TI-Nspire™ CX
Manual de Referência

Saiba mais sobre a tecnologia TI através da ajuda online em education.ti.com/eguide.

Informações importantes

Excepto se indicado expressamente na Licença que acompanha um programa, Texas Instruments não dá garantia, explícita ou implícita, incluindo mas não se limitando a quaisquer garantias de comercialização e adequação a um fim particular, relativamente a quaisquer programas ou materiais de documentação e disponibiliza estes materiais unicamente numa base “tal qual”. Em nenhum caso, a Texas Instruments será responsável perante alguém por danos especiais, colaterais, incidentais, ou consequenciais em ligação com a ou provenientes da compra ou utilização destas matérias, e a responsabilidade única e exclusiva da Texas Instruments, independentemente da forma de actuação, não excederá a quantia estabelecida na licença do programa. Além disso, a Texas Instruments não será responsável por qualquer queixa de qualquer tipo apresentada contra a utilização destes materiais por terceiros.

Os produtos reais podem variar ligeiramente das imagens fornecidas.

Índice

Modelos de expressão **1**

Lista alfabética **7**

A 7
B 16
C 20
D 37
E 46
F 54
G 62
I 73
L 82
M 97
N 106
O 115
P 118
Q 125
R 128
S 143
T 163
U 176
V 177
W 178
X 181
Z 182

Símbolos **188**

TI-Nspire™ CX II - Comandos de desenho **211**

Programação de gráficos 211
Ecrã de gráficos 211
Vista e definições padrão 212
Mensagens de erro no ecrã de gráficos 213
Comandos inválidos no modo de gráficos 213
C 214
D 215
F 218
G 220
P 221
S 223
U 225

Elementos (nulos) vazios **226**

Atalhos para introduzir expressões matemáticas **228**

Hierarquia do EOS™ (Equation Operating System) **230**

TI-Nspire CX II - Funcionalidades de programação TI-Basic **232**

Recuos automáticos no Editor de Programação 232
Mensagens de erro melhoradas para TI-Basic 232

Constantes e valores **235**

Mensagens e códigos de erros **236**

Códigos de aviso e mensagens **245**

Informações gerais **247**

Ajuda online 247
Contacte a assistência técnica da TI 247
Informações da Assistência e Garantia 247

Índice remissivo **248**

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 190.

Exemplo:

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

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 (▶).

Nota: Consulte também **^ (potência)**, página 191.

Exemplo:

$$2^3 \qquad 8$$

Modelo de raiz quadrada

Teclas ctrl x²

Nota: Consulte também **√() (raiz quadrada)**, página 200.

Exemplo:

$$\sqrt{4} \qquad 2$$

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

Modelo de raiz de índice N

Teclas ctrl ^

$\sqrt[\square]{\square}$ **Nota:** Consulte também **raiz()**, página 140.

Exemplo:

$\sqrt[3]{8}$ → 2

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

Modelo de expoente e

Tecla e^x

$e^{\square}$

Exponencial natural *e* elevado à potência

Nota: Consulte também **e ^()**, página 46.

Exemplo:

e^1 → 2.71828182846

Modelo de log

Teclas ctrl 10^x

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

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

Nota: Consulte também **log()**, página 93.

Exemplo:

$\log_4(2.)$ → 0.5

Modelo de Função por ramos (2 ramos)

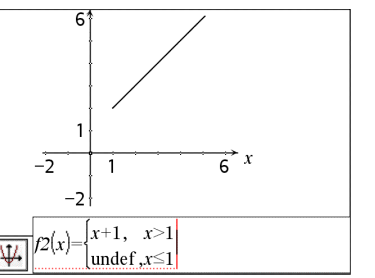
Catálogo> $\left|\square\right|_{\square}^{\square}$

$\begin{cases}\square, & \square \\ \square, & \square\end{cases}$

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 119.

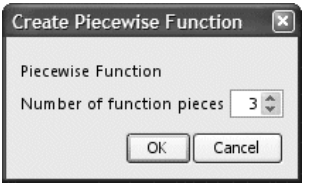
Exemplo:



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.



Nota: Consulte também **piecewise()**, página 119.

Exemplo:

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

Modelo do sistema de 2 equações

Catálogo>

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

Nota: Consulte também **sistema()**, página 163.

Exemplo:

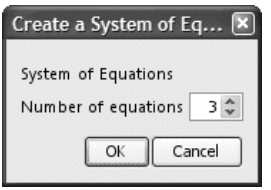
$$\text{solve}\left(\begin{cases} x+y=0 \\ x-y=5 \end{cases}, x, y\right) \quad x=\frac{5}{2} \text{ and } y=\frac{-5}{2}$$

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

Modelo do sistema de N equações

Catálogo>

Permite criar um sistema de N equações lineares. Pede N.



Nota: Consulte também **sistema()**, página 163.

Exemplo:

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

Modelo do valor absoluto

Catálogo>

|▯| **Nota:** Consulte também **abs()**, página 7.

Exemplo:

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

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:

$$30°15'10" \qquad 0.528011$$

Modelo da matriz (2 x 2)

Catálogo>

$$\begin{bmatrix} ▯ & ▯ \\ ▯ & ▯ \end{bmatrix}$$

Cria uma matriz 2 x 2.

Exemplo:

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

Modelo da matriz (1 x 2)

Catálogo>

$$\begin{bmatrix} ▯ & ▯ \end{bmatrix}.$$

Exemplo:

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

Modelo da matriz (2 x 1)

Catálogo>

$$\begin{bmatrix} ▯ \\ ▯ \end{bmatrix}$$

Exemplo:

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

Modelo da matriz (m x n)

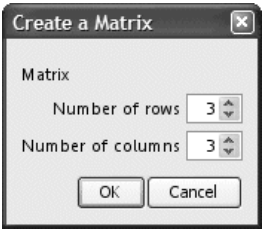
Catálogo>

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

Exemplo:

Modelo da matriz (m x n)

Catálogo>



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

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

Modelo da soma (Σ)

Catálogo>

$$\sum_{\square=\square}^{\square}(\square)$$

Nota: Consulte também **Σ() (sumSeq)**, página 201.

Exemplo:

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

Modelo do produto (Π)

Catálogo>

$$\prod_{\square=\square}^{\square}(\square)$$

Nota: Consulte também **Π () (prodSeq)**, página 200.

Exemplo:

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

Modelo da primeira derivada

Catálogo >

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

Exemplo:

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

Modelo da primeira derivada

Catálogo >

Pode utilizar o modelo da primeira derivada para calcular a primeira derivada num ponto numericamente com métodos de diferenciação automáticos.

Nota: Consulte também **d() (derivada)** , página 199.

Modelo da segunda derivada

Catálogo >

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

Pode utilizar o modelo da segunda derivada para calcular a segunda derivada num ponto numericamente com métodos de diferenciação automáticos.

Nota: Consulte também **d() (derivada)** , página 199.

Exemplo:

$$\frac{d^2}{dx^2}\left(x^3\right)\Big|_{x=3} \qquad 18$$

Modelo do integral definido

Catálogo>

$$\int_{\square}^{\square} \square \, d\square$$

Pode utilizar o modelo do integral definido para definir o integral definido numericamente com o mesmo método de nInt().

Nota: Consulte também **nInt() , página 110.**

Exemplo:

$$\int_0^{10} x^2 \, dx \qquad 333.333$$

Lista alfabética

Os itens cujos nomes não sejam alfabéticos (como +, !, e >) são listados no fim desta secção, começando (página 188). 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(*Valor1***)** ⇒*valor*

abs(*Lista1***)** ⇒*lista*

abs(*Matriz1***)** ⇒*matriz*

Devolve o valor absoluto do argumento.

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.

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

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

amortTbl() Catálogo >

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

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

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 174.

- 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*

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

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

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 202 e **bal()**, página 16.

ExprBooleana1 **and** *ExprBooleana2* ⇒*Expressão booleana*

ListaBooleana1 **and** *ListaBooleana2* ⇒*Lista booleana*

MatrizBooleana1 **and** *MatrizBooleana2* ⇒*Matriz booleana*

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

Inteiro1 **and** *Inteiro2* ⇒*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

and

Catálogo >

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(*Valor1***)** ⇒*valor*

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

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:

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

$\text{angle}(\{1+2\cdot i, 3+0\cdot i, 0-4\cdot i\})$ $\{1.10715, 0., -1.5708\}$

angle(*Lista1***)** ⇒*lista*

angle(*Matriz1***)** ⇒*matriz*

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

Catálogo >

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

ANOVA

Catálogo >

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 158).

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

Catálogo >

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 158).

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
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

Variável de saída	Descrição
stat.MSRow	Quadrados médios para o factor da linha

Saídas de interacção

Variável de saída	Descrição
stat.FInteragir	Festatística 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 ctrl (–)

Ans⇒*valor*

Devolve o resultado da expressão avaliada mais recentemente.

56	56
56+4	60
60+4	64

approx()

Catálogo >

approx(*Valor1*) ⇒*número*

Devolve a avaliação do argumentos como uma expressão com valores decimais, quando possível, independentemente do modo **Auto ou Aproximado** actual.

Isto é equivalente a introduzir o argumento e a introduzir ctrl enter.

$\text{approx}\left(\frac{1}{3}\right)$	0.333333
$\text{approx}\left(\left\{\frac{1}{3},\frac{1}{9}\right\}\right)$	{0.333333,0.111111}
$\text{approx}(\{\sin(\pi),\cos(\pi)\})$	{0.,-1.}
$\text{approx}\left(\left[\sqrt{2}\quad\sqrt{3}\right]\right)$	[1.41421 1.73205]
$\text{approx}\left(\begin{bmatrix}\frac{1}{3} & \frac{1}{9}\end{bmatrix}\right)$	[0.333333 0.111111]

approx() Catálogo >

approx(*Lista1***)** ⇒*lista*

approx(*Matriz1***)** ⇒*matriz*

Devolve uma lista ou uma *matriz* em que cada elemento foi avaliado para um valor decimal, quando possível.

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

$\text{approx}([\sqrt{2} \quad \sqrt{3}])$ $[1.41421 \quad 1.73205]$

▸approxFraction() Catálogo >

Valor ▸*approxFraction(*[*Tol*]*)*⇒*valor*

Lista ▸**approxFraction(**[*Tol*]**)**⇒*lista*

Matriz ▸**approxFraction(**[*Tol*]**)**⇒*matriz*

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

Nota: Pode introduzir esta função através da escrita de **@>approxFraction(...)** no teclado do computador.

$\frac{1}{2}+\frac{1}{3}+\tan(\pi)$ 0.833333

$0.83333333333333 \blacktriangleright \text{approxFraction}(5.\text{E}^{-}14)$ $\frac{5}{6}$

$\{\pi,1.5\} \blacktriangleright \text{approxFraction}(5.\text{E}^{-}14)$ $\left\{\frac{5419351}{1725033},\frac{3}{2}\right\}$

approxRational() Catálogo >

approxRational(*Valor*[**,** *Tol*]**)**⇒*valor*

approxRational(*Lista* [**,** *Tol*] **)** ⇒*lista*

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

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

$\text{approxRational}(0.333,5\cdot10^{-5})$ $\frac{333}{1000}$

$\text{approxRational}(\{0.2,0.33,4.125\},5.\text{E}^{-}14)$ $\left\{\frac{1}{5},\frac{33}{100},\frac{33}{8}\right\}$

arccos() Consulte cos⁻¹(), página 28.

arccosh() Consulte cosh⁻¹(), página 29.

arccot() **Consulte cot^{-1}(), página 30.**

arccoth() **Consulte $coth^{-1}$(), página 31.**

arccsc() **Consulte csc^{-1}(), página 33.**

arccsch() **Consulte $csch^{-1}$(), página 34.**

arcsec() **Consulte sec^{-1}(), página 144.**

arcsech() **Consulte $sech^{-1}$(), página 144.**

arcsin() **Consulte sin^{-1}(), página 153.**

arcsinh() **Consulte $sinh^{-1}$(), página 154.**

arctan() **Consulte tan^{-1}(), página 164.**

arctanh() **Consulte $tanh^{-1}$(), página 166.**

augment() Catálogo >

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

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

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

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*.

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \rightarrow m1$ $\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$

$\begin{bmatrix}5\\ 6\end{bmatrix} \rightarrow m2$ $\begin{bmatrix}5\\ 6\end{bmatrix}$

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.

x:=2 2

avgRC(x^2-x+2,*x*) 3.001

avgRC(x^2-x+2,*x*,.1) 3.1

avgRC(x^2-x+2,*x*,3) 6

bal() Catálogo >

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

bal(*NPmt***,** *TabelaDeDepreciação***)** ⇒*valor*

Função de amortização que calcula o saldo do plano após um pagamento especificado.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* e *PmtAt* são descritos na tabela de argumentos TVM, página 174.

NPmt especifica o número de pagamentos a partir dos quais quer os dados calculados.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* e *PmtAt* são descritos na tabela de argumentos TVM, página 174.

- 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.

bal(*NPmt*, *TabelaDeDepreciação***)** calcula o saldo após o número de pagamentos *NPmt*, baseado na tabela de amortização *TabelaDeDepreciação*. O argumento *TabelaDeDepreciação* tem de ser uma matriz no forma descrita em **amortTbl()**, página 7.

Nota: Consulte também Σ**Int()** e Σ**Prn()**, página 202.

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

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

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

bal(4,*tbl*)	1674.27

NúmeroInteiro1 ▶**Base2** ⇒*número inteiro*

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

-2^{63} 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

256▶Base2	0b100000000
0h1F▶Base2	0b11111

▸Base2 Catálogo >

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

▸Base10 Catálogo >

NúmeroInteiro1 **▸Base10** ⇒*número inteiro*

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.

0b10011▸Base10	19
0h1F▸Base10	31

▸Base16 Catálogo >

NúmeroInteiro1 **▸Base16** ⇒*número inteiro*

256▸Base16	0h100
0b111100001111▸Base16	0hF0F

▸Base16 Catálogo >

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 17.

binomCdf() Catálogo >

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≤X ≤*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.

binomCdf() **Catálogo >**

Para P(X ≤ *LimiteSuperior*), defina *LimiteInferior*=0

binomPdf() **Catálogo >**

binomPdf(*n*, *p***)** ⇒*lista*

binomPdf(*n*, *p*, *ValX***)** ⇒*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() **Catálogo >**

ceiling(*Valor1***)** ⇒*valor*

Devolve o número inteiro mais próximo que é ≥ o argumento.

O argumento pode ser um número complexo ou real.

Nota: Consulte também **floor()**.

ceiling(*Lista1***)** ⇒*lista*

ceiling(*Matriz1***)** ⇒*matriz*

Devolve uma lista ou matriz do ceiling de cada elemento.

$\text{ceiling}(.456)$ → 1.

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

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

centralDiff() **Catálogo >**

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*

$\text{centralDiff}(\cos(x),x)|x=\frac{\pi}{2}$ → -1.

centralDiff() Catálogo >

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*, predefine-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()**.

char() Catálogo >

char(*Número inteiro***)** ⇒*carácter*

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.

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

$\chi^2$2way Catálogo >

$\chi^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 158).

Para mais informações sobre o efeito dos elementos vazios numa matriz, consulte "Elementos (nulos) vazios" (página 226).

Variável de saída	Descrição
stat.χ^2	Estatística do Qui quadrado: soma (observada - prevista) ^{2} /prevista

Variável de saída	Descrição
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^2Cdf()

Catálogo >

χ^2Cdf
(
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($X \le$ *LimiteSuperior*), defina *LimiteInferior* = 0.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

χ^2GOF

Catálogo >

χ^2GOF *Lista obs*, *Lista exp*, *df*

chi2GOF *Lista obs*, *Lista exp*, *df*

χ^2GOF

Catálogo >

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 158).

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

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.CompList	Matriz de contribuições da estatística do Qui quadrado dos elementos

χ^2Pdf()

Catálogo >

χ^2Pdf(*ValX*,*df*)⇒*número* se *ValX* for um número, *lista* se *ValX* for uma lista

chi2Pdf(*ValX*,*df*)⇒*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 226).

ClearAZ

Catálogo >

ClearAZ

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

$5 \rightarrow b$	5
b	5
ClearAZ	*Done*
b	"Error: Variable is not defined"

ClearAZ

Catálogo >

Se uma ou mais variáveis estiverem bloqueadas, este comando mostra uma mensagem de erro e só elimina as variáveis desbloqueadas. Consulte **unLock, página 177.**

ClrErr

Catálogo >

ClrErr

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 119, e **Try**, página 170.

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.

Para ver um exemplo de **ClrErr**, consulte o exemplo 2 no comando **Try**, página 170.

colAugment()

Catálogo >

colAugment(*Matriz1***,** *Matriz2***)** ⇒*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*.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\rightarrow m1 \qquad \begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$$

$$\begin{bmatrix}5 & 6\end{bmatrix}\rightarrow m2 \qquad \begin{bmatrix}5 & 6\end{bmatrix}$$

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

colDim()

Catálogo >

colDim(*Matriz***)** ⇒*expressão*

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

colDim() Catálogo >

Devolve o número de colunas contidas em *Matriz*.

Nota: Consulte também **rowDim()**.

colNorm() Catálogo >

colNorm(*Matriz***)** ⇒*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()**.

$$\begin{bmatrix} 1 & -2 & 3 \\ 4 & 5 & -6 \end{bmatrix} \rightarrow mat \qquad \begin{bmatrix} 1 & -2 & 3 \\ 4 & 5 & -6 \end{bmatrix}$$

$$\text{colNorm}(mat) \qquad 9$$

conj() Catálogo >

conj(*Valor1***)** ⇒*valor*

conj(*Lista1***)** ⇒*lista*

conj(*Matriz1***)** ⇒*matriz*

Devolve o conjugado complexo do argumento.

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

$$\text{conj}(1+2\cdot i) \qquad 1-2\cdot i$$

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

constructMat() Catálogo >

constructMat
(*Expr***,***Var1***,***Var2***,***NúmLinhas***,***NúmColunas***)**
⇒*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*.

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

CopyVar

Catálogo >

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 a,c: $c(4)$	$\frac{1}{4}$
CopyVar b,c: $c(4)$	16

$aa.a:=45$	45
$aa.b:=6.78$	6.78
CopyVar $aa.$,$bb.$	*Done*
getVarInfo()	$aa.a$ "NUM" "" 0 $aa.b$ "NUM" "" 0 $bb.a$ "NUM" "" 0 $bb.b$ "NUM" "" 0

corrMat()

Catálogo >

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

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

cos()

Tecla trig

cos(*Valor1***)** ⇒*valor*

cos(*Lista1***)** ⇒*lista*

No modo de ângulo Graus:

cos(*Valor1***)** devolve o co-seno do argumento como um valor.

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

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.

$\cos\left(\left\langle\frac{\pi}{4}\right\rangle_r\right)$	0.707107
$\cos(45)$	0.707107
$\cos(\{0,60,90\})$	$\{1.,0.5,0.\}$

No modo de ângulo Gradianos:

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

No modo de ângulo Radianos:

$\cos\left(\frac{\pi}{4}\right)$	0.707107
$\cos(45°)$	0.707107

cos(*MatrizQuadrada1***)** ⇒*Matriz quadrada*

Devolve o co-seno da matriz da *MatrizQuadrada1*. Isto não é o mesmo que calcular o co-seno de cada elemento.

Quando uma função escalar f(A) operar na *MatrizQuadrada1* (A), o resultado é calculado pelo algoritmo:

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

MatrizQuadrada1 tem de ser diagnolizável. 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) =

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()

Tecla trig

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

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

$\cos^{-1}()$

Tecla trig

cos⁻¹(*Valor1***)** ⇒*valor*

cos⁻¹(*Lista1***)** ⇒*lista*

cos⁻¹(*Valor1***)** devolve o ângulo cujo co-seno é *Valor1*.

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***)** ⇒*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 diagnolizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Graus:

$\cos^{-1}(1)$ → 0.

No modo de ângulo Gradianos:

$\cos^{-1}(0)$ → 100.

No modo de ângulo Radianos:

$\cos^{-1}(\{0,0.2,0.5\})$ → $\{1.5708, 1.36944, 1.0472\}$

No modo de ângulo Radianos e Formato complexo rectangular:

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

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

cosh()

Catálogo >

cosh(*Valor1***)** ⇒*valor*

cosh(*Lista1***)** ⇒*lista*

No modo de ângulo Graus:

cosh() Catálogo >

cosh(*Valor1***)** devolve o co-seno hiperbólico do argumento.

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

cosh (*MatrizQuadrada1***)** ⇒*Matriz quadrada*

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()**.

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

$\cosh\left(\left(\frac{\pi}{4}\right)^r\right)$ 1.74671E19

No modo de ângulo Radianos:

$$\cosh\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}$$

$$\begin{bmatrix}421.255 & 253.909 & 216.905\\ 327.635 & 255.301 & 202.958\\ 226.297 & 216.623 & 167.628\end{bmatrix}$$

cosh⁻¹() Catálogo >

cosh⁻¹(*Valor1***)** ⇒*valor*

cosh⁻¹(*Lista1***)** ⇒*lista*

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

$\cosh^{-1}(\{1,2.1,3\})$ $\{0,1.37286,1.76275\}$

cosh⁻¹(*Valor1***)** devolve o co-seno hiperbólico inverso do argumento.

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 diagnolizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Radianos e Formato complexo rectangular:

$$\cosh^{-1}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}$$

$$\begin{bmatrix}2.52503+1.73485\cdot i & -0.009241-1.4908\\ 0.486969-0.725533\cdot i & 1.66262+0.623491\\ -0.322354-2.08316\cdot i & 1.26707+1.79018\end{bmatrix}$$

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

cot()

Tecla trig

cot(*Valor1***)** ⇒ *valor*

cot(*Lista1***)** ⇒ *lista*

Devolve a co-tangente de *Valor1* ou devolve uma lista das co-tangentes de todos os elementos em *Lista1*.

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.

No modo de ângulo Graus:

$\cot(45)$ → 1.

No modo de ângulo Gradianos:

$\cot(50)$ → 1.

No modo de ângulo Radianos:

$\cot(\{1,2.1,3\})$ → $\{0.642093,-0.584848,-7.01525\}$

$\cot^{-1}()$

Tecla trig

$\cot^{-1}$(*Valor1***)** ⇒*valor*

$\cot^{-1}$(*Lista1***)** ⇒*lista*

Devolve o ângulo cuja co-tangente é *Valor1* ou devolve uma lista com as co-tangentes inversas 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.

No modo de ângulo Graus:

$\cot^{-1}(1)$ → 45.

No modo de ângulo Gradianos:

$\cot^{-1}(1)$ → 50.

No modo de ângulo Radianos:

$\cot^{-1}(1)$ → 0.785398

coth()

Catálogo >

coth(*Valor1***)** ⇒*valor*

coth(*Lista1***)** ⇒*lista*

Devolve a co-tangente hiperbólica de *Valor1* ou devolve uma lista das co-tangentes hiperbólicas de todos os elementos de *Lista1*.

$\coth(1.2)$ → 1.19954

$\coth(\{1,3.2\})$ → $\{1.31304,1.00333\}$

coth⁻¹() Catálogo >

coth⁻¹(*Valor1***)** ⇒*valor*

coth⁻¹(*Lista1***)** ⇒*lista*

Devolve a co-tangente hiperbólica inversa de *Valor1* ou devolve uma lista com as co-tangentes hiperbólicas inversas de cada elemento de *Lista1*.

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

$\coth^{-1}(3.5)$	0.293893
$\coth^{-1}(\{-2,2.1,6\})$	$\{-0.549306,0.518046,0.168236\}$

count() Catálogo >

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

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

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

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

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 226.

$\text{count}(2,4,6)$	3
$\text{count}(\{2,4,6\})$	3
$\text{count}\left(2,\{4,6\},\begin{bmatrix}8 & 10\\ 12 & 14\end{bmatrix}\right)$	7

countif() Catálogo >

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

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

Critérios podem ser:

- Um valor, uma expressão ou uma cadeia.

$\text{countIf}(\{1,3,\text{"abc"},\text{undef},3,1\},3)$	2

Conta o número de elementos igual a 3.

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

Conta o número de elementos igual a “def.”

countif() Catálogo >

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 226.

Nota: Consulte também **sumIf()**, página 162 e **frequency()**, página 59.

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

Conta 1 e 3.

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

Conta 3, 5, e 7.

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

Conta 1, 3, 7 e 9.

cPolyRoots() Catálogo >

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 na forma expandida. Não utilize formatos não expandidos, como, por exemplo, $y^2 \cdot y+1$ ou $x \cdot x+2 \cdot x+1$

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

Nota: Consulte também **polyRoots()**, página 121.

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

cPolyRoots(y^3+1,y) {-1,0.5−0.866025·*i*,0.5+0.866025·*i*}

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

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

crossP() Catálogo >

crossP(*Lista1*, *Lista2*) ⇒*lista*

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

crossP({0.1,2.2,-5},{1,-0.5,0}) {-2.5,-5.,-2.25}

crossP() Catálogo >

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.

$\text{crossP}([1\ \ 2\ \ 3],[4\ \ 5\ \ 6])$ $[-3\ \ 6\ \ -3]$

$\text{crossP}([1\ \ 2],[3\ \ 4])$ $[0\ \ 0\ \ -2]$

csc() Tecla trig

csc(*Valor1***)** ⇒*valor*

csc(*Lista1***)** ⇒*lista*

Devolve a co-secante de *Valor1* ou devolve uma lista com as co-secantes de todos os elementos em *Lista1*.

No modo de ângulo Graus:

$\csc(45)$ 1.41421

No modo de ângulo Gradianos:

$\csc(50)$ 1.41421

No modo de ângulo Radianos:

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

csc⁻¹() Tecla trig

csc⁻¹(*Valor1***)** ⇒ *valor*

csc⁻¹(*Lista1***)** ⇒ *lista*

Devolve o ângulo cuja co-secante é *Valor1* ou devolve uma lista com as co-secantes inversas 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.

No modo de ângulo Graus:

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

No modo de ângulo Gradianos:

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

No modo de ângulo Radianos:

csc⁻¹()

Tecla trig

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

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

csch()

Catálogo >

csch(*Valor1***)** ⇒ *valor*

csch(*Lista1***)** ⇒ *lista*

Devolve a co-secante hiperbólica de *Valor1* ou devolve uma lista das co-secantes hiperbólicas de todos os elementos de *Lista1*.

$\operatorname{csch}(3)$ 0.099822

$\operatorname{csch}(\{1,2.1,4\})$ $\{0.850918, 0.248641, 0.036644\}$

csch⁻¹()

Catálogo >

csch⁻¹(*Valor***)** ⇒ *valor*

csch⁻¹(*Lista1***)** ⇒ *lista*

Devolve a co-secante hiperbólica inversa de *Valor1* ou devolve uma lista com as cosesecantes hiperbólicas inversas de cada elemento de *Lista1*.

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

$\operatorname{csch}^{-1}(1)$ 0.881374

$\operatorname{csch}^{-1}(\{1,2.1,3\})$ $\{0.881374, 0.459815, 0.32745\}$

CubicReg

Catálogo >

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 158).

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 numéricos ou de cadeias 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 226).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: a · x^3+b · x^2+c · 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 *Lista X* modificada utilizada na regressão com base em restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.YReg	Lista de pontos de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.FreqReg	Lista de frequências correspondentes a *stat.XReg* e *stat.YReg*

cumulativeSum() Catálogo >

cumulativeSum(*Lista1***)⇒***lista***

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

cumulativeSum({1,2,3,4}) {1,3,6,10}

cumulativeSum() Catálogo >

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.

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 226.

$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}$$

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

Cycle Catálogo >

Cycle

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.

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

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

▸Cylind Catálogo >

Vector **▸Cylind**

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, ∠θ, z].

