Guia do Editor de programas da TI-Nspire™

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

Informação importante

Exceto quando indicado de outra forma expressamente na Licença que acompanha um programa, a Texas Instruments não oferece nenhuma garantia, expressa ou implícita, incluindo, entre outras, quaisquer garantias implícitas de comercialização e adequação para um propósito específico, em relação a qualquer programa ou material de livro e faz isso materiais disponíveis exclusivamente em "como está". Em nenhum caso, a Texas Instruments será responsável por qualquer pessoa por danos especiais, colaterais, incidentais ou conseqüentes relacionados com ou decorrentes da compra ou uso desses materiais, e a única e exclusiva responsabilidade da Texas Instruments, independentemente da forma de ação, não deve exceder o valor estabelecido na licença para o programa. Além disso, a Texas Instruments não será responsável por qualquer reclamação de qualquer tipo contra o uso desses materiais por qualquer outra parte.

O software TI-Nspire™ usa o Lua como ambiente de script. Para informações sobre direitos autorais e licenças, consulte http://www.lua.org/license.html.

O software TI-Nspire™ usa Chipmunk Physics versão 5.3.4 como ambiente de simulação. Para obter informações sobre licenças, consulte http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/.

Microsoft® e Windows® são marcas registradas da Microsoft Corporation nos Estados Unidos e / ou em outros países.

Mac OS®, iPad® e OS X® são marcas registradas da Apple Inc.

Unicode® é uma marca registrada da Unicode, Inc. nos Estados Unidos e em outros países.

Os produtos reais podem variar ligeiramente das imagens fornecidas.

Conteúdo

Como começar com o Editor de programas

Pode criar programas ou funções definidas pelo utilizador, introduzindo as afirmações da definição na linha de entrada da Calculadora ou utilizando o Editor de programas. O Editor de programas oferece algumas vantagens, descritas nesta secção. Para obter mais informações, consulte a secção *Calculadora*.

- O editor tem modelos de programação e caixas de diálogo para ajudar a definir funções e programas com a sintaxe correta.
- O editor permite introduzir instruções de programação em diferentes linhas sem requerer uma sequência principal especial para adicionar cada linha.
- Pode criar facilmente objetos da biblioteca pública e privada (variáveis, funções e programas). Para mais informações, consulte *Bibliotecas*.

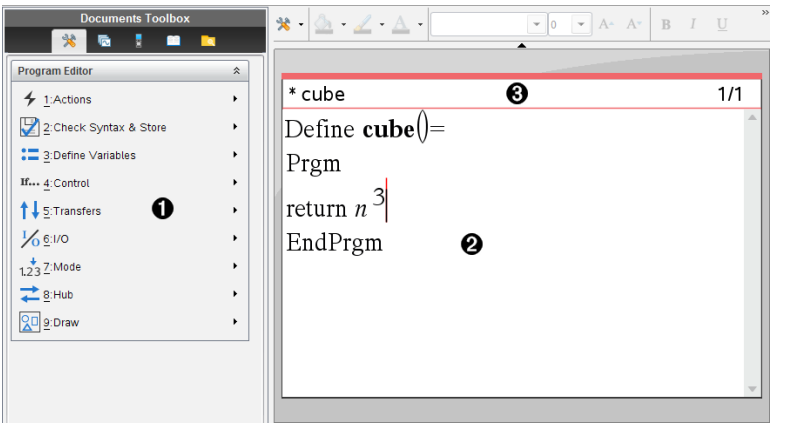
Iniciar o Editor de programas

▶ Para adicionar uma nova página do Editor de programas ao problema atual:

Na barra de ferramentas, clique em **Inserir > Editor de programas > Novo**.

Unidade portátil: Prima doc e selecione **Inserir > Editor de programas > Novo**.

Nota: Também é possível aceder ao editor a partir do menu **Funções & Programas** de uma página da Calculadora.



❶ Menu Editor de programas – Este menu está disponível sempre que estiver na área de trabalho do Editor de programas em modo de vista Normal.

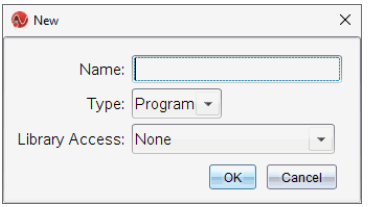
❷ Área de trabalho do Editor de programas

❸ A linha de estado mostra informações do número da linha e o nome da função ou do programa a editar. Um asterisco (*) indica que esta função está "suja", o que significa que foi alterada desde a última vez que a sintaxe foi verificada e guardada.

Definir um programa ou uma função

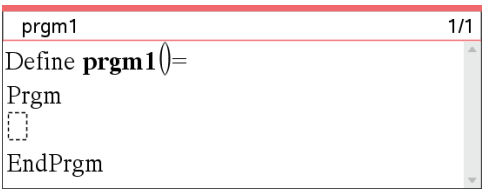
Iniciar um novo Editor de programas

1. Certifique-se de que está no documento no qual pretende criar o programa ou função.
2. Clique no botão **Insert (Inserir)** na barra de ferramentas da aplicação e selecione **Program Editor (Editor de Programas) > New (Novo)**. (Na unidade portátil, prima doc e selecione **Insert (Inserir) > Program Editor (Editor de Programas) > New (Novo)**.



3. Escreva um nome para a função ou o programa que está a definir.
4. Selecione o **Type (Tipo)** (**Program (Programa)** ou **Function (Função)**).
5. Defina o **Acesso à biblioteca**:
   - Se quiser utilizar a função ou o programa apenas a partir do documento e do problema atuais, selecione **None (Nenhum)**.
   - Toque em **LibPriv** para fazer com que a função ou o programa seja acessível a partir de qualquer documento, mas não visível no Catálogo.
   - Se quiser que a função ou o programa seja acessível a partir de qualquer documento e também visível no Catálogo, selecione **LibPub (Show in Catalog / Mostrar no Catálogo)**. Para mais informações, consulte *Bibliotecas*.
6. Clique em **OK**.

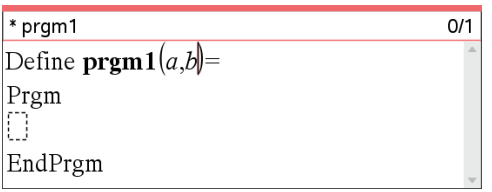
   Uma nova instância do Editor de Programas abre-se com um modelo correspondente às seleções efetuadas.



