolerância predefinida for calculada como:
 $5E-14 \cdot \max(\dim(\text{Matriz})) \cdot \text{rowNorm}(\text{Matriz})$

A factorização QR é calculada numericamente com as transformações Householder. A solução simbólica é calculada com Gram-Schmidt. As colunas em *qMatName* são vectores de base ortonormal que ligam o espaço definido pela *matriz*.

O número de ponto flutuante (9.) em m1 faz com que os resultados sejam calculados na forma de ponto flutuante.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9. \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9. \end{bmatrix}$
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	$\begin{bmatrix} 8.12404 & 9.60114 & 11.0782 \\ 0. & 0.904534 & 1.80907 \\ 0. & 0. & 0. \end{bmatrix}$

$\begin{bmatrix} m & n \\ o & p \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} m & n \\ o & p \end{bmatrix}$
QR m1,qm,rm Done	
qm	$\begin{bmatrix} \frac{m}{\sqrt{m^2+o^2}} & \frac{-\text{sign}(m \cdot p - n \cdot o) \cdot o}{\sqrt{m^2+o^2}} \\ \frac{o}{\sqrt{m^2+o^2}} & \frac{m \cdot \text{sign}(m \cdot p - n \cdot o)}{\sqrt{m^2+o^2}} \end{bmatrix}$
rm	$\begin{bmatrix} \sqrt{m^2+o^2} & \frac{m \cdot n + o \cdot p}{\sqrt{m^2+o^2}} \\ 0 & \frac{m \cdot p - n \cdot o}{\sqrt{m^2+o^2}} \end{bmatrix}$

QuadReg

Catálogo >

QuadReg *X*,*Y* [, *Freq*] [, *Categoria*, *Incluir*]

Calcula a regressão polinomial quadrática $y = a \cdot x^2 + b \cdot x + c$ a partir das listas X e Y com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter dimensões iguais, excepto para *Incluir*.

X e Y são listas de variáveis independentes e dependentes.

Freq é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados X e Y correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

Categoria é uma lista de códigos de categorias para os dados X e Y correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são incluídos no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $a \cdot x^2 + b \cdot x + c$
stat.a, stat.b, stat.c	Parâmetros de regressão
stat.R ²	Coeficiente de determinação
stat.Resid	Resíduos da regressão
stat.XReg	Lista de dados na <i>Lista X</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.YReg	Lista de dados na <i>Lista Y</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.FreqReg	Lista de frequências correspondentes a <i>stat.XReg</i> e <i>stat.YReg</i>

QuartReg *X*,*Y* [, *Freq*] [, *Categoria*,
Incluir]]

Calcula a regressão polinomial quártica $y = a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$ a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

X e *Y* são listas de variáveis independentes e dependentes.

Freq é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

Categoria é uma lista de códigos de categorias para os dados *X* e *Y* correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$
stat.a, stat.b, stat.c, stat.d, stat.e	Parâmetros de regressão
stat.R ²	Coefficiente de determinação
stat.Resid	Resíduos da regressão
stat.XReg	Lista de dados na <i>Lista X</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>

Variável de saída	Descrição
stat.YReg	Lista de dados na <i>Lista Y</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.FreqReg	Lista de frequências correspondentes a <i>stat.XReg</i> e <i>stat.YReg</i>

R

R►Pθ()

Catálogo > 

R►Pθ (*xExpr*, *yExpr*) ⇒ expressão

R►Pθ (*xLista*, *yLista*) ⇒ lista

R►Pθ (*xMatriz*, *yMatriz*) ⇒ matriz

Devolve a coordenada θ equivalente dos argumentos dos pares (x,y).

Nota: O resultado é devolvido como um ângulo expresso em graus, grados ou radianos, de acordo com a definição do modo de ângulo atual.

Nota: Pode introduzir esta função através da escrita de **R@Ptheta (...)** no teclado do computador

No modo de ângulo de grau:

$$\text{R►P}\theta(x,y) \quad 90 \cdot \text{sign}(y) - \tan^{-1}\left(\frac{x}{y}\right)$$

No modo de ângulo Grados:

$$\text{R►P}\theta(x,y) \quad 100 \cdot \text{sign}(y) - \tan^{-1}\left(\frac{x}{y}\right)$$

No modo de ângulo de Radianos:

$$\begin{aligned} \text{R►P}\theta(3,2) & \quad \tan^{-1}\left(\frac{2}{3}\right) \\ \text{R►P}\theta\left(\begin{bmatrix} 3 & -4 & 2 \end{bmatrix}, \begin{bmatrix} 0 & \frac{\pi}{4} & 1.5 \end{bmatrix}\right) & \quad \begin{bmatrix} 0 & \tan^{-1}\left(\frac{16}{\pi}\right) + \frac{\pi}{2} & 0.643501 \end{bmatrix} \end{aligned}$$

R►Pr()

Catálogo > 

R►Pr (*xExpr*, *yExpr*) ⇒ expressão

R►Pr (*xLista*, *yLista*) ⇒ lista

R►Pr (*xMatriz*, *yMatriz*) ⇒ matriz

Devolve a coordenada r equivalente dos argumentos dos pares (x,y).

Nota: Pode introduzir esta função através da escrita de **R@Pr (...)** no teclado do computador

No modo de ângulo de Radianos:

$$\begin{aligned} \text{R►Pr}(3,2) & \quad \sqrt{13} \\ \text{R►Pr}(x,y) & \quad \sqrt{x^2+y^2} \\ \text{R►Pr}\left(\begin{bmatrix} 3 & -4 & 2 \end{bmatrix}, \begin{bmatrix} 0 & \frac{\pi}{4} & 1.5 \end{bmatrix}\right) & \quad \begin{bmatrix} 3 & \frac{\sqrt{\pi^2+256}}{4} & 2.5 \end{bmatrix} \end{aligned}$$

Expr1 ► Rad ⇒ expressão

No modo de ângulo de grau:

Converte o argumento para a medição do ângulo de radianos.

$(1.5) \blacktriangleright \text{Rad}$	$(0.02618)^r$
--	---------------

Nota: Pode introduzir esta função através da escrita de @Rad no teclado do computador

No modo de ângulo Grados:

$(1.5) \blacktriangleright \text{Rad}$	$(0.023562)^r$
--	----------------

rand()

rand() ⇒ expressão

rand(#Tentativas) ⇒ lista

rand() devolve um valor aleatório entre 0 e 1.

Define a semente do número aleatório.

RandSeed 1147	Done
rand(2)	{ 0.158206, 0.717917 }

rand(#Tentativas) devolve uma lista com # valores aleatórios entre 0 e 1

randBin()

randBin(n, p) ⇒ expressão

randBin(n, p, #Trials) ⇒ lista

randBin(n, p) devolve um número real aleatório de uma distribuição binomial especificada.

randBin(80,0.5)	42
randBin(80,0.5,3)	{ 41, 32, 39 }

randBin(n, p, #Tentativas) devolve uma lista com números reais aleatórios #Tentativas de uma distribuição binomial especificada.

randInt()

randInt
(lowBound, upBound)
⇒ expressão

randInt
(LimiteInferior,
LimiteSuperior,
#Tentativas) ⇒
lista

randInt(3,10)	5
randInt(3,10,4)	{ 9, 7, 5, 8 }

randInt

(*LimiteInferior*,
LimiteSuperior)
devolve um número
inteiro aleatório no
intervalo
especificado pelos
limites dos números
inteiros
LimiteInferior e
LimiteSuperior.

randInt

(*LimiteInferior*,
LimiteSuperior
,*#Tentativas*)
devolve uma lista
com # números
inteiros aleatórios no
intervalo
especificado.

randMat()

randMat(*LinhasNum*, *ColunasNum*) ⇒
matriz

Devolve uma matriz de números inteiros
entre -9 e 9 da dimensão especificada.

Ambos os argumentos têm de ser
simplificados para números inteiros.

RandSeed 1147	Done		
randMat(3,3)	8	-3	6
	-2	3	-6
	0	4	-6

Nota: Os valores desta matriz mudam sempre
que prime .

randNorm()

randNorm(μ , σ) ⇒ *expressão*
randNorm(μ , σ , *#Tentativas*) ⇒ *lista*

randNorm(μ , σ) devolve um número
decimal da distribuição normal
específica. Pode ser qualquer número
real, mas estará fortemente concentrado
no intervalo $[\mu-3\cdot\sigma, \mu+3\cdot\sigma]$.

randNorm(μ , σ , *#Tentativas*) devolve
uma lista com números decimais
#Tentativas de uma distribuição normal
especificada.

RandSeed 1147	Done
randNorm(0,1)	0.492541
randNorm(3,4.5)	-3.54356

randPoly()**Catálogo >** **randPol y (Var, Ordem) ⇒ expressão**

Devolve um polinómio em *Var* da *Ordem* especificada. Os coeficientes são números inteiros aleatórios no intervalo -9 a 9. O coeficiente à esquerda não será zero.

Ordem tem de ser 0–99.

RandSeed 1147	Done
randPoly(<i>x</i> ,5)	$-2 \cdot x^5 + 3 \cdot x^4 - 6 \cdot x^3 + 4 \cdot x - 6$

randSamp()**Catálogo >** **randSamp(Lista,#Tentativas [,SemSubstituição]) ⇒ lista**

Devolve uma lista com uma amostra aleatória de tentativas #*Tentativas* de *Lista* com uma opção para substituição da amostra (*SemSubstituição*=0) ou sem substituição da amostra (*SemSubstituição*=1). A predefinição é com substituição da amostra.

Define list3={ 1,2,3,4,5 }	Done
Define list4=randSamp(list3,6)	Done
list4	{ 2,3,4,3,1,2 }

RandSeed**Catálogo >** **RandSeed Número**

Se *Número* = 0, define as sementes para as predefinições de fábrica para o gerador de números aleatórios. Se *Número* ≠ 0, é utilizado para gerar duas sementes, que são guardadas nas variáveis do sistema seed1 e seed2.

RandSeed 1147	Done
rand()	0.158206

real()**Catálogo >** **real(Expr1) ⇒ expressão**

Devolve a parte real do argumento.

Nota: Todas as variáveis indefinidas são tratadas como variáveis reais. Consulte também **imag()**, página 98.

real(Lista1) ⇒ lista

Devolve as partes reais de todos os elementos.

real(2+3·i)	2
real(<i>z</i>)	<i>z</i>
real(<i>x</i> +i· <i>y</i>)	<i>x</i>

real({ <i>a</i> +i· <i>b</i> ,3,i })	{ <i>a</i> ,3,0 }
--------------------------------------	-------------------

real(MatrizI) ⇒ *matriz*

Devolve as partes reais de todos os elementos.

$$\text{real}\left(\begin{bmatrix} a+i \cdot b & 3 \\ c & i \end{bmatrix}\right) \quad \begin{bmatrix} a & 3 \\ c & 0 \end{bmatrix}$$

► Rect

Vetor ► **Rect**

Nota: Pode introduzir este operador através da escrita de **@Rect** no teclado do computador.

Apresenta o *Vetor* na forma retangular [x, y, z] O vetor tem de ser de dimensão 2 ou 3 e pode ser uma linha ou uma coluna.

Nota: ► **Rect** é uma instrução de formato de visualização, não uma função de conversão. Só pode utilizá-la no fim de uma linha de entrada e não actualiza *ans*.

Nota: Consulte também ► **Polar**, página 145.

ValorComplexo ► **Rect**

Apresenta o *ValorComplexo* na forma retangular a+bi. O *ValorComplexo* pode ter qualquer forma complexa. No entanto, uma entrada $\text{re}^{i\theta}$ provoca um erro no modo de ângulo Graus.

Nota: Tem de utilizar os parêntesis para uma entrada em coordenadas polares ($r \angle \theta$).

$$\left(\begin{bmatrix} 3 & \angle \frac{\pi}{4} & \angle \frac{\pi}{6} \end{bmatrix}\right) \text{► Rect} \quad \begin{bmatrix} \frac{3 \cdot \sqrt{2}}{4} & \frac{3 \cdot \sqrt{2}}{4} & \frac{3 \cdot \sqrt{3}}{2} \end{bmatrix}$$

$$\begin{bmatrix} a & \angle b & \angle c \end{bmatrix} \quad \begin{bmatrix} a \cdot \cos(b) \cdot \sin(c) & a \cdot \sin(b) \cdot \sin(c) & a \cdot \cos(c) \end{bmatrix}$$

No modo de ângulo de Radianos:

$$\left(4 \cdot e^{\frac{\pi}{3}}\right) \text{► Rect} \quad 4 \cdot e^{\frac{\pi}{3}}$$

$$\left(\left(4 \angle \frac{\pi}{3}\right)\right) \text{► Rect} \quad 2 + 2 \cdot \sqrt{3} \cdot i$$

No modo de ângulo Grados:

$$\left(\left(1 \angle 100\right)\right) \text{► Rect} \quad i$$

No modo de ângulo de grau:

$$\left(\left(4 \angle 60\right)\right) \text{► Rect} \quad 2 + 2 \cdot \sqrt{3} \cdot i$$

Nota: Para escrever $\angle$, selecione-o na lista de símbolos no Catálogo.

ref(Matriz1[, Tol]) $\Rightarrow$ *matriz*

Devolve a forma de escalo~o-linha de *Matriz1*.

Opcionalmente, qualquer elemento da matriz é tratado como zero se o valor absoluto for inferior a *Tol*. Esta tolerância é utilizada apenas se a matriz tiver entradas de ponto flutuante e não contiver nenhuma variável simbólica sem nenhum valor atribuído. Caso contrário, *Tol* é ignorado.

- Se utilizar   ou definir o modo **Auto ou Aproximado** para Aproximado, os cálculos são efetuados com a aritmética de ponto flutuante.
- Se *Tol* for omitido ou não utilizado, a tolerância predefinida é calculada como:
 $5E-14 \cdot \max(\dim(\text{Matriz1})) \cdot \text{rowNorm}(\text{Matriz1})$

Evite elementos indefinidos em *Matriz1*. Podem originar resultados inesperados.

Por exemplo, se *a* for indefinido na expressão seguinte, aparece uma mensagem de aviso e o resultado é mostrado como:

$$\text{ref}\left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) \quad \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

O aviso aparece porque o elemento generalizado $1/a$ não seria válido para $a=0$.

Pode evitar isto guardando um valor para *a* anteriormente ou utilizando o operador de limite ("|") para substituir um valor, conforme indicado no exemplo seguinte.

$$\text{ref}\left(\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}\right) \quad \begin{bmatrix} 1 & \frac{-2}{5} & \frac{-4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow m1 \quad \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

ref(m1)  $\begin{bmatrix} 1 & \frac{d}{c} \\ 0 & 1 \end{bmatrix}$

$$\text{ref} \left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) | a=0 \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Nota: Consulte também **rref()**, página 171.

AtualizarVarsSonda

AtualizarVarsSonda

Permite-lhe aceder a dados de sensor a partir de todas as sondas de sensor ligadas no seu programa TI-Basic.

Valor	Estado
EstadoVar	
<i>estadoVar</i> =0	Normal (continue com o programa) A aplicação Vernier DataQuest™ está no modo de recolha de dados.
<i>estadoVar</i> =1	Nota: A aplicação Vernier DataQuest™ tem de estar no modo de medidor para que este comando funcione. 
<i>estadoVar</i> =2	A aplicação Vernier DataQuest™ não foi lançada.
<i>estadoVar</i> =3	A aplicação Vernier DataQuest™ foi lançada, mas não foram conectados sensores.

Exemplo

```

Define temp()=
Prgm
© Verifique se o sistema está pronto
Estado de AtualizarVarsSonda
Se estado=0 então
Apres "pronto"
Para n,1,50
Estado de AtualizarVarsSonda
temperatura:=medidor:temperatura
Apres "Temperatura": ",temperatura
Se temperatura>30 então
Apres "Demasiado quente"
EndIf
© Aguarde 1 segundo entre amostras
Aguarde 1
EndFor
Else
Apres "Não pronto, Tente novamente mais tarde"
EndIf
EndPrgm

```

Nota: Isto também pode ser utilizado com o Hub TI-Innovator™.

remain(*Expr1*, *Expr2*) ⇒ expressão
remain(*Lista1*, *Lista2*) ⇒ lista
remain(*Matriz1*, *Matriz2*) ⇒ matriz

Devolve o resto do primeiro argumento em relação ao segundo argumento conforme definido pelas identidades:

$\text{remain}(x, 0) = x$
 $\text{remain}(x, y) = x - y \cdot \text{iPart}(x/y)$

Por consequência, não se esqueça de que **remain**(-x,y) - remain(x,y). O resultado é zero ou tem o mesmo sinal do primeiro argumento.

Nota: Consulte também **mod()**, página 127.

$\text{remain}(7, 0)$	7
$\text{remain}(7, 3)$	1
$\text{remain}(-7, 3)$	-1
$\text{remain}(7, -3)$	1
$\text{remain}(-7, -3)$	-1
$\text{remain}(\{12, -14, 16\}, \{9, 7, -5\})$	$\{3, 0, 1\}$

$\text{remain}\left(\begin{bmatrix} 9 & -7 \\ 6 & 4 \end{bmatrix}, \begin{bmatrix} 4 & 3 \\ 4 & -3 \end{bmatrix}\right)$	$\begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$
--	---

Request (Pedido)

Pedido *promptString*, *var*[, *DispFlag* [, *statusVar*]]

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

Programar comando: Interrompe o programa e mostra uma caixa de diálogo com a mensagem *CadeiaDePedido* e uma caixa de entrada para a resposta do utilizador.

Quando o utilizador escrever uma resposta e clicar em **OK**, os conteúdos da caixa de entrada são atribuídos à variável *var*.

Se o utilizador clicar em **Cancelar**, o programa continua sem aceitar qualquer entrada. O programa utiliza o valor anterior de *var* se *var* já tiver sido definida.

O argumento *DispFlag* opcional podem ser qualquer expressão.

- Se *DispFlag* for omitido ou avalia para 1, a mensagem de pedido e a resposta do utilizador são apresentadas no histórico de Calculadora.

Definir um programa:

```
Definir request_demo()=Prgm
  Pedido "Raio: ",r
  Apres "Área = ",pi*r^2
EndPrgm
```

Execute o programa e escreva uma resposta:

request_demo()



Resultado após seleccionar **OK**:

Raio: 6/2
 Área= 28,2743

- Se *DispFlag* avaliar para **0**, o pedido e a resposta não são apresentados no histórico.

O argumento *statusVar* opcional proporciona uma forma de determinar como o utilizador ignorou a caixa de diálogo. Atente que *statusVar* requer o argumento *DispFlag*.

- Se o utilizador tiver clicado em **OK** ou premido **Enter** ou **Ctrl+Enter**, a variável *statusVar* é definida para um valor de **1**.
- Caso contrário, a variável *statusVar* é definida para um valor de **0**.

O argumento *func()* permite que um programa armazene a resposta do utilizador como uma definição de função. Esta sintaxe funciona como se o utilizador executasse o comando:

Definir *func(arg1, ...argn) = resposta do utilizador*

O programa pode então usar a função definida *func()*. A *CadeiaDePedido* deve guiar o utilizador para introduzir uma *resposta de utilizador* adequada que complete a definição de função.

Nota: Pode utilizar o comando **Pedido** dentro de um programa definido pelo utilizador, mas não dentro de uma função.

Para parar um programa que contém um comando **Pedido** dentro de um ciclo infinito:

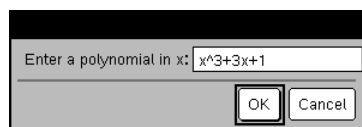
- **Dispositivo portátil:** Manter pressionada a tecla  e pressionar  repetidamente.
- **Windows®:** Manter pressionada a tecla **F12** e pressionar **Enter** repetidamente.
- **Macintosh®:** Manter pressionada a tecla **F5** e pressionar **Enter**

Definir um programa:

```
Definir polynomial()=Prgm
  Pedido "Introduza um polinómio em
x:",p(x)
  Apres "As raizes reais
são:",polyRoots(p(x),x)
EndPrgm
```

Execute o programa e escreva uma resposta:

polynomial()



Resultado depois de introduzir x^3+3x+1 e selecionar **OK**:

As raizes reais são: $\{-0.322185\}$

repetidamente.

- **iPad®:** A aplicação apresenta um pedido. Pode continuar a aguardar ou pode cancelar.

Nota: Consulte também **CadeiaDePedido**, página 165.

CadeiaDePedido

CadeiaDePedido *CadeiaDePedido, var [, DispFlag]*

Programar comando: Funciona de forma idêntica à primeira sintaxe do comando **Pedido**, exceto no facto de a resposta do utilizador ser sempre interpretada como uma cadeia. Em contraste, o comando **Pedido** interpreta a resposta como uma expressão, a não ser que o utilizador o coloque entre aspas ("").

Nota: Pode usar o comando **CadeiaDePedido** dentro de um programa definido pelo utilizador, mas não dentro de uma função.

Para parar um programa que contém um comando **CadeiaDePedido** dentro de um ciclo infinito:

- **Dispositivo portátil:** Manter pressionada a tecla  e pressionar  repetidamente.
- **Windows®:** Manter pressionada a tecla **F12** e pressionar **Enter** repetidamente.
- **Macintosh®:** Manter pressionada a tecla **F5** e pressionar **Enter** repetidamente.
- **iPad®:** A aplicação apresenta um pedido. Pode continuar a aguardar ou pode cancelar.

Nota: Consulte também **Pedido**, página 163.

Definir um programa:

```
Definir requestStr_demo()=Prgm
    CadeiaDePedido "O seu nome:",name,0
    Apres "A resposta tem ",dim(name),"
    caracteres."
EndPrgm
```

Execute o programa e escreva uma resposta:

requestStr_demo()



Resultado depois de se seleccionar **OK** (De referir que o argumento *DispFlag* de 0 omite o pedido e a resposta do histórico):

requestStr_demo()

A resposta tem 5 caracteres.

Return [*Expr*]

Devolve *Expr* como resultado da função.
Utilize num bloco **Func ... EndFunc**.

Nota: Utilize **Return** sem um argumento num bloco **Prgm...EndPrgm** para sair de um programa.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

```
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*

Devolve os elementos *Num* mais à direita contidos em *List1*.

Se omitir *Num*, devolve todos os elementos de *List1*.

right(*sourceString* [, *Num*]) ⇒ *cadeia*

Devolve os caracteres *Num* mais à direita na cadeia de caracteres *sourceString*

Se omitir *Num*, devolve todos os caracteres de *sourceString*.

right(*Comparação*) ⇒ *expressão*

Devolve o lado direito de uma equação ou desigualdade.

right({ 1,3,-2,4 },3) { 3,-2,4 }

right("Hello",2) "lo"

right(x<3) 3

rk23 ()

rk23(*Expr*, *Var*, *depVar*, {*Var0*, *VarMax*}, *depVar0*, *VarStep* [, *diftoI*])
⇒ *matriz*

rk23(*SystemOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *diftoI*]) ⇒ *matriz*

Equação diferencial:

$y' = 0.001 \cdot y \cdot (100 - y)$ e $y(0) = 10$

rk23{0.001·y·(100-y),t,y,{ 0,100 },10,1}

0.	1.	2.	3.	4.
10.	10.9367	11.9493	13.042	14.2

rk23(*ListOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep*[, *difto1*]) $\Rightarrow$ *matriz*

Utiliza o método Runge-Kutta para resolver o sistema

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

com *depVar*(*Var0*)=*depVar0* no intervalo [*Var0*,*VarMax*]. Apresenta uma matriz cuja primeira fila define os valores de saída *Var* conforme definido por *VarStep*. A segunda fila define o valor da primeira componente da solução nos valores *Var* correspondentes, e assim por diante.

Expr é o segundo membro que define a equação diferencial ordinária (EDO).

SystemOfExpr é o sistema de segundos membros que definem o sistema de EDOs (corresponde à ordem de variáveis dependentes em *ListOfDepVars*).

ListOfExpr é uma lista de segundos membros que definem o sistema de EDOs (corresponde à ordem de variáveis dependentes em *ListOfDepVars*).

Var é a variável independente.

ListOfDepVars é uma lista de variáveis dependentes.

{*Var0*, *VarMax*} é uma lista de dois elementos que informa a função para integrar de *Var0* a *VarMax*.

ListOfDepVars0 é uma lista de valores iniciais para variáveis dependentes.

Para ver o resultado completo, prima $\blacktriangle$ e, de seguida, utilize $\blacktriangleleft$ e $\blacktriangleright$ para mover o cursor.

Mesma equação com *difto1* definido para 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

Compare o resultado acima com a solução exacta CAS obtida através de *deSolve*() e *seqGen*():

$$\text{deSolve}\left\{y'=0.001 \cdot y \cdot \{100-y\} \text{ and } y(0)=10, t, y\right\}$$

$$y = \frac{100 \cdot \{1.10517\}^t}{\{1.10517\}^t + 9}$$

$$\text{seqGen}\left\{\frac{100 \cdot \{1.10517\}^t}{\{1.10517\}^t + 9}, t, y, \{0, 100\}\right\}$$

{ 10., 10.9367, 11.9494, 13.0423, 14.2189, 15.49
--

Sistema de equações:

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

com *y1*(0)=2 e *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

Se $VarStep$ avalia para um número diferente de zero: $signal(VarStep) = signal(VarMax - Var0)$ e soluções são apresentadas em $Var0 + i * VarStep$ para todos os $i=0,1,2,...$ tal como $Var0 + i * VarStep$ está em $[var0, VarMax]$ (pode não obter um valor de solução em $VarMax$).

se *VarStep* avaliar para zero, as soluções são apresentadas nos valores *Var* Runge-Kutta".

difftol é a tolerância de erro (passa para 0,001).

root()

$$\text{root}(Expr) \Rightarrow raiz$$
$$\text{root}(Expr1, Expr2) \Rightarrow raiz$$

root(*Expr*) devolve a raiz quadrada de *Expr*.

root(*Expr1*, *Expr2*) devolve a raiz de *Expr2* de *Expr1*. *Expr1* pode ser uma constante de ponto flutuante complexa, uma constante racional complexa ou número inteiro, ou uma expressão simbólica geral.

Nota: Consulte também **Modelo da raiz de índice N**, página 2.

$\sqrt[3]{8}$	2
$\sqrt[3]{3}$	$\frac{1}{3^3}$
$\sqrt[3]{3.}$	1.44225

rotate()

rotate(*NúmeroInteiro*l[, #deRotações])
⇒ *número inteiro*

No modo base Bin:

Roda os bits num número inteiro binário. Pode introduzir *NúmeroInteiro1* em qualquer base numérica; é convertido automaticamente para uma forma binária de 64 bits assinada. Se a magnitude de *NúmeroInteiro1* for demasiado grande para esta forma, uma operação do módulo simétrico coloca-o no intervalo. (Para mais informações, consulte ► **Base2**, página 19.