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

```
[2  2  3]▸Cylind
          [2.82843  ∠0.785398  3.]
```

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°

▸Decimal

Catálogo >

Número1 ▸*Decimal* ⇒*valor*

Lista1 ▸**Decimal** ⇒*valor*

Matriz1 ▸**Decimal** ⇒*valor*

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

Catálogo >

Define *Var* = *Expressão*

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

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

Os parâmetros como, por exemplo, *Parâm1*, 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* → *Função*(*Parâm1,Parâm2*).

```
Define g(x,y)=2·x−3·y                          Done
g(1,2)                                          -4
1→a: 2→b: g(a,b)                                -4
Define h(x)=when(x<2,2·x−3,-2·x+3)             Done
h(-3)                                           -9
h(4)                                            -5
```

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

Bloco

EndFunc

```
Define g(x,y)=Func                              Done
                If x>y Then
                Return x
                Else
                Return y
                EndIf
                EndFunc
g(3,-7)                                          3
```

Define

Catálogo >

Define *Programa*(*Parâm1*, *Parâm2*, ...) = **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 39, e **Define LibPub**, página 40.

```
Define g(x,y)=Prgm
              If x>y Then
              Disp x," greater than ",y
              Else
              Disp x," not greater than ",y
              EndIf
              EndPrgm
                                        Done
g(3,-7)
                   3 greater than -7
                                        Done
```

Define LibPriv

Catálogo >

Define LibPriv *Var* = *Expressão*

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

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

Bloco

EndFunc

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

Bloco

EndPrgm

Define LibPriv **Catálogo >**

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 38, e **Define LibPub**, página 40.

Define LibPub **Catálogo >**

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 38, e **Define LibPriv**, página 39.

deltaList() **Consulte ΔList(), página 89.**

DelVar

Catálogo >

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

DelVar *Var*.

Elimina a variável ou o grupo de variáveis especificado da memória.

Se uma ou mais variáveis estiverem bloqueadas, este comando mostra uma mensagem de erro e só elimina as variáveis desbloqueadas. Consulte **unLock**, página 177.

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.

$2 \rightarrow a$ → 2

$(a+2)^2$ → 16

DelVar a → *Done*

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

$aa.a := 45$ → 45

$aa.b := 5.67$ → 5.67

$aa.c := 78.9$ → 78.9

getVarInfo() →

aa.a	"NUM"	"⬚"
aa.b	"NUM"	"⬚"
aa.c	"NUM"	"⬚"

DelVar *aa*. → *Done*

getVarInfo() → "NONE"

delVoid()

Catálogo >

delVoid(*Lista1***)**⇒*lista*

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

Para mais informações sobre os elementos vazios, consulte página 226.

delVoid({1,void,3}) → {1,3}

det()

Catálogo >

det(*MatrizQuadrada*[**,** *Tolerância*]**)**⇒*expressão*

Apresenta o determinante de *MatrizQuadrada*.

$\det\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$ → −2

$\begin{bmatrix}1.\text{E}20 & 1\\ 0 & 1\end{bmatrix} \rightarrow mat1$ → $\begin{bmatrix}1.\text{E}20 & 1\\ 0 & 1\end{bmatrix}$

$\det(mat1)$ → 0

$\det(mat1,.1)$ → $1.\text{E}20$

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 ctrl enter 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 · **max(dim(***MatrizQuadrada***))** · **rowNorm(***MatrizQuadrada***)**

diag(*Lista***)** ⇒*matriz*

diag(*MatrizLinha***)** ⇒*matriz*

diag(*MatrizColuna***)** ⇒*matriz*

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

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

diag(*MatrizQuadrada***)** ⇒*MatrizLinha*

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

MatrizQuadrada tem de ser quadrada.

$\begin{bmatrix}4&6&8\\1&2&3\\5&7&9\end{bmatrix}$ → $\begin{bmatrix}4&6&8\\1&2&3\\5&7&9\end{bmatrix}$

$\text{diag}(Ans)$ → $[4\ \ 2\ \ 9]$

dim(*Lista***)** ⇒*número inteiro*

Devolve a dimensão de *Lista*.

$\dim(\{0,1,2\})$ → 3

dim(*Matriz***)** ⇒*lista*

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

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

dim()

Catálogo >

dim(*Cadeia***)** ⇒*número inteiro*

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

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

Disp

Catálogo >

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

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.

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

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

DispAt

Catálogo >

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

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

O número da linha pode ser especificado como uma expressão.

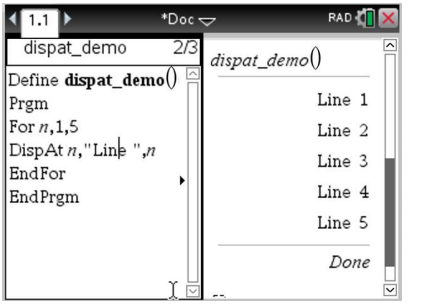
Tenha em atenção que o número da linha não se destina ao ecrã inteiro, mas à área imediatamente a seguir ao comando/programa.

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.

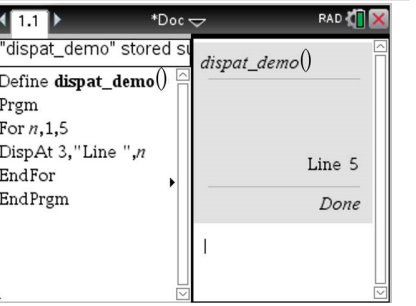
DispAte Disp podem ser utilizados no mesmo programa.

```
DispAt
```

Exemplo



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

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)

▸DMS **Catálogo >**

Valor ▸*DMS*

Lista ▸**DMS**

Matriz ▸**DMS**

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 206) 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.

No modo de ângulo Graus:

$(45.371)\blacktriangleright\text{DMS}$ $45°22'15.6"$

$(\{45.371,60\})\blacktriangleright\text{DMS}$ $\{45°22'15.6",60°\}$

dotP() Catálogo >

dotP(*Lista1*, *Lista2*) ⇒*expressão*

Devolve o produto do "ponto" de duas listas.

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

dotP(*Vector1*, *Vector2*) ⇒*expressão*

Devolve o produto do "ponto" de dois vectores.

Ambos têm de ser vectores da linha ou da coluna.

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

E

e^() Tecla eˣ

e^(*Valor1***)** ⇒*valor*

Devolve **e** elevado à potência *Valor1*.

e^{1} 2.71828

e^{3^2} 8103.08

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

Nota: Premir eˣ para ver e ^(é diferente de premir o carácter **E** 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*

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

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

e^(*MatrizQuadrada1***)** ⇒*MatrizQuadrada*

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()**.

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

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

eff() Catálogo >

eff(*TaxaNominal,CpY***)** ⇒*valor*

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 110.

eff(5.75,12) 5.90398

eigVc() Catálogo >

eigVc(*MatrizQuadrada***)** ⇒*matriz*

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, …, x_n]:

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

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.

No Formato complexo rectangular:

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

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

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

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

eigVl() Catálogo >

eigVl(*MatrizQuadrada***)** ⇒*lista*

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

No modo de formato complexo rectangular:

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

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

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

eigVl() **Catálogo >**

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.

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

Else **Consulte If, página 74.**

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

EndFor **Consulte For, página 57.**

EndFunc **Consulte Func, página 61.**

EndIf **Consulte If, página 74.**

EndLoop **Consulte Loop, página 96.**

EndPrgm **Consulte Prgm, página 123.**

EndTry **Consulte Try, página 170.**

EndWhile **Consulte While, página 180.**

euler ( ) Catálogo >

euler(*Expr*, *Var*, *depVar*, {*Var0, VarMax*}, *depVar0*, *VarStep* [, *eulerStep*]**)** ⇒*matriz*

euler(*SystemOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *eulerStep*]**)** ⇒*matriz*

euler(*ListOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *eulerStep*]**)** ⇒*matriz*

Utiliza o método de Euler para resolver o sistema

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

com *depVar*(*Var0*)=*depVar0* no intervalo [*Var0*,*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.

Equação diferencial:

y'=0.001*y*(100-y) e y(0)=10

$$\text{euler}(0.001\cdot y\cdot(100-y),t,y,\{0,100\},10,1)$$
$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9 & 11.8712 & 12.9174 & 14.042 \end{bmatrix}$$

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

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 $y2(0)=5$

euler ()

Catálogo >

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*).

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*.

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

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. & 5. \\ 2. & 1. & 1. & 3. & 27. & 243. \\ 5. & 10. & 30. & 90. & 90. & -2070. \end{bmatrix}$$

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.

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

lum:=127	127
Send "SET COLOR.BLUE eval(lum)"	*Done*

Repor o elemento azul para DESLIGADO.

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

```
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.

```
fadein()                                     Done
```

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

```
n:=0.25                                      0.25
m:=8                                            8
n·m                                             2.
Send "SET COLOR.BLUE ON TIME eval(n·m)"
                                             Done
iostr.SendAns    "SET COLOR.BLUE ON TIME 2"
```

Nota: Ver também **Get** (página 63), **GetStr** (página 71) e **Send** (página 145).

Exit

Catálogo >

Exit

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

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

Listagem de funções:

Exit

Catálogo >

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→temp
  For i,1,100,1
  temp+i→temp
  If temp>20 Then
  Exit
  EndIf
  EndFor
  EndFunc

g()                                21
```

exp()

Tecla e^x

exp(*Valor1***)** ⇒*valor*

Devolve **e** elevado à potência *Valor1*.

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.

e^1 → 2.71828

e^{3^2} → 8103.08

exp(*Lista1***)** ⇒*lista*

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

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

exp(*MatrizQuadrada1***)** ⇒*MatrizQuadrada*

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 diagnolizável. O resultado contém sempre os números de ponto flutuante.

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

expr() Catálogo >

expr(*Cadeia***)** ⇒*expressão*

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

"Define cube(x)=x^3" → *funcstr*	
	"Define cube(x)=x^3"
expr(*funcstr*)	*Done*
cube(2)	8

ExpReg Catálogo >

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

Calcula a regressão exponencialy = a ·(b)ˣa partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

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 numéricos ou de cadeias 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 226).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: a ·(b)ˣ
stat.a, stat.b	Parâmetros da regressão

Variável de saída	Descrição
stat.r^2	Coeficiente de determinação linear para dados transformados
stat.r	Coeficiente 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 *Lista X* modificada utilizada na regressão com base em restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.YReg	Lista de pontos de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.FreqReg	Lista de frequências correspondentes a *stat.XReg* e *stat.YReg*

F

factor() Catálogo >

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 demorarar mais de um século.

factor(152417172689)	123457·1234577
isPrime(152417172689)	false

Para parar um cálculo manualmente,

- **Dispositivo portátil:** Manter pressionada a tecla [on] e pressionar [enter] 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.

factor()

Catálogo >

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()

Catálogo >

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

FCdf(*LimiteInferior*, *LimiteSuperior*, *dfNumer*, *dfDenom***)** ⇒*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($X \le$ *LimiteSuperior*), definir *LimiteInferior* = 0.

Fill

Catálogo >

Fill *Valor, VarMatriz* ⇒*matriz*

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

matrizVar já tem de existir.

Fill *Valor, VarLista* ⇒*lista*

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

VarLista já tem de existir.

$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \rightarrow amatrix$ $\quad \begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$

Fill 1.01,*amatrix* $\quad$ *Done*

amatrix $\quad \begin{bmatrix}1.01 & 1.01\\1.01 & 1.01\end{bmatrix}$

$\{1,2,3,4,5\} \rightarrow alist$ $\quad \{1,2,3,4,5\}$

Fill 1.01,*alist* $\quad$ *Done*

alist $\quad \{1.01,1.01,1.01,1.01,1.01\}$

FiveNumSummary

Catálogo >

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

FiveNumSummary

Catálogo >

Fornece uma versão abreviada da estatística de 1 variável na lista X. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

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 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. Para mais informações sobre os elementos vazios, consulte página 226.

Variável de saída	Descrição
stat.MinX	Mínimo dos valores x
stat.Q_1X	1º quartil de x
stat.MedianX	Mediana de x
stat.Q_3X	3º quartil de x
stat.MaxX	Máximo dos valores x

floor()

Catálogo >

floor(*Valor1***)** ⇒*número inteiro*

floor(−2.14) −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() Catálogo >

floor(*Lista1***)** ⇒*lista*

floor(*Matriz1***)** ⇒*matriz*

Devolve uma lista ou matriz do floor de cada elemento.

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

$\text{floor}\left(\left\{\frac{3}{2},0,-5.3\right\}\right)$ $\{1.0,-6.\}$

$\text{floor}\begin{bmatrix}1.2 & 3.4\\ 2.5 & 4.8\end{bmatrix}$ $\begin{bmatrix}1. & 3.\\ 2. & 4.\end{bmatrix}$

For Catálogo >

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()=Func                    Done
  Local tempsum,step,i
  0→tempsum
  1→step
  For i,1,100,step
  tempsum+i→tempsum
  EndFor
  EndFunc
g()                                5050
```

format() Catálogo >

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

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

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

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"

format() Catálogo >

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.

E[n]: Formato de engenharia. n é o número de dígitos após o primeiro dígito significante. O exponente 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() Catálogo >

fPart(*Expr1***)** ⇒*expressão*

fPart(*Lista1***)** ⇒*lista*

fPart(*Matriz1***)** ⇒*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.

fPart(-1.234)	-0.234
fPart({1,-2.3,7.003})	{0,-0.3,0.003}

FPdf() Catálogo >

FPdf(*ValX***, *dfNumer*, *dfDenom*)** ⇒*número*
se *ValX* for um número, *lista* se *ValX* for uma lista

FPdf() Catálogo >

Calcula a probabilidade da distribuição F no *ValX* para o *dfNumer* (graus de liberdade) e o *dfDenom* especificados.

freqTable▸list() Catálogo >

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 226.

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() Catálogo >

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*.

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

frequency() Catálogo >

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*.

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)>**?**)}.

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 226.

Na aplicação Listas e Folha de cálculo, pode utilizar um intervalo de células no lugar de ambos os argumentos.

Nota: Consulte também **countIf()**, página 31.

4 elementos da *Lista de dados* são >2.5 e ≤ 4.5

3 elementos da *Lista de dados* são >4.5

O elemento "hello" é uma cadeia e não pode ser colocado em nenhum lote definido.

FTest_2Samp Catálogo >

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 158).

ou H_a: σ1 > σ2, defina *Hipótese*>0

Para H_a: σ1 ≠ σ2 (predefinição), defina *Hipótese* =0

Para H_a: σ1 < σ2, defina *Hipótese*<0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

Variável de saída	Descrição
stat.F	Estatística Û 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" = n1-1
stat.dfDenom	graus de liberdade do "denominador"= n2-1
stat.sx1, stat.sx2	Desvios padrão da amostra das sequências de dados em *Lista 1* e *Lista 2*
stat.x1_bar stat.x2_bar	Médias da amostra das sequência de dados em *Lista 1* e *Lista 2*
stat.n1, stat.n2	Tamanho das amostras

Func

Bloco

EndFunc

Modelo para criar uma função definida pelo utilizador.

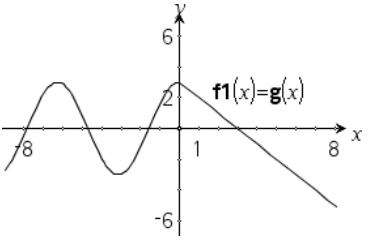
Bloco pode ser uma declaração, uma série de declarações separadas pelo carácter ":" ou uma série de declarações em linhas separadas. A função pode utilizar a função **Return** para devolver um resultado específicos.

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.

Definir uma função por ramos:

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

Resultado do gráfico g(x)



gcd() Catálogo >

gcd(*Valor1, Valor2***)** ⇒*expressão*

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.

$\gcd(18,33)$ 3

gcd(*Lista1, Lista2***)** ⇒*lista*

Devolve os máximos divisores comuns dos elementos correspondentes em *Lista1* e *Lista2*.

$\gcd(\{12,14,16\},\{9,7,5\})$ $\{3,7,1\}$

gcd(*Matriz1, Matriz2***)** ⇒*matriz*

Devolve os máximos divisores comuns dos elementos correspondentes em *Matriz1* e *Matriz2*.

$\gcd\left(\begin{bmatrix}2 & 4\\ 6 & 8\end{bmatrix},\begin{bmatrix}4 & 8\\ 12 & 16\end{bmatrix}\right)$ $\begin{bmatrix}2 & 4\\ 6 & 8\end{bmatrix}$

geomCdf() Catálogo >

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

geomCdf(*p*,*LimiteSuperior***)**para P(1≤X ≤*LimiteSuperior*) ⇒*número* se *LimiteSuperior* for um número, *lista* se *LimiteSuperior* for uma lista

Calcula uma probabilidade geométrica cumulativa do *LimiteInferior* ao *LimiteSuperior* com a probabilidade de sucesso especificada *p*.

Para P(X ≤ *LimiteSuperior*), defina *LimiteInferior* = 1.

geomPdf(*p*, *ValX*) ⇒*número* se *ValX* 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 *lightval*	*Done*
lightval	0.347922

Incorporar o pedido READ no comando **Get**.

Get "READ BRIGHTNESS",*lightval*	*Done*
lightval	0.378441

Define *func*(*arg1*, ...*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 71 e **Send**, página 145.

getDenom() Catálogo >

getDenom(*Fracção1***)** ⇒*valor*

Transforma o argumento numa expressão que tem um denominador comum simplificado e, em seguida, devolve o denominador.

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

getKey() Catálogo >

codeTouch([0|1]) ⇒ Cadeiadevolvida

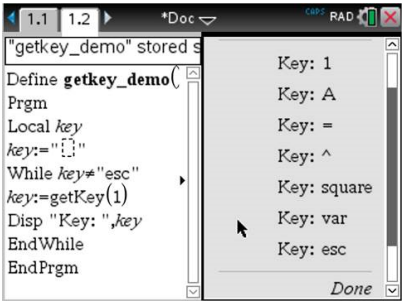
Descrição:codeTouch() - permite a um programa em TI-Basic obter introduções com o teclado - portátil, computador de secretária e emulador no computador de secretária.

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 alterna 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() Catálogo >

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() Catálogo >

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 92, e**unLock**, página 177.

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*

getMode(*NúmeroInteiroNomeModo***)** ⇒*valor*

getMode(0) ⇒*lista*

getMode(*NúmeroInteiroNomeModo***)** devolve um valor que representa a definição actual do modo *NúmeroInteiroNomeModo*.

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 147.

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

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
Formato vectorial	6	**1** =Rectangular, **2** =Cilíndrico, **3** =Esférico
Base	7	**1** =Decimal, **2** =Hex, **3** =Binário

getNum() Catálogo >

getNum(*Fracção1***)** ⇒*valor*

Transforma o argumento numa expressão que tem um denominador comum simplificado e, em seguida, devolve o numerador.

$x:=5: y:=6$	6
$\text{getNum}\left(\frac{x+2}{y-3}\right)$	7
$\text{getNum}\left(\frac{2}{7}\right)$	2
$\text{getNum}\left(\frac{1}{x}+\frac{1}{y}\right)$	11

GetStr Hub Menu

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

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

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 63 e **Send**, página 145.

Para exemplos, ver **Get**.

getType() Catálogo >

getType(*var***)**⇒*cadeia de texto*

Apresenta uma cadeia de texto que indica o tipo de dados da variável *var*.

Se *var* 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"
DelVar *temp*	*Done*
$\text{getType}(temp)$	"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()**.

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.

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()	$\begin{bmatrix} x & \text{"NUM"} & \text{"}\square\text{"} & 1 \\ y & \text{"LIST"} & \text{"LibPriv "} & 0 \\ z & \text{"FUNC"} & \text{"LibPub "} & 0 \end{bmatrix}$
getVarInfo(*tmp3*)	"Error: Argument must be a string"
getVarInfo("tmp3")	$\begin{bmatrix} volcyl2 & \text{"NONE"} & \text{"LibPub "} & 0 \end{bmatrix}$

$a:=1$	1
$b:=\begin{bmatrix}1 & 2\end{bmatrix}$	$\begin{bmatrix}1 & 2\end{bmatrix}$
$c:=\begin{bmatrix}1 & 3 & 7\end{bmatrix}$	$\begin{bmatrix}1 & 3 & 7\end{bmatrix}$
$vs:=$getVarInfo()	$\begin{bmatrix} a & \text{"NUM"} & \text{"}\square\text{"} & 0 \\ b & \text{"MAT"} & \text{"}\square\text{"} & 0 \\ c & \text{"MAT"} & \text{"}\square\text{"} & 0 \end{bmatrix}$
$vs[1]$	$\begin{bmatrix}1 & \text{"NUM"} & \text{"}\square\text{"} & 0\end{bmatrix}$
$vs[1,1]$	1
$vs[2]$	"Error: Invalid list or matrix"
$vs[2,1]$	$\begin{bmatrix}1 & 2\end{bmatrix}$

Goto

Catálogo >

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→temp
            1→i
            Lbl top
            temp+i→temp
            If i<10 Then
            i+1→i
            Goto top
            EndIf
            Return temp
            EndFunc
g()                                55
```

▸Grad

Catálogo >

Expr1 **▸Grad** ⇒*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)▸Grad                         (1.66667)ᵍ
```

No modo de ângulo Radianos:

```
(1.5)▸Grad                         (95.493)ᵍ
```

I

identity ()

Catálogo >

identity(*Número inteiro***)** ⇒ *matriz*

Devolve a matriz identidade com uma dimensão de *Número inteiro*.

Número inteiro tem de ser um número natural.

```
identity(4)
⎡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.

Define $g(x)$=Func
If $x<0$ Then
Return x^2
EndIf
EndFunc
	Done

$g(-2)$	4

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
If $x<0$ Then
Return $-x$
Else
Return x
EndIf
EndFunc
	Done

$g(12)$	12
$g(-12)$	12

If

Catálogo >

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 -x
              ElseIf x≥0 and x≠10 Then
              Return x
              ElseIf x=10 Then
              Return 3
              EndIf
              EndFunc
                                        Done
g(-4)                                      4
g(10)                                      3
```

ifFn ()

Catálogo >

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 avaliada 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 *ExprBooleana*.

```
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 ()

Catálogo >

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.

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

Value_If_false não é especificado. Undef é utilizado.

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

Um elemento seleccionado de *Value_If_true*. Um elemento seleccionado de *Value_If_unknown*.

imag()

Catálogo >

imag(*Value1***)** ⇒ *valor*

Devolve a parte imaginária do argumento.

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

imag(*Lista1***)** ⇒ *lista*

Devolve uma lista de partes imaginárias dos elementos.

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

imag(*Matriz1***)** ⇒ *matriz*

Devolve uma matriz das partes imaginárias dos elementos.

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

indirecta

Consultar #(), página 203.

inString ()

Catálogo >

inString(*CadeiaDeOrigem***,** *CadeiaSecundária*[**,** *Início*]**)** ⇒ *número inteiro*

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*).

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

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

inString ()

Catálogo >

Se *CadeiaDeOrigem* não contiver *CadeiaSecundária* ou *Início* for > o comprimento de *CadeiaDeOrigem*, devolve zero.

int ()

Catálogo >

int(*Value***)** ⇒ *número inteiro*
int(*List1***)** ⇒ *lista*
int(*Matrix1***)** ⇒ *matriz*

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.

int(-2.5)	-3.
int([-1.234 0 0.37])	[-2. 0 0.]

intDiv ()

Catálogo >

intDiv(*Number1***, *Number2***)** ⇒ *número inteiro*
intDiv(*List1***, *List2***)** ⇒ *lista*
intDiv(*Matrix1***, *Matrix2***)** ⇒ *matriz*

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.

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

interpolar ()

Catálogo >

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

Esta função efectua o seguinte:

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

$rk:=\text{rk23}(-3 \cdot y+6 \cdot t+5, t, y, \{0,10\}, 5, 1)$

[0. 1. 2. 3. 4.
5. 3.19499 5.00394 6.99957 9.00593 10

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

interpolar ()

Catálogo >

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 um número ou uma lista de números.

Utilize a função de interpolação() para calcular os valores de função para xvaluelist:

$xvaluelist:=\text{seq}(i,i,0,10,0.5)$

$\{0,0.5,1.,1.5,2.,2.5,3.,3.5,4.,4.5,5.,5.5,6.,6.5,$

$xlist:=\text{mat}\blacktriangleright\text{list}(rk[1])$

$\{0.,1.,2.,3.,4.,5.,6.,7.,8.,9.,10.\}$

$ylist:=\text{mat}\blacktriangleright\text{list}(rk[2])$

$\{5.,3.19499,5.00394,6.99957,9.00593,10.9978$

$yprimelist:=-3\cdot y+6\cdot t+5|y=ylist \text{ and } t=xlist$

$\{-10.,1.41503,1.98819,2.00129,1.98221,2.006$

$\text{interpolate}(xvaluelist,xlist,ylist,yprimelist)$

$\{5.,2.67062,3.19499,4.02782,5.00394,6.00011$

invχ²()

Catálogo >

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 *Área* debaixo da curva

invF()

Catálogo >

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 *Área* debaixo da curva.

invBinom() Catálogo >

invBinom
(*CumulativeProb*,*NumTrials*,*Prob*,
OutputForm)⇒ *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%?

$\text{invBinom}\left(0.77,30,\frac{1}{6}\right)$ → 6

$\text{invBinom}\left(0.77,30,\frac{1}{6},1\right)$ →
$\begin{bmatrix} 5 & 0.616447 \\ 6 & 0.776537 \end{bmatrix}$

invBinomN() Catálogo >

invBinomN(*CumulativeProb*,*Prob*,
NumSuccess,*OutputForm*)⇒ *escalar* ou *matriz*

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*).

OutputForm=**0**, apresenta o resultado como uma escalar (predefinição).

OutputForm=**1**, apresenta o resultado como uma 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?

$\text{invBinomN}(0.01,0.7,49)$ → 86

$\text{invBinomN}(0.01,0.7,49,1)$ →
$\begin{bmatrix} 85 & 0.010451 \\ 86 & 0.00709 \end{bmatrix}$

invNorm() Catálogo >

invNorm(*Área*[,μ[,σ]]**)**

Calcula 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***)** ⇒ *número inteiro*
iPart(*List1***)** ⇒ *lista*
iPart(*Matrix1***)** ⇒ *matriz*

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.

$\text{iPart}(-1.234)$	$-1.$
$\text{iPart}\left(\left\{\frac{3}{2},-2.3,7.003\right\}\right)$	$\{1.,-2.,7.\}$

irr() Catálogo >

irr(*CF0*,*ListaCF* [,*FreqCF*]**)** ⇒ *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 102.

$list1:=\{6000,-8000,2000,-3000\}$	$\{6000,-8000,2000,-3000\}$
$list2:=\{2,2,2,1\}$	$\{2,2,2,1\}$
$\text{irr}(5000,list1,list2)$	-4.64484

isPrime() Catálogo >

isPrime(*Número***)** ⇒ *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.

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 *nextprim*(*n*)=	Func	*Done*
	Loop	
	n+1 → *n*	
	If isPrime(*n*)	
	Return *n*	
	EndLoop	
	EndFunc	
nextprim(7)		11

isVoid() Catálogo >

isVoid(*Var***)** ⇒ *Expressão constante booleana*
isVoid(*Expr***)** ⇒ *Expressão constante booleana*
isVoid(*List***)** ⇒ *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 226.

a:=_	_
isVoid(*a*)	true
isVoid({1,_,3})	{false,true,false}

Lbl

Catálogo >

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→temp
  1→i
  Lbl top
  temp+i→temp
  If i<10 Then
  i+1→i
  Goto top
  EndIf
  Return temp
  EndFunc
g()                                 55
```

lcm()

Catálogo >

lcm(*Número1***,** *Número2***)** ⇒*expressão*

lcm(*Lista1***,** *Lista2***)** ⇒*lista*

lcm(*Matriz1***,** *Matriz2***)** ⇒*matriz*

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.

$$\text{lcm}(6,9) \qquad 18$$

$$\text{lcm}\left(\left\{\frac{1}{3},-14,16\right\},\left\{\frac{2}{15},7,5\right\}\right) \qquad \left\{\frac{2}{3},14,80\right\}$$

left()

Catálogo >

left(*CadeiaDeOrigem* **[,** *Num* **])** ⇒*cadeia*

Devolve os caracteres *Num* mais à esquerda contidos na cadeia de caracteres *CadeiaDeOrigem*.

