

Masterclass programmeren op de GR TI-84 (les 4)

Kevin van As

April 11, 2016

Recap!

We hebben gekeken naar:

- Herhaling / Repetitie
 - Goto & Labels
 - For-loops
 - While-loops
- Pause
- Datatype: Lists

Recap!

We hebben gekeken naar:

- Herhaling / Repetitie
 - Goto & Labels
 - For-loops
 - While-loops
- Pause
- Datatype: Lists

Recap!

We hebben gekeken naar:

- Herhaling / Repetitie
 - Goto & Labels
 - For-loops
 - While-loops
- Pause
- Datatype: Lists

Recap!

We hebben gekeken naar:

- Herhaling / Repetitie
 - Goto & Labels
 - For-loops
 - While-loops
- Pause
- Datatype: Lists

Recap!

We hebben gekeken naar:

- Herhaling / Repetitie
 - Goto & Labels
 - For-loops
 - While-loops
- Pause
- Datatype: Lists

Recap!

We hebben gekeken naar:

- Herhaling / Repetitie
 - Goto & Labels
 - For-loops
 - While-loops
- Pause
- Datatype: Lists

Recap!

We hebben gekeken naar:

- Herhaling / Repetitie
 - Goto & Labels
 - For-loops
 - While-loops
- Pause
- Datatype: Lists

Vooruitzicht!

Vandaag zullen we kijken naar:

- Advanced I/O:
 - Menu
 - Output
 - ClrHome
- Custom functions
- Tips 'n Tricks:
 - `[CLEAR]` om een menu te sluiten
 - `[ALPHA]`-scrolling
 - `[CATALOG]`
 - Debugging
 - Afrondingsfouten

Vooruitzicht!

Vandaag zullen we kijken naar:

- Advanced I/O:
 - Menu
 - Output
 - ClrHome
- Custom functions
- Tips 'n Tricks:
 - `CLEAR` om een menu te sluiten
 - `ALPHA`-scrolling
 - `CATALOG`
 - Debugging
 - Afrondingsfouten

Vooruitzicht!

Vandaag zullen we kijken naar:

- Advanced I/O:
 - Menu
 - Output
 - ClrHome
- Custom functions
- Tips 'n Tricks:
 - `[CLEAR]` om een menu te sluiten
 - `[ALPHA]`-scrolling
 - `[CATALOG]`
 - Debugging
 - Afrondingsfouten

Vooruitzicht!

Vandaag zullen we kijken naar:

- Advanced I/O:
 - Menu
 - Output
 - ClrHome
- Custom functions
- Tips 'n Tricks:
 - `CLEAR` om een menu te sluiten
 - `ALPHA`-scrolling
 - `CATALOG`
 - Debugging
 - Afrondingsfouten

Vooruitzicht!

Vandaag zullen we kijken naar:

- Advanced I/O:
 - Menu
 - Output
 - ClrHome
- Custom functions
- Tips 'n Tricks:
 - `[CLEAR]` om een menu te sluiten
 - `[ALPHA]`-scrolling
 - `[CATALOG]`
 - Debugging
 - Afrondingsfouten

Vooruitzicht!

Vandaag zullen we kijken naar:

- Advanced I/O:
 - Menu
 - Output
 - ClrHome
- Custom functions
- Tips 'n Tricks:
 - `[CLEAR]` om een menu te sluiten
 - `[ALPHA]`-scrolling
 - `[CATALOG]`
 - Debugging
 - Afrondingsfouten

Vooruitzicht!

Vandaag zullen we kijken naar:

- Advanced I/O:
 - Menu
 - Output
 - ClrHome
- Custom functions
- Tips 'n Tricks:
 - `CLEAR` om een menu te sluiten
 - `ALPHA`-scrolling
 - `CATALOG`
 - Debugging
 - Afrondingsfouten

Vooruitzicht!

Vandaag zullen we kijken naar:

- Advanced I/O:
 - Menu
 - Output
 - ClrHome
- Custom functions
- Tips 'n Tricks:
 - `CLEAR` om een menu te sluiten
 - `ALPHA`-scrolling
 - `CATALOG`
 - Debugging
 - Afrondingsfouten

Vooruitzicht!

Vandaag zullen we kijken naar:

- Advanced I/O:
 - Menu
 - Output
 - ClrHome
- Custom functions
- Tips 'n Tricks:
 - `CLEAR` om een menu te sluiten
 - `ALPHA`-scrolling
 - `CATALOG`
 - Debugging
 - Afrondingsfouten

Vooruitzicht!

Vandaag zullen we kijken naar:

- Advanced I/O:
 - Menu
 - Output
 - ClrHome
- Custom functions
- Tips 'n Tricks:
 - `CLEAR` om een menu te sluiten
 - `ALPHA`-scrolling
 - `CATALOG`
 - Debugging
 - Afrondingsfouten

Vooruitzicht!

Vandaag zullen we kijken naar:

- Advanced I/O:
 - Menu
 - Output
 - ClrHome
- Custom functions
- Tips 'n Tricks:
 - `CLEAR` om een menu te sluiten
 - `ALPHA`-scrolling
 - `CATALOG`
 - Debugging
 - Afrondingsfouten

Vooruitzicht!

Vandaag zullen we kijken naar:

- Advanced I/O:
 - Menu
 - Output
 - ClrHome
- Custom functions
- Tips 'n Tricks:
 - `CLEAR` om een menu te sluiten
 - `ALPHA`-scrolling
 - `CATALOG`
 - Debugging
 - Afrondingsfouten

Outline

- 1 Tips 'n Tricks
 - Random tips
 - Debugging
 - Catalog & CatalogHelp
 - Precisie en afrondingsfouten
- 2 Functions
 - Theorie
 - Voorbeelden
- 3 Advanced I/O
 - Menu
 - Output
 - num2str
- 4 Exercises
 - Exercises
 - Answers

Outline

- 1 Tips 'n Tricks
 - Random tips
 - Debugging
 - Catalog & CatalogHelp
 - Precisie en afrondingsfouten
- 2 Functions
 - Theorie
 - Voorbeelden
- 3 Advanced I/O
 - Menu
 - Output
 - num2str
- 4 Exercises
 - Exercises
 - Answers

Trial 'n Error

- Soms is het lastig om te beredeneren of er nu $A+1 \rightarrow A$ moet staan, of juist $A-1 \rightarrow A$.
- Of is de juiste initialisatie nou $-1 \rightarrow A$ of $0 \rightarrow A$?
- In dit soort gevallen kan de “Trial 'n Error” tactiek nuttig zijn:
 - “Probeer gewoon wat, en zie wat er gebeurt!”
 - Weét wat de uitkomst van je `PRGM` moet zijn, en corrigeer de code indien het resultaat niet klopt.
- Let wel: Test genoeg cases om zeker te weten dat je `PRGM` nu klopt! Immers, je hebt je code ‘gegokt’, niet bedacht.

Trial 'n Error

- Soms is het lastig om te beredeneren of er nu $A+1 \rightarrow A$ moet staan, of juist $A-1 \rightarrow A$.
- Of is de juiste initialisatie nou $-1 \rightarrow A$ of $0 \rightarrow A$?
- In dit soort gevallen kan de “Trial 'n Error” tactiek nuttig zijn:
 - “Probeer gewoon wat, en zie wat er gebeurt!”
 - Weét wat de uitkomst van je `prog` moet zijn, en corrigeer de code indien het resultaat niet klopt.
- Let wel: Test genoeg cases om zeker te weten dat je `prog` nu klopt! Immers, je hebt je code ‘gegokt’, niet bedacht.

Trial 'n Error

- Soms is het lastig om te beredeneren of er nu $A+1 \rightarrow A$ moet staan, of juist $A-1 \rightarrow A$.
- Of is de juiste initialisatie nou $-1 \rightarrow A$ of $0 \rightarrow A$?
- In dit soort gevallen kan de “Trial 'n Error” tactiek nuttig zijn:
 - “Probeer gewoon wat, en zie wat er gebeurt!”
 - Wéét wat de uitkomst van je `PRGM` moet zijn, en corrigeer de code indien het resultaat niet klopt.
- Let wel: Test genoeg cases om zeker te weten dat je `PRGM` nu klopt! Immers, je hebt je code ‘gegokt’, niet bedacht.

Trial 'n Error

- Soms is het lastig om te beredeneren of er nu $A+1 \rightarrow A$ moet staan, of juist $A-1 \rightarrow A$.
- Of is de juiste initialisatie nou $-1 \rightarrow A$ of $0 \rightarrow A$?
- In dit soort gevallen kan de “Trial 'n Error” tactiek nuttig zijn:
 - “Probeer gewoon wat, en zie wat er gebeurt!”
 - Wéét wat de uitkomst van je `PRGM` moet zijn, en corrigeer de code indien het resultaat niet klopt.
- Let wel: Test genoeg cases om zeker te weten dat je `PRGM` nu klopt! Immers, je hebt je code ‘gegokt’, niet bedacht.

Trial 'n Error

- Soms is het lastig om te beredeneren of er nu $A+1 \rightarrow A$ moet staan, of juist $A-1 \rightarrow A$.
- Of is de juiste initialisatie nou $-1 \rightarrow A$ of $0 \rightarrow A$?
- In dit soort gevallen kan de “Trial 'n Error” tactiek nuttig zijn:
 - “Probeer gewoon wat, en zie wat er gebeurt!”
 - Wéét wat de uitkomst van je `prog` moet zijn, en corrigeer de code indien het resultaat niet klopt.
- Let wel: Test genoeg cases om zeker te weten dat je `prog` nu klopt! Immers, je hebt je code ‘gegokt’, niet bedacht.

Trial 'n Error

- Soms is het lastig om te beredeneren of er nu $A+1 \rightarrow A$ moet staan, of juist $A-1 \rightarrow A$.
- Of is de juiste initialisatie nou $-1 \rightarrow A$ of $0 \rightarrow A$?
- In dit soort gevallen kan de “Trial 'n Error” tactiek nuttig zijn:
 - “Probeer gewoon wat, en zie wat er gebeurt!”
 - Wéét wat de uitkomst van je `program` moet zijn, en corrigeer de code indien het resultaat niet klopt.
- Let wel: Test genoeg cases om zeker te weten dat je `program` nu klopt! Immers, je hebt je code ‘gegokt’, niet bedacht.

CLEAR om een menu te sluiten

- Ik heb altijd de neiging om **2nd****MODE**=**[QUIT]** te gebruiken om een menu te sluiten.
- Klinkt logisch, maar indien je dit doet in een **PRGM**, dan sluit je zowel het menu als het **PRGM** en moet je je **PRGM** weer overnieuw openen...
- De correcte manier om een menu te sluiten is daarom dus ook de **CLEAR**-knop. Dit sluit elk menu, maar sluit niet je **PRGM**.
- Let wel: clear niet per ongeluk een regel van je **PRGM** door 2x te drukken op **CLEAR**.



CLEAR om een menu te sluiten

- Ik heb altijd de neiging om `2nd``MODE``=``[QUIT]` te gebruiken om een menu te sluiten.
- Klinkt logisch, maar indien je dit doet in een `PRGM`, dan sluit je zowel het menu als het `PRGM` en moet je je `PRGM` weer overnieuw openen...
- De correcte manier om een menu te sluiten is daarom dus ook de `CLEAR`-knop. Dit sluit elk menu, maar sluit niet je `PRGM`.
- Let wel: clear niet per ongeluk een regel van je `PRGM` door 2x te drukken op `CLEAR`.



CLEAR om een menu te sluiten

- Ik heb altijd de neiging om `2nd``MODE``=``[QUIT]` te gebruiken om een menu te sluiten.
- Klinkt logisch, maar indien je dit doet in een `PRGM`, dan sluit je zowel het menu als het `PRGM` en moet je je `PRGM` weer overnieuw openen...
- De correcte manier om een menu te sluiten is daarom dus ook de `CLEAR`-knop. Dit sluit elk menu, maar sluit niet je `PRGM`.
- Let wel: clear niet per ongeluk een regel van je `PRGM` door 2x te drukken op `CLEAR`.