```
rotate(0b11111111111111111111111111111111)
    0b1000000000000000000000000000000001>
rotate(256,1)                                0b1000000000
```

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

Se *#deRotações* for positivo, a rotação é para a esquerda. Se *#deRotações* for negativo, a rotação é para a direita. A predefinição é -1 (rodar um elemento para a direita).

Por exemplo, numa rotação para a direita:

Cada bit roda para a direita.

0b00000000000001111010110000110101

O bit mais à direita roda para o extremo esquerdo.

produz:

0b10000000000000111101011000011010

Os resultados aparecem de acordo com o modo base.

rotate(*Lista1*[, *#deRotações*]) ⇒ *lista*

Devolve uma cópia de *Lista1* rodada para a direita ou para a esquerda pelos elementos *#deRotações*. Não altera *Lista1*.

Se *#deRotações* for positivo, a rotação é para a esquerda. Se *#deRotações* for negativo, a rotação é para a direita. A predefinição é -1 (rodar um elemento para a direita).

rotate(*Cadeia1*[, *#deRotações*]) ⇒ *cadeia*

Devolve uma cópia de *Cadeia1* rodada para a direita ou para a esquerda pelos caracteres *#deRotações*. Não altere *Cadeia1*.

Se *#deRotações* for positivo, a rotação é para a esquerda. Se *#deRotações* for negativo, a rotação é para a direita. A predefinição é -1 (rodar um carácter para a direita).

No modo base Hex:

rotate(0h78E)	0h3C7
rotate(0h78E,-2)	0h80000000000001E3
rotate(0h78E,2)	0h1E38

Importante: Para introduzir um número binário ou hexadecimal, utilize sempre o prefixo 0b ou 0h (zero, não a letra O).

No modo base Dec:

rotate({ 1,2,3,4 })	{ 4,1,2,3 }
rotate({ 1,2,3,4 },-2)	{ 3,4,1,2 }
rotate({ 1,2,3,4 },1)	{ 2,3,4,1 }

rotate("abcd")	"dabc"
rotate("abcd",-2)	"cdab"
rotate("abcd",1)	"bcda"

round(*Expr1*[, *dígitos*]) ⇒ *expressão*

round(1.234567,3)	1.235
-------------------	-------

round()Catálogo > 

Devolve o argumento arredondado para o número especificado de dígitos após o ponto decimal.

dígitos tem de ser um número inteiro no intervalo 0–12. Se *dígitos* não for incluído, devolve o argumento arredondado para 12 dígitos significativos.

Nota: A visualização do modo de dígitos pode afetar como este é apresentado.

round (*Lista1* [, *dígitos*]) $\Rightarrow$ *lista*

Devolve uma lista dos elementos arredondada para o número especificado de dígitos.

round (*Matriz1* [, *dígitos*]) $\Rightarrow$ *matriz*

Devolve uma matriz dos elementos arredondados para o número especificado de dígitos.

$$\text{round}(\{\pi, \sqrt{2}, \ln(2)\}, 4)$$

$$\{3.1416, 1.4142, 0.6931\}$$

$$\text{round}\left(\begin{pmatrix} \ln(5) & \ln(3) \\ \pi & e^1 \end{pmatrix}, 1\right)$$

$$\begin{bmatrix} 1.6 & 1.1 \\ 3.1 & 2.7 \end{bmatrix}$$

rowAdd()Catálogo > 

rowAdd(*Matriz1*, *rIndex1*, *rIndex2*) $\Rightarrow$ *matriz*

Devolve uma cópia de *Matriz1* com a linha *rIndex2* substituída pela soma das linhas *rIndex1* e *rIndex2*.

$$\text{rowAdd}\left(\begin{bmatrix} 3 & 4 \\ -3 & -2 \end{bmatrix}, 1, 2\right)$$

$$\begin{bmatrix} 3 & 4 \\ 0 & 2 \end{bmatrix}$$

$$\text{rowAdd}\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}, 1, 2\right)$$

$$\begin{bmatrix} a & b \\ a+c & b+d \end{bmatrix}$$

rowDim()Catálogo > 

rowDim(*Matriz*) $\Rightarrow$ *expressão*

Devolve o número de linhas em *Matriz*.

Nota: Consulte também **colDim()**, página 28.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow m1$$

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

3

rowNorm()Catálogo > 

rowNorm(*Matriz*) $\Rightarrow$ *expressão*

$$\text{rowNorm}\left(\begin{bmatrix} -5 & 6 & -7 \\ 3 & 4 & 9 \\ 9 & -9 & -7 \end{bmatrix}\right)$$

25

Devolve o máximo das somas dos valores absolutos dos elementos nas linhas em *Matriz*.

Nota: Todos os elementos da matriz têm de ser simplificados para números. Consulte também **colNorm()**, página 29.

rowSwap()

rowSwap (*Matriz1*, *rIndex1*, *rIndex2*)
 $\Rightarrow$ *matriz*

Devolve *Matriz1* com as linhas *rIndex1* e *rIndex2* trocadas.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow mat$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
$rowSwap(mat, 1, 3)$	$\begin{bmatrix} 5 & 6 \\ 3 & 4 \\ 1 & 2 \end{bmatrix}$

rref()

rref(*Matriz1* [, *Tol*]) $\Rightarrow$ *matriz*

Devolve a forma de escaloão-linha reduzida de *Matriz1*.

$rref\left(\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}\right)$	$\begin{bmatrix} 1 & 0 & 0 & \frac{66}{71} \\ 0 & 1 & 0 & \frac{147}{71} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$
---	---

Opcionalmente, qualquer elemento da matriz é tratado como zero se o valor absoluto for inferior a *Tol*. Esta tolerância é utilizada apenas se a matriz tiver entradas de ponto flutuante e não contiver nenhuma variável simbólica sem nenhum valor atribuído. Caso contrário, *Tol* é ignorado.

 $rref\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right)$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
---	--

- Se utilizar   ou definir o modo **Auto ou Aproximado** para Aproximado, os cálculos são efetuados com a aritmética de ponto flutuante.
- Se *Tol* for omitido ou não utilizado, a tolerância predefinida é calculada como:
 $5E-14 \cdot \max(\dim(Matriz1)) \cdot rowNorm(Matriz1)$

Nota: Consulte também **ref()**, página 161.

sec()

Tecla

sec(Expr1) ⇒ expressão

No modo de ângulo Graus:

sec(Lista1) ⇒ lista

Devolve a secante de *Expr1* ou devolve uma lista com as secantes de todos os elementos em *Lista1*.

$$\begin{array}{l} \text{sec}(45) \qquad \qquad \qquad \sqrt{2} \\ \text{sec}(\{1,2,3,4\}) \qquad \left\{ \frac{1}{\cos(1)}, 1.00081, \frac{1}{\cos(4)} \right\} \end{array}$$

Nota: O argumento é interpretado como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual. Pode utilizar °, G ou r para substituir o modo de ângulo temporariamente.

sec⁻¹()

Tecla

sec⁻¹(Expr1) ⇒ expressão

No modo de ângulo Graus:

sec⁻¹(Lista1) ⇒ lista

Devolve o ângulo cuja secante é *Expr1* ou devolve uma lista com as secantes inversas de cada elemento de *Lista1*.

$$\text{sec}^{-1}(1) \qquad \qquad \qquad 0$$

No modo de ângulo Gradianos:

Nota: O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

$$\text{sec}^{-1}(\sqrt{2}) \qquad \qquad \qquad 50$$

Nota: Pode introduzir esta função através da escrita de **arcsec (...)** no teclado do computador.

No modo de ângulo Radianos:

$$\text{sec}^{-1}(\{1,2,5\}) \qquad \qquad \qquad \left\{ 0, \frac{\pi}{3}, \text{sec}^{-1}\left(\frac{1}{5}\right) \right\}$$

sech()

Catálogo >

sech(Expr1) ⇒ expressão

$$\begin{array}{l} \text{sech}(3) \qquad \qquad \qquad \frac{1}{\cosh(3)} \end{array}$$

sech(Lista1) ⇒ lista

Devolve a secante hiperbólica de *Expr1* ou devolve uma lista com as secantes hiperbólicas dos elementos *Lista1*.

$$\begin{array}{l} \text{sech}(\{1,2,3,4\}) \qquad \left\{ \frac{1}{\cosh(1)}, 0.198522, \frac{1}{\cosh(4)} \right\} \end{array}$$

sech⁻¹(Expr1) ⇒ expressão

sech⁻¹(Lista1) ⇒ lista

Devolve a secante hiperbólica inversa de *Expr1* ou devolve uma lista com as secantes hiperbólicas inversas de cada elemento de *Lista1*.

Nota: Pode introduzir esta função através da escrita de **arcsech** (...) no teclado do computador.

No modo de ângulo Radianos e Formato complexo rectangular:

sech ⁻¹ (1)	0
sech ⁻¹ ({ 1, -2, 2.1 })	$\left\{ 0, \frac{2 \cdot \pi}{3} \cdot i, 8. \text{E}^{-15} + 1.07448 \cdot i \right\}$

Send

Menu Hub

Send *exprOrString1* [, *exprOrString2*] ...

Programar comando: envia um ou mais TI-Innovator™ Hub comandos para um hub conectado.

exprOrString tem de ser um TI-Innovator™ Hub comando válido. Tipicamente, *exprOrString* contém um comando **"SET ..."** para controlar um dispositivo ou um comando **"READ ..."** para pedir dados.

Os argumentos são enviados sequencialmente para o hub.

Nota: pode usar o comando **Send** dentro de um programa definido pelo utilizador, mas não dentro de uma função.

Nota: ver também **Get** (página 85), **GetStr** (página 93) e **eval()** (página 67).

Exemplo: ligar o elemento azul do LED RGB incorporado durante 0,5 segundos.

Send "SET COLOR.BLUE ON TIME .5"	Done
----------------------------------	------

Exemplo: pedir o valor atual do sensor de nível de luz incorporado no hub. Um comando **Get** recupera o valor e atribui-o à variável *lightval*.

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

Exemplo: enviar uma frequência calculada para o altifalante incorporado no hub. Usar a variável especial *iostr.SendAns* para mostrar o comando do hub com a expressão avaliada.

<i>n</i> :=50	50
<i>m</i> :=4	4
Send "SET SOUND eval(m·n)"	Done
<i>iostr.SendAns</i>	"SET SOUND 200"

seq()

Catálogo > 

seq(Expr, Var, Baixo, Alto [, Passo])
⇒lista

Incrementa *Var* de *Baixo* até *Alto* por um incremento de *Passo*, avalia *Expr* e apresenta os resultados como uma lista. O conteúdo original de *Var* ainda está aqui após a conclusão de **seq()**.

O valor predefinido para *Passo* = 1.

$$\begin{array}{l} \text{seq}\left(n^2, n, 1, 6\right) \quad \{1, 4, 9, 16, 25, 36\} \\ \text{seq}\left(\frac{1}{n}, n, 1, 10, 2\right) \quad \left\{1, \frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{9}\right\} \\ \text{sum}\left(\text{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right) \quad \frac{1968329}{1270080} \end{array}$$

Obs: Para forçar um resultado aproximado,

Unidade portátil: Premir  .

Windows®: Premir **Ctrl+Enter**.

Macintosh®: Premir +**Enter**.

iPad®: Manter pressionada a tecla **Enter** e selecionar .

$$\text{sum}\left(\text{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right) \quad 1.54977$$

seqGen()

Catálogo > 

seqGen(Expr, Var, depVar, {Var0, VarMax}[, ListOfInitTerms [, VarStep [, CeilingValue]]]) ⇒lista

Gera uma lista de termos para sequência *depVar(Var)=Expr* da seguinte forma: Incrementa a variável independente *Var* de *Var0* até *VarMax* por *VarStep*, avalia *depVar(Var)* para os valores correspondentes de *Var* utilizando a fórmula *Expr* e *ListOfInitTerms* e apresenta os resultados como uma lista.

seqGen(ListOrSystemOfExpr, Var, ListOfDepVars, {Var0, VarMax}[, MatrixOfInitTerms [, VarStep [, CeilingValue]]]) ⇒matriz

Gere o primeiros 5 termos da sequência $u(n) = u(n-1)^2/2$, com $u(1)=2$ e $VarStep=1$.

$$\begin{array}{l} \text{seqGen}\left(\frac{(u(n-1))^2}{n}, n, u, \{1, 5\}, \{2\}\right) \\ \left\{2, 2, \frac{4}{3}, \frac{4}{9}, \frac{16}{405}\right\} \end{array}$$

Exemplo no qual $Var0=2$:

$$\begin{array}{l} \text{seqGen}\left(\frac{u(n-1)+1}{n}, n, u, \{2, 5\}, \{3\}\right) \\ \left\{3, \frac{4}{3}, \frac{7}{12}, \frac{19}{60}\right\} \end{array}$$

Exemplo no qual o termo inicial é simbólico:

$$\begin{array}{l} \text{seqGen}\left(u(n-1)+2, n, u, \{1, 5\}, \{a\}\right) \\ \{a, a+2, a+4, a+6, a+8\} \end{array}$$

Sistema de duas seqüências:

Gera uma matriz de termos de um sistema (ou lista) de sequências

ListOfDepVars

(*Var*)=*ListOrSystemOfExpr* da seguinte

forma: Incrementa a variável

independente *Var* de *Var0* até *VarMax*

por *VarStep*, avalia *ListOfDepVars(Var)*

para os valores correspondentes de *Var* utilizando a fórmula

ListOrSystemOfExpr e

MatrixOfInitTerms e apresenta os

resultados como uma matriz.

O conteúdo original de *Var* está inalterado após a conclusão de **seqGen()**.

O valor predefinido para *VarStep* = 1.

$$\text{seqGen}\left(\left\{\frac{1}{n}, \frac{u_1(n-1)}{2} + u_1(n-1)\right\}, n, \{u_1, u_2\}, \{1, 5\}, \begin{bmatrix} - \\ 2 \end{bmatrix}\right)$$

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{2} & \frac{19}{2} \end{bmatrix}$$

Nota: O Vazio () na matriz do termo inicial acima, é utilizado para indicar que o 1º termo para $u_1(n)$ é calculado utilizando a fórmula de sucessão $u_1(n)=1/n$.

seqn()

seqn(*Expr(u, n [, ListOfInitTerms[, nMax [, CeilingValue]]])*)⇒*lista*

Gera uma lista de termos para uma sucessão $u(n)=Expr(u, n)$, da seguinte forma: Incrementa *n* a partir de 1 até *nMax* por 1, avalia $u(n)$ para os valores correspondentes de *n* utilizando a fórmula $Expr(u, n)$ e *ListOfInitTerms* e apresenta os resultados como uma lista.

seqn(*Expr(n [, nMax [, CeilingValue]]*)⇒*lista*

Gera uma lista de termos para uma sucessão não recorrente $u(n)=Expr(n)$, da seguinte forma: Incrementa *n* a partir de 1 até *nMax* por 1, avalia $u(n)$ para os valores correspondentes de *n* utilizando a fórmula $Expr(n)$ e apresenta os resultados como uma lista.

Se *nMax* estiver em falta, *nMax* é definido para 2500

Se *nMax*=0, *nMax* é definido para 2500

Nota: **seqn()** chamadas **seqGen()** com $n0=1$ e $nstep=1$

Gere o primeiros 6 termos da sequência $u(n) = u(n-1)/2$, com $u(1)=2$.

$$\text{seqn}\left(\frac{u(n-1)}{n}, \{2\}, 6\right)$$

$$\left\{2, 1, \frac{1}{3}, \frac{1}{12}, \frac{1}{60}, \frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2}, 6\right)$$

$$\left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25}, \frac{1}{36}\right\}$$

série(*Expr1*, *Var*, *Ordem* [, *Ponto*]) $\Rightarrow$ expressão

série(*Expr1*, *Var*, *Ordem* [, *Ponto*]) | *Var*>*Ponto* $\Rightarrow$ expressão

série(*Expr1*, *Var*, *Ordem* [, *Ponto*]) *Var*<*Ponto* $\Rightarrow$ expressão

Devolve uma representação da série da potência truncada generalizada de *Expr1* expandida sobre *Ponto* através do grau *Ordem*. *Ordem* pode ser qualquer número racional. As potências resultantes de (*Var* – *Ponto*) podem incluir expoentes fraccionais e/ou negativos. Os coeficientes destas potências podem incluir logaritmos de (*Var* – *Ponto*) e outras funções de *Var* que são dominadas pelas potências de (*Var* – *Ponto*) com o mesmo sinal de expoente.

Ponto predefine-se para 0. *Ponto* pode ser ∞ ou $-\infty$, nestes casos, a expansão é efectuada através de grau *Ordem* em $1/(Var - Ponto)$.

série(...) devolve “**série(...)**” se não for capaz de determinar uma representação, como para singularidades essenciais, como, por exemplo, $\sin(1/z)$ a $z=0$, $e^{-1/z}$ a $z=0$, ou e^z a $z = \infty$ ou $-\infty$.

$$\text{series}\left(\frac{1-\cos(x-1)}{(x-1)^2}, x, 4, 1\right) \quad \frac{1}{2} - \frac{(x-1)^2}{24} + \frac{(x-1)^4}{720}$$

$$\text{series}\left(\frac{-1}{e^{z-}}, z, -1\right) \quad z - 1$$

$$\text{series}\left(\left(1 + \frac{1}{n}\right)^n, n, 2, \infty\right) \quad e - \frac{e}{2 \cdot n} + \frac{11 \cdot e}{24 \cdot n^2}$$

$$\text{series}\left(\tan^{-1}\left(\frac{1}{x}\right), x, 5\right), x > 0 \quad \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5}$$

$$\text{series}\left(\int \frac{\sin(x)}{x} dx, x, 6\right) \quad x - \frac{x^3}{18} + \frac{x^5}{600}$$

$$\text{series}\left(\int_0^x \sin(x \cdot \sin(t)) dt, x, 7\right) \quad \frac{x^3}{2} - \frac{x^5}{24} - \frac{29 \cdot x^7}{720}$$

$$\text{series}\left(\left(1 + e^x\right)^2, x, 2, 1\right) \quad (e+1)^2 + 2 \cdot e \cdot (e+1) \cdot (x-1) + e \cdot (2 \cdot e+1) \cdot (x-1)^2$$

Se a série ou um das derivadas tiver uma descontinuidade em *Ponto*, o resultado contém provavelmente subexpressões do sinal(...) ou abs(...) da forma para uma variável de expansão real ou $(-1)^{\text{floor}(\dots \text{ângulo}(\dots) \dots)}$ para uma variável de expansão complexa, que é uma que termina com “_”. Se quiser utilizar a série apenas para valores num lado de *Ponto*, adicione o valor adequado de “| *Var* > *Ponto*”, “| *Var* < *Ponto*”, “| *Var* ≥ *Ponto*”, ou “*Var* ≤ *Ponto*” para obter um resultado mais simples.

série() pode fornecer aproximações simbólicas para integrais indefinidos para os quais soluções simbólicas podem não ser obtidas.

série() distribui-se pelas listas e matrizes do 1º argumento.

série() é uma versão generalizada de **taylor()**.

Como ilustrado pelo último exemplo da direita, o fluxo das rotinas do visor do resultado produzido pela série(...) pode reorganizar os termos para que o termo dominante não seja o termo mais à esquerda.

Nota: Consulte também **dominantTerm()**, página 61.

setMode()

setMode(NúmeroInteiroNomeModo, NúmeroInteiroDefinição) ⇒ *número inteiro*

setMode(lista) ⇒ *lista de números inteiros*

Válido apenas numa função ou num programa.

Apresente o valor aproximado de π com a predefinição para Ver dígitos e apresente π com uma definição de Fix2. Certifique-se de que a predefinição é restaurada após a execução do programa.

setMode(NúmeroInteiroNomeModo, NúmeroInteiroDefinição) define temporariamente o modo *NúmeroInteiroNomeModo* para a nova definição *NúmeroInteiroDefinição* e devolve um número inteiro correspondente à definição original desse modo. A alteração é limitada à duração da execução do programa/função.

NúmeroInteiroNomeModo especifica que modo quer definir. Tem de ser um dos números inteiros do modo da tabela abaixo.

NúmeroInteiroDefinição especifica a nova definição do modo. Tem de ser um dos números inteiros da definição listados abaixo para o modo específico que está a definir.

setMode(lista) permite alterar várias definições. *lista* contém os pares de números inteiros do modo e da lista. **setMode(lista)** devolve uma lista similar cujos pares de números inteiros representam as definições e os modos originais.

Se guardou todas as definições do modo com **getMode(0) → var**, pode utilizar **setMode(var)** para restaurar essas definições até sair da função ou do programa. Consulte **getMode()**, página 92.

Nota: As definições do modo actual são passadas para subrotinas. Se uma subrotina alterar uma definição do modo, a alteração do modo perder-se-á quando o controlo voltar à rotina.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

Define <i>prog1()</i> =Prgm	Done
Disp approx(π)	
setMode(1,16)	
Disp approx(π)	
EndPrgm	
<i>prog1()</i>	
	3.14159
	3.14
	Done

Nome do modo	Número inteiro do modo	Números inteiros da definição
Ver dígitos	1	1 =Flutuante, 2 =Flutuante1, 3 =Flutuante2, 4 =Flutuante3, 5 =Flutuante4, 6 =Flutuante5, 7 =Flutuante6, 8 =Flutuante7, 9 =Flutuante8, 10 =Flutuante9, 11 =Flutuante10, 12 =Flutuante11, 13 =Flutuante12, 14 =Fixo0, 15 =Fixo1, 16 =Fixo2, 17 =Fixo3, 18 =Fixo4, 19 =Fixo5, 20 =Fixo6, 21 =Fixo7, 22 =Fixo8, 23 =Fixo9, 24 =Fixo10, 25 =Fixo11, 26 =Fixo12
Ângulo	2	1 =Radianos, 2 =Graus, 3 =Gradianos
Formato exponencial	3	1 =Normal, 2 =Científica, 3 =Engenharia
Real ou Complexo	4	1 =Real, 2 =Rectangular, 3 =Polar
Auto or Aprox.	5	1 =Auto, 2 =Aproximado, 3 =Exacto
Formato vectorial	6	1 =Rectangular, 2 =Cilíndrico, 3 =Esférico
Base	7	1 =Decimal, 2 =Hex, 3 =Binário
Sistema de unidades	8	1 =SI, 2 =Eng/EUA

shift()

Catálogo > 

shift(*NúmeroInteiro1* [, *#deDeslocações*]) ⇒ *número inteiro*

Desloca os bits num número inteiro binário. Pode introduzir *NúmeroInteiro1* em qualquer base numérica; é convertido automaticamente para uma forma binária de 64 bits assinada. Se a magnitude de *NúmeroInteiro1* for muito grande para esta forma, uma operação do módulo simétrico coloca-o no intervalo. Para mais informações, consulte **Base2**, página 19.

Se *#deDeslocações* for positivo, a deslocação é para a esquerda. Se *#deDeslocações* for negativo, a deslocação é para a direita. A predefinição é -1 (deslocar um bit para a direita).

No modo base Bin:

```
shift(0b1111010110000110101)
                                0b111101011000011010
shift(256,1)                    0b1000000000
```

No modo base Hex:

```
shift(0h78E)                    0h3C7
shift(0h78E,-2)                 0h1E3
shift(0h78E,2)                  0h1E38
```

Importante: Para introduzir um número binário ou hexadecimal, utilize sempre o prefixo 0b ou 0h (zero, não a letra O).

Numa deslocação para a direita, o bit mais à direita cai e 0 ou 1 é inserido para corresponder ao bit mais à esquerda. Numa deslocação para a esquerda, o bit mais à esquerda cai e 0 é inserido como o bit mais à direita.

Por exemplo, numa deslocação para a direita:

Cada bit desloca-se para a direita.

```
0b00000000000000111101011000011010
```

Inserir 0 se o bit mais à esquerda for 0

ou 1 se o bit mais à esquerda for 1.

produz:

```
0b000000000000000111101011000011010
```

O resultado aparece de acordo com o modo base. Os zeros à esquerda não aparecem.

shift(Lista1 [, #deDeslocações]) ⇒ lista

Devolve uma cópia de *Listas1* deslocada para a direita ou para a esquerda pelos elementos *#deDeslocações*. Não altere *Listas1*.

Se *#deDeslocações* for positivo, a deslocação é para a esquerda. Se *#deDeslocações* for negativo, a deslocação é para a direita. A predefinição é -1 (deslocar um elemento para a direita).

Os elementos introduzidos no início ou no fim de *lista* pela deslocação são definidos para o símbolo "undef".

shift(Cadeia1 [, #deDeslocações]) ⇒ cadeia

Devolve uma cópia de *Cadeia1* rodada para a direita ou para a esquerda pelos caracteres *#deDeslocações*. Não altere *Cadeia1*.

No modo base Dec:

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 }

shift("abcd")	" abc "
shift("abcd", -2)	" ab "
shift("abcd", 1)	"bcd "

Se *#deDeslocações* for positivo, a deslocação é para a esquerda. Se *#deDeslocações* for negativo, a deslocação é para a direita. A predefinição é -1 (deslocar um carácter para a direita).

Os caracteres introduzidos no início ou no fim de *lista* pela deslocação são definidos para um espaço.

sign()

sign(*Expr1*) $\Rightarrow$ expressão

$\text{sign}(-3.2)$ -1.

sign(*Lista1*) $\Rightarrow$ lista

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

sign(*Matriz1*) $\Rightarrow$ matriz

$\text{sign}(1+|x|)$ 1

Para *Expr1* real ou complexa, devolve *Expr1* / **abs**(*Expr1*) quando *Expr1* $\neq$ 0.

Se o modo do formato complexo for Real:

Devolve 1 se *Expr1* for positiva.

$\text{sign}\left(\begin{bmatrix} -3 & 0 & 3 \end{bmatrix}\right)$ $\begin{bmatrix} -1 & \pm 1 & 1 \end{bmatrix}$

Devolve -1 se *Expr1* for negativa.

sign(0) devolve ± 1 se o modo do formato complexo for Real; caso contrário, devolve-se a si próprio.

sign(0) representa o círculo no domínio complexo.

Para uma lista ou matriz, devolve os sinais de todos os elementos.

simult()

simult(*MatrizCoef*, *VectorConst* [, *Tol*]) $\Rightarrow$ matriz

Resolver para x e y:

Devolve um vector da coluna que contém as soluções para um sistema de equações lineares.

$$x + 2y = 1$$

$$3x + 4y = -1$$

Nota: Consulte também **linSolve()**, página 112.

$\text{simult}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right)$ $\begin{bmatrix} -3 \\ 2 \end{bmatrix}$

A solução é $x = -3$ e $y = 2$.

MatrizCoef tem de ser uma matriz quadrada que contenha os coeficientes das equações.

Resolver:

VectorConst tem de ter o mesmo número de linhas (a mesma dimensão) que *MatrizCoef* e conter as constantes.

Opcionalmente, qualquer elemento da matriz é tratado como zero se o valor absoluto for inferior a *Tol*. Esta tolerância só é utilizada se a matriz tiver entradas de ponto flutuante e não contiver variáveis simbólicas sem um valor atribuído. Caso contrário, *Tol* é ignorado.

- Se definir o modo **Auto** ou **Aproximado** para Aproximado, os cálculos são efectuados com a aritmética de ponto flutuante.
- Se *Tol* for omitido ou não utilizado, a tolerância predefinida for calculada como:
 $5E-14 \cdot \max(\dim(\text{MatrizCoef})) \cdot \text{rowNorm}(\text{MatrizCoef})$

simult(MatrizCoef, MatrizConst [, Tol]) $\Rightarrow$ *matriz*

Resolve vários sistema de equações lineares, em que cada sistema tem os mesmo coeficientes de equações, mas constantes diferentes.

Cada coluna em *MatrizConst* tem de conter as constantes para um sistema de equações. Cada coluna da matriz resultante contém a solução para o sistema correspondente.

$$ax + by = 1$$

$$cx + dy = 2$$

$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow \text{matx1}$	$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$
$\text{simult}\left(\text{matx1}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$	$\begin{bmatrix} -(2 \cdot b - d) \\ a \cdot d - b \cdot c \\ 2 \cdot a - c \\ a \cdot d - b \cdot c \end{bmatrix}$

Resolver:

$$x + 2y = 1$$

$$3x + 4y = -1$$

$$x + 2y = 2$$

$$3x + 4y = -3$$

$\text{simult}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ -1 & -3 \end{bmatrix}\right)$	$\begin{bmatrix} -3 & -7 \\ 2 & \frac{9}{2} \end{bmatrix}$
--	--

Para o primeiro sistema, $x = -3$ e $y = 2$. Para o segundo sistema, $x = -7$ e $y = 9/2$.

Expr **sin**

Nota: Pode introduzir este operador através da escrita de **@>sin** no teclado do computador.

$(\cos(x))^2 \blacktriangleright \text{sin}$	$1 - (\sin(x))^2$
--	-------------------

Representa *Expr* em função do seno.
Este é um operador de conversão.
Apenas pode ser utilizado no fim da linha de entrada.

sin reduz todas as potências de cos(...) módulo 1-seno(...)^2 para que qualquer polinómio residual de potências de seno (...) tenha expoentes no intervalo [0, 2]. Por conseguinte, o resultado sem cos(...) se e só se cos(...) ocorrer na expressão fornecida apenas em potências pares.

Nota: Este operador de conversão não é suportado nos modos de ângulos Graus ou Grados. Antes de o utilizar, certifique-se de que o modo Ângulo está definido para Radianos e que *Expr* não contém referências explícitas a ângulos em graus ou grados.

sin()

Tecla 

sin(*Expr1*) ⇒ *expressão*

sin(*Lista1*) ⇒ *lista*

sin(*Expr1*) devolve o seno do argumento como uma expressão.

sin(*Lista1*) devolve uma lista de senos de todos os elementos em *Lista1*.

Nota: O argumento é interpretado como um ângulo expresso em graus, gradianos ou radianos, de acordo com o modo de ângulo actual. Pode utilizar °, G ou r para substituir a definição do modo de ângulo temporariamente.

No modo de ângulo Graus:

$$\begin{array}{l} \sin\left(\frac{\pi}{4}\right) \quad \frac{\sqrt{2}}{2} \\ \sin(45) \quad \frac{\sqrt{2}}{2} \\ \sin(\{0,60,90\}) \quad \left\{0, \frac{\sqrt{3}}{2}, 1\right\} \end{array}$$

No modo de ângulo Gradianos:

$$\sin(50) \quad \frac{\sqrt{2}}{2}$$

No modo de ângulo Radianos:

$$\begin{array}{l} \sin\left(\frac{\pi}{4}\right) \quad \frac{\sqrt{2}}{2} \\ \sin(45^\circ) \quad \frac{\sqrt{2}}{2} \end{array}$$

sin()

Tecla 

sin(MatrizQuadrada1)
 $\Rightarrow$ MatrizQuadrada

Devolve o seno da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o seno de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Radianos:

$$\sin \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} 0.9424 & -0.04542 & -0.031999 \\ -0.045492 & 0.949254 & -0.020274 \\ -0.048739 & -0.00523 & 0.961051 \end{bmatrix}$$

sin⁻¹()

Tecla 

sin⁻¹(Expr1) $\Rightarrow$ expressão

sin⁻¹(Lista1) $\Rightarrow$ lista

sin⁻¹(Expr1) devolve o ângulo cujo seno é *Expr1* como uma expressão.

sin⁻¹(Lista1) devolve uma lista de senos inversos de cada elemento de *Lista1*.

Nota: O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

Nota: Pode introduzir esta função através da escrita de **arcsin(...)** no teclado do computador.

sin⁻¹(MatrizQuadrada1)
 $\Rightarrow$ MatrizQuadrada

Devolve o seno inverso da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o seno inverso de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Graus:

$$\sin^{-1}(1) = 90$$

No modo de ângulo Gradianos:

$$\sin^{-1}(1) = 100$$

No modo de ângulo Radianos:

$$\sin^{-1}(\{0,0,2,0,5\}) = \{0,0,201358,0,523599\}$$

Nos modos de ângulo Radianos e Formato complexo rectangular:

$$\sin^{-1} \begin{pmatrix} 1 & 5 \\ 4 & 2 \end{pmatrix} = \begin{bmatrix} -0.174533-0.12198 \cdot i & 1.74533-2.35591 \cdot i \\ 1.39626-1.88473 \cdot i & 0.174533-0.593162 \cdot i \end{bmatrix}$$

sinh()**Catálogo >** **sinh**(*Expr1*) $\Rightarrow$ *expressão*

sinh(1.2) 1.50946

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

sinh({0,1.2,3}) {0,1.50946,10.0179}

sinh(*Expr1*) devolve o seno hiperbólico do argumento como uma expressão.**sinh**(*Lista1*) devolve uma lista dos senos hiperbólicos de cada elemento de *Lista1*.**sinh**(*MatrizQuadrada1*) $\Rightarrow$ *MatrizQuadrada*Devolve o seno hiperbólico da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o seno hiperbólico de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.*MatrizQuadrada1* tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Radianos:

sinh	$\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$			
		360.954	305.708	239.604
		352.912	233.495	193.564
		298.632	154.599	140.251

sinh⁻¹()**Catálogo >** **sinh⁻¹**(*Expr1*) $\Rightarrow$ *expressão*sinh⁻¹(0) 0**sinh⁻¹**(*Lista1*) $\Rightarrow$ *lista*sinh⁻¹({0,2.1,3}) {0,1.48748,sinh⁻¹(3)}**sinh⁻¹**(*Expr1*) devolve o seno hiperbólico inverso do argumento como uma expressão.**sinh⁻¹**(*Lista1*) devolve uma lista de senos hiperbólicos inversos de cada elemento de *Lista1*.**Nota:** Pode introduzir esta função através da escrita de **arcsinh(...)** no teclado.**sinh⁻¹**(*MatrizQuadrada1*) $\Rightarrow$ *MatrizQuadrada*Devolve o seno hiperbólico inverso da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o seno hiperbólico inverso de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

No modo de ângulo Radianos:

sinh ⁻¹	$\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$			
		0.041751	2.15557	1.1582
		1.46382	0.926568	0.112557
		2.75079	-1.5283	0.57268

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

SinReg

SinReg *X*, *Y* [, [*Repetições*],[*Ponto*] [, *Categoria*, *Incluir*]]

Calcula a regressão sinusoidal nas listas *X* e *Y*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

X e *Y* são listas de variáveis independentes e dependentes.

Iterações é um valor opcional que especifica o número máximo de vezes (de 1 a 16) que uma solução será tentada. Se for omitido, 8 é utilizado. Em geral, valores maiores resultam numa melhor precisão, mas maiores tempos de execução, e vice-versa.

Período especifica um período previsto. Se for omitido, a diferença entre os valores em *X* deve ser igual e por ordem sequencial. Se especificar *Período*, as diferenças entre os valores *x* podem ser desiguais.

Categoria é uma lista de códigos de categorias para os dados *X* e *Y* correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

A saída de **SinReg** é sempre em radianos, independentemente da definição do modo de ângulo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: $a \cdot \sin(bx+c)+d$
stat.a, stat.b, stat.c, stat.d	Parâmetros de regressão
stat.Resid	Resíduos da regressão
stat.XReg	Lista de dados na <i>Lista X</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.YReg	Lista de dados na <i>Lista Y</i> modificada utilizada na regressão com base nas restrições de <i>Freq</i> , <i>Lista de categorias</i> e <i>Incluir categorias</i>
stat.FreqReg	Lista de frequências correspondentes a <i>stat.XReg</i> e <i>stat.YReg</i>

solve()

Catálogo > 

solve(Equação, Var)⇒Expressão
booleana

solve(Equação,
Var=Hipótese)⇒Expressão booleana

solve(Desigualdade, Var)⇒Expressão
booleana

Apresenta soluções reais candidatas de uma equação ou uma inequação para *Var*. O objectivo é apresentar candidatos para todas as soluções. No entanto, podem existir equações ou inequações para as quais o número de soluções é infinito.

As soluções candidatas podem não ser soluções finitas reais para algumas combinações para variáveis indefinidas.

Para a definição Auto do modo **Auto ou Aproximado**, o objectivo é produzir soluções exactas quando forem concisas, e complementadas pelas procuras iterativas com a aritmética aproximada quando as soluções exactas não forem práticas.

$$\text{solve}(a \cdot x^2 + b \cdot x + c = 0, x)$$

$$x = \frac{\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a} \text{ or } x = \frac{-\left(\sqrt{b^2 - 4 \cdot a \cdot c} + b\right)}{2 \cdot a}$$

$$\text{Ans}|a=1 \text{ and } b=1 \text{ and } c=1$$

$$x = \frac{-1 + \sqrt{3}}{2} \cdot i \text{ or } x = \frac{-1 - \sqrt{3}}{2} \cdot i$$

$$\text{solve}((x-a) \cdot e^x = x \cdot (x-a), x)$$

$$x=a \text{ or } x=-0.567143$$

Devido ao cancelamento predefinido do maior divisor comum do numerador e do denominador de frações, as soluções podem ser soluções apenas limite de um ou ambos os lados.

Para desigualdades de tipos $\geq$, $\leq$, $<$, ou $>$, as soluções explícitas são improváveis, excepto se a desigualdade for linear e só conter *Var*.

Para o modo Exacto, as partes que não podem ser resolvidas são devolvidas como uma desigualdade ou equação implícita.

Utilize o operador de limite ("|") para restringir o intervalo da solução e/ou outras variáveis que ocorram na equação ou na desigualdade. Quando encontrar uma solução num intervalo, pode utilizar os operadores de desigualdade para excluir esse intervalo das procuras subsequentes.

É devolvido falso quando não forem encontradas soluções reais. É devolvido verdadeiro se **solve()** conseguir determinar que qualquer valor real finito de *Var* satisfaz a equação ou a desigualdade.

Como **solve()** devolve sempre um resultado booleano, pode utilizar "and," "or," e "not" para combinar os resultados de **solve()** uns com os outros ou com outras expressões booleanas.

As soluções podem conter uma constante indefinida nova única no formato *nj*, sendo *j* um número inteiro no intervalo 1–255. Essas variáveis indicam um número inteiro arbitrário.

$$(x+1) \cdot \frac{x-1}{x-1} + x - 3 \quad 2 \cdot x - 2$$

$$\text{solve}(5 \cdot x - 2 \geq 2 \cdot x, x) \quad x \geq \frac{2}{3}$$

$$\text{exact}\left(\text{solve}\left((x-a) \cdot e^x = x \cdot (x-a), x\right)\right) \quad e^x + x = 0 \text{ or } x = a$$

No modo de ângulo Radianos:

$$\text{solve}\left(\tan(x) = \frac{1}{x}, x\right) | x > 0 \text{ and } x < 1 \quad x = 0.860334$$

$$\begin{array}{ll} \text{solve}(x = x + 1, x) & \text{false} \\ \text{solve}(x = x, x) & \text{true} \end{array}$$

$$2 \cdot x - 1 \leq 1 \text{ and } \text{solve}(x^2 \neq 9, x) \quad x \neq -3 \text{ and } x \leq 1$$

No modo de ângulo Radianos:

$$\text{solve}(\sin(x) = 0, x) \quad x = n1 \cdot \pi$$

No modo Real, as potências fraccionárias que tenham denominadores ímpares indicam apenas a derivação real. Caso contrário, as várias expressões derivadas, como potências fraccionárias, logaritmos e funções trigonométricas indicam apenas a derivação principal. Por consequência, **solve()** só produz soluções correspondentes a essa derivação principal ou real.

Nota: Consulte também **cSolve()**, **cZeros()**, **nSolve()** e **zeros()**.

solve(*Eqn1* and *Eqn2* [and...],
VarOuHipótese1, *VarOuHipótese2* [, ...
]) ⇒ *Expressão booleana*

solve(*SistemaDeEquações*,
VarOuHipótese1, *VarOuHipótese2* [, ...
]) ⇒ *Expressão booleana*

solve({*Eqn1*, *Eqn2* [...]}
{*VarOuHipótese1*, *VarOuHipótese2* [, ...
]}) ⇒ *Expressão booleana*

Apresenta soluções reais candidatas para equações algébricas simultâneas, em que cada *VarOuHipótese* especifica uma variável que pretenda resolver.

Pode separar as equações com o operador **and** ou pode introduzir um *SistemaDeEquações* com um modelo do Catálogo. O número de argumentos *VarOuHipótese* tem de corresponder ao número de equações. Opcionalmente, pode especificar uma hipótese inicial para uma variável. Cada *VarOuHipótese* tem de ter a forma:

variável

– ou –

variável = *número real ou não real*

Por exemplo, x é válido e, por isso, $x=3$.

$\text{solve}\left(x^{\frac{1}{3}}=1, x\right)$	$x=-1$
$\text{solve}\left(\sqrt{x}=2, x\right)$	false
$\text{solve}\left(\sqrt{x}=-2, x\right)$	$x=4$

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

Se todas as equações forem polinomiais e se não especificar qualquer tentativa inicial, **solve()** utiliza o método de eliminação Gröbner/Buchberger para tentar determinar todas as soluções reais.

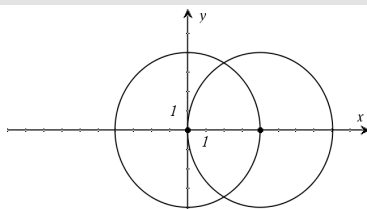
Por exemplo, suponha que tem um círculo de raio r na origem e outro círculo de raio r centrado onde o primeiro círculo cruza o eixo x positivo. Utilize **solve()** para localizar as intersecções.

Como ilustrado pelo r no exemplo à direita, as equações polinomiais simultâneas podem ter variáveis adicionais sem valores, mas representam valores numéricos dados que podem ser substituídos posteriormente.

Pode também (ou em vez de) incluir variáveis da solução que não aparecem nas equações. Por exemplo, pode incluir z como uma variável da solução para expandir o exemplo anterior para dois cilindros de intersecção paralelos de raio r .

As soluções dos cilindros ilustram como as famílias de soluções podem conter constantes arbitrárias da forma $c \cdot k$, em que k é um sufixo com valor inteiro de 1 a 255.

Para sistemas polinomiais, o tempo de cálculo ou o esgotamento da memória podem depender fortemente da ordem em que liste as variáveis das soluções. Se a escolha inicial esgotar a memória ou a sua paciência, tente reorganizar as variáveis nas equações e/ou na lista *varOuTentativa*.



$$\text{solve}\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2, \{x,y\}\right)$$

$$x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3}\cdot r}{2} \text{ or } x=\frac{r}{2} \text{ and } y=-\frac{\sqrt{3}\cdot r}{2}$$

$$\text{solve}\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2, \{x,y,z\}\right)$$

$$x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3}\cdot r}{2} \text{ and } z=c1 \text{ or } x=\frac{r}{2} \text{ and } y=-\frac{\sqrt{3}\cdot r}{2} \text{ and } z=c1$$

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

Se não incluir nenhuma tentativa e se a equação for não polinomial em qualquer variável, mas todas as equações forem lineares em todas as variáveis da solução, **solve()** utiliza a eliminação Gaussiana para tentar determinar todas as soluções.

Se um sistema não for polinomial em todas as variáveis nem linear nas variáveis das soluções, **solve()** determina no máximo uma solução com um método iterativo aproximado. Para o fazer, o número de variáveis de soluções tem de ser igual ao número de equações e todas as outras variáveis nas equações têm de ser simplificadas para números.

Cada variável da solução começa no valor tentado se existir um; caso contrário, começa em 0.0.

Utilize as tentativas para procurar soluções adicionais uma por uma. Para convergência, uma tentativa pode ter de ficar próxima a uma solução.

$$\text{solve}\left(x+e^z \cdot y=1 \text{ and } x-y=\sin(z), \{x,y\}\right)$$

$$x=\frac{e^z \cdot \sin(z)+1}{e^z+1} \text{ and } y=\frac{-(\sin(z)-1)}{e^z+1}$$

$$\text{solve}\left(e^z \cdot y=1 \text{ and } y=\sin(z), \{y,z\}\right)$$

$$y=2.812\text{E-}10 \text{ and } z=21.9911 \text{ or } y=0.001871$$

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

$$\text{solve}\left(e^z \cdot y=1 \text{ and } y=\sin(z), \{y,z=2 \cdot \pi\}\right)$$

$$y=0.001871 \text{ and } z=6.28131$$

SortA

SortA *Lista1* [, *Lista2*] [, *Lista3*] ...

SortA *Vector1* [, *Vector2*] [, *Vector3*]

...

Ordena os elementos do primeiro argumento por ordem crescente.

Se incluir argumentos adicionais, ordena os elementos para que as novas posições correspondam às novas posições dos elementos no primeiro argumento.

Todos os argumentos têm de ter nomes de listas ou vectores. Todos os argumentos têm de ter dimensões iguais.

Os elementos (nulos) vazios do primeiro argumento movem-se para a parte inferior. Para mais informações sobre os elementos vazios, consulte página 274.

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
SortA <i>list1</i>	<i>Done</i>
<i>list1</i>	$\{1,2,3,4\}$
$\{4,3,2,1\} \rightarrow list2$	$\{4,3,2,1\}$
SortA <i>list2,list1</i>	<i>Done</i>
<i>list2</i>	$\{1,2,3,4\}$
<i>list1</i>	$\{4,3,2,1\}$

SortD *Lista1* [, *Lista2*] [, *Lista3*] ...

$\{2,1,4,3\} \rightarrow list1$ $\{2,1,4,3\}$

SortD *Vector1* [, *Vector*] [, *Vector3*] ...

$\{1,2,3,4\} \rightarrow list2$ $\{1,2,3,4\}$

Idêntico a **SortA**, excepto que **SortD** ordena os elementos por ordem decrescente.

SortD *list1,list2* *Done*

list1 $\{4,3,2,1\}$

list2 $\{3,4,1,2\}$

Os elementos (nulos) vazios do primeiro argumento movem-se para a parte inferior. Para mais informações sobre os elementos vazios, consulte página 274.

►Sphere

Vector ►Sphere

Nota: Pode introduzir esta função através da escrita de @>Sphere no teclado.

Apresenta o vector da linha ou coluna em forma esférica [ρ $\angle$ θ $\angle$ ϕ].

O *vector* tem de ser de dimensão 3 e pode ser um vector da linha ou coluna.

Nota: ►Sphere é uma instrução de formato de visualização, não uma função de conversão. Só pode utilizá-la no fim da linha de entrada.

Obs: Para forçar um resultado aproximado,

Unidade portátil: Premir  .

Windows®: Premir **Ctrl+Enter**.

Macintosh®: Premir **⌘+Enter**.

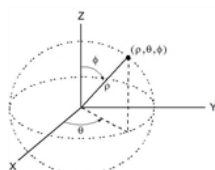
iPad®: Manter pressionada a tecla **Enter** e seleccionar .

$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \text{►Sphere}$
 $\begin{bmatrix} 3.74166 & \angle 1.10715 & \angle 0.640522 \end{bmatrix}$

$\left(2 \angle \frac{\pi}{4} \ 3 \right) \text{►Sphere}$
 $\begin{bmatrix} 3.60555 & \angle 0.785398 & \angle 0.588003 \end{bmatrix}$

Prima 

$\left(2 \angle \frac{\pi}{4} \ 3 \right) \text{►Sphere}$
 $\left[\sqrt{13} \angle \frac{\pi}{4} \angle \sin^{-1} \left(\frac{2 \cdot \sqrt{13}}{13} \right) \right]$



Catálogo >

$\sqrt{4}$
2

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

Para uma lista, devolve as raízes quadradas de todos os elementos em *Lista1*.

Catálogo >

$$xlist:=\{1,2,3,4,5\} \qquad \{1,2,3,4,5\}$$

$$ylist:=\{4,8,11,14,17\} \qquad \{4,8,11,14,17\}$$

