TI-Nspire™ programma-editor gids

Voor meer informatie over de technologie van TI kunt u de online hulppagina raadplegen op education.ti.com/eguide.

Belangrijke gegevens

Tenzij uitdrukkelijk anders vermeld in de licentie die een programma begeleidt, geeft Texas Instruments geen garantie, uitdrukkelijk of impliciet, met inbegrip van maar niet beperkt tot enige impliciete garanties van verkoopbaarheid en geschiktheid voor een bepaald doel, met betrekking tot programma's of boekenmateriaal en maakt dergelijke materialen die uitsluitend op een "as-is" basis zijn. Texas Instruments is in geen geval aansprakelijk voor speciaal, onderpand, incidentele of gevolgschade in verband met of voortvloeiende uit de aankoop of het gebruik van deze materialen en de exclusieve aansprakelijkheid van Texas Instruments, ongeacht de vorm van actie, mag niet hoger zijn dan het bedrag dat in de licentie voor het programma is vermeld. Bovendien is Texas Instruments niet aansprakelijk voor enige vordering van welke aard dan ook tegen het gebruik van deze materialen door een andere partij.

De TI-Nspire™ software maakt gebruik van Lua als scriptomgeving. Zie http://www.lua.org/license.html voor informatie over auteursrechten en licenties.

De TI-Nspire™ software maakt gebruik van Chipmunk Physics versie 5.3.4 als simulatie omgeving. Zie voor licentie informatie http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/.

Microsoft® en Windows® zijn gedeponeerde handelsmerken van Microsoft Corporation in de Verenigde Staten en / of andere landen.

Mac OS®, iPad® en OS X® zijn geregistreerde handelsmerken van Apple Inc.

Unicode® is een geregistreerd handelsmerk van Unicode, Inc. in de Verenigde Staten en andere landen.

Feitelijke producten kunnen enigszins afwijken van de getoonde afbeeldingen.

Inhoud

Aan de slag met de Programma-editor

U kunt door de gebruiker gedefinieerde functies of programma´s maken door definitiebeweringen te typen op de invoerregel van Rekenmachine of door de Programma-editor te gebruiken. De Programma-editor biedt enkele voordelen die in deze paragraaf aan de orde komen. Voor meer informatie zie *Rekenmachine*.

- De editor heeft programmeertemplates en dialoogvensters om om u te helpen functies en programma's te definiëren met de juiste syntax.
- Met de editor kunt u meerregelige programmabeweringen invoeren zonder dat een speciale toetsenreeks voor het toevoegen van elke regel nodig is.
- U kunt eenvoudig persoonlijke en openbare bibliotheekobjecten (variabelen, functies en programma's) creëren. Voor meer informatie zie *Bibliotheken*.

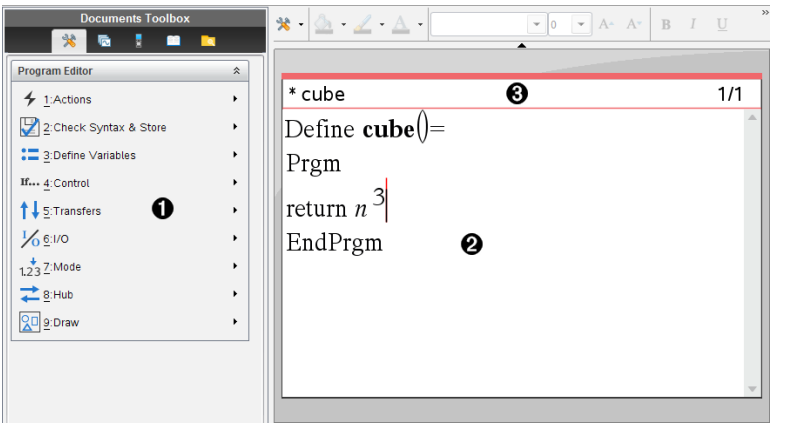
De Programma-editor starten

▶ Een nieuwe Programma-editorpagina aan de huidige opgave toevoegen:

Klik vanuit de taakbalk op **Invoegen> Programma-editor > Nieuw**.

Rekenmachine: Druk op doc en selecteer **Invoegen > Programma editor > Nieuw**.

Opmerking: De editor is toegankelijk vanuit het menu **Functies &Programma's** van een Rekenmachine-pagina.



❶ Menu van de Programma-editor – Dit menu is steeds beschikbaar wanneer u in het werkgebied van de Programma-editor bent en de normale weergavemodus gebruikt.

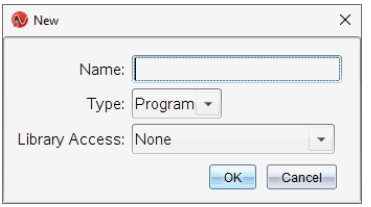
❷ Werkgebied Programma-editor

❸ De statusregel toont informatie over het regelnummer en de naam van de functie of het programma dat bewerkt wordt. Een asterisk (*) geeft aan dat deze functie "vuil" is, wat betekent dat deze gewijzigd is sinds de laatste keer dat de syntax werd gecontroleerd en de functie werd opgeslagen.

Een programma of functie definiëren

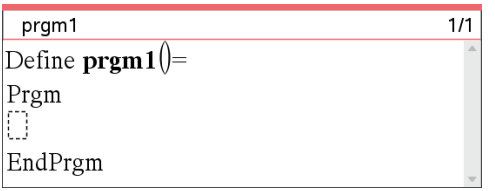
Een nieuwe Programma-editor starten

1. Zorg ervoor dat u zich in het document en probleem bevindt waarin u het programma of de functie wilt maken.
2. Klik op de knop **Invoegen** op de toepassingswerkbalk en selecteer **Programma-editor> Nieuw**. (Druk op de rekenmachine op doc en selecteer **Invoegen > Programma-editor > Nieuw**.)



3. Vul een naam in voor de functie of het programma dat u definieert.
4. Selecteer het **type** (**programma** of **functie**).
5. Stel de **Bibliotheektoegang** in:
   - Om de functie of het programma alleen uit het huidige document en probleem te gebruiken, selecteert u **Geen**
   - Om de functie of het programma toegankelijk te maken vanuit elk document maar niet zichtbaar in de Catalogus, selecteert u **LibPriv**.
   - Om de functie of het programma toegankelijk te maken vanuit elk document en ook zichtbaar in de Catalogus, selecteert u **LibPub (Weergeven in catalogus)**. Voor meer informatie zie *Bibliotheken*.
6. Klik op **OK**.

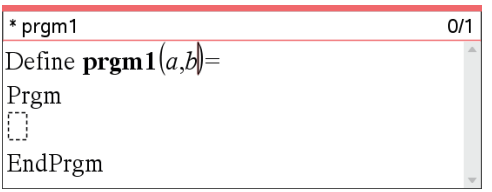
   Een nieuw exemplaar van de Programma-editor wordt geopend, met een sjabloon dat overeenkomt met de selecties die u hebt gemaakt.