CLEAR om een menu te sluiten

- Ik heb altijd de neiging om **2nd****MODE**=**[QUIT]** te gebruiken om een menu te sluiten.
- Klinkt logisch, maar indien je dit doet in een **PRGM**, dan sluit je zowel het menu als het **PRGM** en moet je je **PRGM** weer overnieuw openen...
- De correcte manier om een menu te sluiten is daarom dus ook de **CLEAR**-knop. Dit sluit elk menu, maar sluit niet je **PRGM**.
- Let wel: clear niet per ongeluk een regel van je **PRGM** door 2x te drukken op **CLEAR**.





ALPHA-scrolling

- Naarmate je **PRGM** langer wordt, wordt het vervelender om door je **PRGM** te navigeren.
- Door **ALPHA**  te gebruiken i.p.v. , scroll je significant sneller.
- Met behulp van **2nd****ALPHA**=**[A-LOCK]**, kun je herhaaldelijk snel scrollen. Try it yourself!





ALPHA-scrolling

- Naarmate je **PRGM** langer wordt, wordt het vervelender om door je **PRGM** te navigeren.
- Door **ALPHA**  te gebruiken i.p.v. , scroll je significant sneller.
- Met behulp van **2nd****ALPHA**=**[A-LOCK]**, kun je herhaaldelijk snel scrollen. Try it yourself!



ALPHA-scrolling

- Naarmate je **PRGM** langer wordt, wordt het vervelender om door je **PRGM** te navigeren.
- Door **ALPHA**  te gebruiken i.p.v. , scroll je significant sneller.
- Met behulp van **2nd****ALPHA**=**[A-LOCK]**, kun je herhaaldelijk snel scrollen. Try it yourself!



Meet your new best friend: Google

- Whenever you run into a problem, someone else has most certainly had the same problem.
- Therefore, your problem has already been solved before.
- So, when you want to do something, but you do not know how to do it, you ask your best friend.
- Note: The internet is English (like this slide). So use English search terms for the best result.

Meet your new best friend: Google

- Whenever you run into a problem, someone else has most certainly had the same problem.
- Therefore, your problem has already been solved before.
- So, when you want to do something, but you do not know how to do it, you ask your best friend.
- Note: The internet is English (like this slide). So use English search terms for the best result.

Meet your new best friend: Google

- Whenever you run into a problem, someone else has most certainly had the same problem.
- Therefore, your problem has already been solved before.
- So, when you want to do something, but you do not know how to do it, you ask your best friend.
- Note: The internet is English (like this slide). So use English search terms for the best result.

Meet your new best friend: Google

- Whenever you run into a problem, someone else has most certainly had the same problem.
- Therefore, your problem has already been solved before.
- So, when you want to do something, but you do not know how to do it, you ask your best friend.
- Note: The internet is English (like this slide). So use English search terms for the best result.

Vooral veel commentaar leveren

- Het is handig om “comments” in je code achter te laten.
- Dit zijn kleine notes die uitleggen wat je code doet.
- Indien je later weer naar je code kijkt, helpen die je begrijpen wat je code doet.
- Dit kun je met behulp van een string doen: `"COMMENT"`

Vooral veel commentaar leveren

- Het is handig om “comments” in je code achter te laten.
- Dit zijn kleine notes die uitleggen wat je code doet.
- Indien je later weer naar je code kijkt, helpen die je begrijpen wat je code doet.
- Dit kun je met behulp van een string doen: `"COMMENT"`

Voor al veel commentaar leveren

- Het is handig om “comments” in je code achter te laten.
- Dit zijn kleine notes die uitleggen wat je code doet.
- Indien je later weer naar je code kijkt, helpen die je begrijpen wat je code doet.
- Dit kun je met behulp van een string doen: `"COMMENT"`

Vooral veel commentaar leveren

- Het is handig om “comments” in je code achter te laten.
- Dit zijn kleine notes die uitleggen wat je code doet.
- Indien je later weer naar je code kijkt, helpen die je begrijpen wat je code doet.
- Dit kun je met behulp van een string doen: `"COMMENT"`

```
PROGRAM:ABC
: "SOLVE AX2+BX+C=0
: Prompt A,B,C
: B2-4AC→D
: Disp (-B+√(D))/(2A)
: Disp (-B-√(D))/(2A)
```

Outline

1 Tips 'n Tricks

- Random tips
- **Debugging**
- Catalog & CatalogHelp
- Precisie en afrondingsfouten

2 Functions

- Theorie
- Voorbeelden

3 Advanced I/O

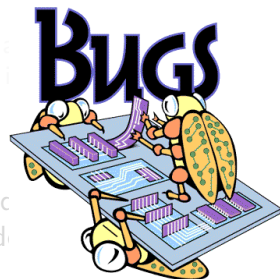
- Menu
- Output
- num2str

4 Exercises

- Exercises
- Answers

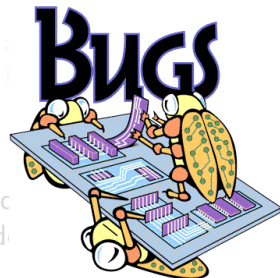
Wat is “debugging”?

- Wat is een ‘bug’?
 - Wikipedia: “A software bug is an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.”
 - Oftewel, je `program` doet niet wat je wilt.
- Waar komen die beesten vandaan?
 - Wikipedia: “Most bugs arise from mistakes as people in either a program's source code or in the hardware.”
 - Oftewel, een programmeer- of denkfout.
- Debugging is “het verwijderen van bugs”
- Indien je de ‘Trial 'n Error’ tactiek gebruikt, creëer je opzet bugs om ze vervolgens gelijk te verwijderen.



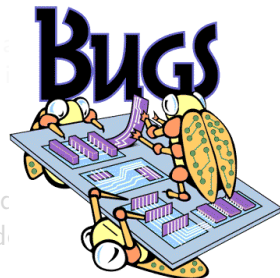
Wat is “debugging”?

- Wat is een ‘bug’?
 - Wikipedia: “A software bug is an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.”
 - Oftewel, je `prog` doet niet wat je wilt.
- Waar komen die beesten vandaan?
 - Wikipedia: “Most bugs arise from mistakes as people in either a program’s source code or in the hardware.”
 - Oftewel, een programmeer- of denkfout.
- Debugging is “het verwijderen van bugs”
- Indien je de ‘Trial ’n Error’ tactiek gebruikt, creëer je opzet bugs om ze vervolgens gelijk te verwijderen.



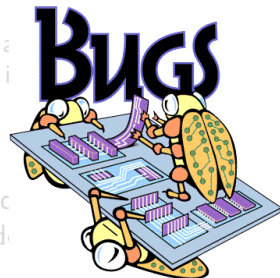
Wat is “debugging”?

- Wat is een ‘bug’?
 - Wikipedia: “A software bug is an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.”
 - Oftewel, je **prog** doet niet wat je wilt.
- Waar komen die beesten vandaan?
 - Wikipedia: “Most bugs arise from mistakes : people in either a program's source code or i
 - Oftewel, een programmeer- of denkfout.
- Debugging is “het verwijderen van bugs”
- Indien je de ‘Trial 'n Error’ tactiek gebruikt, c opzet bugs om ze vervolgens gelijk te verwijd



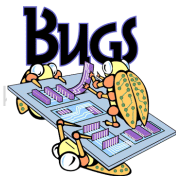
Wat is “debugging”?

- Wat is een ‘bug’?
 - Wikipedia: “A software bug is an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.”
 - Oftewel, je **prog** doet niet wat je wilt.
- Waar komen die beesten vandaan?
 - Wikipedia: “Most bugs arise from mistakes as people in either a program’s source code or in the program’s design.”
 - Oftewel, een programmeer- of denkfout.
- Debugging is “het verwijderen van bugs”
- Indien je de ‘Trial ’n Error’ tactiek gebruikt, creëer je opzet bugs om ze vervolgens gelijk te verwijderen.



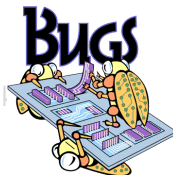
Wat is “debugging”?

- Wat is een ‘bug’?
 - Wikipedia: “A software bug is an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.”
 - Oftewel, je ~~prog~~ doet niet wat je wilt.
- Waar komen die beesten vandaan?
 - Wikipedia: “Most bugs arise from mistakes and errors made by people in either a program’s source code or its design.”
 - Oftewel, een programmeer- of denkfout.
- Debugging is “het verwijderen van bugs”
- Indien je de ‘Trial ’n Error’ tactiek gebruikt, dan maak je opzet bugs om ze vervolgens gelijk te verwijderen.



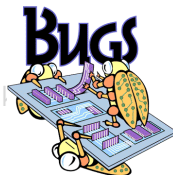
Wat is “debugging”?

- Wat is een ‘bug’?
 - Wikipedia: “A software bug is an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.”
 - Oftewel, je **PRogram** doet niet wat je wilt.
- Waar komen die beesten vandaan?
 - Wikipedia: “Most bugs arise from mistakes and errors made by people in either a program’s source code or its design.”
 - Oftewel, een programmeer- of denkfout.
- Debugging is “het verwijderen van bugs”
- Indien je de ‘Trial ’n Error’ tactiek gebruikt, dan maak je opzet bugs om ze vervolgens gelijk te verwijderen.



Wat is “debugging”?

- Wat is een ‘bug’?
 - Wikipedia: “A software bug is an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.”
 - Oftewel, je **PRogram** doet niet wat je wilt.
- Waar komen die beesten vandaan?
 - Wikipedia: “Most bugs arise from mistakes and errors made by people in either a program’s source code or its design.”
 - Oftewel, een programmeer- of denkfout.
- Debugging is “het verwijderen van bugs”
- Indien je de ‘Trial ’n Error’ tactiek gebruikt, dan maak je opzet bugs om ze vervolgens gelijk te verwijderen.



Wat is “debugging”?

- Wat is een ‘bug’?
 - Wikipedia: “A software bug is an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.”
 - Oftewel, je ~~prog~~ doet niet wat je wilt.
- Waar komen die beesten vandaan?
 - Wikipedia: “Most bugs arise from mistakes and errors made by people in either a program’s source code or its design.”
 - Oftewel, een programmeer- of denkfout.
- Debugging is “het verwijderen van bugs”
- Indien je de ‘Trial 'n Error’ tactiek gebruikt, dan maak je met opzet bugs om ze vervolgens gelijk te verwijderen.

Hoe kun je debuggen?

- Test je `PRGM` vaak, en negeer onverwachte resultaten niet: als iets één keer gebeurt, dan zal het vaker gebeuren.
- Traceer stap voor stap:
 - welke input waardes tot het probleem leiden → test cases.
 - waar in je `PRGM` alles nog klopt (zie volgende slide), en waarna er iets mis gaat = onverwachte waardes.
- Het helpt om het `PRGM` zelf (in je hoofd - brain power) uit te voeren. Doet je rekenmachine hetzelfde als jij met de hand doet? Zo niet, dan werkt een functie anders dan je dacht.



Hoe kun je debuggen?

- Test je `PROGRAM` vaak, en negeer onverwachte resultaten niet: als iets één keer gebeurt, dan zal het vaker gebeuren.
- Traceer stap voor stap:
 - welke input waardes tot het probleem leiden → test cases.
 - waar in je `PROGRAM` alles nog klopt (zie volgende slide), en waarna er iets mis gaat = onverwachte waardes.
- Het helpt om het `PROGRAM` zelf (in je hoofd - brain power) uit te voeren. Doet je rekenmachine hetzelfde als jij met de hand doet? Zo niet, dan werkt een functie anders dan je dacht.



Hoe kun je debuggen?

- Test je `PRGM` vaak, en negeer onverwachte resultaten niet: als iets één keer gebeurt, dan zal het vaker gebeuren.
- Traceer stap voor stap:
 - welke input waardes tot het probleem leiden → test cases.
 - waar in je `PRGM` alles nog klopt (zie volgende slide), en waarna er iets mis gaat = onverwachte waardes.
- Het helpt om het `PRGM` zelf (in je hoofd - brain power) uit te voeren. Doet je rekenmachine hetzelfde als jij met de hand doet? Zo niet, dan werkt een functie anders dan je dacht.



Hoe kun je debuggen?