```
LinRegMx xlist,ylist,1: stat.results
```

"Title"	"Linear Regression (mx+b)"
"RegEqn"	"m*x+b"
"m"	3.2
"b"	1.2
"r ² "	0.996109
"r"	0.998053
"Resid"	" {... } "

<i>star.values</i>	"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 ²	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 ²	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. $\bar{x}$
stat.b9	stat.FBlock	stat. $\hat{p}$	stat. Σx^2	stat. $\bar{x}1$
stat.b10	stat.Fcol	stat. $\hat{p}1$	stat. Σxy	stat. $\bar{x}2$
stat.bList	stat.FInteract	stat. $\hat{p}2$	stat. Σy	stat. $\bar{x}$ Diff
stat. χ^2	stat.FreqReg	stat. $\hat{p}$ Diff	stat. Σy^2	stat. $\bar{x}$ List
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: Sempre que a aplicação Listas e Folha de Cálculo calcula parâmetros estatísticos, copia as variáveis do grupo “stat.” para um grupo “stat#.”, em que # é um número que é incrementado automaticamente. Isto permite manter os resultados anteriores durante a execução de vários cálculos.

stat.values

Catálogo > 

stat.values

Consulte o exemplo de **stat.results**.

Apresenta uma matriz dos valores calculados para o comando ou a função estatística avaliada mais recentemente.

Ao contrário de **stat.results**, **stat.valu** omite os nomes associados aos valores.

Pode copiar um valor e colá-lo noutras localizações.

stDevPop(*Lista* [, *ListFreq*]) $\Rightarrow$

Devolve o desvio padrão da população dos elementos em *Lista*.

Cada elemento de *ListFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

Nota: *Lista* tem de ter pelo menos dois elementos. Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

stDevPop(*Matriz1* [, *MatrizFreq*])
 $\Rightarrow$ *matriz*

Devolve um vector da linha dos desvios padrão da população das colunas em *Matriz1*.

Cada elemento de *ListaFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Matriz1*.

Nota: *Matriz1* tem de ter pelo menos duas linhas. Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

Nos modos auto e de ângulo Radianos:

$$\text{stDevPop}\left(\left\{a, b, c\right\}\right) = \frac{\sqrt{2 \cdot \left(a^2 - a \cdot (b+c) + b^2 - b \cdot c + c^2\right)}}{3}$$

$$\text{stDevPop}\left(\left\{1, 2, 5, -6, 3, -2\right\}\right) = \frac{\sqrt{465}}{6}$$

$$\text{stDevPop}\left(\left\{1.3, 2.5, -6.4\right\}, \left\{3, 2, 5\right\}\right) = 4.11107$$

$$\text{stDevPop}\left(\begin{pmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{pmatrix}, \begin{pmatrix} 4 \cdot \sqrt{6} & \sqrt{78} & 2 \cdot \sqrt{6} \\ 3 & 3 & 3 \end{pmatrix}\right)$$

$$\text{stDevPop}\left(\begin{pmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{pmatrix}, \begin{pmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{pmatrix}\right) = \begin{bmatrix} 2.52608 & 5.21506 \end{bmatrix}$$

stDevSamp()

stDevSamp(*Lista* [, *ListFreq*])
 $\Rightarrow$ *expressão*

Devolve o desvio padrão da amostra dos elementos em *Lista*.

Cada elemento de *ListFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

Nota: *Lista* tem de ter pelo menos dois elementos. Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

$$\text{stDevSamp}\left(\left\{a, b, c\right\}\right) = \frac{\sqrt{3 \cdot \left(a^2 - a \cdot (b+c) + b^2 - b \cdot c + c^2\right)}}{3}$$

$$\text{stDevSamp}\left(\left\{1, 2, 5, -6, 3, -2\right\}\right) = \frac{\sqrt{62}}{2}$$

$$\text{stDevSamp}\left(\left\{1.3, 2.5, -6.4\right\}, \left\{3, 2, 5\right\}\right) = 4.33345$$

stDevSamp()Catálogo > 

stDevSamp(*Matriz1* [, *MatrizFreq*])
 $\Rightarrow$ *matriz*

Devolve um vector da coluna dos desvios padrão da amostra das colunas em *Matriz1*.

Cada elemento de *ListaFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Matriz1*.

Nota: *Matriz1* tem de ter pelo menos duas linhas. Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

$$\text{stDevSamp}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix}\right) \quad \left[4 \quad \sqrt{13} \quad 2\right]$$

$$\text{stDevSamp}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right) \quad \left[2.7005 \quad 5.44695\right]$$

Stop (Parar)Catálogo > **Stop**

Programar comando: Termina o programa.

Stop não é permitido em funções.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

<i>i</i> :=0	0
Define <i>prog1</i> ()=Prgm	<i>Done</i>
For <i>i</i> ,1,10,1	
If <i>i</i> =5	
Stop	
EndFor	
EndPrgm	
<i>prog1</i> ()	<i>Done</i>
<i>i</i>	5

Store (Guardar)Consulte $\rightarrow$ (guardar), página 255.**string()**Catálogo > 

string(*Expr*) $\Rightarrow$ *cadeia*

Simplifica *Expr* e devolve o resultado como uma cadeia de caracteres.

<i>string</i> (1.2345)	"1.2345"
<i>string</i> (1+2)	"3"
<i>string</i> ($\cos(x) + \sqrt{3}$)	"cos(x)+√(3)"

subMat()Catálogo > 

subMat(*Matriz1* [, *LinhaInicial*] [, *ColInicial*] [, *LinhaFinal*] [, *ColFinal*]) $\Rightarrow$ *matrix*

Devolve a submatriz especificada de *Matriz1*.

Predefinições: *LinhaInicial* =1, *ColInicial* =1, *LinhaFinal* =última linha, *ColFinal* =última coluna.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
$\text{subMat}(m1, 2, 1, 3, 2)$	$\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}$
$\text{subMat}(m1, 2, 2)$	$\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$

Sigma (Soma)Consulte $\Sigma()$, página 245.**sum()**Catálogo > 

sum(*Lista* [, *Início*] [, *Fim*]]) $\Rightarrow$ *expressão*

Devolve a soma dos elementos em *Lista*.

Início e *Fim* são opcionais. Especificam um intervalo de elementos.

Qualquer argumento vazio produz um resultado vazio. Os elementos (nulos) vazios da *Lista* são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

sum(*Matrix1* [, *Início*] [, *Fim*]]) $\Rightarrow$ *matriz*

Devolve um vector da linha com as somas dos elementos nas colunas em *Matrix1*.

Início e *Fim* são opcionais. Especificam um intervalo de linhas.

Qualquer argumento vazio produz um resultado vazio. Os elementos (nulos) vazios da *Matrix1* são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

$\text{sum}(\{1, 2, 3, 4, 5\})$	15
$\text{sum}(\{a, 2 \cdot a, 3 \cdot a\})$	$6 \cdot a$
$\text{sum}(\text{seq}(n, n, 1, 10))$	55
$\text{sum}(\{1, 3, 5, 7, 9\}, 3)$	21

$\text{sum}\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}\right)$	$\begin{bmatrix} 5 & 7 & 9 \end{bmatrix}$
$\text{sum}\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)$	$\begin{bmatrix} 12 & 15 & 18 \end{bmatrix}$
$\text{sum}\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, 2, 3\right)$	$\begin{bmatrix} 11 & 13 & 15 \end{bmatrix}$

sumIf(Lista, Critérios [, ListaDeSomas]) ⇒ *valor*

Devolve a soma acumulada de todos os elementos em *Lista* que satisfazem os *Critérios* especificados. Opcionalmente, pode especificar uma lista alternativa, *ListaDeSomas*, para fornecer os elementos a acumular.

Lista pode ser uma expressão, lista ou matriz. *ListaDeSomas*, se especificada, tem de ter as mesmas dimensões que *Lista*.

Critérios podem ser:

- Um valor, uma expressão ou uma cadeia. Por exemplo, **34** acumula apenas os elementos em *Lista* que são simplificados para o valor 34.
- Uma expressão booleana com o símbolo **?** como um identificador para cada elemento. Por exemplo, **?<10** acumula apenas os elementos em *Lista* que são inferiores a 10.

Quando um elementos da *Lista* cumprir os *Critérios*, o elemento é adicionado à soma acumulada. Se incluir *ListaDeSomas*, o elemento correspondente de *ListaDeSomas* é adicionado à soma.

Na aplicação Listas e Folha de cálculo, pode utilizar um intervalo de células no lugar de *Lista* e de *ListaDeSomas*.

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte página 274.

Nota: Consulte também **countIf()**, página 38.

 $\text{sumIf}(\{1, 2, e, 3, \pi, 4, 5, 6\}, 2.5 < ? < 4.5)$
 $e + \pi + 7$

 $\text{sumIf}(\{1, 2, 3, 4\}, 2 < ? < 5, \{10, 20, 30, 40\})$

70

system()Catálogo > **system**(*Equ1* [, *Equ2* [, *Equ3* [, ...]]])

$\text{solve}\left(\begin{cases} x+y=0 \\ x-y=8 \end{cases}, x, y\right)$	$x=4$ and $y=-4$
---	------------------

system(*Expr1* [, *Expr2* [, *Expr3* [, ...]]])

Devolve um sistema de equações formatado como uma lista. Pode também criar um sistema com um modelo.

Nota: Consulte também **Sistema de equações**, página 3.

T**T (transpor)**Catálogo > *Matriz* **T** $\Rightarrow$ *matriz*

Apresenta a transposta dos conjugados dos complexo da *Matriz1*.

Nota: Pode introduzir este operador através da escrita de **@t** no teclado do computador.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}^T$	$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$
$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^T$	$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$
$\begin{bmatrix} 1+i & 2+i \\ 3+i & 4+i \end{bmatrix}^T$	$\begin{bmatrix} 1-i & 3-i \\ 2-i & 4-i \end{bmatrix}$

tan()Tecla **tan**(*Expr1*) $\Rightarrow$ *expressão*

No modo de ângulo Graus:

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

tan(*Expr1*) devolve a tangente do argumento como uma expressão.

tan(*Lista1*) devolve uma lista das tangentes de todos os elementos em *Lista1*.

Nota: O argumento é interpretado como um ângulo expresso em graus, gradianos ou radianos, de acordo com o modo de ângulo actual. Pode utilizar °, G ou r para substituir a definição do modo de ângulo temporariamente.

$\tan\left(\frac{\pi}{4}\right)$	1
$\tan(45)$	1
$\tan(\{0,60,90\})$	$\{0,\sqrt{3},\text{undef}\}$

No modo de ângulo Gradianos:

$\tan\left(\frac{\pi}{4}\right)$	1
$\tan(50)$	1
$\tan(\{0,50,100\})$	$\{0,1,\text{undef}\}$

No modo de ângulo Radianos:

$$\tan\left(\frac{\pi}{4}\right) \quad 1$$

$$\tan(45^\circ) \quad 1$$

$$\tan\left(\left\{\pi, \frac{\pi}{3}, \pi, \frac{\pi}{4}\right\}\right) \quad \{0, \sqrt{3}, 0, 1\}$$

tan(MatrizQuadradaI) $\Rightarrow$ MatrizQuadrada

Devolve a tangente da matriz de *MatrizQuadradaI*. Isto não é o mesmo que calcular a tangente de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

MatrizQuadradaI tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Radianos:

$$\tan\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right) \quad \begin{bmatrix} -28.2912 & 26.0887 & 11.1142 \\ 12.1171 & -7.83536 & -5.48138 \\ 36.8181 & -32.8063 & -10.4594 \end{bmatrix}$$

tan⁻¹()Tecla **tan⁻¹(ExprI)** $\Rightarrow$ expressão**tan⁻¹(ListaI)** $\Rightarrow$ lista

tan⁻¹(ExprI) devolve o ângulo cuja tangente é *ExprI* como uma expressão.

tan⁻¹(ListaI) devolve uma lista das tangentes inversas de cada elemento de *ListaI*.

Nota: O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

Nota: Pode introduzir esta função através da escrita de **arctan(...)** no teclado.

tan⁻¹(MatrizQuadradaI) $\Rightarrow$ MatrizQuadrada

Devolve a tangente inversa da matriz de *MatrizQuadradaI*. Isto não é o mesmo que calcular a tangente inversa de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

No modo de ângulo Graus:

$$\tan^{-1}(1) \quad 45$$

No modo de ângulo Gradianos:

$$\tan^{-1}(1) \quad 50$$

No modo de ângulo Radianos:

$$\tan^{-1}(\{0, 0.2, 0.5\}) \quad \{0, 0.197396, 0.463648\}$$

No modo de ângulo Radianos:

$$\tan^{-1}\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right) \quad \begin{bmatrix} -0.083658 & 1.26629 & 0.62263 \\ 0.748539 & 0.630015 & -0.070012 \\ 1.68608 & -1.18244 & 0.455126 \end{bmatrix}$$

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

tangentLine()

Catálogo > 

tangentLine
(*Expr1*,*Var*,*Ponto*)⇒*expressão*

tangentLine
(*Expr1*,*Var*=*Ponto*)⇒*expressão*

Apresenta a recta tangente à curva representada por *Expr1* no ponto especificado em *Var*=*Ponto*.

Certifique-se de que a variável independente não está definida. Por exemplo, se $f1(x):=5$ e $x:=3$, então **tangentLine**($f1(x),x,2$) apresenta “falso.”

$\text{tangentLine}(x^2, x, 1)$	$2 \cdot x - 1$
$\text{tangentLine}((x-3)^2 - 4, x=3)$	-4
$\text{tangentLine}\left(x^{\frac{1}{3}}, x=0\right)$	$x=0$
$\text{tangentLine}(\sqrt{x^2 - 4}, x=2)$	undef
$x:=3: \text{tangentLine}(x^2, x, 1)$	5

tanh()

Catálogo > 

tanh(*Expr1*) ⇒*expressão*

tanh(*Lista1*) ⇒*lista*

tanh(*Expr1*) devolve a tangente hiperbólica do argumento como uma expressão.

tanh(*Lista1*) devolve uma lista das tangentes hiperbólicas de cada elemento de *Lista1*.

tanh(*MatrizQuadrada1*)
⇒*MatrizQuadrada*

Devolve a tangente hiperbólica da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular a tangente hiperbólica de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

No modo de ângulo Radianos:

$\text{tanh}\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$	$\begin{bmatrix} -0.097966 & 0.933436 & 0.425972 \\ 0.488147 & 0.538881 & -0.129382 \\ 1.28295 & -1.03425 & 0.428817 \end{bmatrix}$
---	---

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

tanh⁻¹(Expr1) ⇒ expressão

tanh⁻¹(Lista1) ⇒ lista

tanh⁻¹(Expr1) devolve a tangente hiperbólica inversa do argumento como uma expressão.

tanh⁻¹(Lista1) devolve uma lista das tangentes hiperbólicas inversas de cada elemento de *Lista1*.

Nota: Pode introduzir esta função através da escrita de **arctanh**(...) no teclado.

tanh⁻¹(MatrizQuadrada1)
⇒ *MatrizQuadrada*

Devolve a tangente hiperbólica inversa da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular a tangente hiperbólica inversa de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

No Formato complexo rectangular:

$$\begin{array}{l} \text{tanh}^{-1}(0) \\ \hline \text{tanh}^{-1}(\{1, 2, 1, 3\}) \\ \hline \left\{ \text{undef}, 0.518046 - 1.5708 \cdot i, \frac{\ln(2)}{2} - \frac{\pi}{2} \cdot i \right\} \end{array}$$

No modo de ângulo Radianos e Formato complexo rectangular:

$$\begin{array}{l} \text{tanh}^{-1} \left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix} \right) \\ \hline \begin{bmatrix} -0.099353 + 0.164058 \cdot i & 0.267834 - 1.4908 \\ -0.087596 - 0.725533 \cdot i & 0.479679 - 0.94730 \\ 0.511463 - 2.08316 \cdot i & -0.878563 + 1.7901 \end{bmatrix} \end{array}$$

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

taylor(Expr1, Var, Ordem[, Ponto]) ⇒ expressão

Devolve o polinómio de Taylor requerido. O polinómio inclui termos não nulos de graus inteiros de zero a *Ordem* em (*Var* menos *Ponto*). **taylor()** devolve-se se não existir nenhuma série de potência truncada desta ordem ou se quiser expoentes fraccionais ou negativos. Utilize a multiplicação temporária e/ou a substituição por uma potência de (*Var* menos *Ponto*) para determinar séries de potências mais gerais.

Ponto predefine-se para zero e é o ponto de expansão.

$$\begin{array}{l} \text{taylor}(e^{\sqrt{x}}, x, 2) \qquad \text{taylor}(e^{\sqrt{x}}, x, 2, 0) \\ \text{taylor}(e^{t, t, 4}) / t = \sqrt{x} \qquad \frac{3}{x^2} + \frac{x^2}{6} + \frac{x}{2} + \sqrt{x+1} \\ \hline \text{taylor}\left(\frac{1}{x \cdot (x-1)}, x, 3\right) \qquad \text{taylor}\left(\frac{1}{x \cdot (x-1)}, x, 3, 0\right) \\ \hline \text{expand}\left(\frac{\text{taylor}\left(\frac{x}{x \cdot (x-1)}, x, 4\right)}{x}, x\right) \\ \hline -x^3 - x^2 - x - \frac{1}{x} - 1 \end{array}$$

tCdf(*LimiteInferior*, *LimiteSuperior*, *dfs*)
 $\Rightarrow$ número se *LimiteInferior* e *LimiteSuperior* forem números, lista se *LimiteInferior* e *LimiteSuperior* forem listas

Calcula a probabilidade da distribuição Student- *t* entre *LimiteInferior* e *LimiteSuperior* para os graus de liberdade especificados *dfs*.

Para $P(X \leq \text{LimiteSuperior})$, defina *LimiteInferior* = $-\infty$.

tCollect()

tCollect(*Expr1*) $\Rightarrow$ expressão

Devolve uma expressão em que as potências dos números inteiros e produtos de senos e co-senos são convertidos para uma combinação linear de senos e co-senos de vários ângulos, somas de ângulos e diferenças de ângulos. A transformação converte polinómios trigonométricos para uma combinação linear das harmónicas.

Por vezes, **tCollect()** acompanhará os objectivos quando a simplificação trigonométrica predefinida não acompanhar. **tCollect()** trata de transformações inversas efectuadas por **tExpand()**. Por vezes, a aplicação de **tExpand()** num resultado de **tCollect()**, ou vice-versa, em dois passos separados simplifica uma expressão.

$$\begin{array}{l} \text{tCollect}(\{\cos(\alpha)\}^2) \quad \frac{\cos(2 \cdot \alpha) + 1}{2} \\ \text{tCollect}(\sin(\alpha) \cdot \cos(\beta)) \quad \frac{\sin(\alpha - \beta) + \sin(\alpha + \beta)}{2} \end{array}$$

tExpand()

tExpand(*Expr1*) $\Rightarrow$ expressão

$$\begin{array}{l} \text{tExpand}(\sin(3 \cdot \phi)) \quad 4 \cdot \sin(\phi) \cdot \{\cos(\phi)\}^2 - \sin(\phi) \\ \text{tExpand}(\cos(\alpha - \beta)) \quad \cos(\alpha) \cdot \cos(\beta) + \sin(\alpha) \cdot \sin(\beta) \end{array}$$

Devolve uma expressão em que os senos e os co-senos de ângulos de números inteiros, somas de ângulos e diferenças de ângulo são expandidos. Devido à identidade $(\sin(x))^2 + (\cos(x))^2 = 1$, existem muitos resultados equivalentes possíveis. Por consequência, um resultado pode diferir de um resultado apresentado noutras publicações.

Por vezes, **tExpand()** acompanhará os objectivos quando a simplificação trigonométrica predefinida não acompanhar. **tExpand()** trata de transformações inversas efectuadas por **tCollect()**. Por vezes, a aplicação de **tCollect()** num resultado de **tExpand()**, ou vice-versa, em dois passos separados simplifica uma expressão.

Nota: A escala do modo de graus por $\pi/180$ interfere com a capacidade de **tExpand()** para reconhecer as formas expansíveis. Para obter melhores resultados, **tExpand()** deve ser utilizado em modo Radianos.

Text

Text*CadeiaDePedido*[, *MostrarMarcador*]

Programar comando: Interrompe o programa e mostra a cadeia de caracteres *CadeiaDoPedido* numa caixa de diálogo.

Quando o utilizador seleccionar **OK**, a execução do programa continua.

O argumento *marcador* opcional pode ser qualquer expressão.

- Se omitir *MostrarMarcador* e avaliar para **1**, a mensagem de texto é adicionada ao histórico da Calculadora.
- Se *MostrarMarcador* avaliar para **0**, a mensagem de texto não é adicionada ao histórico.

Defina um programa que interrompa a visualização após cinco números aleatórios numa caixa de diálogo.

No modelo `Prgm...EndPrgm`, complete cada linha, premindo  em vez de **enter**. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
Define text_demo()=Prgm
  For i,1,5
    strinfo:="Random number" &
string(rand(i))
    Text strinfo
  EndFor
EndPrgm
```

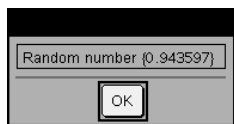

Se o programa necessitar de uma resposta escrita do utilizador, consulte **Request**, página 163, ou **RequestStr**, página 165.

Nota: Pode utilizar este comando num programa definido pelo utilizador, mas não numa função.

Executar o programa:

```
text_demo()
```

Amostra de uma caixa de diálogo:



tInterval *Lista* [, *Freq* [, *NívelC*]]

(Entrada da lista de dados)

tInterval $\bar{x}$, *sx*, *n* [, *NívelC*]

(Entrada estatística do resumo)

Calcula um intervalo de confiança *t*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.CLower, stat.CUpper	Intervalo de confiança para uma média de população desconhecida
stat. $\bar{x}$	Média da amostra da sequência de dados da distribuição aleatória normal
stat.ME	Margem de erro
stat.df	Graus de liberdade
stat. σ_x	Desvio padrão da amostra
stat.n	Comprimento da sequência de dados com a média da amostra

tInterval_2Samp *Lista1, Lista2* [, *Freq1* [, *Freq2* [, *NívelC* [, *Combinado*]]]]

(Entrada da lista de dados)

tInterval_2Samp $\bar{x}1$, $sx1$, $n1$, $\bar{x}2$, $sx2$, $n2$ [, *NívelC* [, *Combinado*]]

(Entrada estatística do resumo)

Calcula um intervalo de confiança t de duas amostras. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Combinado = 1 combina variações;

Combinado = 0 não combina variações.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.CLower, stat.CUpper	Intervalo de confiança com probabilidade da distribuição do nível de confiança
stat. $\bar{x}1$ - $\bar{x}2$	Médias das amostras das sequências de dados da distribuição aleatória normal
stat.ME	Margem de erro
stat.df	Graus de liberdade
stat. $\bar{x}1$, stat. $\bar{x}2$	Médias das amostras das sequências de dados da distribuição aleatória normal
stat. $\sigma x1$, stat. $\sigma x2$	Desvios padrão das amostras para <i>Lista 1</i> e <i>Lista 2</i>
stat. $n1$, stat. $n2$	Número de amostras em sequências de dados
stat.sp	Desvio padrão combinado. Calculado quando <i>Combinado</i> = SIM.

tmpCnv()

tmpCnv(*Expr*, °*UnidTemp*, °*UnidTemp2*) ⇒ *expressão* °*UnidTemp2*

tmpCnv(100·_°C,_°F)	212·_°F
tmpCnv(32·_°F,_°C)	0·_°C
tmpCnv(0·_°C,_°K)	273.15·_°K
tmpCnv(0·_°F,_°R)	459.67·_°R

Converte um valor da temperatura especificado pela *Expr* de uma unidade para a outra. As unidades de temperatura válidas são:

_ °C Celsius

_ °F Fahrenheit

_ °K Kelvin

_ °R Rankine

Para escrever °, seleccione-o nos símbolos do Catálogo.

para escrever _, prima  .

