Introduction à TI-Nspire™ Programming Editor

Pour en savoir plus sur la technologie TI, consultez l'aide en ligne à l'adresse education.ti.com/eguide.

Informations importantes

Sauf indication expresse contraire dans la Licence qui accompagne un programme, Texas Instruments n'offre aucune garantie, expresse ou implicite, y compris, mais sans s'y limiter, les garanties implicites de qualité marchande et de conditionnement physique à des fins particulières, concernant tout programme ou matériel de livre les matériaux disponibles uniquement sous la forme "tel quel". En aucun cas, Texas Instruments ne sera responsable de tout dommage spécial, collatéral, accessoire ou consécutif lié à l'achat ou à l'utilisation de ces matériaux, ou à la responsabilité exclusive de Texas Instruments, quelle que soit la forme de action, ne doit pas dépasser le montant indiqué dans la licence pour le programme. En outre, Texas Instruments ne sera pas responsable de toute réclamation de quelque nature que ce soit contre l'utilisation de ces documents par toute autre partie.

Marques et droits d'auteur

Le logiciel TI-Nspire ™ utilise Lua comme environnement de script. Pour plus d'informations sur les droits d'auteur et les licences, consultez http://www.lua.org/license.html.

Le logiciel TI-Nspire ™ utilise Chipmunk Physics version 5.3.4 comme environnement de simulation. Pour obtenir des informations sur les licences, consultez http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/.

Microsoft® et Windows® sont des marques déposées de Microsoft Corporation aux États-Unis et / ou dans d'autres pays.

Mac OS®, iPad® et OS X® sont des marques déposées d'Apple Inc.

Unicode® est une marque déposée de Unicode, Inc. aux États-Unis et dans d'autres pays.

La marque verbale Bluetooth® et les logos sont des marques déposées appartenant à Bluetooth SIG, Inc.

Table des matières

Premiers contacts avec l'Éditeur de programmes

Vous pouvez créer des fonctions ou des programmes définis par l'utilisateur en saisissant les instructions de définition dans la ligne de saisie de l'application Calculs ou en faisant appel à l'Éditeur de programmes. L'Éditeur de programmes offre plusieurs avantages, qui sont abordés dans cette section. Pour plus d'informations, reportez-vous à la section consacrée à l'application *Calculs*.

- Il intègre des modèles de programmation et des boîtes de dialogue qui vous aident à définir des fonctions et des programmes en utilisant la syntaxe appropriée.
- Il vous permet de saisir des instructions de programmation réparties sur plusieurs lignes sans avoir à utiliser de séquence de touches particulière pour ajouter chacune des lignes.
- Vous pouvez créer aisément des objets de bibliothèque privée et publique (variables, fonctions et programmes). Pour plus d'informations, reportez-vous à la section *Bibliothèques*.

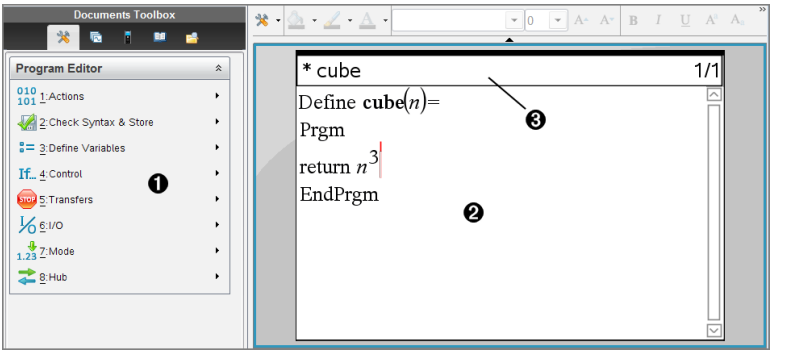
Lancement de l'Éditeur de programmes

▶ Pour ajouter une nouvelle page Éditeur de programmes dans l'activité courante :

Dans la barre d'outils, cliquez sur **Insérer > Éditeur de programmes > Nouveau**.

Unité : Appuyez sur doc et sélectionnez **Insérer > Éditeur de programmes > Nouveau**.

Remarque : L'éditeur est également accessible à partir du menu **Fonctions & Programmes** d'une page de Calculs.



❶ Menu de l'Éditeur de programmes : ce menu est accessible à tout moment dans l'espace de travail Éditeur de programmes via le mode d'affichage normal.

❷ Espace de travail Éditeur de programmes

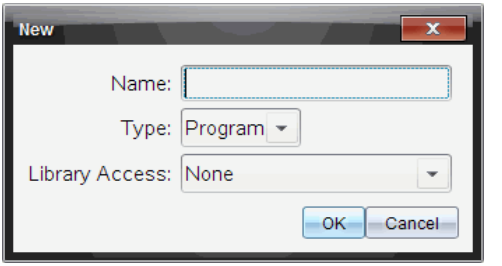
❸ La barre d'état affiche des informations sur le numéro de ligne et le nom de la fonction ou du programme en cours d'édition. Un astérisque (*) indique que la fonction a été modifiée depuis la dernière vérification de sa syntaxe et qu'elle a été

stockée.

Définition d'un programme ou d'une fonction

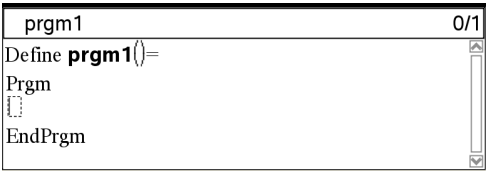
Démarrage de l'Éditeur de programmes

1. Pour ouvrir l'éditeur à partir d'une page Calculs,
   - Dans le menu **Insertion**, sélectionnez **Éditeur de programmes**, puis **Nouveau**.
2. Pour ouvrir l'éditeur, si vous n'êtes pas dans une page Calculs,
   - Dans le menu **Insertion**, sélectionnez **Éditeur de programmes**, puis **Nouveau**.



3. Entrez le nom de la fonction ou du programme que vous souhaitez définir.
4. Sélectionnez le **Type** (**Programme (Program)** ou **Fonction (Function)**).
5. Définissez l'**Accès à la bibliothèque (Library Access)** :
   - Si vous ne souhaitez utiliser la fonction ou le programme qu'à partir du classeur et de l'activité courante, sélectionnez **Aucun (None)**.
   - Pour pouvoir accéder à la fonction ou au programme depuis n'importe quel classeur, sans l'afficher dans le Catalogue (Catalog), sélectionnez **LibPriv**.
   - Si vous souhaitez pouvoir accéder à la fonction ou au programme depuis n'importe quel classeur et l'afficher dans le Catalogue (Catalog), sélectionnez **LibPub (Afficher dans le catalogue) (LibPub Show in Catalog)**. Pour plus d'informations à ce sujet, consultez le chapitre consacré aux bibliothèques.
6. Cliquez sur **OK**.