- Test je `PROGRAM` vaak, en negeer onverwachte resultaten niet: als iets één keer gebeurt, dan zal het vaker gebeuren.
- Traceer stap voor stap:
 - welke input waardes tot het probleem leiden → test cases.
 - waar in je `PROGRAM` alles nog klopt (zie volgende slide), en waarna er iets mis gaat = onverwachte waardes.
- Het helpt om het `PROGRAM` zelf (in je hoofd - brain power) uit te voeren. Doet je rekenmachine hetzelfde als jij met de hand doet? Zo niet, dan werkt een functie anders dan je dacht.



Hoe kun je debuggen?

- Test je `PRGM` vaak, en negeer onverwachte resultaten niet: als iets één keer gebeurt, dan zal het vaker gebeuren.
- Traceer stap voor stap:
 - welke input waardes tot het probleem leiden → test cases.
 - waar in je `PRGM` alles nog klopt (zie volgende slide), en waarna er iets mis gaat = onverwachte waardes.
- Het helpt om het `PRGM` zelf (in je hoofd - brain power) uit te voeren. Doet je rekenmachine hetzelfde als jij met de hand doet? Zo niet, dan werkt een functie anders dan je dacht.



Pause statement: Localiseer de bug

- Om een bug te localiseren (=vinden waar in je `PRGM` de fout zit), is het handig om `Pause` te gebruiken.
- Zo kun je je `PRGM` breaken (=middenin zijn executie stoppen) waar je wilt.
- Dit doe je door op de `[ON]`-knop te drukken terwijl je `PRGM` gepauseerd is.
- Kies dan `Quit..` (`Goto` kan ook nuttig zijn - probeer het zelf.)
- Nu kun je alle variabelen bekijken. Is de waarde wat je verwacht had? Zo niet? Dan moet de bug *boven* je `Pause` statement zitten!

```
PROGRAM: PAUSETST  
:0→A  
:Pause  
:1→A
```

Pause statement: Localiseer de bug

- Om een bug te localiseren (=vinden waar in je `PRGM` de fout zit), is het handig om `Pause` te gebruiken.
- Zo kun je je `PRGM` breaken (=middenin zijn executie stoppen) waar je wilt.
- Dit doe je door op de `[ON]`-knop te drukken terwijl je `PRGM` gepauseerd is.
- Kies dan `Quit..` (`Goto` kan ook nuttig zijn - probeer het zelf.)
- Nu kun je alle variabelen bekijken. Is de waarde wat je verwacht had? Zo niet? Dan moet de bug *boven* je `Pause` statement zitten!

```
PROGRAM: PAUSETST
:0→A
:Pause
:1→A
```

Pause statement: Localiseer de bug

- Om een bug te localiseren (=vinden waar in je `PRGM` de fout zit), is het handig om `Pause` te gebruiken.
- Zo kun je je `PRGM` breaken (=middenin zijn executie stoppen) waar je wilt.
- Dit doe je door op de `[ON]`-knop te drukken terwijl je `PRGM` gepauseerd is.
- Kies dan `Quit`. (`Goto` kan ook nuttig zijn - probeer het zelf.)
- Nu kun je alle variabelen bekijken. Is de waarde wat je verwacht had? Zo niet? Dan moet de bug *boven* je `Pause` statement zitten!



```
PRGM PAUSETST
```

```
PROGRAM: PAUSETST
:0→A
:Pause
:1→A
```

Pause statement: Localiseer de bug

- Om een bug te localiseren (=vinden waar in je `PRGM` de fout zit), is het handig om `Pause` te gebruiken.
- Zo kun je je `PRGM` breaken (=middenin zijn executie stoppen) waar je wilt.
- Dit doe je door op de `[ON]`-knop te drukken terwijl je `PRGM` gepauseerd is.
- Kies dan `Quit`. (`Goto` kan ook nuttig zijn - probeer het zelf.)
- Nu kun je alle variabelen bekijken. Is de waarde wat je verwacht had? Zo niet? Dan moet de bug *boven* je `Pause` statement zitten!

<pre>ERR: BREAK 1: Quit 2: Goto</pre>	<pre>PROGRAM: PAUSETST : 0→A : Pause : 1→A</pre>
---------------------------------------	--

Pause statement: Localiseer de bug

- Om een bug te localiseren (=vinden waar in je `Pr9m` de fout zit), is het handig om `Pause` te gebruiken.
- Zo kun je je `Pr9m` breaken (=middenin zijn executie stoppen) waar je wilt.
- Dit doe je door op de `[ON]`-knop te drukken terwijl je `Pr9m` gepauseerd is.
- Kies dan `Quit`. (`Goto` kan ook nuttig zijn - probeer het zelf.)
- Nu kun je alle variabelen bekijken. Is de waarde wat je verwacht had? Zo niet? Dan moet de bug *boven* je `Pause` statement zitten!

<pre>Pr9mPAUSETST A 0</pre>	<pre>PROGRAM: PAUSETST :0→A :Pause :1→A</pre>
-----------------------------	---

Example: Debug deze code

Hier is het BIRTHDAY prgm van les 2. Alleen...Ik heb een fout gemaakt! Debug de code m.b.v. test cases. Bekijk extreme gevallen:

```
PROGRAM: BIRTHDAY
:Input "YOU Y",Y
:Input "YOU M",M
:Input "YOU D",D
:Input "CUR Y",Z
:Input "CUR M",N
:Input "CUR D",E
:
:Z-Y→A
:If M>N:Then
:A+1→A
:Else
:If M=N and D>E
:Then
:A+1→A
:End
:End
:Disp A
```

Example: Debug deze code

Hier is het `BIRTHDAY` prog van les 2. Alleen...Ik heb een fout gemaakt! Debug de code m.b.v. test cases. Bekijk extreme gevallen:

Wat gebeurt er als ik geboren ben op

- 1 Januari 0 en het is 10 Februari 1?
- 1 Januari 0 en het is 10 Januari 1?
- 5 Januari 0 en het is 1 Januari 1?

```
PROGRAM: BIRTHDAY
:Input "YOU Y",Y
:Input "YOU M",M
:Input "YOU D",D
:Input "CUR Y",Z
:Input "CUR M",N
:Input "CUR D",E
:
:Z-Y→A
:If M>N:Then
:A+1→A
:Else
:If M=N and D>E
:Then
:A+1→A
:End
:End
:Disp A
```


Example: Debug deze code

Hier is het `BIRTHDAY` program van les 2. Alleen...Ik heb een fout gemaakt! Debug de code m.b.v. test cases. Bekijk extreme gevallen:

Wat gebeurt er als ik geboren ben op

- 1 Januari 0 en het is 10 Februari 1?
 - `A=1`
- 1 Januari 0 en het is 10 Januari 1?
- 5 Januari 0 en het is 1 Januari 1?

```
PROGRAM: BIRTHDAY
:Input  "YOU Y",Y
:Input  "YOU M",M
:Input  "YOU D",D
:Input  "CUR Y",Z
:Input  "CUR M",N
:Input  "CUR D",E
:
::Z-Y→A
:If M>N: Then
: A+1→A
: Else
: If M=N and D>E
: Then
: A+1→A
: End
: End
: Disp A
```

Example: Debug deze code

Hier is het `BIRTHDAY` program van les 2. Alleen...Ik heb een fout gemaakt! Debug de code m.b.v. test cases. Bekijk extreme gevallen:

Wat gebeurt er als ik geboren ben op

- 1 Januari 0 en het is 10 Februari 1?
 - $A=1$
 - $M \leq N$, dus A blijft 1.
- 1 Januari 0 en het is 10 Januari 1?
- 5 Januari 0 en het is 1 Januari 1?

```
PROGRAM: BIRTHDAY
:Input "YOU Y",Y
:Input "YOU M",M
:Input "YOU D",D
:Input "CUR Y",Z
:Input "CUR M",N
:Input "CUR D",E
:
:Z-Y→A
:If M>N:Then
:A+1→A
:Else
:If M=N and D>E
:Then
:A+1→A
:End
:End
:Disp A
```

Example: Debug deze code

Hier is het `BIRTHDAY` program van les 2. Alleen...Ik heb een fout gemaakt! Debug de code m.b.v. test cases. Bekijk extreme gevallen:

Wat gebeurt er als ik geboren ben op

- 1 Januari 0 en het is 10 Februari 1?
 - $A=1$
 - $M \leq N$, dus A blijft 1.
 - Dat is wat je verwacht: 1 jaar oud.
- 1 Januari 0 en het is 10 Januari 1?
- 5 Januari 0 en het is 1 Januari 1?

```
PROGRAM: BIRTHDAY
:Input "YOU Y",Y
:Input "YOU M",M
:Input "YOU D",D
:Input "CUR Y",Z
:Input "CUR M",N
:Input "CUR D",E
:
:Z-Y→A
:If M>N:Then
:A+1→A
:Else
:If M=N and D>E
:Then
:A+1→A
:End
:End
:Disp A
```

Example: Debug deze code

Hier is het `BIRTHDAY` program van les 2. Alleen...Ik heb een fout gemaakt! Debug de code m.b.v. test cases. Bekijk extreme gevallen:

Wat gebeurt er als ik geboren ben op

- 1 Januari 0 en het is 10 Februari 1?
- 1 Januari 0 en het is 10 Januari 1?
 - A=1
- 5 Januari 0 en het is 1 Januari 1?

```
PROGRAM: BIRTHDAY
:Input "YOU Y",Y
:Input "YOU M",M
:Input "YOU D",D
:Input "CUR Y",Z
:Input "CUR M",N
:Input "CUR D",E
:
:Z-Y→A
:If M>N:Then
:A+1→A
:Else
:If M=N and D>E
:Then
:A+1→A
:End
:End
:Disp A
```

Example: Debug deze code

Hier is het `BIRTHDAY` program van les 2. Alleen...Ik heb een fout gemaakt! Debug de code m.b.v. test cases. Bekijk extreme gevallen:

Wat gebeurt er als ik geboren ben op

- 1 Januari 0 en het is 10 Februari 1?
- 1 Januari 0 en het is 10 Januari 1?
 - $A=1$
 - $M=N$ en $E>D$, dus A blijft 1.
- 5 Januari 0 en het is 1 Januari 1?

```
PROGRAM: BIRTHDAY
:Input "YOU Y",Y
:Input "YOU M",M
:Input "YOU D",D
:Input "CUR Y",Z
:Input "CUR M",N
:Input "CUR D",E
:
:Z-Y→A
:If M>N:Then
:A+1→A
:Else
:If M=N and D>E
:Then
:A+1→A
:End
:End
:Disp A
```

Example: Debug deze code

Hier is het `BIRTHDAY` program van les 2. Alleen...Ik heb een fout gemaakt! Debug de code m.b.v. test cases. Bekijk extreme gevallen:

Wat gebeurt er als ik geboren ben op

- 1 Januari 0 en het is 10 Februari 1?
- 1 Januari 0 en het is 10 Januari 1?
 - $A=1$
 - $M=N$ en $E>D$, dus A blijft 1.
 - Dat is wat je verwacht: 1 jaar oud.
- 5 Januari 0 en het is 1 Januari 1?

```
PROGRAM: BIRTHDAY
:Input "YOU Y",Y
:Input "YOU M",M
:Input "YOU D",D
:Input "CUR Y",Z
:Input "CUR M",N
:Input "CUR D",E
:
:Z-Y→A
:If M>N:Then
:A+1→A
:Else
:If M=N and D>E
:Then
:A+1→A
:End
:End
:Disp A
```

Example: Debug deze code

Hier is het BIRTHDAY program van les 2. Alleen...Ik heb een fout gemaakt! Debug de code m.b.v. test cases. Bekijk extreme gevallen:
 Wat gebeurt er als ik geboren ben op

- 1 Januari 0 en het is 10 Februari 1?
- 1 Januari 0 en het is 10 Januari 1?
- 5 Januari 0 en het is 1 Januari 1?
 - A=1

```
PROGRAM: BIRTHDAY
:Input "YOU Y",Y
:Input "YOU M",M
:Input "YOU D",D
:Input "CUR Y",Z
:Input "CUR M",N
:Input "CUR D",E
:
:Z-Y→A
:If M>N:Then
:A+1→A
:Else
:If M=N and D>E
:Then
:A+1→A
:End
:End
:Disp A
```

Example: Debug deze code

Hier is het `BIRTHDAY` program van les 2. Alleen...Ik heb een fout gemaakt! Debug de code m.b.v. test cases. Bekijk extreme gevallen:
 Wat gebeurt er als ik geboren ben op