```
left("Hello",2)                    "He"
```

left() Catálogo >

Se omitir *Num*, devolve todos os caracteres de *CadeiaDeOrigem*.

left(*Lista1* [**,** *Num*]**)** ⇒*lista*

Devolve os elementos *Num* mais à esquerda em *Lista1*.

Se omitir *Num*, devolve todos os elementos de *Lista1*.

left(*Comparação***)** ⇒*expressão*

Devolve o lado esquerdo de uma equação ou desigualdade.

left({1,3,-2,4},3) {1,3,-2}

libShortcut() Catálogo >

libShortcut(*CadeiaDoNomeDaBiblioteca***,** *CadeiaDoNomeDoAtalho* [**,** *MarcadorDeBibPriv*]**)**⇒*lista de variáveis*

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 26.

Para eliminar um grupo de variáveis, consulte **DelVar**, página 41.

Este exemplo assume um documento de biblioteca actualizado e guardado adequadamente denominado **linalg2** que contém objectos definidos como *clearmat*, *gauss1* e *gauss2*.

getVarInfo("linalg2")

clearmat	"FUNC"	"LibPub "
gauss1	"PRGM"	"LibPriv "
gauss2	"FUNC"	"LibPub "

libShortcut("linalg2","la") {*la.clearmat*,*la.gauss2*}

libShortcut("linalg2","la",1) {*la.clearmat*,*la.gauss1*,*la.gauss2*}

LinRegBx Catálogo >

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 158).

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 numéricos ou de cadeias 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 226).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: a+b·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 *Lista X* modificada utilizada na regressão com base nas restrições de $Freq$, $Lista\ de\ categorias$ e $Incluir\ categorias$
stat.YReg	Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de $Freq$, $Lista\ de\ categorias$ e $Incluir\ categorias$
stat.FreqReg	Lista de frequências correspondentes a $stat.XReg$ e $stat.YReg$

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

Calcula a regressão linear y = m ·x+b a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

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 numéricos ou de cadeias 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 226).

Variável de saída	**Descrição**
stat.RegEqn	Equação de regressão: m ·x+b
stat.m, 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 *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.YReg	Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*

Variável de saída	Descrição
stat.FreqReg	Lista de frequências correspondentes a *stat.XReg* e *stat.YReg*

LinRegtIntervals

Catálogo >

LinRegtIntervals *X*,*Y*[,*F*[,**0**[,*NívC*]]]

Para declive. Calcula o intervalo de confiança de nível C do declive.

LinRegtIntervals *X*,*Y*[,*F*[,**1**,*ValX*[,*NívC*]]]

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 158).

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 226).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: a+b·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 ·XVal

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=α+βx. Testa a hipótese nula H_0:β=0 (equivalentemente, ρ=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: $\beta\neq0$ e $\rho\neq0$ (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 158).

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

Variável de saída	**Descrição**
stat.RegEqn	Equação de regressão: $a + b \cdot x$
stat.t	*t*-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	Coeficiente de determinação
stat.r	Coeficiente de correlação
stat.Resid	Resíduos da regressão

linSolve() Catálogo >

linSolve(*SistemaDeEquaçõesLineares*, *Var1*, *Var2*, ...**)** ⇒*lista*

linSolve(*EquaçãoLinear1* **and** *EquaçãoLinear2* **e** ..., *Var1*, *Var2*, ...**)** ⇒*lista*

linSolve({*EquaçãoLinear1*, *EquaçãoLinear2*, ...**}**, *Var1*, *Var2*, ...**)** ⇒*lista*

linSolve(*SistemaDeEquaçõesLineares*, **{***Var1*, *Var2*, ...**})** ⇒*lista*

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".

$\text{linSolve}\left(\begin{cases}2\cdot x+4\cdot y=3\\5\cdot x-3\cdot y=7\end{cases},\{x,y\}\right)$ $\left\{\frac{37}{26},\frac{1}{26}\right\}$

$\text{linSolve}\left(\begin{cases}2\cdot x=3\\5\cdot x-3\cdot y=7\end{cases},\{x,y\}\right)$ $\left\{\frac{3}{2},\frac{1}{6}\right\}$

$\text{linSolve}\left(\begin{cases}apple+4\cdot pear=23\\5\cdot apple-pear=17\end{cases},\{apple,pear\}\right)$ $\left\{\frac{13}{3},\frac{14}{3}\right\}$

$\text{linSolve}\left(\begin{cases}apple\cdot 4+\frac{pear}{3}=14\\-apple+pear=6\end{cases},\{apple,pear\}\right)$ $\left\{\frac{36}{13},\frac{114}{13}\right\}$

ΔList() Catálogo >

ΔList(*Lista1***)** ⇒*lista*

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.

$\Delta\text{List}\{20,30,45,70\}$ $\{10,15,25\}$

list▸mat() Catálogo >

list▸mat(*Lista* [**,** *elementosPorLinha*]**)** ⇒*matriz*

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.

$\text{list▸mat}(\{1,2,3\}) \quad \begin{bmatrix}1 & 2 & 3\end{bmatrix}$

$\text{list▸mat}(\{1,2,3,4,5\},2) \quad \begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 0\end{bmatrix}$

ln() Teclas ctrl e^x

ln(*Valor1***)** ⇒*valor*

ln(*Lista1***)** ⇒*lista*

Devolve o logaritmo natural do argumento.

Para uma lista, devolve os logaritmos naturais dos elementos.

$\ln(2.) \quad 0.693147$

Se o modo do formato complexo for Real:

$\ln(\{-3,1.2,5\})$

"Error: Non−real calculation"

Se o modo do formato complexo for Rectangular:

$\ln(\{-3,1.2,5\}) \quad \{1.09861+3.14159\cdot i, 0.182322, 1.60944\}$

ln(*MatrizQuadrada1***)** ⇒*MatrizQuadrada*

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.

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

No modo de ângulo Radianos e Formato complexo rectangular:

$$\ln\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$

$$\begin{bmatrix}1.83145+1.73485\cdot i & 0.009193-1.49086\\ 0.448761-0.725533\cdot i & 1.06491+0.623491\\ -0.266891-2.08316\cdot i & 1.12436+1.79018\cdot\end{bmatrix}$$

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

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

Calcula a regressão logarítmica y = a+b·ln(x) a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

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 numéricos ou de cadeias 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 226).

Variável de saída	**Descrição**
stat.RegEqn	Equação de regressão: a+b·ln(x)
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), y)
stat.Resid	Resíduos associados ao modelo logarítmico
stat.ResidTrans	Resíduos associados ao ajuste linear dos dados transformados
stat.XReg	Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*

Variável de saída	Descrição
stat.YReg	Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.FreqReg	Lista de frequências correspondentes a *stat.XReg* e *stat.YReg*

Local

Catálogo >

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

Catálogo >

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*.

```
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
```

Nota: O comando **Bloquear** (**Lock**) apaga o histórico de Anular/Repetir quando aplicado a variáveis desbloqueadas.

Consulte **unLock**, página 177, e **getLockInfo ()**, página 69.

log() Teclas ctrl 10^x

log (*Valor1* [**,** *Valor2*]**)** ⇒*valor*

log (*Lista1* [**,** *Valor2*]**)** ⇒*lista*

Devolve o logaritmo -*Valor2* base do primeiro argumento.

Nota: Consulte também **Modelo do logaritmo**, página 2.

Para uma lista, devolve o logaritmo -*Valor2* base dos elementos.

Se omitir o segundo argumento, 10 é utilizado como a base.

$\log_{10}(2.)$ 0.30103

$\log_{4}(2.)$ 0.5

$\log_{3}(10)-\log_{3}(5)$ 0.63093

Se o modo do formato complexo for Real:

$\log_{10}(\{-3,1.2,5\})$

"Error: Non−real calculation"

Se o modo do formato complexo for Rectangular:

$\log_{10}(\{-3,1.2,5\})$

$\{0.477121+1.36438\cdot i,0.079181,0.69897\}$

log (*MatrizQuadrada1* [**,** *Valor*]**)** ⇒*MatrizQuardrada*

Devolve o logaritmo *Valor* base da matriz de *MatrizQuadrada1*. Isto não é mesmo que calcular o logaritmo *Valor* base de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

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

Se omitir o argumento base, 10 é utilizado como a base.

No modo de ângulo Radianos e Formato complexo rectangular:

$$\log_{10}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}$$

$$\begin{bmatrix}0.795387+0.753438\cdot i & 0.003993-0.6474\\ 0.194895-0.315095\cdot i & 0.462485+0.2707\\ -0.115909-0.904706\cdot i & 0.488304+0.7774\end{bmatrix}$$

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

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

Calcula a regressão logísticay = (c/(1+a ·e^-bx))a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

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 numéricos ou de cadeias 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 226).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: c/(1+a ·e^-bx)
stat.a, stat.b, stat.c	Parâmetros de regressão
stat.Resid	Resíduos da regressão
stat.XReg	Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.YReg	Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.FreqReg	Lista de frequências correspondentes a *stat.XReg* e *stat.YReg*

LogisticD *X*, *Y* [, [*Repetições*], [*Freq*] [, *Categoria*, *Incluir*]]

Calcula a regressão logística y = (c/(1+a ·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 158).

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 numéricos ou de cadeias 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 226).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: c/(1+a ·e^-bx)+d)
stat.a, stat.b, stat.c, stat.d	Parâmetros de regressão

Variável de saída	Descrição
stat.Resid	Resíduos da regressão
stat.XReg	Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.YReg	Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.FreqReg	Lista de frequências correspondentes a *stat.XReg* e *stat.YReg*

Loop

Catálogo >

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 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
```

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*.

$MatrizI \cdot Matrizu = Matrizp \cdot matriz$

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 ctrl enter 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 · max(dim(*Matriz*)) · rowNorm (*Matriz*)

O algortimo de factorização **LU** utiliza a articulação parcial com as trocas de linhas.

$$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix}$$

LU *m1*,*lower*,*upper*,*perm* — *Done*

lower
$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{5}{6} & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}$$

upper
$$\begin{bmatrix} 6 & 12 & 18 \\ 0 & 4 & 16 \\ 0 & 0 & 1 \end{bmatrix}$$

perm
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

M

mat▸lis t(*Matriz***)** ⇒*lista*

Devolve uma lista preenchida com os elementos em *Matriz*. Os elementos são copiados de *Matriz* linha por linha.

Nota: Pode introduzir esta função através da escrita de **mat@>list(**...**)** no teclado do computador.

$$\text{mat}\blacktriangleright\text{list}\left(\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}\right) \qquad \{1,2,3\}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$\text{mat}\blacktriangleright\text{list}(m1) \qquad \{1,2,3,4,5,6\}$$

max() Catálogo >

max(*Valor1*, *Valor2*) ⇒*expressão*

max(*Lista1*, *Lista2*) ⇒*lista*

max(*Matriz1*, *Matriz2*) ⇒*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.

$\max(2.3,1.4)$ → 2.3

$\max(\{1,2\},\{-4,3\})$ → $\{1,3\}$

max(*Lista*) ⇒*expressão*

Devolve o elemento máximo em *lista*.

$\max(\{0,1,-7,1.3,0.5\})$ → 1.3

max(*Matriz1*) ⇒*matriz*

Devolve um vector da linha com o elemento máximo de cada coluna em *Matriz1*.

$\max\begin{bmatrix}1 & -3 & 7\\ -4 & 0 & 0.3\end{bmatrix}$ → $\begin{bmatrix}1 & 0 & 7\end{bmatrix}$

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte página 226.

Nota: Consulte também **min()**.

mean() Catálogo >

mean(*Lista* [, *freqList*]) ⇒*expressão*

Devolve a média dos elementos em *Lista*.

$\text{mean}(\{0.2,0,1,-0.3,0.4\})$ → 0.26

$\text{mean}(\{1,2,3\},\{3,2,1\})$ → $\frac{5}{3}$

Cada elemento de *ListaFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

mean(*Matriz1* [, *MatrizFreq*]) ⇒*matriz*

No Formato de vector rectangular:

Devolve um vector da linha da média de todas as colunas em *Matriz1*.

$\text{mean}\begin{bmatrix}0.2 & 0\\ -1 & 3\\ 0.4 & -0.5\end{bmatrix}$ → $\begin{bmatrix}-0.133333 & 0.833333\end{bmatrix}$

$\text{mean}\begin{bmatrix}\frac{1}{5} & 0\\ -1 & 3\\ \frac{2}{5} & \frac{-1}{2}\end{bmatrix}$ → $\begin{bmatrix}\frac{-2}{15} & \frac{5}{6}\end{bmatrix}$

$\text{mean}\left(\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix},\begin{bmatrix}5 & 3\\ 4 & 1\\ 6 & 2\end{bmatrix}\right)$ → $\begin{bmatrix}\frac{47}{15} & \frac{11}{3}\end{bmatrix}$

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 226.

median() Catálogo >

median(*Lista*[**,** *ListaFreq*]**)⇒***expressão*

Devolve a mediana dos elementos em *Lista*.

Cada elemento de *ListFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

$\text{median}(\{0.2,0,1,-0.3,0.4\}) \quad 0.2$

median(*Matriz1*[**,** *MatrizFreq*]**)⇒***matriz*

Devolve um vector em linha com as medianas das colunas da *Matriz1*.

Cada elemento de *MatrizFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Matriz1*.

$\text{median}\begin{bmatrix} 0.2 & 0 \\ 1 & -0.3 \\ 0.4 & -0.5 \end{bmatrix} \quad [0.4 \quad -0.3]$

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 226.

MedMed Catálogo >

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

Calcula a recta média-médiay = (m ·x+b)a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

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 numéricos ou de cadeias 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 226).

Variável de saída	Descrição
stat.RegEqn	Equação da recta mediana-mediana: m ·x+b
stat.m, stat.b	Parâmetros do modelo
stat.Resid	Resíduos da recta mediana-mediana
stat.XReg	Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.YReg	Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.FreqReg	Lista de frequências correspondentes a *stat.XReg* e *stat.YReg*

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("Hello there",2)	"ello there"
mid("Hello there",7,3)	"the"
mid("Hello there",1,5)	"Hello"
mid("Hello there",1,0)	""

mid() Catálogo >

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.

$\text{mid}(\{9,8,7,6\},3)$ $\{7,6\}$

$\text{mid}(\{9,8,7,6\},2,2)$ $\{8,7\}$

$\text{mid}(\{9,8,7,6\},1,2)$ $\{9,8\}$

$\text{mid}(\{9,8,7,6\},1,0)$ $\{\}$

mid(*ListaDaCadeiaDeOrigem*, *Início* [, *Contagem*]**)** ⇒*lista*

Devolve as cadeias *Contagem* da lista de cadeias *ListaDaCadeiaDeOrigem*, começando pelo número de elementos *Início*.

$\text{mid}(\{\text{"A"},\text{"B"},\text{"C"},\text{"D"}\},2,2)$ $\{\text{"B"},\text{"C"}\}$

min() Catálogo >

min(*Valor1*, *Valor2***)** ⇒*expressão*

min(*Lista1, Lista2***)** ⇒*lista*

min(*Matriz1, Matriz2***)** ⇒*matriz*

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.

$\min(2.3,1.4)$ 1.4

$\min(\{1,2\},\{-4,3\})$ $\{-4,2\}$

min(*Lista***)** ⇒*expressão*

Devolve o elemento mínimo de *Lista*.

$\min(\{0,1,-7,1.3,0.5\})$ -7

min(*Matriz1***)** ⇒*matriz*

Devolve um vector da linha com o elemento mínimo de cada coluna em *Matriz1*.

$\min\begin{bmatrix}1 & -3 & 7\\ -4 & 0 & 0.3\end{bmatrix}$ $\begin{bmatrix}-4 & -3 & 0.3\end{bmatrix}$

Nota: Consulte também **max().**

mirr()

Catálogo >

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 80.

$list1:=\{6000,-8000,2000,-3000\}$	$\{6000,-8000,2000,-3000\}$
$list2:=\{2,2,2,1\}$	$\{2,2,2,1\}$
$\text{mirr}(4.65,12,5000,list1,list2)$	13.41608607

mod()

Catálogo >

mod(*Valor1***,** *Valor2***)** ⇒*expressão*

mod(*Lista1***,** *Lista2***)** ⇒*lista*

mod(*Matriz1***,** *Matriz2***)** ⇒*matriz*

Devolve o primeiro módulo de argumentos do segundo argumento conforme definido pelas identidades:

mod(x,0) = x

mod(x,y) = x − y 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.

$\text{mod}(7,0)$	7
$\text{mod}(7,3)$	1
$\text{mod}(-7,3)$	2
$\text{mod}(7,-3)$	−2
$\text{mod}(-7,-3)$	−1
$\text{mod}(\{12,-14,16\},\{9,7,-5\})$	$\{3,0,-4\}$

mod()

Catálogo >

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 135

mRow()

Catálogo >

mRow(*Valor***,** *Matriz1***,** *Índice***)** ⇒*matriz*

Devolve uma cópia de *Matriz1* com cada elemento na linha *Índice* de *Matriz1* multiplicado por *Valor*.

$$\text{mRow}\left(\frac{-1}{3},\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix},2\right) \qquad \begin{bmatrix}1 & 2\\ -1 & \frac{-4}{3}\end{bmatrix}$$

mRowAdd()

Catálogo >

mRowAdd(*Valor***,** *Matriz1***,** *Índice1***,** *Índice2***)** ⇒*matriz*

Devolve uma cópia de *Matriz1* com cada elemento na linha *Índice2* de *Matriz1* substituído por:

Valor · *linha Índice1* + *linha Índice2*

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

MultReg

Catálogo >

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 158).

Todas as listas têm de ter dimensões iguais.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: b0+b1 ·x1+b2 ·x2+ ...
stat.b0, stat.b1, ...	Parâmetros de regressão

Variável de saída	Descrição
stat.R^2	Coeficiente de determinação múltipla
stat.$\hat{y}$Lista	$\hat{y}$Lista = b0+b1 ·x1+ …
stat.Resid	Resíduos da regressão

MultRegIntervals

Catálogo >

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 158).

Todas as listas têm de ter dimensões iguais.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: b0+b1 ·x1+b2 ·x2+ …
stat.$\hat{y}$	Um ponto prevê: $\hat{y}$ = b0 + b1 · xl + … para *ListaDeValoresX*
stat.dfError	Erro dos graus de liberdade
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 *Y*, *X1*[,*X2*[,*X3*,...[,*X10*]]]

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 158).

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

Saídas

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: b0+b1 ·x1+b2 ·x2+ ...
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
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

Variável de saída	Descrição
stat.$\hat{y}$Lista	$\hat{y}$Lista = b0+b1 ·x1+ . . .
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*

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*
⇒*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.

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}

nand

Teclas ctrl =

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).

nCr()

Catálogo >

nCr(*Valor1***,** *Valor2***)** ⇒*expressão*

Para o número inteiro *Valor1* e *Valor2* com *Valor1* ≥ *Valor2* ≥ 0, **nCr()** é o número de combinações de coisas de *Valor1* retiradas de *Valor2* de uma vez. (Isto também é conhecido como um coeficiente binomial.)

$\mathrm{nCr}(z,3)|z=5$ → 10

$\mathrm{nCr}(z,3)|z=6$ → 20

nCr(*Valor***, 0)**⇒**1**

nCr(*Valor***,** *NúmeroInteiroNeg***)**⇒**0**

nCr(*Valor***,** *NúmeroInteiroPos***)** ⇒ *Valor* · **(***Valor* −**1)...**

(*Valor* −*NúmeroInteiroPos* **+1)/** *NúmeroInteiroPos***!**

nCr(*Valor***,** *NúmeroNãoInteiro***)** ⇒*expressão* **!/**

((*Valor* −*NúmeroNãoInteiro***)!** · *NúmeroNãoInteiro* **!)**

nCr(*Lista1***,** *Lista2***)** ⇒*lista*

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.

$\mathrm{nCr}(\{5,4,3\},\{2,4,2\})$ → $\{10,1,3\}$

nCr(*Matriz1***,** *Matriz2***)** ⇒*matriz*

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.

$\mathrm{nCr}\left(\begin{bmatrix}6 & 5\\ 4 & 3\end{bmatrix},\begin{bmatrix}2 & 2\\ 2 & 2\end{bmatrix}\right)$ → $\begin{bmatrix}15 & 10\\ 6 & 3\end{bmatrix}$

nDerivative() Catálogo >

nDerivative(*Expr1*,*Var*=*Valor* [,*Ordem*])⇒*valor*

nDerivative(*Expr1*,*Var*[,*Ordem*]) | *Var*=*Valor*⇒*valor*

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.

Se a variável *Var* não contiver um valor numérico, tem de fornecer *Valor*.

Ordem da derivada tem de ser **1** ou **2**.

Nota: O algoritmo nDerivative() tem uma limitação: funciona recursivamente através da expressão não simplificada, computação do valor numérico da primeira derivada (e a segunda, se aplicável) e a avaliação de cada subexpressão, que pode conduzir a um resultado imprevisto.

Considere o exemplo da direita. A primeira derivada de x·(x^2+x)^(1/3) em x=0 é igual a 0. No entanto, como a primeira derivada da subexpressão (x^2+x)^(1/3) está indefinida em x=0, e este valor é utilizado para calcular a derivada da expressão total, **nDerivative()** reporta o resultado como indefinido e apresenta uma mensagem de aviso.

Se encontrar esta limitação, verifique a solução graficamente. Pode também tentar com **centralDiff()**.

$\text{nDerivative}(|x|,x=1)$ 1

$\text{nDerivative}(|x|,x)|x=0$ undef

$\text{nDerivative}(\sqrt{x-1},x)|x=1$ undef

$\text{nDerivative}\left(x\cdot\left(x^2+x\right)^{\frac{1}{3}},x,1\right)|x=0$ undef

$\text{centralDiff}\left(x\cdot\left(x^2+x\right)^{\frac{1}{3}},x\right)|x=0$ 0.000033

newList() Catálogo >

newLis t(*ElementosNum***)** ⇒*lista*

Devolve uma lista com uma dimensão de *ElementosNum*. Cada elemento é zero.

$\text{newList}(4)$ $\{0,0,0,0\}$

newMat() Catálogo >

newMa t(*LinhaNum***,** *ColunasNum***)** ⇒*matriz*

Devolve uma matriz de zeros com a dimensão *LinhasNum* por *ColunasNum*.

newMat(2,3)	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

nfMax() Catálogo >

nfMax(*Expr***,** *Var***)** ⇒*valor*

nfMax(*Expr***,** *Var, LimiteInferior***)** ⇒*valor*

nfMax(*Expr***,** *Var, LimiteInferior***,** *LimiteSuperior***)** ⇒*valor*

nfMax(*Expr, Var***) | *LimiteInferior*≤*Var* ≤*LimiteSuperior* ⇒*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*].

nfMax(-x²−2·x−1,x)	-1.
nfMax(0.5·x³−x−2,x,-5,5)	5.

nfMin() Catálogo >

nfMin(*Expr***,** *Var***)** ⇒*valor*

nfMin(*Expr***,** *Var, LimiteInferior***)** ⇒*valor*

nfMin(*Expr***,** *Var, LimiteInferior***,** *LimiteSuperior***)** ⇒*valor*

nfMin(*Expr, Var***) | *LimiteInferior*≤*Var* ≤*LimiteSuperior* ⇒*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*].

nfMin(x²+2·x+5,x)	-1.
nfMin(0.5·x³−x−2,x,-5,5)	-5.

nInt() Catálogo >

nInt(*Expr1, Var, Inferior, Superior***)** ⇒*expressão*

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 ∫(*Expr1*, *Var*, *Inferior*, *Superior*). Esta aproximação é uma média ponderada de alguns valores de amostra da expressão a integrar no intervalo *Inferior* <*Var* <*Superior*.

$\text{nInt}\left(e^{-x^2},x,-1,1\right)$ 1.49365

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.

$\text{nInt}\left(\cos(x),x,-\pi,\pi,1.\text{E}-12\right)$ $-1.04144\text{E}-12$

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.

$\text{nInt}\left(\text{nInt}\left(\frac{e^{-x\cdot y}}{\sqrt{x^2-y^2}},y,-x,x\right),x,0,1\right)$ 3.30423

nom() Catálogo >

nom(*TaxaEfectiva,CpY***)** ⇒*valor*

$\text{nom}(5.90398,12)$ 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 47.

nor Teclas ctrl =

ExprBooleana1 **nor** *ExprBooleana2* devolve *expressão booleana*

ListaBooleana1 **nor** *ListaBooleana2* devolve

nor

Teclas ctrl =

lista booleana

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*⇒*número inteiro*

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.

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 or 4	7
3 nor 4	-8
{1,2,3} or {3,2,1}	{3,2,3}
{1,2,3} nor {3,2,1}	{-4,-3,-4}

norm()

Catálogo >

norm(*Matriz***)**⇒*expressão*

norm(*Vector***)**⇒*expressão*

Apresenta a norma Frobenius.

$\text{norm}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right)$	5.47723
$\text{norm}\left(\begin{bmatrix}1 & 2\end{bmatrix}\right)$	2.23607
$\text{norm}\left(\begin{bmatrix}1\\ 2\end{bmatrix}\right)$	2.23607

normCdf()

Catálogo >

normCdf(*LimiteInferior***,***LimiteSuperior***[,**μ

[,σ]])⇒*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 ≤ *LimiteSuperior*), defina *LimiteInferior* = **-9E999**.

normPdf(*ValX* [,μ[,σ]]**)** ⇒*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.

no t *ExprBooleana* ⇒*Expressão booleana*

Devolve falso, verdadeiro ou uma forma simplificada do argumento.

not (2≥3)	true
not 0hB0▸Base16	0hFFFFFFFFFFFFFF4F
not not 2	2

não *NúmeroInteiro1* ⇒*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).

No modo base Hex:

Importante: Zero, não a letra O.

not 0h7AC36	0hFFFFFFFFFF853C9

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.

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 17.

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

nPr(*Valor1***,** *Valor2***)** ⇒*expressão*

Para o número inteiro *Valor1* e *Valor2* com *Valor1* ≥ *Valor2* ≥ 0, **nPr()** é o número de permutações de coisas de *Valor1* retiradas de *Valor2* de uma vez.

$\text{nPr}(z,3)|z=5$ → 60

$\text{nPr}(z,3)|z=6$ → 120

$\text{nPr}(\{5,4,3\},\{2,4,2\})$ → $\{20,24,6\}$

$\text{nPr}\left(\begin{bmatrix}6&5\\4&3\end{bmatrix},\begin{bmatrix}2&2\\2&2\end{bmatrix}\right)$ → $\begin{bmatrix}30&20\\12&6\end{bmatrix}$

nPr(*Valor***, 0)** ⇒**1**

nPr(*Valor***,** *NúmeroInteiroNeg***)**
⇒ **1/((***Valor* **+1)** · **(***Valor* **+2)** … **(***Valor* −*NúmeroInteiroNeg***))**

nPr(*Valor***,** *NúmeroInteiroPos***)**
⇒*Valor* ·**(***Valor* −**1)… (***Valor* −*NúmeroInteiroPos* **+1)**

nPr(*Valor***,** *NúmeroNãoInteiro***)** ⇒*Valor***! / (***Valor* −*NúmeroNãoInteiro***)!**

nPr(*Lista1***,** *Lista2***)** ⇒*lista*

$\text{nPr}(\{5,4,3\},\{2,4,2\})$ → $\{20,24,6\}$

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(*Matriz1***,** *Matriz2***)** ⇒*matriz*

$\text{nPr}\left(\begin{bmatrix}6&5\\4&3\end{bmatrix},\begin{bmatrix}2&2\\2&2\end{bmatrix}\right)$ → $\begin{bmatrix}30&20\\12&6\end{bmatrix}$

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.

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.

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.

$list1:=\{6000,-8000,2000,-3000\}$	$\{6000,-8000,2000,-3000\}$
$list2:=\{2,2,2,1\}$	$\{2,2,2,1\}$
$\text{npv}(10,5000,list1,list2)$	4769.91

nSolve() Catálogo >

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:

$\text{nSolve}(x^2+5\cdot x-25=9,x)$	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.

variável

– ou –

variável = número real

Por exemplo, x é válido e logo é x=3.

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 cadeira "nenhuma solução encontrada."

$\text{nSolve}(x^2+5\cdot x-25=9,x)|x<0$ -8.84429

$\text{nSolve}\left(\frac{(1+r)^{24}-1}{r}=26,r\right)|r>0 \text{ and } r<0.25$ 0.006886

$\text{nSolve}(x^2=-1,x)$ "No solution found"

O

OneVar [**1,**] *X* [**,** [*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 158.)