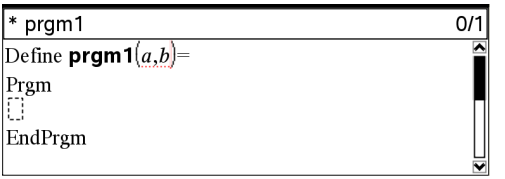
   Une nouvelle session de l'Éditeur de programmes s'ouvre, avec le modèle correspondant aux sélections effectuées.



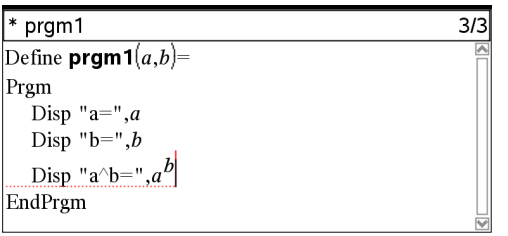
Saisie de lignes dans une fonction ou un programme

L'Éditeur de programmes n'est pas conçu pour exécuter ou évaluer les expressions au moment de leur saisie. Leur exécution intervient uniquement lorsque vous évaluez la fonction ou exécutez le programme.

1. Si votre fonction ou programme exige la saisie d'arguments par l'utilisateur, spécifiez les noms de paramètres entre les parenthèses qui suivent son nom. Séparez les paramètres par des virgules.



2. Entre les lignes Func et EndFunc (ou Prgm et EndPrgm), saisissez les lignes d'instructions qui constituent votre fonction ou programme.



- Vous pouvez saisir le nom des fonctions et des commandes ou les insérer directement à partir du Catalogue (Catalog).
- La longueur d'une ligne peut excéder la largeur de l'écran, mais dans ce cas, vous devez faire défiler son contenu pour afficher l'instruction complète.
- À la fin de chaque ligne, appuyez sur enter pour insérer une nouvelle ligne et ainsi poursuivre votre saisie.
- Utilisez les touches fléchées ◀, ▶, ▲ et ▼ pour faire défiler les lignes de la fonction ou du programme afin de saisir ou de modifier des commandes.

Insertion de commentaires

Un symbole de commentaire (©) vous permet d'insérer une remarque. Les commentaires peuvent s'avérer utiles pour toutes les personnes voulant lire ou modifier le contenu du programme. Les commentaires ne s'affichent pas lors de l'exécution du programme et ils n'en n'affectent en aucune façon le déroulement.

```
Define LibPub volcyl(ht,r) =
Prgm
©volcyl(ht,r) => volume du cylindre ❶
  Disp "Volume =", approx(π • r^2 • ht)
```

```
©Ceci est un autre commentaire.
EndPrgm
```

❶ Commentaire indiquant la syntaxe à utiliser. S'agissant d'un objet de bibliothèque public et ce commentaire correspondant à la première ligne d'un bloc Func ou Prgm, il s'affiche dans le Catalogue (Catalog) en guise d'aide. Pour plus d'informations à ce sujet, consultez le chapitre consacré aux bibliothèques.

Pour insérer un commentaire :

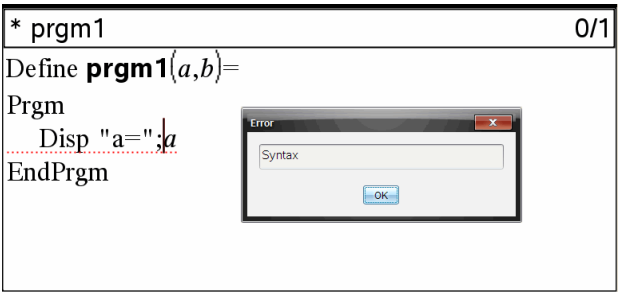
1. Placez le curseur à l'extrémité de la ligne où vous souhaitez insérer un commentaire.
2. Dans le menu **Actions**, sélectionnez **Insérer un commentaire**.
3. Tapez le texte du commentaire à la suite du symbole ©.

Vérification de la syntaxe

L'Éditeur de programmes vous permet de vérifier la syntaxe de vos fonctions et programmes.

- Dans le menu **Vérifier la syntaxe et enregistrer**, sélectionnez **Vérifier la syntaxe**.

  Si des erreurs de syntaxe sont détectées, un message d'erreur s'affiche et le curseur est placé, dans la mesure du possible, au niveau de la première erreur pour vous permettre de la corriger.



Stockage d'une fonction ou d'un programme

Vous devez stocker votre fonction ou programme pour la/le rendre accessible. L'Éditeur de programmes vérifie automatiquement la syntaxe de l'objet avant de procéder à son stockage.

Un astérisque (*) s'affiche dans l'angle supérieur gauche de l'Éditeur de programmes pour indiquer que la fonction ou le programme n'a pas encore été stocké.

- Dans le menu **Vérifier la syntaxe et enregistrer**, sélectionnez **Vérifier la syntaxe et enregistrer**.

  Si des erreurs de syntaxe sont détectées, un message d'erreur s'affiche et le curseur est placé, dans la mesure du possible, au niveau de la première erreur.

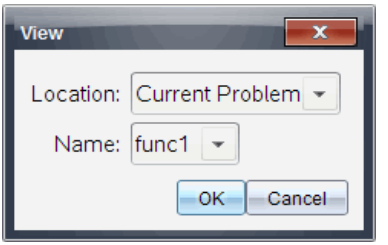
En l'absence d'erreur de syntaxe, le message “Enregistrement effectué“ s'affiche dans la ligne d'état qui se trouve dans la partie supérieure de la fenêtre de l'Éditeur de programmes.

Remarque : si la fonction ou le programme a été défini comme objet de bibliothèque, vous devez également enregistrer le classeur dans le dossier de bibliothèque spécifié et rafraîchir les bibliothèques pour rendre l'objet accessible aux autres classeurs. Pour plus d'informations à ce sujet, consultez le chapitre consacré aux bibliothèques.

Affichage d'un programme ou d'une fonction

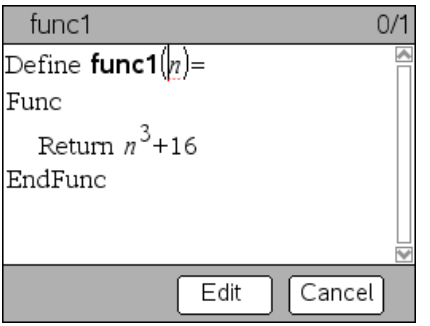
1. Dans le menu **Actions**, sélectionnez **Afficher**.

   La boîte de dialogue Affichage (View) apparaît.



2. Si la fonction ou le programme est un objet de bibliothèque, sélectionnez celle-ci dans la liste **Emplacement (Location)**.
3. Sélectionnez le nom de la fonction ou du programme dans la liste **Nom (Name)**.