- 1 Januari 0 en het is 10 Februari 1?
- 1 Januari 0 en het is 10 Januari 1?
- 5 Januari 0 en het is 1 Januari 1?
 - $A=1$
 - $M=N$ en $D>E$, dus A wordt 2.

```
PROGRAM: BIRTHDAY
:Input "YOU Y",Y
:Input "YOU M",M
:Input "YOU D",D
:Input "CUR Y",Z
:Input "CUR M",N
:Input "CUR D",E
:
:Z-Y→A
:If M>N:Then
:A+1→A
:Else
:If M=N and D>E
:Then
:A+1→A
:End
:End
:Disp A
```


Example: Debug deze code

Hier is het `BIRTHDAY` `prog` van les 2. Alleen...Ik heb een fout gemaakt! Debug de code m.b.v. test cases. Bekijk extreme gevallen:
 Wat gebeurt er als ik geboren ben op

- 1 Januari 0 en het is 10 Februari 1?
- 1 Januari 0 en het is 10 Januari 1?
- 5 Januari 0 en het is 1 Januari 1?
 - $A=1$
 - $M=N$ en $D>E$, dus A wordt 2.
 - Maar je verwacht 0 jaar! Een bug!

```
PROGRAM: BIRTHDAY
:Input "YOU Y",Y
:Input "YOU M",M
:Input "YOU D",D
:Input "CUR Y",Z
:Input "CUR M",N
:Input "CUR D",E
:
:Z-Y→A
:If M>N:Then
:A+1→A
:Else
:If M=N and D>E
:Then
:A+1→A
:End
:End
:Disp A
```

Example: Debug deze code

Hier is het `BIRTHDAY` program van les 2. Alleen...Ik heb een fout gemaakt! Debug de code m.b.v. test cases. Bekijk extreme gevallen:
Wat gebeurt er als ik geboren ben op

- 1 Januari 0 en het is 10 Februari 1?
- 1 Januari 0 en het is 10 Januari 1?
- 5 Januari 0 en het is 1 Januari 1?
 - $A=1$
 - $M=N$ en $D>E$, dus A wordt 2.
 - Maar je verwacht 0 jaar! Een bug!
 - Waarschijnlijk moet hier $A-1 \rightarrow A$ staan!

```
PROGRAM: BIRTHDAY
:Input "YOU Y",Y
:Input "YOU M",M
:Input "YOU D",D
:Input "CUR Y",Z
:Input "CUR M",N
:Input "CUR D",E
:
:Z-Y→A
:If M>N:Then
:A+1→A
:Else
:If M=N and D>E
:Then
:A+1→A
:End
:End
:Disp A
```

Outline

1 Tips 'n Tricks

- Random tips
- Debugging
- Catalog & CatalogHelp
- Precisie en afrondingsfouten

2 Functions

- Theorie
- Voorbeelden

3 Advanced I/O

- Menu
- Output
- num2str

4 Exercises

- Exercises
- Answers

De bibliotheek

- **Alle** functies van de rekenmachine staan alfabetisch in de catalogus: `[2nd][0]=[CATALOG]`.
- Dit zijn **meer functies** dan in de menus staan!
Check it out yourself!
- Een nuttige app (`[APPS]`) is `Ct19Help`.
(Download van het internet, of link met iemand die hem heeft.)



- Voer de app uit. De hulpfunctie is nu actief.
- Blader naar een functie naar keuze.
Bijvoorbeeld `randInt()`.
- Druk nu op `[+]` voor hulp.
- Je ziet nu de argumenten van de `randInt()`-functie: wat de GR van je verwacht.
- Alles tussen blokhaken (`()`) is optioneel
(=niet verplicht).

```
CATALOG
▶abs(
  and
  angle(
  ANOVA(
  Ans
  Archive
  Asm(
```

De bibliotheek

- **Alle** functies van de rekenmachine staan alfabetisch in de catalogus: `[2nd][0]=[CATALOG]`.
- Dit zijn **meer functies** dan in de menus staan!
Check it out yourself!
- Een nuttige app (`[APPS]`) is `Ct19Help`.
(Download van het internet, of link met iemand die hem heeft.)



- Voer de app uit. De hulpfunctie is nu actief.
- Blader naar een functie naar keuze.
Bijvoorbeeld `randInt()`.
- Druk nu op `[+]` voor hulp.
- Je ziet nu de argumenten van de `randInt()`-functie: wat de GR van je verwacht.
- Alles tussen blokhaken (`()`) is optioneel
(=niet verplicht).

```
CATALOG
Get(
GetCalc(
▶getDate
getDtFmt
getDtStr(
getTime
getTmFmt
```

De bibliotheek

- **Alle** functies van de rekenmachine staan alfabetisch in de catalogus: $\boxed{2nd}\boxed{0}=[\text{CATALOG}]$.
- Dit zijn **meer functies** dan in de menus staan!
Check it out yourself!
- Een nuttige app (\boxed{APPS}) is **CtlgHelp**.
(Download van het internet, of link met iemand die hem heeft.)
 - Voer de app uit. De hulpfunctie is nu actief.
 - Blader naar een functie naar keuze.
Bijvoorbeeld `randInt()`.
 - Druk nu op $\boxed{+}$ voor hulp.
 - Je ziet nu de argumenten van de `randInt()`-functie: wat de GR van je verwacht.
 - Alles tussen blokhaken (`()`) is optioneel
(=niet verplicht).



De bibliotheek

- **Alle** functies van de rekenmachine staan alfabetisch in de catalogus: $\boxed{2nd}\boxed{0}=[\text{CATALOG}]$.
- Dit zijn **meer functies** dan in de menus staan!
Check it out yourself!
- Een nuttige app (\boxed{APPS}) is **CtlgHelp**.
(Download van het internet, of link met iemand die hem heeft.)
 - Voer de app uit. De hulpfunctie is nu actief.
 - Blader naar een functie naar keuze.
Bijvoorbeeld `randInt()`.
 - Druk nu op $\boxed{+}$ voor hulp.
 - Je ziet nu de argumenten van de `randInt()`-functie: wat de GR van je verwacht.
 - Alles tussen blokhaken (`()`) is optioneel
(=niet verplicht).



De bibliotheek

- **Alle** functies van de rekenmachine staan alfabetisch in de catalogus: $\boxed{2\text{nd}}\boxed{0}=[\text{CATALOG}]$.
- Dit zijn **meer functies** dan in de menus staan!
Check it out yourself!
- Een nuttige app ($\boxed{\text{APPS}}$) is **Ctl9Help**.
(Download van het internet, of link met iemand die hem heeft.)
 - Voer de app uit. De hulpfunctie is nu actief.
 - Blader naar een functie naar keuze.
Bijvoorbeeld **randInt()**.
 - Druk nu op $\boxed{+}$ voor hulp.
 - Je ziet nu de argumenten van de **randInt()**-functie: wat de GR van je verwacht.
 - Alles tussen blokhaken ($\boxed{[]}$) is optioneel
(=niet verplicht).



```

MATH NUM CPX 123
1:rand(
2:nPr
3:nCr
4:!
5:randInt(
6:randNorm(
7:randBin(
    
```


De bibliotheek

- **Alle** functies van de rekenmachine staan alfabetisch in de catalogus: $\boxed{2\text{nd}}\boxed{0}=[\text{CATALOG}]$.
- Dit zijn **meer functies** dan in de menus staan!
Check it out yourself!
- Een nuttige app ($\boxed{\text{APPS}}$) is **CtlgHelp**.
(Download van het internet, of link met iemand die hem heeft.)
 - Voer de app uit. De hulpfunctie is nu actief.
 - Blader naar een functie naar keuze.
Bijvoorbeeld `randInt(`.
 - Druk nu op $\boxed{+}$ voor hulp.
 - Je ziet nu de argumenten van de `randInt(`-functie: wat de GR van je verwacht.
 - Alles tussen blokhaken (`[]`) is optioneel
(=niet verplicht).



```
randInt(  
  (lower,upper[,nu  
  mtrials])
```

De bibliotheek

- **Alle** functies van de rekenmachine staan alfabetisch in de catalogus: $\boxed{2\text{nd}}\boxed{0}=[\text{CATALOG}]$.
- Dit zijn **meer functies** dan in de menus staan!
Check it out yourself!
- Een nuttige app ($\boxed{\text{APPS}}$) is **CtlgHelp**.
(Download van het internet, of link met iemand die hem heeft.)
 - Voer de app uit. De hulpfunctie is nu actief.
 - Blader naar een functie naar keuze.
Bijvoorbeeld `randInt(`.
 - Druk nu op $\boxed{+}$ voor hulp.
 - Je ziet nu de argumenten van de `randInt(`-functie: wat de GR van je verwacht.
 - Alles tussen blokhaken (`[]`) is optioneel
(=niet verplicht).



```
randInt(  
  (lower,upper[,nu  
  mtrials])
```

De bibliotheek

- **Alle** functies van de rekenmachine staan alfabetisch in de catalogus: $\boxed{2nd}\boxed{0}=[\text{CATALOG}]$.
- Dit zijn **meer functies** dan in de menus staan!
Check it out yourself!
- Een nuttige app (\boxed{APPS}) is **CtlgHelp**.
(Download van het internet, of link met iemand die hem heeft.)
 - Voer de app uit. De hulpfunctie is nu actief.
 - Blader naar een functie naar keuze.
Bijvoorbeeld `randInt(`.
 - Druk nu op $\boxed{+}$ voor hulp.
 - Je ziet nu de argumenten van de `randInt(`-functie: wat de GR van je verwacht.
 - Alles tussen blokhaken (`[]`) is optioneel
(=niet verplicht).



```
randInt(  
  (lower,upper[,nu  
  mtrials])
```

Outline

- 1 Tips 'n Tricks
 - Random tips
 - Debugging
 - Catalog & CatalogHelp
 - **Precisie en afrondingsfouten**
- 2 Functions
 - Theorie
 - Voorbeelden
- 3 Advanced I/O
 - Menu
 - Output
 - num2str
- 4 Exercises
 - Exercises
 - Answers

Precisie

- Je rekenmachine slaat niet oneindig veel getallen van een rationeel getal op.
- Om precies te zijn, hij slaat 14 getallen op:
- Dit kan leiden tot afrondingsfouten.
- Zo is `1/9*9=0.9999999999999999≠1` ...
- Wat geeft de volgende functie?
- Gelukkig geeft `:If 1/9*9=1` wel gewoon “true”!
- Enig idee hoe we met afrondingsfouten kunnen omgaan?

Precisie

- Je rekenmachine slaat niet oneindig veel getallen van een rationeel getal op.
- Om precies te zijn, hij slaat 14 getallen op:

```
1/9
0.111 111 111 111 11
```

- Dit kan leiden tot afrondingsfouten.
 - Zo is `1/9*9=0.9999999999999999≠1` ...
 - Wat geeft de volgende functie?
-
- Gelukkig geeft `:If 1/9*9=1` wel gewoon “true”!
 - Enig idee hoe we met afrondingsfouten kunnen omgaan?

Precisie

- Je rekenmachine slaat niet oneindig veel getallen van een rationeel getal op.
- Om precies te zijn, hij slaat 14 getallen op:

```
1/9
0.111 111 111 111 11
```

- Dit kan leiden tot afrondingsfouten.
 - Zo is `1/9*9=0.9999999999999999≠1` ...
 - Wat geeft de volgende functie?
-
- Gelukkig geeft `:If 1/9*9=1` wel gewoon “true”!
 - Enig idee hoe we met afrondingsfouten kunnen omgaan?

Precisie

- Je rekenmachine slaat niet oneindig veel getallen van een rationeel getal op.
- Om precies te zijn, hij slaat 14 getallen op:

```
1/9
0.111 111 111 111 11
```

- Dit kan leiden tot afrondingsfouten.
- Zo is `1/9*9=0.9999999999999999≠1` ...
- Wat geeft de volgende functie?

- Gelukkig geeft `:If 1/9*9=1` wel gewoon “true”!
- Enig idee hoe we met afrondingsfouten kunnen omgaan?