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.

OneVar

Catálogo >

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 226.

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.σx	Desvio padrão da população de x
stat.n	Número de pontos de dados
stat.MinX	Mínimo dos valores x
stat.Q_1X	1º quartil de x
stat.MedianX	Mediana de x
stat.Q_3X	3º quartil de x
stat.MaxX	Máximo de valores x
stat.SSX	Soma de quadrados de desvios da média de x

or (ou)

Catálogo >

ExprBooleana1 **or** *ExprBooleana2* devolve *expressão booleana*

ListaBooleana1 **or** *ListaBooleana2* devolve *lista booleana*

MatrizBooleana1 **or** *MatrizBooleana2* devolve *matriz booleana*

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

Define $g(x)$=Func	*Done*
If $x\le 0$ or $x\ge 5$	
Goto *end*	
Return $x\cdot 3$	
Lbl *end*	
EndFunc	
$g(3)$	9
$g(0)$	*A function did* not return *a value*

or (ou) Catálogo >

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 17.

Nota: Consulte **xor.**

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***)** ⇒*número inteiro*

ord(*Lista1***)** ⇒*lista*

ord("hello")	104
char(104)	"h"
ord(char(24))	24
ord({"alpha","beta"})	{97,98}

ord() Catálogo >

Devolve o código numérico do primeiro carácter na cadeia de caracteres *Cadeia* ou uma lista dos primeiros caracteres de cada elemento da lista.

P

P▸Rx() Catálogo >

P▸Rx(*rExpr*, θ*Expr*) ⇒*expressão*

P▸Rx(*rList*, θ*List*) ⇒*lista*

P▸Rx(*rMatrix*, θ*Matrix*) ⇒*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:

$$P\blacktriangleright Rx(4,60°) \qquad 2.$$

$$P\blacktriangleright Rx\left(\{-3,10,1.3\},\left\{\frac{\pi}{3},\frac{-\pi}{4},0\right\}\right) \qquad \{-1.5,7.07107,1.3\}$$

P▸Ry() Catálogo >

P▸Ry(*rValue*, θ*Value*) ⇒*valor*

P▸Ry(*rList*, θ*List*) ⇒*lista*

P▸Ry(*rMatrix*, θ*Matrix*) ⇒*matriz*

Devolve a coordenada y 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.

Nota: Pode introduzir esta função através da escrita de **P@>Ry(...)** no teclado do computador.

No modo de ângulo Radianos:

$$P\blacktriangleright Ry(4,60°) \qquad 3.4641$$

$$P\blacktriangleright Ry\left(\{-3,10,1.3\},\left\{\frac{\pi}{3},\frac{-\pi}{4},0\right\}\right) \qquad \{-2.59808,-7.07107,0.\}$$

PassErr

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 24, e **Try**, página 170.

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.

Para ver um exemplo de **PassErr**, consulte o exemplo 2 no comando **Try**, página 170.

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.

Nota: Consulte também **Modelo de Função por ramos**, página 3.

Define $p(x):=\begin{cases} x, & x>0 \\ \text{undef}, & x\le 0 \end{cases}$ *Done*

$p(1)$ 1

$p(-1)$ undef

poissCdf
(
λ**,***LimiteInferior*,*LimiteSuperior***)**⇒*número*
se *LimiteInferior* e *LimiteSuperior* forem números, *lista* se *LimiteInferior* e *LimiteSuperior* forem listas

poissCdf(λ**,***LimiteSuperior***)**(para P(0≤X

≤*LimiteSuperior*)⇒*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 ≤ *LimiteSuperior*), defina *LimiteInferior*=0

poissPdf(λ**,** *ValX***)** ⇒*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.

Vector **▶Polar**

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

Apresenta o *vector* em forma polar [r∠θ]. O vector tem de ser de dimensão 2 e pode ser uma linha ou uma coluna.

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 132.

$[1 \quad 3.]\blacktriangleright\text{Polar} \qquad [3.16228 \quad \angle 71.5651]$

ValorComplexo **▶Polar**

Apresenta *VectorComplexo* em forma polar.

- O modo de ângulo Graus devolve (r∠θ).
- 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.

No modo de ângulo Radianos:

$(3+4\cdot i)\blacktriangleright\text{Polar} \qquad e^{0.927295\cdot i}\cdot 5$

$\left(\left(4 \angle \frac{\pi}{3}\right)\right)\blacktriangleright\text{Polar} \qquad e^{1.0472\cdot i}\cdot 4.$

No modo de ângulo Gradianos:

▶ Polar

Catálogo >

Nota: Tem de utilizar os parêntesis para uma entrada polar (r∠θ).

$(4\cdot i)\blacktriangleright\text{Polar}$ $(4\angle 100.)$

No modo de ângulo Graus:

$(3+4\cdot i)\blacktriangleright\text{Polar}$ $(5\angle 53.1301)$

polyEval()

Catálogo >

polyEval(*Lista1***,** *Expr1***)** ⇒*expressão*

polyEval(*Lista1***,** *Lista2***)** ⇒*expressão*

Interpreta o primeiro argumento como o coeficiente de um polinómio de grau descendente e devolve o polinómio avaliado para o valor do segundo argumento.

$\text{polyEval}(\{1,2,3,4\},2)$ 26

$\text{polyEval}(\{1,2,3,4\},\{2,-7\})$ $\{26,-262\}$

polyRoots()

Catálogo >

polyRoots(*Poli***,***Var***)** ⇒*lista*

polyRoots(*ListaDeCoeficientes***)** ⇒*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 na forma expandida. Não utilize formatos não expandidos, como, por exemplo, $y^2\cdot y+1$ ou $x\cdot x+2\cdot x+1$

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

Nota: Consulte também **cPolyRoots()**, página 32.

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

$\text{cPolyRoots}(y^3+1,y)$ $\{-1,0.5-0.866025\cdot i,0.5+0.866025\cdot i\}$

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

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

PowerReg

Catálogo >

PowerReg *X*,*Y* [**,** *Freq*] [**,** *Categoria*,

Incluir]]

Calcula a regressão de potênciay = (a · (x)^b)nas listas X e Y com a frequência $Freq$. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

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 numéricos ou de cadeias 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 226).

Variável de saída	**Descrição**
stat.RegEqn	Equação de regressão: a ·(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 *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*

Variável de saída	Descrição
stat.YReg	Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.FreqReg	Lista de frequências correspondentes a *stat.XReg* e *stat.YReg*

Prgm — Catálogo >

Prgm

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.

Calcule o GCD e visualize os resultados intermédios.

```
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 Π ( ), página 200.

Produto (PI) — Consulte Π (), página 200.

product() — Catálogo >

product(*Lista* [**,** *Início* [**,** *fim*]]**)**
⇒*expressão*

```
product({1,2,3,4})                  24
product({4,5,8,9},2,3)              40
```

product()

Catálogo >

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.

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte página 226.

$\text{product}\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}$ $\quad [28 \quad 80 \quad 162]$

$\text{product}\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix},1,2\end{pmatrix}$ $\quad [4 \quad 10 \quad 18]$

propFrac()

Catálogo >

propFrac(*Valor1* [**,** *Var*]**)** ⇒*valor*

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 polinominal são efectuados adequadamente em relação à primeira variável principal, etc.

$\text{propFrac}\left(\frac{4}{3}\right)$ $\quad 1+\frac{1}{3}$

$\text{propFrac}\left(\frac{-4}{3}\right)$ $\quad -1-\frac{1}{3}$

propFrac() Catálogo >

Pode utilizar a função **propFrac()** para representar as fracções mistas e demonstrar a adição e a subtracção de fracções mistas.

$\text{propFrac}\left(\frac{11}{7}\right)$	$1+\frac{4}{7}$
$\text{propFrac}\left(3+\frac{1}{11}+5+\frac{3}{4}\right)$	$8+\frac{37}{44}$
$\text{propFrac}\left(3+\frac{1}{11}-\left(5+\frac{3}{4}\right)\right)$	$-2-\frac{29}{44}$

Q

QR Catálogo >

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 ctrl enter 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 · max(dim(*Matriz*)) · rowNorm (*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	$\begin{bmatrix}8.12404&9.60114&11.0782\\0.&0.904534&1.80907\\0.&0.&0.\end{bmatrix}$

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

Calcula a regressão polinomial quadráticay = a ·x²+b ·x+ca partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

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 numéricos ou de cadeias 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 226).

Variável de saída	**Descrição**
stat.RegEqn	Equação de regressão: a ·x²+b ·x+c
stat.a, stat.b, stat.c	Parâmetros de regressão
stat.R²	Coeficiente de determinação
stat.Resid	Resíduos da regressão
stat.XReg	Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.YReg	Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*

stat.FreqReg	Lista de frequências correspondentes a *stat.XReg* e *stat.YReg*

QuartReg

Catálogo >

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

Calcula a regressão polinomial quárticay = a ·x4+b ·x3+c · x2+d ·x+ea partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

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 numéricos ou de cadeias 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 226).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: a ·x4+b ·x3+c · x2+d ·x+e
stat.a, stat.b, stat.c, stat.d, stat.e	Parâmetros de regressão
stat.R2	Coeficiente de determinação
stat.Resid	Resíduos da regressão

Variável de saída	Descrição
stat.XReg	Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.YReg	Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.FreqReg	Lista de frequências correspondentes a *stat.XReg* e *stat.YReg*

R

R▶Pθ() Catálogo >

R▶Pθ (*xValor*, *yValor*) ⇒ *valor*
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:

$R▶P\theta(2,2)$ 45.

No modo de ângulo Grados:

$R▶P\theta(2,2)$ 50.

No modo de ângulo de Radianos:

$R▶P\theta(3,2)$ 0.588003

$R▶P\theta\left(\left[3\ \ -4\ \ 2\right],\left[0\ \ \frac{\pi}{4}\ \ 1.5\right]\right)$

$\left[0.\ \ 2.94771\ \ 0.643501\right]$

R▶Pr() Catálogo >

R▶Pr (*xValor*, *yValor*) ⇒ *valor*
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:

$R▶Pr(3,2)$ 3.60555

$R▶Pr\left(\left[3\ \ -4\ \ 2\right],\left[0\ \ \frac{\pi}{4}\ \ 1.5\right]\right)$

$\left[3\ \ 4.07638\ \ \frac{5}{2}\right]$

▶Rad Catálogo >

Valor1▶*Rad* ⇒ *valor*

No modo de ângulo de grau:

▶ Rad

Catálogo >

Converte o argumento para a medição do ângulo de radianos.

Nota: Pode introduzir esta função através da escrita de **@Rad** no teclado do computador

$(1.5)\blacktriangleright\text{Rad}$ $(0.02618)^r$

No modo de ângulo Grados:

$(1.5)\blacktriangleright\text{Rad}$ $(0.023562)^r$

rand()

Catálogo >

rand() ⇒ *expressão*
rand(*#Tentativas***)** ⇒ *lista*

rand() devolve um valor aleatório entre 0 e 1.

rand(*#Tentativas***)** devolve uma lista com *# valores* aleatórios entre 0 e 1

Define a semente do número aleatório.

RandSeed 1147 *Done*

rand(2) {0.158206,0.717917}

randBin()

Catálogo >

randBin(*n***,** *p***)** ⇒ *expressão*
randBin(*n***,** *p***,** *#Trials***)** ⇒ *lista*

randBin(*n***,** *p***)** devolve um número real aleatório de uma distribuição binomial especificada.

randBin(*n***,** *p***,** *#Tentativas***)** devolve uma lista com números reais aleatórios *#Tentativas* de uma distribuição binomial especificada.

randBin(80,0.5) 46.

randBin(80,0.5,3) {43.,39.,41.}

randInt()

Catálogo >

randInt (*lowBound***,***upBound***)** ⇒ *expressão*
randInt (*LimiteInferior* **,***LimiteSuperior* **,***#Tentativas***)** ⇒ *lista*

randInt(3,10) 3.

randInt(3,10,4) {9.,3.,4.,7.}

randInt() Catálogo >

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() Catálogo >

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)	$\begin{bmatrix} 8 & -3 & 6 \\ -2 & 3 & -6 \\ 0 & 4 & -6 \end{bmatrix}$

Nota: Os valores desta matriz mudam sempre que prime .

randNorm() Catálogo >

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 [μ−3•σ, μ+3•σ].

RandSeed 1147	*Done*
randNorm(0,1)	0.492541
randNorm(3,4.5)	−3.54356

randNorm()

Catálogo >

randNorm(μ**,** σ**,** *#Tentativas***)** devolve uma lista com números decimais *#Tentativas* de uma distribuição normal especificada.

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(*x*,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	{1.,3.,3.,1.,3.,1.}

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(*Valor1***) ⇒** *valor*

Devolve a parte real do argumento.

real(*Lista1***) ⇒** *lista*

real(2+3·*i*)	2
real({1+3·*i*,3,*i*})	{1,3,0}

real() Catálogo >

Devolve as partes reais de todos os elementos.

real(*Matriz1***)** ⇒ *matriz*

Devolve as partes reais de todos os elementos.

$$\text{real}\begin{bmatrix} 1+3\cdot i & 3 \\ 2 & i \end{bmatrix} \qquad \begin{bmatrix} 1 & 3 \\ 2 & 0 \end{bmatrix}$$

▶Rect Catálogo >

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 120.

$$\begin{bmatrix} 3 & \angle\frac{\pi}{4} & \angle\frac{\pi}{6} \end{bmatrix}\blacktriangleright\text{Rect}$$
$$\begin{bmatrix} 1.06066 & 1.06066 & 2.59808 \end{bmatrix}$$

ValorComplexo **▶Rect**

Apresenta o *ValorComplexo* na forma retangular a+bi. O *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 em coordenadas polares ($r\angle\theta$).

No modo de ângulo de Radianos:

$$\left(4\cdot e^{\frac{\pi}{3}}\right)\blacktriangleright\text{Rect} \qquad 11.3986$$

$$\left(4\angle\frac{\pi}{3}\right)\blacktriangleright\text{Rect} \qquad 2.+3.4641\cdot i$$

No modo de ângulo Grados:

$$(1\angle 100)\blacktriangleright\text{Rect} \qquad i$$

No modo de ângulo de grau:

$$(4\angle 60)\blacktriangleright\text{Rect} \qquad 2.+3.4641\cdot i$$

Nota: Para escrever $\angle$, selecione-o na lista de símbolos no Catálogo.

ref(*Matriz1*[**,** *Tol*]**)** ⇒ *matriz*

Devolve a forma de escalão-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 ctrl enter· 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 •máx(dim(*Matriz1*)) •rowNorm (*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:

$$\text{ref}\left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) \qquad \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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.

$$\text{ref}\left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) | a=0 \qquad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\text{ref}\left(\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}\right) \qquad \begin{bmatrix} 1 & \frac{-2}{5} & \frac{-4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$$

ref() Catálogo >

Nota: Consulte também **rref()**, página 143.

AtualizarVarsSonda Catálogo >

AtualizarVarsSonda

Permite-lhe aceder a dados de sensor a partir de todas as sondas de sensor ligadas no seu programa TI-Basic.

Valor EstadoVar	Estado
estadoVar =0	Normal (continue com o programa)
estadoVar =1	A aplicação Vernier DataQuest™ está no modo de recolha de dados. **Nota:** A aplicação Vernier DataQuest™ tem de estar no modo de medidor para que este comando funcione. 
estadoVar =2	A aplicação Vernier DataQuest™ não foi lançada.
estadoVar =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()

Catálogo >

remain(*Valor1***, *Valor2*)** ⇒ *valor*
remain(*Lista1***, *Lista2*)** ⇒ *lista*
remain(*Matriz1***, *Matriz2*)** ⇒ *matriz*

Devolve o resto do primeiro argumento em relação ao segundo argumento conforme definido pelas identidades:

remain(x,0) x
remain(x,y) x−y•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 102.

remain(7,0)	7
remain(7,3)	1
remain(−7,3)	−1
remain(7,−3)	1
remain(−7,−3)	−1
remain({12,−14,16},{9,7,−5})	{3,0,1}

$$\text{remain}\left(\begin{bmatrix}9 & -7\\ 6 & 4\end{bmatrix},\begin{bmatrix}4 & 3\\ 4 & -3\end{bmatrix}\right) \qquad \begin{bmatrix}1 & -1\\ 2 & 1\end{bmatrix}$$

Request (Pedido)

Catálogo >

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.

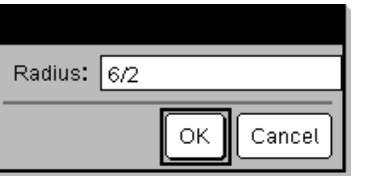
- Se *DispFlag* for omitido ou avalia para **1**, a mensagem de pedido e a resposta do utilizador são apresentadas no histórico de Calculadora.
- Se *DispFlag* avaliar para **0**, o pedido e a resposta não são apresentados no

Definir um programa:

```
Definir request_demo()=Prgm
    Pedido "Raio: ",r
    Apres "Área = ",pi*r²
EndPrgm
```

Execute o programa e escreva uma resposta:

```
request_demo()
```



Resultado após selecionar **OK**:

```
Raio: 6/2
Área= 28,2743
```

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 [on] e pressionar [enter] 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.

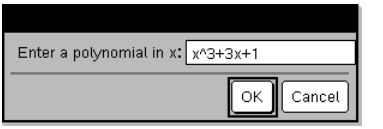
Nota: Consulte também **CadeiaDePedido**, página 137.

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}
```

CadeiaDePedido

Catálogo >

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 [on] e pressionar [enter] 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 135.

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 selecionar **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

Catálogo >

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.

```
Define factorial(nn)=
Func
Local answer,counter
1→answer
For counter,1,nn
answer·counter→answer
EndFor
Return answer
EndFunc
```

factorial(3) 6

Return

Catálogo >

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.

right()

Catálogo >

right(*List1*[**,** *Num*]**)** ⇒ *lista*

Devolve os elementos *Num* mais à direita contidos em *Lista1*.

Se omitir *Num*, devolve todos os elementos de *Lista1*.

right({1,3,-2,4},3) {3,-2,4}

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("Hello",2) "lo"

right(*Comparação***)** ⇒ *expressão*

Devolve o lado direito de uma equação ou desigualdade.

rk23 ()

Catálogo >

rk23(*Expr***,** *Var***,** *depVar***, {***Var0***,** *VarMax***},** *depVar0***,** *VarStep* [**,** *diftol*]**)** ⇒ *matriz*

rk23(*SystemOfExpr***,** *Var***,** *ListOfDepVars***,** {*Var0*, *VarMax*}**,** *ListOfDepVars0***,** *VarStep*[**,** *diftol*]**)** ⇒ *matriz*

rk23(*ListOfExpr***,** *Var***,** *ListOfDepVars***,** {*Var0*, *VarMax*}**,** *ListOfDepVars0***,** *VarStep*[**,** *diftol*]**)** ⇒ *matriz*

Equação diferencial:

y'=0.001*y*(100-y) e y(0)=10

$$\text{rk23}\left(0.001\cdot y\cdot(100-y),t,y,\{0,100\},10,1\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4 \\ 10. & 10.9367 & 11.9493 & 13.042 & 14.2 \end{bmatrix}$$

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

Mesma equação com *diftol* definido para 1.E-6

$$\text{rk23}\left(0.001\cdot y\cdot(100-y),t,y,\{0,100\},10,1,1.\text{E}{-6}\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9367 & 11.9495 & 13.0423 & 14.2189 \end{bmatrix}$$

Sistema de equações:

Utiliza o método Runge-Kutta para resolver o sistema

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

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.

Se $VarStep$ avalia para um número diferente de zero: sinal($VarStep$) = sinal ($VarMax$-$Var0$) 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".

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

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

$$\text{rk23}\left(\begin{cases} -y1+0.1 \cdot y1 \cdot y2 \\ 3 \cdot y2 - y1 \cdot y2 \end{cases}, t, \{y1, y2\}, \{0,5\}, \{2,5\}, 1\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 2. & 1.94103 & 4.78694 & 3.25253 & 1.82848 \\ 5. & 16.8311 & 12.3133 & 3.51112 & 6.27245 \end{bmatrix}$$

diftol é a tolerância de erro (passa para 0,001).

root(*Value***)** ⇒ *raiz*
root(*Value1***, *Value2***)** ⇒ *raiz*

root(*Valor*) devolve a raiz quadrada de *Valor*.

root(*Valor1***, *Valor2***)** devolve a raiz de *Valor2* de *Valor1*. *Valor1* pode ser uma constante de ponto flutuante complexa ou uma constante racional complexa ou número inteiro.

Nota: Consulte também **Modelo da raiz de índice N**, página 2.

$\sqrt[3]{8}$ 2

$\sqrt[3]{3}$ 1.44225

rotate(*NúmeroInteiro1***[, *#deRotações*]) ⇒** *número inteiro*

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 17.

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.

0b00000000000000111101011000011010 1

O bit mais à direita roda para o extremo esquerdo.

No modo base Bin:

rotate(0b11111111111111111111111111111111)
0b10000000000000000000000000000001
rotate(256,1) 0b1000000000

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

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).

produz:

0b10000000000000011110101100001101O

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).

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(*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).

rotate("abcd")	"dabc"
rotate("abcd",-2)	"cdab"
rotate("abcd",1)	"bcda"

round() Catálogo >

round (*Valor1*[**,** *dígitos*]**)** ⇒ *valor*

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(1.234567,3)	1.235

round (*Lista1*[**,** *dígitos*]**)** ⇒ *lista*

round({π,√2,ln(2)},4)	{3.1416,1.4142,0.6931}

round()

Catálogo >

Devolve uma lista dos elementos arredondada para o número especificado de dígitos.

round (*Matriz1***[, *dígitos*]) ⇒** *matriz*

Devolve uma matriz dos elementos arredondados para o número especificado de dígitos.

$$\text{round}\left(\begin{bmatrix} \ln(5) & \ln(3) \\ \pi & e^1 \end{bmatrix}, 1\right) \qquad \begin{bmatrix} 1.6 & 1.1 \\ 3.1 & 2.7 \end{bmatrix}$$

rowAdd()

Catálogo >

rowAdd(*Matriz1***, *rIndex1*, *rIndex2*) ⇒** *matriz*

Devolve uma cópia de *Matriz1* com a linha *rIndex2* substituída pela soma das linhas *rIndex1* e *rIndex2*.

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

rowDim()

Catálogo >

rowDim(*Matriz***) ⇒** *expressão*

Devolve o número de linhas em *Matriz*.

Nota: Consulte também **colDim()**, página 24.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

$$\text{rowDim}(m1) \qquad 3$$

rowNorm()

Catálogo >

rowNorm(*Matriz***) ⇒** *expressão*

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 25.

$$\text{rowNorm}\left(\begin{bmatrix} -5 & 6 & -7 \\ 3 & 4 & 9 \\ 9 & -9 & -7 \end{bmatrix}\right) \qquad 25$$

rowSwap() Catálogo >

rowSwap (*Matriz1***, *rIndex1*, *rIndex2*) ⇒ *matriz***

Devolve *Matriz1* com as linhas *rIndex1* e *rIndex2* trocadas.

$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix} \rightarrow mat \qquad \begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}$$

$$\text{rowSwap}(mat,1,3) \qquad \begin{bmatrix}5 & 6\\ 3 & 4\\ 1 & 2\end{bmatrix}$$

rref() Catálogo >

rref(*Matriz1*[**,** *Tol*]**)** ⇒ *matriz*

Devolve a forma de escalão-linha reduzida de *Matriz1*.

$$\text{rref}\left(\begin{bmatrix}-2 & -2 & 0 & -6\\ 1 & -1 & 9 & -9\\ -5 & 2 & 4 & -4\end{bmatrix}\right) \qquad \begin{bmatrix}1 & 0 & 0 & \frac{66}{71}\\ 0 & 1 & 0 & \frac{147}{71}\\ 0 & 0 & 1 & \frac{-62}{71}\end{bmatrix}$$

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.

- Se utilizar ctrl 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 •máx(dim(*Matriz1*)) •rowNorm (*Matriz1*)

Nota: Consulte também **ref()**, página 133.

S

sec() Tecla trig

sec(*Valor1***)** ⇒ *valor*

sec(*Lista1***)** ⇒ *lista*

Devolve a secante de *Valor1* ou devolve uma lista com as secantes de todos os elementos em *Lista1*.

No modo de ângulo Graus:

$$\sec(45) \qquad 1.41421$$

$$\sec(\{1,2.3,4\}) \qquad \{1.00015,1.00081,1.00244\}$$

sec()

Tecla trig

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 trig

sec⁻¹(*Valor1***)** ⇒ *valor*

sec⁻¹(*Lista1***)** ⇒ *lista*

Devolve o ângulo cuja secante é *Valor1* ou devolve uma lista com as secantes inversas 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 **arcsec(...)** no teclado do computador.

No modo de ângulo Graus:

$\sec^{-1}(1)$ 0.

No modo de ângulo Gradianos:

$\sec^{-1}(\sqrt{2})$ 50.

No modo de ângulo Radianos:

$\sec^{-1}(\{1,2,5\})$ $\{0,1.0472,1.36944\}$

sech()

Catálogo >

sech(*Valor1***)** ⇒ *valor*

sech(*Lista1***)** ⇒ *lista*

Devolve a secante hiperbólica de *Valor1* ou devolve uma lista com as secantes hiperbólicas dos elementos *Lista1*.

$\text{sech}(3)$ 0.099328

$\text{sech}(\{1,2.3,4\})$ $\{0.648054,0.198522,0.036619\}$

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

Catálogo >

sech⁻¹(*Valor1***)** ⇒ *valor*

sech⁻¹(*Lista1***)** ⇒ *lista*

Devolve a secante hiperbólica inversa de *Valor1* ou devolve uma lista com as secantes hiperbólicas inversas de cada elemento de *Lista1*.

No modo de ângulo Radianos e Formato complexo rectangular:

$\text{sech}^{-1}(1)$ 0

$\text{sech}^{-1}(\{1,-2,2.1\})$ $\{0,2.0944\cdot i,8.E{-}15+1.07448\cdot i\}$

sech^{-1}() Catálogo >

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

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 63), **GetStr** (página 71) e **eval()** (página 50).

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 *lightval*	*Done*
lightval	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.

$n:=50$	50
$m:=4$	4
Send "SET SOUND eval(m· n)"	*Done*
iostr.SendAns	"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.

$\text{seq}\left(n^2,n,1,6\right)$	$\{1,4,9,16,25,36\}$
$\text{seq}\left(\frac{1}{n},n,1,10,2\right)$	$\left\{1,\frac{1}{3},\frac{1}{5},\frac{1}{7},\frac{1}{9}\right\}$
$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$	$\frac{1968329}{1270080}$

Obs: Para forçar um resultado aproximado,

seq() Catálogo >

Unidade portátil: Premir ctrl enter.
Windows®: Premir **Ctrl+Enter**.
Macintosh®: Premir ⌘+**Enter**.
iPad®: Manter pressionada a tecla **Enter** e selecionar ≈.

$$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right) \qquad 1.54977$$

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*

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**.

Gere o primeiros 5 termos da sequência $u(n) = u(n\text{-}1)^2/2$, com $u(1)$=**2** e *VarStep*=**1**.

$$\text{seqGen}\left(\frac{(u(n-1))^2}{n},n,u,\{1,5\},\{2\}\right) \qquad \left\{2,2,\frac{4}{3},\frac{4}{9},\frac{16}{405}\right\}$$

Exemplo no qual Var0=2:

$$\text{seqGen}\left(\frac{u(n-1)+1}{n},n,u,\{2,5\},\{3\}\right) \qquad \left\{3,\frac{4}{3},\frac{7}{12},\frac{19}{60}\right\}$$

Sistema de duas sequências:

$$\text{seqGen}\left(\left\{\frac{1}{n},\frac{u2(n-1)}{2}+u1(n-1)\right\},n,\{u1,u2\},\{1,5\},\begin{bmatrix}\_\\2\end{bmatrix}\right) \qquad \begin{bmatrix}1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5}\\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24}\end{bmatrix}$$

Nota: O Vazio (_) na matriz do termo inicial acima, é utilizado para indicar que o 1º termo para u1(n) é calculado utilizando a fórmula de sucessão u1(n)=1/n.

seqn() Catálogo >

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**.

$$\text{seqn}\left(\frac{u(n-1)}{n},\{2\},6\right) \qquad \left\{2,1,\frac{1}{3},\frac{1}{12},\frac{1}{60},\frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2},6\right) \qquad \left\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16},\frac{1}{25},\frac{1}{36}\right\}$$

setMode() Catálogo >

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.

```
Define prog1()=Prgm                    Done
               Disp π
               setMode(1,16)
               Disp π
               EndPrgm
prog1()
                                       3.14159
                                       3.14
                                       Done
```

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 70.