   La fonction ou le programme sélectionné s'affiche dans une visionneuse.



4. Utilisez les touches fléchées pour parcourir la fonction ou le programme.
5. Pour modifier le programme, cliquez sur **Éditer**.

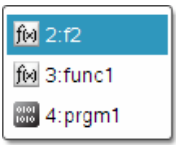
   Remarque : l'option **Modifier (Edit)** est uniquement disponible pour les fonctions et les programmes définis dans l'activité courante. Pour modifier un objet de bibliothèque, vous devez préalablement ouvrir le classeur de bibliothèque associé.

Ouverture d'une fonction ou d'un programme à des fins d'édition

L'ouverture d'une fonction ou d'un programme n'est possible qu'à partir de l'activité courante.

Remarque : vous ne pouvez pas modifier une fonction ou un programme verrouillé. Pour déverrouiller l'objet, affichez une page Calculs et utilisez la commande **unLock**.

1. Affichez la liste des fonctions et programmes disponibles.
   - Dans le menu **Actions**, sélectionnez **Ouvrir**.

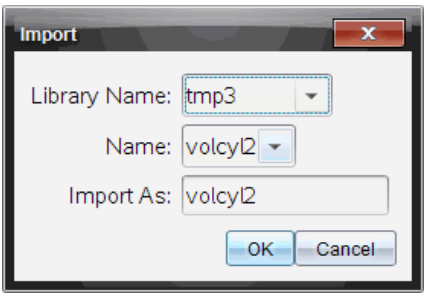


2. Sélectionnez l'élément à ouvrir.

Importation d'un programme à partir d'une bibliothèque

Vous pouvez importer dans l'activité courante une fonction ou un programme défini comme objet de bibliothèque dans l'Éditeur de programmes. La copie importée n'est pas verrouillée, même si l'élément d'origine l'est.

1. Dans le menu **Actions**, sélectionnez **Importer**.



2. Sélectionnez le **nom de la bibliothèque (Library Name)**.
3. Sélectionnez le **nom (Name)** de l'objet.
4. Pour importer l'objet sous un autre nom, saisissez celui-ci dans le champ **Importer en tant que (Import As)**.

Création d'une copie d'une fonction ou d'un programme

Lors de la création d'une fonction ou d'un programme, il peut être plus simple de démarrer en utilisant la copie d'une fonction ou d'un programme existant. La copie créée n'est pas verrouillée, même si l'élément d'origine l'est.

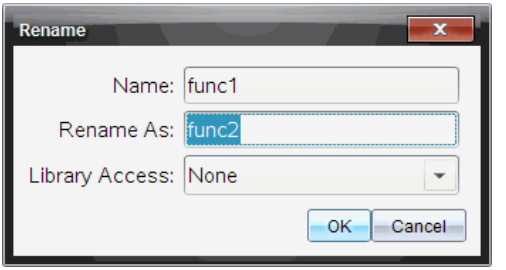
1. Dans le menu **Actions**, sélectionnez **Créer une copie**.
2. Saisissez un nouveau nom ou cliquez sur **OK** pour accepter le nom suggéré.

3. Si vous souhaitez modifier le niveau d'accès, sélectionnez **Accès à la bibliothèque (Library Access)**, puis choisissez le nouveau niveau d'accès.

Changement de nom d'un programme ou d'une fonction

Vous pouvez renommer la fonction ou le programme courant et (facultativement) en modifier le niveau d'accès.

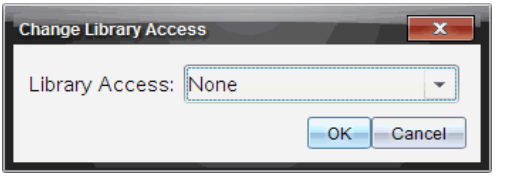
1. Dans le menu **Actions**, sélectionnez **Renommer**.



2. Saisissez un nouveau nom ou cliquez sur **OK** pour accepter le nom suggéré.
3. Si vous souhaitez modifier le niveau d'accès, sélectionnez **Accès à la bibliothèque (Library Access)**, puis choisissez le nouveau niveau d'accès.

Modification du niveau d'accès à la bibliothèque

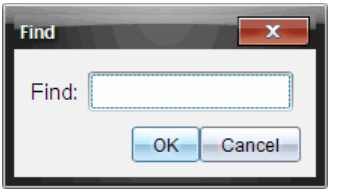
1. Dans le menu **Actions**, sélectionnez **Changer l'accès à la bibliothèque**.



2. Sélectionnez le niveau d'**Accès à la bibliothèque (Library Access)** :
   - Si vous ne souhaitez utiliser la fonction ou le programme qu'à partir de l'activité courante, sélectionnez **Aucun (None)**.
   - Pour pouvoir accéder à la fonction ou au programme depuis n'importe quel classeur, sans l'afficher dans le Catalogue (Catalog), sélectionnez **LibPriv**.
   - Si vous souhaitez pouvoir accéder à la fonction ou au programme depuis n'importe quel classeur et l'afficher dans le Catalogue (Catalog), sélectionnez **LibPub**.

Recherche de texte

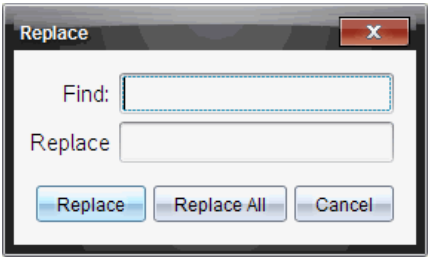
1. Dans le menu **Actions**, sélectionnez **Rechercher**.



2. Saisissez le texte à rechercher, puis cliquez sur **OK**.
   - Si une occurrence du texte est trouvée, elle est mise en surbrillance dans le programme.
   - Si aucune occurrence du texte n'est trouvée, un message de notification vous en informe.

Recherche et remplacement de texte

1. Dans le menu **Actions**, sélectionnez **Rechercher et remplacer**.



2. Saisissez le texte à rechercher.
3. Entrez le texte de remplacement.
4. Cliquez sur **Remplacer** pour remplacer la première occurrence du texte après le curseur ou cliquez sur **Tout remplacer** pour remplacer toutes les occurrences trouvées.

   Remarque : si une occurrence du texte est trouvée dans un modèle mathématique, un message s'affiche pour vous indiquer que le texte de remplacement sera utilisé à la place de l'intégralité du modèle et pas simplement du texte trouvé.

Fermeture de la fonction ou du programme courant

- Dans le menu **Actions**, sélectionnez **Fermer**.

  Si des modifications ont été apportées à la fonction ou au programme et n'ont pas été enregistrées, vous êtes invité à vérifier la syntaxe de l'objet et à l'enregistrer avant de le fermer.