Precisie

- Je rekenmachine slaat niet oneindig veel getallen van een rationeel getal op.
- Om precies te zijn, hij slaat 14 getallen op:

```
1/9
0.111 111 111 111 11
```

- Dit kan leiden tot afrondingsfouten.
- Zo is `1/9*9=0.9999999999999999≠1` ...
- Wat geeft de volgende functie?

```
int(1/9*9)
```

- Gelukkig geeft `:If 1/9*9=1` wel gewoon “true”!
- Enig idee hoe we met afrondingsfouten kunnen omgaan?

Precisie

- Je rekenmachine slaat niet oneindig veel getallen van een rationeel getal op.
- Om precies te zijn, hij slaat 14 getallen op:

```
1/9
0.111 111 111 111 11
```

- Dit kan leiden tot afrondingsfouten.
- Zo is `1/9*9=0.9999999999999999≠1` ...
- Wat geeft de volgende functie? → Niet 1, wat je wel wilt...!

```
int(1/9*9)
```

```
0
```

- Gelukkig geeft `:If 1/9*9=1` wel gewoon “true”!
- Enig idee hoe we met afrondingsfouten kunnen omgaan?

Precisie

- Je rekenmachine slaat niet oneindig veel getallen van een rationeel getal op.
- Om precies te zijn, hij slaat 14 getallen op:

```
1/9
0.111 111 111 111 11
```

- Dit kan leiden tot afrondingsfouten.
- Zo is `1/9*9=0.9999999999999999≠1` ...
- Wat geeft de volgende functie? → Niet 1, wat je wel wilt...!

```
int(1/9*9)
0
```

- Gelukkig geeft `:If 1/9*9=1` wel gewoon “true”!
- Enig idee hoe we met afrondingsfouten kunnen omgaan?

Precisie

- Je rekenmachine slaat niet oneindig veel getallen van een rationeel getal op.
- Om precies te zijn, hij slaat 14 getallen op:

```
1/9
0.111 111 111 111 11
```

- Dit kan leiden tot afrondingsfouten.
- Zo is `1/9*9=0.9999999999999999≠1` ...
- Wat geeft de volgende functie? → Niet 1, wat je wel wilt...!

```
int(1/9*9)
```

```
0
```

- Gelukkig geeft `:If 1/9*9=1` wel gewoon “true”!
- Enig idee hoe we met afrondingsfouten kunnen omgaan?

iPart en fPart

- Eerst, hoe weet ik dat er 14 getallen zijn?
- Dit kan bewezen worden m.b.v. `fPart`.
- Dit weergeeft de fraction-part van een getal (alles rechts van de punt).
- Similarly, `iPart` is de integer-part (links van de punt).
- Door de punt te verschuiven, kunnen we tellen hoeveel getallen de rekenmachine op slaat! 14!

iPart en fPart

- Eerst, hoe weet ik dat er 14 getallen zijn?
- Dit kan bewezen worden m.b.v. fPart.
- Dit weergeeft de fraction-part van een getal (alles rechts van de punt).
- Similarly, iPart is de integer-part (links van de punt).
- Door de punt te verschuiven, kunnen we tellen hoeveel getallen de rekenmachine op slaat! 14!



```
MATH NUM CPX PRB
1:abs(
2:round(
3:iPart(
4:fPart(
5:int(
6:min(
7↓max(
```

iPart en fPart

- Eerst, hoe weet ik dat er 14 getallen zijn?
- Dit kan bewezen worden m.b.v. fPart.
- Dit weergeeft de fraction-part van een getal (alles rechts van de punt).
- Similarly, iPart is de integer-part (links van de punt).
- Door de punt te verschuiven, kunnen we tellen hoeveel getallen de rekenmachine op slaat! 14!



```
fPart(2.5431)
0.5431
iPart(2.5431)
2
```

```
MATH NUM CPX PRB
1:abs(
2:round(
3:iPart(
4:fPart(
5:int(
6:min(
7:max(
```

iPart en fPart

- Eerst, hoe weet ik dat er 14 getallen zijn?
- Dit kan bewezen worden m.b.v. fPart.
- Dit weergeeft de fraction-part van een getal (alles rechts van de punt).
- Similarly, iPart is de integer-part (links van de punt).
- Door de punt te verschuiven, kunnen we tellen hoeveel getallen de rekenmachine op slaat! 14!



```
fPart(2.5431)
0.5431
iPart(2.5431)
2
```

```
MATH NUM CPX PRB
1:abs(
2:round(
3:iPart(
4:fPart(
5:int(
6:min(
7:max(
```


iPart en fPart

- Eerst, hoe weet ik dat er 14 getallen zijn?
- Dit kan bewezen worden m.b.v. **fPart**.
- Dit weergeeft de fraction-part van een getal (alles rechts van de punt).
- Similarly, **iPart** is de integer-part (links van de punt).
- Door de punt te verschuiven, kunnen we tellen hoeveel getallen de rekenmachine op slaat! 14!



```
fPart(1.2345678
901234567*1E10)
0.23%
```

```
MATH NOW CPX PRB
1:abs(
2:round(
3:iPart(
4:fPart(
5:int(
6:min(
7:max(
```

Hoe wordt $1/9 \times 9$ weer gelijk aan 1?

- Hoe kun je `:If X=1` aanpassen om met afrondingsfouten om te gaan?

Hoe wordt $1/9 \times 9$ weer gelijk aan 1?

- Hoe kun je `:If X=1` aanpassen om met afrondingsfouten om te gaan?
- Dan moeten we een “ongeveer gelijk aan” zelf maken...
- Dat kan op veel verschillende manieren! Dit is de makkelijkste:
- Maar niet (Waarom?):

Hoe wordt $1/9 \times 9$ weer gelijk aan 1?

- Hoe kun je `:If X=1` aanpassen om met afrondingsfouten om te gaan?
- Dan moeten we een “ongeveer gelijk aan” zelf maken...
- Dat kan op veel verschillende manieren! Dit is de makkelijkste:

```
:1/9→X
:1→Y
:If X/Y+1E-9≥1 and X/Y-1E-9≤1
:If abs(X-Y)+1E-9≥0 and abs(X-Y)-1E-9≤0
```

- Maar niet (Waarom?):

Hoe wordt $1/9 \times 9$ weer gelijk aan 1?

- Hoe kun je `:If X=1` aanpassen om met afrondingsfouten om te gaan?
- Dan moeten we een “ongeveer gelijk aan” zelf maken...
- Dat kan op veel verschillende manieren! Dit is de makkelijkste:

```
:1/9*9→X
:1→Y
:If X/Y+1E-9≥1 and X/Y-1E-9≤1
:If abs(X-Y)+1E-9≥0 and abs(X-Y)-1E-9≤0
```

- Maar niet (Waarom?):

```
:If X+1E-9≥Y and X-1E-9≤Y
```

Integer checking

- Wat is een integer (NL: geheel getal)?

Integer checking

- Wat is een integer (NL: geheel getal)?
- Een integer heeft een `fPart` van 0. Dus:

```
:If fPart(X)=0
:Then
:  "X INTEGER
:Else
:  "X RATIONAL
:End
```

- Door afrondingsfouten gaat dit echter mis, zoals in het voorbeeld: `fPart(1/9*9)=0.999...=1 \neq 0`
- Dit kun je afvangen door een heel klein getal bij `X` op te tellen:
- Interesting fact: In de wetenschap doet men iets soortgelijks wanneer het mogelijk is dat een getal `X=0` wordt, maar je er toch door wilt kunnen delen: `Y/(X+1E-30)`

Integer checking

- Wat is een integer (NL: geheel getal)?
- Een integer heeft een `fPart` van 0. Dus:

```
:If fPart(X)=0
:Then
:  "X INTEGER
:Else
:  "X RATIONAL
:End
```

- Door afrondingsfouten gaat dit echter mis, zoals in het voorbeeld: `fPart(1/9*9)=0.999...=1≠0`
- Dit kun je afvangen door een heel klein getal bij `X` op te tellen:
- Interesting fact: In de wetenschap doet men iets soortgelijks wanneer het mogelijk is dat een getal `X=0` wordt, maar je er toch door wilt kunnen delen: `Y/(X+1E-30)`

Integer checking

- Wat is een integer (NL: geheel getal)?
- Een integer heeft een `fPart` van 0. Dus:

```
:If fPart(X)=0
:Then
:  "% INTEGER
:Else
:  "% RATIONAL
:End
```

```
:If fPart(abs(X)+1E-9)<1E-7
:Then
:  "% INTEGER
:Else
:  "% RATIONAL
:End
```

- Door afrondingsfouten gaat dit echter mis, zoals in het voorbeeld: `fPart(1/9*9)=0.999...=1≠0`
- Dit kun je afvangen door een heel klein getal bij `%` op te tellen:
- Interesting fact: In de wetenschap doet men iets soortgelijks wanneer het mogelijk is dat een getal `X=0` wordt, maar je er toch door wilt kunnen delen: `Y/(X+1E-30)`

Integer checking

- Wat is een integer (NL: geheel getal)?
- Een integer heeft een `fPart` van 0. Dus:

```
:If fPart(X)=0
:Then
:  "% INTEGER
:Else
:  "% RATIONAL
:End
```

```
:If fPart(abs(X)+1E-9)<1E-7
:Then
:  "% INTEGER
:Else
:  "% RATIONAL
:End
```

- Door afrondingsfouten gaat dit echter mis, zoals in het voorbeeld: `fPart(1/9*9)=0.999...=1≠0`
- Dit kun je afvangen door een heel klein getal bij `X` op te tellen:
- Interesting fact: In de wetenschap doet men iets soortgelijks wanneer het mogelijk is dat een getal `X=0` wordt, maar je er toch door wilt kunnen delen: `Y/(X+1E-30)`

Outline

- 1 Tips 'n Tricks
 - Random tips
 - Debugging
 - Catalog & CatalogHelp
 - Precisie en afrondingsfouten
- 2 Functions
 - Theorie
 - Voorbeelden
- 3 Advanced I/O
 - Menu
 - Output
 - num2str
- 4 Exercises
 - Exercises
 - Answers

Wat is een functie?

- In “echte” programmeertalen op de computer heb je functies/methodes
- Dit zijn kleine stukjes code die iets doen wat je heel vaak nodig hebt
- Het gebruik van een functie zorgt ervoor dat je die stukjes code makkelijk kunt herbruiken
- Op de rekenmachine zijn er voorgeprogrammeerde functies:
 - 1 `abs()` berekent de absolute waarde van een getal
 - 2 `max()` geeft de grootste waarde van twee getallen
- Algemeen: je stopt er getallen in, en je krijgt er berekende getallen uit

Wat is een functie?

- In “echte” programmeertalen op de computer heb je functies/methodes
- Dit zijn kleine stukjes code die iets doen wat je heel vaak nodig hebt
- Het gebruik van een functie zorgt ervoor dat je die stukjes code makkelijk kunt herbruiken
- Op de rekenmachine zijn er voorgeprogrammeerde functies:
 - 1 `abs()` berekent de absolute waarde van een getal
 - 2 `max()` geeft de grootste waarde van twee getallen
- Algemeen: je stopt er getallen in, en je krijgt er berekende getallen uit

Wat is een functie?

- In “echte” programmeertalen op de computer heb je functies/methodes
- Dit zijn kleine stukjes code die iets doen wat je heel vaak nodig hebt
- Het gebruik van een functie zorgt ervoor dat je die stukjes code makkelijk kunt herbruiken
- Op de rekenmachine zijn er voorgeprogrammeerde functies:
 - 1 `abs()` berekent de absolute waarde van een getal
 - 2 `max()` geeft de grootste waarde van twee getallen
- Algemeen: je stopt er getallen in, en je krijgt er berekende getallen uit

Wat is een functie?

- In “echte” programmeertalen op de computer heb je functies/methodes
- Dit zijn kleine stukjes code die iets doen wat je heel vaak nodig hebt
- Het gebruik van een functie zorgt ervoor dat je die stukjes code makkelijk kunt herbruiken
- Op de rekenmachine zijn er voorgeprogrammeerde functies:
 - 1 `abs()` berekent de absolute waarde van een getal
 - 2 `max()` geeft de grootste waarde van twee getallen
- Algemeen: je stopt er getallen in, en je krijgt er berekende getallen uit

Wat is een functie?