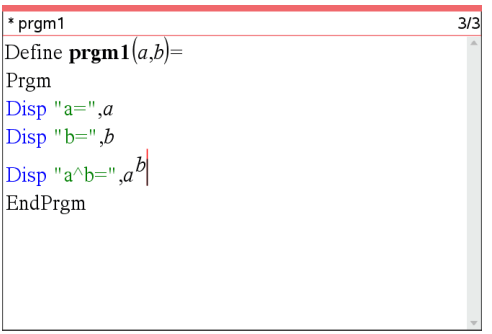
Introduzir linhas numa função ou num programa

O Editor de programas não executa os comandos nem avalia as expressões à medida que as escreve. São executadas apenas quando avaliar a função ou executar o programa.

1. Se a função ou o programa requerer que o utilizador forneça argumentos, escreva os nomes dos parâmetros nos parêntesis a seguir ao nome. Separe os parâmetros com uma vírgula.



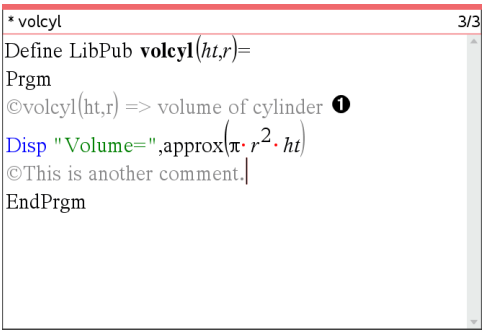
2. Entre as linhas Func e EndFunc (ou Prgm e EndPrgm), introduza as linhas de instruções que compõem a função ou o programa.



- Pode escrever os nomes das funções e dos comandos ou inseri-los a partir do Catálogo.
- Uma linha pode ser mais longa do que a largura do ecrã; nesse caso, desloque para ver a instrução completa.
- Depois de escrever cada linha, prima **Enter**. Este procedimento insere uma nova linha em branco e permite continuar a escrever noutra linha.
- Utilize as teclas de setas ◄, ►, ▲, e ▼ para percorrer a função ou o programa e introduzir ou editar comandos.

Inserir comentários

Os comentários podem ser úteis para alguém que veja ou edite o programa. Os comentários não aparecem durante a execução do programa e não têm efeitos no fluxo do programa. O símbolo © é exibido no início da linha com o comentário.



❶ Comentário que mostra a sintaxe necessária. Como este objeto da biblioteca é público e este comentário é a primeira linha num bloco Func ou Prgm, o comentário aparece no Catálogo como ajuda. Para mais informações, consulte *Bibliotecas*.

Para inserir um comentário:

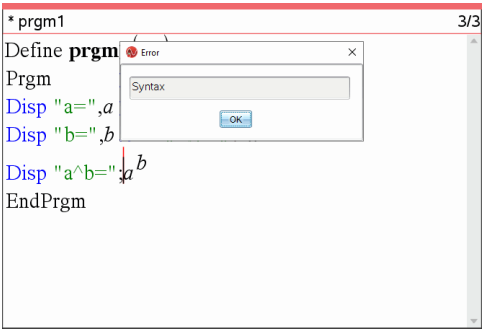
1. Posicione o cursor no fim da linha em que pretende inserir um comentário.
2. No menu **Actions (Ações)**, clique em **Insert Comment (Inserir Comentário)**, ou prima **Ctrl+T**.
3. Escreva o texto do comentário após o símbolo ©.

Verificar sintaxe

O Editor de Programas permite verificar a função ou o programa para a sintaxe correta.

- A partir do menu **Check Syntax & Store (Verificar Sintaxe e Armazenar)**, clique em **Check Syntax (Verificar 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.



Guardar a função ou o programa

Tem de guardar a função ou o programa para o tornar acessível. O Editor de programas verifica automaticamente a sintaxe antes de guardar.

Um asterisco (*) aparece no canto superior esquerdo do Editor de Programas para indicar que a função ou o programa não foi guardado.

- A partir do menu **Check Syntax & Store (Verificar Sintaxe e Armazenar)**, clique em **Check Syntax & Store (Verificar Sintaxe e Armazenar)**.

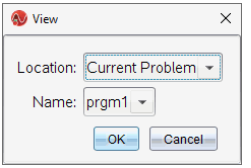
  Se o verificador de sintaxe encontrar algum erro de sintaxe, apresenta uma mensagem de erro e tenta posicionar o cursor junto ao primeiro erro.

  Se não for localizado nenhum erro de sintaxe, a mensagem "Guardado com sucesso" aparece na linha de estado no topo do Editor de programas.

Nota: Se a função ou o programa for definido como um objecto da biblioteca, tem também de guardar o documento na pasta da biblioteca indicada e atualizar as bibliotecas para tornar o objeto acessível a outros documentos. Para mais informações, consulte *Bibliotecas*.

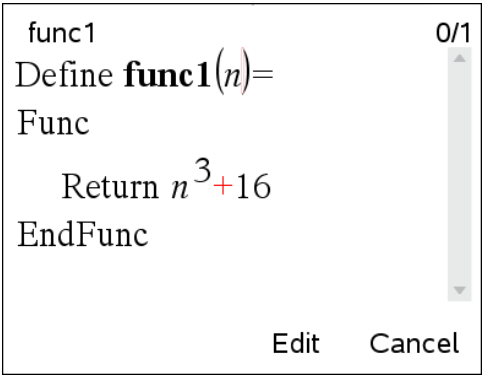
Ver um programa ou uma função

1. No menu **Acções**, seleccione **Ver**.



2. Se a função ou o programa for um objecto da biblioteca, seleccione a biblioteca da lista **Localização**.
3. Seleccione o nome da função ou do programa da lista **Nome**.

   A função ou o programa aparece num visualizador.



4. Utilize as teclas de setas para ver a função ou o programa.
5. Se quiser editar o programa, clique em **Editar**.

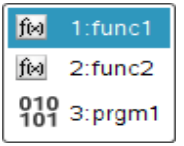
   Nota: A selecção de **Editar** só está disponível para funções e programas definidos no problema actual. Para editar um objecto da biblioteca, tem de abrir primeiro o documento da biblioteca.

Abrir uma função ou um programa para edição

Só pode abrir uma função ou um programa a partir do problema actual.

Nota: Não pode modificar uma função ou um programa bloqueado. Para desbloquear o objecto, vá para a página Calculadora e utilize o comando **Desbloquear** (**unLock**).