Exécution de programmes et évaluation de fonctions

Après avoir défini et enregistré une fonction ou un programme, vous pouvez l'utiliser dans une application. Toutes les applications permettent d'évaluer des fonctions, mais seules les applications Calculs et Éditeur mathématique sont capables d'exécuter des programmes.

Les instructions de programme sont exécutées selon un ordre séquentiel (même si certaines commandes modifient le déroulement de l'exécution du programme). Le résultat, le cas échéant, est affiché dans l'espace de travail de l'application.

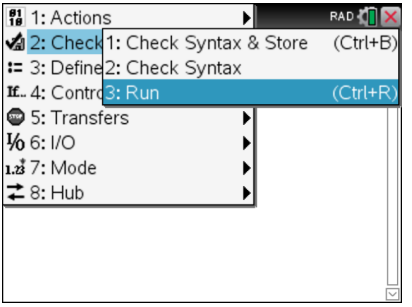
- L'exécution d'un programme se poursuit jusqu'à ce que la dernière instruction ou qu'une commande **Stop** soit atteinte.
- L'exécution d'une fonction se poursuit jusqu'à ce qu'une commande **Return** soit atteinte.

Exécution d'un programme ou d'une fonction dans l'éditeur de programmes

1. Assurez-vous d'avoir défini un programme ou une fonction, et que l'éditeur de programmes est le panneau actif (ordinateur) ou la page active (unité nomade).

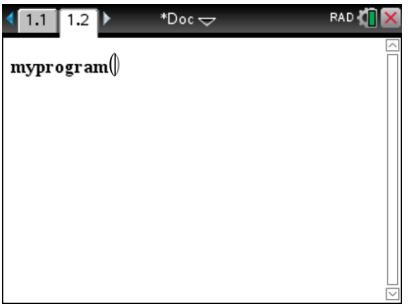
2. Dans la barre d'outils, cliquez sur le bouton **Document Tools** et sélectionnez **Check Syntax & Store > Run**.
   -ou-
   Appuyez sur **Ctrl+R**.

   Unité nomade : Appuyez sur menu 2 3, ou ctrl R.



   Ceci automatiquement :

   - vérifiera la syntaxe et enregistrera le programme ou la fonction ;
   - collera le nom du programme ou de la fonction dans la première ligne disponible de l'application Calculs, suivant immédiatement l'Éditeur de programmes. Si aucune application Calculs n'existe à cet endroit, une nouvelle sera insérée.



3. Si votre fonction ou programme exige la saisie d'un ou plusieurs arguments, spécifiez les valeurs ou les noms de variables dans les parenthèses qui suivent son nom.

4. Appuyez sur enter.

Remarque : Vous pouvez également exécuter un programme ou une fonction dans les applications Calculs ou Éditeur mathématique en saisissant le nom du programme avec des parenthèses et tout argument requis et en appuyant sur enter.

Utilisation des noms abrégés et des noms complets

Chaque fois que vous êtes dans une même activité dans laquelle un objet est défini, vous pouvez y accéder en saisissant son nom abrégé (le nom donné à l'objet dans la commande **Define**). Ceci est le cas pour tout objet défini, incluant les objets privés, publics, et n'appartenant pas à une bibliothèque.

Vous pouvez accéder à un objet de bibliothèque depuis tout classeur en saisissant le nom complet de l'objet. Un nom complet comprend le nom du classeur de bibliothèque de l'objet suivi d'une barre oblique inversée « \ », suivi du nom de l'objet. Par exemple, le nom de l'objet défini comme **func1** dans le classeur de bibliothèque **lib1** est **lib1\func1**. Pour entrer le caractère « \ » dans l'unité nomade, appuyez sur shift ÷.

Remarque : Si vous avez oublié le nom exact ou l'ordre des arguments requis pour un objet de bibliothèque privée, vous pouvez ouvrir le classeur de bibliothèque ou utiliser l'Éditeur de programmes pour afficher l'objet en question. Vous pouvez utiliser **getVarInfo** pour afficher une liste des objets dans une bibliothèque.

Utilisation d'une fonction ou d'un programme de bibliothèque publique

1. Assurez-vous d'avoir défini l'objet dans la première activité du classeur, de l'avoir enregistré, d'avoir enregistré le classeur de bibliothèque dans le dossier MyLib et d'avoir rafraîchi les bibliothèques.

2. Ouvrez l'application TI-Nspire™ dans laquelle vous souhaitez utiliser la fonction ou le programme.

   Remarque : Toutes les applications permettent d'évaluer des fonctions, mais seules les applications Calculs et Éditeur mathématique sont capables d'exécuter des programmes.

3. Ouvrez le Catalogue (Catalog) et utilisez l'onglet des bibliothèques pour rechercher et insérer l'objet voulu
   -ou-
   Saisissez le nom de l'objet. Dans le cas d'une fonction ou d'un programme, faites toujours suivre le nom de parenthèses.

   ```
   libs2\func1()
   ```

4. Si votre fonction ou programme exige la saisie d'un ou plusieurs arguments, spécifiez les valeurs ou les noms de variables dans les parenthèses qui suivent son nom.

   ```
   libs2\func1(34,power)
   ```

5. Appuyez sur enter.

Utilisation d'une fonction ou d'un programme de bibliothèque privée

Pour utiliser un objet de bibliothèque privée, vous devez connaître son nom complet. Par exemple, le nom complet de l'objet défini comme **func1** dans le classeur de bibliothèque **lib1** est **lib1\func1**.

Remarque : Si vous avez oublié le nom exact ou l'ordre des arguments requis pour un objet de bibliothèque privée, vous pouvez ouvrir le classeur de bibliothèque ou utiliser l'Éditeur de programmes pour afficher l'objet en question

1. Assurez-vous d'avoir défini l'objet dans la première activité du classeur, de l'avoir enregistré, d'avoir enregistré le classeur de bibliothèque dans le dossier MyLib et d'avoir rafraîchi les bibliothèques.
2. Ouvrez l'application TI-Nspire™ dans laquelle vous souhaitez utiliser la fonction ou le programme.

   Remarque : Toutes les applications permettent d'évaluer des fonctions, mais seules les applications Calculs et Éditeur mathématique sont capables d'exécuter des programmes.

3. Saisissez le nom de l'objet. Dans le cas d'une fonction ou d'un programme, faites toujours suivre le nom de parenthèses.

   ```
   libs2\func1()
   ```

4. Si votre fonction ou programme exige la saisie d'un ou plusieurs arguments, spécifiez les valeurs ou les noms de variables dans les parenthèses qui suivent son nom.

   ```
   libs2\func1(34,power)
   ```

5. Appuyez sur enter.

Interruption de l'exécution d'un programme ou d'une fonction

Lors de l'exécution d'une fonction ou d'un programme, le pointeur en forme d'horloge ⏲ s'affiche.