- In “echte” programmeertalen op de computer heb je functies/methodes
- Dit zijn kleine stukjes code die iets doen wat je heel vaak nodig hebt
- Het gebruik van een functie zorgt ervoor dat je die stukjes code makkelijk kunt herbruiken
- Op de rekenmachine zijn er voorgeprogrammeerde functies:
 - 1 **abs**(berekent de absolute waarde van een getal
 - 2 **max**(geeft de grootste waarde van twee getallen
- Algemeen: je stopt er getallen in, en je krijgt er berekende getallen uit

```
:If X>0
:Then
:Disp X
:Else
:Disp -X
:End
```

abs(-5)	5
abs(6)	6

Wat is een functie?

- In “echte” programmeertalen op de computer heb je functies/methodes
- Dit zijn kleine stukjes code die iets doen wat je heel vaak nodig hebt
- Het gebruik van een functie zorgt ervoor dat je die stukjes code makkelijk kunt herbruiken
- Op de rekenmachine zijn er voorgeprogrammeerde functies:
 - ① **abs()** berekent de absolute waarde van een getal
 - ② **max()** geeft de grootste waarde van twee getallen
- Algemeen: je stopt er getallen in, en je krijgt er berekende getallen uit

```
:If X>Y
:Then
:Disp X
:Else
:Disp Y
:End
```

max(-10,10)	10
max(42,3)	42

Wat is een functie?

- In “echte” programmeertalen op de computer heb je functies/methodes
- Dit zijn kleine stukjes code die iets doen wat je heel vaak nodig hebt
- Het gebruik van een functie zorgt ervoor dat je die stukjes code makkelijk kunt herbruiken
- Op de rekenmachine zijn er voorgeprogrammeerde functies:
 - ① `abs()` berekent de absolute waarde van een getal
 - ② `max()` geeft de grootste waarde van twee getallen
- Algemeen: je stopt er getallen in, en je krijgt er berekende getallen uit

Je eigen functie definiëren

- Het is zeer nuttig om je eigen functies te definiëren voor kleine stukjes code die je wenst te herhalen
- Hoe doe je dat?

Je eigen functie definiëren

- Het is zeer nuttig om je eigen functies te definiëren voor kleine stukjes code die je wenst te herhalen
- Hoe doe je dat?

Je eigen functie definiëren

- Het is zeer nuttig om je eigen functies te definiëren voor kleine stukjes code die je wenst te herhalen
- Hoe doe je dat?
 - Kort antwoord: het **kan niet**.
 - Ok, dat is deprimerend...

Je eigen functie definiëren

- Het is zeer nuttig om je eigen functies te definiëren voor kleine stukjes code die je wenst te herhalen
- Hoe doe je dat?
 - Kort antwoord: het **kan niet**.
 - Ok, dat is deprimerend...

Je eigen functie definiëren

- Het is zeer nuttig om je eigen functies te definiëren voor kleine stukjes code die je wenst te herhalen
- Hoe doe je dat?
 - Kort antwoord: het **kan niet**.
 - Ok, dat is deprimerend...
 - Lang antwoord: we kunnen iets soortgelijks maken!

Je eigen functie definiëren

- Dit doen we door een los `prog` te maken, die geen I/O heeft.
 - Hij vraagt de gebruiker niet om input (e.g. `PROMPT`)
 - Hij weergeeft ook geen output (e.g. `DISP`)
- Het `prog` communiceert met andere `prog`'s door variabelen aan te passen
- Onderstaand (nutteloos) voorbeeld illustreert hoe we dit kunnen doen
- De naam van het functie-`prog` begint met een `0` om het `prog` te onderscheiden van een “executable”. (Optioneel.)

Je eigen functie definiëren

- Dit doen we door een los `prog` te maken, die geen I/O heeft.
 - Hij vraagt de gebruiker niet om input (e.g. `Prompt`)
 - Hij weergeeft ook geen output (e.g. `Disp`)
- Het `prog` communiceert met andere `prog`'s door variabelen aan te passen
- Onderstaand (nutteloos) voorbeeld illustreert hoe we dit kunnen doen
- De naam van het functie-`prog` begint met een `0` om het `prog` te onderscheiden van een “executable”. (Optioneel.)

Je eigen functie definiëren

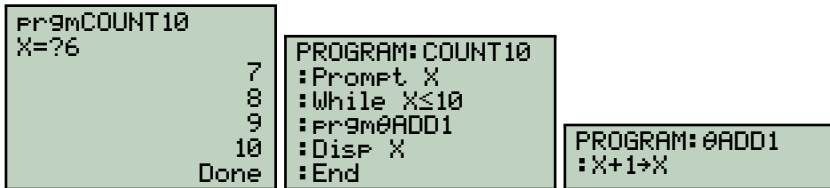
- Dit doen we door een los `PRGM` te maken, die geen I/O heeft.
 - Hij vraagt de gebruiker niet om input (e.g. `PROMPT`.)
 - Hij weergeeft ook geen output (e.g. `DISP`)
- Het `PRGM` communiceert met andere `PRGM`'s door variabelen aan te passen
- Onderstaand (nutteloos) voorbeeld illustreert hoe we dit kunnen doen
- De naam van het functie-`PRGM` begint met een \emptyset om het `PRGM` te onderscheiden van een “executable”. (Optioneel.)

Je eigen functie definiëren

- Dit doen we door een los `prog` te maken, die geen I/O heeft.
 - Hij vraagt de gebruiker niet om input (e.g. `Prompt`.)
 - Hij weergeeft ook geen output (e.g. `Disp`)
- Het `prog` communiceert met andere `prog`'s door variabelen aan te passen
- Onderstaand (nutteloos) voorbeeld illustreert hoe we dit kunnen doen
- De naam van het functie-`prog` begint met een `0` om het `prog` te onderscheiden van een “executable”. (Optioneel.)

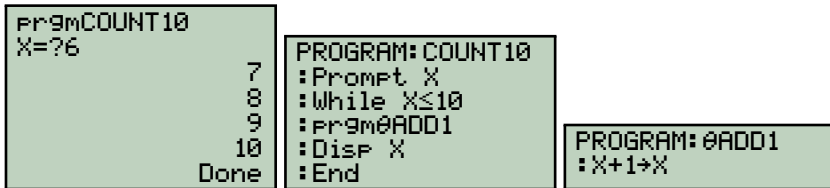
Je eigen functie definiëren

- Dit doen we door een los `PrgrM` te maken, die geen I/O heeft.
 - Hij vraagt de gebruiker niet om input (e.g. `Prompt`.)
 - Hij weergeeft ook geen output (e.g. `Disp`)
- Het `PrgrM` communiceert met andere `PrgrM`'s door variabelen aan te passen
- Onderstaand (nutteloos) voorbeeld illustreert hoe we dit kunnen doen
- De naam van het functie-`PrgrM` begint met een `θ` om het `PrgrM` te onderscheiden van een “executable”. (Optioneel.)



Je eigen functie definiëren

- Dit doen we door een los **Prgrm** te maken, die geen I/O heeft.
 - Hij vraagt de gebruiker niet om input (e.g. **Prompt**.)
 - Hij weergeeft ook geen output (e.g. **Disp**)
- Het **Prgrm** communiceert met andere **Prgrm**'s door variabelen aan te passen
- Onderstaand (nutteloos) voorbeeld illustreert hoe we dit kunnen doen
- De naam van het functie-**Prgrm** begint met een **θ** om het **Prgrm** te onderscheiden van een “executable”. (Optioneel.)



Outline

- 1 Tips 'n Tricks
 - Random tips
 - Debugging
 - Catalog & CatalogHelp
 - Precisie en afrondingsfouten
- 2 Functions
 - Theorie
 - Voorbeelden
- 3 Advanced I/O
 - Menu
 - Output
 - num2str
- 4 Exercises
 - Exercises
 - Answers

Functie voorbeeld

- De volgende functie eist dat de input waarde van `N` positief is.
- Zo niet, dan crashed het `PRGM` met een error.
- Het voordeel van hiervoor een aparte functie-`PRGM` gebruiken, is dat het functie-`PRGM` herbruikt kan worden:
 - Indien er bijv. twee input waarden zijn die positief moeten zijn.
 - Of voor andere `PRGMs`!

```
PRGMTHROWDIC
N=?6
21
Done
```

```
PROGRAM:THROWDIC
:"THROW N DICE
Disp SUM
:Prompt N
:N→X
:PRGM0MUSTGT0
:Disp
sum(randInt(1,6
,N))
```

```
PROGRAM:0MUSTGT0
:If X≤0
:Then
:Disp "VALUE
MUST BE >0, BUT
WAS:",X
:Stop
:Else
:Return
:End
```

Functie voorbeeld

- De volgende functie eist dat de input waarde van `N` positief is.
- Zo niet, dan crashed het `PRGM` met een error.
- Het voordeel van hiervoor een aparte functie-`PRGM` gebruiken, is dat het functie-`PRGM` herbruikt kan worden:
 - Indien er bijv. twee input waarden zijn die positief moeten zijn.
 - Of voor andere `PRGMs`!

```
PRGMTHROWDIC
N=? -3
VALUE MUST BE
>0, BUT WAS:
-3
Done
```

```
PROGRAM:THROWDIC
: "THROW N DICE
Disp SUM
: Prompt N
: N→X
: PRGM0MUSTGT0
: Disp
sum(randInt(1,6
,N))
```

```
PROGRAM:0MUSTGT0
: If X≤0
: Then
: Disp "VALUE
MUST BE >0, BUT
WAS: ",X
: Stop
: Else
: Return
: End
```


Functie awesomerest voorbeeld

- Het volgende **PRGM** vraagt je naar een functie (e.g. $y = x^2$).
- Daarna vraagt hij wat jij denkt dat de afgeleide van die functie is (e.g. $y = 2x$).
- Vervolgens plot hij de **echte** afgeleide, en jouw afgeleide tegelijkertijd.
- Zo kun je jouw antwoord grafisch controleren!

```

PROGRAM:PLTDERIV
:Disp "START
WITH QUOTE:"
:Disp "ORIG
FUNC:"
:Prompt Y1
:Disp "YOUR
DERIVATIVE:"
:Prompt Y2
:PRGMθPLTDERV
  
```

```

PROGRAM:θPLTDERV
:Disp "Y1=FUNC
:Disp "Y2=YOUR
DERIV
:FnOff 1,2
:GraphStyle(8,7)
:GraphStyle(9,5)
:"nDeriv(Y1,X,X)
"→Y8
:EquString(Y2
,Str0)
:Str0→Y9
:ZoomFit
:DispGraph
  
```

Basic I/O

- We hebben al de volgende I/O (input/output) functies gezien:
 - `Disp` voor simpele output
 - `Prompt` voor simpele input
 - `Input` voor iets uitgebreidere input
- Maar er zijn er nog veel meer!

Basic I/O

- We hebben al de volgende I/O (input/output) functies gezien:
 - **Disp** voor simpele output
 - **Prompt** voor simpele input
 - **Input** voor iets uitgebreidere input
- Maar er zijn er nog veel meer!

Basic I/O

- We hebben al de volgende I/O (input/output) functies gezien:
 - `Disp` voor simpele output
 - `Prompt` voor simpele input
 - `Input` voor iets uitgebreidere input
- Maar er zijn er nog veel meer!

Basic I/O

- We hebben al de volgende I/O (input/output) functies gezien:
 - `Disp` voor simpele output
 - `Prompt` voor simpele input
 - `Input` voor iets uitgebreidere input
- Maar er zijn er nog veel meer!

Basic I/O

- We hebben al de volgende I/O (input/output) functies gezien:
 - `Disp` voor simpele output
 - `Prompt` voor simpele input
 - `Input` voor iets uitgebreidere input
- Maar er zijn er nog veel meer!