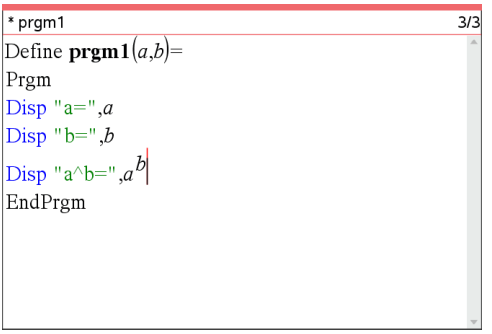
Regels invoeren in een functie of programma

De Programma-editor voert de opdrachten niet uit of werkt uitdrukkingen niet uit terwijl u ze typt. Ze worden pas uitgevoerd wanneer u de functie uitwerkt of het programma uitvoert.

1. Als uw functie of programma de gebruiker nodig heeft om argumenten te leveren, vult u de parameternamen tussen de haakjes achter de naam in. Scheidt parameters met een komma.



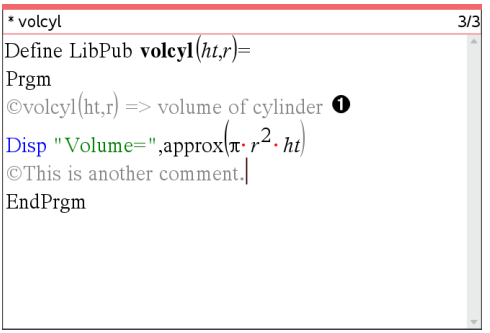
2. Tussen de regels Func en EndFunc (of Prgm en EndPrgm) vult u de regels met instructies in die deel uitmaken van uw functie of programma.



- U kunt de namen van functies en opdrachten invullen of deze invoegen vanuit de Catalogus.
- Een lijn kan langer zijn dan de breedte van het scherm; in dat geval moet u misschien scrollen om de volledige instructie te bekijken.
- Druk na het invullen van elke regel op **Enter**. Hierdoor wordt een lege regel ingevoegd en kunt u nog een regel invoegen.
- Gebruik de pijltoetsen ◄, ►, ▲ en ▼ om door de functie of het programma te bladeren voor het invoeren of bewerken van opdrachten.

Opmerkingen invoegen

Opmerkingen kunnen nuttig zijn voor iemand die het programma bekijkt of aanpast. Ze worden niet weergegeven als het programma wordt uitgevoerd en hebben geen effect op de programmastroom. Het ©-symbool verschijnt aan het begin van de regel met de opmerking.



❶ Commentaar met vereiste syntax. Omdat dit bibliotheekobject openbaar is en deze opmerking de eerste regel in een Func- of Prgm-blok is, wordt de opmerking als Help in de Catalogus weergegeven. Voor meer informatie zie *Bibliotheken*.

Om een opmerking in te voegen:

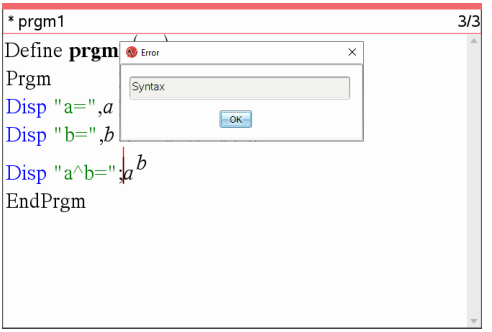
1. Plaats de cursor aan het einde van de regel waarin u een opmerking wilt invoegen.
2. Klik in het menu **Acties** op **Opmerking invoegen** of druk op **Ctrl+T**.
3. Vul de tekst van de opmerking achter het ©-symbool in.

Syntax controleren

Met de Programma-editor kunt u de functie of het programma controleren op correcte syntax.

- Klik in het menu **Syntax controleren en opslaan** op **Syntax controleren**.

  Als de syntaxcontrole eventuele syntaxfouten vindt, wordt er een foutmelding weergegeven. Ook wordt dan geprobeerd de cursor bij de eerste fout te plaatsen, zodat u deze kunt corrigeren.



De functie of het programma opslaan

U moet uw functie of programma opslaan om deze toegankelijk te maken. De Programma-editor controleert automatisch de syntax voor het opslaan.

Een sterretje (*) wordt weergegeven in de linkerbovenhoek van de Programma-editor om aan te geven dat de functie of het programma niet is opgeslagen.

- Klik in het menu **Syntax controleren en opslaan** op **Syntax controleren en opslaan**.

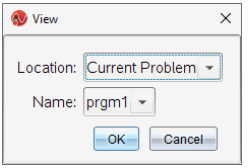
  Als de syntaxcontrole eventuele syntaxfouten vindt, geeft hij een foutmelding weer en probeert hij de cursor bij de eerste fout te plaatsen.

  Als er geen syntaxfouten gevonden worden, verschijnt de melding "Opgeslagen" in de statusregel aan de bovenkant van de programma-editor.

Opmerking: Als de functie of het programma is gedefinieerd als een bibliotheekobject, moet u het document ook opslaan in de aangewezen bibliotheekmap en bibliotheken vernieuwen om het object toegankelijk te maken voor andere documenten. Voor meer informatie zie *Bibliotheken*.

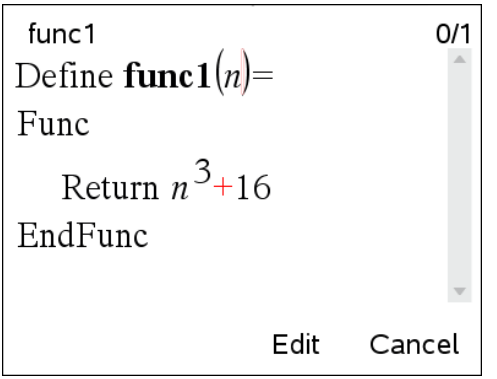
Een programma of functie bekijken

1. Selecteer in het menu **Acties** de optie **Beeld**.



2. Als de functie of het programma een bibliotheekobject is, selecteer dan de bibliotheek vanuit de lijst **Locatie**.
3. Selecteer de naam van de functie of het programma uit de lijst **Naam**.

   De functie of het programma wordt weergegeven in een viewer.



4. Gebruik de pijltoetsen om de functie of het programma te bekijken.
5. Klik op **Bewerken** als u het programma wilt bewerken.

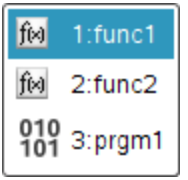
Opmerking: de selectie **Bewerken** is alleen beschikbaar voor functies en programma's die gedefinieerd zijn in de huidige opgave. Om een bibliotheekobject te bewerken moet u eerst het bibliotheekdocument openen.

Een functie of programma openen om te bewerken

U kunt een functie of programma alleen vanuit de huidige opgave openen.

Opmerking: u kunt een vergrendeld programma of een vergrendelde functie niet wijzigen. Om het object te ontgrendelen gaat u naar een Rekenmachine-pagina en gebruikt u het commando **unLock**.

1. Geef de lijst met beschikbare functies en programma's weer.
   - Selecteer in het menu **Acties** de optie **Openen**.

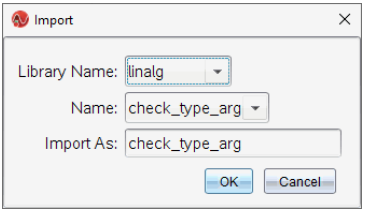


2. Selecteer het item dat u wilt openen.

Een programma importeren vanuit een bibliotheek

U kunt een functie of programma dat gedefinieerd is als bibliotheekobject importeren in een Programma-editor binnen de huidige opgave. De geïmporteerde kopie is niet vergrendeld, ook al is het origineel dat wel.