- Pour arrêter la fonction ou le programme,
  - Windows® : Maintenez enfoncée la touche **F12** et appuyez sur **Entrée** plusieurs fois.
  - Sur Mac® : Maintenez enfoncée la touche **F5** et appuyez sur **Entrée** plusieurs fois.
  - Unité nomade : Maintenez enfoncée la touche [on] et appuyez sur [enter] plusieurs fois.

  Un message s'affiche. Pour modifier le programme ou la fonction dans l'Éditeur de programmes, sélectionnez **Go To**. Le curseur apparaît là où l'interruption a eu lieu.

Saisie de valeurs dans un programme

Vous pouvez utiliser plusieurs méthodes pour saisir les valeurs qu'utilise une fonction ou un programme pour les calculs.

Fixation des valeurs dans le programme ou la fonction

Cette méthode s'utilise principalement avec les valeurs qui doivent rester les mêmes chaque fois que le programme ou la fonction est utilisé.

1. Définissez le programme.

```
Define calculatearea()=
Prgm
w:=3
h:=23.64
area:=w*h
EndPrgm
```

2. Exécutez le programme.

```
calculatearea() :area                    70.92
```

Affectation des valeurs aux variables par l'utilisateur

Un programme ou une fonction peut faire référence à des variables créées au préalable. Cette méthode nécessite que les utilisateurs se rappellent du nom des variables et affectent des valeurs à celles-ci avant d'utiliser l'objet.

1. Définissez le programme.

```
Define calculatearea()=
Prgm
area:=w*h
EndPrgm
```

2. Spécifiez les variables, puis exécutez le programme.

```
w:=3 : h:=23.64
calculatearea() :area                    70.92
```

Saisie des valeurs sous la forme d'arguments par l'utilisateur

Cette méthode permet aux utilisateurs de saisir une ou plusieurs valeurs sous la forme d'arguments dans l'expression qui appelle le programme ou la fonction.

Le programme suivant, **volcyl**, permet de calculer le volume d'un cylindre. Avec cette méthode, l'utilisateur doit spécifier deux valeurs : la hauteur et le rayon du cylindre.

1. Définissez le programme **volcyl.**

```
Definevolcyl(hauteur,rayon) =
Prgm
Disp "Volume =", approx(π • rayon^2 • hauteur)
EndPrgm
```

2. Exécutez ce programme pour afficher le volume d'un cylindre ayant une hauteur de 34 mm et un rayon de 5 mm.

```
volcyl(34,5)              Volume = 534.071
```

Remarque : il n'est pas nécessaire d'utiliser les noms des paramètres lorsque vous exécutez le programme **volcyl**, mais vous devez spécifier deux arguments (sous la forme de valeurs, variables ou expressions). Le premier correspond à la hauteur et le deuxième au rayon.

Demande des valeurs à l'utilisateur (programmes uniquement)

Vous pouvez utiliser les commandes **Request** et **RequestStr** dans un programme pour marquer une pause dans celui-ci et afficher une boîte de dialogue invitant l'utilisateur à fournir des informations. Cette méthode ne nécessite pas que les utilisateurs se rappellent du nom des variables ni de l'ordre dans lequel elles doivent être utilisées.

Vous ne pouvez pas utiliser la commande **Request** ou **RequestStr** dans une fonction.

1. Définissez le programme.

```
Define calculatearea()=
Prgm
  Request "Largeur : ",w
  Request "Hauteur : ",h
area:=w*h
EndPrgm
```

2. Exécutez le programme et répondez aux différentes demandes d'information.

```
calculatearea() : area
Largeur : 3          (3 saisi comme réponse)
Hauteur : 23.64  (23.64 saisi comme réponse)
                                        70.92
```

Utilisez **RequestStr** plutôt que **Request** si vous souhaitez que le programme interprète la réponse de l'utilisateur en tant que chaîne de caractère et nom comme une expression mathématique. Cela évite à l'utilisateur d'avoir à saisir sa réponse entre guillemets (“”).

Affichage d'informations

L'exécution d'un programme ou d'une fonction ne permet pas d'afficher les calculs intermédiaires, à moins de spécifier une commande permettant de les afficher. Il s'agit d'une différence importante entre un calcul effectué au niveau de la ligne de saisie et un calcul effectué dans le cadre d'une fonction ou d'un programme.

Les calculs ci-dessous, par exemple, n'affichent pas de résultat dans une fonction ou un programme (alors qu'ils en affichent un dans la ligne de saisie).

```
⋮
x:=12•6
cos(π/4)→
⋮
```

Affichage d'informations dans l'historique

Vous pouvez utiliser la commande **Disp** dans un programme ou une fonction pour afficher des informations, y compris des résultats intermédiaires, dans l'historique.

```
⋮
Disp 12•6
Disp "Résultat :",cos(π/4)
⋮
```

Affichage d'informations dans une boîte de dialogue

Vous pouvez utiliser la commande **Text** pour marquer une pause dans l'exécution d'un programme et afficher des informations dans une boîte de dialogue. Dans ce cas, les utilisateurs doivent sélectionner **OK** pour continuer ou **Annuler** pour arrêter le programme.

Vous ne pouvez pas afficher la commande **Text** dans une fonction.

```
⋮
Text "Surface =" & area
⋮
```

Remarque : l'affichage d'un résultat à l'aide de la commande **Disp** ou **Text** ne signifie pas que celui-ci est enregistré. Si vous envisagez de réutiliser ultérieurement un résultat, enregistrez-le dans une variable globale.

```
⋮
cos(π/4)→maximum
Disp maximum
⋮
```

Utilisation des variables locales

Une variable locale est une variable temporaire qui n'existe que pendant la durée d'évaluation d'une fonction définie par l'utilisateur ou d'exécution d'un programme défini par l'utilisateur.

Exemple de variable locale

L'extrait de programme suivant montre une **boucle For...EndFor** (décrite plus loin dans ce chapitre). La variable *i* correspond au compteur de boucles. Dans la plupart des cas, la variable *i* n'est utilisée que pendant l'exécution du programme.

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

❶ Déclare la variable *i* comme variable locale.

Remarque : Dans la mesure du possible, déclarez comme variable locale toutes les variables qui ne sont utilisées qu'à l'intérieur du programme et dont la disponibilité n'est pas nécessaire une fois l'exécution de celui-ci terminée.

Quelle est l'origine de l'affichage du message d'erreur relatif à une variable indéfinie ?

Un message d'erreur relatif à une variable **indéfinie** s'affiche lorsque vous évaluez une fonction définie par l'utilisateur ou exécutez un programme défini par l'utilisateur qui fait référence à une variable qui n'a pas été initialisée (c'est-à-dire, à laquelle aucune valeur n'a été assignée).

Par exemple :

```
Define fact(n)=Func
  Local m ❶
  While n>1
    n•m→m: n–1→n
  EndWhile
  Return m
  EndFunc
```

❶ Aucune valeur initiale n'a été assignée à la variable locale *m*.