Por exemplo, 100_ °C converte-se para 212_ °F.

Para converter um intervalo de temperatura, utilize Δ tmpCnv().

Nota: Pode utilizar o Catálogo para seleccionar as unidades de temperatura.

Δ tmpCnv()

Δ tmpCnv(*Expr* _ °UnidTemp , _ °UnidTemp2) $\Rightarrow$ expressão _ °UnidTemp2

Nota: Pode introduzir esta função através da escrita de **deltaTmpCnv** (...) no teclado.

Converte um intervalo de temperatura (a diferença entre dois valores de temperatura) especificado pela *Expr* de uma unidade para a outra. As unidades de temperatura válidas são:

_ °CCelsius

_ °FFahrenheit

_ °KKelvin

_ °RRankine

Para introduzir °, seleccione-o na Paleta de símbolos ou escreva @d.

para escrever _, prima  .

Para escrever Δ , seleccione-o nos símbolos do Catálogo.

Δ tmpCnv(100· _ °C, _ °F)	180· _ °F
Δ tmpCnv(180· _ °F, _ °C)	100· _ °C
Δ tmpCnv(100· _ °C, _ °K)	100· _ °K
Δ tmpCnv(100· _ °F, _ °R)	100· _ °R
Δ tmpCnv(1· _ °C, _ °F)	1.8· _ °F

Nota: Pode utilizar o Catálogo para seleccionar as unidades de temperatura.

$\Delta\text{tmpCnv}()$

Catálogo > 

1_ °C e 1_ °K têm a mesma magnitude, como 1_ °F e 1_ °R. No entanto, 1_ °C é tão largo 9/5 como 1_ °F.

Por exemplo, um intervalo de 100_ °C (de 0_ °C a 100_ °C) é equivalente a um intervalo de 180_ °F.

Para converter um valor de temperatura específico em vez de um intervalo, utilize **tmpCnv()**.

tPdf()

Catálogo > 

tPdf(ValX, df) $\Rightarrow$ *número* se *ValX* for um número, *lista* se *ValX* for uma lista

Calcula a função de densidade da probabilidade (pdf) para a distribuição Student- *t* num valor *x* especificado com os graus de liberdade especificados *df*.

trace()

Catálogo > 

trace(MatrizQuadrada) $\Rightarrow$ *expressão*

Apresenta o traço (soma de todos os elementos na diagonal principal) de *MatrizQuadrada*.

$\text{trace}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}\right)$	15
$\text{trace}\left(\begin{pmatrix} a & 0 \\ 1 & a \end{pmatrix}\right)$	$2 \cdot a$

Try

bloco1

Else

bloco2

EndTry

Executa o *bloco1* excepto se ocorrer um erro. A execução do programa transfere-se para *bloco2* se ocorrer um erro em *bloco1*. A variável do sistema *errCode* contém o código de erro para permitir que o programa efectue a recuperação do erro. Para obter uma lista de códigos de erros, consulte "*Mensagens e códigos de erros*", página 284.

bloco1 e *bloco2* podem ser uma única palavra ou uma série de palavras separadas pelo carácter ":".

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

Exemplo 2

Para ver os comandos **Try**, **ClrErr** e **PassErr** na operação, introduza o programa de valores próprios() apresentado à direita. Execute o programa através da execução de cada uma das seguintes expressões.

$$\text{eigenvals}\left(\begin{bmatrix} -3 \\ -41 \\ 5 \end{bmatrix}, \begin{bmatrix} -1 & 2 & -3.1 \end{bmatrix}\right)$$

$$\text{eigenvals}\left(\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$$

Nota: Consulte também **ClrErr**, página 28, e **PassErr**, página 144.

Define *progI()*=Prgm

Try

z:=z+1

Disp "z incremented."

Else

Disp "Sorry, z undefined."

EndTry

EndPrgm

Done

z:=1:progI()

z incremented.

Done

DelVar *z:progI()*

Sorry, z undefined.

Done

Definir valores próprios(a,b)=Prgm

© Os valores próprios do programa(A,B)
mostra os valores próprios de A-B

Ensaio

Disp "A= ",a

Disp "B= ",b

Disp " "

Disp "Valores próprios de A-B são:","eigVl
(a*b)

Else

If *errCode*=230 Then

Disp "Error: Produto de A·B tem de ser uma matriz quadrada"

ClrErr

Else

PassErr

EndIf

EndTry

EndPrgm

tTest

tTest μ_0 , Lista [, Freq [, Hipótese]]

(Entrada da lista de dados)

tTest μ_0 , $\bar{x}$, sx , n , [Hipótese]

(Entrada estatística do resumo)

Efectua um teste da hipótese para uma média da população desconhecida μ quando o desvio padrão da população σ for desconhecido. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Teste $H_0: \mu = \mu_0$, em relação a uma das seguintes:

Para $H_a: \mu < \mu_0$, defina *Hipótese*<0

Para $H_a: \mu \neq \mu_0$ (predefinição), defina *Hipótese*=0

Para $H_a: \mu > \mu_0$, defina *Hipótese*>0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.t	$(\bar{x} - \mu_0) / (stdev / \sqrt{n})$
stat.PVal	Menor nível de significância para o qual a hipótese nula pode ser rejeitada

Variável de saída	Descrição
stat.df	Graus de liberdade
stat.x̄	Média da amostra da sequência de dados em <i>Lista</i>
stat.sx	Desvio padrão da amostra da sequência de dados
stat.n	Tamanho da amostra

tTest_2Samp

Catálogo > 

tTest_2Samp *Lista1, Lista2* [, *Freq1* [, *Freq2* [, *Hipótese* [, *Combinado*]]]]

(Entrada da lista de dados)

tTest_2Samp $\bar{x}1$, *sx1*, *n1*, $\bar{x}2$, *sx2*, *n2* [, *Hipótese* [, *Combinado*]]

(Entrada estatística do resumo)

Calcula um teste *t* de duas amostras. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Teste H_0 : $\mu_1 = \mu_2$, em relação a uma das seguintes:

Para H_a : $\mu_1 < \mu_2$, defina *Hipótese*<0

Para H_a : $\mu_1 \neq \mu_2$ (predefinição), defina *Hipótese*=0

Para H_a : $\mu_1 > \mu_2$, defina *Hipótese*>0

Combinado=1 combina as variâncias

Combinado=0 não combina as variâncias

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.t	Valor normal padrão calculado para a diferença de médias
stat.PVal	Menor nível de significância para o qual a hipótese nula pode ser rejeitada
stat.df	Graus de liberdade para a t-statistic
stat.x̄1, stat.x̄2	Médias da amostra das sequências de dados em <i>Lista 1</i> e <i>Lista 2</i>

Variável de saída	Descrição
stat.sx1, stat.sx2	Desvios padrão da amostra das sequências de dados em <i>Lista 1</i> e <i>Lista 2</i>
stat.n1, stat.n2	Tamanho das amostras
stat.sp	Desvio padrão combinado. Calculado quando <i>Combinado</i> =1.

tvmFV()

Catálogo > 

tvmFV(*N*, *I*, *PV*, *Pmt*, [*PpY*], [*CpY*], [*PmtAt*]) ⇒ *valor*

tvmFV(120,5,0,-500,12,12) 77641.1

Função financeira que calcula o valor futuro do dinheiro.

Nota: Os argumentos utilizados nas funções TVM são descritos na tabela de argumentos TVM, página 213. Consulte também **amortTbI()**, página 8.

tvmI()

Catálogo > 

tvmI(*N*, *PV*, *Pmt*, *FV*, [*PpY*], [*CpY*], [*PmtAt*]) ⇒ *valor*

tvmI(240,100000,-1000,0,12,12) 10.5241

Função financeira que calcula a taxa de juro por ano.

Nota: Os argumentos utilizados nas funções TVM são descritos na tabela de argumentos TVM, página 213. Consulte também **amortTbI()**, página 8.

tvmN()

Catálogo > 

tvmN(*I*, *PV*, *Pmt*, *FV*, [*PpY*], [*CpY*], [*PmtAt*]) ⇒ *valor*

tvmN(5,0,-500,77641,12,12) 120.

Função financeira que calcula o número de períodos de pagamento.

Nota: Os argumentos utilizados nas funções TVM são descritos na tabela de argumentos TVM, página 213. Consulte também **amortTbI()**, página 8.

tvmPmt($N, I, PV, FV, [PpY], [CpY], [PmtAt]$) $\Rightarrow$ valor

tvmPmt(60,4,30000,0,12,12) -552.496

Função financeira que calcula o montante de cada pagamento.

Nota: Os argumentos utilizados nas funções TVM são descritos na tabela de argumentos TVM, página 213. Consulte também **amortTbl()**, página 8.

tvmPV($N, I, Pmt, FV, [PpY], [CpY], [PmtAt]$) $\Rightarrow$ valor

tvmPV(48,4,-500,30000,12,12) -3426.7

Função financeira que calcula o valor actual.

Nota: Os argumentos utilizados nas funções TVM são descritos na tabela de argumentos TVM, página 213. Consulte também **amortTbl()**, página 8.

Argumento TVM*	Descrição	Tipo de dados
N	Número de períodos de pagamento	número real
I	Taxa de juro anual	número real
PV	Valor actual	número real
Pmt	Montante do pagamento	número real
FV	Valor actual	número real
PpY	Pagamentos por ano, predefinição=1	número inteiro > 0
CpY	Períodos compostos por ano, predefinição=1	número inteiro > 0
$PmtAt$	Pagamento devido no fim ou no início de cada período, predefinição=fim	número inteiro (0=fim, 1=início)

* Estes nomes dos argumentos do valor temporal do dinheiro são similares aos nomes das variáveis TVM (como **tvm.pv** e **tvm.pmt**) que são utilizados pelo resolutor financeiro da aplicação *Calculadora*. No entanto, as funções financeiras não guardam os resultados ou os valores dos argumentos nas variáveis TVM.

TwoVar *X*, *Y*, [*Freq*] [, *Categoria*, *Incluir*]]

Calcula a estatística TwoVar. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

X e *Y* são listas de variáveis dependentes e independentes.

Freq é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

Categoria é uma lista de códigos de categorias para os dados *X* e *Y* correspondentes.

Incluir é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são incluídos no cálculo.

Um elemento (nulo) vazio em qualquer das listas *X*, *Freq* ou *Category* resulta num nulo para o elemento correspondente de todas essas listas. Um elemento vazio em qualquer uma das listas de *X1* a *X20* resulta num vazio para o elemento correspondente de todas essas listas. Para mais informações sobre os elementos vazios, consulte página 274.

Variável de saída	Descrição
stat. $\bar{x}$	Média dos valores x
stat. x	Soma dos valores x
stat. x2	Soma de valores x2
stat.sx	Desvio padrão da amostra de x
stat. x	Desvio padrão da população de x
stat.n	Número de pontos de dados

Variável de saída	Descrição
stat. $\bar{y}$	Média de valores y
stat. y	Soma de valores y
stat. y^2	Soma de valores y^2
stat.sy	Desvio padrão da amostra de y
stat. y	Desvio padrão da população de y
stat. xy	Soma de valores $x \cdot y$
stat.r	Coeficiente de correlação
stat.MinX	Mínimo dos valores x
stat.Q ₁ X	1º quartil de x
stat.MedianX	Mediana de x
stat.Q ₃ X	3º quartil de x
stat.MaxX	Máximo de valores x
stat.MinY	Mínimo dos valores y
stat.Q ₁ Y	1º quartil de y
stat.MedY	Mediana de y
stat.Q ₃ Y	3º quartil de y
stat.MaxY	Máximo de valores y
stat. $(x -)^2$	Soma de quadrados de desvios da média de x
stat. $(y -)^2$	Soma de quadrados de desvios da média de y

unitV()

Catálogo >

unitV(*Vector1*) ⇒ *vector*

Devolve um vector unitário da linha ou da coluna na forma de *Vector1*.

Vector1 tem de ser uma matriz de coluna ou linha individual.

$$\text{unitV}\left(\begin{bmatrix} a & b & c \end{bmatrix}\right)$$
$$\left[\frac{a}{\sqrt{a^2+b^2+c^2}} \quad \frac{b}{\sqrt{a^2+b^2+c^2}} \quad \frac{c}{\sqrt{a^2+b^2+c^2}} \right]$$
$$\text{unitV}\left(\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}\right)$$
$$\left[\frac{\sqrt{6}}{6} \quad \frac{\sqrt{6}}{3} \quad \frac{\sqrt{6}}{6} \right]$$
$$\text{unitV}\left(\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}\right)$$
$$\begin{bmatrix} \frac{\sqrt{14}}{14} \\ \frac{14}{\sqrt{14}} \\ \frac{7}{3\sqrt{14}} \\ \frac{14}{14} \end{bmatrix}$$

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

unLock

Catálogo >

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

unLock*Var*.

Desbloqueia as variáveis ou o grupo de variáveis especificadas. Não pode eliminar ou modificar as variáveis bloqueadas.

Consulte **Lock**, página 116, e **getLockInfo** (), página 91.

$a:=65$ 65 $\text{Lock } a$ *Done* $\text{getLockInfo}(a)$ 1 $a:=75$ "Error: Variable is locked." $\text{DelVar } a$ "Error: Variable is locked." $\text{Unlock } a$ *Done* $a:=75$ 75 $\text{DelVar } a$ *Done*

V

varPop()

Catálogo >

varPop(*Lista* [,*ListFreq*]) ⇒ *expressão*

Devolve a variação da população de *Lista*.

Cada elemento de *ListFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

$\text{varPop}(\{5,10,15,20,25,30\})$ 87512Ans: 1.72.9167

216 Lista alfabética

Nota: *Lista* tem de conter pelo menos dois elementos.

Se um elemento numa das listas estiver vazio (nulo), esse elemento é ignorado e o elemento correspondente na outra lista também é ignorado. Para mais informações sobre os elementos vazios, consulte página 274.

varSamp()

varSamp(*Lista* [, *ListaFreq*])
⇒*expressão*

Devolve a variação da amostra de *Lista*.

Cada elemento de *ListaFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

Nota: *Lista* tem de conter pelo menos dois elementos.

Se um elemento numa das listas estiver vazio (nulo), esse elemento é ignorado e o elemento correspondente na outra lista também é ignorado. Para mais informações sobre os elementos vazios, consulte página 274.

varSamp(*Matriz1* [, *MatrizFreq*])
⇒*matriz*

Devolve um vector da coluna com a variação da amostra de cada coluna em *Matriz1*.

Cada elemento de *ListaFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Matriz1*.

Nota: *Matriz1* tem de conter pelo menos duas linhas.

Se um elemento numa das matrizes estiver vazio (nulo), esse elemento é ignorado e o elemento correspondente na outra matriz também é ignorado. Para mais informações sobre os elementos vazios, consulte página 274.

$\text{varSamp}(\{a,b,c\})$ $\frac{a^2 - a \cdot (b+c) + b^2 - b \cdot c + c^2}{3}$	
$\text{varSamp}(\{1,2,5,-6,3,-2\})$	$\frac{31}{2}$
$\text{varSamp}(\{1,3,5\},\{4,6,2\})$	$\frac{68}{33}$

$\text{varSamp}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{bmatrix}\right)$		$\begin{bmatrix} 4.75 & 1.03 & 4 \end{bmatrix}$
$\text{varSamp}\left(\begin{bmatrix} -1.1 & 2.2 \\ 3.4 & 5.1 \\ -2.3 & 4.3 \end{bmatrix}, \begin{bmatrix} 6 & 3 \\ 2 & 4 \\ 5 & 1 \end{bmatrix}\right)$		$\begin{bmatrix} 3.91731 & 2.08411 \end{bmatrix}$

Wait

Catálogo > **Wait** *tempoEmSegundos*

Suspende a execução durante um período de *tempoEmSegundos* segundos.

Wait é particularmente útil num programa que precise de algum tempo para permitir que os dados se tornem disponíveis.

O argumento *tempoEmSegundos* tem de ser uma expressão que se simplifique num valor decimal no intervalo de 0 a 100. O comando arredonda este valor para cima em 0,1 segundos.

Para cancelar uma **Wait** que está em andamento,

- **Dispositivo portátil:** Manter pressionada a tecla  e pressionar  repetidamente.
- **Windows®:** Manter pressionada a tecla **F12** e pressionar **Enter** repetidamente.
- **Macintosh®:** Manter pressionada a tecla **F5** e pressionar **Enter** repetidamente.
- **iPad®:** A aplicação apresenta um pedido. Pode continuar a aguardar ou pode cancelar.

Nota: Pode usar o comando **Wait** dentro de um programa definido pelo utilizador, mas não dentro de uma função.

Para aguardar 4 segundos:

Wait 4

Para aguardar 1/2 segundo:

Wait 0.5

Para aguardar 1,3 segundos usando a variável *seccount*:

seccount:=1.3

Wait seccount

Este exemplo acende um LED verde durante 0,5 segundos e depois, apaga-o.

Send "SET GREEN 1 ON"

Wait 0.5

Send "SET GREEN 1 OFF"

warnCodes ()Catálogo > 

warnCodes(*Expr1*,
StatusVar) $\Rightarrow$ expressão

Avalia a expressão *Expr1*, apresenta o resultado e guarda os códigos de quaisquer avisos gerados na variável da lista *StatusVar*. Se não forem gerados avisos, esta função atribui a *StatusVar* uma lista vazia.



$$\text{warnCodes}\left(\text{solve}\left(\sin(10 \cdot x) = \frac{x^2}{x}, x\right), \text{warn}\right)$$

$x = -0.84232$ or $x = -0.706817$ or $x = -0.2852$

warn	{ 10007, 10009 }
------	------------------

Para ver o resultado completo, prima $\blacktriangle$ e, de seguida, utilize $\blacktriangleleft$ e $\blacktriangleright$ para mover o cursor.

Expr1 pode ser qualquer expressão matemática TI-Nspire™ ou TI-Nspire™ CAS válida. Não pode utilizar um comando ou atribuição como *Expr1*.

StatusVar tem de ser um nome de variável válido.

Para uma lista dos códigos de aviso e mensagens associadas, consulte página 293.

when()

when(*Condição*, *ResultadoVerdadeiro* [, *ResultadoFalso*], *ResultadoDesconhecido*) $\Rightarrow$ *expressão*

Devolve *ResultadoVerdadeiro*, *ResultadoFalso* ou *ResultadoDesconhecido*, dependendo se a *Condição* é verdadeira, falsa ou desconhecida. Devolve a entrada se existirem poucos argumentos para especificar o resultado adequado.

Omite *ResultadoFalso* e *ResultadoDesconhecido* para definir uma expressão apenas na região em que a *Condição* é verdadeira.

Utilize um **undef** *ResultadoFalso* para definir uma expressão representada graficamente apenas num intervalo.

when() é útil para definir funções 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 *Condição*

Bloco

EndWhile

Executa as declarações em *Bloco* desde que *Condição* seja verdadeira.

Bloco pode ser uma declaração ou uma sequência de declarações separadas pelo carácter “.”.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

Define $sum_of_recip(n)=Func$

Local $i,tempsum$

$1 \rightarrow i$

$0 \rightarrow tempsum$

While $i \leq n$

$tempsum + \frac{1}{i} \rightarrow tempsum$

$i + 1 \rightarrow i$

EndWhile

Return $tempsum$

EndFunc

Done

$sum_of_recip(3)$

$\frac{11}{6}$

X

ExprBooleana1 **xor** *ExprBooleana2*
devolve *expressão booleana*

ListaBooleana1 **xor** *ListaBooleana2*
devolve *lista booleana*

MatrizBooleana1 **xor** *MatrizBooleana2*
devolve *matriz booleana*

Devolve verdadeiro se *ExprBooleana1* for verdadeira e *ExprBooleana2* for falsa ou vice-versa.

Devolve falso se ambos os argumentos forem verdadeiros ou falsos. Devolve uma expressão booleana simplificada se não for possível resolver um dos argumentos para verdadeiro ou falso.

Nota: Consulte **or**, xxxx.

NúmeroInteiro1 **xor** *NúmeroInteiro2* $\Rightarrow$ *número inteiro*

true xor true

false

5>3 xor 3>5

true

Importante: Zero, não a letra O.

$0h7AC36 \text{ xor } 0h3D5F$

$0h79169$

Compara dois números inteiros reais bit a bit com uma operação **xor**. Internamente, ambos os números inteiros são convertidos para números binários de 64 bits assinados. Quando os bits correspondentes forem comparados, o resultado é 1 se um dos bits (mas não ambos) for 1; caso contrário, o resultado é 0. O valor devolvido representa os resultados dos bits e aparece de acordo com o modo base.

Pode introduzir os números inteiros em qualquer base numérica. Para uma entrada binária ou hexadecimal, tem de utilizar o prefixo 0b ou 0h, respectivamente. Sem um prefixo, os números inteiros são tratados como decimais (base 10).

Se introduzir um número inteiro na base 10 muito grande para uma forma binária de 64 bits assinada, é utilizada uma operação de módulo simétrico para colocar o valor no intervalo adequado. Para mais informações, consulte ►**Base2**, página 19.

Nota: Consulte **or**, xxxx.

No modo base Bin:

0b100101 xor 0b100	0b100001
--------------------	----------

Nota: Uma entrada binária pode ter até 64 dígitos (não contando com o prefixo 0b). Uma entrada hexadecimal pode ter até 16 dígitos.

Z

zeros()

zeros(*Expr*, *Var*)⇒*lista*

zeros(*Expr*, *Var*=*Hipótese*)⇒*lista*

Apresenta uma lista de valores reais candidatos de *Var* que tornam *Expr*=0. **zeros()** faz isto, calculando **exp**►**lista**(**solve**(*Expr*=0, *Var*), *Var*).

$$\text{zeros}\left(a \cdot x^2 + b \cdot x + c, x\right) = \left\{ \frac{\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a}, \frac{-\left(\sqrt{b^2 - 4 \cdot a \cdot c} + b\right)}{2 \cdot a} \right\}$$

$$a \cdot x^2 + b \cdot x + c | x = \text{Ans}[2] \quad 0$$

Para alguns fins, a forma do resultado para **zeros()** é mais conveniente que a forma de **solve()**. No entanto, a forma do resultado de **zeros()** não pode exprimir soluções implícitas, soluções que requerem desigualdades ou soluções que não envolvam *Var*.

Nota: Consulte também **cSolve()**, **cZeros()** e **solve()**.

zeros{ *Expr1*, *Expr2* },
{*VarOuTentativa1*, *VarOrTentativa2* [,
...] } } $\Rightarrow$ *matriz*

Devolve zeros reais candidatos das expressões algébricas simultâneas, em que cada *VarOrTentativa* especifica um desconhecido cujo valor procura.

Opcionalmente, pode especificar uma tentativa inicial para uma variável. Cada *VarOuTentativa* tem de ter a forma:

variável

– ou –

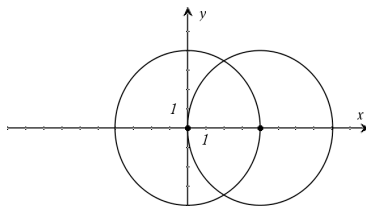
variável = número *real* ou não *real*

Por exemplo, *x* é válido e logo é *x=3*.

Se todas as expressões forem polinomiais e não especificar qualquer tentativa inicial, **zeros()** utiliza o método de eliminação Gröbner/Buchberger lexical para tentar para determinar todos os zeros reais.

Por exemplo, suponha que tem um círculo de raio *r* na origem e outro círculo de raio *r* centrado onde o primeiro círculo cruza o eixo *x* positivo. Utilize **zeros()** para localizar as intersecções.

$$\begin{array}{l} \text{exact}\left(\text{zeros}\left(a \cdot (e^x + x) \cdot (\text{sign}(x) - 1), x\right)\right) \quad \{\} \\ \text{exact}\left(\text{solve}\left(a \cdot (e^x + x) \cdot (\text{sign}(x) - 1) = 0, x\right)\right) \\ e^x + x = 0 \text{ or } x > 0 \text{ or } a = 0 \end{array}$$



Como ilustrado pelo r no exemplo à direita, as expressões polinomiais simultâneas podem ter variáveis adicionais sem valores, mas representam valores numéricos dados que podem ser substituídos posteriormente.

Cada linha da matriz resultante representa um zero alternativo com os componentes ordenados da mesma forma que na lista *VarOuTentativa*. Para extrair uma linha, indexe a matriz por [linha].

Pode também (ou em vez de) incluir variáveis da solução que não aparecem nas expressões. Por exemplo, pode incluir z como um desconhecido para expandir o exemplo anterior para dois cilindros de intersecção paralelos de raio r . Os zeros do cilindro ilustram como as famílias de zeros podem conter constantes arbitrárias na forma ck , em que k é um sufixo com valor inteiro de 1 a 255.

Para sistemas polinomiais, o tempo de cálculo ou o esgotamento da memória podem depender fortemente da ordem em que liste os desconhecidos. Se a escolha inicial esgotar a memória ou a sua paciência, tente reorganizar as variáveis nas expressões e/ou na lista *VarOuTentativa*.

Se não incluir qualquer tentativa ou se qualquer expressão for não polinomial em qualquer variável, mas todas as expressões forem lineares em todos os desconhecidos, **zeros()** utiliza a eliminação Gaussiana para tentar determinar todos os zeros reais.

$$\text{zeros}\left(\left\{x^2+y^2-r^2, (x-r)^2+y^2-r^2\right\}, \{x, y\}\right)$$

$$\begin{bmatrix} r & -\sqrt{3}\cdot r \\ 2 & 2 \\ r & \sqrt{3}\cdot r \\ 2 & 2 \end{bmatrix}$$

Extrair linha 2:

$$\text{Ans}[2] \quad \begin{bmatrix} r & \sqrt{3}\cdot r \\ 2 & 2 \end{bmatrix}$$

$$\text{zeros}\left(\left\{x^2+y^2-r^2, (x-r)^2+y^2-r^2\right\}, \{x, y, z\}\right)$$

$$\begin{bmatrix} r & -\sqrt{3}\cdot r & c1 \\ 2 & 2 & \\ r & \sqrt{3}\cdot r & c1 \\ 2 & 2 & \end{bmatrix}$$

$$\text{zeros}\left(\left\{x+e^z\cdot y-1, x-y-\sin(z)\right\}, \{x, y\}\right)$$

$$\begin{bmatrix} e^z\cdot \sin(z)+1 & -(\sin(z)-1) \\ e^z+1 & e^z+1 \end{bmatrix}$$

Se um sistema não for polinomial em todas as variáveis nem linear nos desconhecidos, **zeros()** determina no máximo um zero com um método iterativo aproximado. Para o fazer, o número de valores desconhecidos tem de ser igual ao número de expressões, e todas as outras variáveis nas expressões têm de ser simplificadas para números.

Cada valor desconhecido começa no valor tentado se existir um; caso contrário, começa em 0.0.

Utilize as tentativas para procurar zeros adicionais um por um. Para convergência, uma tentativa pode ter de ficar próxima a um zero.

zeros($\{ \{ e^z \cdot y - 1, y - \sin(z) \}, \{ y, z \} \}$)	
	$\begin{bmatrix} 0.041458 & 3.18306 \\ 0.001871 & 6.28131 \\ 4.76\text{E-}11 & 1796.99 \\ 2.\text{E-}13 & 254.469 \end{bmatrix}$

zeros($\{ \{ e^z \cdot y - 1, y - \sin(z) \}, \{ y, z = 2 \cdot \pi \} \}$)	
	$\begin{bmatrix} 0.001871 & 6.28131 \end{bmatrix}$

zInterval

zInterval σ , *Lista* [, *Freq* [, *NivelC*]]

(Entrada da lista de dados)

zInterval σ , $\bar{x}$, *n* [, *NivelC*]

(Entrada estatística do resumo)

Calcula um intervalo de confiança *z*. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.CLower, stat.CUpper	Intervalo de confiança para uma média de população desconhecida
stat. $\bar{x}$	Média da amostra da sequência de dados da distribuição aleatória normal
stat.ME	Margem de erro
stat.sx	Desvio padrão da amostra
stat.n	Comprimento da sequência de dados com a média da amostra

Variável de saída	Descrição
stat. σ	Desvio padrão da população conhecido para a sequência de dados <i>Lista</i>

zInterval_1Prop

Catálogo > 

zInterval_1Prop x, n [, *NívelC*]

Calcula um intervalo de confiança z de uma proporção. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

x é um número inteiro não negativo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.CLower, stat.CUpper	Intervalo de confiança com probabilidade da distribuição do nível de confiança
stat. $\hat{p}$	Proporção calculada de sucessos
stat.ME	Margem de erro
stat.n	Número de amostras na sequência de dados

zInterval_2Prop

Catálogo > 

zInterval_2Prop $x1, n1, x2, n2$ [, *NívelC*]

Calcula um intervalo de confiança z de duas proporções. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

$x1$ e $x2$ são números inteiros não negativos.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.CLower, stat.CUpper	Intervalo de confiança com probabilidade da distribuição do nível de confiança
stat. $\hat{p}$ Diff	Diferença calculada entre proporções

Variável de saída	Descrição
stat.ME	Margem de erro
stat. $\hat{p}1$	Primeira previsão da proporção da amostra
stat. $\hat{p}2$	Segunda previsão da proporção da amostra
stat.n1	Tamanho da amostra na sequência de dados um
stat.n2	Tamanho da amostra na sequência de dados dois

zInterval_2Samp

Catálogo > 

zInterval_2Samp $\sigma_1, \sigma_2, Lista1, Lista2$ [,
 Freq1 [, *Freq2*, [*NívelC*]]]

(Entrada da lista de dados)

zInterval_2Samp $\sigma_1, \sigma_2, \bar{x}1, n1, \bar{x}2, n2$ [,
 NívelC]

(Entrada estatística do resumo)

Calcula um intervalo de confiança z de duas amostras. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.CLower, stat.CUpper	Intervalo de confiança com probabilidade da distribuição do nível de confiança
stat. $\bar{x}1 - \bar{x}2$	Médias das amostras das sequências de dados da distribuição aleatória normal
stat.ME	Margem de erro
stat. $\bar{x}1$, stat. $\bar{x}2$	Médias das amostras das sequências de dados da distribuição aleatória normal
stat. $\sigma x1$, stat. $\sigma x2$	Desvios padrão da amostra para <i>Lista 1</i> e <i>Lista 2</i>
stat.n1, stat.n2	Número de amostras em sequências de dados
stat.r1, stat.r2	Desvios padrão da população conhecidos para sequência de dados <i>Lista 1</i> e <i>Lista 2</i>

zTest $\mu_0, \sigma, Lista, [Freq [, Hipótese]]$

(Entrada da lista de dados)

zTest $\mu_0, \sigma, \bar{x}, n [, Hipótese]$

(Entrada estatística do resumo)

Efectua um teste z com a frequência *listfreq*.

Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Teste $H_0: \mu = \mu_0$, em relação a uma das seguintes:

Para $H_a: \mu < \mu_0$, defina *Hipótese*<0

Para $H_a: \mu \neq \mu_0$ (predefinição), defina *Hipótese*=0

Para $H_a: \mu > \mu_0$, defina *Hipótese*>0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.z	$(\bar{x} - \mu_0) / (\sigma / \text{sqrt}(n))$
stat.P Value	Menor probabilidade de rejeição da hipótese nula
stat. $\bar{x}$	Média da amostra da sequência de dados em <i>Lista</i>
stat.sx	Desvio padrão da amostra da sequência de dados. Apenas devolvido para a entrada <i>Dados</i> .
stat.n	Tamanho da amostra

zTest_1Prop

zTest_1Prop $p_0, x, n [, Hipótese]$

Calcula um teste z de uma proporção. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

x é um número inteiro não negativo.

Teste $H_0: p = p_0$ em relação a uma das seguintes:

Para $H_a: p > p_0$, defina *Hipótese*>0

Para $H_a: p \neq p_0$ (*predefinição*), defina *Hipótese*=0

Para $H_a: p < p_0$, defina *Hipótese*<0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.p0	Proporção da população suposta
stat.z	Valor normal padrão calculado para a proporção
stat.PVal	Menor nível de significância para o qual a hipótese nula pode ser rejeitada
stat. $\hat{p}$	Proporção da amostra prevista
stat.n	Tamanho da amostra

zTest_2Prop $x1, n1, x2, n2$ [, *Hipótese*]

Calcula um teste z de duas proporções. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

$x1$ e $x2$ são números inteiros não negativos.