1. Veja a lista de programas e funções disponíveis.
   - No menu **Acções**, seleccione **Abrir**.

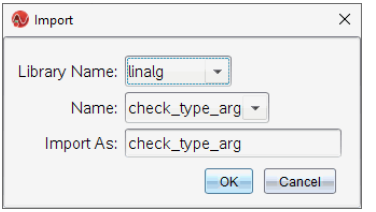


2. Seleccione o item para abrir.

Importar um programa de uma biblioteca

Pode importar uma função ou um programa definido como um objecto da biblioteca para um Editor de programas no problema actual. A cópia importada não está bloqueada, mesmo que o original esteja bloqueado.

1. No menu **Acções**, seleccione **Importar**.



2. Seleccione o **Nome da biblioteca**.
3. Seleccione o **Nome** do objecto.
4. Se quiser que o objecto importado tenha um nome diferente, escreva o nome em **Importar como**.

Criar uma cópia de uma função ou de um programa

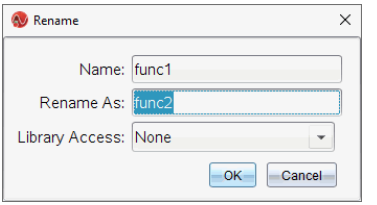
Quando criar um novo programa ou função, pode ser mais fácil começar com uma cópia do programa ou da função actual. A cópia criada não está bloqueada, mesmo que o original esteja bloqueado.

1. No menu **Acções**, seleccione **Criar cópia**.
2. Escreva um novo nome ou clique em **OK** para aceitar o nome proposto.
3. Se quiser alterar o nível de acesso, seleccione **Acesso à biblioteca** e seleccione um nível novo.

Renomear um programa ou uma função

Pode renomear e alterar (opcionalmente) o nível de acesso do programa ou da função actual.

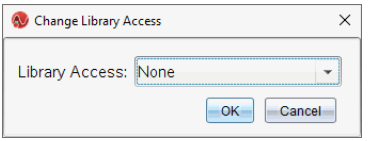
1. No menu **Acções**, seleccione **Renomear**.



2. Escreva um novo nome ou clique em **OK** para aceitar o nome proposto.
3. Se quiser alterar o nível de acesso, seleccione **Acesso à biblioteca** e seleccione um nível novo.

Alterar o nível de acesso à biblioteca

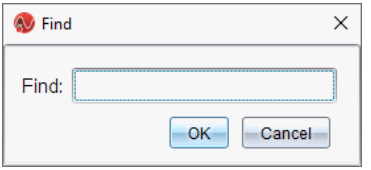
1. No menu **Acções**, seleccione **Alterar acesso à biblioteca**.



2. Seleccione o **Acesso à biblioteca**:
   - Se quiser utilizar apenas a função ou o programa do problema actual da Calculadora, seleccione **Nenhum**.
   - Se quiser que a função ou o programa seja acessível a partir de qualquer documento, mas não visível no Catálogo, seleccione **LibPriv**.
   - Se quiser que a função ou o programa seja acessível a partir de qualquer documento e também visível no Catálogo, seleccione **LibPub**.

Localizar texto

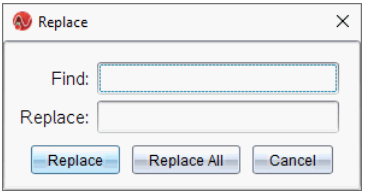
1. No menu **Acções**, seleccione **Localizar**.



2. Escreva o texto que pretende localizar e clique em **OK**.
   - Se o texto for localizado, é realçado no programa.
   - Se o texto não for localizado, aparece uma mensagem de notificação.

Localizar e substituir texto

1. No menu **Acções**, seleccione **Localizar e Substituir**.



2. Escreva o texto que pretende localizar.
3. Escreva o texto de substituição.
4. Clique em **Substituir** para substituir a primeira ocorrência após a posição do cursor ou clique em **Substituir tudo** para substituir todas as ocorrências.

   Nota: Se o texto estiver num modelo matemático, uma mensagem aparece para o avisar que o texto de substituição substituirá o modelo completo—não apenas o texto encontrado.

Fechar o programa ou a função actual

▸ No menu **Acções**, seleccione **Fechar**.

Se a função ou o programa tiver alterações não guardadas, é-lhe pedido para verificar a sintaxe e guardar antes de fechar.

Executar programas e avaliar funções

Depois de definir e armazenar um programa ou função, pode usá-lo a partir de uma aplicação. Todas as aplicações podem avaliar funções, mas apenas as aplicações Calculadora e Notas podem executar programas.

As instruções de programação são executadas por ordem sequencial (embora alguns comandos possam alterar o fluxo do programa). O resultado, se algum, é apresentado na área de trabalho da aplicação.

- A execução do programa continua até atingir a última instrução ou um comando de **Stop**.

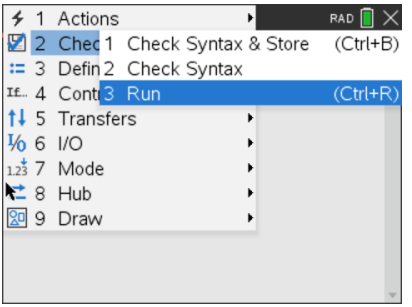
- A execução da função continua até atingir um comando de **Retroceder**.

Executar um programa ou uma função a partir do Editor de Programa

1. Assegure que tem um programa ou função definidos e que o Editor do Programa é o painel (computador) ativo ou página ativa (unidade portátil).

2. Na barra de ferramentas, clique no **botão** Ferramentas do documento e selecione **Verificar sintaxe e armazenar > Executar**
   —ou—
   Prima **Ctrl+R**.

   Unidade portátil: Prima menu 2 3, ou prima ctrl R.



   Irá automaticamente:

   - verificar a sintaxe e armazenar o programa ou função,
   - colar o nome do programa ou função na primeira linha disponível da aplicação Calculadora imediatamente, seguindo o Editor do Programa. Se não existir uma Calculadora nessa posição, é inserida uma nova.

3. Se a função ou o programa requerer que o utilizador forneça um ou mais argumentos, escreva os nomes dos valores ou das variáveis entre parêntesis.

4. Prima enter.

Nota: Também pode executar um programa ou função nas aplicações Calculadora ou Notas escrevendo o nome do programa entre parêntesis e quaisquer argumentos requeridos e premindo enter.

Utilizar nomes curtos e longos

Sempre que estiver no mesmo problema em que é definido um objeto, pode aceder ao mesmo introduzindo o seu nome curto (o nome dado no comando **Definir** do objeto). Este é o caso em todos os objetos definidos, incluindo objetos privados, públicos e que não sejam da biblioteca.

