**Module 4: Lussen (loops )** **Toepassing: Waarschuwingen**

Deze toepassing (WAARSCHUWINGEN) maakt gebruik van lussen om zo veel variabelen als nodig in te voeren en controleert, als uitbreiding, of de ingevoerde waarden geldig zijn en gebruikt **If**-opdrachten om een passend bericht te tonen.

**Doelen:**

- Leren over de opdrachten teller (counter) en (accumulator).
- Een lus gebruiken in een programma om een onbepaalde hoeveelheid gegevens te verkrijgen.
- Een 'vlag'-waarde gebruiken om een lus te beëindigen.

**Docenten Tip:** Dit project illustreert de neiging tot complexiteit bij het ontwikkelen van een stuk software. *Nesten (nesting)* hangt samen met het idee om de ene besturingsstructuur op te nemen binnen een andere. Zoals hieronder wordt uitgelegd weet het OS (besturingssysteem) welke End bij welke opdracht hoort.

**Geneste structuren**

- *Nesten* is de programmeertechniek waarbij een besturingsopdracht binnen een andere wordt geplaatst. De term is afgeleid van het idee van het plaatsen van de ene kartonnen doos in de andere om zo ruimte te besparen.
- Het is belangrijk om een een *complete* structuur *volledig* binnen een blok van een andere te plaatsen om fouten te voorkomen.
- De programmaregels rechts laten een **If**-structuur binnen een **Else**-blok van een andere **If**-structuur zien. Let op het meerdere keren gebruiken van de **End**-opdracht; de computer weer welke **End** bij welke **If** hoort.
- Het inspringen van de regels is alleen gedaan voor de uitleg. Je kunt in TI-Basic regels niet laten inspringen.
- Het programma test eerst of **A<0**. Als dat zo is, dan toont het programma het bericht "A is negative!" (A is negatief) en niets anders. Maar als **A** niet negatief is dan worden de **onderstreepte** berekening van de wortel, nog een **If**-opdracht en de opdracht **Disp** allemaal uitgevoerd.
- De programmeur plaatst **If**s binnen lussen en lussen binnen **If**s om de meer complexe taken die nodig zijn voor het programma te realiseren.

```
prgmSQUARE
Prompt A
If A<Ø
Then
    Disp "A is negative!"
Else
    √(A)→S
    If S=int(S)
    Then
      Disp "A is a perfect
square."
    Else
      Disp "A is not a perfect
square."
    End
    Disp "Its square root is",S
End
```

**Samenvatting van de drie lussen:**

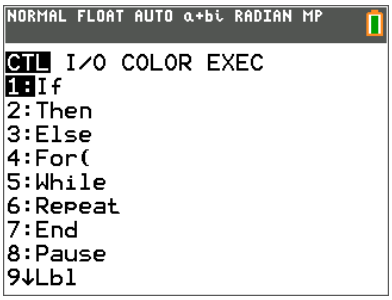
| **For(***var*, *begin*, *eind***)** | **While** *voorwaarde is waar* | **Repeat** *totdat voorwaarde is waar* |
|---|---|---|
| **End** | **End** | **End** |

**For(** wordt gebruikt bij 'tellen' of bij het verwerken van een rekenkundige rij van waarden (iteratie).
**While** wordt gebruikt wanneer je mogelijk de volledige kern van een lus kunt overslaan
**Repeat** wordt gebruikt wanneer je zeker weet dat je de kern van een lus minstens een keer wilt uitvoeren.

**Over End**
Het sleutelwoord End wordt gebruikt voor alle meerregelige besturingsstructuren:

| | | | | |
|---|---|---|---|---|
| **If …**<br>**Then** | **If …**<br>**Then** | **For( … )** | **While …** | **Repeat …** |
| | **Else** | | | |
| **End** | **End** | **End** | **End** | **End** |

En dit is de reden dat het $\frac{1}{4}$ CTL menu georganiseerd is op deze manier:




**7: End** komt na de eerste zes opties van het menu CTL menu omdat het elk van deze besturingsstructuren van het programmeren afsluit (‘ends’).