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.

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
Formato vectorial	6	**1** =Rectangular, **2** =Cilíndrico, **3** =Esférico
Base	7	**1** =Decimal, **2** =Hex, **3** =Binário

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 17.

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)	0b1111010110000011010
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

Insere 0 se o bit mais à esquerda for 0

ou 1 se o bit mais à esquerda for 1.

produz:

0b00000000000000011110101100001101O

O resultado aparece de acordo com o modo base. Os zeros à esquerda não aparecem.

shift(*Lista1* [**,** *#deDeslocações*]**)** ⇒*lista*

Devolve uma cópia de *Lista1* deslocada para a direita ou para a esquerda pelos elementos *#deDeslocações*. Não altere *Lista1*.

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".

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(*Cadeia1* [**,** *#deDeslocações*]**)** ⇒*cadeia*

Devolve uma cópia de *Cadeia1* rodada para a direita ou para a esquerda pelos caracteres *#deDeslocações*. Não altere *Cadeia1*.

shift("abcd")	" abc"
shift("abcd",−2)	" ab"
shift("abcd",1)	"bcd "

shift() Catálogo >

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() Catálogo >

sign(*Valor1***)** ⇒*valor*

sign(*Lista1***)** ⇒*lista*

sign(*Matriz1***)** ⇒*matriz*

Para *Valor1* real ou complexo, devolve *Valor1* / **abs(***Valor1***)** quando *Valor1* ≠ 0.

Devolve 1 se *Valor1* for positivo.

Devolve −1 se *Valor1* for negativo.

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.

$\text{sign}(-3.2)$ → -1

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

Se o modo do formato complexo for Real:

$\text{sign}([-3 \quad 0 \quad 3])$ → $[-1 \quad \text{undef} \quad 1]$

simult() Catálogo >

simult(*MatrizCoef***,** *VectorConst* **[,** *Tol* **])** ⇒*matriz*

Devolve um vector da coluna que contém as soluções para um sistema de equações lineares.

Nota: Consulte também **linSolve()**, página 89.

MatrizCoef tem de ser uma matriz quadrada que contenha os coeficientes das equações.

Resolver para x e y:

x + 2y = 1

3x + 4y = −1

$\text{simult}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}, \begin{bmatrix}1\\ -1\end{bmatrix}\right)$ → $\begin{bmatrix}-3\\ 2\end{bmatrix}$

A solução é x=−3 e y=2.

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 · max(dim(*MatrizCoef*)) · rowNorm(*MatrizCoef*)

simult(*MatrizCoef***,** *MatrizConst* **[,** *Tol* **])** ⇒*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

$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \rightarrow matx1$ $\quad \begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$

$\text{simult}\left(matx1, \begin{bmatrix}1\\2\end{bmatrix}\right)$ $\quad \begin{bmatrix}0\\\frac{1}{2}\end{bmatrix}$

Resolver:

x + 2y = 1

3x + 4y = ⁻1

x + 2y = 2

3x + 4y = ⁻3

$\text{simult}\left(\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}, \begin{bmatrix}1 & 2\\-1 & -3\end{bmatrix}\right)$ $\quad \begin{bmatrix}-3 & -7\\2 & \frac{9}{2}\end{bmatrix}$

Para o primeiro sistema, x= ⁻3 e y=2. Para o segundo sistema, x= ⁻7 e y=9/2.

sin() Tecla trig

sin(*Valor1***)** ⇒*valor*

sin(*Lista1***)** ⇒*lista*

sin(*Valor1***)** devolve o seno do argumento.

sin(*Lista1***)** devolve uma lista de senos de todos os elementos em *Lista1*.

No modo de ângulo Graus:

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

$\sin(45)$ $\quad 0.707107$

$\sin(\{0,60,90\})$ $\quad \{0.,0.866025,1.\}$

sin()

Tecla trig

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 Gradianos:

$\sin(50)$ 0.707107

No modo de ângulo Radianos:

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

$\sin(45°)$ 0.707107

sin(*MatrizQuadrada1***)** ⇒*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 diagnolizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Radianos:

$$\sin\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$

$$\begin{bmatrix}0.9424 & -0.04542 & -0.031999\\ -0.045492 & 0.949254 & -0.020274\\ -0.048739 & -0.00523 & 0.961051\end{bmatrix}$$

$\sin^{-1}()$

Tecla trig

sin⁻¹(*Valor1***)** ⇒*valor*

sin⁻¹(*Lista1***)** ⇒*lista*

sin⁻¹(*Valor1***)** devolve o ângulo cujo seno é *Valor1*.

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.

No modo de ângulo Graus:

$\sin^{-1}(1)$ 90.

No modo de ângulo Gradianos:

$\sin^{-1}(1)$ 100.

No modo de ângulo Radianos:

$\sin^{-1}(\{0,0.2,0.5\})$ $\{0.,0.201358,0.523599\}$

sin⁻¹(*MatrizQuadrada1***)** ⇒*MatrizQuadrada*

Nos modos de ângulo Radianos e Formato complexo rectangular:

sin⁻¹() Tecla trig

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 diagnolizável. O resultado contém sempre os números de ponto flutuante.

$$\sin^{-1}\left(\begin{bmatrix}1 & 5\\ 4 & 2\end{bmatrix}\right)$$
$$\begin{bmatrix}-0.174533-0.12198\cdot i & 1.74533-2.35591\cdot i\\ 1.39626-1.88473\cdot i & 0.174533-0.593162\cdot i\end{bmatrix}$$

sinh() Catálogo >

sinh(*Numver1***)** ⇒*valor*

sinh(*Lista1***)** ⇒*lista*

sinh(*Valor1***)** devolve o seno hiperbólico do argumento.

sinh(*Lista1***)** devolve uma lista dos senos hiperbólicos de cada elemento de *Lista1*.

sinh(*MatrizQuadrada1***)** ⇒*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 diagnolizável. O resultado contém sempre os números de ponto flutuante.

$$\sinh(1.2) \qquad 1.50946$$
$$\sinh(\{0,1.2,3.\}) \qquad \{0,1.50946,10.0179\}$$

No modo de ângulo Radianos:

$$\sinh\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$
$$\begin{bmatrix}360.954 & 305.708 & 239.604\\ 352.912 & 233.495 & 193.564\\ 298.632 & 154.599 & 140.251\end{bmatrix}$$

sinh⁻¹() Catálogo >

sinh⁻¹(*Valor1***)** ⇒*valor*

sinh⁻¹(*Lista1***)** ⇒*lista*

sinh⁻¹(*Valor1***)** devolve o seno hiperbólico inverso do argumento.

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.

$$\sinh^{-1}(0) \qquad 0$$
$$\sinh^{-1}(\{0,2.1,3\}) \qquad \{0,1.48748,1.81845\}$$

sinh⁻¹(*MatrizQuadrada1***)**
⇒*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()**.

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

No modo de ângulo Radianos:

$$\sinh^{-1}\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$
$$\begin{bmatrix}0.041751 & 2.15557 & 1.1582\\ 1.46382 & 0.926568 & 0.112557\\ 2.75079 & -1.5283 & 0.57268\end{bmatrix}$$

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 158).

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 numéricos ou de cadeias 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.

SinReg

Catálogo >

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 226).

Variável de saída	Descrição
stat.RegEqn	Equação de regressão: a ·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 *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.YReg	Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias*
stat.FreqReg	Lista de frequências correspondentes a *stat.XReg* e *stat.YReg*

SortA

Catálogo >

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 226.

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
SortA *list1*	*Done*
list1	$\{1,2,3,4\}$
$\{4,3,2,1\} \rightarrow list2$	$\{4,3,2,1\}$
SortA *list2*,*list1*	*Done*
list2	$\{1,2,3,4\}$
list1	$\{4,3,2,1\}$

SortD

Catálogo >

SortD *Lista1* [**,** *Lista2*] [**,** *Lista3*] ...

SortD *Vector1* [**,** *Vector*] [**,** *Vector3*] ...

Idêntico a **SortA**, excepto que **SortD** ordena os elementos por ordem decrescente.

Os elementos (nulos) vazios do primeiro argumento movem-se para a parte inferior. Para mais informações sobre os elementos vazios, consulte página 226.

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
SortD *list1*,*list2*	*Done*
list1	$\{4,3,2,1\}$
list2	$\{3,4,1,2\}$

▸Sphere

Catálogo >

Vector **▸Sphere**

Nota: Pode introduzir esta função através da escrita de **@>Sphere** no teclado.

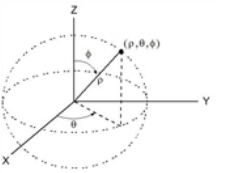
Apresenta o vector da linha ou coluna em forma esférica [ρ ∠θ ∠ϕ].

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.

$[1 \quad 2 \quad 3] \blacktriangleright \text{Sphere}$
$[3.74166 \quad \angle 1.10715 \quad \angle 0.640522]$

$\left(\begin{bmatrix} 2 & \angle \frac{\pi}{4} & 3 \end{bmatrix}\right) \blacktriangleright \text{Sphere}$
$[3.60555 \quad \angle 0.785398 \quad \angle 0.588003]$



sqrt ()

Catálogo >

sqrt(*Valor1***)** ⇒*valor*

sqrt(*Lista1***)** ⇒*lista*

Devolve a raiz quadrada do argumento.

Para uma lista, devolve as raízes quadradas de todos os elementos em *Lista1*.

Nota: Consulte também **Modelo de raiz quadrada**, página 1.

$\sqrt{4}$	2
$\sqrt{\{9,2,4\}}$	$\{3,1.41421,2\}$

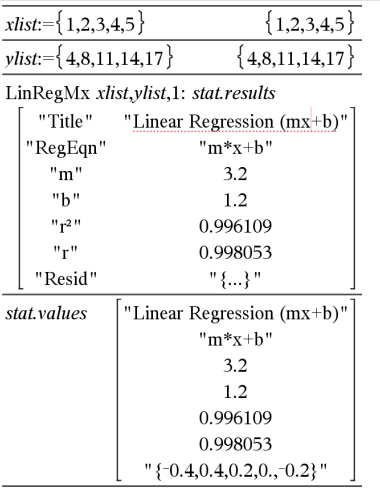
stat.results

Apresenta os resultados de um cálculo estatístico.

Os resultados aparecem como um conjunto de pares de valores de nomes. Os nomes específicos apresentados estão dependentes do comando ou da função estatística avaliada mais recentemente.

Pode copiar um nome ou um valor e colá-lo noutra localização.

Nota: Evite definir variáveis que utilizem os mesmos nomes das variáveis utilizadas para análise estatística. Em alguns casos, pode ocorrer uma condição de erro. Os nomes das variáveis utilizados para análise estatística são listados na tabela abaixo.



stat.a	stat.dfDenom	stat.MedianY	stat.Q3X	stat.SSBlock
stat.AdjR²	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.r²	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.σx	stat.SSRow
stat.b5	stat.DW	stat.MSInteract	stat.σy	stat.tList
stat.b6	stat.e	stat.MSReg	stat.σx1	stat.UpperPred
stat.b7	stat.ExpMatrix	stat.MSRow	stat.σx2	stat.UpperVal
stat.b8	stat.$\bar{F}$	stat.n	stat.Σx	stat.$\bar{x}$
stat.b9	stat.$\bar{F}$Block	stat.$\hat{p}$	stat.Σx²	stat.$\bar{x}$1
stat.b10	stat.$\bar{F}$col	stat.$\hat{p}$1	stat.Σxy	stat.$\bar{x}$2
stat.bList	stat.$\bar{F}$Interact	stat.$\hat{p}$2	stat.Σy	stat.$\bar{x}$Diff
stat.χ^2	stat.FreqReg	stat.$\hat{p}$Diff	stat.Σy²	stat.$\bar{x}$List
stat.c	stat.$\bar{F}$row	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

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.

Consulte o exemplo de **stat.results**.

stDevPop()

Catálogo >

stDevPop(*Lista* **[,** *ListFreq* **])**⇒

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 226.

Nos modos auto e de ângulo Radianos:

$\text{stDevPop}(\{1,2,5,-6,3,-2\})$	3.59398
$\text{stDevPop}(\{1.3,2.5,-6.4\},\{3,2,5\})$	4.11107

stDevPop() Catálogo >

stDevPop(*Matriz1* [**,** *MatrizFreq*]**)**
⇒*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 226.

$$\text{stDevPop}\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix} \quad \begin{bmatrix} 3.26599 & 2.94392 & 1.63299 \end{bmatrix}$$

$$\text{stDevPop}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right) \quad \begin{bmatrix} 2.52608 & 5.21506 \end{bmatrix}$$

stDevSamp() Catálogo >

stDevSamp(*Lista* [**,** *ListaFreq*]**)**
⇒*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 226.

$$\text{stDevSamp}(\{1,2,5,-6,3,-2\}) \quad 3.937$$

$$\text{stDevSamp}(\{1.3,2.5,-6.4\},\{3,2,5\}) \quad 4.33345$$

stDevSamp(*Matriz1* [**,** *MatrizFreq*]**)**
⇒*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 226.

$$\text{stDevSamp}\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix} \quad \begin{bmatrix} 4. & 3.60555 & 2. \end{bmatrix}$$

$$\text{stDevSamp}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right) \quad \begin{bmatrix} 2.7005 & 5.44695 \end{bmatrix}$$

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:=0                                   0
Define prog1()=Prgm                    Done
    For i,1,10,1
    If i=5
    Stop
    EndFor
    EndPrgm
prog1()                                Done
i                                      5
```

Store (Guardar)

Consulte → (guardar), página 209.

string() — Catálogo >

strin g(*Expr***)** ⇒*cadeia*

Simplifica *Expr* e devolve o resultado como uma cadeia de caracteres.

```
string(1.2345)                         "1.2345"
string(1+2)                            "3"
```

subMat() — Catálogo >

subMa t(*Matriz1* [**,** *LinhaInicial*] [**,** *ColInicial*] [**,** *LinhaFinal*] [**,** *ColFinal*]**)** ⇒*matrix*

Devolve a submatriz especificada de *Matriz1*.

Predefinições: *LinhaInicial* =1, *ColInicial* =1, *LinhaFinal* =última linha, *ColFinal* =última coluna.

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\rightarrow m1 \qquad \begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$$

$$\text{subMat}(m1,2,1,3,2) \qquad \begin{bmatrix}4&5\\7&8\end{bmatrix}$$

$$\text{subMat}(m1,2,2) \qquad \begin{bmatrix}5&6\\8&9\end{bmatrix}$$

Sigma (Soma)

Consulte Σ(), página 201.

sum() Catálogo >

sum(*Lista* [**,** *Início* [**,** *Fim*]]**)** ⇒*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 226.

$\text{sum}(\{1,2,3,4,5\})$ 15

$\text{sum}(\{a,2\cdot a,3\cdot a\})$ "Error: Variable is not defined"

$\text{sum}(\text{seq}(n,n,1,10))$ 55

$\text{sum}(\{1,3,5,7,9\},3)$ 21

sum(*Matrix1* [**,** *Início* [**,** *Fim*]]**)** ⇒*matriz*

Devolve um vector da linha com as somas dos elementos nas colunas em *Matriz1*.

Início e *Fim* são opcionais. Especificam um intervalo de linhas.

Qualquer argumento vazio produz um resultado vazio. Os elementos (nulos) vazios da *Matriz1* são ignorados. Para mais informações sobre os elementos vazios, consulte página 226.

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

$\text{sum}\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$ $\begin{bmatrix}12&15&18\end{bmatrix}$

$\text{sum}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},2,3\right)$ $\begin{bmatrix}11&13&15\end{bmatrix}$

sumIf() Catálogo >

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 **?**

$\text{sumIf}(\{1,2,\mathbf{e},3,\pi,4,5,6\},2.5<?<4.5)$ 12.859874482

$\text{sumIf}(\{1,2,3,4\},2<?<5,\{10,20,30,40\})$ 70

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 226.

Nota: Consulte também **countIf()**, página 31.

sumSeq() **Consulte Σ(), página 201.**

system() **Catálogo >**

system(*Valor1* [**,** *Valor2* [**,** *Valor3* [**,** ...]]]**)**

Devolve um sistema de equações formatado como uma lista. Pode também criar um sistema com um modelo.

T

T (transpor) **Catálogo >**

*Matriz1***T**⇒*matriz*

Apresenta a transposta dos conjugados dos complexo da *Matriz1*.

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

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}^{T} \qquad \begin{bmatrix}1&4&7\\2&5&8\\3&6&9\end{bmatrix}$$

tan()

Tecla trig

tan(*Valor1***)** ⇒*valor*

tan(*Lista1***)** ⇒*lista*

tan(*Valor1***)** devolve a tangente do argumento.

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.

No modo de ângulo Graus:

$\tan\left(\left\{\frac{\pi}{4}r\right\}\right)$ → 1.

$\tan(45)$ → 1.

$\tan(\{0,60,90\})$ → $\{0.,1.73205,\text{undef}\}$

No modo de ângulo Gradianos:

$\tan\left(\left\{\frac{\pi}{4}r\right\}\right)$ → 1.

$\tan(50)$ → 1.

$\tan(\{0,50,100\})$ → $\{0.,1.,\text{undef}\}$

No modo de ângulo Radianos:

$\tan\left(\frac{\pi}{4}\right)$ → 1.

$\tan(45°)$ → 1.

$\tan\left(\left\{\pi,\frac{\pi}{3},-\pi,\frac{\pi}{4}\right\}\right)$ → $\{0.,1.73205,0.,1.\}$

tan(*MatrizQuadrada1***)** ⇒*MatrizQuadrada*

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

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

No modo de ângulo Radianos:

$$\tan\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right) \rightarrow \begin{bmatrix}-28.2912 & 26.0887 & 11.1142\\ 12.1171 & -7.83536 & -5.48138\\ 36.8181 & -32.8063 & -10.4594\end{bmatrix}$$

tan⁻¹()

Tecla trig

tan⁻¹(*Valor1***)** ⇒*valor*

tan⁻¹(*Lista1***)** ⇒*lista*

tan⁻¹(*Valor1***)** devolve o ângulo cuja tangente é *Valor1*.

No modo de ângulo Graus:

$\tan^{-1}(1)$ → 45

No modo de ângulo Gradianos:

tan⁻¹() Tecla trig

tan⁻¹(*Lista1***)** devolve uma lista das tangentes inversas de cada elemento de *Lista1*.

$\tan^{-1}(1)$ 50

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 Radianos:

$\tan^{-1}(\{0,0.2,0.5\})$ $\{0,0.197396,0.463648\}$

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

tan⁻¹(*MatrizQuadrada1***)** ⇒*MatrizQuadrada*

No modo de ângulo Radianos:

$$\tan^{-1}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}$$
$$\begin{bmatrix}-0.083658 & 1.26629 & 0.62263\\ 0.748539 & 0.630015 & -0.070012\\ 1.68608 & -1.18244 & 0.455126\end{bmatrix}$$

Devolve a tangente inversa da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular a tangente inversa de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

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

tanh() Catálogo >

tanh(*Valor1***)** ⇒*valor*

$\tanh(1.2)$ 0.833655

$\tanh(\{0,1\})$ $\{0.,0.761594\}$

tanh(*Lista1***)** ⇒*lista*

tanh(*Valor1***)** devolve a tangente hiperbólica do argumento.

tanh(*Lista1***)** devolve uma lista das tangentes hiperbólicas de cada elemento de *Lista1*.

tanh(*MatrizQuadrada1***)** ⇒*MatrizQuadrada*

No modo de ângulo Radianos:

$$\tanh\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}$$
$$\begin{bmatrix}-0.097966 & 0.933436 & 0.425972\\ 0.488147 & 0.538881 & -0.129382\\ 1.28295 & -1.03425 & 0.428817\end{bmatrix}$$

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()**.

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

tanh⁻¹() Catálogo >

tanh⁻¹(*Valor1***)** ⇒*valor*

tanh⁻¹(*Lista1***)** ⇒*lista*

tanh⁻¹(*Valor1***)**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 diagnolizável. O resultado contém sempre os números de ponto flutuante.

No Formato complexo rectangular:

$$\tanh^{-1}(0) \qquad 0.$$

$$\tanh^{-1}(\{1,2.1,3\})$$
$$\{\text{undef}, 0.518046-1.5708\cdot i, 0.346574-1.570\ldots$$

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

No modo de ângulo Radianos e Formato complexo rectangular:

$$\tanh^{-1}\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$
$$\begin{bmatrix}-0.099353+0.164058\cdot i & 0.267834-1.4908\\ -0.087596-0.725533\cdot i & 0.479679-0.9473\ldots\\ 0.511463-2.08316\cdot i & -0.878563+1.7901\end{bmatrix}$$

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

tCdf() Catálogo >

tCdf(*LimiteInferior***,** *LimiteSuperior***,** *dfs***)** ⇒*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 *df*.

Para P(X ≤ *LimiteSuperior*), defina *LimiteInferior* = **–9E**999.

Text

Catálogo >

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.

Se o programa necessitar de uma resposta escrita do utilizador, consulte **Request**, página 135, ou**RequestStr**, página 137.

Nota: Pode utilizar este comando num programa definido pelo utilizador, mas não numa função.

Defina um programa que interrompa a visualização após cinco números aleatórios numa caixa de diálogo.

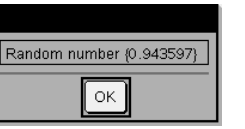
No modelo Prgm...EndPrgm, complete cada linha, premindo ↵ em vez de enter. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
Define text_demo()=Prgm
  For i,1,5
    strinfo:="Random number" &
string(rand(i))
    Text strinfo
  EndFor
EndPrgm
```

Executar o programa:

```
text_demo()
```

Amostra de uma caixa de diálogo:



Then

Consulte If, página 74.

tInterval

Catálogo >

tInterval *Lista*[**,***Freq*[**,***NívelC*]]

(Entrada da lista de dados)

tInterval $\bar{x}$**,***sx*,*n*[**,***NívelC*]

(Entrada estatística do resumo)

tInterval

Catálogo >

Calcula um intervalo de confiança *t*. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

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.σx	Desvio padrão da amostra
stat.n	Comprimento da sequência de dados com a média da amostra

tInterval_2Samp

Catálogo >

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 158).

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 226).

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 *Lista 1* e *Lista 2*
stat.n1, stat.n2	Número de amostras em sequências de dados
stat.sp	Desvio padrão combinado. Calculado quando *Combinado* = SIM.

tPdf() Catálogo >

tPdf(*ValX*, *df*) ⇒*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*)⇒*valor*

Apresenta o traço (soma de todos os elementos na diagonal principal) de *MatrizQuadrada*.

$\text{trace}\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$ 15

$a:=12$ 12

$\text{trace}\begin{bmatrix}a&0\\1&a\end{bmatrix}$ 24

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 236.

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.

```
Define prog1()=Prgm
  Try
  z:=z+1
  Disp "z incremented."
  Else
  Disp "Sorry, z undefined."
  EndTry
  EndPrgm
                                    Done
z:=1:prog1()
                                z incremented.
                                    Done
DelVar z:prog1()
                                Sorry, z undefined.
                                    Done
```

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.

```
eigenvals({{-3},{-41},{5}},[-1  2  -3.1])
```

Nota: Consulte também **ClrErr**, página 24, e **PassErr**, página 119.

```
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 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 158).

Teste H_0: μ = μ0, em relação a uma das seguintes:

Para H_a: μ < μ0, defina *Hipótese*<0

Para H_a: μ ≠ μ0 (predefinição), defina *Hipótese*=0

Para H_a: μ > μ0, defina *Hipótese*>0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

Variável de saída	Descrição
stat.t	$(\bar{x} - \mu 0) / (\text{stdev} / \text{sqrt}(n))$
stat.PVal	Menor nível de significância para o qual a hipótese nula pode ser rejeitada
stat.df	Graus de liberdade

Variável de saída	Descrição
stat.$\bar{x}$	Média da amostra da sequência de dados em *Lista*
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 158).

Teste H_0: $\mu1 = \mu2$, em relação a uma das seguintes:

Para H_a: $\mu1 < \mu2$, defina *Hipótese*<0

Para H_a: $\mu1 \neq \mu2$ (predefinição), defina *Hipótese*=0

Para H_a: $\mu1 > \mu2$, 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 226).

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.$\bar{x}$1, stat.$\bar{x}$2	Médias da amostra das sequências de dados em *Lista 1* e *Lista 2*
stat.sx1, stat.sx2	Desvios padrão da amostra das sequências de dados em *Lista 1* e *Lista 2*

Variável de saída	Descrição
stat.n1, stat.n2	Tamanho das amostras
stat.sp	Desvio padrão combinado. Calculado quando *Combinado* =1.

tvmFV()

Catálogo >

tvmFV(*N*, *I*, *PV*, *Pmt*, [*PpY*], [*CpY*], [*PmtAt*]**)** ⇒*valor*

Função financeira que calcula o valor futuro do dinheiro.

Nota: Os argumentos utilizados nas funções TVM são descritos na tabela de argumentos TVM, página 174. Consulte também **amortTbl()**, página 7.

tvmFV(120,5,0,-500,12,12) 77641.1

tvmI()

Catálogo >

tvmI(*N*, *PV*, *Pmt*, *FV*, [*PpY*], [*CpY*], [*PmtAt*]**)** ⇒*valor*

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 174. Consulte também **amortTbl()**, página 7.

tvmI(240,100000,-1000,0,12,12) 10.5241

tvmN()

Catálogo >

tvmN(*I*, *PV*, *Pmt*, *FV*, [*PpY*], [*CpY*], [*PmtAt*]**)** ⇒*valor*

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 174. Consulte também **amortTbl()**, página 7.

tvmN(5,0,-500,77641,12,12) 120.

tvmPmt()

Catálogo >

tvmPmt(*N*, *I*, *PV*, *FV*, [*PpY*], [*CpY*], [*PmtAt*]**)** ⇒*valor*

tvmPmt(60,4,30000,0,12,12) -552.496

tvmPmt() Catálogo >

Função financeira que calcula o montante de cada pagamento.

Nota: Os argumentos utilizados nas funções TVM são descritos na tabela de argumentos TVM, página 174. Consulte também **amortTbl()**, página 7.

tvmPV() Catálogo >

tvmPV(*N***,** *I***,** *Pmt***,** *FV***,** [*PpY*]**,** [*CpY*]**,** [*PmtAt*]**)** ⇒*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 174. Consulte também **amortTbl()**, página 7.

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, predefiniçãot=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 Catálogo >

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

Calcula a estatística TwoVar. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

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 numéricos ou de cadeias 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 226.

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
stat.$\bar{y}$	Média de valores y

Variável de saída	Descrição
stat. y	Soma de valores y
stat. y^2	Soma de valores y2
stat.sy	Desvio padrão da amostra de y
stat. y	Desvio padrão da população de y
stat. xy	Soma de valores x ·y
stat.r	Coeficiente de correlação
stat.MinX	Mínimo dos valores x
stat.Q_1X	1º quartil de x
stat.MedianX	Mediana de x
stat.Q_3X	3º quartil de x
stat.MaxX	Máximo de valores x
stat.MinY	Mínimo dos valores y
stat.Q_1Y	1º quartil de y
stat.MedY	Mediana de y
stat.Q_3Y	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

U

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.

