Module 4: Lussen (loops) **Oefenblad 2: De While…End lus**

In deze tweede les van module 4 leer je over de **While…End**-lus. We vergelijken deze met de **For**… -lus en laten je ook zien waarom deze krachtiger en flexibeler is dan de **For…** -lus.

Doelen:

- De structuur van de **While…End**-lus leren.
- Deze vergelijken met de **For…End**-lus.
- Zien hoe deze gebruikt wordt om te zorgen dat de invoer geldig is.

De While… End-lus

De **While…End**-lus loopt door zolang de <voorwaarde> Waar is. De lus ziet er zo uit:

While <voorwaarde>
<lus kern>
End

Een oneindige lus! Waarom?

Opmerking
De <voorwaarde> is een logische uitdrukking zoals **X>0.**
De <lus kern> is een willekeurige verzameling opdrachten, die ook lussen en **If***-structuren kan bevatten. De kern wordt uitgevoerd wanneer en zolang <voorwaarde>* waar *is.*
Het sleutelwoord **End** *wordt gebruikt om het einde ('de bodem') van de <lus kern> aan te geven. Bij de opdracht* **End** *'lust' het programma terug naar de opdracht* **While** *en test de <voorwaarde> opnieuw.*

Wat zal dit programma doen?

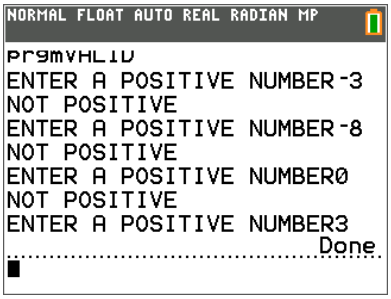
'Initialiseer' de voorwaarde **voor While**:
Stel een waarde in zodat de voorwaarde op de goede manier is ingesteld als waar of onwaar. Als de beginvoorwaarde onwaar is dan wordt de lus helemaal overgeslagen. Als de voorwaarde waar is dan wordt de kern van de lus verwerkt.
De **0→K** bovenaan in dit programma stelt de beginvoorwaarde in op onwaar. Zonder deze opdracht kun je niet weten wat er zal gebeuren omdat elke willekeurige waarde kan zijn opgeslagen in de variabele **K** voordat het programma wordt uitgevoerd.
Ergens in de <lus kern> zou er een opdracht moeten zijn die een effect heeft op de <voorwaarde> zodat de lust uiteindelijk zal stoppen en de opdrachten na de lus zullen worden uitgevoerd. Gewoonlijk staat deze opdracht onderaan dichtbij de <lus kern>. **K+1→ K** zorgt ervoor dat **K** uiteindelijk groter zal zijn dan 10.

Docenten Tip: Het is belangrijk om te benadrukken dat de **While**-lus, misschien helemaal niet wordt uitgevoerd. In de volgende les bespreken we de **Repeat**-lus die altijd minstens één keer wordt uitgevoerd. Dit is een subtiel maar belangrijk onderscheid.
Er zijn drie componenten nodig om een succesvolle **While**-lus te maken: ***Initialiseren, testen en veranderen***:

- ***Initialiseren*** van een variabele (beginwaarde instellen).
- ***Testen*** van een voorwaarde gebaseerd op die variabele.
- ***Veranderen*** van de variabele zodat uiteindelijke de voorwaarde onwaar wordt, zodat de lus zal stoppen.

Nagaan of Input geldig is met While…End

We zullen een gedeelte (code-segment) van een programma schrijven dat er voor zorgt dat de gebruiker een positief getal invoert, haar vertelt wanneer de invoer ongeldig is en dan verzoekt om een andere waarde in de plaats ervan in te voeren.



De uitvoer van dit programmagedeelte staat hier rechts (in het Engels) waarbij enkele niet-positieve getallen zijn ingevoerd om het effect te zien. Kijk eens of je dit programmagedeelte kunt schrijven zonder te spieken op de volgende pagina!

Opmerking: Gebruikers van de TI-84 Plus moeten misschien een korter bericht gebruiken omdat het scherm minder breed is.

We beginnen met een **Input**-opdracht met een bericht om een waarde te krijgen van de gebruiker …

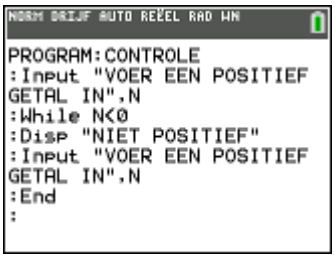
Docenten Tip: Dit ***initialiseert*** N.

Begin dan de **While**-lus, om te controleren of de ingevoerde waarde negatief is. SAls dat zo is dan geven we met **Disp** een foutmelding weer.

De kern van de lus zal alleen binnengegaan worden als N negatief is, anders zal de hele kern overgeslagen worden.

Docenten Tip: Dit stelt de voorwaarde in om te ***test*en** op basis van N.

Gebruik tenslotte de opdracht **Input** opnieuw aan het eind van de om een nieuwe waarde te vragen aan de gebruiker en dan de <lus kern> te beëindigen (met **End**).



Dit programmagedeelte vraagt een waarde van de gebruiker en zorgt ervoor dat de invoer een positief getal is. Onder dit programmagedeelte zullen meer opdrachten volgen om het ingevoerde getal te verwerken.

Docent Tip: Dit geeft de mogelijkheid om de waarde van N te veranderen.