1. Selecteer in het menu **Acties** de optie **Importeren**.



2. Selecteer de **Bibliotheeknaam**.
3. Selecteer de **Naam** van het object.
4. Als u wilt dat het geïmporteerde object een andere naam krijgt, type de naam dan onder **Importeren als**.

Een kopie van een functie of programma maken

Wanneer u een nieuwe functie of een nieuw programma creëert, vindt u het misschien makkelijker om te beginnen met een kopie van het huidige programma. De kopie die u creëert is niet vergrendeld, ook al is het origineel dat wel.

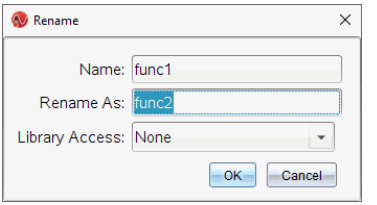
1. Selecteer in het menu **Acties** de optie **Kopie creëren**.

2. Typ een nieuwe naam of klik op **OK** om de voorgestelde naam te accepteren.
3. Als u het toegangsniveau wilt veranderen, selecteer dan **Bibliotheektoegang** en selecteer een nieuw niveau.

Een programma of functie een andere naam geven

U kunt de huidige functie of het huidige programma een andere naam geven, of (optioneel) het toegangsniveau ervan wijzigen.

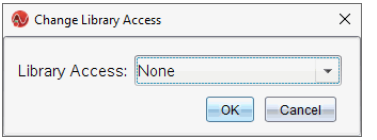
1. Selecteer in het menu **Acties** de optie **Hernoemen**.



2. Typ een nieuwe naam of klik op **OK** om de voorgestelde naam te accepteren.
3. Als u het toegangsniveau wilt veranderen, selecteer dan **Bibliotheektoegang** en selecteer een nieuw niveau.

Het toegangsniveau van de bibliotheek veranderen

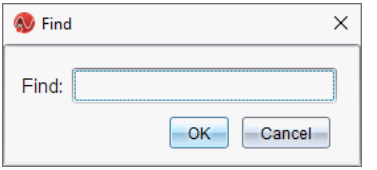
1. Selecteer in het menu **Acties** de optie **Bibliotheektoegang veranderen**.



2. Selecteer de **Bibliotheektoegang**:
   - Als u de functie of het programma alleen vanuit de huidige Rekenmachine-opgave wilt gebruiken, selecteer dan **Geen**.
   - Als u wilt dat de functie of het programma toegankelijk is vanuit elk document, maar niet zichtbaar in de Catalogus, selecteer dan **LibPriv**.
   - Als u wilt dat de functie of het programma toegankelijk is vanuit elk document en ook zichtbaar in de Catalogus, selecteer dan **LibPub**.

Tekst zoeken

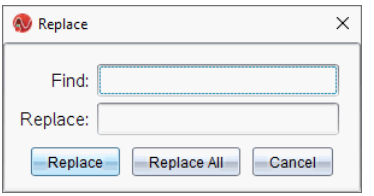
1. Selecteer in het menu **Acties** de optie **Zoeken**.



2. Typ de tekst die u wilt zoeken en klik op **OK**.
   - Als de tekst is gevonden, wordt deze gemarkeerd in he programma.
   - Als de tekst niet is gevonden, verschijnt er een melding.

Tekst zoeken en vervangen

1. Selecteer in het menu **Acties** de optie **Zoeken en vervangen**.



2. Typ de tekst die u wilt zoeken.
3. Typ de vervangende tekst.
4. Klik op **Vervangen** om de eerste keer dat de tekst voorkomt na de cursorpositie deze te vervangen, of klik op **Alles vervangen** om elke keer dat de tekst voorkomt deze te vervangen.

   Opmerking: als de tekst wordt gevonden in een wiskundetemplate, dan wordt er een bericht weergegeven om u te waarschuwen dat uw vervangende tekst de hele template zal vervangen — niet alleen de gevonden tekst.

Huidige functie of programma sluiten

- Selecteer in het menu **Acties** de optie **Sluiten**.

  Als de functie of het programma niet-opgeslagen veranderingen bevat, dan wordt u verzocht om de syntax te controleren en op te slaan voordat u het sluit.

Programma's uitvoeren en functies uitwerken

Nadat u een programma of functie hebt gedefinieerd en opgeslagen, kunt u deze vanuit een toepassing gebruiken. Alle toepassingen kunnen functies uitwerken, maar alleen de toepassingen rekenmachine en notitie kunnen programma's uitvoeren.

De opdrachten van het programma worden in volgorde uitgevoerd (hoewel sommige opdrachten het programmaverloop wijzigen). De uitvoer, indien aanwezig, wordt weergegeven in het werkgebied van de toepassing.

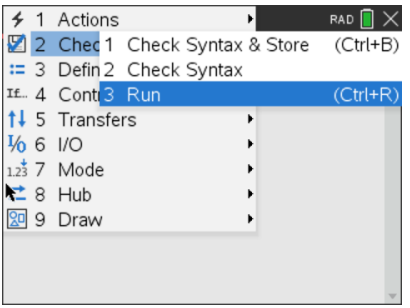
- De uitvoering van een programma gaat door tot de laatste opdracht of een **Stop** opdracht bereikt wordt.
- Functie-uitvoering gaat door totdat het een **Return**-opdracht bereikt.

Een programma of functie uitvoeren vanuit de Programma-editor

1. Zorg ervoor dat u een programma of functie hebt gedefinieerd en dat de Programma-editor het actieve paneel (computer) of de actieve pagina (rekenmachine) is.

2. Klik op de werkbalk op de **knop** Document Tools  en selecteer **Check Syntax & Store > Run**.
   —of—
   Druk op **Ctrl+R**.

   Rekenmachine: Druk op menu 2 3, of druk op ctrl R.



   Dit zal automatisch:

   - de syntax controleren en het programma of de functie opslaan,
   - plak de programma- of functienaam op de eerste beschikbare regel van de toepassing Rekenmachine die onmiddellijk volgt op de Programma-editor. Als er geen Rekenmachine op die plaats bestaat, wordt er een nieuwe ingevoegd.

3. Als het programma of de functie eist dat u een of meerdere argumenten geeft, typt u deze waarden of variabelenamen tussen de haakjes.

4. Druk op enter.

Opmerking: U kunt ook een programma of functie in de toepassingen Rekenmachine of Notitie uitvoeren door de naam van het programma met haakjes en eventuele vereiste argumenten in te typen en te drukken op enter.

Korte en lange namen gebruiken

Wanneer u in dezelfde opdracht bent als waarin een object gedefinieerd is, kunt u dit openen door de korte naam in te voeren (de naam die is gegeven in de opdracht **Define**

van het object). Dit geldt voor alle gedefinieerde objecten, inclusief persoonlijke, publieke en niet-bibliotheekobjecten.