Outline

- 1 Tips 'n Tricks
 - Random tips
 - Debugging
 - Catalog & CatalogHelp
 - Precisie en afrondingsfouten
- 2 Functions
 - Theorie
 - Voorbeelden
- 3 Advanced I/O
 - **Menu**
 - Output
 - num2str
- 4 Exercises
 - Exercises
 - Answers

Advanced input: Menu's

- Indien er voor `Prompt` of `Input` slechts een handjevol waardes is toegestaan (bijv. 1, 2 of 3, zoals bij de risk dobbelstenen), dan is `Menu` zeer nuttig.
- De gebruiker kan dan tussen deze toegestane waardes kiezen.
- Dit hoeven overigens geen getallen te zijn, de gebruiker kan ook bijv. kiezen tussen `AFLEIDEN` en `INTEGREREN`.
 - Afhankelijk van de keuze doet je `Pr9m` wat anders.
- De keuze die de gebruiker maakt werkt als een `Goto` statement:
 - De code springt naar een `Lbl` statement, zoals gegeven in het argument van `Menu`

```

I/O EXEC
9:Lbl
0:Goto
A:IS>(<
B:DS<<
Menu(<
D:Pr9m
E↓Return
    
```


Advanced input: Menu's

- Indien er voor `Prompt` of `Input` slechts een handjevol waardes is toegestaan (bijv. 1, 2 of 3, zoals bij de risk dobbelstenen), dan is `Menu` zeer nuttig.
- De gebruiker kan dan tussen deze toegestane waardes kiezen.
- Dit hoeven overigens geen getallen te zijn, de gebruiker kan ook bijv. kiezen tussen `AFLEIDEN` en `INTEGREREN`.
 - Afhankelijk van de keuze doet je `Pr9m` wat anders.
- De keuze die de gebruiker maakt werkt als een `Goto` statement:
 - De code springt naar een `Lbl` statement, zoals gegeven in het argument van `Menu`

```

I/O EXEC
9:Lbl
0:Goto
A:IS>(<
B:DS<<
Menu(<
D:Pr9m
E↓Return
    
```

Advanced input: Menu's

- Indien er voor **Prompt** of **Input** slechts een handjevol waardes is toegestaan (bijv. 1, 2 of 3, zoals bij de risk dobbelstenen), dan is **Menu** zeer nuttig.
- De gebruiker kan dan tussen deze toegestane waardes kiezen.
- Dit hoeven overigens geen getallen te zijn, de gebruiker kan ook bijv. kiezen tussen **AFLEIDEN** en **INTEGREREN**.
 - Afhankelijk van de keuze doet je **PRGM** wat anders.
- De keuze die de gebruiker maakt werkt als een **Goto** statement:
 - De code springt naar een **Lbl** statement, zoals gegeven in het argument van **Menu**

```

I/O EXEC
9:Lbl
0:Goto
A:IS>(<
B:DS<<
Menu(<
D:PRGM
E↓Return
    
```

Advanced input: Menu's

- Indien er voor **Prompt** of **Input** slechts een handjevol waardes is toegestaan (bijv. 1, 2 of 3, zoals bij de risk dobbelstenen), dan is **Menu** zeer nuttig.
- De gebruiker kan dan tussen deze toegestane waardes kiezen.
- Dit hoeven overigens geen getallen te zijn, de gebruiker kan ook bijv. kiezen tussen **AFLEIDEN** en **INTEGREREN**.
 - Afhankelijk van de keuze doet je **Pr9m** wat anders.
- De keuze die de gebruiker maakt werkt als een **Goto** statement:
 - De code springt naar een **Lbl** statement, zoals gegeven in het argument van **Menu**

```

I/O EXEC
9:Lbl
0:Goto
A:IS>(<
B:DS<<
Menu(<
D:Pr9m
E↓Return
    
```

Advanced input: Menu's

- Indien er voor **Prompt** of **Input** slechts een handjevol waardes is toegestaan (bijv. 1, 2 of 3, zoals bij de risk dobbelstenen), dan is **Menu** zeer nuttig.
- De gebruiker kan dan tussen deze toegestane waardes kiezen.
- Dit hoeven overigens geen getallen te zijn, de gebruiker kan ook bijv. kiezen tussen **AFLEIDEN** en **INTEGREREN**.
 - Afhankelijk van de keuze doet je **PRGM** wat anders.
- De keuze die de gebruiker maakt werkt als een **Goto** statement:

- De code springt naar een **Lbl** statement, zoals gegeven in het argument van **Menu**

```
Menu(  
  ("title", "text1"  
  , label1[, ..., "  
  text7", label7])
```

Advanced input: Menu's

- Indien er voor **Prompt** of **Input** slechts een handjevol waardes is toegestaan (bijv. 1, 2 of 3, zoals bij de risk dobbelstenen), dan is **Menu** zeer nuttig.
- De gebruiker kan dan tussen deze toegestane waardes kiezen.
- Dit hoeven overigens geen getallen te zijn, de gebruiker kan ook bijv. kiezen tussen **AFLEIDEN** en **INTEGREREN**.
 - Afhankelijk van de keuze doet je **prog** wat anders.
- De keuze die de gebruiker maakt werkt als een **Goto** statement:
 - De code springt naar een **Lbl** statement, zoals gegeven in het argument van **Menu**

```
Menu(
    ("title", "text1"
     , label1[, ..., "
     text7", label7])
```

Menu voorbeeld

```
PROGRAM:FRSTMENU
:Menu("WHAT PIE?
", "APPLE", 0, "ABR
ICOT", 1, "BLUEBER
RY", 2)
:Lbl 0
:Disp "YOU
CHOSE APPLE"
:Stop
:Lbl 1
:Disp "YOU
CHOSE ABRICOT"
:Stop
:Lbl 2
:Disp "YOU
CHOSE
BLUEBERRY"
```

prgmFRSTMENU

Menu voorbeeld

```
PROGRAM:FRSTMENU
:Menu("WHAT PIE?
", "APPLE", 0, "ABR
ICOT", 1, "BLUEBER
RY", 2)
:Lbl 0
:Disp "YOU
CHOSE APPLE"
:Stop
:Lbl 1
:Disp "YOU
CHOSE ABRICOT"
:Stop
:Lbl 2
:Disp "YOU
CHOSE
BLUEBERRY"
```

```
WHAT PIE?
1: APPLE
2: ABRICOT
3: BLUEBERRY
```

Menu voorbeeld

```
PROGRAM:FRSTMENU
:Menu("WHAT PIE?
", "APPLE", 0, "ABR
ICOT", 1, "BLUEBER
RY", 2)
:Lbl 0
:Disp "YOU
CHOSE APPLE"
:Stop
:Lbl 1
:Disp "YOU
CHOSE ABRICOT"
:Stop
:Lbl 2
:Disp "YOU
CHOSE
BLUEBERRY"
```

```
prgmFRSTMENU
YOU CHOSE APPLE
Done
```


Outline

- 1 Tips 'n Tricks
 - Random tips
 - Debugging
 - Catalog & CatalogHelp
 - Precisie en afrondingsfouten
- 2 Functions
 - Theorie
 - Voorbeelden
- 3 Advanced I/O
 - Menu
 - **Output**
 - num2str
- 4 Exercises
 - Exercises
 - Answers

Advanced output: formateer je scherm met `Output`

- `Disp` weergeeft een string van links naar rechts op de eerstvolgende regel
- `Output` weergeeft een string op een plek die je zelf kan kiezen
- Het scherm van je rekenmachine bestaat uit 8 rijen en 16 kolommen
- Probeer het volgende voorbeeld zelf:
 - ABC verschijnt links-bovenin en overschrijft "ProgramOutput."
 - G verschijnt rechts-in-het-midden; HI op de volgende regel
 - XY verschijnt rechts-onderin;
Z valt buiten het scherm

Advanced output: formateer je scherm met `Output`

- `Disp` weergeeft een string van links naar rechts op de eerstvolgende regel
- `Output` weergeeft een string op een plek die je zelf kan kiezen
- Het scherm van je rekenmachine bestaat uit 8 rijen en 16 kolommen
- Probeer het volgende voorbeeld zelf:
 - ABC verschijnt links-bovenin en overschrijft "ProgramOutput."
 - G verschijnt rechts-in-het-midden; HI op de volgende regel
 - XY verschijnt rechts-onderin;
 - Z valt buiten het scherm

Advanced output: formateer je scherm met `Output`

- `Disp` weergeeft een string van links naar rechts op de eerstvolgende regel
- `Output` weergeeft een string op een plek die je zelf kan kiezen
- Het scherm van je rekenmachine bestaat uit 8 rijen en 16 kolommen
- Probeer het volgende voorbeeld zelf:
 - ABC verschijnt links-bovenin en overschrijft "ProgramOutput."
 - G verschijnt rechts-in-het-midden; HI op de volgende regel
 - XY verschijnt rechts-onderin;
 - Z valt buiten het scherm

Advanced output: formateer je scherm met Output

- `Disp` weergeeft een string van links naar rechts op de eerstvolgende regel
- `Output` weergeeft een string op een plek die je zelf kan kiezen
- Het scherm van je rekenmachine bestaat uit 8 rijen en 16 kolommen
- Probeer het volgende voorbeeld zelf:
 - `ABC` verschijnt links-bovenin en overschrijft "`PrmOutput`"
 - `G` verschijnt rechts-in-het-midden; `HI` op de volgende regel
 - `XY` verschijnt rechts-onderin;
`Z` valt buiten het scherm

```
PROGRAM: OUTPUT
:Output(1,2,"ABC")
:Output(4,16,"GHI")
:Output(8,15,"XYZ")
```

```
PrmOutput
```

Advanced output: formateer je scherm met Output

- `Disp` weergeeft een string van links naar rechts op de eerstvolgende regel
- `Output` weergeeft een string op een plek die je zelf kan kiezen
- Het scherm van je rekenmachine bestaat uit 8 rijen en 16 kolommen
- Probeer het volgende voorbeeld zelf:
 - `ABC` verschijnt links-bovenin en overschrijft "`ProgramOutput`"
 - `GHI` verschijnt rechts-in-het-midden; `HI` op de volgende regel
 - `XYZ` verschijnt rechts-onderin;
`Z` valt buiten het scherm

```
PROGRAM: OUTPUT
:Output(1,2,"ABC")
:Output(4,16,"GHI")
:Output(8,15,"XYZ")
```

```
ProgramOutput
```

Advanced output: formateer je scherm met Output

- `Disp` weergeeft een string van links naar rechts op de eerstvolgende regel
- `Output` weergeeft een string op een plek die je zelf kan kiezen
- Het scherm van je rekenmachine bestaat uit 8 rijen en 16 kolommen
- Probeer het volgende voorbeeld zelf:
 - `ABC` verschijnt links-bovenin en overschrijft "`ProgramOutput`"
 - `G` verschijnt rechts-in-het-midden; `HI` op de volgende regel
 - `XYZ` verschijnt rechts-onderin;
`Z` valt buiten het scherm

```
PROGRAM: OUTPUT
: Output(1,2,"ABC")
: Output(4,16,"GHI")
: Output(8,15,"XYZ")
```

```
PROGRAMOutput
HI                                     G
XYZ
```

Advanced output: formateer je scherm met Output

- `Disp` weergeeft een string van links naar rechts op de eerstvolgende regel
- `Output` weergeeft een string op een plek die je zelf kan kiezen
- Het scherm van je rekenmachine bestaat uit 8 rijen en 16 kolommen
- Probeer het volgende voorbeeld zelf:
 - `ABC` verschijnt links-bovenin en overschrijft "`ProgramOutput`"
 - `G` verschijnt rechts-in-het-midden; `HI` op de volgende regel
 - `XY` verschijnt rechts-onderin;
`Z` valt buiten het scherm

```
PROGRAM: OUTPUT
: Output(1,2,"ABC")
: Output(4,16,"GHI")
: Output(8,15,"XYZ")
```

```
PROGRAMOutput
                G
HI
                XY
```


Nuttig samen met Output: ClrHome

- “With great power, comes great responsibility”
- Het is handig om het scherm eerst te legen, voordat je alles gaat overschrijven.
- Dit is precies wat `ClrHome` doet

Nuttig samen met Output: C1rHome

- “With great power, comes great responsibility”
- Het is handig om het scherm eerst te legen, voordat je alles gaat overschrijven.
- Dit is precies wat C1rHome doet

Nuttig samen met Output: ClrHome

- “With great power, comes great responsibility”
- Het is handig om het scherm eerst te legen, voordat je alles gaat overschrijven.
- Dit is precies wat `ClrHome` doet

```
CTL [F7] EXEC
2: Prompt
3: Disp
4: DispGraph
5: DispTable
6: Output(
7: getKey
8: ClrHome
```