```
unitV([1  2  1])
        [0.408248  0.816497  0.408248]
unitV([1])      [0.267261]
      [2]       [0.534522]
      [3]       [0.801784]
```

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 92, e **getLockInfo()**, página 69.

$a:=65$	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*

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*.

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 226.

varPop$(\{5,10,15,20,25,30\})$	72.9167

varSamp()

Catálogo >

varSamp(*Lista* [**,** *ListaFreq*]**)** ⇒*expressão*

Devolve a variação da amostra de *Lista*.

Cada elemento de *ListFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

Nota: *Lista* tem de conter pelo menos dois elementos.

varSamp$(\{1,2,5,-6,3,-2\})$	$\frac{31}{2}$
varSamp$(\{1,3,5\},\{4,6,2\})$	$\frac{68}{33}$

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 226.

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 226.

$$\text{varSamp}\begin{bmatrix}1 & 2 & 5\\ -3 & 0 & 1\\ .5 & .7 & 3\end{bmatrix} \qquad \begin{bmatrix}4.75 & 1.03 & 4\end{bmatrix}$$

$$\text{varSamp}\left(\begin{bmatrix}-1.1 & 2.2\\ 3.4 & 5.1\\ -2.3 & 4.3\end{bmatrix},\begin{bmatrix}6 & 3\\ 2 & 4\\ 5 & 1\end{bmatrix}\right) \qquad \begin{bmatrix}3.91731 & 2.08411\end{bmatrix}$$

W

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 [on] e pressionar [enter]

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"
```

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.

warnCodes () **Catálogo >**

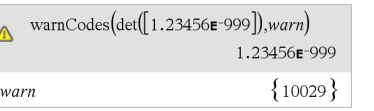
warnCodes(*Expr1***,** *StatusVar***)⇒***expressão*

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.

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 245.



when() **Catálogo >**

when(*Condição***,** *ResultadoVerdadeiro* **[,** *ResultadoFalso* **][,** *ResultadoDesconhecido* **]) ⇒***expressão*

when() Catálogo >

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.

$\text{when}(x<0, x+3) \mid x=5$	undef

$\text{when}(n>0, n\cdot factoral(n-1), 1) \rightarrow factoral(n)$	*Done*
$factoral(3)$	6
$3!$	6

While Catálogo >

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 $i \le n$ $tempsum + \frac{1}{i} \rightarrow tempsum$ $i+1 \rightarrow i$ EndWhile Return $tempsum$ EndFunc	*Done*
$sum\_of\_recip(3)$	$\frac{11}{6}$

xor (xou) — Catálogo >

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**, página 116.

true xor true	false
5>3 xor 3>5	true

NúmeroInteiro1 **xor** *NúmeroInteiro2* ⇒ *número inteiro*

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 17.

No modo base Hex:

Importante: Zero, não a letra O.

0h7AC36 xor 0h3D5F	0h79169

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.

xor (xou) Catálogo >

Nota: Consulte **or**, página 116.

Z

zInterval Catálogo >

zInterval σ**,** *Lista* [**,** *Freq* [**,** *NívelC*]]

(Entrada da lista de dados)

zInterval σ**,** $\bar{x}$**,** *n* [**,** *NívelC*]

(Entrada estatística do resumo)

Calcula um intervalo de confiança *z*. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

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
stat.σ	Desvio padrão da população conhecido para a sequência de dados *Lista*

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 158).

x é um número inteiro não negativo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

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 158).

x1 e *x2* são números inteiros não negativos.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

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
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 σ_1, σ_2, *Lista1*, *Lista2* [, *Freq1* [, *Freq2*, [*NívelC*]]]

(Entrada da lista de dados)

zInterval_2Samp σ_1, σ_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 158).

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

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.σx1, stat.σx2	Desvios padrão da amostra para *Lista 1* e *Lista 2*
stat.n1, stat.n2	Número de amostras em sequências de dados
stat.r1, stat.r2	Desvios padrão da população conhecidos para sequência de dados *Lista 1* e *Lista 2*

zTest

zTest μ*0*, σ, *Lista*, [*Freq* [, *Hipótese*]]

(Entrada da lista de dados)

zTest μ*0*, σ, $\bar{x}$, *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 158).

Teste H_0: μ = μ0, em relação a uma das seguintes:

zTest

Catálogo >

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 226).

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 *Lista*
stat.sx	Desvio padrão da amostra da sequência de dados. Apenas devolvido para a entrada *Dados*.
stat.n	Tamanho da amostra

zTest_1Prop

Catálogo >

zTest_1Prop *p0*, *x*, *n* [, *Hipótese*]

Calcula um teste *z* de uma proporção. Um resumo dos resultados é guardado na variável *stat.results* (página 158).

x é um número inteiro não negativo.

Teste H_0: $p = p0$ em relação a uma das seguintes:

Para H_a: $p > p0$, defina *Hipótese*>0

Para H_a: $p \neq p0$ *(predefinição)*, defina *Hipótese*=0

Para H_a: $p < p0$, defina *Hipótese*<0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

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

Catálogo >

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 158).

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 ≠ p2 (predefinição)*, defina *Hipótese*=0

Para H_a: *p < p0*, defina *Hipótese*<0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" (página 226).

Variável de saída	Descrição
stat.z	Valor normal padrão calculado para a diferença de proporções
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 σ_1, σ_2, *Lista1*, *Lista2* [, *Freq1*

[, *Freq2* [, *Hipótese*]]]

(Entrada da lista de dados)

zTest_2Samp σ_1, σ_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 158).

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 226).

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 *Lista1* e *Lista2*
stat.sx1, stat.sx2	Desvios padrão da amostra das sequências de dados em *Lista1* e *Lista2*
stat.n1, stat.n2	Tamanho das amostras

Símbolos

+ (adicionar) — Tecla +

Valor1 + *Valor2* ⇒*valor*

Devolve a soma dos dois argumentos.

56	56
56+4	60
60+4	64
64+4	68
68+4	72

Lista1 + *Lista2* ⇒*lista*

Matriz1 + *Matriz2* ⇒*matriz*

Devolve uma lista (ou matriz) com as somas dos elementos correspondentes em *Lista1* e *Lista2* (ou *Matriz1* e *Matriz2*).

As dimensões dos argumentos têm de ser iguais.

$\left\{22,\pi,\frac{\pi}{2}\right\}\rightarrow l1$	$\{22,3.14159,1.5708\}$
$\left\{10,5,\frac{\pi}{2}\right\}\rightarrow l2$	$\{10,5,1.5708\}$
l1+*l2*	$\{32,8.14159,3.14159\}$

Valor + *Lista1* ⇒*lista*

Lista1 + *Valor* ⇒*lista*

Devolve uma lista com as somas de *Valor* e de cada elemento em *Lista1*.

15+{10,15,20}	{25,30,35}
{10,15,20}+15	{25,30,35}

Valor + *Matriz1* ⇒*matriz*

Matriz1 + *Valor* ⇒*matriz*

Devolve uma matriz com *Valor* adicionado a cada elemento na diagonal de *Matriz1*. *Matriz1* tem de ser quadrada.

Nota: Utilize **.+** (ponto mais) para adicionar uma expressão a cada elemento.

$20+\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$	$\begin{bmatrix}21 & 2\\ 3 & 24\end{bmatrix}$

− (subtrair) — Tecla −

Valor1 − *Valor2* ⇒*valor*

Devolve *Valor1* menos *Valor2*.

6−2	4
$\pi-\frac{\pi}{6}$	2.61799

Lista1 − *Lista2* ⇒*lista*

Matriz1 − *Matriz2* ⇒*matriz*

$\left\{22,\pi,\frac{\pi}{2}\right\}-\left\{10,5,\frac{\pi}{2}\right\}$	$\{12,^{-}1.85841,0.\}$
[3 4]−[1 2]	[2 2]

− (subtrair)

Tecla −

Subtrai cada elmento em *Lista2* (ou *Matriz2*) do elemento correspondente em *Lista1* (ou *Matriz1*) e devolve os resultados.

As dimensões dos argumentos têm de ser iguais.

Valor − *Lista1* ⇒*lista*

Lista1 − *Valor* ⇒*lista*

Subtrai cada elemento de *Lista1* de *Valor* ou subtrai *Valor* de cada elemento de *Lista1* e devolve uma lista dos resultados.

$15-\{10,15,20\}$ → $\{5,0,-5\}$

$\{10,15,20\}-15$ → $\{-5,0,5\}$

Valor − *Matriz1* ⇒*matriz*

Matriz1 − *Valor* ⇒*matriz*

Valor − *Matriz1* devolve uma matriz de *Valor* vezes a matriz de identidade menos *Matriz1*. *Matriz1* tem de ser quadrada.

Matriz1 − *Valor* devolve uma matriz de *Valor* vezes a matriz de identidade subtraída de *Matriz1*. *Matriz1* tem de ser quadrada.

$20-\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$ → $\begin{bmatrix}19 & -2\\ -3 & 16\end{bmatrix}$

Nota: Utilize **.**− (ponto menos) para subtrair uma expressão de cada elemento.

· (multiplicar)

Tecla ×

Valor1 · *Valor2* ⇒*valor*

Devolve o produto dos dois argumentos.

$2\cdot 3.45$ → 6.9

Lista1 · *Lista2* ⇒*lista*

Devolve uma lista com os produtos dos elementos correspondentes em *Lista1* e *Lista2*.

As dimensões das listas têm de ser iguais.

$\{1.,2,3\}\cdot\{4,5,6\}$ → $\{4,10,18\}$

Matriz1 · *Matriz2* ⇒*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*.

$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}.\begin{bmatrix}7 & 8\\ 7 & 8\\ 7 & 8\end{bmatrix}$ → $\begin{bmatrix}42 & 48\\ 105 & 120\end{bmatrix}$

· (multiplicar)

Tecla ×

Valor · *Lista1* ⇒*lista*

Lista1 · *Valor* ⇒*lista*

Devolve uma lista com os produtos de *Valor* e de cada elemento em *Lista1*.

$\pi \cdot \{4,5,6\}$ → $\{12.5664, 15.708, 18.8496\}$

Valor · *Matriz1* ⇒*matriz*

Matriz1 · *Valor* ⇒*matriz*

Devolve uma matriz com os produtos de *Valor* e de cada elemento em *Matriz1*.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 0.01$ → $\begin{bmatrix} 0.01 & 0.02 \\ 0.03 & 0.04 \end{bmatrix}$

$6 \cdot \text{identity}(3)$ → $\begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix}$

Nota: Utilize **.** · (ponto multiplicar) para multiplicar uma expressão por cada elemento.

/ (dividir)

Tecla ÷

Valor1 / *Valor2* ⇒*valor*

Devolve o quociente de *Valor1* dividido pelo *Valor2*.

Nota: Consulte também **Modelo da fracção**, página 1.

$\frac{2}{3.45}$ → 0.57971

Lista1 / *Lista2* ⇒*lista*

Devolve uma lista com os quocientes de *Lista1* divididos pela *Lista2*.

As dimensões das listas têm de ser iguais.

$\frac{\{1.,2,3\}}{\{4,5,6\}}$ → $\left\{0.25, \frac{2}{5}, \frac{1}{2}\right\}$

Valor / *Lista1* ⇒ *lista*

Lista1 / *Valor* ⇒ *lista*

Devolve uma lista com os quocientes de *Valor* divididos pela *Lista1* ou de *Lista1* divididos pelo *Valor*.

$\frac{6}{\{3,6,\sqrt{6}\}}$ → $\{2,1,2.44949\}$

$\frac{\{7,9,2\}}{7 \cdot 9 \cdot 2}$ → $\left\{\frac{1}{18}, \frac{1}{14}, \frac{1}{63}\right\}$

Valor / *Matriz1* ⇒ *matriz*

Matriz1 / *Valor* ⇒ *matriz*

Devolve uma matriz com os quocientes de *Matriz1* / *Valor*.

$\frac{[7 \quad 9 \quad 2]}{7 \cdot 9 \cdot 2}$ → $\begin{bmatrix} \frac{1}{18} & \frac{1}{14} & \frac{1}{63} \end{bmatrix}$

Nota: Utilize **. /** (ponto dividir) para dividir uma expressão por cada elemento.

^ (potência) — Tecla ^

Valor1 ^ *Valor2* ⇒ *valor*

Lista1 **^** *Lista2* ⇒ *lista*

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.

4^2 → 16

$\{2,4,6\}^{\{1,2,3\}}$ → $\{2,16,216\}$

Valor ^ *Lista1* ⇒ *lista*

Devolve *Valor* elevado à potência dos elementos em *Lista1*.

$\pi^{\{1,2,-3\}}$ → $\{3.14159,9.8696,0.032252\}$

Lista1 ^ *Valor* ⇒ *lista*

Devolve os elementos em *Lista1* elevados à potência de *Valor*.

$\{1,2,3,4\}^{-2}$ → $\left\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16}\right\}$

MatrizQuadrada1 **^** *número inteiro* ⇒ *matriz*

Devolve *MatrizQuadrada1* elevada à potência do *número inteiro*.

MatrizQuadrada1 tem de ser uma matriz quadrada.

Se *número inteiro* = −1, calcula a matriz inversa.

Se *número inteiro* < −1, calcula a matriz inversa para uma potência positiva adequada.

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^2$ → $\begin{bmatrix}7 & 10\\ 15 & 22\end{bmatrix}$

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1}$ → $\begin{bmatrix}-2 & 1\\ \frac{3}{2} & \frac{-1}{2}\end{bmatrix}$

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-2}$ → $\begin{bmatrix}\frac{11}{2} & \frac{-5}{2}\\ \frac{-15}{4} & \frac{7}{4}\end{bmatrix}$

x 2 (quadrado)

Tecla x²

Valor1 ² ⇒ *valor*

Devolve o quadrado do argumento.

Lista1 **²** ⇒ *lista*

Devolve uma lista com os quadrados dos elementos em *Lista1*.

MatrizQuadrada1 ² ⇒ *matriz*

Devolve a matriz quadrada de *MatrizQuadrada1*. Isto não é o mesmo que calcular o quadrado de cada elemento. Utilize .^2 para calcular o quadrado de cada elemento.

$4^2 \quad 16$

$\{2,4,6\}^2 \quad \{4,16,36\}$

$\begin{bmatrix}2&4&6\\3&5&7\\4&6&8\end{bmatrix}^2 \quad \begin{bmatrix}40&64&88\\49&79&109\\58&94&130\end{bmatrix}$

$\begin{bmatrix}2&4&6\\3&5&7\\4&6&8\end{bmatrix}.\wedge 2 \quad \begin{bmatrix}4&16&36\\9&25&49\\16&36&64\end{bmatrix}$

.+ (ponto adicionar)

Teclas . +

Matriz1 **.+** *Matriz2* ⇒ *matriz*

Valor **.+***Matriz1* ⇒ *matriz*

Matriz1 **.+** *Matriz2* devolve uma matriz que é a soma de cada par dos elementos correspondentes em *Matriz1* e *Matriz2*.

Valor **.+** *Matriz1* devolve uma matriz que é a soma de *Valor* e de cada elemento em *Matriz1*.

$\begin{bmatrix}1&2\\3&4\end{bmatrix}.+\begin{bmatrix}10&30\\20&40\end{bmatrix} \quad \begin{bmatrix}11&32\\23&44\end{bmatrix}$

$5.+\begin{bmatrix}10&30\\20&40\end{bmatrix} \quad \begin{bmatrix}15&35\\25&45\end{bmatrix}$

.− (ponto subtracção)

Teclas . −

Matriz1 **.−** *Matriz2* ⇒ *matriz*

Valor **.−***Matriz1* ⇒ *matriz*

Matriz1 **.−***Matriz2* devolve uma matriz que é a diferença entre cada par de elementos correspondentes em *Matriz1* e *Matriz2*.

Valor **.−***Matriz1* devolve uma matriz que é a diferença de *Valor* e de cada elemento em *Matriz1*.

$\begin{bmatrix}1&2\\3&4\end{bmatrix}.-\begin{bmatrix}10&20\\30&40\end{bmatrix} \quad \begin{bmatrix}-9&-18\\-27&-36\end{bmatrix}$

$5.-\begin{bmatrix}10&20\\30&40\end{bmatrix} \quad \begin{bmatrix}-5&-15\\-25&-35\end{bmatrix}$

. · (ponto mult.)

Teclas . ×

Matriz1 .· *Matriz2* ⇒ *matriz*

Valor . · *Matriz1* ⇒ *matriz*

Matriz1 . · *Matriz2* devolve uma matriz que é o produto de cada par dos elementos correspondentes em *Matriz1* e *Matriz2*.

Valor . · *Matriz1* devolve uma matriz com os produtos de *Valor* e de cada elemento em *Matriz1*.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} .\cdot \begin{bmatrix}10 & 20\\ 30 & 40\end{bmatrix} \qquad \begin{bmatrix}10 & 40\\ 90 & 160\end{bmatrix}$$

$$5 .\cdot \begin{bmatrix}10 & 20\\ 30 & 40\end{bmatrix} \qquad \begin{bmatrix}50 & 100\\ 150 & 200\end{bmatrix}$$

. / (ponto dividir)

Teclas . ÷

Matriz1 . / *Matriz2* ⇒ *matriz*

Valor . / *Matriz1* ⇒ *matriz*

Matriz1 ./ *Matriz2* devolve uma matriz que é o quociente de cada par de elementos correspondente em *Matriz1* e *Matriz2*.

Valor ./ *Matriz1* devolve uma matriz que é o quociente de *Valor* e de cada elemento em *Matriz1*.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} ./ \left(\begin{bmatrix}10 & 20\\ 30 & 40\end{bmatrix}\right) \qquad \begin{bmatrix}\frac{1}{10} & \frac{1}{10}\\ \frac{1}{10} & \frac{1}{10}\end{bmatrix}$$

$$5 ./ \left(\begin{bmatrix}10 & 20\\ 30 & 40\end{bmatrix}\right) \qquad \begin{bmatrix}\frac{1}{2} & \frac{1}{4}\\ \frac{1}{6} & \frac{1}{8}\end{bmatrix}$$

.^ (ponto potência)

Teclas . ^

Matriz1 .^ *Matriz2* ⇒ *matriz*

Valor . ^ *Matriz1* ⇒ *matriz*

Matriz1 .^ *Matriz2* devolve uma matriz em que cada elemento em *Matriz2* é o expoente para o elemento correspondente em *Matriz1*.

Valor .^ *Matriz1* devolve uma matriz em que cada elemento em *Matriz1* é o expoente para *Valor*.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} .^\wedge \begin{bmatrix}0 & 2\\ 3 & -1\end{bmatrix} \qquad \begin{bmatrix}1 & 4\\ 27 & \frac{1}{4}\end{bmatrix}$$

$$5 .^\wedge \begin{bmatrix}0 & 2\\ 3 & -1\end{bmatrix} \qquad \begin{bmatrix}1 & 25\\ 125 & \frac{1}{5}\end{bmatrix}$$

- (negação)

Tecla (−)

-*Valor1* ⇒ *valor*

-*Lista1* ⇒ *lista*

$$-2.43 \qquad -2.43$$

$$-\{-1,0.4,1.2\text{E}19\} \qquad \{1.,-0.4,-1.2\text{E}19\}$$

- (negação)

Tecla (−)

-Matriz1 ⇒ *matriz*

Devolve a negação do argumento.

Para uma lista ou matriz, devolve todos os elementos negados.

Se o argumento for um número inteiro binário ou hexadecimal, a negação dá o complemento de dois.

No modo base Bin:

Importante: Zero, não a letra O

```
-0b100101
0b1111111111111111111111111111111
```

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

% (percentagem)

Teclas ctrl 📖

Valor1 % ⇒ *valor*

Lista1 **%** ⇒ *lista*

Matriz1 **%** ⇒ *matriz*

Devolve $\frac{argumento}{100}$

Para uma lista ou matriz, devolve uma lista ou matriz com cada elemento dividido por 100.

13%	0.13
{{1,10,100}}%	{0.01,0.1,1.}

= (igual)

Tecla =

Expr1 **=** *Expr2* ⇒*Expressão booleana*

Lista1 **=** *Lista2* ⇒ *Lista booleana*

Matriz1 **=** *Matriz2* ⇒ *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.

Exemplo de função que utiliza os símbolos de teste matemático: =, ≠, <, ≤, >, ≥

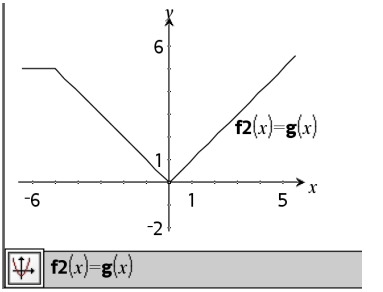
```
Define g(x)=Func
    If x≤-5 Then
    Return 5
    ElseIf x>-5 and x<0 Then
    Return -x
    ElseIf x≥0 and x≠10 Then
    Return x
    ElseIf x=10 Then
    Return 3
    EndIf
    EndFunc
                                Done
```

= (igual)

Tecla =

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)



≠ (diferente)

Teclas ctrl =

Expr1 ≠ *Expr2* ⇒ *Expressão booleana*

Lista1 ≠ *Lista2* ⇒ *Lista booleana*

Matriz1 ≠ *Matriz2* ⇒ *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.

Para listas e matrizes, devolve comparações elemento por elemento.

Nota: Pode introduzir este operador através da escrita de **/=** no teclado.

Consulte exemplo "=" (igual).

< (menor que)

Teclas ctrl =

Expr1 < *Expr2* ⇒ *Expressão booleana*

Lista1 < *Lista2* ⇒ *Lista booleana*

Matriz1 < *Matriz2* ⇒ *Matriz booleana*

Devolve verdadeiro se *Expr1* for determinada para ser menor que *Expr2*.

Consulte exemplo "=" (igual).

< (menor que)

Teclas ctrl =

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.

≤ (igual ou menor que)

Teclas ctrl =

Expr1 ≤ *Expr2* ⇒ *Expressão booleana*

Consulte exemplo "=" (igual).

Lista1 ≤ *Lista2* ⇒ *Lista booleana*

Matriz1 ≤ *Matriz2* ⇒ *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 **<=** no teclado

> (maior que)

Teclas ctrl =

Expr1 > *Expr2* ⇒ *Expressão booleana*

Consulte exemplo "=" (igual).

Lista1 > *Lista2* ⇒ *Lista booleana*

Matriz1 > *Matriz2* ⇒ *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.

> (maior que)

Teclas ctrl =

Para listas e matrizes, devolve comparações elemento por elemento.

≥ (igual ou maior que)

Teclas ctrl =

Expr1 ≥ *Expr2* ⇒ *Expressão booleana*

Lista1 ≥ *Lista2* ⇒ *Lista booleana*

Matriz1 ≥ *Matriz2* ⇒ *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.

Consulte exemplo "=" (igual).

⇒ (implicação lógica)

Teclas ctrl =

ExprBooleana1 ⇒ *ExprBooleana2* devolve *expressão booleana*

ListaBooleana1 ⇒ *ListaBooleana2* devolve *lista booleana*

MatrizBooleana1 ⇒ *MatrizBooleana2* devolve *matriz booleana*

NúmeroInteiro1 ⇒ *NúmeroInteiro2* devolve *número inteiro*

5>3 or 3>5	true
5>3 ⇒ 3>5	false
3 or 4	7
3 ⇒ 4	-4
{1,2,3} or {3,2,1}	{3,2,3}
{1,2,3} ⇒ {3,2,1}	{-1,-1,-3}

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.

⇒ (implicação lógica)

Teclas ctrl =

Nota: Pode introduzir este operador ao escrever **=>** com o teclado

⇔ (implicação lógica dupla, XNOR)

Teclas ctrl =

ExprBooleana1 ⇔ *ExprBooleana2*
devolve *expressão booleana*

ListaBooleana1 ⇔ *ListaBooleana2*
devolve *lista booleana*

MatrizBooleana1 ⇔ *MatrizBooleana2*
devolve *matriz booleana*

NúmeroInteiro1 ⇔ *NúmeroInteiro2*
devolve *número inteiro*

5>3 xor 3>5	true
5>3 ⇔ 3>5	false
3 xor 4	7
3 ⇔ 4	-8
{1,2,3} xor {3,2,1}	{2,0,2}
{1,2,3} ⇔ {3,2,1}	{-3,-1,-3}

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.

Nota: Pode introduzir este operador ao escrever **<=>** com o teclado

! (factorial)

Tecla ?!

Valor1! ⇒ *valor*

*Lista1***!** ⇒ *lista*

*Matriz1***!** ⇒ *matriz*

5!	120
{{5,4,3}}!	{120,24,6}
$\left\{\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right\}!$	$\begin{bmatrix}1 & 2\\ 6 & 24\end{bmatrix}$

Devolve o factorial do argumento.

Para uma lista ou matriz, devolve uma lista ou matriz de factoriais dos elementos.

& (acrescentar)

Teclas ctrl

Cadeia1 **&** *Cadeia2* ⇒ *cadeia*

"Hello "&"Nick"	"Hello Nick"

& (acrescentar)

Teclas ctrl

Devolve uma cadeia de texto que é *Cadeia2* acrescentada a *Cadeia1*.

d() (derivada)

Catálogo >

d(*Expr1***, *Var*[, *Ordem*]) | *Var*=*Valor*⇒*valor***

d(*Expr1***, *Var*[, *Ordem*])⇒*valor***

d(*Lista1***, *Var*[, *Ordem*])⇒*lista***

d(*Matriz1***, *Var*[, *Ordem*])⇒*matriz***

Excepto quando utilizar a primeira sintaxe, tem de guardar um valor numérico na variável *Var* antes de avaliar **d()**. Consulte os exemplos.

Pode utilizar **d()** para calcular a derivada de primeira e segunda ordem num ponto numericamente com os métodos de diferenciação automáticos.

Ordem, se incluída, tem de ser=**1** ou **2**. A predefinição é **1**.

Nota: Pode introduzir isto através da escrita de **`derivada(...) no teclado`**.

Nota: Consulte também **Primeira derivada**, página 5 ou **Segunda derivada**, página 6.

Nota: O algoritmo **d()** tem uma limitação: funciona recursivamente através da expressão não simplificada, computação do valor numérico da primeira derivada (e a segunda, se aplicável) e a avaliação de cada subexpressão, que pode conduzir a um resultado imprevisto.

Considere o exemplo da direita. A primeira derivada de x·(x^2+x)^(1/3) em x=0 é igual a 0. No entanto, como a primeira derivada da subexpressão (x^2+x)^(1/3) está indefinida em x=0, e este valor é utilizado para calcular a derivada da expressão total, **d()** reporta o resultado como indefinido e apresenta uma mensagem de aviso.

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

$x:=0:\frac{d}{dx}(|x|)$ undef

$x:=3:\frac{d}{dx}(\{x^2,x^3,x^4\})$ $\{6,27,108\}$

$\frac{d}{dx}\left(x\cdot(x^2+x)^{\frac{1}{3}}\right)|x=0$ undef

$\text{centralDiff}\left(x\cdot(x^2+x)^{\frac{1}{3}},x\right)|x=0$ 0.000033

d() (derivada) — Catálogo >

Se encontrar esta limitação, verifique a solução graficamente. Pode também tentar com **centralDiff()**.

∫() (integral) — Catálogo >

∫(*Expr1*, *Var*, *Inferior*, *Superior*) ⇒ *valor*

Devolve o integral de *Expr1* em relação à variável *Var* de *Inferior* a *Superior*. Pode ser utilizada para calcular o integral definido numericamente com o mesmo método de nInt().

Nota: Pode introduzir esta função através do teclado, escrevendo **`integral(...)`**.

Nota: Consulte também **nInt()**, página 110, e **modelo do integral definido**, página 6.

$$\int_0^1 x^2\,dx \qquad 0.333333$$

√() (raiz quadrada) — Teclas ctrl x²

√(*Valor1*) ⇒*valor*

√(*Lista1*) ⇒*lista*

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.

$$\sqrt{4} \qquad 2$$

$$\sqrt{\{9,2,4\}} \qquad \{3,1.41421,2\}$$

Π() (prodSeq) — Catálogo >

Π (*Expr1*, *Var*, *Baixo*, *Alto*) ⇒*expressão*

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

Avalia *Expr1* para cada valor de *Var* de *Baixo* a *Alto* e devolve o produto dos resultados.

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

$$\prod_{n=1}^{5}\left(\left\{\frac{1}{n},n,2\right\}\right) \qquad \left\{\frac{1}{120},120,32\right\}$$

Π () (prodSeq)

Catálogo >

Nota: Consulte também **Modelo do produto (Π)** , página 5.