U kunt toegang tot een bibliotheekobject krijgen vanuit elk document door de lange naam van het object in te typen. Een lange naam bestaat uit de naam van het bibliotheekdocument van het object, gevolgd door een backslash "\", gevolgd door de naam van het object. Bijvoorbeeld, de lange naam van het object gedefinieerd als **func1** in het bibliotheekdocument is **lib1 lib1\func1**. Om het \-teken op de rekenmachine te typen, drukt u op [shift] [÷].

Opmerking: Als u zich de exacte naam of de volgorde van argumenten die u nodig hebt voor een persoonlijk bibliotheekobject niet kunt herinneren, kunt u het bibliotheekdocument openen of de Programma-editor gebruiken om het object te bekijken. U kunt ook **getVarInfo** gebruiken om een lijst met objecten in een bibliotheek te bekijken.

Een persoonlijke bibliotheek programma of functie gebruiken

1. Zorg ervoor dat u het object in de eerste opdracht van het document hebt gedefinieerd, het object hebt opgeslagen, het bibliotheekdocument hebt opgeslagen in de map MyLib en de bibliotheken heeft vernieuwd.
2. Open de TI-Nspire™-toepassing waarin u het programma of de functie wilt gebruiken.

   Opmerking: Alle toepassingen kunnen functies uitwerken, maar alleen de toepassingen Rekenmachine en Notitie kunnen programma's uitvoeren.
3. Open de Catalogus en gebruik het tabblad Bibliotheek om het object te vinden en in te voegen.
   —of—
   Typ de naam van het object. Zet bij een programma of functie altijd haakjes achter de naam.

   ```
   libs2\func1()
   ```
4. Als het programma of de functie eist dat u een of meerdere argumenten geeft, typt u deze waarden of variabelenamen tussen de haakjes.

   ```
   libs2\func1(34,power)
   ```
5. Druk op [enter].

Een programma of functie uit een Persoonlijke bibliotheek gebruiken

Om een object uit een Privé bibliotheek te gebruiken, moet u de lange naam kennen. Bijvoorbeeld, de lange naam van het object gedefinieerd als **func1** in het bibliotheekdocument is **lib1 lib1\func1**.

Opmerking: Als u zich de exacte naam of de volgorde van argumenten die u nodig hebt voor een persoonlijk bibliotheekobject niet kunt herinneren, kunt u het bibliotheekdocument openen of de Programma-editor gebruiken om het object te bekijken.

1. Zorg ervoor dat u het object in de eerste opdracht van het document hebt gedefinieerd, het object hebt opgeslagen, het bibliotheekdocument hebt opgeslagen in de map MyLib en de bibliotheken heeft vernieuwd.
2. Open de TI-Nspire™-toepassing waarin u het programma of de functie wilt gebruiken.

   Opmerking: Alle toepassingen kunnen functies uitwerken, maar alleen de Rekenmachine- en Notitie-toepassingen kunnen programma's uitvoeren.
3. Typ de naam van het object. Zet bij een programma of functie altijd haakjes achter de naam.

   ```
   libs2\func1()
   ```
4. Als het programma of de functie eist dat u een of meerdere argumenten geeft, typt u deze waarden of variabele namen tussen de haakjes.

   ```
   libs2\func1(34,power)
   ```
5. Druk op enter.

Een lopend programma of functie onderbreken

Wanneer een programma of functie wordt uitgevoerd, wordt de aanwijzer "bezig" ⏲ weergegeven.

- Het programma of de functie stoppen,
  - Windows®: Houd **F12** ingedrukt en druk herhaaldelijk op **Enter**.
  - Mac®: Houd **F5** ingedrukt en druk herhaaldelijk op **Enter**.
  - Rekenmachine: Houd de toets ⌂ on ingedrukt en druk herhaaldelijk op enter.

  Er verschijnt een bericht. Als u het programma of de functie in de programmabeheerder wilt wijzigen, selecteert u **Ga naar**. De cursor verschijnt bij het commando waar de pauze plaatsvond.

Waarden aan een programma doorgeven

U kunt kiezen uit verschillende manieren om de waarden, die een functie of programma bij berekeningen gebruikt, aan te leveren.

De waarden binnen het programma of de functie inbedden

Deze methode wordt voornamelijk gebruikt voor waarden die elke keer als het programma of de functie wordt gebruik hetzelfde moeten zijn.

1. Definieer het programma.

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

2. Voer het programma uit.

```
calculatearea()
                                  70.92
                                  Done
```

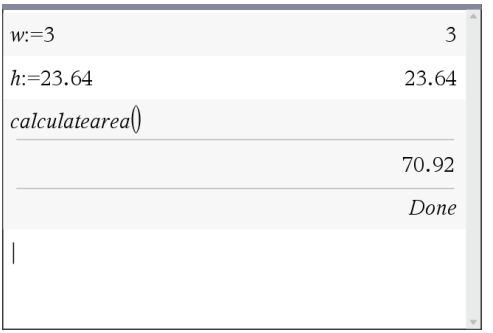
De gebruiker waarden laten toekennen aan variabelen

Een programma of functie kan verwijzen naar variabelen die van tevoren zijn gecreëerd. Bij deze methode moeten de gebruikers de variabelenamen onthouden en er waarden aan toekennen voor ze het object gebruiken.

1. Definieer het programma.

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

2. Voer de variabelen in en voer het programma uit.

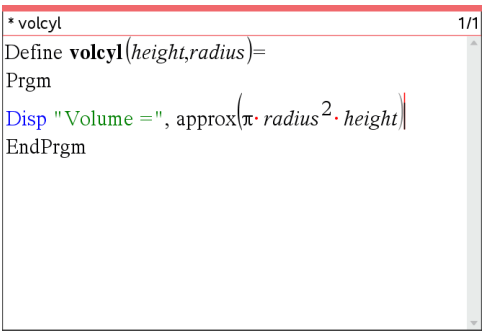


De gebruiker de waarden als argumenten laten aanleveren

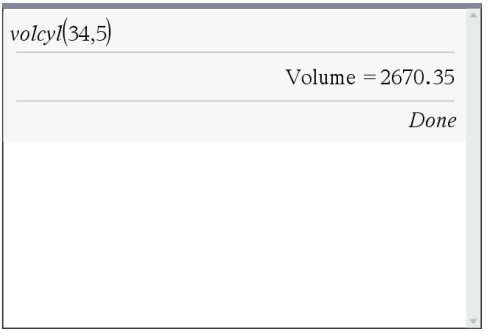
Bij deze methode leveren de gebruikers één of meer waarden aan als argumenten in de uitdrukking die het programma of de functie aanroept.

Het volgende programma **volcyl**, berekent het volume van een cilinder. De gebruiker moet hierin twee waarden aanleveren: de hoogte en de straal van de cilinder.

1. Definieer het **volcyl**-programma.



2. Voer het programma uit om het volume van een cilinder met een hoogte van 34 mm en een straal van 5 mm weer te geven.



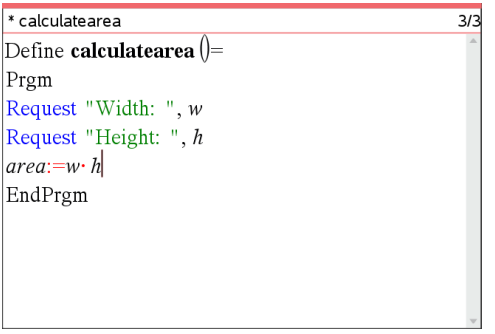
Opmerking: u hoeft de parameternamen niet te gebruiken wanneer u het **volcyl**-programma uitvoert, maar u moet twee argumenten aanleveren (als waarden, variabelen of uitdrukkingen). Het eerste argument moet de hoogte voorstellen en het tweede de straal.