Pode aceder a um objeto da biblioteca a partir de qualquer documento escrevendo o nome longo do objeto. Um nome longo consiste no nome do documento da biblioteca do objeto seguido por um travessão "\" seguido do nome do objeto. Por exemplo, o nome longo do objeto definido como **func1** no documento da biblioteca **lib1** é **lib1\func1**. Para escrever o caratere "\" na unidade portátil, prima shift ÷.

Nota: Se não se conseguir lembrar do nome exacto ou da ordem dos argumentos requerida para um objecto da biblioteca privada, pode abrir o documento da biblioteca ou utilizar o Editor de programas para ver o objecto. Também pode usar **getVarInfo** para ver uma lista dos objetos de uma biblioteca.

Utilizar um programa ou uma função da biblioteca pública

1. Certifique-se de que definiu o objecto no primeiro problema do documento, guardou o objecto guardou o documento da biblioteca na pasta MyLib e atualizou as bibliotecas.
2. Abra a aplicação TINspire™ em que pretende utilizar a função ou o programa.

   Nota: Todas as aplicações podem avaliar funções, mas apenas as aplicações Calculadora e Notas podem executar programas.
3. Abra o Catálogo e utilize o separador da biblioteca para localizar e inserir o objecto.
   —ou—
   Escreva o nome do objeto. No caso de uma função ou de um programa, coloque sempre parêntesis a seguir ao nome.

   ```
   libs2\func1()
   ```
4. Se a função ou o programa requerer que o utilizador forneça um ou mais argumentos, escreva os nomes dos valores ou das variáveis entre parêntesis.

   ```
   libs\2func1(34,potência)
   ```
5. Prima enter.

Utilizar um programa ou uma função da biblioteca privada

Para utilizar um objecto da biblioteca privada, tem de saber o nome longo. Por exemplo, o nome longo do objecto definido como **func1** no documento da biblioteca **lib1** é **lib1\func1**.

Nota: Se não conseguir lembrar-se do nome exacto ou da ordem dos argumentos requerida para um objecto da biblioteca privada, pode abrir o documento da biblioteca ou utilizar o Editor de programas para ver o objecto.

1. Certifique-se de que definiu o objecto no primeiro problema do documento, guardou o objecto guardou o documento da biblioteca na pasta MyLib e atualizou as bibliotecas.
2. Abra a aplicação TINspire™ em que pretende utilizar a função ou o programa.

   Nota: Todas as aplicações podem avaliar funções, mas apenas as aplicações Calculadora e Notas podem executar programas.
3. Escreva o nome do objeto. No caso de uma função ou de um programa, coloque sempre parêntesis a seguir ao nome.

   ```
   libs2\func1()
   ```
4. Se a função ou o programa requerer que o utilizador forneça um ou mais argumentos, escreva os nomes dos valores ou das variáveis entre parêntesis.

   ```
   libs2\func1(34,potência)
   ```
5. Prima enter.

Interromper um programa ou função em execução

Enquanto uma função ou um programa estiver em execução, aparece o ponteiro de ocupado ⧖.

- Para interromper o programa ou a função,
  - Windows®: Manter pressionada a tecla **F12** e pressionar a tecla **Enter** repetidamente.
  - Mac®: Manter pressionada a tecla **F5** e pressionar a tecla **Enter** repetidamente.
  - Unidade portátil: Manter pressionada a tecla ⌂ on e pressionar enter repetidamente.

  Aparece uma mensagem. Para editar o programa ou a função no Editor do Programa, selecione **Ir Para**. O cursor aparece no comando onde ocorreu a interrupção.

Utilizar valores na execução de um programa

Pode seleccionar vários métodos para fornecer os valores que uma função ou um programa utiliza nos cálculos.

Integrar os valores no programa ou na função

Este método é utilizado principalmente para os valores que têm de ser iguais sempre que utilizar o programa ou a função.

1. Defina o programa.

```
* calculatearea                    4/4
Define calculatearea()=
Prgm
w:=3
h:=23.64
area:=w· h
Disp area
EndPrgm
```

2. Execute o programa.

```
calculatearea()
                                  70.92
                                  Done
```

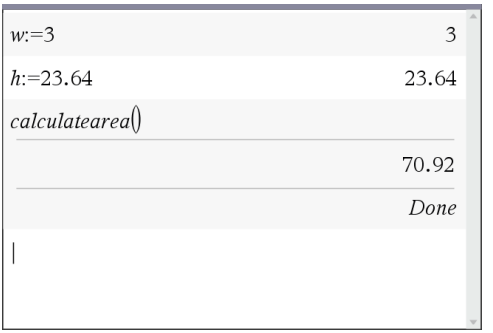
Deixar o utilizador atribuir os valores às variáveis

Um programa ou uma função pode fazer referência às variáveis criadas anteriormente. Este método requer que os utilizadores não se esqueçam dos nomes das variáveis e atribuam valores às variáveis antes de as utilizar.

1. Defina o programa.

```
* calculatearea                    2/2
Define calculatearea()=
Prgm
area:=w· h
Disp area
EndPrgm
```

2. Forneça as variáveis e, em seguida, execute o programa.

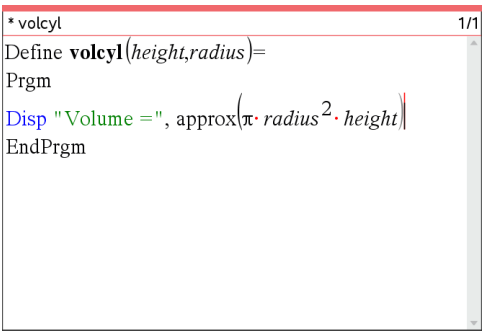


Deixar o utilizador fornecer os valores como argumentos

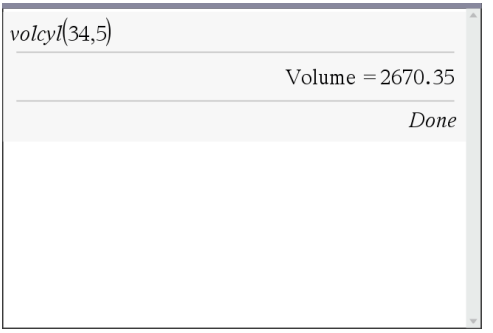
Este método permite passar um ou mais valores como argumentos na expressão que chama o programa ou a função.

O programa seguinte, **volcyl**, calcula o volume de um cilindro. Requer que o utilizador forneça dois valores: altura e raio da base do cilindro.