Teste $H_0: p1 = p2$ em relação a uma das seguintes:

Para $H_a: p1 > p2$, defina *Hipótese*>0

Para $H_a: p1 \neq p2$ (*predefinição*), defina *Hipótese*=0

Para $H_a: p < p_0$, defina *Hipótese*<0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.z	Valor normal padrão calculado para a diferença de proporções

Variável de saída	Descrição
stat.PVal	Menor nível de significância para o qual a hipótese nula pode ser rejeitada
stat. $\hat{p}_1$	Primeira previsão da proporção da amostra
stat. $\hat{p}_2$	Segunda previsão da proporção da amostra
stat. $\hat{p}$	Previsão da proporção da amostra combinada
stat.n1, stat.n2	Números de amostras retiradas das tentativas 1 e 2

zTest_2Samp

Catálogo > 

zTest_2Samp $\sigma_1, \sigma_2, Lista1, Lista2$ [, *Freq1* [, *Freq2* [, *Hipótese*]]]

(Entrada da lista de dados)

zTest_2Samp $\sigma_1, \sigma_2, \bar{x}_1, n1, \bar{x}_2, n2$ [, *Hipótese*]

(Entrada estatística do resumo)

Calcula um teste z de duas amostras. Um resumo dos resultados é guardado na variável *stat.results* (página 193).

Teste $H_0: \mu_1 = \mu_2$, em relação a uma das seguintes:

Para $H_a: \mu_1 < \mu_2$, defina *Hipótese*<0

Para $H_a: \mu_1 \neq \mu_2$ (predefinição), defina *Hipótese*=0

Para $H_a: \mu_1 > \mu_2$, *Hipótese*>0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte “Elementos (nulos) vazios” (página 274).

Variável de saída	Descrição
stat.z	Valor normal padrão calculado para a diferença de médias
stat.PVal	Menor nível de significância para o qual a hipótese nula pode ser rejeitada
stat. $\bar{x}_1$, stat. $\bar{x}_2$	Médias das amostras das sequências de dados em <i>Lista1</i> e <i>Lista2</i>
stat.sx1, stat.sx2	Desvios padrão da amostra das sequências de dados em <i>Lista1</i> e <i>Lista2</i>
stat.n1, stat.n2	Tamanho das amostras

Símbolos

+ (adicionar)		Tecla 
$Expr1 + Expr2 \Rightarrow expressão$	56	56
Devolve a soma dos dois argumentos.	56+4	60
	60+4	64
	64+4	68
	68+4	72
$Lista1 + Lista2 \Rightarrow lista$	$\left\{ 22, \pi, \frac{\pi}{2} \right\} \rightarrow l1$	$\left\{ 22, \pi, \frac{\pi}{2} \right\}$
$Matriz1 + Matriz2 \Rightarrow matriz$	$\left\{ 10, 5, \frac{\pi}{2} \right\} \rightarrow l2$	$\left\{ 10, 5, \frac{\pi}{2} \right\}$
Devolve uma lista (ou matriz) com as somas dos elementos correspondentes em <i>Lista1</i> e <i>Lista2</i> (ou <i>Matriz1</i> e <i>Matriz2</i>).	$l1+l2$	$\{ 32, \pi+5, \pi \}$
	$Ans+\{ \pi, -5, \pi \}$	$\{ \pi+32, \pi, 0 \}$
As dimensões dos argumentos têm de ser iguais.	$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} a+1 & b \\ c & d+1 \end{bmatrix}$
$Expr + Lista1 \Rightarrow lista$	$15+\{ 10, 15, 20 \}$	$\{ 25, 30, 35 \}$
$Lista1 + Expr \Rightarrow lista$	$\{ 10, 15, 20 \}+15$	$\{ 25, 30, 35 \}$
Devolve uma lista com as somas de <i>Expr</i> e de cada elemento em <i>Lista1</i> .		
$Expr + Matriz1 \Rightarrow matriz$	$20+\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 21 & 2 \\ 3 & 24 \end{bmatrix}$
$Matriz1 + Expr \Rightarrow matriz$		
Devolve uma matriz com <i>Expr</i> adicionada a cada elemento na diagonal de <i>Matriz1</i> . <i>Matriz1</i> tem de ser quadrada.		
Nota: Utilize .+ (ponto mais) para adicionar uma expressão a cada elemento.		

- (subtrair)		Tecla 
$Expr1 - Expr2 \Rightarrow expressão$	6-2	4
Devolve <i>Expr1</i> menos <i>Expr2</i> .	$\pi - \frac{\pi}{6}$	$\frac{5 \cdot \pi}{6}$
		6

- (subtrair)

Tecla 

$Lista1 - Lista2 \Rightarrow lista$

$$\left\{ 22, \pi, \frac{\pi}{2} \right\} - \left\{ 10, 5, \frac{\pi}{2} \right\} = \left\{ 12, \pi - 5, 0 \right\}$$

$Matriz1 - Matriz2 \Rightarrow matriz$

$$\begin{bmatrix} 3 & 4 \end{bmatrix} - \begin{bmatrix} 1 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 2 \end{bmatrix}$$

Subtrai cada elemento em *Lista2* (ou *Matriz2*) do elemento correspondente em *Lista1* (ou *Matriz1*) e devolve os resultados.

As dimensões dos argumentos têm de ser iguais.

$Expr - Lista1 \Rightarrow lista$

$$15 - \{ 10, 15, 20 \} = \{ 5, 0, -5 \}$$

$Lista1 - Expr \Rightarrow lista$

$$\{ 10, 15, 20 \} - 15 = \{ -5, 0, 5 \}$$

Subtrai cada elemento de *Lista1* de *Expr* ou subtrai *Expr* de cada elemento de *Lista1* e devolve uma lista de resultados.

$Expr - Matriz1 \Rightarrow matriz$

$$20 - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 19 & -2 \\ -3 & 16 \end{bmatrix}$$

$Matriz1 - Expr \Rightarrow matriz$

$Expr - Matriz1$ devolve uma matriz de *Expr* vezes a matriz de identidade menos *Matriz1*. *Matriz1* tem de ser quadrada.

$Matriz1 - Expr$ devolve uma matriz de *Expr* vezes a matriz de identidade subtraída de *Matriz1*. *Matriz1* tem de ser quadrada.

Nota: Utilize .- (ponto menos) para subtrair uma expressão de cada elemento.

· (multiplicar)

Tecla 

$Expr1 \cdot Expr2 \Rightarrow expressão$

$$2 \cdot 3,45 = 6,9$$

Devolve o produto dos dois argumentos.

$$x \cdot y \cdot x = x^2 \cdot y$$

$Lista1 \cdot Lista2 \Rightarrow lista$

$$\{ 1, 2, 3 \} \cdot \{ 4, 5, 6 \} = \{ 4, 10, 18 \}$$

Devolve uma lista com os produtos dos elementos correspondentes em *Lista1* e *Lista2*.

$$\left\{ \frac{2}{a}, \frac{3}{2} \right\} \cdot \left\{ a^2, \frac{b}{3} \right\} = \left\{ 2 \cdot a, \frac{b}{2} \right\}$$

· (multiplicar)

Tecla 

As dimensões das listas têm de ser iguais.

$Matriz1 \cdot Matriz2 \Rightarrow matriz$

Devolve o produto da matriz de $Matriz1$ e $Matriz2$.

O número de colunas em $Matriz1$ tem de ser igual ao número de linhas em $Matriz2$.

$Expr \cdot Lista1 \Rightarrow lista$

$Lista1 \cdot Expr \Rightarrow lista$

Devolve uma lista com os produtos de $Expr$ e de cada elemento em $Lista1$.

$Expr \cdot Matriz1 \Rightarrow matriz$

$Matriz1 \cdot Expr \Rightarrow matriz$

Devolve uma matriz com os produtos de $Expr$ e de cada elemento em $Matriz1$.

Nota: Utilize $\cdot$ (ponto multiplicar) para multiplicar uma expressão por cada elemento.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix} = \begin{bmatrix} a+2 \cdot b+3 \cdot c & d+2 \cdot e+3 \cdot f \\ 4 \cdot a+5 \cdot b+6 \cdot c & 4 \cdot d+5 \cdot e+6 \cdot f \end{bmatrix}$$

$$\pi \cdot \{4, 5, 6\} = \{\pi \cdot 4, \pi \cdot 5, \pi \cdot 6\}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 0.01 = \begin{bmatrix} 0.01 & 0.02 \\ 0.03 & 0.04 \end{bmatrix}$$

$$\lambda \cdot \text{identity}(3) = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix}$$

/ (dividir)

Tecla 

$Expr1 / Expr2 \Rightarrow expressão$

Devolve o quociente de $Expr1$ dividido pela $Expr2$.

Nota: Consulte também **Modelo da fracção**, página 1.

$Lista1 / Lista2 \Rightarrow lista$

Devolve uma lista com os quocientes de $Lista1$ divididos pela $Lista2$.

As dimensões das listas têm de ser iguais.

$Expr / Lista1 \Rightarrow lista$

$Lista1 / Expr \Rightarrow lista$

$$\frac{2}{3.45} = 0.57971$$

$$\frac{x^3}{x} = x^2$$

$$\frac{\{1, 2, 3\}}{\{4, 5, 6\}} = \left\{0.25, \frac{2}{5}, \frac{1}{2}\right\}$$

$$\frac{a}{3, a, \sqrt{a}} = \left\{\frac{a}{3}, 1, \sqrt{a}\right\}$$

$$\frac{\{a, b, c\}}{a \cdot b \cdot c} = \left\{\frac{1}{b \cdot c}, \frac{1}{a \cdot c}, \frac{1}{a \cdot b}\right\}$$

/ (dividir)

Tecla $\div$

Devolve uma lista com os quocientes de *Expr* divididos pela *Lista1* ou de *Lista1* divididos pela *Expr*.

Matriz1 / *Expr* $\Rightarrow$ *matriz*

$$\frac{\begin{bmatrix} a & b & c \end{bmatrix}}{a \cdot b \cdot c} \quad \begin{bmatrix} \frac{1}{b \cdot c} & \frac{1}{a \cdot c} & \frac{1}{a \cdot b} \end{bmatrix}$$

Devolve uma matriz com os quocientes de *Matriz1* / *Expr*.

Nota: Utilize . / (ponto dividir) para dividir uma expressão por cada elemento.

^ (potência)

Tecla $\wedge$

Expr1 ^ *Expr2* $\Rightarrow$ *expressão*

$$4^2 \quad 16$$

Lista1 ^ *Lista2* $\Rightarrow$ *lista*

$$\{a, 2, c\}^{\{1, b, 3\}} \quad \{a, 2^b, c^3\}$$

Devolve o primeiro argumento elevado à potência do segundo argumento.

Nota: Consulte também **Modelo do expoente**, página 1.

Para uma lista, devolve os elementos em *Lista1* elevados à potência dos elementos correspondentes em *Lista2*.

No domínio real, as potências fraccionárias que tenham expoentes simplificados com denominadores ímpares utilizam a derivação real versus a derivação principal para o modo complexo.

Expr ^ *Lista1* $\Rightarrow$ *lista*

$$p^{\{a, 2, 3\}} \quad \left\{p^a, p^2, \frac{1}{p^3}\right\}$$

Devolve *Expr* elevada à potência dos elementos em *Lista1*.

Lista1 ^ *Expr* $\Rightarrow$ *lista*

$$\{1, 2, 3, 4\}^{-2} \quad \left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}\right\}$$

Devolve os elementos em *Lista1* elevados à potência de *Expr*.

^ (potência)**Tecla** $\boxed{\wedge}$

MatrizQuadrada1 $\wedge$ número inteiro $\Rightarrow$ matriz

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^2 \quad \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

Devolve *MatrizQuadrada1* elevada à potência do número inteiro.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} \quad \begin{bmatrix} -2 & 1 \\ 3 & -1 \\ 2 & 2 \end{bmatrix}$$

MatrizQuadrada1 tem de ser uma matriz quadrada.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-2} \quad \begin{bmatrix} 11 & -5 \\ 2 & 2 \\ -15 & 7 \\ 4 & 4 \end{bmatrix}$$

Se número inteiro = -1, calcula a matriz inversa.

Se número inteiro < -1, calcula a matriz inversa para uma potência positiva adequada.

x 2 (quadrado)**Tecla** $\boxed{x^2}$

Expr1 $\wedge 2 \Rightarrow$ expressão

$$4^2 \quad 16$$

Devolve o quadrado do argumento.

$$\{2,4,6\}^2 \quad \{4,16,36\}$$

Lista1 $\wedge 2 \Rightarrow$ lista

$$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}^2 \quad \begin{bmatrix} 40 & 64 & 88 \\ 49 & 79 & 109 \\ 58 & 94 & 130 \end{bmatrix}$$

Devolve uma lista com os quadrados dos elementos em *Lista1*.

$$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix} \cdot \wedge 2 \quad \begin{bmatrix} 4 & 16 & 36 \\ 9 & 25 & 49 \\ 16 & 36 & 64 \end{bmatrix}$$

MatrizQuadrada1 $\wedge 2 \Rightarrow$ matriz

Devolve a matriz quadrada de *MatrizQuadrada1*. Isto não é o mesmo que calcular o quadrado de cada elemento. Utilize $\cdot \wedge 2$ para calcular o quadrado de cada elemento.

.+ (ponto adicionar)**Teclas** $\boxed{\cdot} \boxed{+}$

Matriz1 $\cdot +$ *Matriz2* $\Rightarrow$ matriz

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} \cdot + \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \quad \begin{bmatrix} a+c & 6 \\ b+5 & d+3 \end{bmatrix}$$

Expr $\cdot +$ *Matriz1* $\Rightarrow$ matriz

$$x \cdot + \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \quad \begin{bmatrix} x+c & x+4 \\ x+5 & x+d \end{bmatrix}$$

Matriz1 $\cdot +$ *Matriz2* devolve uma matriz que é a soma de cada par dos elementos correspondentes em *Matriz1* e *Matriz2*.

Expr $\cdot +$ *Matriz1* devolve uma matriz que é a soma de *Expr* e de cada elemento em *Matriz1*.

.- (ponto subtração)

Teclas $\boxed{-}$ $\boxed{-}$

$Matriz1 \text{ .- } Matriz2 \Rightarrow \text{matriz}$

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} \text{ .- } \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \Rightarrow \begin{bmatrix} a-c & -2 \\ b-d & -2 \end{bmatrix}$$

$Expr \text{ .- } Matriz1 \Rightarrow \text{matriz}$

$$x \text{ .- } \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \Rightarrow \begin{bmatrix} x-c & x-4 \\ x-d & x-5 \end{bmatrix}$$

$Matriz1 \text{ .- } Matriz2$ devolve uma matriz que é a diferença entre cada par de elementos correspondentes em $Matriz1$ e $Matriz2$.

$Expr \text{ .- } Matriz1$ devolve uma matriz que é a diferença de $Expr$ e de cada elemento em $Matriz1$.

.· (ponto mult.)

Teclas $\boxed{\cdot}$ $\boxed{\times}$

$Matriz1 \text{ .} \cdot Matriz2 \Rightarrow \text{matriz}$

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} \text{ .} \cdot \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \Rightarrow \begin{bmatrix} a \cdot c & 8 \\ 5 \cdot b & 3 \cdot d \end{bmatrix}$$

$Expr \text{ .} \cdot Matriz1 \Rightarrow \text{matriz}$

$$x \text{ .} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} \Rightarrow \begin{bmatrix} a \cdot x & b \cdot x \\ c \cdot x & d \cdot x \end{bmatrix}$$

$Matriz1 \text{ .} \cdot Matriz2$ devolve uma matriz que é o produto de cada par dos elementos correspondentes em $Matriz1$ e $Matriz2$.

$Expr \text{ .} \cdot Matriz1$ devolve uma matriz com os produtos de $Expr$ e de cada elemento em $Matriz1$.

./ (ponto dividir)

Teclas $\boxed{\div}$ $\boxed{\div}$

$Matriz1 \text{ .} / Matriz2 \Rightarrow \text{matriz}$

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} \text{ .} / \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{a}{c} & \frac{1}{2} \\ \frac{b}{5} & \frac{3}{d} \end{bmatrix}$$

$Expr \text{ .} / Matriz1 \Rightarrow \text{matriz}$

$$x \text{ .} / \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{x}{c} & \frac{x}{4} \\ \frac{x}{5} & \frac{x}{d} \end{bmatrix}$$

$Matriz1 \text{ .} / Matriz2$ devolve uma matriz que é o quociente de cada par de elementos correspondente em $Matriz1$ e $Matriz2$.

$Expr \text{ .} / Matriz1$ devolve uma matriz que é o quociente de $Expr$ e de cada elemento em $Matriz1$.

.^ (ponto potência)

Teclas  

$$Matriz1 \wedge Matriz2 \Rightarrow matriz$$

a	2	$\cdot$	c	4		a^c	16
b	3		5	d		5	d

$$Expr . ^{Matrix} l \Rightarrow matrix$$

Matriz1 .^ *Matriz2* devolve uma matriz em que cada elemento em *Matriz2* é o expoente para o elemento correspondente em *Matriz1*.

$$x \cdot \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} = \begin{bmatrix} x^c & x^4 \\ x^5 & x^d \end{bmatrix}$$

`Expr.^MatrizI` devolve uma matriz em que cada elemento em *MatrizI* é o expoente para *Expr*.

- (negação)

Teccla

-*Expr1* $\Rightarrow$ expressão

-2.43	-2.43
-------	-------

$$\neg List a \Rightarrow list a$$

$$-\{-1, 0.4, 1.2\text{E}19\} \quad \{1, -0.4, -1.2\text{E}19\}$$

-Matrizl $\Rightarrow$ matriz

$$-a \cdot -b \qquad a \cdot b$$

Devolve a negação do argumento.

Para uma lista ou matriz, devolve todos os elementos negados.

No modo base Bin:

Importante: Zero, não a letra O

Se o argumento for um número inteiro binário ou hexadecimal, a negação dá o complemento de dois.

```
-0b100101  
0b11111111111111111111111111111111▶
```

Para ver o resultado completo, prima **▲** e, de seguida, utilize **◀** e **▶** para mover o cursor.

% (percentagem)

Teclas  

Expr1 % $\Rightarrow$ expressão

Obs: Para forçar um resultado aproximado.

$$Listal \% \Rightarrow lista$$

Unidade portátil: Premir ctrl enter.

$$Matriz1 \% \Rightarrow matriz$$

Windows®: Premir Ctrl+Enter.

Macintosh®: Premir ⌘+Enter.

iPad®: Manter pressionada a tecla **Enter** e selecionar .

argument

Devolve	100
---------	-----

13%	0.13
-----	------

Para uma lista ou matriz, devolve uma lista ou matriz com cada elemento dividido por 100.

$$\{\{1,10,100\}\}\% \qquad \{0.01,0.1,1.\}$$

= (igual)**Tecla**  $Expr1 = Expr2 \Rightarrow \text{Expressão booleana}$ $Lista1 = Lista2 \Rightarrow \text{Lista booleana}$ $Matriz1 = Matriz2 \Rightarrow \text{Matriz booleana}$

Devolve verdadeiro se *Expr1* for determinada para ser igual a *Expr2*.

Devolve falso se *Expr1* for determinada para ser diferente a *Expr2*.

Outra coisa qualquer devolve uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