De gebruiker om de waarden verzoeken (alleen bij programma's)

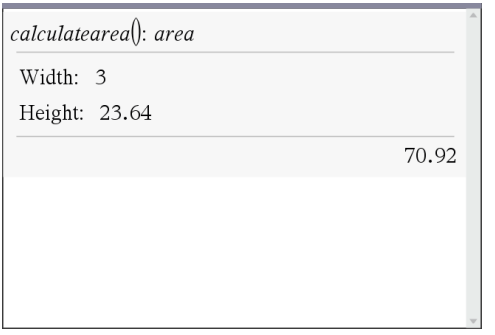
U kunt de commando's **Request** en **RequestStr** gebruiken in een programma om het programma te laten pauzeren en een dialoogvenster te laten weergeven, waarin de gebruiker om informatie wordt verzocht. Bij deze methode hoeven de gebruikers geen variabelenamen te onthouden, noch de volgorde waarin ze nodig zijn.

U kunt het commando **Request** of **RequestStr** niet gebruiken in een functie.

1. Definieer het programma.



2. Voer het programma uit en geef antwoord op de vragen.

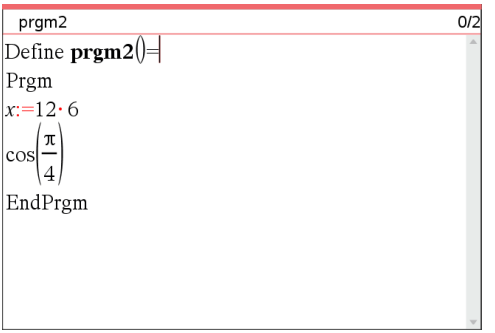


Gebruik **RequestStr** in plaats van **Request** als u wilt dat het programma het antwoord van de gebruiker interpreteert als een tekenreeks in plaats van als een wiskundige uitdrukking. Hierdoor hoeft de gebruiker het antwoord niet tussen aanhalingstekens (“”) te zetten.

Informatie weergeven

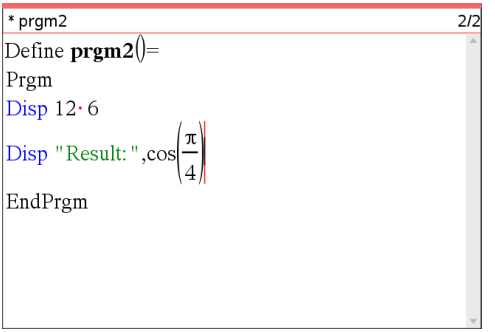
Een lopende functie of programma geeft geen tussentijds berekende resultaten weer, tenzij u een commando opneemt om deze wel weer te geven. Dit is een belangrijk verschil tussen het uitvoeren van een berekening op de invoerregel en het uitvoeren van een berekening in een functie of programma.

De volgende berekeningen bijvoorbeeld, geven geen resultaat weer in een functie of programma (terwijl ze dit wel doen vanaf de invoerregel).



Informatie in de geschiedenis weergeven

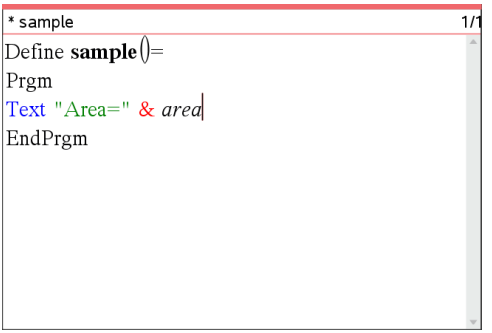
U kunt het commando **Disp** in een programma of functie gebruiken om informatie, inclusief tussentijdse resultaten, in de geschiedenis weer te geven.



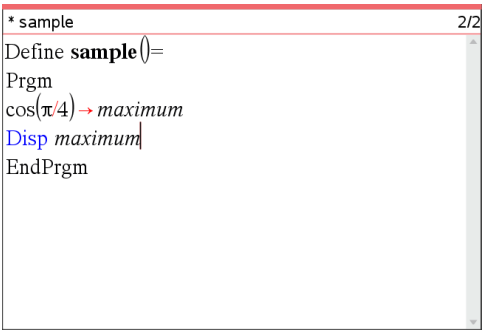
Informatie in een dialoogvenster weergeven

U kunt het commando **Text** gebruiken om een lopend programma te laten pauzeren en informatie in een dialoogvenster te laten weergeven. De gebruiker selecteert **OK** om verder te gaan of **Annuleren** om het programma te stoppen.

U kunt het commando **Text niet in een functie gebruiken.**



Opmerking: Het laten weergeven van een resultaat met **Disp** of **Text**, laat dit resultaat niet opslaan. Als u later naar het resultaat wilt kunnen verwijzen, sla het dan op in een algemene variabele.

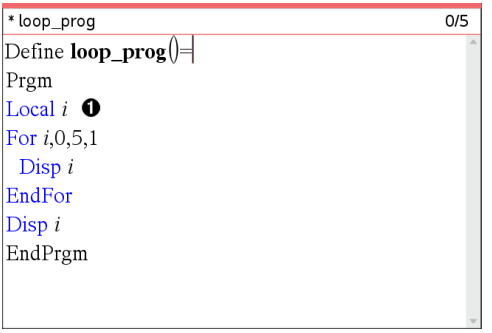


Lokale variabelen gebruiken

Een lokale variabele is een tijdelijke variabele die alleen bestaat zolang een door de gebruiker gedefinieerde functie wordt uitgewerkt, of zolang een door de gebruiker gedefinieerd programma wordt uitgevoerd.

Voorbeeld van een lokale variabele

Het volgende programma-onderdeel toont een **For...EndFor loop** (die later wordt besproken in deze module). De variabele *i* is de teller. In de meeste gevallen wordt de variabele *i* alleen gebruikt terwijl het programma wordt uitgevoerd.



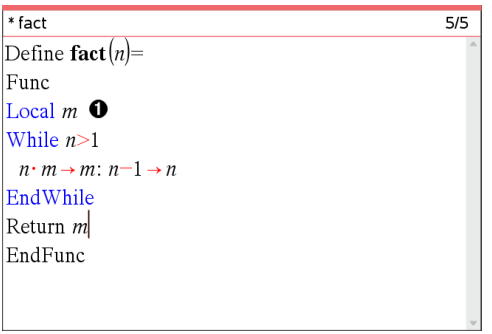
❶ Declareert variabele *i* als lokaal.

Opmerking: Declareer indien mogelijk elke variabele, die alleen binnen het programma wordt gebruikt en niet beschikbaar hoeft te zijn nadat het programma is gestopt, als lokaal.

Waardoor wordt een foutmelding m.b.t. een ongedefinieerde variabele veroorzaakt?