Nuttig samen met Output: ClrHome

- “With great power, comes great responsibility”
- Het is handig om het scherm eerst te legen, voordat je alles gaat overschrijven.
- Dit is precies wat `ClrHome` doet

```
2+2
2*2
prgmCLRHOME
A
```

```
PROGRAM: CLRHOME
:Disp "A"
:Pause
:ClrHome
:Output(4,4,"HELL
O")
```

```
CTL [F1] EXEC
2: Prompt
3: Disp
4: DispGraph
5: DispTable
6: Output(
7: getKey
8: ClrHome
```

Nuttig samen met Output: ClrHome

- “With great power, comes great responsibility”
- Het is handig om het scherm eerst te legen, voordat je alles gaat overschrijven.
- Dit is precies wat `ClrHome` doet

HELLO

```
PROGRAM: CLRHOME
:Disp "A"
:Pause
:ClrHome
:Output(4,4,"HELL
O")
```

```
CTL [F7] EXEC
2: Prompt
3: Disp
4: DispGraph
5: DispTable
6: Output(
7: getKey
8: ClrHome
```

Outline

- 1 Tips 'n Tricks
 - Random tips
 - Debugging
 - Catalog & CatalogHelp
 - Precisie en afrondingsfouten
- 2 Functions
 - Theorie
 - Voorbeelden
- 3 Advanced I/O
 - Menu
 - Output
 - **num2str**
- 4 Exercises
 - Exercises
 - Answers

num2str

- Om mooie output te krijgen, is het handig om getallen naar strings te converteren
- Zo kun je als antwoord `↓(2)` laten zien, i.p.v. op meerdere regels:
- Op de TI84 gaat dit zeer ongemakkelijk...
- Maar met behulp van een custom functie hoeven we het slechts 1 keer te maken!

num2str

- Om mooie output te krijgen, is het handig om getallen naar strings te converteren
- Zo kun je als antwoord `√(2)` laten zien, i.p.v. op meerdere regels:

```
√(
    2
```

- Op de TI84 gaat dit zeer ongemakkelijk...
- Maar met behulp van een custom functie hoeven we het slechts 1 keer te maken!

num2str

- Om mooie output te krijgen, is het handig om getallen naar strings te converteren
- Zo kun je als antwoord `√(2)` laten zien, i.p.v. op meerdere regels:

```
√(  
2
```

- Op de TI84 gaat dit zeer ongemakkelijk...
- Maar met behulp van een custom functie hoeven we het slechts 1 keer te maken!

num2str

- Om mooie output te krijgen, is het handig om getallen naar strings te converteren
- Zo kun je als antwoord `√(2)` laten zien, i.p.v. op meerdere regels:

```
√(  
2
```

- Op de TI84 gaat dit zeer ongemakkelijk...
- Maar met behulp van een custom functie hoeven we het slechts 1 keer te maken!

num2str

- `0NUM2STR` zet het getal `X` om in een string `Str1`.
- `DISPSQRT` weergeeft het getal `X` in een wortel-teken.
- Nadeel is wel: de variabelen `X`, `Y0` en `Str1` worden overschreven door het `Prgrm`.
- Merk op: de comments geven aan wat de functie overschrijft!
Handig als je later terug kijkt!

```
PrgrmDISPSQRT
X=?2
√(2)
```

```
PROGRAM:DISPSQRT
:REQ:Prgrm0NUM2STR
:Promet X
:Prgrm0NUM2STR
:Disp "√("+Str1+"")"
```

```
PROGRAM:0NUM2STR
:"IN: X
:"OUT: Str1
:"USES: Y0
:(X,X)→iNSY
:(0,1)→iNSX
:LinReg(a+bx) iNSX,
iNSY,Y0
:EquString(Y0,Str1)
:sub(Str1,1,length(
Str1)-3)→Str1
:FnOff 0
```

num2str

- `0NUM2STR` zet het getal `X` om in een string `Str1`.
- `DISPSQRT` weergeeft het getal `X` in een wortel-teken.
- Nadeel is wel: de variabelen `X`, `Y0` en `Str1` worden overschreven door het `Pr9m`.
- Merk op: de comments geven aan wat de functie overschrijft!
Handig als je later terug kijkt!

```
Pr9mDISPSQRT
X=?2
√(2)
```

```
PROGRAM:DISPSQRT
:REQ:Pr9m0NUM2STR
:Promet X
:Pr9m0NUM2STR
:Disp "√("+Str1+"")"
```

```
PROGRAM:0NUM2STR
:"IN: X
:"OUT: Str1
:"USES: Y0
:(X,X)→iNSY
:(0,1)→iNSX
:LinReg(a+bx) iNSX,
iNSY,Y0
:EquString(Y0,Str1)
:sub(Str1,1,length(
Str1)-3)→Str1
:FnOff 0
```

num2str

- `0NUM2STR` zet het getal `X` om in een string `Str1`.
- `DISPSQRT` weergeeft het getal `X` in een wortel-teken.
- Nadeel is wel: de variabelen `X`, `Y0` en `Str1` worden overschreven door het `Prgrm`.
- Merk op: de comments geven aan wat de functie overschrijft!
Handig als je later terug kijkt!

```
PrgrmDISPSQRT
X=?2
√(2)
```

```
PROGRAM:DISPSQRT
:REQ:Prgrm0NUM2STR
:Promet X
:Prgrm0NUM2STR
:Disp "√("+Str1+"")"
```

```
PROGRAM:0NUM2STR
:"IN: X
:"OUT: Str1
:"USES: Y0
:(X,X)→iNSY
:(0,1)→iNSX
:LinReg(a+bx) iNSX,
iNSY,Y0
:EquString(Y0,Str1)
:sub(Str1,1,length(
Str1)-3)→Str1
:FnOff 0
```

Outline

- 1 Tips 'n Tricks
 - Random tips
 - Debugging
 - Catalog & CatalogHelp
 - Precisie en afrondingsfouten
- 2 Functions
 - Theorie
 - Voorbeelden
- 3 Advanced I/O
 - Menu
 - Output
 - num2str
- 4 Exercises
 - Exercises
 - Answers

Exercises

Voeg comments toe aan alle `prgms` die je maakt!

- ① Gebruik `0NUM2STR` om een functie-`prgm` (`0FRC2STR`) te schrijven dat de getallen `X` en `Y` neemt, en het omzet in een string van de vorm: `X/Y`. Bijv. `Str1=7/5` indien `X=7` en `Y=5`.
- ② Soortgelijk aan `0PLTDERIV`, schrijf een `prgm` dat de primitieve functie plot. (Hint: gebruik `fnInt` i.p.v. `nDeriv`; gebruik Google of `CtlgHelp` om te uit te zoeken hoe `fnInt` werkt.)
- ③ Schrijf een functie-`prgm` `0ISINT` die controleert of `X` integer is. Return `X=1` indien “true”, `X=0` indien “false”.

Exercises

Voeg comments toe aan alle `Frags` die je maakt!

- ❶ Maak een spiek`Frage` waarin de gebruiker de afgeleide van een functie kan opvragen. Gebruik hiervoor een `Menu` met items als x^n , a^x , $\cos x$, etc.
- ❷ Maak met behulp van `Output` een tekening met letters
EN/OF Gebruik `Output` om te weergeven dat $2+2=5$.
- ❸ Schrijf een functie-`Frage` (`ENUM2SQR`) dat x omzet in de vorm `NJ(M)`, waarbij $x = n\sqrt{m}$ en n en m integer zijn.
(Hint: schrijf $x^2 = n^2m$ en loop over alle mogelijke waarde van n . Begin bij de grootst mogelijke waarde van n en tel af. Kijk vervolgens of x^2/n^2 een integer is.)

Exercises: Debugging

- 1 Debug het volgende `PRGM`

```
PRGMPRIMES
X=?5
      (2 3 4 5)
      Done

PRGMPRIMES
X=?9
      (2 3 4 5 6 7 8 9)
      Done
```

```
PROGRAM: PRIMES
: Prompt X
: (2)→L1
: For(I,3,X)
: "CHECK IF I PRIME
: 1→P
: For(J,1,dim(L1))
: If fPart(I/L1(J
: )=0: Then
: "I NOT PRIME
: dim(L1)+1→J
: End
: End
: If P: Then
: augment(L1,(I))→L1
: End: End
: Disp L1
```

Outline

- 1 Tips 'n Tricks
 - Random tips
 - Debugging
 - Catalog & CatalogHelp
 - Precisie en afrondingsfouten
- 2 Functions
 - Theorie
 - Voorbeelden
- 3 Advanced I/O
 - Menu
 - Output
 - num2str
- 4 Exercises
 - Exercises
 - Answers

Answers

```
PROGRAM: 0ISINT
: "IN: X
: "OUT: X=1 If X
INTEGER Else 0
:
: If fPart(abs(X)+1E-9)
<1E-7
: Then
: "X INTEGER
: 1→X
: Else
: "X RATIONAL
: 0→X
: End
```

```
PROGRAM: 0FRC2STR
: "IN: X/Y
: "OUT: Str1
: "USE: Str0
: "REQ: prgm0NUM2STR
:
: prgm0NUM2STR
: If Y≠1: Then
: Str1→Str0
: Y→X
: prgm0NUM2STR
: Str0+"/"+Str1→Str1
: End
```

Answers

```
PROGRAM: AFGELYDN
:Lbl 0
:ClrHome
:Menu("KIES: ", "X^N", 1, "cos(X)
", 2, "sin(X)", 3, "e^X", 4, "EXIT
PROGRAM", 99)
:Lbl 1
:Output(1, 1, "C*X^N: ")
:Output(2, 1, "C*N*X^(N-1)")
:Output(4, 1, "C*(AX)^N: ")
:Output(5, 1, "C*AN*(AX)^(N-1)=
C*(A^N)*N*X^(N-1)")
:Goto 0
:Lbl 2
:Output(1, 1, "C*cos(X): ")
:Output(2, 1, "C*sin(X)")
:Output(4, 1, "C*cos(AX): ")
:Output(5, 1, "C*A*sin(AX)")
:Goto 0
:Lbl 3
:"EN NU ZO DOORGAAN...
:Lbl 99
```

Answers

Dit kan op veel verschillende manieren...Onderstaande maakt gebruik van `Pr9m0ISINT`.

```
PROGRAM: 0NUM2SQR
: "IN: X
: "OUT: N√(M) or
M=0 IF NOT √(
: "REQ: Pr9m0ISINT
: "USES: Y
:
: X→Y
: X²→X
: Pr9m0ISINT
: If X=0: Then
: "X NOT √( BC X²
NOT INT
: Y→N: 0→M
: Return
: End
: Y→X
```

```
:
: "X²=N²M?
: int(X)+1→N
: X²→Y
: 0→X
: While X=0
: N-1→N
: Y/N²→M
: M→X
: Pr9m0ISINT
: End
```

Answers: Debugging

De redenering is als volgt:

- We zien dat de output van het `PRGM` alle getallen bevat van 2 t/m X , i.p.v. alle priemgetallen $\leq X$. DUS:
 - Het `PRGM` checked incorrect voor priemgetallen.
 - Of het weergeeft te veel.
- Controleer het makkelijkste eerst: “het weergeeft te veel”. Wat weergeeft het `PRGM`?
 - Het weergeeft L_1 .
- Waar wordt L_1 gemaakt?
 - Bij de `augment` functie...Maar alleen als $P \neq 0$.
- Maar P kan nergens 0 worden in het `PRGM`! Daar is iets mis...
- We missen `0→P` onder de comment `"I NOT PRIME"`.

Answers: Debugging

- Corrigeer de fout.
- En controleer of het `PRgm` nu doet wat je verwacht!
- Yup! Awesome!

```
PRgmPRIMES
X=?5
(2 3 5)
Done

PRgmPRIMES
X=?9
(2 3 5 7)
Done
```

```
PROGRAM: PRIMES
: Prompt X
: (2)→L1
: For(I,3,X)
: "CHECK IF I PRIME
: 1→P
: For(J,1,dim(L1))
: If fPart(I/L1(J)
: )=0: Then
: "I NOT PRIME
: 0→P
: dim(L1)+1→J
: End
: End
: If P: Then
: augment(L1,(I))→L1
: End: End
: Disp L1
```