Exemplo de função que utiliza os símbolos de teste matemático: =, ≠, <, ≤, >, ≥

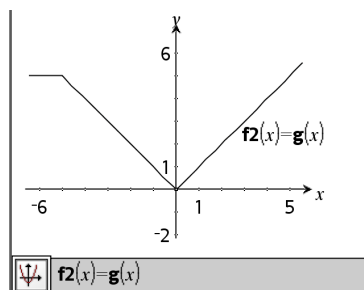
```

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 do gráfico $g(x)$

**≠ (diferente)****Teclas**   $Expr1 \neq Expr2 \Rightarrow \text{Expressão booleana}$ $Lista1 \neq Lista2 \Rightarrow \text{Lista booleana}$ $Matriz1 \neq Matriz2 \Rightarrow \text{Matriz booleana}$

Devolve verdadeiro se *Expr1* for determinada para ser diferente a *Expr2*.

Devolve falso se *Expr1* for determinada para ser igual a *Expr2*.

Outra coisa qualquer devolve uma forma simplificada da equação.

Consulte exemplo “=” (igual).

$\neq$ (diferente)

Teclas  

Para listas e matrizes, devolve comparações elemento por elemento.

Nota: Pode introduzir este operador através da escrita de $\neq$ no teclado.

$<$ (menor que)

Teclas  

$Expr1 < Expr2 \Rightarrow Express\tilde{a}o\ booleana$

Consulte exemplo “=” (igual).

$Lista1 < Lista2 \Rightarrow Lista\ booleana$

$Matriz1 < Matriz2 \Rightarrow Matriz\ booleana$

Devolve verdadeiro se $Expr1$ for determinada para ser menor que $Expr2$.

Devolve falso se $Expr1$ for determinada para ser igual ou maior que $Expr2$.

Outra coisa qualquer devolve uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

$\leq$ (igual ou menor que)

Teclas  

$Expr1 \leq Expr2 \Rightarrow Express\tilde{a}o\ booleana$

Consulte exemplo “=” (igual).

$Lista1 \leq Lista2 \Rightarrow Lista\ booleana$

$Matriz1 \leq Matriz2 \Rightarrow Matriz\ booleana$

Devolve verdadeiro se $Expr1$ for determinada para igual ou menor que $Expr2$.

Devolve falso se $Expr1$ for determinada para ser maior que $Expr2$.

Outra coisa qualquer devolve uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

Nota: Pode introduzir este operador através da escrita de $\leq$ no teclado

> (maior que)

Teclas  

$Expr1 > Expr2 \Rightarrow \text{Expressão booleana}$

Consulte exemplo “=” (igual).

$Lista1 > Lista2 \Rightarrow \text{Lista booleana}$

$Matriz1 > Matriz2 \Rightarrow \text{Matriz booleana}$

Devolve verdadeiro se $Expr1$ for determinada para ser maior que $Expr2$.

Devolve falso se $Expr1$ for determinada para ser igual ou menor que $Expr2$.

Outra coisa qualquer devolve uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

$\geq$ (igual ou maior que)

Teclas  

$Expr1 \geq Expr2 \Rightarrow \text{Expressão booleana}$

Consulte exemplo “=” (igual).

$Lista1 \geq Lista2 \Rightarrow \text{Lista booleana}$

$Matriz1 \geq Matriz2 \Rightarrow \text{Matriz booleana}$

Devolve verdadeiro se $Expr1$ for determinada para ser igual ou maior que $Expr2$.

Devolve falso se $Expr1$ for determinada para ser menor que $Expr2$.

Outra coisa qualquer devolve uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

Nota: Pode introduzir este operador através da escrita de $>=$ no teclado.

⇒ (implicação lógica)Teclas **ctrl** **=***ExprBooleana1 ⇒ ExprBooleana2*
devolve expressão booleana

5>3 or 3>5 true

ListaBooleana1 ⇒ ListaBooleana2
devolve lista booleana

5>3 ⇒ 3>5 false

3 or 4 7

MatrizBooleana1 ⇒ MatrizBooleana2
devolve matriz booleana

3 ⇒ 4 -4

{1,2,3} or {3,2,1} {3,2,3}

{1,2,3} ⇒ {3,2,1} {-1,-1,-3}

NúmeroInteiro1 ⇒ NúmeroInteiro2
devolve número inteiro

Avalia a expressão **not <argumento1> or <argumento2>** e devolve falso, verdadeiro ou uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

Nota: Pode introduzir este operador ao escrever => com o teclado

⇔ (implicação lógica dupla, XNOR)Teclas **ctrl** **=***ExprBooleana1 ⇔ ExprBooleana2*
devolve expressão booleana

5>3 xor 3>5 true

ListaBooleana1 ⇔ ListaBooleana2
devolve lista booleana

5>3 ⇔ 3>5 false

3 xor 4 7

MatrizBooleana1 ⇔ MatrizBooleana2
devolve matriz booleana

3 ⇔ 4 -8

{1,2,3} xor {3,2,1} {2,0,2}

{1,2,3} ⇔ {3,2,1} {-3,-1,-3}

NúmeroInteiro1 ⇔ NúmeroInteiro2
devolve número inteiro

Devolve a negação de uma operação booleana **XOR** nos dois argumentos. Devolve falso, verdadeiro ou uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

$\Leftrightarrow$ (implicação lógica dupla, XNOR)

Teclas  

Nota: Pode introduzir este operador ao escrever $\Leftrightarrow$ com o teclado

! (factorial)

Tecla 

Expr1! $\Rightarrow$ expressão

5! 120

Lista1! $\Rightarrow$ lista

$\{\{5,4,3\}\}!$ $\{120,24,6\}$

Matriz1! $\Rightarrow$ matriz

$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}!$ $\begin{bmatrix} 1 & 2 \\ 6 & 24 \end{bmatrix}$

Devolve o factorial do argumento.

Para uma lista ou matriz, devolve uma lista ou matriz de factoriais dos elementos.

& (acrescentar)

Teclas  

Cadeia1 & *Cadeia2* $\Rightarrow$ cadeia

"Hello "&"Nick" "Hello Nick"

Devolve uma cadeia de texto que é *Cadeia2* acrescentada a *Cadeia1*.

d() (derivada)

Catálogo > 

d(*Expr1*, *Var*[, *Ordem*]) $\Rightarrow$ expressão

$\frac{d}{dx}(f(x) \cdot g(x))$ $\frac{d}{dx}(f(x)) \cdot g(x) + \frac{d}{dx}(g(x)) \cdot f(x)$

d(*Lista1*, *Var*[, *Ordem*]) $\Rightarrow$ lista

$\frac{d}{dy}\left(\frac{d}{dx}(x^2 \cdot y^3)\right)$ $6 \cdot y^2 \cdot x$

d(*Matriz1*, *Var*[, *Ordem*]) $\Rightarrow$ matriz

Devolve a primeira derivada do primeiro argumento em relação à variável *Var*.

$\frac{d}{dx}\left(\left\{x^2, x^3, x^4\right\}\right)$ $\left\{2 \cdot x, 3 \cdot x^2, 4 \cdot x^3\right\}$

Ordem, se incluída, tem de ser um número inteiro. Se a ordem for inferior a zero, o resultado será uma antiderivada.

Nota: Pode introduzir isto através da escrita de **derivada (...)** no teclado.

d() não segue o mecanismo de avaliação normal, simplificando completamente os argumentos e aplicando a definição da função para estes argumentos completamente simplificados. Em vez disso, **d()** efectue os seguintes passos:

1. Simplifique o segundo argumento apenas até ao ponto de não originar a uma não variável.
2. Simplifique o primeiro argumento até ao ponto de rechamar qualquer valor guardado para a variável determinada pelo passo 1.
3. Determine a derivada simbólica do resultado do passo 2 em relação à variável do passo 1.

Se a variável do passo 1 possuir um valor guardado ou especificado com um operador de limite ("|"), substitua esse valor pelo resultado do passo 3.

Nota: Consulte também

Primeira derivada, página 5;

Segunda derivada, página 6; ou **derivada**

Nth, página 6.

∫() (integrar)

$\int(\text{Expr1}, \text{Var}[, \text{Inferior}, \text{Superior}]) \Rightarrow$
expressão

$\int(\text{Expr1}, \text{Var}[, \text{Constante}]) \Rightarrow$ expressão

$$\int_a^b x^2 dx = \frac{b^3}{3} - \frac{a^3}{3}$$

Devolve o integral de *Expr1* em relação à variável *Var* de *Inferior* a *Superior*.

Nota: Consulte também o modelo de integral **definido** ou **indefinido**, página 6.

Nota: Pode introduzir esta função através do teclado, escrevendo **integral (...)**.

Devolve uma antiderivada se *Inferior* e *Superior* forem omitidos. Uma constante simbólica de integração é omitida, excepto se fornecer o argumento *Constante*.

$$\int x^2 dx \quad \frac{x^3}{3}$$

$$\int(a \cdot x^2, x, c) \quad \frac{a \cdot x^3}{3} + c$$

As primitivas igualmente válidas podem diferir por uma constante numérica. Essa constante pode estar disfarçada—em especial, quando uma primitiva contiver logaritmos ou funções trigonométricas inversas. Além disso, as expressões constantes piecewise são por vezes adicionadas para validar uma primitiva sobre um intervalo maior que a fórmula usual.

∫() devolve-se por partes de *Expr1* que não pode ser determinada como uma combinação finita explícita dos operadores e das funções integrados.

$$\int b \cdot e^{-x^2} + \frac{a}{x^2 + a^2} dx \quad b \cdot \int e^{-x^2} dx + \tan^{-1}\left(\frac{x}{a}\right)$$

Quando fornecer *Inferior* e *Superior*, é efectuada uma tentativa para localizar qualquer descontinuidade ou derivada descontinua no intervalo *Inferior* < *Var* < *Superior* e subdividir o intervalo nesses locais.

Para a definição Auto do modo **Auto** ou **Aproximado**, a integração numérica é utilizada onde aplicável quando não for possível determinar uma primitiva ou um limite.

Para a definição Aproximado, a integração numérica é tentada primeiro, se aplicável. As primitivas são procuradas apenas onde essa integração numérica não seja aplicável ou falhar.

Obs: Para forçar um resultado aproximado,

Unidade portátil: Premir  .

Windows®: Premir **Ctrl+Enter**.

Macintosh®: Premir +**Enter**.

iPad®: Manter pressionada a tecla **Enter** e seleccionar .

$$\int_{-1}^1 e^{-x^2} dx \quad 1.49365$$

$\int()$ (integrar)

Catálogo > 

$\int()$ pode ser aninhada para fazer vários integrais. Os limites da integração podem depender das variáveis de integração fora dos limites.

Nota: Consulte também **nInt()**, página 134.

$$\int_0^a \int_0^x \ln(x+y) \, dy \, dx$$

$$\frac{a^2 \cdot \ln(a)}{2} + \frac{a^2 \cdot (4 \cdot \ln(2) - 3)}{4}$$

$\sqrt{()}$ (raiz quadrada)

Teclas  

$\sqrt{(\text{Expr1})} \Rightarrow \text{expressão}$

$$\sqrt{4} \quad 2$$

$\sqrt{(\text{Lista1})} \Rightarrow \text{lista}$

$$\sqrt{\{9, a, 4\}} \quad \{3, \sqrt{a}, 2\}$$

Devolve a raiz quadrada do argumento.

Para uma lista, devolve as raízes quadradas de todos os elementos em *Lista1*.

Nota: Pode introduzir esta função através da escrita de **sqrt (...)** no teclado

Nota: Consulte também **Modelo de raiz quadrada**, página 1.

$\Pi()$ (prodSeq)

Catálogo > 

$\Pi(\text{Expr1}, \text{Var}, \text{Baixo}, \text{Alto})$
 $\Rightarrow \text{expressão}$

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

Nota: Pode introduzir esta função através da escrita de **prodSeq (...)** no teclado.

$$\prod_{k=1}^n (k^2) \quad (n!)^2$$

Avalia *Expr1* para cada valor de *Var* de *Baixo* a *Alto* e devolve o produto dos resultados.

$$\prod_{n=1}^5 \left(\left\{ \frac{1}{n}, n, 2 \right\} \right) \quad \left\{ \frac{1}{120}, 120, 32 \right\}$$

Nota: Consulte também **Modelo do produto (Π)**, página 5.

$\Pi(\text{Expr1}, \text{Var}, \text{Baixo}, \text{Baixo} - 1) \Rightarrow 1$

$$\prod_{k=4}^3 (k) \quad 1$$

$\Pi(\text{Expr1}, \text{Var}, \text{Baixo}, \text{Alto}) \Rightarrow 1 / \Pi(\text{Expr1}, \text{Var}, \text{Alto} + 1, \text{Baixo} - 1)$ se *Alto* < *Baixo* - 1

As fórmulas do produto utilizadas derivam da seguinte referência:

Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley, 1994.

$$\prod_{k=4}^1 \left(\frac{1}{k} \right) \quad 6$$

$$\prod_{k=4}^1 \left(\frac{1}{k} \right) \cdot \prod_{k=2}^4 \left(\frac{1}{k} \right) \quad \frac{1}{4}$$

$\Sigma()$ (sumSeq)

$\Sigma(Expr1, Var, Baixo, Alto) \Rightarrow$ expressão

Nota: Pode introduzir esta função através da escrita de **sumSeq (...)** no teclado.

Avalia *Expr1* para cada valor de *Var* de *Baixo* a *Alto* e devolve a soma dos resultados.

Nota: Consulte também **Modelo da soma**, página 5.

$\Sigma(Expr1, Var, Baixo, Baixo - 1) \Rightarrow 0$

$\Sigma(Expr1, Var, Baixo, Alto) \Rightarrow -\Sigma(Expr1, Var, Alto+1, Baixo - 1)$ se $Alto < Baixo - 1$

As fórmulas da soma utilizadas derivam da seguinte referência :

Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley, 1994.

$$\sum_{n=1}^5 \left(\frac{1}{n} \right) \quad \frac{137}{60}$$

$$\sum_{k=1}^n (k^2) \quad \frac{n \cdot (n+1) \cdot (2 \cdot n+1)}{6}$$

$$\sum_{n=1}^{\infty} \left(\frac{1}{n^2} \right) \quad \frac{\pi^2}{6}$$

$$\sum_{k=4}^3 (k) \quad 0$$

$$\sum_{k=4}^1 (k) \quad -5$$

$$\sum_{k=4}^1 (k) + \sum_{k=2}^4 (k) \quad 4$$

$\Sigma Int()$

$\Sigma Int(NPmt1, NPmt2, N, I, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [ValorArredondado]) \Rightarrow valor$

$$\Sigma Int(1, 3, 12, 4, 75, 20000, , 12, 12) \quad -218.11$$

ΣInt(NPmt1, NPmt2, TabelaDeDepreciação) ⇒ valor

Função de amortização que calcula a soma do juro durante um intervalo especificado de pagamentos.

NPmt1 e NPmt2 definem os limites iniciais e finais do intervalo de pagamentos.

N, I, PV, Pmt, FV, PpY, CpY e PmtAt são descritos na tabela de argumentos TVM, página 213.

- Se omitir Pmt, predefine-se para Pmt = tvmpmt(N, I, PV, FV, PpY, CpY, PmtAt).
- Se omitir FV, predefine-se para FV = 0.
- As predefinições para PpY, CpY e PmtAt são iguais às predefinições para as funções TVM.

ValorArredondado especifica o número de casas decimais para arredondamento. Predefinição=2.

ΣInt(NPmt1, NPmt2, TabelaDeDepreciação) calcula a soma dos juros com base na tabela de amortização TabelaDeDepreciação. O argumento TabelaDeDepreciação tem de ser uma matriz na forma descrita em amortTbl(), página 8.

Nota: Consulte também ΣPrn(), abaixo, e Bal(), página 18.

tbl:=amortTbl(12,12,4.75,20000,,,12,12)				
0	0.	0.	20000.	
1	-79.17	-1630.69	18369.3	
2	-72.71	-1637.15	16732.2	
3	-66.23	-1643.63	15088.5	
4	-59.73	-1650.13	13438.4	
5	-53.19	-1656.67	11781.7	
6	-46.64	-1663.22	10118.5	
7	-40.05	-1669.81	8448.7	
8	-33.44	-1676.42	6772.28	
9	-26.81	-1683.05	5089.23	
10	-20.14	-1689.72	3399.51	
11	-13.46	-1696.4	1703.11	
12	-6.74	-1703.12	-0.01	
ΣInt(1,3,tbl)				-218.11

ΣPrn(NPmt1, NPmt2, N, I, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [ValorArredondado]) ⇒ valor

ΣPrn(NPmt1, NPmt2, TabelaDeDepreciação) ⇒ valor

Função de amortização que calcula a soma do capital durante um intervalo especificado de pagamentos.

ΣPrn(1,3,12,4.75,20000,,,12,12)	-4911.47
---------------------------------	----------

NPmt1 e *NPmt2* definem os limites iniciais e finais do intervalo de pagamentos.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* e *PmtAt* são descritos na tabela de argumentos TVM, página 213.

- Se omitir *Pmt*, predefine-se para *Pmt* = **tvmPmt**(*N*, *I*, *PV*, *FV*, *PpY*, *CpY*, *PmtAt*).
- Se omitir *FV*, predefine-se para *FV* = 0.
- As predefinições para *PpY*, *CpY* e *PmtAt* são iguais às predefinições para as funções TVM.

ValorArredondado especifica o número de casas decimais para arredondamento. Predefinição=2.

ΣPrn(*NPmt1*, *NPmt2*, *TabelaDeDepreciação*) calcula a soma do capital pago com base na tabela de amortização *TabelaDeDepreciação*. O argumento *TabelaDeDepreciação* tem de ser uma matriz na forma descrita em **amortTbl()**, página 8.

Nota: Consulte também **ΣInt()**, acima, e **Bal()**, página 18.

<i>tbl:=amortTbl</i> (12,12,4.75,20000,,,12,12)				
	0	0.	0.	20000.
1	-79.17	-1630.69	18369.3	
2	-72.71	-1637.15	16732.2	
3	-66.23	-1643.63	15088.5	
4	-59.73	-1650.13	13438.4	
5	-53.19	-1656.67	11781.7	
6	-46.64	-1663.22	10118.5	
7	-40.05	-1669.81	8448.7	
8	-33.44	-1676.42	6772.28	
9	-26.81	-1683.05	5089.23	
10	-20.14	-1689.72	3399.51	
11	-13.46	-1696.4	1703.11	
12	-6.74	-1703.12	-0.01	
$\Sigma Prn(1,3,tbl)$				-4911.47

(indirecta)

Teclas **ctrl** 

CadeiaDeNomeDaVar

Refere-se à variável cujo nome é *CadeiaDeNomeDaVar*. Permite utilizar cadeias para criar nomes das variáveis a partir de uma função.

#("x"&"y"&"z") xyz

Cria ou refere-se à variável xyz.

10→ <i>r</i>	10
"1"→ <i>s1</i>	"1"
#s1	10

Devolve o valor da variável (*r*) cujo nome é guardado na variável *s1*.

E (notação científica)		Tecla EE
<i>mantissa</i> E <i>expoente</i>	23000.	23000.
Introduz um número em notação científica. O número é interpretado como <i>mantissa</i> × 10 expoente.	2300000000.+4.1E15	4.1E15
	3·10 ⁴	30000
Sugestão: Se quiser introduzir uma potência de 10 sem resultar num resultado de valor decimal, utilize 10^{número inteiro}. Nota: Pode introduzir este operador através da escrita de @E no teclado do computador. por exemplo, escreva 2.3@E4 para introduzir 2.3E4.		

g (gradianos)		Tecla 1
<i>Expr1</i> g⇒expressão	No modo Graus, Gradianos ou Radianos:	
<i>Lista1</i> g⇒lista	cos(50 ^g)	$\sqrt{2}$ 2
<i>Matriz1</i> g⇒matriz	cos({{0,100 ^g ,200 ^g }})	{1,0,-1}
Esta função fornece uma forma para especificar um ângulo de gradianos enquanto está no modo Graus ou Radianos. No modo de ângulo Radianos, multiplica <i>Expr1</i> por $\pi/200$. No modo de ângulo Graus, multiplica <i>Expr1</i> por g/100. No modo Gradianos, devolve <i>Expr1</i> inalterada. Nota: Pode introduzir este símbolo através da escrita de @g no teclado do computador.		

r (radianos)		Tecla 1
<i>Expr1</i> r⇒expressão	No modo de ângulo Graus, Gradianos ou Radianos:	
<i>Lista1</i> r⇒lista		

ʀ (radianos)

Tecla **1**

Matriz Iʀ ⇒ matriz

Esta função fornece uma forma para especificar um ângulo de radianos enquanto está no modo Graus ou Gradianos.

No modo de ângulo Graus, multiplica o argumento por $180/\pi$.

No modo de ângulo Radianos, devolve o argumento inalterado.

No modo Gradianos, multiplica o argumento por $200/\pi$.

Sugestão: Utilize ʀ se quiser impor os radianos numa definição da função, independentemente do modo que prevalece quando a função é utilizada.

Nota: Pode introduzir este símbolo através da escrita de @ʀ no teclado.

$$\cos\left(\frac{\pi}{4^{\text{ʀ}}}\right) \quad \frac{\sqrt{2}}{2}$$

$$\cos\left(\left\{0^{\text{ʀ}}, \frac{\pi}{12}, \frac{\pi}{12}, \left(\pi\right)^{\text{ʀ}}\right\}\right) \quad \left\{1, \frac{\left(\sqrt{3+1}\right) \cdot \sqrt{2}}{4}, -1\right\}$$

° (graus)

Tecla **1**

Expr1 ° ⇒ expressão

Lista1 ° ⇒ lista

Matriz1 ° ⇒ matriz

Esta função fornece uma forma para especificar um ângulo expresso em graus enquanto está no modo Radianos ou Radianos.

No modo de ângulo Radianos, multiplica o argumento por $\pi/180$.

No modo de ângulo Graus, devolve o argumento inalterado.

No modo de ângulo Gradianos, multiplica o argumento por $10/9$.

Nota: Pode introduzir este símbolo através da escrita de @d no teclado do computador.

No modo de ângulo Graus, Gradianos ou Radianos:

$$\cos(45^{\circ}) \quad \frac{\sqrt{2}}{2}$$

No modo de ângulo Radianos:

Obs: Para forçar um resultado aproximado,

Unidade portátil: Premir **ctrl** **enter**.

Windows®: Premir **Ctrl+Enter**.

Macintosh®: Premir **⌘+Enter**.

iPad®: Manter pressionada a tecla **Enter** e selecionar **≈**.

$$\cos\left(\left\{0, \frac{\pi}{4}, 90^{\circ}, 30.12^{\circ}\right\}\right) \quad \{1, 0.707107, 0., 0.864976\}$$

° , ' , " (grau/minuto/segundo)

Teclas  

$gg^{\circ}mm'ss.ss'' \Rightarrow \text{expressão}$

gg Um número positivo ou negativo

mm Um número não negativo

$ss.ss$ Um número não negativo

Devolve $gg + (mm / 60) + (ss.ss / 3600)$.

Este formato de entrada base -60 permite:

- Introduza um ângulo em graus/minutos/segundos sem se preocupar com o modo de ângulo actual.
- Introduza o tempo como horas/minutos/segundos.

Nota: Introduza dois apóstrofes a seguir $ss.ss''$ (""), não um símbolo de aspas ("").

No modo de ângulo Graus:

$25^{\circ}13'17.5''$	25.2215
$25^{\circ}30'$	$\frac{51}{2}$

∠ (ângulo)

Teclas  

$[Raio, \angle \theta _ \hat{A}ngulo] \Rightarrow \text{vector}$

(entrada polar)

$[Raio, \angle \theta _ \hat{A}ngulo, Z_Coordenada] \Rightarrow \text{vector}$

(entrada cilíndrica)

$[Raio, \angle \theta _ \hat{A}ngulo, \angle \theta _ \hat{A}ngulo] \Rightarrow \text{vector}$

(entrada esférica)

Devolve coordenadas como um vector dependendo da definição do modo
Formato do vector: rectangular, cilíndrico ou esférico.

Nota: Pode introduzir este símbolo através da escrita de $\angle$ no teclado do computador.

(Magnitude ∠ Ângulo)
 $\Rightarrow \text{ValorComplexo}$

No modo Radianos e formato do vector definido para:

rectangular

$[5 \angle 60^{\circ} \angle 45^{\circ}]$	$\left[\frac{5 \cdot \sqrt{2}}{4} \quad \frac{5 \cdot \sqrt{6}}{4} \quad \frac{5 \cdot \sqrt{2}}{2} \right]$
---	---

cilíndrico

$[5 \angle 60^{\circ} \angle 45^{\circ}]$	$\left[\frac{5 \cdot \sqrt{2}}{2} \quad \angle \frac{\pi}{3} \quad \frac{5 \cdot \sqrt{2}}{2} \right]$
---	---

esférico

$[5 \angle 60^{\circ} \angle 45^{\circ}]$	$\left[5 \quad \angle \frac{\pi}{3} \quad \angle \frac{\pi}{4} \right]$
---	--

No modo de ângulo Radianos e Formato complexo rectangular:

∠ (ângulo)

Teclas  

(entrada polar)

Introduz um valor complexo em forma polar ($r \angle \theta$). O *Ângulo* é interpretado de acordo com a definição do modo Ângulo actual.

$$5+3 \cdot i - \left(10 \angle \frac{\pi}{4} \right) \quad 5-5 \cdot \sqrt{2} + (3-5 \cdot \sqrt{2}) \cdot i$$

Obs: Para forçar um resultado aproximado,

Unidade portátil: Premir  .

Windows®: Premir **Ctrl+Enter**.

Macintosh®: Premir **⌘+Enter**.

iPad®: Manter pressionada a tecla **Enter** e seleccionar .

$$5+3 \cdot i - \left(10 \angle \frac{\pi}{4} \right) \quad -2.07107-4.07107 \cdot i$$

' (plica)

Tecla 

variável '

variável ''

Introduz um símbolo de plica numa equação diferencial. Um símbolo de plica indica uma equação diferencial de 1ª ordem, dois símbolos de números primos indicam uma 2ª ordem, etc.

$$\text{deSolve}\left(y''=y^{\frac{-1}{2}} \text{ and } y(0)=0 \text{ and } y'(0)=0,t,y\right)$$
$$\frac{2 \cdot y^{\frac{3}{4}}}{3} = t$$

_ (carácter de sublinhado como um elemento vazio)

Consulte “Elementos (nulos) vazios”, página 274.

_ (carácter de sublinhado como designação da unidade)

Teclas  

*Expr*_Unidade

$$3 \cdot \text{m} \text{ } \text{ft} \quad 9.84252 \cdot \text{ft}$$

Indica as unidades para uma *Expr*. Todos os nomes das unidades têm de começar por um carácter de sublinhado.

Nota: Pode encontrar o símbolo de conversão, , no Catálogo. Clique em  e, em seguida, em **Operadores matemáticos**.

(carácter de sublinhado como designação da unidade)

Teclas  

Pode utilizar unidades predefinidas ou criar as suas próprias unidades. Para uma lista de unidades predefinidas, abra o Catálogo e veja o separador Conversões de unidades. Pode seleccionar os nomes das unidades do Catálogo ou escrever os nomes das unidades directamente.

*Variável*_

Quando *Variável* não tiver valor, é tratada como se representasse um número complexo. Por predefinição, sem o _, a variável é tratada como real.

Se *Variável* tiver um valor, o _ é ignorado e *Variável* retém o tipo de dados originais.

Nota: Pode guardar um número complexo numa variável sem utilizar _. No entanto, para obter melhores resultados em cálculos como **cSolve()** e **cZeros()**, o _ é recomendado.

Partindo do princípio que z é indefinido:

$\text{real}(z)$	z
$\text{real}(z_)$	$\text{real}(z_)$
$\text{imag}(z)$	0
$\text{imag}(z_)$	$\text{imag}(z_)$

► (converter)

Teclas  

*Expr*_*Unidade1* ► *_**Unidade2* $\Rightarrow$ *Expr* *_**Unidade2*

3·_m►_ft 9.84252·_ft

Converte uma expressão de uma unidade para a outra.

O carácter de sublinhado _ indica as unidades. As unidades têm de ser da mesma categoria, como, por exemplo, Comprimento ou Área.

Para uma lista de unidades predefinidas, abra o Catálogo e veja o separador Conversões de unidades:

- Pode seleccionar um nome da unidade da lista.
- Pode seleccionar o operador de conversão, ►, a partir do topo da lista.

► (converter)

Teclas  

Pode também escrever os nomes das unidades manualmente. Para escrever “_” quando escrever os nomes das unidades na unidade portátil, prima

 .

Nota: Para converter as unidades de temperatura, utilize **tmpCnv()** e **ΔtmpCnv()**. O operador de conversão ► não processa unidades de temperatura.

10^()

Catálogo > 

10^(Expr1) ⇒ expressão

$10^{1.5}$ 31.6228

10^(Lista1) ⇒ lista

$10^{\{0,-2,2,a\}}$ $\left\{1, \frac{1}{100}, 100, 10^a\right\}$

Devolve 10 elevado à potência do argumento.

Para uma lista, devolve 10 elevado à potência dos elementos em *Lista1*.

10^(MatrizQuadrada1) ⇒ MatrizQuadrada

$10^{\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}}$

1.14336E7	8.17155E6	6.67589E6
9.95651E6	7.11587E6	5.81342E6
7.65298E6	5.46952E6	4.46845E6

Devolve 10 elevado à potência de *MatrizQuadrada1*. Isto não é o mesmo que calcular 10 elevado à potência de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

MatrizQuadrada1 tem de ser diagonalizável. O resultado contém sempre os números de ponto flutuante.

^-1 (recíproco)

Catálogo > 

Expr1 ^-1 ⇒ expressão

$(3.1)^{-1}$ 0.322581

Lista1 ^-1 ⇒ lista

$\{a, 4, 0.1, x, -2\}^{-1}$ $\left\{\frac{1}{a}, \frac{1}{4}, -10, \frac{1}{x}, \frac{-1}{2}\right\}$

Devolve o recíproco do argumento.

Para uma lista, devolve os recíprocos dos elementos em *Lista1*.

MatrizQuadrada1

$\wedge^{-1} \Rightarrow$ *MatrizQuadrada*

Devolve o inverso de *MatrizQuadrada1*.

MatrizQuadrada1 tem de ser uma matriz quadrada não singular.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} = \begin{bmatrix} -2 & 1 \\ 3 & -1 \\ 2 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ a & 4 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{-2}{a-2} & \frac{1}{a-2} \\ \frac{a}{2 \cdot (a-2)} & \frac{-1}{2 \cdot (a-2)} \end{bmatrix}$$

| (operador de limite)

Teclas  

Expr | *ExprBooleana1*

[**and***ExprBooleana2*]...

Expr | *ExprBooleana1*

[**or***ExprBooleana2*]...

$$\begin{array}{ll} x+1|x=3 & 4 \\ x+y|x=\sin(y) & \sin(y)+y \\ x+y|\sin(y)=x & x+y \end{array}$$

O símbolo de limite (“|”) serve como um operador binário. O operando à esquerda de | é uma expressão. O operando à direita de | especifica uma ou mais relações que servem para afetar a simplificação da expressão. Várias relações após | têm de ser reunidas por operadores “**and**” ou “**or**” lógicos.

O operador de limite fornece três tipos de funcionalidades básicas:

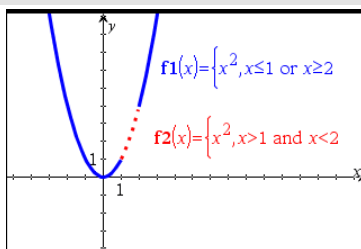
- Substituições
- Limites de intervalo
- Exclusões

As substituições estão na forma de uma igualdade, como $x=3$ ou $y=\sin(x)$. Para ser mais eficaz, o membro esquerdo deve ser uma variável simples. *Expr* | *Variável* = *valor* substituem *valor* para todas as ocorrências de *Variável* em *Expr*.

Os limites de intervalos tomam a forma de uma ou mais desigualdades reunidas pelos operadores “**and**” ou “**or**” lógicos. Os limites de intervalos também permitem a simplificação que caso contrário pode ser inválida ou não calculável.

$$\begin{array}{ll} x^3-2 \cdot x+7 \rightarrow f(x) & \text{Done} \\ f(x)|x=\sqrt{3} & \sqrt{3}+7 \\ (\sin(x))^2+2 \cdot \sin(x)-6|\sin(x)=d & d^2+2 \cdot d-6 \end{array}$$

$$\begin{array}{ll} \text{solve}(x^2-1=0,x)|x>0 \text{ and } x<2 & x=1 \\ \sqrt{x} \cdot \sqrt{\frac{1}{x}}|x>0 & 1 \\ \sqrt{x} \cdot \sqrt{\frac{1}{x}} & \sqrt{\frac{1}{x}} \cdot \sqrt{x} \end{array}$$



As exclusões utilizam o operador relacional “diferentes” ($\neq$ ou $\neq$) para excluir um valor específico de consideração. São utilizados principalmente para excluir uma solução exata quando utilizar **cSolve()**, **cZeros()**, **fMax()**, **fMin()**, **solve()**, **zeros()**, etc.

$$\text{solve}(x^2 - 1 = 0, x) | x \neq 1 \quad x = -1$$

→ (guardar)

Teclas ctrl var

Expr → *Var*

Lista → *Var*

Matriz → *Var*

Expr → *Função*(*Parâml*,...)

Lista → *Função*(*Parâml*,...)

Matriz → *Função*(*Parâml*,...)

Se a variável *Var* não existir, cria-a e inicia-a para *Expr*, *Lista* ou *Matriz*.

Se a variável *Var* já existir e não estiver bloqueada nem protegida, substitui o conteúdo por *Expr*, *Lista* ou *Matriz*.

Sugestão: Se planejar fazer cálculos simbólicos com variáveis indefinidas, evite guardar o quer que seja nas variáveis de uma letra mais utilizadas, como a, b, c, x, y, z, e por aí adiante.

Nota: Pode introduzir este operador através da escrita de **=:** no teclado como um atalho. Por exemplo, escreva **pi/4 =: myvar**.

$\frac{\pi}{4} \rightarrow \text{myvar}$	$\frac{\pi}{4}$
$2 \cdot \cos(x) \rightarrow y1(x)$	Done
$\{1, 2, 3, 4\} \rightarrow \text{lst5}$	$\{1, 2, 3, 4\}$
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow \text{matg}$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
"Hello" → <i>str1</i>	"Hello"

:= (atribuir)

Teclas  

Var := *Expr*

Var := *Lista*

Var := *Matriz*

Função(*Parâml*,...) := *Expr*

Função(*Parâml*,...) := *Lista*

Função(*Parâml*,...) := *Matriz*

Se a variável *Var* não existir, cria *Var* e inicia-a para *Expr*, *Lista* ou *Matriz*.

Se *Var* já existir e não estiver bloqueada nem protegida, substitui o conteúdo por *Expr*, *Lista* ou *Matriz*.

Sugestão: Se planejar fazer cálculos simbólicos com variáveis indefinidas, evite guardar o quer que seja nas variáveis de uma letra mais utilizadas, como a, b, c, x, y, z, e por aí adiante.

<i>myvar</i> := $\frac{\pi}{4}$	$\frac{\pi}{4}$
<i>y1</i> (<i>x</i>):=2·cos(<i>x</i>)	<i>Done</i>
<i>lst5</i> :={ 1,2,3,4 }	{ 1,2,3,4 }
<i>matg</i> := $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
<i>str1</i> := "Hello "	"Hello "

© (comentário)

Teclas  

© [*texto*]

© processa *texto* como uma linha de comentário, permitindo anotar as funções e os programas criados.

© pode estar no início ou em qualquer parte da linha. Tudo à direita de ©, no fim da linha, é o comentário.

Obs para introdução do exemplo: Para obter instruções sobre como introduzir programas com várias linhas e definições de funções, consulte a secção Calculadora do manual do utilizador do produto.

Define <i>g</i> (<i>n</i>)=Func	
© <i>Declare variables</i>	
Local <i>i,result</i>	
<i>result</i> :=0	
For <i>i</i> ,1, <i>n</i> ,1 ©Loop <i>n</i> times	
<i>result</i> := <i>result</i> + <i>i</i> ²	
EndFor	
Return <i>result</i>	
EndFunc	
<i>g</i> (3)	<i>Done</i> 14

Ob, Oh

Teclas  , teclas  

Ob *NúmeroBinário*

No modo base Dec:

Oh *NúmeroHexadecimal*

0b10+0hF+10 27

Indica um número binário ou hexadecimal, respectivamente. Para introduzir um número binário ou hexadecimal, utilize sempre o prefixo Ob ou Oh independentemente do modo Base. Sem um prefixo, um número é tratado como decimal (base 10).

Os resultados aparecem de acordo com o modo base.

No modo base Bin:

<u>Ob10+OhF+10</u>	<u>Ob11011</u>
--------------------	----------------

No modo base Hex:

<u>Ob10+OhF+10</u>	<u>Oh1B</u>
--------------------	-------------

TI-Nspire™ CX II - Comandos de desenho

Este é um documento suplementar ao Guia de Referência TI-Nspire™ e Guia de Referência TI-Nspire™ CAS. Todos os comandos TI-Nspire™ CX II serão integrados e publicados na versão 5.1 do Guia de Referência TI-Nspire™ e no Guia de Referência TI-Nspire™ CAS.

Programação de gráficos

Foram adicionados novos comandos às aplicações para desktop Unidades Portáteis TI-Nspire™ CX II e TI-Nspire™ para programação de gráficos.

As Unidade Portatéis TI-Nspire™ CX II alternam entre o modo de gráficos, ao executar comandos de gráficos, e voltam ao contexto no qual o programa foi executado após a conclusão do programa.

O ecrã irá exibir “Running...(Em funcionamento)” na barra superior enquanto o programa está a ser executado. Irá exibir “Finished (Concluído)” quando o programa concluir o processo. Qualquer ação premir-tecla irá passar o sistema para fora do modo de gráficos.

- A transição para o modo de gráficos é acionada automaticamente quando um dos comandos de Desenho (gráficos) é encontrado durante a execução do programa TI-Basic.
- Esta transição acontece apenas ao executar um programa a partir da calculadora; num documento ou calculadora no bloco de notas.
- A transição para sair do modo de gráficos acontece após a conclusão do programa.
- O modo de gráficos está disponível apenas na vista Unidades Portáteis TI-Nspire™ CX II e TI-Nspire™ CX II para desktop. Isto significa que não está disponível na vista de documento do computador no desktop ou iOS.
 - Se um comando de gráficos for encontrado ao executar um programa TI-Basic no contexto incorreto, é exibida uma mensagem de erro e o programa TI-Basic é terminado.

Ecrã de gráficos

O ecrã de gráficos irá conter um título na parte superior do ecrã que não pode ser escrito pelos comandos dos gráficos.

A área de desenho do ecrã de gráficos será limpa (cor = 255,255,255) quando o ecrã de gráficos é iniciado.

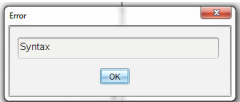
Ecrã de gráficos	Predefinição
Altura	212
Largura	318
Cor	branco: 255,255,255

Vista e definições padrão

- Os ícones de estado na barra superior (estado de bateria, estado premir para testar, indicador de rede, etc.) não estará visível enquanto o programa de gráficos estiver a funcionar.
- Cor de desenho padrão: Preto (0,0,0)
- Estilo de caneta padrão - normal, suave
 - Espessura: 1 (fina), 2 (normal), 3 (mais espessa)
 - Estilo 1 (suave), 2 (pontilhado), 3 (tracejado)
- Todos os comandos de desenho irão usar a cor e as definições de caneta atuais; tanto os valores padrão ou os valores definidos através dos comandos TI-Basic.
- A fonte do texto é fixa e não pode ser alterada.
- Qualquer saída para o ecrã de gráficos será desenhada numa janela de recorte, sendo do tamanho da área de desenho do ecrã de gráficos. Qualquer saída de desenho que se estenda para além desta área de desenho do ecrã de gráficos recortados não será desenhada. Não será exibida uma mensagem de erro.
- Todas as coordenadas x,y especificadas para os comandos de desenho são definidas como 0,0 no canto superior esquerdo da área de desenho do ecrã de gráficos.
 - **Exceções:**
 - **DrawText** utiliza as coordenadas do canto inferior esquerdo da caixa delimitadora do texto.
 - **SetWindow** utiliza o canto inferior esquerdo do ecrã
- Todos os parâmetros para os comandos podem ser fornecidos como expressões associadas a um número, que é arredondado para o seu número inteiro mais próximo.

Mensagens de erro no ecrã de gráficos

Se a validação falhar, será exibida uma mensagem de erro.

Mensagem de erro	Descrição	Vista
Erro Sintaxe	Se o verificador de sintaxe encontrar algum erro de sintaxe, apresenta uma mensagem de erro e tenta posicionar o cursor junto ao primeiro erro.	
Erro Poucos argumentos	A função ou o comando não tem um ou mais argumentos	Error Too few arguments The function or command is missing one or more arguments. OK
Erro Demasiados argumentos	A função ou o comando contém um número excessivo de argumentos e não pode ser avaliada.	Error Too many arguments The function or command contains an excessive number of arguments and cannot be evaluated. OK
Erro Tipo de dados inválido	Um argumento é do tipo de dados errado.	Error Invalid data type An argument is of the wrong data type. OK

Comandos inválidos no modo de gráficos

Alguns comandos não são permitidos assim que o programa passa para o modo de gráficos. Se os comandos forem encontrados enquanto o programa está no modo de gráficos, será exibido um erro e o programa termina.

Comando desativado	Mensagem de erro
Pedido	Request não pode ser executado no modo gráfico
CadeiaDePedido	RequestStr não pode ser executado no modo gráfico
Texto	Texto não pode ser executado no modo gráfico

Os comandos que imprimem texto na calculadora - **disp** e **dispAt** - são os comandos suportados no contexto de gráficos. O texto destes comandos será enviado para o ecrã da calculadora (não em Gráficos) e ficará visível após o programa sair e o sistema passar novamente para a app de Calculadora.

Apag.

Catálogo > 
CXII**Limpar x , y , largura, altura**

Limpa todo o ecrã se não forem especificados parâmetros.

Se x , y , largura e altura forem especificadas, o retângulo definido pelos parâmetros será limpo.

Apag.

Limpa todo o ecrã

Limpa 10,10,100,50

Limpa uma área de retângulo com o canto superior esquerdo em (10, 10) e com largura de 100, altura de 50

DrawArcCatálogo > 
CXII**DrawArc** *x, y, largura, altura, startAngle, arcAngle*

Desenhe um arco no retângulo delimitador definido com os ângulos de início e de arco fornecidos.

x, y: coordenada superior esquerda do retângulo delimitador

largura, altura: dimensões do retângulo delimitador

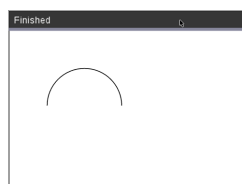
O “ângulo do arco” define a configuração angular do arco.

Estes parâmetros podem ser fornecidos como expressões que se associam a um número que é arredondado para o número inteiro mais próximo.

DrawArc 20,20,100,100,0,90



DrawArc 50,50,100,100,0,180



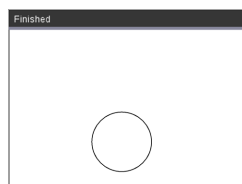
Ver também: [FillArc](#)

DrawCircleCatálogo > 
CXII**DrawCircle** *x, y, raio*

x, y: coordenada do centro

raio: raio do círculo

DrawCircle 150,150,40



Ver também: [FillCircle](#)

DrawLine

Catálogo > 
CXII

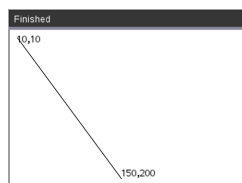
DrawLine $x1, y1, x2, y2$

Desenhe uma reta a partir de $x1, y1, x2, y2$.

Expressões que se associam a um número que será arredondado para o número inteiro mais próximo.

Limites do ecrã: Se as coordenadas especificadas fizerem com que uma parte do segmento de reta seja desenhada fora do ecrã do gráfico, essa parte será recortada e não será exibida uma mensagem de erro.

DrawLine 10,10,150,200



DrawPoly

Catálogo > 
CXII

Os comandos têm duas variantes:

DrawPoly $xlist, ylist$

ou

DrawPoly $x1, y1, x2, y2, x3, y3...xn, yn$

Nota: DrawPoly $xlist, ylist$

A forma irá ligar $x1, y1$ a $x2, y2, x2, y2$ a $x3, y3$ e por aí fora.

Nota: DrawPoly $x1, y1, x2, y2, x3, y3...xn, yn$

xn, yn **NÃO** serão automaticamente ligados a $x1, y1$.

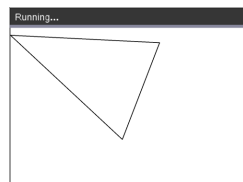
Expressões que se associam a uma lista de números reais flutuante
 $xlist, ylist$

Expressões que se associam a uma única precisão de número real
 $x1, y1...xn, yn$ = coordenadas dos vértices do polígono

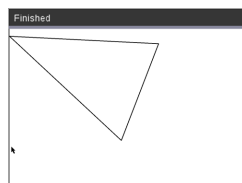
$xlist:=\{0,200,150,0\}$

$ylist:=\{10,20,150,10\}$

DrawPoly $xlist,ylist$



DrawPoly 0,10,200,20,150,150,0,10



Nota: DrawPoly: Insira as dimensões de tamanho (largura/altura) relativo para desenhar as retas.

As retas são desenhadas numa caixa delimitadora à volta da coordenada especificada e as dimensões para que o tamanho real do polígono desenhado seja maior do que a largura e altura.

Ver também: [FillPoly](#)

DrawRect

DrawRect *x, y, largura, altura*

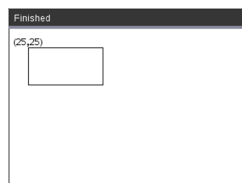
x, y: coordenada superior esquerda do retângulo

largura, altura: largura e altura do retângulo (retângulo desenhado para baixo e à direita da coordenada de início)

Nota: As retas são desenhadas na caixa delimitadora à volta da coordenada especificada e as dimensões para que o tamanho real do retângulo desenhado sejam maiores do que a largura e altura indicadas.

Ver também: [FillRect](#)

DrawRect 25,25,100,50



DrawText

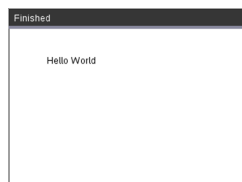
DrawText *x, y, exprOrString1*
[,exprOrString2]...

x, y: coordenada de saída de texto

Desenha o texto em *exprOrString* na localização de coordenada *x, y* especificada.

As regras para *exprOrString* são as mesmas que para **Disp** – **DrawText** pode ter diversos argumentos.

DrawText 50,50,"Hello World"



FillArc

Catálogo > 
CXII**FillArc** *x, y, largura, altura startAngle, arcAngle*

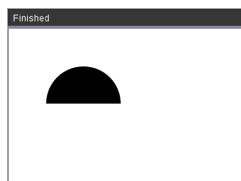
FillArc 50,50,100,100,0,180

x, y: coordenada superior esquerda do retângulo delimitador

Desenha e preenche um arco dentro do retângulo delimitador definido com os ângulos de início e de arco fornecidos.

A cor de preenchimento padrão é o preto. A cor de preenchimento pode ser definida pelo comando [SetColor](#)

O “ângulo do arco” define a configuração angular do arco



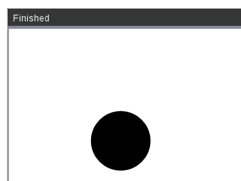
FillCircle

Catálogo > 
CXII**FillCircle** *x, y, raio*

FillCircle 150,150,40

x, y: coordenada do centro

Desenha e preenche um círculo no centro especificado com o raio especificado.

A cor de preenchimento padrão é o preto. A cor de preenchimento pode ser definida pelo comando [SetColor](#)

Aqui!

FillPoly

Catálogo > 
CXII**FillPoly** *xlist, ylist*

ou

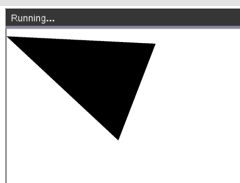
FillPoly *x1, y1, x2, y2, x3, y3...xn, yn*

xlist:={0,200,150,0}

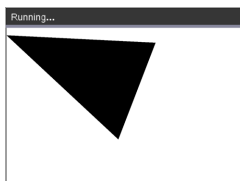
ylist:={10,20,150,10}

FillPoly xlist,ylist

Nota: A reta e cor são especificadas por [SetColor](#) e [SetPen](#)