Een foutmelding met betrekking tot een **ongedefinieerde** variabele verschijnt als u een door de gebruiker gedefinieerde functie uitwerkt of een door de gebruiker gedefinieerd programma uitvoert dat verwijst naar een lokale variabele die niet geïnitialiseerd is (waaraan geen waarde is toegekend).

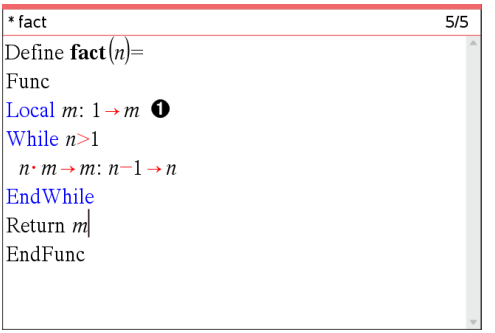
Bijvoorbeeld:



❶ Lokale variabele *m* heeft geen beginwaarde toegewezen gekregen.

Lokale variabelen initialiseren

Alle lokale variabelen moeten een beginwaarde krijgen voordat ernaar verwezen wordt.



❶ 1 is opgeslagen als de beginwaarde voor m.

Opmerking (CAS): functies en programma's kunnen geen lokale variabele gebruiken om symbolische berekeningen uit te voeren.

CAS: Symbolische berekeningen uitvoeren

Als u wilt dat een functie of programma symbolische berekeningen uitvoert, moet u een globale in plaats van een lokale variabele gebruiken. U moet er echter zeker van zijn dat de globale variabele nog niet bestaat buiten het programma. Gebruik één van de volgende methoden.

- Verwijs naar een globale variabelenaam, doorgaans met twee of meer tekens, waarvan het niet waarschijnlijk is dat deze buiten de functie of het programma bestaat.
- Neem **DelVar** binnen een programma op om de globale variabele te wissen, als deze bestaat, voordat u ernaar verwijst. (**DelVar** wist geen beveiligde of gekoppelde variabelen.)

Verschillen tussen functies en programma's

Een functie die gedefinieerd is in de Programma-editor lijkt op de functies die in de TI-Nspire™-software zijn ingebouwd.

- Functies moeten een resultaat geven, dat in een grafiek kan worden getekend of in een tabel kan worden ingevoerd. Programma's kunnen geen resultaat geven.
- U kunt een functie (maar geen programma) in een uitdrukking gebruiken. Bijvoorbeeld: **3 • func1(3)** is geldig, maar **3 • prog1(3)** niet.
- U kunt programma's alleen vanuit de Rekenmachine- en Notitie-toepassing gebruiken. U kunt echter functies uitwerken in Rekenmachine, Notities, Lijsten & Spreadsheet, Grafieken & Meetkunde en Gegevensverwerking & Statistiek.

- Een functie kan verwijzen naar elke variabele; hij kan echter alleen een waarde opslaan naar een lokale variabele. Programma's kunnen opslaan naar lokale en globale variabelen.

  Opmerking: argumenten die gebruikt worden om waarden door te geven aan een functie worden automatisch behandeld als lokale variabelen. Als u naar andere variabelen wilt opslaan, moet u deze als **Local** declareren vanuit de functie.

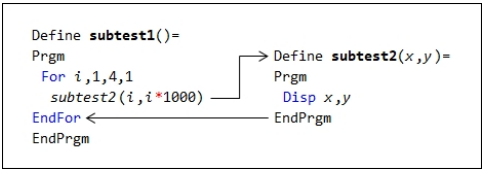
- Een functie kan geen programma oproepen als subroutine, maar kan wel een andere door de gebruiker gedefinieerde functie oproepen.
- U kunt geen programma binnen een functie definiëren.
- Een functie kan geen globale functie definiëren, maar wel een lokale functie.

Een programma vanuit een ander programma oproepen

Een programma kan een ander programma oproepen als een subroutine. De subroutine kan extern (een apart programma) of intern (opgenomen in het hoofdprogramma) zijn. Subroutines zijn handig als een programma dezelfde groep commando's op verschillende plaatsen moet herhalen.

Een apart programma oproepen

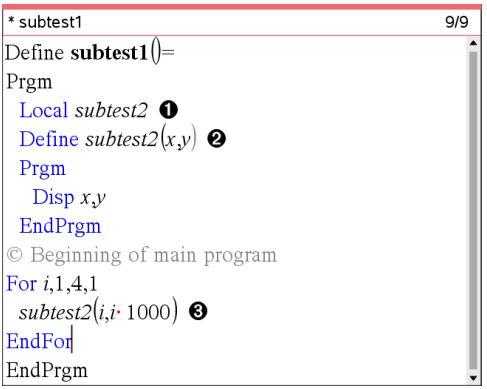
Om een apart programma op te roepen gebruikt u dezelfde syntax die u gebruikt om het programma uit te voeren vanaf de invoerregel.



Een interne subroutine definiëren en oproepen

Als u een interne subroutine wilt definiëren, gebruikt u de opdracht **Define** met **Prgm...EndPrgm**. Aangezien een subroutine gedefinieerd moet worden voordat deze kan worden opgeroepen, is het verstandig om subroutines aan het begin van het hoofdprogramma te definiëren.

Een interne subroutine wordt op dezelfde wijze opgeroepen en uitgevoerd als een apart programma.



❶ Declareert de subroutine als een lokale variabele.

❷ Definieert de subroutine.

❸ Roept de subroutine op.

Opmerking: gebruik het menu **Var** van de Programma-editor om de commando's **Define** en **Prgm...EndPrgm** in te voeren.

Opmerkingen over het gebruik van subroutines

Aan het einde van een subroutine wordt het oproepende programma verder uitgevoerd. Als u een subroutine op een ander moment wilt afsluiten, gebruikt u de opdracht **Return** zonder argument.

Een subroutine kan geen toegang krijgen tot lokale variabelen die in het oproepende programma gedeclareerd zijn. Evenmin kan het oproepende programma toegang krijgen tot lokale variabelen die in een subroutine gedeclareerd zijn.

Lbl-commando's zijn lokaal voor de programma's waarin ze zich bevinden. Daarom kan een commando **Goto** in het oproepende programma niet naar een label in een subroutine verwijzen of omgekeerd.

Cirkeldefinitiefouten voorkomen

Wanneer u een door de gebruiker gedefinieerde functie uitwerkt of een programma uitvoert, kunt u een argument specificeren dat dezelfde variabele bevat als die gebruikt is om de functie te definiëren of het programma te creëren. Om cirkeldefinitiefouten te vermijden, moet u een waarde toekennen aan variabelen die gebruikt worden bij het uitwerken van de functie of het uitvoeren van het programma. Bijvoorbeeld:

$x+1 \rightarrow x$ ❶

– of –

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

❶ Veroorzaakt een **Cirkelfdefinitie**-foutmelding als x of i geen waarde heeft. De fout treedt niet op als x of i al een waarde toegekend heeft gekregen.