Π **(***Expr1***,** *Var***,** *Baixo***,** *Baixo* −1**)**⇒1

Π **(***Expr1***,** *Var***,** *Baixo***,** *Alto***)** ⇒**1/** Π **(***Expr1***,** *Var***,** *Alto*+1**,** *Baixo* −1**)** se *Alto* < *Baixo* −1

$$\prod_{k=4}^{3}(k) \qquad 1$$

$$\prod_{k=4}^{1}\left(\frac{1}{k}\right) \qquad 6$$

$$\prod_{k=4}^{1}\left(\frac{1}{k}\right)\cdot\prod_{k=2}^{4}\left(\frac{1}{k}\right) \qquad \frac{1}{4}$$

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.

Σ() (sumSeq)

Catálogo >

Σ**(***Expr1***,** *Var***,** *Baixo***,** *Alto***)** ⇒*expressão*

Nota: Pode introduzir esta função através da escrita de sum**Seq(**...**)** 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.

Σ**(***Expr1***,** *Var***,** *Baixo***,** *Baixo* −1**)**⇒0

Σ**(***Expr1***,** *Var***,** *Baixo***,** *Alto***)** ⇒-Σ**(***Expr1***,** *Var***,** *Alto*+1**,** *Baixo* −1**)** se *Alto* < *Baixo* −1

$$\sum_{n=1}^{5}\left(\frac{1}{n}\right) \qquad \frac{137}{60}$$

$$\sum_{k=4}^{3}(k) \qquad 0$$

$$\sum_{k=4}^{1}(k) \qquad -5$$

$$\sum_{k=4}^{1}(k)+\sum_{k=2}^{4}(k) \qquad 4$$

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.

ΣInt() Catálogo >

ΣInt(*NPmt1***, *NPmt2*, *N*, *I*, *PV*, [*Pmt*], [*FV*], [*PpY*], [*CpY*], [*PmtAt*], [*ValorArredondado*]) ⇒*valor***

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 174.

- 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 7.

Nota: Consulte também ΣPrn(), abaixo, e **Bal()**, página 16.

ΣInt(1,3,12,4.75,20000,,12,12) −213.48

tbl:=amortTbl(12,12,4.75,20000,,12,12)

0	0.	0.	20000.
1	−77.49	−1632.43	18367.6
2	−71.17	−1638.75	16728.8
3	−64.82	−1645.1	15083.7
4	−58.44	−1651.48	13432.2
5	−52.05	−1657.87	11774.4
6	−45.62	−1664.3	10110.1
7	−39.17	−1670.75	8439.32
8	−32.7	−1677.22	6762.1
9	−26.2	−1683.72	5078.38
10	−19.68	−1690.24	3388.14
11	−13.13	−1696.79	1691.35
12	−6.55	−1703.37	−12.02

ΣInt(1,3,*tbl*) −213.48

ΣPrn() Catálogo >

ΣPrn(*NPmt1***, *NPmt2*, *N*, *I*, *PV*, [*Pmt*], [*FV*], [*PpY*], [*CpY*], [*PmtAt*], [*ValorArredondado*]) ⇒*valor***

ΣPrn(*NPmt1***, *NPmt2*, *TabelaDeDepreciação*) ⇒*valor***

ΣPrn(1,3,12,4.75,20000,,12,12) −4916.28

ΣPrn()

Catálogo >

Função de amortização que calcula a soma do capital 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 174.

- 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 7.

Nota: Consulte também ΣInt(), acima, e **Bal()**, página 16.

tbl:=amortTbl(12,12,4.75,20000,,12,12)

0	0.	0.	20000.
1	-77.49	-1632.43	18367.57
2	-71.17	-1638.75	16728.82
3	-64.82	-1645.1	15083.72
4	-58.44	-1651.48	13432.24
5	-52.05	-1657.87	11774.37
6	-45.62	-1664.3	10110.07
7	-39.17	-1670.75	8439.32
8	-32.7	-1677.22	6762.1
9	-26.2	-1683.72	5078.38
10	-19.68	-1690.24	3388.14
11	-13.13	-1696.79	1691.35
12	-6.55	-1703.37	-12.02

ΣPrn(1,3,*tbl*) -4916.28

(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.

xyz:=12	12
#("x"&"y"&"z")	12

Cria ou refere-se à variável xyz.

10→*r*	10
"r"→*s1*	"r"
#*s1*	10

Devolve o valor da variável (r) cujo nome é guardado na variável s1.

E (notação científica)

Tecla EE

mantissa **E** *expoente*

Introduz um número em notação científica. O número é interpretado como *mantissa* × 10 expoente.

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 2.3**E**4.

23000.	23000.
2300000000.+4.1E15	4.1E15
$3 \cdot 10^4$	30000

g (gradianos)

Tecla 1

*Expr1***g**⇒*expressão*

*Lista1***g** ⇒*lista*

*Matriz1***g** ⇒*matriz*

Esta função fornece uma forma para especificar um ângulo de gradianos enquanto está no modo Graus ou Radianos.

No modo de ângulo Radianos, multiplica *Expr1* por π/200.

No modo de ângulo Graus, multiplica *Expr1* por g/100.

No modo Gradianos, devolve *Expr1* inalterada.

Nota: Pode introduzir este símbolo através da escrita de **@g** no teclado do computador.

No modo Graus, Gradianos ou Radianos:

$\cos(50^g)$	0.707107
$\cos(\{0,100^g,200^g\})$	{1,0.,-1.}

r (radianos)

Tecla 1

*Valor1***r**⇒*valor*

*Lista1***r**⇒*lista*

*Matriz1***r**⇒*matriz*

No modo de ângulo Graus, Gradianos ou Radianos:

ʳ (radianos) — Tecla 1

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/ π.

No modo de ângulo Radianos, devolve o argumento inalterado.

No modo Gradianos, multiplica o argumento por 200/ π.

Sugestão: Utilize ʳ 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 `@r` no teclado.

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

$\cos\left(\left\{0^r, \left(\frac{\pi}{12}\right)^r, -(\pi)^r\right\}\right)$ → {1,0.965926,-1.}

° (graus) — Tecla 1

Valor1 °⇒*valor*

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 π/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:

$\cos(45°)$ → 0.707107

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 ≈.

°, ', " (grau/minuto/segundo)

Teclas ctrl 🕮

gg °*mm* ' *ss.ss* " ⇒*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 apóstrofos a seguir ss.ss (''), não um símbolo de aspas (").

No modo de ângulo Graus:

25°13'17.5"	25.2215
25°30'	$\frac{51}{2}$

∠ (ângulo)

Teclas ctrl 🕮

[*Raio*, ∠θ_*Ângulo*] ⇒*vector*

(entrada polar)

[*Raio*, ∠θ_*Ângulo*, *Z_Coordenada*] ⇒*vector*

(entrada cilíndrica)

[*Raio*, ∠θ_*Ângulo*, ∠θ_*Ângulo*] ⇒*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 **@<** no teclado do computador.

No modo Radianos e formato do vector definido para:

rectangular

$[5 \quad \angle 60° \quad \angle 45°]$
$[1.76777 \quad 3.06186 \quad 3.53553]$

cilíndrico

$[5 \quad \angle 60° \quad \angle 45°]$
$[3.53553 \quad \angle 1.0472 \quad 3.53553]$

esférico

$[5 \quad \angle 60° \quad \angle 45°]$
$[5. \quad \angle 1.0472 \quad \angle 0.785398]$

(*Magnitude* ∠ *Ângulo*) ⇒*ValorComplexo*

(entrada polar)

No modo de ângulo Radianos e Formato complexo rectangular:

∠ (ângulo)

Teclas ctrl 📖

Introduz um valor complexo em forma polar (r ∠θ). O *Ângulo* é interpretado de acordo com a definição do modo Ângulo actual.

_ (carácter de sublinhado como um elemento vazio)

Consulte "Elementos (nulos) vazios" , página 226.

10^()

Catálogo > 📖

10^(*Valor1***)** ⇒*valor*

10^(*Lista1***)** ⇒*lista*

Devolve 10 elevado à potência do argumento.

Para uma lista, devolve 10 elevado à potência dos elementos em *Lista1*.

$10^{1.5}$ → 31.6228

10^(*MatrizQuadrada1***)** ⇒*MatrizQuadrada*

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 diagnolizável. O resultado contém sempre os números de ponto flutuante.

$$10^{\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}} \rightarrow \begin{bmatrix}1.14336\text{E}7 & 8.17155\text{E}6 & 6.67589\text{E}6\\ 9.95651\text{E}6 & 7.11587\text{E}6 & 5.81342\text{E}6\\ 7.65298\text{E}6 & 5.46952\text{E}6 & 4.46845\text{E}6\end{bmatrix}$$

^-1 (recíproco)

Catálogo > 📖

Valor1 ^-1⇒*valor*

Lista1 ^-1⇒*lista*

Devolve o recíproco do argumento.

Para uma lista, devolve os recíprocos dos elementos em *Lista1*.

$(3.1)^{-1}$ → 0.322581

MatrizQuadrada1 ^-1⇒*MatrizQuadrada*

Devolve o inverso de *MatrizQuadrada1*.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1} \rightarrow \begin{bmatrix}-2 & 1\\ \frac{3}{2} & \frac{-1}{2}\end{bmatrix}$$

^-1 (recíproco)

Catálogo >

MatrizQuadrada1 tem de ser uma matriz quadrada não singular.

| (operador de limite)

Teclas ctrl

Expr | *ExprBooleana1* [**and***ExprBooleana2*]...

Expr | *ExprBooleana1* [**or***ExprBooleana2*]...

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

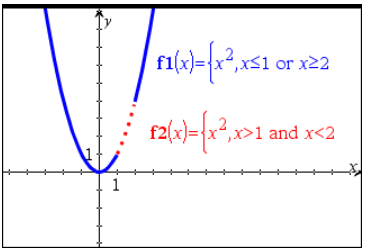
As substituições estão na forma de uma igualdade, como x=3 ou y=sin(x). Para ser mais eficaz, o membro esquerdo deve ser uma variável simples. *Expr* | *Variável* = *valor* substituem *valor* para todas as ocorrências de *Variável* em *Expr*.

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.

$x+1 \mid x=3$	4
$x+55 \mid x=\sin(55)$	54.0002

$x^3-2\cdot x+7 \rightarrow f(x)$	*Done*
$f(x) \mid x=\sqrt{3}$	8.73205

$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0, x)$	0.
$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0, x) \mid x>0 \text{ and } x<5$	3.



| (operador de limite)

Teclas ctrl 🕮

As exclusões utilizam o operador relacional "diferentes" (/= ou ≠) para excluir um valor específico de consideração.

→ (guardar)

Teclas ctrl var

Value → *Var*

Lista → *Var*

Matriz → *Var*

Expr → *Função*(*Parâm1*,...)

Lista → *Função*(*Parâm1*,...)

Matriz → *Função*(*Parâm1*,...)

Se a variável *Var* não existir, cria-a e inicia-a para *Valor*, *Lista* ou *Matriz*.

Se a variável *Var* já existir e não estiver bloqueada nem protegida, substitui o conteúdo por *Valor*, *Lista* ou *Matriz*.

Nota: Pode introduzir este operador através da escrita de **=: no teclado** como um atalho. Por exemplo, escreva **pi/4 =: myvar**.

$\frac{\pi}{4} \rightarrow myvar$ 0.785398

$2\cdot\cos(x) \rightarrow y1(x)$ *Done*

$\{1,2,3,4\} \rightarrow lst5$ $\{1,2,3,4\}$

$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} \rightarrow matg$ $\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$

"Hello" → *str1* "Hello"

:= (atribuir)

Teclas ctrl

Var := *Valor*

Var **:=** *Lista*

Var **:=** *Matriz*

Função(*Parâm1*,...) **:=** *Expr*

Função(*Parâm1*,...) **:=** *Lista*

Função(*Parâm1*,...) **:=** *Matriz*

Se a variável *Var* não existir, cria *Var* e inicia-a para *Valor*, *Lista* ou *Matriz*.

$myvar := \frac{\pi}{4}$ .785398

$y1(x) := 2\cdot\cos(x)$ *Done*

$lst5 := \{1,2,3,4\}$ $\{1,2,3,4\}$

$matg := \begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$ $\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$

str1 := "Hello" "Hello"

:= (atribuir)

Teclas ctrl

Se *Var* já existir e não estiver bloqueada nem protegida, substitui o conteúdo por *Valor*, *Lista* ou *Matriz*.

© (comentário)

Teclas ctrl

© [*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 g(n)=Func
    © Declare variables
    Local i,result
    result:=0
    For i,1,n,1 ©Loop n times
    result:=result+i^2
    EndFor
    Return result
    EndFunc
                                    Done
g(3)                                 14
```

0b, 0h

Teclas 0 B, **teclas** 0 H

0b *NúmeroBinário*

0h *NúmeroHexadecimal*

Indica um número binário ou hexadecimal, respectivamente. Para introduzir um número binário ou hexadecimal, utilize sempre o prefixo 0b ou 0h 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 Dec:

```
0b10+0hF+10                          27
```

No modo base Bin:

```
0b10+0hF+10                     0b11011
```

No modo base Hex:

```
0b10+0hF+10                      0h1B
```

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 ou PublishView (.tnsp) 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.

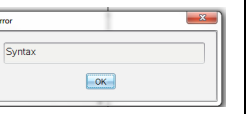
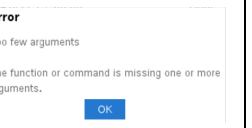
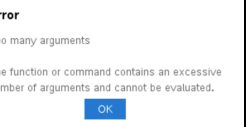
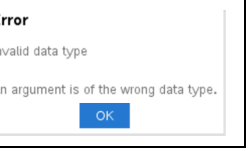
Ecrã de gráficos	Predefinição
Altura	212
Largura	318
Cor	branco: 255,255,255

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	
Erro Demasiados argumentos	A função ou o comando contém um número excessivo de argumentos e não pode ser avaliada.	
Erro Tipo de dados inválido	Um argumento é do tipo de dados errado.	

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
```

DrawArc

Catálogo > CXII

DrawArc *x, y, largura, altura, startAngle, arcAngle*

Desenhe um arco no retângulo delimitador definido com os ângulos de início e de arco fornecidos.

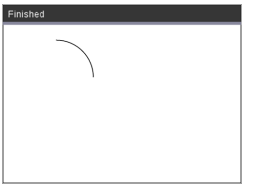
x, *y*: coordenada superior esquerda do retângulo delimitador

largura, altura: dimensões do retângulo delimitador

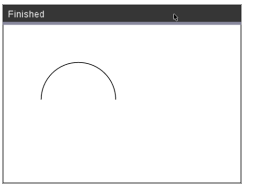
O "ângulo do arco" define a configuração angular do arco.

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

DrawCircle

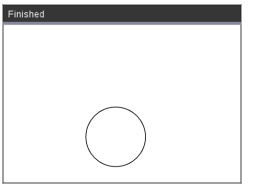
Catálogo > CXII

DrawCircle *x, y, raio*

x, *y*: coordenada do centro

raio: raio do círculo

```
DrawCircle 150,150,40
```



Ver também: FillCircle

DrawLine

Catálogo > CXII

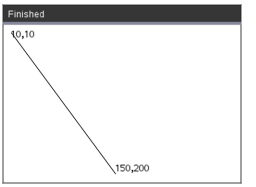
DrawLine *x1, y1, x2, y2*

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 so 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*

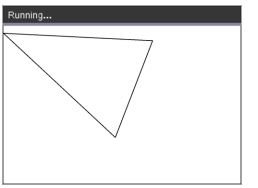
Nota: DrawPoly *xlist, ylist*
A forma irá ligar *x1, y1* a *x2, y2*, *x2, y2* a *x3, y3* e por aí fora.

Nota: DrawPoly *x1, y1, x2, y2, x3, y3...xn, yn*
xn, yn **NÃO** serão automaticamente ligados a *x1, y1*.

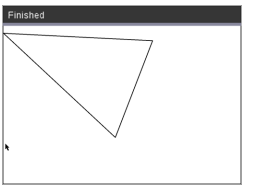
Expressões que se associam a uma lista de números reais flutuante
xlist, ylist

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
```



DrawPoly

Catálogo > CXII

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

Catálogo > CXII

DrawRect *x, y, largura, altura*

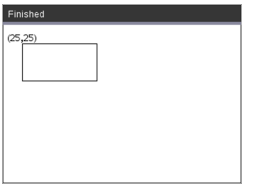
x, y: coordenada superior esquerda do retângulo

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

Catálogo > CXII

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*

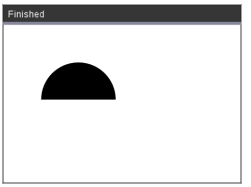
x, *y*: coordenada superior esquerda do retângulo delimitador

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

```
FillArc 50,50,100,100,0,180
```



FillCircle

Catálogo > CXII

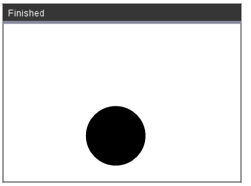
FillCircle *x, y, raio*

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

```
FillCircle 150,150,40
```




```
Aqui!
```

FillPoly

Catálogo > CXII

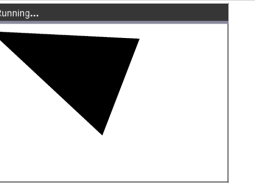
FillPoly *xlist, ylist*

ou

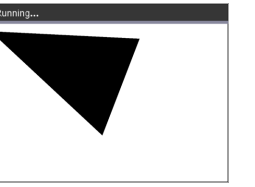
FillPoly *x1, y1, x2, y2, x3, y3...xn, yn*

Nota: A reta e cor são especificadas por SetColor e SetPen

```
xlist:={0,200,150,0}
ylist:={10,20,150,10}
FillPoly xlist,ylist
```




```
FillPoly 0,10,200,20,150,150,0,10
```



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

Catálogo > CXII

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

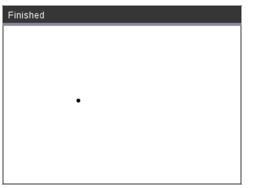
10 - ponto maior, sólido

11 - ponto maior, vazio

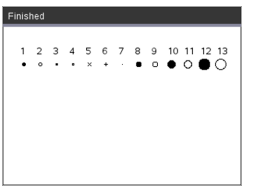
12 - o maior ponto, sólido

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
```



SetColor

Catálogo > CXII

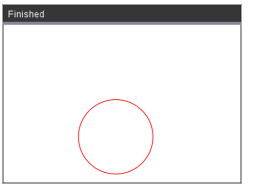
SetColor

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

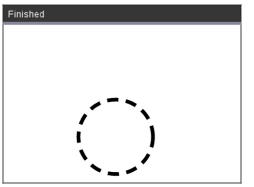
espessura, estilo

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

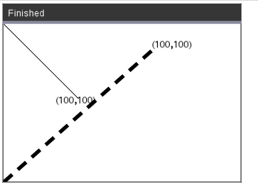
xMin, xMax, yMin, yMax

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 redesenhados na nova configuração.

Para repor os parâmetros da janela para os parâmetros padrão, utilize:

SetWindow 0,0,0,0

UseBuffer **Catálogo > CXII**

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™ 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 41, e **isVoid()**, página 81.

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 ctrl ␣.

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.

$l1:=\{1,2,3,4,5\}$: $l2:=\{2,\_,3,5,6.6\}$	$\{2,\_,3,5,6.6\}$
LinRegMx *l1*,*l2*	*Done*
stat.Resid	$\{0.434286,\_,-0.862857,-0.011429,0.44\}$
stat.XReg	$\{1.,\_,3.,4.,5.\}$
stat.YReg	$\{2.,\_,3.,5.,6.6\}$
stat.FreqReg	$\{1.,\_,1.,1.,1.\}$

Uma categoria omitida nas regressões introduz um nulo para o elemento correspondente do residual.

$l1:=\{1,3,4,5\}$: $l2:=\{2,3,5,6.6\}$	$\{2,3,5,6.6\}$
$cat:=\{"M","M","F","F"\}$: $incl:=\{"F"\}$	$\{"F"\}$
LinRegMx *l1*,*l2*,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 residuo.

$l1:=\{1,3,4,5\}$: $l2:=\{2,3,5,6.6\}$	$\{2,3,5,6.6\}$
LinRegMx *l1*,*l2*,$\{1,0,1,1\}$	*Done*
stat.Resid	$\{0.069231,\_,-0.276923,0.207692\}$
stat.XReg	$\{1.,\_,4.,5.\}$
stat.YReg	$\{2.,\_,5.,6.6\}$
stat.FreqReg	$\{1.,\_,1.,1.\}$

Atalhos para introduzir expressões matemáticas

Os atalhos permitem introduzir elementos das expressões matemáticas, escrevendo, em vez da utilização do Catálogo ou da Palete 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(...)**
Σ() (Modelo da soma)	**sumSeq(...)**
Π() (Modelo da produto)	**prodSeq(...)**
sin^{-1}(), cos^{-1}(), ...	**arcsin(...), arccos(...)**, ...
Δ**List()**	**deltaList(...)**

No teclado do computador

Para introduzir este:	Escreva este atalho:
i (constante imaginária)	**@i**
e (base logarítmica natural e)	**@e**
E (notação científica)	**@E**
^{T} (transpor)	**@t**

Para introduzir este:	Escreva este atalho:
ʳ (radianos)	**@r**
° (graus)	**@d**
g (grados)	**@g**
∠ (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™. 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 (^{r}), 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™ 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→r e "r"→s1, #s1=10.

Pós-operadores

Os pós-operadores são operadores que vêm directamente após um argumento, como 5!, 25% ou 60°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 (−) seguida pelo número. As pós-operações e a exponenciação são efectuadas antes da negação. Por exemplo, o resultado de −x2 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

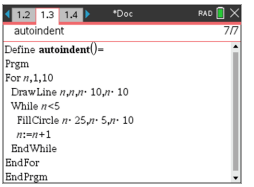
Recuos automáticos no Editor de Programação

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

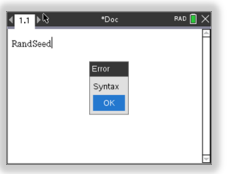
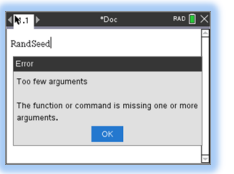
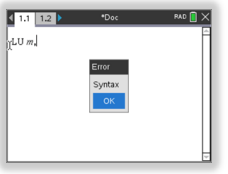
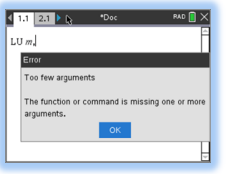
Erros

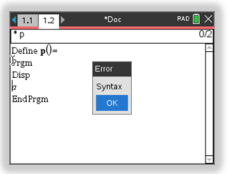
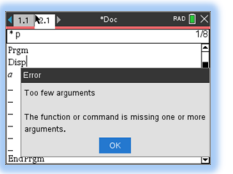
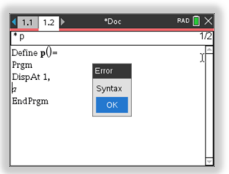
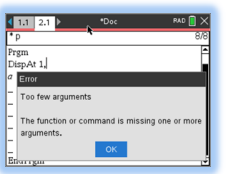
Condição de erro	**Nova mensagem**
Erro na instrução de condição (**Se/Enquanto**)	Uma instrução condicional não foi resolvida como **TRUE (VERDADEIRA)** ou **FALSE (FALSA)** **NOTA:** Com a alteração de colocar o cursor na reta com o erro, deixamos de especificar se o erro é uma instrução "**If (Se)**" 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**

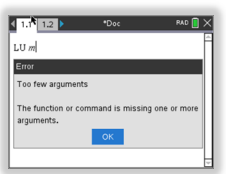
Erros de sintaxe

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
	
	

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. 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	8987551787.3682 _m/_F
_Fc	Constante de Faraday	96485.33289 _coul/_mol
_g	Aceleração da gravidade	9.80665 _m/_s^2
_Gc	Constante gravitacional	6.67408E-11 _m^3/_kg/_s^2
_h	Constante de Planck	6.626070040E-34 _J _s
_k	Constante de Boltzmann	1.38064852E-23 _J/_°K
_μ0	Permeabilidade no vazio	1.2566370614359E-6 _N/_A^2
_μb	Magnetão de Bohr	9.274009994E-24 _J _m^2/_Wb
_Me	Massa de repouso do eletrão	9.10938356E-31 _kg
_Mμ	Massa de Muão	1.883531594E-28 _kg
_Mn	Massa de repouso do neutrão	1.674927471E-27 _kg
_Mp	Massa de repouso do protão	1.672621898E-27 _kg
_Na	Número de Avogradro	6.022140857E23 /_mol
_q	Carga do eletrão	1.6021766208E-19 _coul
_Rb	Raio Bohr	5.2917721067E-11 _m
_Rc	Constante molar do gás	8.3144598 _J/_mol/_°K
_Rdb	Constante de Rydberg	10973731.568508/_m
_Re	Raio do eletrão	2.8179403227E-15 _m
_u	Massa atómica	1.660539040E-27 _kg
_Vm	Volume molar	2.2413962E-2 _m^3/_mol
_ε0	Permissividade no vazio	8.8541878176204E-12 _F/_m
_σ	Constante de Stefan-Boltzmann	5.670367E-8 _W/_m^2/_°K^4
_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**, página 170.

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 If a<b provocará este erro se a ou b forem indefinidos quando a afirmação If 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.

Código de erro	Descrição
170	Limite O limite inferior tem de ser inferior ao limite superior para definir o intervalo da procura.
180	Pausa A tecla esc ou on 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->a, em que a é uma variável indefinida, provocará este erro.
200	Expressão de constrangimento inválida Por exemplo, solve(3x^2-4=0,x) | x<0 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, 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.
565	Programa exterior inválido
570	Nome do caminho inválido

Código de erro	Descrição
	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
740	Falta Then no bloco If..EndIf
750	Nome não é uma função nem um programa

Código de erro	Descrição
765	Nenhuma função seleccionada
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	Demsiados 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** para atribuir valores às variáveis.

Código de erro	Descrição
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
1120	Nenhuma alteração de sinal
1130	O argumento não pode ser uma lista ou matriz

Código de erro	Descrição
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 *xxx**yyy*, em que: • A parte *xxx* pode ter de 1 a 16 caracteres. • A parte *yyy* 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**, **:=**, ou sto →. • 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 ou BibPriv. • Actualizar bibliotecas. Consulte a secção Biblioteca na documentação para obter mais informações.

Código de erro	Descrição
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 Tem de passar os coeficientes do polinómio para valores numéricos.
1310	Erro do argumento: Uma função não conseguiu avaliar um ou mais argumentos.

Código de erro	Descrição
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()**, página 179.

Código de aviso	Mensagem
10000	A operação pode introduzir soluções falsas.
10001	A diferenciação de uma equação pode produzir uma equação falsa.
10002	Solução questionável
10003	Precisão questionável
10004	A operação pode perder as soluções.
10005	cSolve pode especificar mais zeros.
10006	Solve pode especificar mais zeros.
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)
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
10017	Capacidade excedida substituída por ∞ ou -∞
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.

Código de aviso	Mensagem
	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 aritmético aproximado.
10025	A equivalência não pode ser verificada no modo EXACTO.
10026	A restrição pode ser ignorada. Especifique a restrição na forma "\" 'Variable MathTestSymbol Constant' ou uma associação destas formas, por exemplo 'x<3 e 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 mais informações sobre a duração e os termos da garantia ou a assistência.

Garantia Limitada. Esta garantia não afeta os seus direitos legais.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243

Índice remissivo

'

', notação de minutos 206

–

–, subtrair[*] 188

!

!, factorial 198

"

", notação de segundos 206

#

#, indirecta 203
#, operador da indirecta 231

%

%, percentagem 194

&

&, acrescentar 198

*****

*, multiplicar 189

.