Initialisez les variables locales

Toutes les valeurs locales doivent se voir assigner une valeur initiales avant de pouvoir être référencées.

```
Define fact(n)=Func
  Local m: 1→m ❶
  While n>1
    n•m→m: n–1→n
  EndWhile
  Return m
  EndFunc
```

❶ 1 est stocké comme valeur initiale de la variable *m*.

Remarque (CAS) : les fonctions et les programmes ne peuvent pas utiliser une variable locale pour effectuer des calculs symboliques.

CAS: Exécution de calculs symboliques

Pour qu'une fonction ou un programme exécute des calculs symboliques, vous devez utiliser une variable globale et non locale. Il convient néanmoins de s'assurer que la variable globale n'existe pas déjà indépendamment du programme. À cet effet, les méthodes suivantes peuvent vous aider.

- Utilisez un nom de variable globale, généralement de deux caractères ou plus, qui a peu de chances d'exister indépendamment de la fonction ou du programme.
- Insérez la commande **DelVar** dans votre programme afin de supprimer la variable globale, si elle existe, avant d'y faire référence. (**DelVar** ne permet pas de supprimer les variables verrouillées ou liées.)

Différences entre les fonctions et les programmes

Une fonction définie via l'Éditeur de programmes présente de nombreuses similitudes avec les fonctions créées dans le logiciel TI-Nspire™.

- Les fonctions doivent donner un résultat, lequel peut être représenté graphiquement ou saisi dans un tableau. Les programmes ne donnent aucun résultat.
- Vous pouvez utiliser une fonction (mais pas un programme) dans une expression. Par exemple : **3 • func1(3)** est valide, mais pas **3 • prog1(3)**.
- Vous pouvez exécuter des programmes uniquement à partir des applications Calculs et Éditeur mathématique. Toutefois, les fonctions peuvent être évaluées dans les applications Calculs, Éditeur mathématique, Tableur & listes, Graphiques & géométrie et Données & statistiques.
- Une fonction peut faire référence à n'importe quelle variable, mais ne peut stocker de valeur que dans une variable locale. Les programmes permettent de stocker des valeurs dans les variables locales et globales.

  Remarque : les arguments utilisés pour transmettre les valeurs à une fonction sont considérés automatiquement comme des variables locales. Pour les stocker dans d'autres types de variables, vous devez déclarer celles-ci comme variables locales (**Local**) depuis la fonction.
- Une fonction ne permet pas d'appeler un programme comme sous-routine mais peut, en revanche, appeler une autre fonction définie par l'utilisateur.
- Il est impossible de définir un programme à l'intérieur d'une fonction.
- Une fonction ne peut pas définir une fonction globale, mais peut définir une fonction locale.

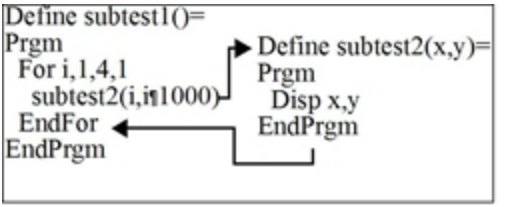
Appel d'un programme depuis un autre programme

Un programme peut appeler un autre programme et l'exécuter comme sous-routine. Cette sous-routine peut être externe (programme distinct) ou interne (programme intégré au programme principal). L'utilisation de sous-routines est utile lorsqu'un

programme doit répéter le même groupe de commandes à plusieurs emplacements différents.

Appel d'un programme distinct

Pour appeler un programme distinct, utilisez la même syntaxe que pour exécuter le programme à partir de la ligne de saisie.



Définition et appel d'une sous-routine interne

Pour définir une sous-routine interne, utilisez la commande **Define** et le bloc **Prgm...EndPrgm**. Dans la mesure où une sous-routine doit être définie avant de pouvoir être appelée, il est judicieux de définir les sous-routines au début du programme principal.

L'appel et l'exécution d'une sous-routine s'effectuent de la même façon que pour un programme distinct.

```
Define subtest1()=
  Prgm
  local subtest2  ❶
  Define subtest2(x,y)=  ❷
  Prgm
    Disp x,y
  EndPrgm
  ©Début du programme principal
  For i,1,4,1
    subtest2(i,I*1000)  ❸
  EndFor
  EndPrgm
```

❶ Déclare la sous-routine comme variable locale.

❷ Définit la sous-routine.

❸ Appelle la sous-routine.

Remarque : Utilisez le menu **Var** de l'Éditeur de programmes pour insérer les commandes **Define** et **Prgm...EndPrgm**.

Remarques relatives à l'utilisation des sous-routines

À la fin d'une sous-routine, le processus d'exécution revient au programme qui l'a appelée. Pour quitter une sous-routine à tout autre moment, utilisez la commande **Return** sans spécifier d'argument.

Une sous-routine n'a pas accès aux variables locales déclarées dans le programme qui l'a appelée. De la même façon, le programme qui appelle la sous-routine n'a pas accès aux variables locales déclarées dans celle-ci.

Les commandes Lbl sont considérées locales par rapport aux programmes dans lesquels elles se trouvent. Par conséquent, une commande **Goto** présente dans le programme appelant ne peut pas accéder à une étiquette située dans une sous-routine et inversement.

Élimination des erreurs de définition circulaire

Lors de l'évaluation d'une fonction définie par l'utilisateur ou de l'exécution d'un programme, vous pouvez spécifier un argument qui inclut la même variable que celle utilisée pour définir la fonction ou créer le programme. Cependant, pour éliminer tout risque d'erreur de définition circulaire, vous devez assigner une valeur aux variables qui sont utilisées dans le cadre de l'évaluation de la fonction ou de l'exécution du programme. Par exemple :

```
x+1→x ❶
```

– ou –

```
For i,i,10,1
  Disp i ❶
EndFor
```

❶ Génère un message d'erreur de **définition circulaire** si aucune valeur n'est assignée à x ou i. Aucune erreur ne se produit si une valeur a été assignée à x ou i.

Contrôle du déroulement d'une fonction ou d'un programme

Lorsque vous exécutez un programme ou évaluez une fonction, leurs lignes sont exécutées en ordre séquentiel. Toutefois, certaines commandes peuvent affecter le déroulement d'un programme. Par exemple :

- Les structures de contrôle telles que **If...EndIf** effectuent un test conditionnel pour déterminer la partie du programme à exécuter.
- Les commandes de boucle comme **For...EndFor** répètent un groupe de commandes.

Utilisation des commandes If, Lbl et Goto pour contrôler l'exécution des programmes

La commande **If** et plusieurs structures **If...EndIf** vous permettent de procéder à l'exécution conditionnelle d'une instruction ou un bloc d'instructions, autrement dit, en fonction du résultat d'un test (par exemple, x>5). **Les commandes Lbl** et **Goto** vous

permettent d'effectuer des enchaînements ou des sauts d'un point à un autre d'une fonction ou d'un programme.