1. Defina o programa **volcyl**.



2. Execute o programa para ver o volume de um cilindro com uma altura de 34 mm e um raio da base de 5 mm.



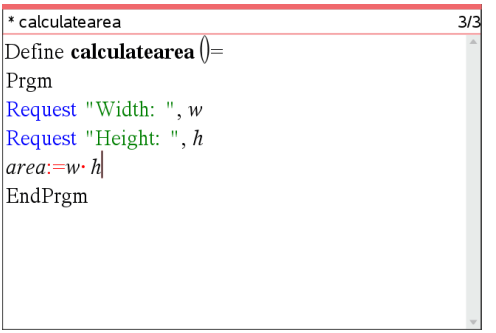
Nota: Não tem de utilizar os nomes dos parâmetros quando executar o programa **volcyl**, mas tem de fornecer dois argumentos (como valores, variáveis ou expressões). O primeiro tem de representar a altura e o segundo tem de representar o raio da base.

Pedir os valores do utilizador (apenas para programas)

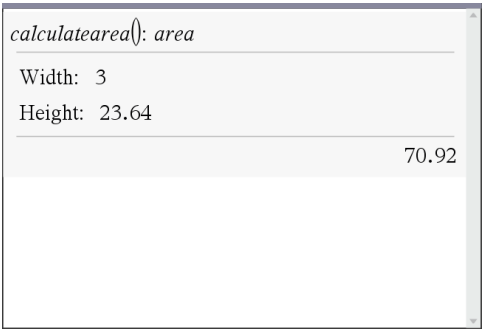
Pode utilizar os comandos **Request** e **RequestStr** num programa para fazer uma pausa e ver uma caixa de diálogo a pedir informações ao utilizador. Este método não requer os utilizadores se lembrem dos nomes das variáveis ou da ordem em que são necessárias.

Não pode utilizar o comando **Request** ou **RequestStr** numa função.

1. Defina o programa.



2. Execute o programa e responda aos pedidos.

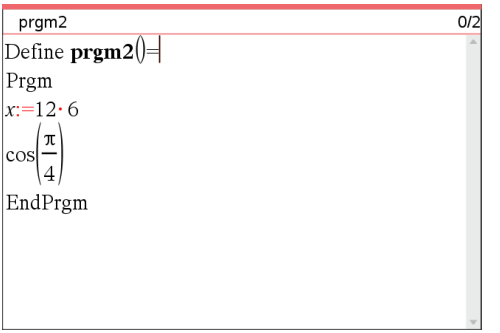


Utilize **RequestStr** em vez de **Request** quando quiser que o programa interprete a resposta do utilizador como uma cadeia de caracteres em vez de uma expressão matemática. Este procedimento evita que o utilizador tenha de colocar a resposta entre aspas (“”).

Ver informações

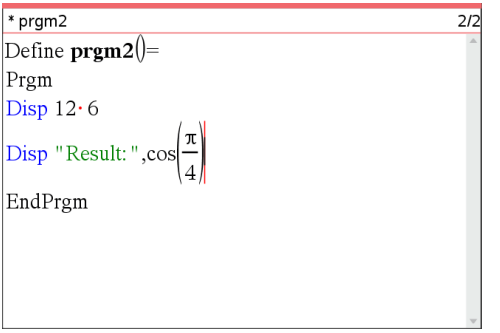
Um programa ou uma função em execução não mostra os cálculos intermédios excepto se incluir um comando para os ver. Esta é uma diferença importante entre efectuar um cálculo na linha de entrada e efectuá-lo numa função ou num programa.

Por exemplo, os cálculos seguintes não mostram um resultado numa função ou num programa (apesar de serem da linha de entrada).



Ver informações no histórico

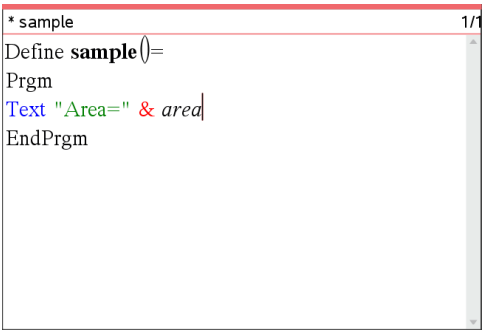
Pode utilizar o comando **Disp** num programa ou numa função para ver informações, incluindo cálculos intermédios, no histórico.



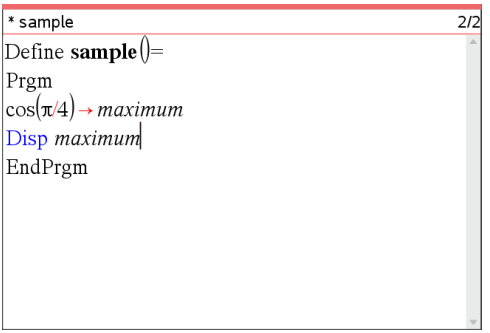
Ver informações numa caixa de diálogo

Pode utilizar o comando **Text** para fazer uma pausa num programa em execução e ver informações numa caixa de diálogo. O utilizador selecciona **OK** para continuar ou selecciona **Cancelar** para parar o programa.

Não pode utilizar o comando **Text** numa função.



Nota: A visualização de um resultado com **Disp** ou **Text** não guarda esse resultado. Se quiser fizer referência posteriormente a um resultado, guarde o resultado numa variável global.

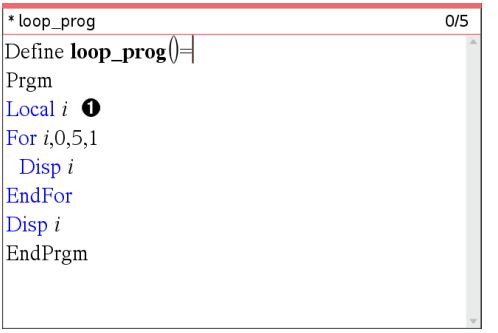


Utilizar variáveis locais

Uma variável local é uma variável temporária que existe apenas enquanto uma função definida pelo utilizador está a ser avaliada ou um programa definido pelo utilizador está a ser executado.

Exemplo de uma variável local

O segmento do programa seguinte apresenta um ciclo **For...EndFor** (que és descrito posteriormente neste módulo). A variável *i* é o contador de ciclos. Na maioria dos casos, a variável *i* só é utilizada enquanto o programa está a ser executado.



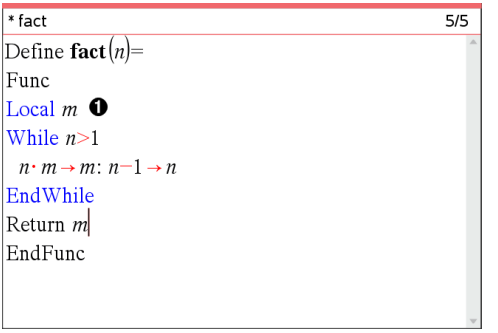
❶ Declara a variável *i* como local.