```
FillPoly 0,10,200,20,150,150,0,10
```



FillRect

FillRect *x, y, largura, altura*

x, y: coordenada superior esquerda do retângulo

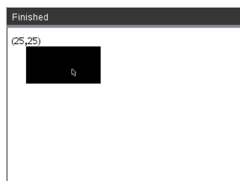
largura, altura: largura e altura do retângulo

Desenha e preenche um retângulo com o canto superior esquerdo na coordenada especificada por (x,y)

A cor de preenchimento padrão é o preto. A cor de preenchimento pode ser definida pelo comando [SetColor](#)

Nota: A reta e cor são especificadas por [SetColor](#) e [SetPen](#)

```
FillRect 25,25,100,50
```



getPlatform()Catálogo > 
CXII**getPlatform()**`getPlatform()``"dt"`

Devolve:

"dt" nas aplicações de software para
desktop

"hh" em unidades portáteis TI-Nspire™

CX

"ios" em app TI-Nspire™ CX para iPad®

PaintBuffer

Pinta o buffer dos gráficos no ecrã

Este comando é usado em conjunto com UseBuffer para aumentar a velocidade de exibição no ecrã quando o programa gera diversos objetos gráficos.

UseBuffer

Para n,1,10

x:=randInt(0,300)

y:=randInt(0,200)

raio:=randInt(10,50)

Wait 0,5

DrawCircle x,y,raio

EndFor

PaintBuffer

Este programa irá exibir todos os 10 círculos de uma só vez.

Se o comando “UseBuffer” for removido, cada círculo será exibido à medida que é desenhado.

Ver também: [UseBuffer](#)

PlotXY $x, y, forma$ x, y : coordenada para delinear a forma $forma$: um número entre 1 e 13 que especifica a forma

- 1 - círculo preenchido
- 2 - círculo vazio
- 3 - quadrado preenchido
- 4 - quadrado vazio
- 5 - cruz
- 6 - mais
- 7 - fino
- 8 - ponto médio, sólido
- 9 - ponto médio, vazio
- 10 - ponto maior, sólido
- 11 - ponto maior, vazio
- 12 - o maior ponto, sólido
- 13 - o maior ponto, vazio

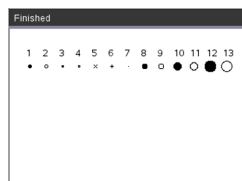
PlotXY 100,100,1

Para $n, 1, 13$

DrawText 1+22*n,40,n

PlotXY 5+22*n,50,n

EndFor



SetColorCatálogo > 
CXII**SetColor**

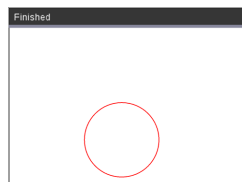
Valor-vermelho, Valor-verde, Valor-azul

Os valores válidos para vermelho, verde e azul são entre 0 e 255

Define a cor para os comandos Draw seguintes

SetColor 255,0,0

DrawCircle 150,150,100

**SetPen**Catálogo > 
CXII**SetPen**

espessura, estilo

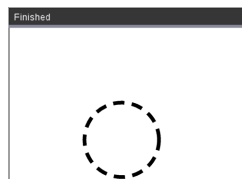
espessura: 1 <= espessura <= 3 | 1 é o mais fino, 3 é o mais grosso

estilo: 1 = suave, 2 = pontilhado, 3 = tracejado

Define o estilo da caneta para os comandos Draw seguintes

SetPen 3,3

DrawCircle 150,150,50

**SetWindow**Catálogo > 
CXII**SetWindow**

xMin, xMax, yMin, yMax

Estabelece a janela lógica mapeada para a área de desenho do gráfico. Todos os parâmetros são necessários.

Se a parte do objeto desenhado estiver fora da janela, a saída será recortada (não exibida) e não será exibida uma mensagem de erro.

SetWindow 0,160,0,120

irá definir a janela de saída para ter 0,0 no canto inferior esquerdo com uma largura de 160 e uma altura de 120

DrawLine 0,0,100,100

SetWindow 0,160,0,120

SetPen 3,3

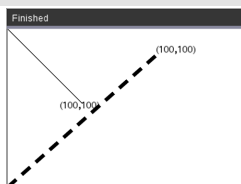
DrawLine 0,0,100,100

Se xmin for maior ou igual a xmax ou ymin for maior ou igual a ymax, é exibida uma mensagem de erro.

Os objetos desenhados antes de um comando SetWindow não serão redenhados na nova configuração.

Para repor os parâmetros da janela para os parâmetros padrão, utilize:

SetWindow 0,0,0,0



UseBuffer**UseBuffer**

Desenhe no buffer de gráficos em vez de no ecrã (para aumentar o desempenho)

Este comando é usado em conjunto com `PaintBuffer` para aumentar a velocidade de exibição no ecrã quando o programa gera diversos objetos gráficos.

Com `UseBuffer`, todos os gráficos são exibidos apenas após o próximo comando `PaintBuffer` ser executado.

`UseBuffer` apenas necessita de ser chamada para o programa uma vez, ou seja, cada utilização do `PaintBuffer` não necessita de uma utilização `UseBuffer` correspondente

UseBuffer