La commande **If** et plusieurs structures **If...EndIf** sont accessibles via le menu **Contrôle (Control)** de l'Éditeur de programmes.

Lorsque vous insérez une structure de type **If...Then...EndIf**, un modèle est inséré à l'emplacement du curseur. Celui-ci est placé de sorte à vous permettre de saisir un test conditionnel.

Commande If

Pour exécuter une seule commande lorsqu'un test conditionnel est vrai, utilisez le format général suivant :

```
If x>5
  Disp "x est supérieur à 5"  ❶
Disp x  ❷
```

❶ La commande est exécutée uniquement si x>5, sinon elle est ignorée.

❷ Affiche toujours la valeur de x.

Dans cet exemple, vous devez stocker une valeur dans la variable x avant de pouvoir exécuter la commande **If** .

Structures If...Then...EndIf

Pour exécuter un group de commandes lorsqu'un test conditionnel est vrai, utilisez la structure suivante :

```
If x>5 Then
  Disp "x est supérieur à 5"  ❶
  2•x→x  ❶
EndIf
Disp x  ❷
```

❶ La commande est exécutée uniquement si x>5.

❷ Affiche la valeur de :
2x if x>5
x if x≤5

Remarque : EndIf marque la fin du bloc **Then** qui est exécuté si la condition est vraie.

Structures If...Then...Else... EndIf

Pour exécuter un groupe de commandes si un test conditionnel est vrai et un autre groupe de commandes si la condition est fausse, utilisez la structure suivante :

```
If x>5 Then
  Disp "x est supérieur à 5"  ❶
```

```
  2•x→x ❶
Else
  Disp "x est inférieur ou égal à 5" ❷
  5•x→x ❷
EndIf
Disp x ❸
```

❶ La commande est exécutée uniquement si x>5.

❷ La commande est exécutée uniquement si x≤5.

❸ Affiche la valeur de :
2x if x>5
5x if x≤5

Structures If...Then...ElseIf... EndIf

Une forme plus complexe de la commande If vous permet de tester plusieurs conditions. Par exemple, vous souhaitez qu'un programme teste un argument spécifié par l'utilisateur qui correspond à une des quatre options possibles.

Pour tester chaque option (If Choix=1, If Choix=2, et ainsi de suite.), utilisez la structure **If...Then...ElseIf...EndIf**.

Commandes Lbl et Goto

Vous pouvez également contrôler l'exécution d'un programme à l'aide des commandes **Lbl** et **Goto**. Ces commandes sont accessibles via le menu **Transfert (Transfer)** de l'Éditeur de programmes.

Utilisez la commande **Lbl** pour marquer d'une étiquette (assigner un nom à) un emplacement spécifique d'une fonction ou d'un programme.

Lbl *nomÉtiquette*	nom à assigner à l'emplacement (utilisez les conventions de dénomination applicables aux noms de variables)

Vous pouvez ensuite utiliser la commande **Goto** en tout point de la fonction ou du programme pour créer un enchaînement avec l'emplacement qui correspond à l'étiquette spécifiée.

Goto *nomÉtiquette*	spécifie la commande **Lbl** avec laquelle créer l'enchaînement

La commande **Goto** étant inconditionnelle (elle exécute toujours l'enchaînement avec l'étiquette spécifiée), elle est souvent utilisée avec une commande **If** de façon à pouvoir spécifier un test conditionnel. Par exemple :

```
If x>5
  Goto GT5 ❶
Disp x
--------
-------- ❷
```

```
Lbl GT5
Disp "Le nombre était > 5"
```

❶ Si x>5, on passe directement à l'étiquette GT5.

❷ Pour cet exemple, le programme doit inclure des commandes (comme **Stop**) qui empêchent l'exécution de la commande **Lbl** GT5 si x≤5.

Utilisation des boucles pour répéter un groupe de commandes

Pour répéter le même groupe de commandes à plusieurs reprises, utilisez l'une des structures de boucle suivantes. Vous disposez de plusieurs types de boucles. Chaque type de boucle propose une méthode de sortie de la boucle différente, basée sur un test conditionnel.

Les commandes de boucle et autres commandes associées sont accessibles via les menus **Contrôle (Control)** et **Transfert (Transfer)** de l'Éditeur de programmes.

Lorsque vous utilisez l'une des structures de boucle, le modèle correspondant est inséré à l'emplacement du curseur. Vous pouvez alors commencer à saisir les commandes à exécuter à l'intérieur de la boucle.

Boucles For...EndFor

Une boucle **For...EndFor** utilise un compteur pour contrôler le nombre d'exécution de la boucle. La syntaxe de la commande **For** est la suivante :

Remarque : La valeur de fin peut être inférieure à la valeur de départ, à condition que l'incrément soit négatif.

For *variable*, *début*, *fin*[, *incrément*]

❶ ❷ ❸ ❹

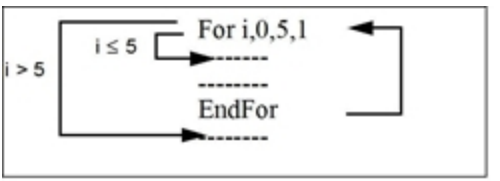
❶ *Variable* utilisée comme compteur

❷ Valeur du compteur utilisée lors de la première exécution de la boucle **For**

❸ Quitte la boucle lorsque la *variable* excède cette valeur.

❹ Ajouté au compteur à chaque nouvelle exécution de la boucle **For** (si cette valeur facultative n'est pas utilisée, la valeur de l'*incrément* est 1.)

Lors de l'exécution de la boucle **For**, la valeur de la *variable* est comparée à la valeur de *fin*. Si la *variable* n'excède pas la valeur de *fin*, la boucle est exécutée. Dans le cas contraire, l'exécution du programme se poursuit en effectuant un saut jusqu'à la commande située après **EndFor**.



Remarque : la commande **For** incrémente automatiquement la variable du compteur de sorte que la fonction ou le programme puisse sortir de la boucle après un certain nombre d'exécutions de celle-ci.

À la fin de la boucle (**EndFor**), le programme revient au niveau de la commande **For**, incrémente la variable et la compare à la valeur de *fin*.

Par exemple :

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

❶ Affiche 0, 1, 2, 3, 4 et 5.

❷ Affiche 6. Lorsque la *variable* atteint 6, la boucle n'est pas exécutée.

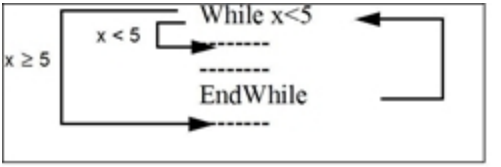
Remarque : Vous pouvez déclarer la variable du compteur comme variable locale si elle ne doit pas être enregistrée après l'arrêt de la fonction ou du programme.