Nota: Quando possível, declare como local qualquer variável que seja utilizada apenas no programa e não necessite de estar disponível após a paragem do programa.

O que provoca uma mensagem de erro de variável indefinida?

Uma mensagem de erro de variável **Indefinida** aparece quando for verificada uma função definida pelo utilizador ou quando executar um programa definido pelo utilizador que faça referência a uma variável local que não esteja inicializada (atribuída a um valor).

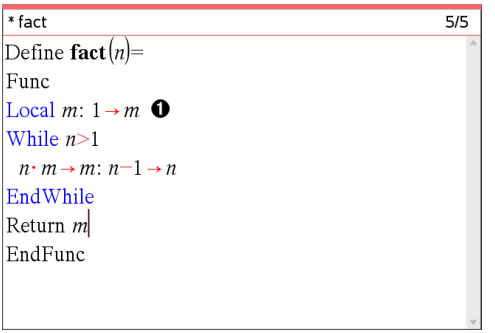
Exemplo:



❶ A variável local *m* não é atribuída a um valor inicial.

Iniciar as variáveis locais

Todas as variáveis locais têm de ser atribuídas a um valor inicial antes de serem referenciadas.



❶ 1 é guardado como o valor inicial para *m*.

Nota (CAS): As funções e os programas não podem utilizar uma variável local para efectuar cálculos simbólicos.

CAS: Efectuar cálculos simbólicos

Se quiser que uma função ou um programa efectue cálculos simbólicos, tem de utilizar uma variável global em vez de uma local. No entanto, tem de ter a certeza que a variável global já não existe fora do programa. Os métodos seguintes podem ajudar.

- Faça referência a um nome da variável global, geralmente com dois ou mais caracteres, que não existe provavelmente fora da função ou do programa.
- Inclua **DelVar** num programa para eliminar a variável global, se existir, antes de fazer referência a essa variável. (**DelVar** não elimina as variáveis bloqueadas nem ligadas.)

Diferenças entre funções e programas

Uma função definida no Editor de programas é semelhante às funções integradas no software TI-Nspire™.

- As funções têm de devolver um resultado, que pode ser representado graficamente ou introduzido numa tabela. Os programas não.
- Pode utilizar uma função (mas não um programa) inserida numa expressão. Por exemplo: **3 • func1(3)** é válido, mas não **3 • prog1(3)**.
- Apenas pode executar programas a partir das aplicações Calculadora e Notas. No entanto, pode avaliar funções nas aplicações Calculadora, Notas, Listas e Folha de Cálculo, Gráficos e Geometria, Dados e Estatística.
- Uma função pode referir-se a qualquer variável; no entanto, só pode guardar um valor numa variável local. Os programas podem guardar variáveis locais e globais.

Nota: Os argumentos utilizados para passar valores para uma função são tratados como variáveis locais automaticamente. Se quiser guardar quaisquer outras variáveis, tem de as declarar como **Local** dentro da função.

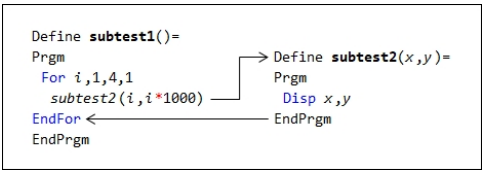
- Uma função não pode chamar um programa como uma subrotina, mas pode chamar outra função definida pelo utilizador.
- Não pode definir um programa numa função.
- Uma função não pode definir uma função global, mas pode definir uma função local.

Chamar um programa a partir de outro

Um programa pode chamar outro programa como uma subrotina. A subrotina pode ser externa (um programa independente) ou interna (incluída no programa principal). As subrotinas são úteis quando um programa necessitar de repetir o mesmo grupo de comandos em vários locais diferentes.

Chamar um programa independente

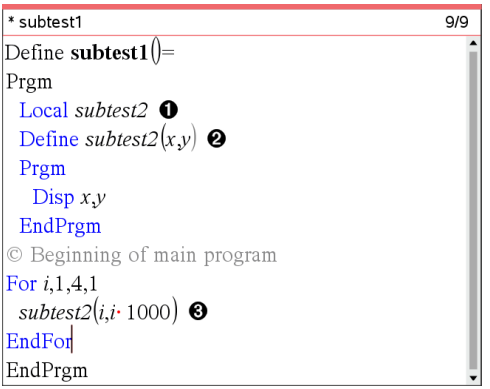
Para chamar um programa independente, utilize a mesma sintaxe que utiliza para executar o programa a partir da linha de entrada.



Definir e chamar uma subrotina interna

Para definir uma subrotina interna, utilize o comando **Define** com **Prgm...EndPrgm**. Como uma subrotina tem de ser definida antes de poder ser chamada, é uma boa prática definir as subrotinas no início do programa principal.

Uma subrotina interna é chamada e executada da mesma forma que um programa independente.



❶ Declara a subrotina como uma variável local.

❷ Define a subrotina.

❸ Chama a subrotina.

Nota: Utilize o menu **Var** do Editor de programas para introduzir os comandos **Define** e **Prgm...EndPrgm**.

Notas sobre a utilização das subrotinas

No fim de uma subrotina, a execução volta ao programa de chamada. Para sair de uma subrotina a qualquer momento, utilize **Return** sem argumento.

Uma subrotina não pode aceder às variáveis locais declaradas no programa de chamada. Do mesmo modo, o programa de chamada não pode aceder às variáveis locais declaradas numa subrotina.

Os comandos Lbl são locais nos programas em que estão localizados. Por exemplo, um comando **Goto** no programa de chamada não pode ramificar-se para uma etiqueta numa subrotina ou vice-versa.

Evitar erros de definições circulares

Quando avaliar uma função definida pelo utilizador ou executar um programa, pode especificar um argumento que inclua a mesma variável que utilizou para definir a função ou criar o programa. No entanto, para evitar erros de definições circulares, tem de atribuir um valor às variáveis utilizadas na avaliação da função ou na execução do programa. Por exemplo:

– —ou— –

❶ Resulta numa mensagem de erro de **Definição circular** se x ou i não tiver um valor. O erro não ocorre se já tiver atribuído um valor a x ou a i.