```
Para n,1,10  
x:=randInt(0,300)  
y:=randInt(0,200)  
raio:=randInt(10,50)  
Wait 0,5  
DrawCircle x,y,raio  
EndFor  
PaintBuffer
```

Este programa irá exibir todos os 10 círculos de uma só vez.

Se o comando “`UseBuffer`” for removido, cada círculo será exibido à medida que é desenhado.

Ver também: [PaintBuffer](#)

Elementos (nulos) vazios

Quando analisar dados do mundo real, pode não ter sempre um conjunto de dados completo. A TI-Nspire™ CAS permite elementos de dados, vazios ou nulos, para que possa continuar com os dados quase completos em vez de ter de reiniciar ou eliminar os casos incompletos.

Pode encontrar um exemplo de dados que envolve elementos vazios no capítulo Listas e Folha de cálculo, em “Representar graficamente os dados da folha de cálculo.”

A função **delVoid()** permite remover os elementos vazios de uma lista. A função **isVoid()** permite testar um elemento vazio. Para mais informações, consulte **delVoid()**, página 53, e **isVoid()**, página 104.

Nota: Para introduzir um elemento vazio manualmente numa expressão de matemática, escreva “_” ou a palavra-chave **void**. A palavra-chave **void** é convertida automaticamente para um símbolo “_” quando a expressão for avaliada. Para escrever “_” na unidade portátil, prima  .

Cálculos que envolvam elementos nulos

A maioria dos cálculos que envolvam uma entrada nula produz um resultado nulo. Consulte os casos especiais abaixo.

	—
$\gcd(100, _)$	—
$3 + _$	—
$\{5, _, 10\} - \{3, 6, 9\}$	$\{2, _, 1\}$

Argumentos da lista que contenham elementos nulos

As seguintes funções e comandos ignoram os elementos nulos encontrados nos argumentos da lista.

count, **countIf**, **cumulativeSum**, **freqTable**, **list**, **frequency**, **max**, **mean**, **median**, **product**, **stDevPop**, **stDevSamp**, **sum**, **sumIf**, **varPop**, e **varSamp**, assim como cálculos de regressão, **OneVar**, **TwoVar**, e estatística **FiveNumSummary**, intervalos de confiança e testes estatísticos

$\text{sum}(\{2, _, 3, 5, 6, 6\})$	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 da lista que contenham elementos nulos

SortA e **SortD** movem todos os elementos nulos no primeiro argumento para a 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	$\{5,3,2,1,4\}$

Nas regressões, um nulo numa lista X ou Y introduz um nulo para o elemento correspondente do resíduo.

$I1:=\{1,2,3,4,5\}; I2:=\{2,_,3,5,6,6\}$	$\{2,_,3,5,6,6\}$
LinRegMx I1,I2	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	$\{1,_,1,1,1\}$

Uma categoria omitida nas regressões introduz um nulo para o elemento correspondente do residual.

$I1:=\{1,3,4,5\}; I2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
cat:={"M","M","F","F"}; incl:={"F"}	$\{ "F" \}$
LinRegMx I1,I2,1,cat,incl	Done
stat.Resid	$\{_,_,0,0\}$
stat.XReg	$\{_,_,4,5\}$
stat.YReg	$\{_,_,5,6,6\}$
stat.FreqReg	$\{_,_,1,1\}$

Uma frequência de 0 nas regressões introduz um nulo para o elemento correspondente do resíduo.

$I1:=\{1,3,4,5\}; I2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
LinRegMx I1,I2,{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	$\{1,_,1,1\}$

Atalhos para introduzir expressões matemáticas

Os atalhos permitem introduzir elementos das expressões matemáticas, escrevendo, em vez da utilização do Catálogo ou da Paleta de símbolos. Por exemplo, para introduzir a expressão $\sqrt{6}$, pode escrever `sqrt (6)` na linha de entrada. Quando premir `Enter`, a expressão `sqrt (6)` é alterada para $\sqrt{6}$. Alguns atalhos são úteis na unidade portátil e no teclado do computador. Outros são úteis principalmente no teclado do computador.

Na unidade portátil ou no teclado do computador

Para introduzir este:	Escreva este atalho:
π	pi
θ	theta
∞	infinity
$\leq$	<=
$\geq$	>=
$\neq$	/=
$\Rightarrow$ (implicação lógica)	=>
$\Leftrightarrow$ (implicação lógica dupla, XNOR)	<=>
$\rightarrow$ (guardar operador)	=:
$ $ (valor absoluto)	abs (...)
$\sqrt{()}$	sqrt (...)
$d()$	derivative (...)
$\int()$	integral (...)
$\Sigma()$ (Modelo da soma)	sumSeq (...)
$\Pi()$ (Modelo da produto)	prodSeq (...)
$\sin^{-1}()$, $\cos^{-1}()$, ...	arcsin (...), arccos (...), ...
$\Delta\text{List}()$	deltaList (...)
$\Delta\text{tmpCnv}()$	deltaTmpCnv (...)

No teclado do computador

Para introduzir este:	Escreva este atalho:
c1, c2, ... (constantes)	@c1, @c2, ...
n1, n2, ... (constantes dos	@n1, @n2, ...

Para introduzir este:	Escreva este atalho:
números inteiros)	
i (constante imaginária)	@i
e (base logarítmica natural e)	@e
E (notação científica)	@E
$\top$ (transpor)	@t
$^{\circ}$ (radianos)	@r
$^{\circ}$ (graus)	@d
g (grados)	@g
$\angle$ (ângulo)	@<
► (conversão)	@>
►Decimal, ►approxFraction (), etc.	@>Decimal, @>approxFraction(), etc.

Hierarquia do EOS™ (Equation Operating System)

Esta secção descreve o Equation Operating System (EOS™) utilizado pela tecnologia de aprendizagem de matemática e ciências TI-Nspire™ CAS. Os números, as variáveis e as funções são introduzidos numa sequência simples. O software EOS™ avalia as expressões e as equações com a associação parentética e de acordo com as prioridades descritas abaixo.

Ordem de avaliação

Nível	Operador
1	Parêntesis curvos (), parêntesis rectos [], chavetas { }
2	Indirecta (#)
3	Chamadas de funções
4	Pós-operadores: graus-minutos-segundos (°, ', "), factorial (!), percentagem (%), radianos (°), carácter de sublinhado ([_]), transpor (T)
5	Exponenciação, operador de potência (^)
6	Negação (-)
7	Concatenação de cadeias (&)
8	Multiplicação (•), divisão (/)
9	Adição (+), subtracção (-)
10	Relações de igualdade: igual (=), não igual (≠ ou ≠), menor que (<), igual ou menor que (≤ ou ≤), maior que (>), igual ou maior que (≥ ou ≥)
11	not lógico
12	and lógico
13	Lógico or
14	xou, nor, nand
15	Implicação lógica (⇒)
16	Implicação lógica dupla, XNOR (⇔)
17	Operador de limite (" ")
18	Guardar (→)

Parêntesis curvos, parêntesis rectos e chavetas

Todos os cálculos dentro de um par de parêntesis rectos, parêntesis curvos ou chavetas são avaliados primeiro. Por exemplo, na expressão $4(1+2)$, o software EOS™ avalia primeiro a parte da expressão dentro dos parêntesis, $1+2$, e, em seguida, multiplica o resultado, 3, por 4.

O número de parêntesis curvos, parêntesis rectos e chavetas de abertura e fecho tem de ser igual numa expressão ou equação. Se não for, aparece uma mensagem de erro que indica o elemento inexistente. Por exemplo, $(1+2)/(3+4)$ mostra a mensagem de erro "Inexistente)."

Nota: Como o software TI-Nspire™ CAS permite definir as suas funções próprias, o nome de uma variável seguido por uma expressão entre parêntesis é considerado uma "chamada de função" em vez de uma multiplicação implícita. Por exemplo, $a(b+c)$ é a função a avaliada por $b+c$. Para multiplicar a expressão $b+c$ pela variável a , utilize a multiplicação explícita: $a*(b+c)$.

Indirecta

O operador da indirecta (#) converte uma cadeia num nome de função ou variável. Por exemplo, $\#("x"&"y"&"z")$ cria o nome de variável xyz . A indirecta permite também a criação e a modificação de variáveis dentro de um programa. Por exemplo, se $10 \rightarrow r$ e $"r" \rightarrow s1$, $\#s1=10$.

Pós-operadores

Os pós-operadores são operadores que vêm directamente após um argumento, como $5!$, 25% ou $60^\circ 15' 45$. Os argumentos seguidos por um pós-operador são avaliados no quarto nível de prioridade. Por exemplo, na expressão $4^3!$, $3!$ é avaliada primeiro. O resultado, 6, torna-se no expoente de 4 para produzir 4096.

Exponenciação

A exponenciação (^) e a exponenciação de elemento por elemento (.^) são avaliadas da direita para a esquerda. Por exemplo, a expressão 2^3^2 é avaliada como $2^{(3^2)}$ para produzir 512. É diferente de $(2^3)^2$, que é 64.

Negação

Para introduzir um número negativo, prima $\boxed{-}$ seguida pelo número. As pós-operações e a exponenciação são efectuadas antes da negação. Por exemplo, o resultado de $-x^2$ é um número negativo e $-9^2 = -81$. Utilize os parêntesis para elevar um número negativo ao quadrado $(-9)^2$ para produzir 81.

Limite ("|")

O argumento a seguir ao operador de limite ("|") fornece um conjunto de limites que afetam a avaliação do argumento antes do operador.

TI-Nspire CX II - Funcionalidades de programação TI-Basic

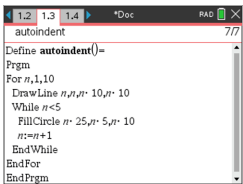
Recuos automáticos no Editor de Programação

O editor do programa TI-Nspire™ dá instruções de recuos automáticos dentro de um comando de bloco.

Os comandos de bloco são If/EndIf, For/EndFor, While/EndWhile, Loop/EndLoop, Try/EndTry

O editor irá digitar automaticamente os espaçamentos nos comandos do programa dentro de um comando de bloco. O comando de encerramento do bloco será alinhado com o comando de abertura.

O exemplo abaixo indica o recuo automático nos comandos de bloco agrupados.



Os fragmentos de código que são copiados e colados manterão o recuo original.

Ao abrir um programa criado na versão anterior do software irá manter o recuo original.

Mensagens de erro melhoradas para TI-Basic

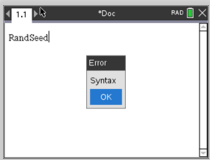
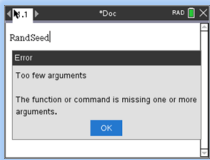
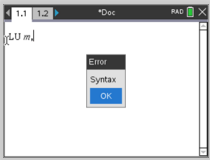
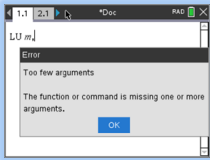
Erros

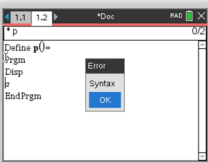
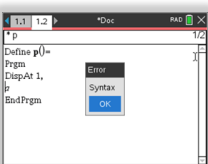
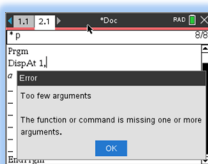
Condição de erro	Nova mensagem
Erro na instrução de condição (Se/Enquanto)	Uma instrução condicional não foi resolvida como TRUE (VERDADEIRA) ou FALSE (FALSA) NOTA: Com a alteração de colocar o cursor na reta com o erro, deixamos de especificar se o erro é uma instrução "If (Se)" ou "While (Enquanto)"
EndIf em falta	Esperado EndIf , mas encontrada uma instrução End diferente
EndFor em falta	Esperado EndFor , mas encontrada uma instrução End diferente
EndWhile em falta	Esperado EndWhile , mas encontrada uma instrução End diferente

Condição de erro	Nova mensagem
EndLoop em falta	Esperado EndLoop , mas encontrada uma instrução End diferente
EndTry em falta	Esperado EndTry , mas encontrada uma instrução End diferente
“Then” omitido depois de If <condition>	If..Then em falta
“Then” omitido depois de Elseif <condition>	Then em falta no bloco: Elseif.
Quando “Then”, “Else” e “Elseif” são encontrados fora dos blocos de controlo	Else, inválido fora dos blocos: If..Then..EndIf ou Try..EndTry
“Elseif” aparece fora do bloco “If..Then..EndIf”	Elseif inválido fora do bloco: If...Then...EndIf
“Then” aparece fora do bloco “If....EndIf”	Then inválido fora do bloco: If..EndIf

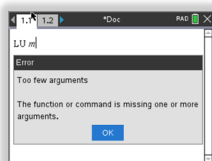
Erros de sintaxe

No caso de comandos que esperam um ou mais argumentos são denominados como uma lista de argumentos incompleta, será emitido um erro “**Erro de poucos argumentos**” ao invés de erro de “sintaxe”

Comportamento atual	Novo comportamento CX II
 The screenshot shows a TI-NSpire CX II interface with a command window containing 'RandSeed'. An error dialog box is displayed with the message 'Error: Syntax' and an 'OK' button.	 The screenshot shows a TI-NSpire CX II interface with a command window containing 'RandSeed'. An error dialog box is displayed with the message 'Error: Too few arguments' and a detailed description: 'The function or command is missing one or more arguments.' with an 'OK' button.
 The screenshot shows a TI-NSpire CX II interface with a command window containing 'LU m'. An error dialog box is displayed with the message 'Error: Syntax' and an 'OK' button.	 The screenshot shows a TI-NSpire CX II interface with a command window containing 'LU m'. An error dialog box is displayed with the message 'Error: Too few arguments' and a detailed description: 'The function or command is missing one or more arguments.' with an 'OK' button.

Comportamento atual	Novo comportamento CX II
	
	

Nota: Quando uma lista de argumentos incompleta não é seguida por uma vírgula, a mensagem de erro é: “poucos argumentos”. Isto é como nas edições anteriores.



Constantes e valores

A tabela que se segue apresenta uma listagem das constantes e respetivos valores disponíveis ao efetuar conversões de unidades (NIST 2018). Podem ser introduzidas manualmente ou selecionadas na lista **Constantes** em **Utilitários > Conversões de unidades** (Portátil: Premir  3).

Constante	Nome	Valor
_c	Velocidade da luz	299792458 _m/_s
_Cc	Constante Coulomb	8987551792.261 _m/_F
_Fc	Constante de Faraday	96485.33212 _coul/_mol
_g	Aceleração da gravidade	9.80665 _m/_s²
_Gc	Constante gravitacional	6.6743E-11 _m³/_kg/_s²
_h	Constante de Planck	6.62607015E-34 _J _s
_k	Constante de Boltzmann	1.380649E-23 _J/_°K
_μ0	Permeabilidade no vazio	1.25663706127E-6 _N/_A²
_μb	Magnetão de Bohr	9.274009994E-24 _J _m²/_Wb
_Me	Massa de repouso do eletrão	9.10938371390E-31 _kg
_Mμ	Massa de Muão	1.883531627E-28 _kg
_Mn	Massa de repouso do neutrão	1.67492750058E-27 _kg
_Mp	Massa de repouso do protão	1.67262192595E-27 _kg
_Na	Número de Avogradro	6.02214076E23 /_mol
_q	Carga do eletrão	1.602176634E-19 _coul
_R	Pressão constante para o potencial da água	0.0831 L bars/mole K
_Rb	Raio Bohr	5.29177210903E-11 _m
_Rc	Constante molar do gás	8.314462618 _J/_mol/_°K
_Rdb	Constante de Rydberg	10973731.568160/_m
_Re	Raio do eletrão	2.8179403262E-15 _m
_u	Massa atómica	1.66053906893E-27 _kg
_Vm	Volume molar	2.241396954E-2 _m³/_mol
_ε0	Permissividade no vazio	8.8541878188E-12 _F/_m
_σ	Constante de Stefan-Boltzmann	5.670367E-8 _W/_m²/_°K⁴
_φ0	Fluxo magnético	2.067833831E-15 _Wb

Mensagens e códigos de erros

Quando ocorre um erro, o código é atribuído à variável *errCode*. As funções e os programas definidos pelos utilizadores podem examinar *errCode* para determinar a causa de um erro. Para obter um exemplo da utilização de *errCode*, consulte o Exemplo 2 no comando **Try**, xxxx.

Nota: Algumas condições de erro aplicam-se apenas aos produtos TI-Nspire™ CAS e algumas aplicam-se apenas aos produtos TI-Nspire™.

Código de erro	Descrição
10	Uma função não devolveu um valor
20	Um teste não resolveu para VERDADEIRO ou FALSO. Geralmente, as variáveis indefinidas não podem ser comparadas. Por exemplo, o teste <code>If a<b</code> provocará este erro se <code>a</code> ou <code>b</code> forem indefinidos quando a afirmação <code>If</code> for executada.
30	O argumento não pode ser o nome de uma pasta.
40	Erro do argumento
50	Argumentos não coincidentes Dois ou mais argumentos têm de ser do mesmo tipo.
60	O argumento tem de ser uma expressão Booleana ou um número inteiro
70	O argumento tem de ser um número decimal
90	O argumento tem de ser uma lista
100	O argumento tem de ser uma matriz
130	O argumento tem de ser um conjunto de caracteres alfanuméricos
140	O argumento tem de ser o nome de uma variável. Certifique-se de que o nome: • não começa por um dígito • não contém espaços ou caracteres especiais • não utiliza o carácter de sublinhado ou um intervalo de forma inválida • não excede as limitações do comprimento Consulte a secção Calculadora para obter mais informações.
160	O argumento tem de ser uma expressão
165	Pilhas demasiado fracas para envio ou recepção Instale pilhas novas antes do envio ou da recepção.
170	Limite

Código de erro	Descrição
	O limite inferior tem de ser inferior ao limite superior para definir o intervalo da procura.
180	Pausa A tecla  ou  foi premida durante um cálculo longo ou a execução do programa.
190	Definição circular Esta mensagem aparece para evitar o esgotamento da memória durante a substituição infinita de valores das variáveis durante a simplificação. Por exemplo, $a+1 \rightarrow a$, em que a é uma variável indefinida, provocará este erro.
200	Expressão de constrangimento inválida Por exemplo, $\text{solve}(3x^2-4=0, x) \mid x < 0 \text{ ou } x > 5$ produzirá esta mensagem de erro porque a restrição é separada por “or” em vez de “and.”
210	Tipo de dados inválido Um argumento é do tipo de dados errado.
220	Limite dependente
230	Dimensão Um índice de lista ou matriz não é válido. Por exemplo, se a lista $\{1,2,3,4\}$ for guardada em $L1$, $L1[5]$ é um erro de dimensão porque $L1$ contém apenas quatro elementos.
235	Erro de dimensão. Elementos insuficientes nas listas.
240	Erro de dimensão Dois ou mais argumentos têm de ter as mesmas dimensões. Por exemplo, $[1,2]+[1,2,3]$ é uma incorrespondência de dimensões porque as matrizes contêm um número de elementos diferentes.
250	Dividir por zero
260	Erro do domínio Um argumento tem de estar num domínio específico. Por exemplo, $\text{rand}(0)$ não válido.
270	Nome da variável duplicado
280	Else e Elseif inválidas fora do bloco If..Endif
290	EndTry não tem a afirmação Else correspondente
295	Iteração excessiva

Código de erro	Descrição
300	Matriz ou lista de 2 ou 3 elementos prevista
310	O primeiro argumento de nSolve tem de ser uma equação de variável individual. Não pode conter uma variável sem valor diferente da variável de interesse.
320	O primeiro argumento de solve ou cSolve tem de ser uma equação ou desigualdade Por exemplo, solve($3x^2-4$,x) não é válido porque o primeiro argumento não é uma equação.
345	Unidades inconsistentes
350	Índice fora do intervalo
360	O nome não é um nome de variável válido
380	Ans indefinida O cálculo anterior não criou Ans ou nenhum cálculo anterior foi introduzido.
390	Atribuição inválida
400	Valor de atribuição inválido
410	Comando inválido
430	Inválido para as definições actuais do modo
435	Tentativa inválida
440	Multiplicação implícita inválida Por exemplo, $x(x+1)$ não é válida; visto que, $x*(x+1)$ é a sintaxe correcta. Esta serve para evitar confusões entre as chamadas de funções e a multiplicação implícita.
450	Inválida numa função ou expressão actual Apenas determinados comandos são válidos numa função definida pelo utilizador.
490	Inválido no bloco Try..EndTry
510	Matriz ou lista inválida
550	Programa ou função exterior inválido Vários comandos não são válidos fora de uma função ou de um programa. Por exemplo, Local não pode ser utilizado excepto se estiver numa função ou num programa.
560	Inválido fora dos blocos Loop..EndLoop, For..EndFor ou While..EndWhile Por exemplo, o comando Exit só válido dentro destes blocos circulares.

Código de erro	Descrição
565	Programa exterior inválido
570	Nome do caminho inválido Por exemplo, \var não é válido.
575	Complexo polar inválido
580	Referência de programa inválida Os programas não podem ser referenciados nas funções ou expressões, como, por exemplo, 1+p(x) em que p é um programa.
600	Tabela inválida
605	Utilização de unidades inválidas
610	Nome de variável inválido numa instrução Local
620	Nome de função ou variável inválido
630	Referência da variável inválida
640	Sintaxe de vector inválida
650	Transmissão da ligação Uma transmissão entre as duas unidades não foi concluída. Verifique se o cabo de ligação foi está ligado correctamente a ambas as extremidades.
665	Matriz não diagonalizável
670	Pouca memória 1. Eliminar alguns dados deste documento 2. Guardar e fechar este documento Se 1 e 2 não resultarem, retirar e reinserir as pilhas
672	Esgotamento de recursos
673	Esgotamento de recursos
680	Falta (
690	Falta)
700	Falta “
710	Falta]
720	Falta }
730	Falta do início ou do fim da sintaxe do bloco

Código de erro	Descrição
740	Falta Then no bloco If..EndIf
750	Nome não é uma função nem um programa
765	Nenhuma função selecionada
780	Nenhuma solução encontrada
800	Resultado não real Por exemplo, se o software estiver na definição real, $\sqrt{-1}$ não é válido. Para permitir resultados em complexos, altere a definição do modo “Real ou Complexo” para RECTANGULAR ou POLAR.
830	Excesso
850	Programa não encontrado Uma referência do programa dentro de outro programa não pode ser encontrada no caminho fornecido durante a execução.
855	Funções de tipo Rand não permitidas no gráfico
860	Recursividade muito profunda
870	Variável do sistema ou nome reservado
900	Erro do argumento O modelo mediana-mediana não pode ser aplicado ao conjunto de dados.
910	Erro de sintaxe
920	Texto não encontrado
930	Poucos argumentos A função ou o comando não tem um ou mais argumentos.
940	Demasiados argumentos A expressão ou equação contém um número excessivo de argumentos e não pode ser avaliada.
950	Demasiados índices
955	Demasiadas variáveis indefinidas
960	Variável indefinida Nenhum valor atribuído à variável. Utilize um dos seguintes comandos: • sto → • := • Define

Código de erro	Descrição
	para atribuir valores às variáveis.
965	SO não licenciado
970	Variável em utilização para que as referências ou as alterações não sejam permitidas
980	Variável protegida
990	Nome da variável inválido Certifique-se de que o nome não excede as limitações de comprimento
1000	Domínio das variáveis da janela
1010	Zoom
1020	Erro interno
1030	Violação da memória protegida
1040	Função não suportada. Esta função requer o Computer Algebra System. Tente o TI-Nspire™ CAS.
1045	Operador não suportado. Este operador requer o Computer Algebra System. Tente o TI-Nspire™ CAS.
1050	Função não suportada. Este operador requer o Computer Algebra System. Tente o TI-Nspire™ CAS.
1060	O argumento de entrada tem de ser numérico. Apenas entradas com valores numéricos são permitidas.
1070	Argumento da função Trig demasiado grande para redução precisa
1080	Utilização não suportada de Ans. Esta aplicação não suporta Ans.
1090	Função indefinida. Utilize um dos seguintes comandos: • Define • := • sto → para definir uma função.
1100	Cálculo não real Por exemplo, se o software estiver na definição real, $\sqrt{(-1)}$ não é válido. Para permitir resultados em complexos, altere a definição do modo “Real ou Complexo” para RECTANGULAR ou POLAR.
1110	Limites inválidos

Código de erro	Descrição
1120	Nenhuma alteração de sinal
1130	O argumento não pode ser uma lista ou matriz
1140	Erro do argumento O primeiro argumento tem de ser uma expressão polinomial no segundo argumento. Se o segundo argumento for omitido, o software tenta seleccionar uma predefinição.
1150	Erro do argumento Os primeiros dois argumentos têm de ser uma expressão polinomial no terceiro argumento. Se o terceiro argumento for omitido, o software tenta seleccionar uma predefinição.
1160	Nome do caminho da biblioteca inválido Um nome do caminho tem de estar no formato <code>xxx\yyy</code>, em que: - A parte <code>xxx</code> pode ter de 1 a 16 caracteres. - A parte <code>yyy</code> pode ter de 1 a 15 caracteres. Consulte a secção Biblioteca na documentação para obter mais informações.
1170	Utilização inválida do nome do caminho da biblioteca - Não pode atribuir um valor a um nome do caminho com Define, <code>:=</code>, ou <code>sto</code> ➔. - Não pode declarar o nome de um caminho como uma variável local ou ser utilizada como um parâmetro numa definição de programa ou função.
1180	Nome da variável da biblioteca inválido. Certifique-se de que o nome: - não contém um ponto - não começa com um carácter de sublinhado - não excede 15 caracteres Consulte a secção Biblioteca na documentação para obter mais informações.
1190	Documento da biblioteca não encontrado: - Verifique se a biblioteca está na pasta MyLib. - Actualizar bibliotecas. Consulte a secção Biblioteca na documentação para obter mais informações.
1200	Variável da biblioteca não encontrada: - Verifique se a variável da biblioteca existe no primeiro problema da biblioteca. - Certifique-se de que a variável da biblioteca foi definida como BibPub

Código de erro	Descrição
	ou BibPriv. • Actualizar bibliotecas. Consulte a secção Biblioteca na documentação para obter mais informações.
1210	Nome de atalho na biblioteca inválido. Certifique-se de que o nome: • não contém um ponto • não começa com um carácter de sublinhado • não excede 16 caracteres • não é um nome reservado Consulte a secção Biblioteca na documentação para obter mais informações.
1220	Erro de domínio: As funções RectaTangente e RectaNormal suportam apenas funções reais de variável real.
1230	Erro de domínio. Os operadores de conversão trigonométrica não são suportados nos modos de ângulos de graus ou grados.
1250	Erro do argumento Utilize um sistema de equações lineares. Exemplo de um sistema de duas equações lineares com variáveis x e y: $3x+7y=5$ $2y-5x=-1$
1260	Erro do argumento: O primeiro argumento de nfMin ou nfMax tem de ser uma expressão numa variável individual. Não pode conter uma variável sem valor diferente da variável de interesse.
1270	Erro do argumento A ordem da derivada tem de ser igual a 1 ou 2.
1280	Erro do argumento Utilize um polinómio num formato expandido numa variável.
1290	Erro do argumento Utilize um polinómio numa variável.
1300	Erro do argumento

Código de erro	Descrição
	Tem de passar os coeficientes do polinómio para valores numéricos.
1310	Erro do argumento: Uma função não conseguiu avaliar um ou mais argumentos.
1380	Erro de domínio: Não são permitidas chamadas aninhadas para a função de domínio().

Códigos de aviso e mensagens

Pode utilizar a função **warnCodes()** para guardar os códigos de avisos gerados ao avaliar uma expressão. Esta tabela lista todos os códigos de aviso numéricos e as mensagens associadas. Para um exemplo de guardar códigos de aviso, consulte **warnCodes()**, aqui.

Código de aviso	Mensagem
10000	A operação pode introduzir soluções falsas. Quando aplicável, tente utilizar métodos gráficos para verificar os resultados.
10001	A diferenciação de uma equação pode produzir uma equação falsa.
10002	Solução questionável Quando aplicável, tente utilizar métodos gráficos para verificar os resultados.
10003	Precisão questionável Quando aplicável, tente utilizar métodos gráficos para verificar os resultados.
10004	A operação pode perder as soluções. Quando aplicável, tente utilizar métodos gráficos para verificar os resultados.
10005	cSolve pode especificar mais zeros.
10006	Solve pode especificar mais zeros. Quando aplicável, tente utilizar métodos gráficos para verificar os resultados.
10007	Podem existir mais soluções. Tente especificar limites inferiores e superiores apropriados e/ou uma tentativa. Exemplos que utilizam solve(): • solve(Equação, Var=Tentativa) LimiteInferior<Var<LimiteSuperior • solve(Equação, Var) LimiteInferior<Var<LimiteSuperior • solve(Equação,Var=Tentativa) Quando aplicável, tente utilizar métodos gráficos para verificar os resultados.
10008	O domínio do resultado pode ser inferior ao domínio da entrada.
10009	O domínio do resultado pode ser superior ao domínio da entrada.
10012	Cálculo não real
10013	∞^0 ou undef^0 substituído por 1
10014	undef^0 substituído por 1
10015	1^∞ ou 1^undef substituído por 1
10016	1^undef substituído por 1

Código de aviso	Mensagem
10017	Excesso substituído por ∞ ou $-\infty$
10018	A operação requer e devolve um valor de 64 bits.
10019	Esgotamento de recursos, a simplificação pode estar incompleta.
10020	Argumento da função trigonométrica demasiado para redução precisa.
10021	A entrada contém um parâmetro indefinido. O resultado pode não ser válido para todos os valores de parâmetros possíveis.
10022	A especificação dos limites superiores e inferiores adequados pode produzir uma solução.
10023	Escalar foi multiplicado pela matriz de identidade.
10024	Resultado obtido utilizando aritmética aproximada.
10025	A equivalência não pode ser verificada no modo EXACTO.
10026	A restrição pode ser ignorada. Especifique uma restrição sob a forma "Variável-Símbolo MathTest-constante" ou um conjunto destas formas, por exemplo, "x<3 and x>-12"

Informações gerais

Ajuda online

education.ti.com/eguide

Selecione o seu país para obter mais informação sobre o produto.

Contacte a assistência técnica da TI

education.ti.com/ti-cares

Selecione o seu país para obter recursos técnicos ou assistência.

Informações da Assistência e Garantia

education.ti.com/warranty

Selecione o seu país para obter informações sobre a duração e os termos da garantia ou sobre a assistência ao produto.

Garantia Limitada. Esta garantia não afeta os seus direitos legais.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243

Índice remissivo

-	
- , subtrair[*]	230
!	
!, factorial	241
"	
", notação de segundos	250
#	
#, indirecta	247
#, operador da indirecta	279
%	
%, percentagem	236
&	
&, acrescentar	241
*	
*, multiplicar	231
'	
', notação de minutos	250
', plica	251
.	
.- , ponto subtracção	235
.* , ponto multiplicação	235
./ , ponto divisão	235
.^ , ponto potência	236
.+ , ponto adição	234
/	
/, dividir[*]	232
:	
:=, atribuir	256

^	
^, potência	233
^-1, recíproco	253
_	
_ , designação da unidade	251
, operador de limite	254
+	
+, adicionar	230
=	
≠ , diferente[*]	237
=, igual	237
>	
>, maior que	239
Π	
Π, produto[*]	244
Σ	
Σ(), soma[*]	245
ΣInt()	245
ΣPrn()	246
√	
√ , raiz quadrada[*]	244
∫	
∫ , integrar[*]	242
≤	
≤ , igual ou menor que	238
≥	
≥ , igual ou maior que	239

►	
►, converter para ângulo de	
gradianos[Grad]	95
►, converter unidades[*]	252
►Base2, visualizar como binário	
[Base2]	19
►Base10, visualizar como número	
inteiro decimal[Base10] ...	20
►Base16, visualizar como	
hexadecimal[Base16]	21
►Cylind, visualizar como vetor	
cilíndrico[Cylind]	46
►DD, visualizar como ângulo	
decimal[DD]	49
►Decimal, visualizar resultado como	
decimal[Decimal]	50
►DMS, visualizar como	
grau/minuto/segundo	
[DMS]	59
►Polar, visualizar como vetor polar	
[Polar]	145
►Sphere, visualizar como vetor	
esférico[Sphere]	192
►cos, apresenta expressão em	
função do co-seno[cos]	32
►exp[exp]	69
►FracçãoAprox()	14
►Rad, converter medida de ângulo	
para radianos	157
►Rect, visualizar como vetor	
rectangular	160
►seno, apresenta em função do seno	
[seno]	182
→	
→, guardar	255
⇒	
⇒, implicação lógica[*]	240, 276
⇔	
⇔, implicação lógica dupla[*]	240

©	
©, comentário	256
°	
°, graus/minutos/segundos[*]	250
°, notação de graus[*]	249
#	
0b, indicador binário	256
#	
0h, indicador hexadecimal	256
#	
10^(), potência de dez	253
A	
a definir	
função ou programa privado ...	51
função ou programa público ...	52
abs(), valor absoluto	8
acrescentar, &	241
adicionar, +	230
aleatória	
matriz, randMat()	158
norma, randNorm()	158
aleatório	
polinómio, randPoly()	159
semente de número, RandSeed	159
amortTbl(), tabela de amortização	8, 18
amostra aleatória	159
and, Boolean operator	9
angle(), ângulo	10
ângulo, angle()	10
ANOVA, análise de variação de uma	
via	11
ANOVA2way, análise de variação	
bidireccional	11
Ans, última resposta	14
Apag.	262
apagar	
erro, ClrErr	28
approx(), aproximado	14-15
Apr., apresentar dados	173
apresentar dados, Apr.	173

aproximado, approx()	14-15	dimensão, dim()	57
arccos()	15	cadeia de caracteres, char()	24
arccosh()	15	cadeia do formato, format()	79
arccot()	15	CadeiaDePedido	165
arccoth()	15	cadeias	
arccsc()	15	acrescentar, &	241
arccsch()	15	cadeia de caracteres, char()	24
arcLen(), comprimento do arco	16	cadeia para expressão, expr()	72, 117
arco-coseno, cos ⁻¹ ()	34	código de carácter, ord()	143
arco-seno, sin ⁻¹ ()	184	deslocar, shift()	179
arco-tangente, tan ⁻¹ ()	200	esquerda, left()	105
arcsec()	16	expressão para cadeia, string()	196
arcsech()	16	formatar	79
arcsin()	16	formato, format()	79
arcsinh()	16	indirecta, #	247
arctan()	16	mid-string, mid()	125
arctanh()	16	right, right()	100, 166
argumentos em funções TVM	213	rodar, rotate()	168
Argumentos TVM	213	utilizar para criar nomes de	
arredondar(), round	169	variáveis	279
arredondar, round()	169	within, inString	99
atalhos do teclado	276	carácter de sublinhado, _	251
atalhos, teclado	276	caracteres	
AtualizarVarsSonda	162	cadeia, char()	24
augment(), aumentar/concatenar	17	código numérico, ord()	143
aumentar/concatenar, aumentar()	17	Cdf()	75
avaliação, ordem de	278	ceiling(), ceiling	22
avaliar polinómio, polyEval()	147	ceiling, ceiling()	22
avgRC(), taxa de câmbio média	17	centralDiff()	23
		cFactor(), factor completo	23
B		char(), cadeia de caracteres	24
BibPriv	51	ciclo, Cycle	46
BibPub	52	ciclo, Loop	120
binário		ClearAZ	27
indicador, Ob	256	ClrErr, apagar erro	28
visualizar, ►Base2	19	CnvTmpDelta()	53
binomCdf()	21, 101	códigos de aviso e mensagens	293
binomPdf()	22	colAugment	28
bloquear variáveis e grupos de		colDim(), dimensão da coluna da	
variáveis	116	matriz	28
Bloquear, bloquear variável ou		colNorm(), norma da coluna da	
grupo de variáveis	116	matriz	29
Boolean operators		com,	254
and	9	Comando Parar	196
C		Comando Text	204
cadeia		Comando Wait	218
comprimento	57	combinações, nCr()	131
		comDenom(), denominador comum	29
		comentário, ©	256

completeSquare(), complete square	30	crossP(), produto cruzado	40
complexo		csc(), co-secante	40
conjugado, conj()	31	csc ⁻¹ (), co-secante inversa	40
factor, cFactor()	23	csch(), co-secante hiperbólica	41
solve, cSolve()	41	csch ⁻¹ (), co-secante hiperbólica	
zeros, cZeros()	46	inversa	41
comprimento da cadeia	57	cSolve(), resolução complexa	41
comprimento do arco, arcLen()	16	CubicReg, regressão cúbica	44
conj(), conjugado complexo	31	Cycle, ciclo	46
constante		cZeros(), zeros complexos	46
em solve()	188		
constantes		D	
atalhos para	276	d(), primeira derivada	241
em cSolve()	43	dbd(), dias entre datas	49
em cZeros()	48	decimal	
em deSolve()	55	visualizar ângulo, ►DD	49
em solve()	190	visualizar número inteiro,	
constructMat(), construir matriz	31	►Base10	20
construir matriz, constructMat()	31	definição, Lbl	104
contar condicionalmente itens numa		definições do modo, getMode()	92
lista, countif()	38	definições, obter actual	92
contar dias entre datas, dbd()	49	Definir	50
contar itens numa lista, contar()	38	definir	
converter		modo, setMode()	177
►Grad	95	Definir BibPriv	51
►Rad	157	Definir BibPub	52
unidades	252	Definir, definir	50
coordenada polar, R►Pr()	156	DelVar, eliminar variável	53
coordenada polar, R►Pθ()	156	delVoid(), remover elementos nulos	53
copiar variável ou função, CopyVar	32	denominador	29
corrMat(), matriz de correlação	32	denominador comum, comDenom()	29
cos(), co-seno	33	densidade da probabilidade,	
cos ⁻¹ , arco-coseno	34	normPdf()	136
co-seno		densidade de probabilidade student-	
apresenta a expressão em		t, tPdf()	208
função do	32	derivada	
co-seno, cos()	33	numérica, nDerivative()	132
cosh(), co-seno hiperbólico	35	derivada implícita, Impdif()	98
cosh ⁻¹ (), arco-coseno hiperbólico	36	derivada ou derivada de índice N	
cot(), co-tangente	36	modelo para	6
cot ⁻¹ (), arco-cotangente	37	derivada()	53
co-tangente, cot()	36	derivadas	
coth(), co-tangente hiperbólica	37	derivada numérica, nDeriv()	133
coth ⁻¹ (), arco-cotangente		derivada numérica, nDerivative()	
hiperbólico	38	primeira derivada, d()	132
count(), contar itens numa lista	38	desbloquear variáveis e grupos de	241
countif(), contar condicionalmente		variáveis	216
itens numa lista	38		
cPolyRoots()	39		

Desbloquear, desbloquear variável ou grupo de variáveis	216	enquanto, While	220
desenhar	263-265	entrada, Input	98
deslocar, shift()	179	EOS (Equation Operating System) ..	278
deSolve(), solução	54	equações simultâneas, simult()	181
desvio padrão, stdDev()	195, 216	Equation Operating System (EOS) ..	278
det(), determinante da matriz	56	erro de passagem, PassErr	144
diag(), diagonal da matriz	56	erros e resolução de problemas apagar erro, ClrErr	28
dias entre datas, dbd()	49	erro de passagem, PassErr	144
diferente, ≠	237	esquerda, left()	105
dim(), dimensão	57	estatística combinações, nCr()	131
dimensão, dim()	57	desvio padrão, stdDev()	195, 216
direita(), right	166	estatística de uma variável, OneVar	140
direita, right()	100, 166	factorial, !	241
Disp, visualizar dados	57	média, mean()	122
DispAt	57	mediana, median()	123
dividir, /	232	norma aleatória, randNorm() ..	158
divisão inteira, intDiv()	99	permutações, nPr()	138
dominantTerm(), termo dominante	61	resultados de duas variáveis, TwoVar	214
domínio(), função de domínio	60	semente de número aleatório, RandSeed	159
dotP(), produto do ponto	62	variação, variance()	217
E		estatística de uma variável, OneVar	140
e	69	euler(), Euler function	66
E, expoente	248	exact(), exacto	68
e para uma potência, e^()	63, 69	exacto, exact()	68
e^(), e para uma potência	63	exclusão com operador " "	254
eff(), converter taxa nominal para efectiva	63	Exit, sair	68
eigVc(), vector eigen	64	exp(), e para uma potência	69
eigVl(), valor próprio	64	exp►list(), expressão para lista	70
elementos (nulos) vazios	274	expand(), expandir	70
elementos nulos	274	expandir, expand()	70
elementos nulos, remover	53	expansão trigonométrica, tExpand()	203
eliminar elementos nulos da lista	53	Expoente e modelo para	2
variável, DelVar	53	expoente, E	248
else if, Elseif	65	expoentes modelo para	1
else, Else	96	expr(), cadeia para expressão	72, 117
Elseif, else if	65	ExpReg, refrsessão exponencial	72
end for, EndFor	79	expressões cadeia para expressão, expr() ..	72, 117
função, EndFunc	83	expressão para lista, exp►lista()	70
loop, EndLoop	120		
programa, EndPrgm	150		
end function, EndFunc	83		
end loop, EndLoop	120		
EndWhile, terminar enquanto	220		

F			
factor(), factor	73	poissPdf()	145
factor, factor()	73	tCdf()	203
factorial, !	241	tPdf()	208
factorização QR, QR	153	χ^2 2way()	25
Fill, preencher matriz	76	χ^2 Cdf()	26
FiveNumSummary	76	χ^2 GOF()	26
floor(), floor	77	χ^2 Pdf()	27
floor, floor()	77	funções definidas pelo utilizador	50
fMax(), função máxima	77	funções e programas definidos pelo utilizador	51-52
fMin(), função mínima	78	funções e variáveis	
For	79	a copiar	32
For, for	79	funções financeiras, tvnFV()	212
for, For	79	funções financeiras, tvnI()	212
forma de escalo-linha reduzida, rref		funções financeiras, tvnN()	212
()	171	funções financeiras, tvnPmt()	213
forma de escalo-linha, ref()	161	funções financeiras, tvnPV()	213
format(), cadeia do formato	79	G	
fpart(), parte da função	80	g, gradianos	248
fracção própria, propFrac	152	gcd(), máximo divisor comum	84
fracções		geomCdf()	84
modelo para	1	geomPdf()	84
propFrac	152	Get	85, 268
fracções mistas, com propFrac() com	152	getDenom(), obter denominador	86
freqTable()	81	getKey()	86
frequência()	81	getLangInfo(), obter/apresentar	
Func, função	83	informações do idioma	91
Func, função do programa	83	getLockInfo(), testar o estado de	
função de domínio, domínio()	60	bloqueio da variável ou do	
função por ramos (2 ramos)		grupo de variáveis	91
modelo para	2	getModel(), obter definições do	
função por ramos (N-ramos)		modo	92
modelo para	3	getNum(), obter número	93
funções		GetStr	93
definidas pelo utilizador	50	getType(), get type of variable	93
função do programa, Func	83	getVarInfo(), obter/apresentar	
máxima, fMax()	77	informações das variáveis	94
mínima, fMin()	78	Goto, ir para	95
parte, fpart()	80	grupos, bloquear e desbloquear	116, 216
funções de distribuição		grupos, testar estado de bloqueio	91
binomCdf()	21, 101	guardar	
binomPdf()	22	símbolo, &	255-256
invNorm()	102	H	
invt()	102	hexadecimal	
Inv χ^2 ()	100	indicador, Oh	256
normCdf()	136		
normPdf()	136		
poissCdf()	145		

visualizar, ►Base16	21
hiperbólica	
tangente, $\tanh()$	201
hiperbólico	
arco-coseno, $\cosh^{-1}()$	36
arco-seno, $\sinh^{-1}()$	185
arco-tangente, $\tanh / ()$	202
co-seno, $\cosh()$	35
seno, $\sinh()$	185

I

identity(), matriz identidade	95
idioma	
obter informações do idioma ..	91
ifFn()	97
igual ou maior que, 	239
igual ou menor que, {	238
igual, =	237
imag(), parte imaginária	98
ImpDif(), derivada implícita	98
implicação lógica dupla, $\Leftrightarrow$	240
implicação lógica, $\Rightarrow$	240, 276
indirecta, #	247
Input, entrada	98
inString(), na cadeia	99
int(), parte inteira	99
intDiv(), divisão inteira	99
integral definido	
modelo para	6
integral indefinido	
modelo para	7
integrar, $\int$	242
interpolate(), interpolar	100
inv F()	101
inverso, $\wedge^{-1}$	253
invNorm((distribuição normal	
cumulativa inversa)	102
invNorm(), normal cumulativa	
inversa)	102
invtt()	102
Inv χ^2 ()	100
iPart(), parte inteira	102
ir para, Goto	95
irr(), taxa de retorno interno	
internal rate of return, irr()	103
isPrime(), teste da plica	103
isVoid(), teste para nulo	104

L

Lbl, definição	104
lcm, mínimo múltiplo comum	104
left(), esquerda	105
limit	
lim()	106
limit()	106
limit() ou lim(), limite	106
limite	
modelo para	7
limite máximo, limite máximo()	23, 39
LinRegBx, regressão linear	107
LinRegMx, regressão linear	108
LinRegIntervals, regressão linear ...	109
LinRegTTest	111
linSolve()	112
list►mat(), lista para matriz	113
lista para matriz, list►mat()	113
lista, contar condicionalmente itens	
numa	38
lista, contar itens em	38
ListaDelta()	52
listas	
aumentar/concatenar,	
aumentar()	17
diferença, $\Delta list()$	113
diferenças numa lista, @ list() ..	113
elementos vazios em	274
expressão para lista, exp►lista() ..	70
lista para matriz, list►mat()	113
matriz para lista, mat►lista()	121
máximo, max()	122
mid-string, mid()	125
mínimo, min()	125
nova, newList()	132
ordenar ascendente, SortA	191
ordenar descendente, SortD	192
produto cruzado, crossP()	40
produto do ponto, dotP()	62
produto, product()	151
soma cumulativa,	
SomaCumulativa()	45
soma, sum()	197-198
ln(), logaritmo natural	114
LnReg, regressão logarítmica	114
local, Local	116
Local, variável local	116

Log		nova, newMat()	133
modelo para	2	operação da linha, mRow()	127
logaritmo natural, ln()	114	ponto adição, .+	234
logaritmos	114	ponto divisão, ./	235
LogisticD, regressão logística	119	ponto multiplicação, .*	235
Loop, ciclo	120	ponto potência, .^	236
LU, decomposição inferior-superior		ponto subtração, .-	235
da matriz	121	preencher, Fill	76
		produto, product()	151
		soma cumulativa,	
		SomaCumulativa()	45
maior que, >	239	soma, sum()	197-198
mat►list(), matriz para lista	121	submatriz, subMat()	197, 199
matriz (1 × 2)		transpor, T	199
modelo para	4	troca da linha, rowSwap()	171
matriz (2 × 1)		valor próprio, eigVl()	64
modelo para	4	vector eigen, eigVc()	64
matriz (2 × 2)		max(), máximo	122
modelo para	4	máximo divisor comum, gcd()	84
matriz (m × n)		máximo, max()	122
modelo para	4	mean(), média	122
matriz de correlação, corrMat()	32	média, mean()	122
matriz identidade, identity()	95	median(), mediana	123
matriz para lista, mat►list()	121	mediana, median()	123
matrizes		MedMed, regressão da recta média-	
adição de linha, rowAdd()	170	média	123
adição e multiplicação da linha,		mensagens e códigos de erros	293
mRowAdd()	127	mid(), mid-string	125
aleatórias, randMat()	158	mid-string, mid()	125
aumentar/concatenar,		min(), mínimo	125
aumentar()	17	mínimo múltiplo comum, lcm	104
decomposição inferior-superior,		mínimo, min()	125
LU	121	mirr(), taxa de retorno interna	
determinante, det()	56	modificada	126
diagonal, diag()	56	mod(), módulo	127
dimensão da coluna, colDim()	28	modelos	
dimensão da linha, rowDim()	170	derivada ou derivada de índice	
dimensão, dim()	57	N	6
factorização QR, QR	153	expoente	1
forma de escalão-linha reduzida,		Expoente e	2
rref()	171	fracção	1
forma de escalão-linha, ref()	161	função por ramos (2 ramos)	2
identity, identity()	95	função por ramos (N-ramos)	3
lista para matriz, list►mat()	113	integral definido	6
matriz para lista, mat►list()	121	integral indefinido	7
máximo, max()	122	limite	7
mínimo, min()	125	Log	2
norma da coluna, colNorm()	29	matriz (1 × 2)	4
norma da linha, rowNorm()	170	matriz (2 × 1)	4

matriz (2 × 2)	4	notação de minutos,	250
matriz (m × n)	4	notação de segundos, "	250
primeira derivada	5	nova	
produto (P)	5	lista, newList()	132
raiz de índice N	2	matriz, newMat()	133
raiz quadrada	1	nPr(), permutações	138
segunda derivada	6	npv(), valor líquido actual	138
sistema de equações (2		nSolve(), solução numérica	139
equações)	3	nulo, teste para	104
sistema de equações (N		numérica	
equações)	3	derivada, nDeriv()	133
soma (G)	5	solução, nSolve()	139
valor absoluto	4	numérico	
modos		integral, nInt()	134
definir, setMode()	177		
módulo, mod()	127	O	
mRow(), operação da linha da		obter	
matriz	127	denominador, getDenom()	86
mRowAdd(), adição e multiplicação		número, getNum()	93
da linha da matriz	127	obter/apresentar	
multiplicar, *	231	informações das variáveis,	
MultReg	128	getVarInfo()	91, 94
MultRegIntervals()	128	OneVar, estatística de uma variável	140
MultRegTests()	129	operador da indirecta (#)	279
		operador de limite " "	254
N		operador de limite, ordem de	
na cadeia, inString()	99	avaliação	278
nand, Operador booleano	130	operadores	
nCr(), combinações	131	ordem de avaliação	278
nDerivative(), derivada numérica ..	132	Operadores booleanos	
negação, introduzir números		⇒	240, 276
negativos	279	⇔	240
newList(), nova lista	132	nand	130
newMat(), nova matriz	133	nor	135
nfMax(), função numérica máxima ..	133	not	137
nfMin(), função numérica mínima ..	133	ou	141
nInt(), integral numérico	134	xou	220
nom), converter taxa efectiva para		ord(), código de carácter numérico	143
nominal	134	ordenar	
nor, Operador booleano	135	ascendente, SortA	191
norma Frobenius, normal()	136	descendente, SortD	192
norma(), norma Frobenius	136	ou (Booleano), or	141
normCdf()	136	ou, Operador booleano	141
normPdf()	136		
not, Operador booleano	137	P	
notação de gradianos, g	248	P►Rx(), rectangular x coordenada ..	143
notação de grau/minuto/segundo ..	250	P►Ry(), rectangular y coordenada ..	143
notação de graus, °	249	parte imaginária, imag()	98

regressão da recta média-média, MedMed	123	rowSwap(), troca da linha da matriz	171
regressão de potência, PowerReg	149, 163, 165	rref(), forma de escalão-linha reduzida	171
regressão exponencial, ExpReg	72	S	
regressão linear, LinRegAx	108	sair, Exit	68
regressão linear, LinRegBx	107, 109	Se, if	96
regressão logarítmica, LnReg	114	se, If	96
regressão logística, LogisticD	119	sec(), secante	172
regressão potencial, PowerReg	149, 204	sec ⁻¹ (), secante inversa	172
regressão quadrática, QuadReg	153	sech(), secante hiperbólica	172
regressão quártica, QuartReg	155	sech ⁻¹ (), secante hiperbólica inversa	173
regressão sinusoidal, SinReg	186	segunda derivada	
regressões		modelo para	6
cúbica, CubicReg	44	seno	
exponencial, ExpReg	72	apresenta a expressão em	
logarítmica, LnReg	114	função do	182
logística, Logística	119	seno, sin()	183
MultReg	128	seq(), sequência	174
quadrática, QuadReg	153	seqGen()	174
quártica, QuartReg	155	seqn()	175
recta média-média, MedMed ..	123	SeqProd()	151
regressão de potência, PowerReg	149, 163, 165	SeqSom()	198
regressão linear, LinRegAx	108	sequence, seq()	174-175
regressão linear, LinRegBx	107, 109	sequência, seq()	174
regressão potencial, PowerReg ..	149, 204	série(), série	176
sinusoidal, SinReg	186	série, série()	176
remain(), resto	163	setMode(), definir modo	177
remover		shift(), deslocar	179
elementos nulos da lista	53	sign(), sinal	181
resolver, solve()	187	simult(), equações simultâneas	181
resposta (última), Ans	14	sin(), seno	183
resto, remain()	163	sin ⁻¹ (), arco-seno	184
resultado		sinal, sign()	181
apresenta em função do co- seno	32	sinh(), seno hiperbólico	185
apresenta em função do seno ..	182	sinh ⁻¹ (), arco-seno hiperbólico	185
resultados de duas variáveis, TwoVar	214	SinReg, regressão sinusoidal	186
resultados, estatística	193	sistema de equações (2 equações)	
right, right()	30, 66, 218	modelo para	3
rk23(), função Runge Kutta	166	sistema de equações (N equações)	
rodar(), rotate	168	modelo para	3
rodar, rotate()	168	solução, deSolve()	54
rowAdd(), adição da linha da matriz	170	solve(), resolver	187
rowDim(), dimensão da linha da matriz	170	soma (G)	
rowNorm(), norma da linha da matriz	170	modelo para	5
		soma cumulativa, SomaCumulativa()	45
		soma de pagamentos principais	246

soma dos pagamentos de juros	245	teste da plica, isPrime()	103
soma, sum()	197	Teste F de 2 amostras	82
soma, Σ ()	245	teste para nulo, isVoid()	104
SomaCumulativa(), soma		teste t, tTest	210
cumulativa	45	Teste t de regressões lineares	
SortA, ordenar ascendente	191	múltiplas	129
SortD, ordenar descendente	192	tExpand(), expansão trigonométrica	203
sqrt(), raiz quadrada	193	tInterval, t intervalo de confiança ...	205
stat.results	193	tInterval_2Samp, intervalo de	
stat.values	194	confiança t de duas	
stdDevPop(), desvio padrão da		amostras	206
população	195	tmpCnv()	206-207
stdDevSamp(), desvio padrão da		tPdf(), densidade de probabilidade	
amostra	195	student t	208
string(), expressão para cadeia	196	transpor, T	199
strings		tTest, teste t	210
right, right()	30, 66, 218	tTest_2Samp, teste t de duas	
subMat(), submatriz	197, 199	amostras	211
submatriz, subMat()	197, 199	tvmFV()	212
substituição com operador " "	254	tvmI()	212
subtrair, -	230	tvmN()	212
sum(), soma	197	tvmPmt()	213
sumIf()	198	tvmPV()	213
		TwoVar, resultados de duas variáveis	214
T			
T, transpor	199	U	
tabela de amortização, amortTbl() ..	8, 18	unidades	
tan(), tangente	199	converter	252
tan ⁻¹ (), arco-tangente	200	unitV(), vector da unidade	216
tangente, tan()	199	V	
tanh(), tangente hiperbólica	201	valor absoluto	
tanh ⁻¹ (), arco-tangente hiperbólico	202	modelo para	4
taxa de câmbio média, avgRC()	17	valor líquido actual, npv()	138
taxa de retorno interna modificada,		valor próprio, eigVl()	64
mirr()	126	valor temporal do dinheiro, juro	212
taxa efectiva, eff()	63	valor temporal do dinheiro,	
taxa nominal, nom()	134	montante do pagamento ..	213
taylor(), polinómio de Taylor	202	valor temporal do dinheiro, número	
tCdf(), probabilidade da distribuição		de pagamentos	212
student t	203	valor temporal do dinheiro, valor	
tCollect(), recolha trigonométrica ..	203	actual	213
terminar		valor temporal do dinheiro, Valor	
enquanto, EndWhile	220	futuro	212
if, EndIf	96	valores dos resultados, estatística ..	194
terminar enquanto, EndWhile	220	variação, variance()	217
terminar se, EndIf	96		
termo dominante, dominantTerm()	61		
Test_2S, Teste F de 2 amostras	82		

variáveis			
apagar todas as letras			
individuais	27	x^2 , quadrado	234
eliminar, DelVar	53	XNOR	240
local, Local	116	xou, Booleano exclusivo ou	220
variáveis, bloquear e desbloquear	91, 116, 216		
variável		Z	
criar nome a partir de uma		zeroes(), zeros	221
cadeia de caracteres	279	zeros, zeroes()	221
variável e funções		zInterval, z intervalo de confiança	224
a copiar	32	zInterval_1Prop, intervalo de	
variável local, Local	116	confiança z de uma	
varPop()	216	proporção	225
varSamp(), variação da amostra	217	zInterval_2Prop, intervalo de	
vector eigen, eigVc()	64	confiança z de duas	
vector unitário, unitV()	216	proporções	225
vectores		zInterval_2Samp, intervalo de	
produto cruzado, crossP()	40	confiança z de duas	
produto do ponto, dotP()	62	amostras	226
unidade, unitV()	216	zTest	227
visualizar vector cilíndrico,		zTest_1Prop, teste z de uma	
►Cylind	46	proporção	227
visualizar como		zTest_2Prop, teste z de duas	
ângulo decimal, ►DD	49	proporções	228
binário, ►Base2	19	zTest_2Samp, teste z de duas	
grau/minuto/segundo, ►DMS	59	amostras	229
hexadecimal, ►Base16	21		
número inteiro decimal,		Δ	
►Base10	20	Δ list(), diferença da lista	113
vector, ►Polar	145	Δ tmpCnv() [tmpCnv]	207
vector cilíndrico, ►Cylind	46		
vector esférico, ►Sphere	192	X	
visualizar como vetor rectangular,		χ^2 2way	25
►Rect	160	χ^2 Cdf()	26
visualizar dados, Disp	57	χ^2 GOF	26
visualizar grau/minuto/segundo,		χ^2 Pdf()	27
►DMS	59		
visualizar vector cilíndrico, ►Cylind	46		
visualizar vector esférico, ►Sphere	192		
visualizar vetor rectangular, ►Rect	160		
Voltar, return	166		
voltar, Return	166		
W			
warnCodes(), Warning codes	218		
when(), quando	219		
While, enquanto	220		