Boucles While...EndWhile

Une boucle **While...EndWhile** répète un bloc de commandes tant qu'une condition spécifiée est vraie. La syntaxe de la commande **While** est la suivante :

While *condition*

Lorsque la commande **While** est exécutée, la *condition* est évaluée. Si la *condition* est vraie, la boucle est exécutée. Dans le cas contraire, l'exécution du programme se poursuit en effectuant un saut jusqu'à la commande située après **EndWhile**.



Remarque : la commande **While** ne change pas automatiquement la condition. Vous devez ajouter des commands qui permettent à la fonction ou au programme de sortir de la boucle.

À la fin de la boucle (**EndWhile**), l'exécution du programme se poursuit de nouveau au niveau de la commande **While**, où la condition est réévaluée.

Pour exécuter la boucle la première fois, la condition doit être vrai.

- Toutes les variables référencées dans la condition doivent être définies avant l'exécution de la commande **While**. (Vous pouvez définir les valeurs dans une fonction ou un programme ou encore demander à l'utilisateur de les saisir.)

- La boucle doit contenir les commandes qui modifient les valeurs de la condition, et éventuellement font qu'elle soit fausse. Sinon, la condition est toujours vraie et la fonction ou le programme ne peut pas sortir de la boucle (laquelle devient une boucle infinie).

Par exemple :

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

❶ Définit la variable x.

❷ Affiche 0, 1, 2, 3 et 4.

❸ Incrémente la variable x.

❹ Affiche 5. Lorsque x atteint 5, la boucle n'est pas exécutée.

Boucles Loop...EndLoop

Loop...EndLoop crée une boucle infinie, qui se répète indéfiniment. La commande **Loop** n'a pas d'argument.

En général, vous insérez dans une boucle les commandes permettant au programme de sortir de celle-ci. Les commandes souvent utilisées à cet effet sont les suivantes : **If**, **Exit**, **Goto** et **Lbl** . Par exemple :

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

❶ Une commande **If** vérifie la condition.

❷ Sort de la boucle et revient à cet emplacement lorsque x atteint 6.

Remarque : la commande **Exit** permet de sortir de la boucle active.

Dans cet exemple, la commande **If** peut se trouver en tout point de la boucle.

Si la commande If se trouve :	La boucle est :
Au début de la boucle	Exécutée sous réserve que la condition soit vraie.
À la fin de la boucle	Exécutée au moins une fois, puis répétée seulement si la condition est vraie.

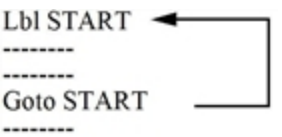
La commande **If** peut être utilisée conjointement à une commande **Goto** pour que l'exécution du programme passe sous le contrôle d'une commande **Lbl** spécifiée.

Répétition immédiate d'une boucle

La commande **Cycle** permet de transférer immédiatement le contrôle de l'exécution d'un programme à l'itération suivante d'une boucle (avant la fin de l'exécution de l'itération en cours). Cette commande s'utilise avec les blocs **For...EndFor**, **While...EndWhile** et **Loop...EndLoop**.

Boucles Lbl et Goto

Bien que les commandes **Lbl** et **Goto** ne soient pas des commandes de boucle à proprement parler, elles peuvent être utilisées pour créer une boucle infinie. Par exemple :



Comme dans le cas du bloc **Loop...EndLoop**, la boucle doit contenir des commandes permettant à la fonction ou au programme de sortir de celle-ci.

Changement des réglages de mode

Les fonctions et les programmes peuvent utiliser la fonction **setMode()** pour définir temporairement des modes de calcul ou d'affichage des résultats spécifiques. Le menu **Mode** de l'Éditeur de programmes permet d'utiliser la syntaxe correcte sans qu'il soit nécessaire de mémoriser des codes numériques.

Remarque : les changements de mode effectués dans une définition de fonction ou de programme ne sont pas conservés une fois la fonction ou le programme exécuté.

Réglage d'un mode

1. Placez le curseur à l'emplacement où insérer la fonction **setMode**.
2. Dans le menu **Mode**, sélectionnez le mode à modifier, puis le nouveau réglage.

   La syntaxe appropriée est insérée à l'emplacement du curseur. Par exemple :

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

Débogage des programmes et gestion des erreurs

Après avoir écrit une fonction ou un programme, vous pouvez utiliser différentes techniques pour rechercher et corriger les erreurs. Vous pouvez également créer une commande de gestion des erreurs à l'intérieur d'une fonction ou d'un programme.

Si votre fonction ou programme permet à l'utilisateur d'effectuer un choix parmi plusieurs options, veillez à l'exécuter et à tester chacune des options.

Techniques de débogage

Les messages d'erreur d'exécution peuvent localiser les erreurs de syntaxe, mais pas celles liées à la logique du programme. Les techniques suivantes peuvent vous être utiles.

- Insérez temporairement des commandes **Disp** pour afficher les valeurs des variables importantes.
- Pour vérifier qu'une boucle a été exécutée le nombre de fois voulu, utilisez la commande **Disp** pour afficher la variable du compteur ou les valeurs du test conditionnel.
- Pour confirmer l'exécution d'une sous-routine, utilisez la commande **Disp** pour afficher des messages comme « Début sous-routine » et « Sortie sous-routine » au début et à la fin de la sous-routine.
- Pour arrêter manuellement un programme ou une fonction,
  - Windows® : maintenez enfoncée la touche **F12** et appuyez sur **Enter** plusieurs fois.
  - Macintosh® : maintenez enfoncée la touche **F5** et appuyez sur **Enter** plusieurs fois.
  - Unité : maintenez enfoncée la touche [on] et appuyez sur [enter] plusieurs fois.

Commandes de gestion des erreurs

Commande	Description
Try...EndTry	Définit un bloc qui permet à une fonction ou un programme d'exécuter une commande et, si nécessaire, d'assurer la reprise de l'exécution après une erreur générée par cette commande.
ClrErr	Efface l'état d'erreur et règle la variable système *errCode sur zéro. Pour un découvrir un exemple d'utilisation de la variable errCode, voir la commande* ***Try dans le Guide de référence****.*
PassErr	Passe une erreur au niveau suivant du bloc **Try...EndTry**.

Informations générales

Aide en ligne

education.ti.com/eguide

Sélectionnez votre pays pour obtenir d'autres informations relatives aux produits.

Contacter l'assistance technique TI

education.ti.com/ti-cares

Sélectionnez votre pays pour obtenir une assistance technique ou d'autres types de support.

Informations Garantie et Assistance

education.ti.com/warranty

Sélectionnez votre pays pour plus de renseignements concernant la durée et les conditions de la garantie ou de l'assistance pour ce produit.

Garantie limitée. Cette garantie n'affecte pas vos droits statutaires.