Controlar o fluxo de uma função ou de um programa

Quando executar um programa ou avaliar uma função, as linhas do programa são executadas por ordem sequencial. No entanto, alguns comandos alteram o fluxo do programa. Por exemplo:

- As estruturas de controlo, como, por exemplo, os comandos **If...EndIf**, utilizam um teste condicional para decidir que parte de um programa deve ser executada.
- Os comandos circulares, como, por exemplo, **For...EndFor** repetem um grupo de comandos.

Utilizar If, Lbl e Goto para controlar o fluxo do programa

O comando **If** e várias estruturas **If...EndIf** permitem executar uma instrução ou o um bloco de instruções condicionalmente, ou seja, baseado no resultado de um teste (como, por exemplo, x>5). **Os comandos Lbl** (etiqueta) e **Goto** permitem ramificar-se ou saltar de um local para outro numa função ou num programa.

O comando **If** e as várias estruturas **If...EndIf** residem no menu **Controlo** do Editor de programas.

Quando inserir uma estrutura, como, por exemplo, **If...Then...EndIf**, um modelo é inserido no local do cursor. O cursor é posicionado de forma a que possa introduzir um teste condicional.

Comando If

Para executar um comando individual quando um teste condicional é verdadeiro, utilize a forma geral:

```
If x>5
  Disp "x is greater than 5" ❶
Disp x ❷
```

❶ Executado apenas se x>5; caso contrário, ignorado.

❷ Mostra sempre o valor de x.

Neste exemplo, tem de guardar um valor para x antes de executar o comando **If**.

Estruturas If...Then...EndIf

Para executar um grupo de comandos se um teste condicional for verdadeiro, utilize a estrutura:

```
If x>5 Then
  Disp "x is greater than 5 " ❶
  2·x→x ❶
EndIf
Disp x ❷
```

❶ Executado apenas se x>5.

❷ Mostra o valor de:
2x se x>5
x se x≤5

Nota: EndIf marca o fim do bloco **Then** que é executado se a condição for verdadeira.

Estruturas If...Then...Else... EndIf

Para executar um grupo de comandos se um teste condicional for verdadeiro e um grupo diferente se a condição for falsa, utilize esta estrutura:

```
If x>5 Then
  Disp "x is greater than 5 " ❶
  2·x→x ❶
Else
  Disp "x is less than or equal to 5 " ❷
  5·x→x ❷
EndIf
Disp x ❸
```

❶ Executado apenas se x>5.

❷ Executado apenas se x≤5.

❸ Mostra o valor de:
2x se x>5
5x se x≤5

Estruturas If...Then...ElseIf... EndIf

Uma forma mais complexa do comando If permite testar várias condições. Suponha que quer um programa para testar um argumento fornecido pelo utilizador que significa uma de quatro opções.

Para testar cada opção (If Escolha=1, If Escolha=2, etc), utilize a estrutura **If...Then...ElseIf...EndIf**.

Comandos Lbl e Goto

Pode também controlar o fluxo com os comandos **Lbl** (etiqueta) e **Goto**. Estes comandos residem no menu **Transferências** do Editor de programas.

Utilize o comando **Lbl** para etiquetar (atribuir um nome a) um local específico na função ou no programa.

Lbl *NomeDaEtiqueta*	nome a atribuir a esta localização (utilize a mesma convenção de renomeação como o nome de uma variável)

Pode utilizar o comando **Goto** em qualquer ponto da função ou do programa para se ramificar para o local que corresponde à etiqueta especificada.

Goto *NomeDaEtiqueta*	especifica qual o comando **Lbl** a ramificar

Como um comando **Goto** é incondicional (ramifica-se sempre para a etiqueta especificada), é utilizado frequentemente com um comando **If** para que possa especificar um teste condicional. Por exemplo:

```
If x>5
  Goto GT5 ❶
Disp x

.............

Lbl GT5 ❷
Disp "The number was > 5"
```

❶ If x>5, ramifica-se directamente para a etiqueta GT5.

❷ Para este exemplo, o programa tem de incluir comandos (como, por exemplo, **Stop**) que impede a **Lbl** GT5 de ser executada se x≤5.

Utilizar ciclos para repetir um grupo de comandos

Para repetir o mesmo grupo de comandos sucessivamente, utilize uma das estruturas circulares. Estão disponíveis vários tipos de ciclos. Cada tipo fornece uma forma diferebte para sair do ciclo, baseado num teste condicional.

Os comandos do ciclo ou relacionados com ciclos residem nos menus **Controlo** e **Transferências** do Editor de programas.

Quando inserir uma das estruturas circulares, o modelo é inserido no local do cursor. Pode iniciar a introdução dos comandos que serão executados no ciclo.

Ciclos For...EndFor

Um ciclo **For...EndFor** utiliza um contador para controlar o número de vezes que o ciclo é repetido. A sintaxe do comando **For** é:

Nota: O valor final pode ser inferior ao valor inicial, desde que o incremento seja negativo.

For *variável*, *início*, *fim* [, *incremento*]

❶ ❷ ❸ ❹

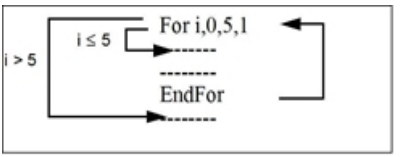
❶ *Variável* utilizada como um contador

❷ Valor do contador utilizado pela primeira vez **For** é executado

❸ Sai do ciclo quando a *variável* excede este valor

❹ Adicionado ao contador nas vezes subsequentes **For** é executado (Se este valor opcional for omitido, o *incremento* é de 1.)

Quando **For** for executado, o valor da *variável* é comparado ao valor *final*. Se a *variável* não exceder *end*, o ciclo é executado; caso contrário, o controlo salta para o comando seguinte **EndFor**.



Nota: O comando **For** incrementa automaticamente a variável do contador para que a função ou o programa possa sair do ciclo após um determinado número de repetições.

No fim do ciclo (**EndFor**), o comando volta novamente ao comando **For**, em que a variável é incrementada e comparada com *end*.

Por exemplo:

```
For i,0,5,1
  Disp i ❶
EndFor
Disp i ❷
```

❶ Mostra 0, 1, 2, 3, 4 e 5.

❷ Mostra 6. Quando a *variável* incrementa para 6, o ciclo não é executado.

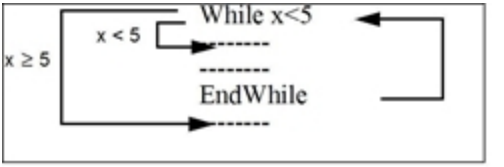
Nota: Pode declarar a variável do contador como local se não necessitar de ser guardada após a paragem do programa ou da função.