Het verloop van een functie of programma besturen

Wanneer u een programma uitvoert of een functie uitwerkt, worden de programmaregels in sequentiële volgorde uitgevoerd. Bepaalde opdrachten beïnvloeden echter het programmaverloop. Bijvoorbeeld:

- Besturingsstructuren zoals **If...EndIf**-opdrachten gebruiken een voorwaardelijke test om te beslissen welk deel van een programma wordt uitgevoerd.
- Lus-opdrachten zoals **For...EndFor** herhalen een groep opdrachten.

If, Lbl en Goto gebruiken om het verloop van het programma te besturen

Met het commando **If** en verschillende **If...EndIf**-structuren kunt u een bewering of een blok beweringen voorwaardelijk uitvoeren, dat wil zeggen, gebaseerd op het resultaat van een test (zoals x>5). **Met de commando's Lbl** (label) en **Goto** kunt u vertakken of van de ene plaats naar de andere springen in een functie of programma.

Het **If**-commando en verschillende **If...EndIf**-structuren staan in het menu **Besturing** van de Programma-editor.

Wanneer u een structuur als **If...Then...EndIf** invoert, wordt er een template ingevoerd op de plaats van de cursor. De cursor wordt zo geplaatst dat u een voorwaardelijke test kunt invoeren.

De opdracht If

Om een enkel commando uit te voeren als de voorwaardelijke test waar is, gebruikt u de algemene vorm:

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

❶ Wordt alleen uitgevoerd als x>5; wordt anders overgeslagen.

❷ Geeft altijd de waarde van x weer.

In dit voorbeeld moet u een waarde aan x toewijzen voordat de opdracht **If** wordt uitgevoerd.

If...Then...EndIf-structuren

Als u één groep opdrachten wilt uitvoeren als een voorwaardelijke test waar is, gebruikt u de volgende structuur:

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

❶ Wordt alleen uitgevoerd als x>5.

❷ Geeft de waarde weer van:
2x if x>5
x if x≤5

Opmerking: EndIf markeert het einde van het **Then**-blok dat wordt uitgevoerd als de voorwaarde waar is.

If...Then...Else... EndIf-structuren

Als u één groep opdrachten wilt uitvoeren als een voorwaardelijke test waar is en een andere groep als de voorwaarde onwaar is, gebruikt u de volgende structuur:

```
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 Wordt alleen uitgevoerd als x>5.

❷ Wordt alleen uitgevoerd als≤5.

❸ Geeft de waarde van:
2x if x>5
5x if x≤5

If...Then...ElseIf... EndIf-structuren

Met een meer complexe vorm van de opdracht If kunt u meerdere voorwaarden testen. Stel dat u een programma wilt maken waarmee u een door de gebruiker ingevoerd argument wilt testen, dat één van vier opties aanduidt.

Om elke optie te testen (If Keuze=1, If Keuze=2 enz.) gebruikt u de **If...Then...ElseIf...EndIf**-structuur.

De opdrachten Lbl en Goto

U kunt het verloop ook besturen door de opdrachten **Lbl** (label) en **Goto** te gebruiken. Deze commando's staan in het menu **Overzendingen** van de Programma-editor.

Gebruik de opdracht **Lbl** als u een naam aan een specifieke lokatie in de functie of het programma wilt toewijzen.

Lbl *labelName*	naam die wordt toegekend aan deze locatie (gebruik dezelfde naamgevingsregels als voor een variabelenaam)

Vervolgens kunt u op elke gewenste positie in de functie of in het programma de opdracht **Goto** gebruiken om te springen naar de lokatie die overeenkomt met het opgegeven label.

Goto *labelName*	specificeert naar welke **Lbl**-opdracht afgetakt moet worden

Aangezien de opdracht **Goto** onvoorwaardelijk is (er wordt altijd naar het opgegeven label gesprongen), wordt deze opdracht vaak met de opdracht **If** gebruikt, zodat u een voorwaardelijke test kunt opgeven. Bijvoorbeeld:

```
If x>5
  Goto GT5 ❶
Disp x

.....

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

❶ Als x>5 wordt rechtstreeks naar label GT5 gesprongen.

❷ In dit voorbeeld moet het programma opdrachten bevatten (zoals **Stop**) die voorkomen dat **Lbl** GT5 wordt uitgevoerd als x≤5.

Lussen gebruiken om een groep opdrachten te herhalen

Als u dezelfde groep opdrachten wilt herhalen, gebruikt u een van de lusstructuren. Er zijn verschillende typen lussen beschikbaar. Elke type geeft u een andere manier om de lus te verlaten, op basis van een voorwaardelijke test.

Lussen en lusgerelateerde commando's staan in de menu's **Besturing** en **Overzendingen** van de Programma-editor.

Wanneer u een van de lusstructuren invoegt, wordt de bijbehorende template op de plaats van de cursor ingevoegd. Vervolgens kunt u beginnen met het invoeren van de commando's die in de lus moeten worden uitgevoerd.

For...EndFor-lussen

Een **For...EndFor**-lus gebruikt een teller om het aantal keren dat de lus herhaald wordt te besturen. De syntax van de opdracht **For** is:

Opmerking: de beginwaarde kan kleiner zijn dan de eindwaarde, mits de stap negatief is.

For *variabele, begin, eind* [, *stap*]
❶ ❷ ❸ ❹

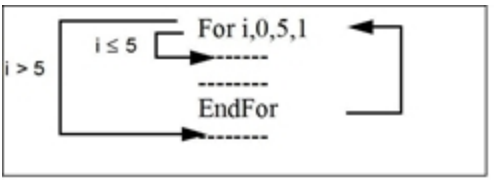
❶ *Variabele* gebruikt als een teller

❷ Tellerwaarde die gebruikt wordt de eerste keer dat **For** wordt uitgevoerd

❸ Verlaat de lus wanneer *variabele* deze waarde overschrijdt

❹ Wordt opgeteld bij de teller, iedere volgende keer dat **For** wordt uitgevoerd (als deze optionele waarde wordt weggelaten, is de *stap* 1.)

Als **For** wordt uitgevoerd, wordt de *variabele*-waarde vergeleken met de *eind*-waarde. Als *variabele* niet groter is dan *eind* wordt de lus uitgevoerd; in andere gevallen springt de besturing naar de opdracht na **EndFor**.



Opmerking: de opdracht **For** verhoogt de tellervariabele automatisch, zodat de functie of het programma de lus na een bepaald aantal herhalingen kan verlaten.

Aan het einde van de lus (**EndFor**) springt de besturing terug naar de opdracht **For** en wordt de variabele verhoogd en met *eind* vergeleken.

Bijvoorbeeld:

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

❶ Geeft 0, 1, 2, 3, 4 en 5 weer.

❷ Geeft 6 weer. Als *variabele* wordt verhoogd naar 6, wordt de lus niet uitgevoerd.

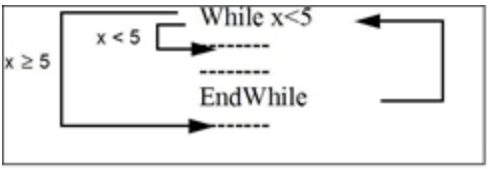
Opmerking: u kunt de tellervariabele tot lokaal verklaren als deze niet hoeft te worden opgeslagen nadat de functie of het programma is gestopt.