.-, ponto subtracção 192
.*, ponto multiplicação 193
./, ponto divisão 193
.^, ponto potência 193
.+, ponto adição 192

/

/, dividir[*] 190

:

:=, atribuir 209

^

^-1, recíproco 207
^, potência 191

|

|, operador de limite 208

+

+, adicionar 188

=

≠, diferente[*] 195
=, igual 194

>

>, maior que 196

∏

∏, produto[*] 200

∑

∑(), soma[*] 201
∑Int() 202
∑Prn() 202

√

√, raiz quadrada[*] 200

∫

∫, integral[*] 200

≤

≤, igual ou menor que 196

≥

≥, igual ou maior que 197

▸

▸, converter para ângulo de gradianos[Grad] 73
▸Base10, visualizar como número inteiro decimal[Base10] ... 18
▸Base16, visualizar como hexadecimal[Base16] 18
▸Base2, visualizar como binário [Base2] 17
▸Cylind, visualizar como vector cilíndrico[Cylind] 36
▸DD, visualizar como ângulo decimal [DD] 37
▸Decimal, visualizar resultado como decimal[Decimal] 38
▸DMS, visualizar como grau/minuto/segundo [DMS] 45
▸Polar, visualizar como vector polar [Polar] 120
▸Sphere, visualizar como vector esférico[Sphere] 157
▸FracçãoAprox( ) 13
▸Rad, converter medida de ângulo para radianos 128
▸Rect, visualizar como vetor rectangular 132

→

→, guardar 209

⇒

⇒, implicação lógica[*] 197, 228

⇔

⇔, implicação lógica dupla[*] 198

©

©, comentário 210

°

°, graus/minutos/segundos[*] 206
°, notação de graus[*] 205

0

0b, indicador binário 210
0h, indicador hexadecimal 210

1

10^( ), potência de dez 207

A

a definir
função ou programa privado .. 39
função ou programa público .. 40
abs( ), valor absoluto 7
acrescentar, & 198
adicionar, + ... 188
aleatória
matriz, randMat( ) 130
norma, randNorm( ) 130
aleatório
polinómio, randPoly( ) 131
semente de número, RandSeed 131
amortTbl( ), tabela de amortização . 7, 16
amostra aleatória 131
and, Boolean operator 8
angle( ), ângulo 9
ângulo, angle( ) 9
ANOVA, análise de variação de uma via 9
ANOVA2way, análise de variação bidireccional 10
Ans, última resposta 12
Apag. 214
apagar
erro, ClrErr 24
approx( ), aproximado 12
Apr., apresentar dados 145
apresentar dados, Apr. 145
aproximado, approx( ) 12
arccos() 13
arccosh() 13
arccot() 14
arccoth() 14
arccsc() 14

arccsch() 14
arco-coseno, $\cos^{-1}$() 28
arco-seno, $\sin^{-1}$() 153
arco-tangente, $\tan^{-1}$() 164
arcsec() 14
arcsech() 14
arcsin() 14
arcsinh() 14
arctan() 14
arctanh() 14
argumentos em funções TVM 174
Argumentos TVM 174
arredondar(), round 141
arredondar, round() 141
atalhos do teclado 228
atalhos, teclado 228
AtualizarVarsSonda 134
augment(), aumentar/concatenar 15
aumentar/concatenar, aumentar() 15
avaliação, ordem de 230
avaliar polinómio, polyEval() 121
avgRC(), taxa de câmbio média 15

B

BibPriv 39
BibPub 40
binário
  indicador, 0b 210
  visualizar, ▸Base2 17
binomCdf() 19, 79
binomPdf() 20
bloquear variáveis e grupos de variáveis 92
Bloquear, bloquear variável ou grupo de variáveis 92
Boolean operators
  and 8

C

cadeia
  comprimento 42
  dimensão, dim() 42
cadeia de caracteres, char() 21
cadeia do formato, format() 57
CadeiaDePedido 137
cadeias
  acrescentar, & 198
  cadeia de caracteres, char() 21
  cadeia para expressão, expr() 53
  código de carácter, ord() 117
  deslocar, shift() 149
  esquerda, left() 82
  expressão para cadeia, string() 161
  formatar 57
  formato, format() 57
  indirecta, # 203
  mid-string, mid() 100
  right, right() 77, 138
  rodar, rotate() 140
  utilizar para criar nomes de variáveis 231
  within, inString 76
caracteres
  cadeia, char() 21
  código numérico, ord() 117
Cdf() 55
ceiling(), ceiling 20
ceiling, ceiling() 20
centralDiff() 20
char(), cadeia de caracteres 21
$\chi^2$2way 21
χ^2GOF 22
χ^2Pdf() 23
χ^2Cdf() 22
ciclo, Cycle 36
ciclo, Loop 96
ClearAZ 23
ClrErr, apagar erro 24
co-seno, cos() 26
co-tangente, cot() 30
códigos de aviso e mensagens 245
colAugment 24
colDim(), dimensão da coluna da matriz 24
colNorm(), norma da coluna da matriz 25

com, | 208
Comando Parar 161
Comando Text 167
Comando Wait 178
combinações, nCr( ) 107
comentário, © 210
complexo
conjugado, conj( ) 25
comprimento da cadeia 42
conj( ), conjugado complexo 25
constructMat( ), construir matriz .. 25
construir matriz, constructMat( ) .. 25
contar condicionalmente itens numa lista , countif( ) 31
contar dias entre datas, dbd( ) 37
contar itens numa lista, contar( ) ... 31
converter
▸Grad 73
▸Rad 128
coordenada polar, R▸Pr( ) 128
coordenada polar, R▸Pθ( ) 128
copiar variável ou função, CopyVar . 26
corrMat( ), matriz de correlação ... 26
cos^{-1} , arco-coseno 28
cos( ), co-seno 26
cosh^{-1}( ), arco-coseno hiperbólico .. 29
cosh( ), co-seno hiperbólico 28
cot^{-1}( ), arco-cotangente 30
cot( ), co-tangente 30
coth^{-1}( ), arco-cotangente hiperbólico 31
coth( ), co-tangente hiperbólica 30
count( ), contar itens numa lista ... 31
countif( ), contar condicionalmente itens numa lista 31
cPolyRoots() 32
crossP( ), produto cruzado 32
csc^{-1}( ), co-secante inversa 33
csc( ), co-secante 33
csch^{-1}( ), co-secante hiperbólica inversa 34
csch( ), co-secante hiperbólica 34
CubicReg, regressão cúbica 34
Cycle, ciclo 36

D

d ( ), primeira derivada 199
dbd( ), dias entre datas 37
decimal
visualizar ângulo, ▸DD 37
visualizar número inteiro, ▸Base10 18
definição, Lbl 82
definições do modo, getMode( ) ... 70
definições, obter actual 70
definir
modo, setMode( ) 147
Definir 38
Definir BibPriv 39
Definir BibPub 40
Definir, definir 38
DelVar, eliminar variável 41
delVoid( ), remover elementos nulos 41
densidade da probabilidade, normPdf( ) 112
densidade de probabilidade student-t , tPdf( ) 169
derivada
numérica, nDerivative( ) 108
derivadas
derivada numérica, nDeriv( ) ... 109
derivada numérica, nDerivative( ) 108
primeira derivada, d( ) 199
desbloquear variáveis e grupos de variáveis 177
Desbloquear, desbloquear variável ou grupo de variáveis 177
desenhar 215-217
deslocar, shift( ) 149
desvio padrão, stdDev( ) 159-160, 177
det( ), determinante da matriz 41
diag( ), diagonal da matriz 42
dias entre datas, dbd( ) 37
diferente, ≠ 195
dim( ), dimensão 42
dimensão, dim( ) 42
direita( ), right 138
direita, right( ) 77, 138

Disp, visualizar dados 43
DispAt 43
dividir, / 190
divisão inteira, intDiv() 77
dotP( ), produto do ponto 46

E

E , expoente 204
e para uma potência, e^( ) 46, 52
e^( ), e para uma potência 46
eff( ), converter taxa nominal para efectiva 47
eigVc( ), vector eigen 47
eigVl( ), valor próprio 47
elementos (nulos) vazios 226
elementos nulos 226
elementos nulos, remover 41
eliminar
elementos nulos da lista 41
variável, DelVar 41
else if, ElseIf 48
else, Else 74
ElseIf, else if 48
end
for, EndFor 57
função, EndFunc 61
loop, EndLoop 96
programa, EndPrgm 123
end function, EndFunc 61
end loop, EndLoop 96
EndWhile, terminar enquanto 180
enquanto, While 180
EOS (Equation Operating System) .. 230
equações simultâneas, simult( ) 151
Equation Operating System (EOS) .. 230
erro de passagem, PassErr 119
erros e resolução de problemas
apagar erro, ClrErr 24
erro de passagem, PassErr 119
esquerda, left( ) 82
estatística
combinações, nCr( ) 107
desvio padrão, stdDev( ) ..159-160, 177
estatística de uma variável, OneVar 115
factorial, ! 198
média, mean( ) 98
mediana, median( ) 99
norma aleatória, randNorm( ) .. 130
permutações, nPr( ) 113
resultados de duas variáveis, TwoVar 174
semente de número aleatório, RandSeed 131
variação, variance( ) 177
estatística de uma variável, OneVar . 115
euler( ), Euler function 49
exclusão com operador "|" 208
Exit, sair 51
exp( ), e para uma potência 52
Expoente e
modelo para 2
expoente, E 204
expoentes
modelo para 1
expr( ), cadeia para expressão 53
ExpReg, refrsessão exponencial 53
expressões
cadeia para expressão, expr( ) .. 53

F

factor( ), factor 54
factor, factor( ) 54
factorial, ! 198
factorização QR, QR 125
Fill, preencher matriz 55
FiveNumSummary 55
floor( ), floor 56
floor, floor( ) 56
For 57
for, For 57
For, for 57
forma de escalão-linha reduzida, rref( ) 143
forma de escalão-linha, ref( ) 133
format( ), cadeia do formato 57

fpart( ), parte da função 58
fracção própria, propFrac 124
fracções
modelo para 1
propFrac 124
fracções mistas, com propFrac(› com 124
freqTable( ) 59
frequência( ) 59
Func, função 61
Func, função do programa 61
função por ramos (2 ramos)
modelo para 2
função por ramos (N-ramos)
modelo para 3
funções
definidas pelo utilizador 38
função do programa, Func 61
parte, fpart( ) 58
funções de distribuição
binomCdf( ) 19, 79
binomPdf( ) 20
invNorm( ) 79
invt( ) 80
Invχ^2( ) 78
normCdf( ) 111
normPdf( ) 112
poissCdf( ) 119
poissPdf( ) 120
tCdf( ) 166
tPdf( ) 169
$\chi^2$2way( ) 21
χ^2Cdf( ) 22
χ^2GOF( ) 22
χ^2Pdf( ) 23
funções definidas pelo utilizador ... 38
funções e programas definidos pelo utilizador 39-40
funções e variáveis
a copiar 26
funções financeiras, tvmFV( ) 173
funções financeiras, tvmI( ) 173
funções financeiras, tvmN( ) 173
funções financeiras, tvmPmt( ) 173
funções financeiras, tvmPV( ) 174

G

g , gradianos 204
gcd( ), máximo divisor comum 62
geomCdf( ) 62
geomPdf( ) 63
Get 63, 220
getDenom( ), obter denominador .. 64
getKey() 64
getLangInfo( ), obter/apresentar informações do idioma ... 69
getLockInfo( ), testar o estado de bloqueio da variável ou do grupo de variáveis 69
getMode( ), obter definições do modo 70
getNum( ), obter número 71
GetStr 71
getType( ), get type of variable 71
getVarInfo( ), obter/apresentar informações das variáveis . 72
Goto, ir para 73
grupos, bloquear e desbloquear ... 92, 177
grupos, testar estado de bloqueio .. 69
guardar
símbolo, & 209

H

hexadecimal
indicador, 0h 210
visualizar, ▸Base16 18
hiperbólica
tangente, tanh( ) 165
hiperbólico
arco-coseno, $\cosh^{-1}$( ) 29
arco-seno, $\sinh^{-1}$( ) 154
arco-tangente, tanh / ( ) 166
co-seno, cosh( ) 28
seno, sinh( ) 154

I

identity( ), matriz identidade 73

idioma
obter informações do idioma .. 69
ifFn () 75
igual ou maior que, | 197
igual ou menor que, { 196
igual, = ... 194
imag( ), parte imaginária 76
implicação lógica dupla, ⇔ 198
implicação lógica, ⇒ 197, 228
indirecta, # 203
inString( ), na cadeia 76
int( ), parte inteira 77
intDiv( ), divisão inteira 77
integral definido
modelo para 6
integral, ∫ 200
interpolate( ), interpolar 77
inv F () 78
inverso, ^-¹ 207
invNorm((distribuição normal
cumulativa inversa) 79
invNorm( ), normal cumulativa
inversa) 79
invt( ) 80
Invχ²( ) 78
iPart( ), parte inteira 80
ir para, Goto 73
irr( ), taxa de retorno interno
internal rate of return, irr( ) 80
isPrime( ), teste da plica 81
isVoid( ), teste para nulo 81

L

Lbl, definição 82
lcm, mínimo múltiplo comum 82
left( ), esquerda 82
limite máximo, limite máximo( ) 20, 32
LinRegBx, regressão linear 83
LinRegMx, regressão linear 85
LinRegtIntervals, regressão linear ... 86
LinRegtTest 87
linSolve() 89
Δlist( ), diferença da lista 89
list▸mat( ), lista para matriz 90
lista para matriz, list▸mat( ) 90
lista, contar condicionalmente itens
numa 31
lista, contar itens em 31
ListaDelta() 40
listas
aumentar/concatenar,
aumentar( ) 15
diferença, Δlist( ) 89
diferenças numa lista, @ list( ) .. 89
elementos vazios em 226
lista para matriz, list▸mat( ) 90
matriz para lista, mat▸ista( ) 97
máximo, max( ) 98
mid-string, mid( ) 100
mínimo, min( ) 101
nova, newList( ) 108
ordenar ascendente, SortA 156
ordenar descendente, SortD .. 157
produto cruzado, crossP( ) 32
produto do ponto, dotP( ) 46
produto, product( ) 123
soma cumulativa,
SomaCumulativa( ) ... 35
soma, sum( ) 162
ln( ), logaritmo natural 90
LnReg, regressão logarítmica 91
local, Local 92
Local, variável local 92
Log
modelo para 2
logaritmo natural, ln( ) 90
logaritmos 90
LogisticD, regressão logística 95
Loop, ciclo 96
LU, decomposição inferior-superior
da matriz 97

M

maior que, > 196
mat▸list( ), matriz para lista 97

matriz (1 × 2)
modelo para 4
matriz (2 × 1)
modelo para 4
matriz (2 × 2)
modelo para 4
matriz (m × n)
modelo para 4
matriz de correlação, corrMat( ) ... 26
matriz identidade, identity() 73
matriz para lista, mat▸list( ) 97
matrizes
adição de linha, rowAdd( ) 142
adição e multiplicação da linha, mRowAdd( ) 103
aleatórias, randMat( ) 130
aumentar/concatenar, aumentar( ) 15
decomposição inferior-superior, LU 97
determinante, det( ) 41
diagonal, diag( ) 42
dimensão da coluna, colDim( ) . 24
dimensão da linha, rowDim( ) .. 142
dimensão, dim( ) 42
factorização QR, QR 125
forma de escalão-linha reduzida, rref( ) 143
forma de escalão-linha, ref( ) ... 133
identity, identity() 73
lista para matriz, list▸mat( ) 90
matriz para lista, mat▸list( ) 97
máximo, max( ) 98
mínimo, min( ) 101
norma da coluna, colNorm( ) .. 25
norma da linha, rowNorm( ) ... 142
nova, newMat( ) 109
operação da linha, mRow( ) 103
ponto adição, .+ 192
ponto divisão, ./ 193
ponto multiplicação, .* 193
ponto potência, .^ 193
ponto subtracção, .- 192
preencher, Fill 55
produto, product( ) 123
soma cumulativa, SomaCumulativa( ) ... 35
soma, sum( ) 162
submatriz, subMat( ) 161, 163
transpor, T 163
troca da linha, rowSwap( ) 143
valor próprio, eigVl( ) 47
vector eigen, eigVc( ) 47
max( ), máximo 98
máximo divisor comum, gcd( ) 62
máximo, max( ) 98
mean( ), média 98
média, mean( ) 98
median( ), mediana 99
mediana, median( ) 99
MedMed, regressão da recta média-média 99
mid-string, mid( ) 100
mid( ), mid-string 100
min( ), mínimo 101
mínimo múltiplo comum, lcm 82
mínimo, min( ) 101
mirr( ), taxa de retorno interna modificada 102
mod( ), módulo 102
modelos
expoente 1
Expoente e 2
fracção 1
função por ramos (2 ramos) ... 2
função por ramos (N-ramos) ... 3
integral definido 6
Log 2
matriz (1 × 2) 4
matriz (2 × 1) 4
matriz (2 × 2) 4
matriz (m × n) 4
primeira derivada 5
produto (P) 5
raiz de índice N 2
raiz quadrada 1
segunda derivada 6

sistema de equações (2 equações) 3
sistema de equações (N equações) 3
soma (G) 5
valor absoluto 4
modos
definir, setMode( ) 147
módulo, mod( ) 102
mRow( ), operação da linha da matriz 103
mRowAdd( ), adição e multiplicação da linha da matriz 103
multiplicar, * 189
MultReg 103
MultRegIntervals( ) 104
MultRegTests( ) 105

N

na cadeia, inString() 76
nand, Operador booleano 106
nCr( ), combinações 107
nDerivative( ), derivada numérica .. 108
negação, introduzir números negativos 231
newList( ), nova lista 108
newMat( ), nova matriz 109
nfMax( ), função numérica máxima . 109
nfMin( ), função numérica mínima .. 109
nInt( ), integral numérico 110
nom), converter taxa efectiva para nominal 110
nor, Operador booleano 110
norma Frobenius, norma( ) 111
norma( ), norma Frobenius 111
normCdf( ) 111
normPdf( ) 112
not, Operador booleano 112
notação de gradianos, g 204
notação de grau/minuto/segundo . 206
notação de graus, ° 205
notação de minutos, ... 206
notação de segundos, " 206
nova
lista, newList( ) 108
matriz, newMat( ) 109
nPr( ), permutações 113
npv( ), valor líquido actual 114
nSolve( ), solução numérica 114
nulo, teste para 81
numérica
derivada, nDeriv( ) 109
solução, nSolve( ) 114
numérico
integral, nInt( ) 110

O

obter
denominador, getDenom( ) ... 64
número, getNum( ) 71
obter/apresentar
informações das variáveis, getVarInfo( ) 69, 72
OneVar, estatística de uma variável . 115
operador da indirecta (#) 231
operador de limite "|" 208
operador de limite, ordem de avaliação 230
operadores
ordem de avaliação 230
Operadores booleanos
⇒ 197, 228
⇔ 198
nand 106
nor 110
not 112
ou 116
xou 181
ord( ), código de carácter numérico 117
ordenar
ascendente, SortA 156
descendente, SortD 157
ou (Booleano), or 116
ou, Operador booleano 116

P

P▸Rx( ), rectangular x coordenada .. 118
P▸Ry( ), rectangular y coordenada .. 118
parte imaginária, imag() 76
parte inteira do número, iPart() 80
parte inteira, int() 77
PassErr, erro de passagem 119
Pdf( ) 58
Pedido 135
percentagem, % 194
permutações, nPr( ) 113
piecewise( ) 119
poissCdf( ) 119
poissPdf( ) 120
polar
  visualizar vector, ▸Polar 120
polinómio
  aleatórios, randPoly( ) 131
polinómios
  avaliar, polyEval( ) 121
polyEval( ), avaliar polinómio 121
PolyRoots() 121
ponto
  adição, .+ 192
  divisão,./ 193
  multiplicação, .* 193
  potência, .^ 193
  produto, dotP( ) 46
  subtracção, .- 192
potência de dez, 10^( ) 207
potência, ^ 191
PowerReg, regressão de potência .. 121
preencher 218-219
Prgm, definir programa 123
primeira derivada
  modelo para 5
probabilidade da distribuição
  normal, normCdf( ) 111
probabilidade da distribuição
  student-t , tCdf( ) 166
product( ), produto 123
produto (P)
  modelo para 5
produto cruzado, crossP( ) 32
produto, ∏( ) 200
produto, product( ) 123
programação
  apresentar dados, Apr. 145
programar
  definir programa, Prgm 123
  erro de passagem, PassErr 119
  visualizar dados, Disp 43
programas
  definir biblioteca privada 39
  definir biblioteca pública 40
programas e programação
  apagar erro, ClrErr 24
  apresentar ecrã I/O, Apr. 145
  terminar programa, EndPrgm . 123
  visualizar ecrã E/S, Disp 43
propFrac, fracção própria 124

Q

QR, factorização QR 125
QuadReg, regressão quadrática 126
quando, when( ) 179
QuartReg, regressão quártica 127

R

R , radianos 204
R▸Pr( ), coordenada polar 128
R▸Pθ( ), coordenada polar 128
RacionalAprox( ) 13
radianos, R 204
raiz de índice N
  modelo para 2
raiz quadrada
  modelo para 1
raiz quadrada, √( ) 157, 200
rand( ), número aleatório 129
randBin, número aleatório 129
randInt( ), inteiro aleatório 129
randMat( ), matriz aleatória 130
randNorm( ), norma aleatória 130
randPoly( ), polinómio aleatório 131
randSamp( ) 131

RandSeed, semente de número aleatório 131
real( ), real 131
real, real( ) 131
recíproco, ^–1 207
rectangular x coordenada, P▸Rx( ) .. 118
rectangular y coordenada, P▸Ry( ) .. 118
ref( ), forma de escalão-linha 133
regressão cúbica, CubicReg 34
regressão da recta média-média, MedMed 99
regressão de potência, PowerReg 121, 135, 137
regressão exponencial, ExpReg 53
regressão linear, LinRegAx 85
regressão linear, LinRegBx 83, 86
regressão logarítmica, LnReg 91
regressão logística, LogisticD 95
regressão potencial, PowerReg 121, 167
regressão quadrática, QuadReg 126
regressão quártica, QuartReg 127
regressão sinusoidal, SinReg 155
regressões
  cúbica, CubicReg 34
  exponencial, ExpReg 53
  logarítmica, LnReg 91
  logística, Logística 95
  MultReg 103
  quadrática, QuadReg 126
  quártica, QuartReg 127
  recta média-média, MedMed .. 99
  regressão de potência, PowerReg 121, 135, 137
  regressão linear, LinRegAx 85
  regressão linear, LinRegBx 83, 86
  regressão potencial, PowerReg 121, 167
  sinusoidal, SinReg 155
remain( ), resto 135
remover
  elementos nulos da lista 41
resposta (última), Ans 12
resto, remain() 135
resultados de duas variáveis, TwoVar 174
resultados, estatística 158
right, right( ) 49, 179
rk23( ), função Runge Kutta 138
rodar( ), rotate 140
rodar, rotate( ) 140
rowAdd( ), adição da linha da matriz 142
rowDim( ), dimensão da linha da matriz 142
rowNorm( ), norma da linha da matriz 142
rowSwap( ), troca da linha da matriz 143
rref( ), forma de escalão-linha reduzida 143

S

sair, Exit 51
se, If 74
Se, if 74
sec–1( ), secante inversa 144
sec( ), secante 143
sech–1( ), secante hiperbólica inversa 144
sech( ), secante hiperbólica 144
segunda derivada modelo para 6
seno, sin( ) 152
seq( ), sequência 145
seqGen( ) 146
seqn( ) 147
SeqProd() 123
SeqSom() 163
sequence, seq( ) 146-147
sequência, seq( ) 145
setMode( ), definir modo 147
shift( ), deslocar 149
sign( ), sinal 151
simult( ), equações simultâneas 151
sin–1( ), arco-seno 153
sin( ), seno 152
sinal, sign( ) 151
sinh–1( ), arco-seno hiperbólico 154
sinh( ), seno hiperbólico 154
SinReg, regressão sinusoidal 155
sistema de equações (2 equações) modelo para 3

sistema de equações (N equações)
modelo para 3
soma (G)
modelo para 5
soma cumulativa, SomaCumulativa( ) 35
soma de pagamentos principais 202
soma dos pagamentos de juros 202
soma, sum( ) 162
soma, Σ( ) 201
SomaCumulativa( ), soma cumulativa 35
SortA, ordenar ascendente 156
SortD, ordenar descendente 157
sqrt( ), raiz quadrada 157
stat.results 158
stat.values 159
stdDevPop( ), desvio padrão da população 159
stdDevSamp( ), desvio padrão da amostra 160
string( ), expressão para cadeia 161
strings
right, right( ) 49, 179
subMat( ), submatriz 161, 163
submatriz, subMat( ) 161, 163
substituição com operador "|" 208
subtrair, - 188
sum( ), soma 162
sumIf( ) 162

T

T, transpor 163
tabela de amortização, amortTbl( ) . 7, 16
$\tan^{-1}$( ), arco-tangente 164
tan( ), tangente 164
tangente, tan( ) 164
$\tanh^{-1}$( ), arco-tangente hiperbólico 166
tanh( ), tangente hiperbólica 165
taxa de câmbio média, avgRC( ) 15
taxa de retorno interna modificada, mirr( ) 102
taxa efectiva, eff( ) 47
taxa nominal, nom( ) 110
tCdf( ), probabilidade da distribuição student t 166
terminar
enquanto, EndWhile 180
if, EndIf 74
terminar enquanto, EndWhile 180
terminar se, EndIf 74
Test_2S, Teste F de 2 amostras 60
teste da plica, isPrime() 81
Teste F de 2 amostras 60
teste para nulo, isVoid( ) 81
teste t , tTest 171
Teste t de regressões lineares múltiplas 105
tInterval, t intervalo de confiança .. 167
tInterval_2Samp, intervalo de confiança t de duas amostras 168
tPdf( ), densidade de probabilidade student t 169
transpor, T 163
tTest, teste t 171
tTest_2Samp, teste t de duas amostras 172
tvmFV( ) 173
tvmI( ) 173
tvmN( ) 173
tvmPmt( ) 173
tvmPV( ) 174
TwoVar, resultados de duas variáveis 174

U

unitV( ), vector da unidade 176

V

valor absoluto
modelo para 4
valor líquido actual, npv( ) 114
valor próprio, eigVl( ) 47
valor temporal do dinheiro, juro ... 173
valor temporal do dinheiro, montante do pagamento . 173
valor temporal do dinheiro, número de pagamentos 173

valor temporal do dinheiro, valor actual 174
valor temporal do dinheiro, Valor futuro 173
valores dos resultados, estatística .. 159
variação, variance( ) 177
variáveis
apagar todas as letras individuais 23
eliminar, DelVar 41
local, Local 92
variáveis, bloquear e desbloquear 69, 92, 177
variável
criar nome a partir de uma cadeia de caracteres .. 231
variável e funções
a copiar 26
variável local, Local 92
varPop( ) 177
varSamp( ), variação da amostra ... 177
vector eigen, eigVc( ) 47
vector unitário, unitV( ) 176
vectores
produto cruzado, crossP( ) 32
produto do ponto, dotP( ) 46
unidade, unitV( ) 176
visualizar vector cilíndrico, ▸Cylind 36
visualizar como
ângulo decimal, ▸DD 37
binário, ▸Base2 17
grau/minuto/segundo, ▸DMS .. 45
hexadecimal, ▸Base16 18
número inteiro decimal, ▸Base10 18
vector , ▸Polar 120
vector cilíndrico, ▸Cylind 36
vector esférico, ▸Sphere 157
visualizar como vetor rectangular, ▸Rect 132
visualizar dados, Disp 43
visualizar grau/minuto/segundo, ▸DMS 45
visualizar vector cilíndrico, ▸Cylind .. 36
visualizar vector esférico, ▸Sphere .. 157
visualizar vetor rectangular, ▸Rect .. 132
voltar, Return 137
Voltar, return 137

W

warnCodes( ), Warning codes 179
when( ), quando 179
While, enquanto 180

X

x² , quadrado 192
XNOR 198
xou, Booleano exclusivo ou 181

Z

zInterval, z intervalo de confiança .. 182
zInterval_1Prop, intervalo de confiança z de uma proporção 182
zInterval_2Prop, intervalo de confiança z de duas proporções 183
zInterval_2Samp, intervalo de confiança z de duas amostras 183
zTest 184
zTest_1Prop, teste z de uma proporção 185
zTest_2Prop, teste z de duas proporções 186
zTest_2Samp, teste z de duas amostras 186