Ciclos While...EndWhile

Um ciclo **While...EndWhile** repete um bloco de comandos desde que uma condição especificada seja verdadeira. A sintaxe do comando **While** é:

Condição While

Quando executar **While**, a *condição* é avaliada. Se a *condição* for verdadeira, o ciclo é executado; caso contrário, o controlo salta para o comando seguinte **EndWhile**.



Nota: O comando **While** não muda automaticamente a condição. Tem de incluir comandos que permitam à função ou ao programa sair do ciclo.

No fim do ciclo (**EndWhile**), o controlo volta novamente para o comando **While**, em que a condição é reavaliada.

Para executar o ciclo pela primeira vez, a condição tem de começar por ser verdadeira.

- Quaisquer variáveis referenciadas na condição têm de ser definidas antes do comando **While**. (Pode construir os valores para a função ou o programa, ou pode pedir ao utilizador para introduzir os valores.)
- O ciclo tem de conter os comandos que alteram os valores na condição, eventualmente fazendo com que seja falsa. Caso contrário, a condição é sempre verdadeira e a função ou o programa não podem sair do ciclo (chamado um ciclo infinito).

Por exemplo:

```
0→x ❶
While x<5
  Disp x ❷
  x+1→x ❸
EndWhile
Disp x ❹
```

❶ Define x inicialmente.

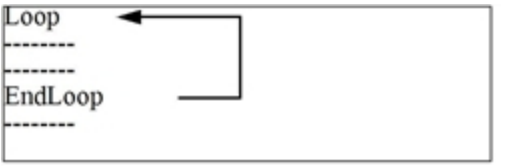
❷ Mostra 0, 1, 2, 3 e 4.

❸ Incrementa x.

❹ Mostra 5. Quando x incrementa para 5, o ciclo não é executado.

Ciclos Loop...EndLoop

Um **Loop...EndLoop** cria um ciclo infinito, que é repetido indefinidamente. O comando **Loop** não tem qualquer argumento.



En geral, introduzem-se no ciclo comandos que lhe permitem sair do ciclo. Os comandos mais utilizados são: **If**, **Exit**, **Goto** e **Lbl** (etiqueta). Por exemplo:

```
0→x
Loop
  Disp x
  x+1→x
  If x>5 ❶
    Exit
EndLoop
Disp x ❷
```

❶ Um comando **If** verifica a condição.

❷ Sai do ciclo e salta para aqui quando x incrementa para 6.

Nota: O comando **Exit** sai do ciclo actual.

Neste exemplo, o comando **If** pode estar em qualquer parte do ciclo.

Quando o comando If está:	**O ciclo é:**
No início do ciclo	Executado apenas se a condição for verdadeira.
No fim do ciclo	Executado pelo menos uma vez e repetido apenas se a condição for verdadeira.

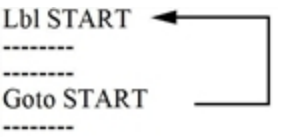
O comando **If** pode também utilizar um comando **Goto** para transferir o controlo do programa para um comando **Lbl** (etiqueta) especificado.

Repetir um ciclo imediatamente

O comando **Cycle** transfere imediatamente o controlo do programa para a iteração seguinte de um ciclo (antes de a iteração actual estar completa). Este comando funciona com **For...EndFor**, **While...EndWhile** e **Loop...EndLoop**.

Ciclos Lbl e Goto

Apesar dos comandos **Lbl** (etiqueta) e **Goto** não serem especificamente comandos circulares, podem ser utilizados para criar um ciclo infinito. Por exemplo:



Tal como **Loop...EndLoop**, o ciclo deve conter comandos que permitem à função ou ao programa sair do ciclo.

Alterar as definições de modos

As funções e os programas podem utilizar a função **setMode()** para definir temporariamente os modos de resultados ou de cálculo específicos. O menu **Mode** do Editor de programas torna mais fácil introduzir a sintaxe correcta sem necessitar que memorize códigos numéricos.

Nota: As alterações de modo efectuadas na definição de uma função ou de um programa não persistem fora da função ou do programa.

Definir um modo

1. Posicione o cursor onde pretende inserir a função **setMode**.
2. No menu **Modo**, seleccione o modo que pretende alterar e seleccione a definição nova.

   A sintaxe correcta é inserida na localização do cursor. Por exemplo:

```
setMode(1,3)
```

Depurar programas e processar erros

Depois de escrever uma função ou um programa, pode utilizar várias técnicas para localizar e corrigir os erros. Pode também construir um comando de processamento de erros na função ou no programa.

Se a função ou o programa permitir seleccionar várias opções, certifique-se de que executa e testa cada opção.

Técnicas de depuração

As mensagens relacionadas com erros de tempo de execução podem localizar erros, mas não erros na lógica do programa. As técnicas seguintes podem ser úteis.

- Introduza temporariamente os comandos **Disp** para ver os valores das variáveis críticas.
- Para verificar se um ciclo é executado o número correcto de vezes, utilize **Disp** para ver a variável do contador ou os valores no teste condicional.

- Para verificar se uma subrotina é executada, utilize **Disp** para ver mensagens, como, por exemplo, “Entrar na subrotina” e “Sair da subrotina” no início e no fim da subrotina.
- Para parar um programa ou função manualmente,
  - Windows®: Manter pressionada a tecla **F12** e pressionar **Enter** repetidamente.
  - Macintosh®: Manter pressionada a tecla **F5** e pressionar **Enter** repetidamente.
  - Unidade portátil: Manter pressionada a tecla [on] e pressionar [enter] repetidamente.

Comandos de processamentos de erros

Comando	Descrição
Try...EndTry	Define um bloco que permite a uma função ou a um programa executar um comando e, se for necessário, recuperar de um erro gerado por esse comando.
ClrErr	Apaga o estado de erro e define a variável do sistema *errCode* como zero. Para um exemplo de utilização de *errCode*, ver o comando **Try** no *Manual de referência*.
PassErr	Passa um erro para o nível seguinte do bloco **Try...EndTry**.

Informações gerais

Ajuda online

education.ti.com/eguide

Selecione o seu país para obter mais informação sobre o produto.

Contacte a assistência técnica da TI

education.ti.com/ti-cares

Selecione o seu país para obter recursos técnicos ou assistência.

Informações da Assistência e Garantia

education.ti.com/warranty

Selecione o seu país para obter informações sobre a duração e os termos da garantia ou sobre a assistência ao produto.

Garantia Limitada. Esta garantia não afeta os seus direitos legais.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243