While...EndWhile-lussen

Een **While...EndWhile**-lus herhaalt een blok opdrachten zolang een opgegeven voorwaarde waar is. De syntax van de opdracht **While** is:

While -*voorwaarde*

Als **While** wordt uitgevoerd, wordt de *voorwaarde* geëvalueerd. Als *voorwaarde* waar is wordt de lus uitgevoerd; in het andere geval springt de besturing naar de opdracht na **EndWhile**.



Opmerking: de opdracht **While** verandert de voorwaarde niet automatisch. U moet opdrachten opnemen die de functie of het programma in staat stellen de lus te verlaten.

Aan het einde van de lus (**EndWhile**) springt de besturing terug naar de opdracht **While** en wordt de voorwaarde opnieuw geëvalueerd.

Om de lus voor de eerste keer uit te voeren, moet de voorwaarde aanvankelijk waar zijn.

- Variabelen waarnaar in de voorwaarde wordt verwezen, moeten vóór de opdracht **While** worden geplaatst. (U kunt de waarden in de functie of in het programma inbouwen, of u kunt de gebruiker verzoeken om de waarden in te voeren).
- De lus moet opdrachten bevatten die de waarden in de voorwaarde veranderen, zodat de voorwaarde uiteindelijk onwaar wordt. Anders is de voorwaarde altijd waar en kan de functie of het programma de lus niet verlaten (dit wordt een oneindige lus genoemd).

Bijvoorbeeld:

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

❶ Stelt de beginwaarde van x in.

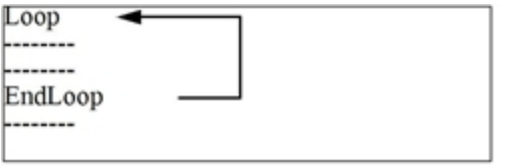
❷ Geeft 0, 1, 2, 3 en 4 weer.

❸ Verhoogt x.

❹ Geeft 5 weer. Wanneer x wordt verhoogd tot 5 wordt de lus niet uitgevoerd.

Loop...EndLoop-lussen

Loop...EndLoop maakt een oneindige lus, die eindeloos herhaald wordt. De opdracht **Loop** heeft geen argumenten.



Doorgaans voert u opdrachten in de lus in die het programma in staat stellen om de lus te verlaten. Veelgebruikte opdrachten zijn: **If**, **Exit**, **Goto** en **Lbl** (label). Bijvoorbeeld:

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

❶ Een **If**-opdracht controleert de voorwaarde.

❷ Verlaat de lus en springt hiernaartoe als x wordt verhoogd naar 6.

Opmerking: met de opdracht **Exit** wordt de huidige lus verlaten.

In dit voorbeeld kan de opdracht **If** zich overal in de lus bevinden.

Als de opdracht If staat:	**De lus wordt:**
aan het begin van de lus	alleen uitgevoerd als de voorwaarde waar is.
aan het einde van de lus	ten minste één maal uitgevoerd en wordt alleen herhaald als de voorwaarde waar is.

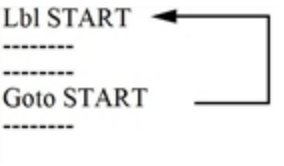
De opdracht **If** zou ook een opdracht **Goto** kunnen gebruiken om de besturing van het programma over te zetten naar een opgegeven **Lbl** (label)-opdracht.

Een lus onmiddellijk herhalen

De opdracht **Cycle** brengt de programmabesturing onmiddellijk over naar de volgende iteratie van een lus (voordat de huidige iteratie voltooid is). Deze opdracht werkt met **For...EndFor**, **While...EndWhile** en **Loop...EndLoop**.

Lbl- en Goto-lussen

Hoewel de opdrachten **Lbl** (label) en **Goto** strikt genomen geen lus-opdrachten zijn, kunnen deze wel gebruikt worden om een oneindige lus te maken. Bijvoorbeeld:



Net als bij **Loop...EndLoop** moet de lus opdrachten bevatten die de functie of het programma in staat stellen om de lus te verlaten.

Modusinstellingen veranderen

Functies en programma's kunnen de functie **setMode()** gebruiken om tijdelijk specifieke berekenings- of resultaatmodi in te stellen. Het menu **Modus** van de Programma-editor maakt het makkelijk om de juiste syntax in te voeren, zonder dat u numerieke codes hoeft te onthouden.

Opmerking: modusveranderingen die binnen een functie- of programmadefinitie zijn gemaakt zijn niet van kracht buiten de functie of het programma.

Een modus instellen

1. Zet de cursor op de plaats waar u de functie **setMode** wilt invoegen.
2. Selecteer in het menu **Modus** de modus die u wilt veranderen, en selecteer de nieuwe instelling.

   De juiste syntax wordt ingevoegd op de plaats van de cursor. Bijvoorbeeld:

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

Fouten in programma's opsporen en fouten afhandelen

Nadat u een functie of programma hebt geschreven, kunt u verschillende technieken gebruiken om fouten op te sporen en te corrigeren. U kunt ook een foutafhandelingsopdracht in de functie of het programma zelf inbouwen.

Als uw functie of programma de gebruiker in staat stelt te kiezen uit meerdere opties, voer het programma dan uit en test elke optie.

Technieken voor het opsporen van fouten

Run-time foutmeldingen kunnen syntaxfouten aangeven, maar niet fouten in de programmalogica. De volgende technieken kunnen van nut zijn.

- Voeg tijdelijk **Disp**-opdrachten in om de waarden van cruciale variabelen weer te geven.
- Als u na wilt gaan of een lus het juiste aantal maal wordt uitgevoerd, gebruikt u **Disp** om de tellervariabele of de waarden in de voorwaardelijke test weer te geven.
- Als u na wilt gaan of een subroutine wordt uitgevoerd, gebruikt u **Disp** om meldingen weer te geven als “Begin subroutine” en “Einde subroutine” aan het begin en einde van de subroutine.
- Een programma of functie handmatig stoppen:
  - **Windows®:** Houd de toets **F12** ingedrukt en druk enkele malen op **Enter**.
  - **Macintosh®:** Houd de toets **F5** ingedrukt en druk enkele malen op **Enter**.
  - **Rekenmachine:** Houd de toets [on] ingedrukt en druk enkele malen op [enter].

Foutafhandelingsopdrachten

Commando	Beschrijving
Try...EndTry	Definieert een programmablok dat een functie of een programma een opdracht laat uitvoeren en zonodig laat herstellen van een fout, veroorzaakt door die opdracht.
ClrErr	Wist de foutstatus en zet de systeemvariabele *errCode op nul. Zie voor een voorbeeld van het gebruik van errCode de opdracht* ***Try in de Referentiehandleiding***.
PassErr	Geeft een fout door aan het volgende niveau van het **Try...EndTry**-blok.

Algemene informatie

Online Help

education.ti.com/eguide

Selecteer uw land voor meer productinformatie.

Neem contact op met TI Ondersteuning

education.ti.com/ti-cares

Selecteer uw land voor technische en andere ondersteuningsbronnen.

Service- en garantie-informatie

education.ti.com/warranty

Selecteer uw land voor meer informatie over de duur en voorwaarden van de garantie of over de productservice.

Beperkte garantie. Deze garantie heeft geen invloed op uw wettelijke rechten.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243