**End**-opdrachten kunnen heel vaak voorkomen in een programma en de computer weet hoe deze **End**s verbonden zijn met hun besturingsstructuren.

**Module 4 toepassing: Programma “Waarschuwingen”**

De meeste scholen versturen regelmatig rapporten gebaseerd op de actuele gemiddelde cijfers van de leerlingen. Een gemiddelde van 5,5 (of 55%) of hoger wordt gezien als voldoende, maar als het gemiddelde onder de 6 (of 60%) ligt dan wordt een leerling soms gezien als ‘in de gevarenzone’ of als bespreekgeval.

Laten we een programma schrijven dat de gebruiker enkele proefwerkcijfers laat invoeren, het gemiddelde berekent en dan het aantal ingevoerde cijfers, het gemiddelde ervan en een passende ‘waarschuwing’ laat zien. Dit waarschuwingsbericht kan zijn: “Hiermee ga je over”, “Je zit in de gevarenzone” en “Hiermee blijf je zitten”.

We kunnen twee manieren gebruiken om een onbekend aantal cijfers in te voeren:

- Manier 1: vraag eerst om het totaalaantal cijfers en gebruik een **For**-lus om de cijfers in te voeren.
- Manier 2: vraag de cijfers, maar gebruik een ‘vlag’-waarde zoals -999 om aan te geven dat er niet meet cijfers zijn. Bij deze manier zal je een **While**-lus of een **Repeat**-lus gebruiken.

In beide gevallen zullen we een lopend totaal van de cijfers bij moeten houden. Bij manier 2 moeten we ook het aantal cijfers tellen zodat we het totaal kunnen delen door dat aantal.

Jouw programma zou het aantal ingevoerde cijfers, het gemiddelde van deze cijfers en de passende waarschuwing moeten weergeven:

Als het gemiddelde minder dan 5,5 (55%) zit: “Hiermee blijf je zitten”

… 5,5 tot 6: “Je zit in de gevarenzone”

… meer dan 6: “Hiermee ga je over”

**Docenten Tip:** In het gedeelte hieronder bespreken we twee gerelateerde programmeer-ideeën: een teller en een 'accumulator 'variabele. Benadruk dat de variabele links van de opslagoperator dezelfde variabele is als die aan de rechterkant, maar dat zij verschillende waarden voorstellen door de manier waarop waarden verwerkt worden.

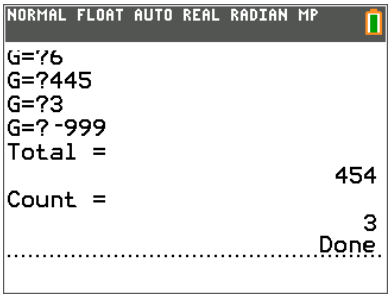
**Tellers en accumulatoren**

Een opdracht zoals **C+1→C** wordt een *teller* genoemd omdat deze zorgt dat er elke keer dat de opdracht wordt uitgevoerd 1 wordt opgeteld bij de variabele **C**.

Een opdracht zoals **T+N→T** wordt een *accumulator* genoemd omdat de opdracht een lopend totaal van de waarden van de variabele **N** bijhoudt. De waarde van **N** wordt opgeteld bij de variabele **T** en vervolgens wordt die som weer 'terug' opgeslagen in variabele **T**. Aan het eind van een lus zal **T** het totaal van de **N** waarden bevatten.

Hier zie je een voorbeeld waarin een teller **C** en een accumulator **A** worden gebruikt en ook een 'vlag'-waarde om de **G'**s (cijfers) in een programma bij te houden:

| | Opmerkingen |
|---|---|
| **0→C** | Initialiseren van de variabelen; |
| **0→G** | G staat voor het cijfer |
| **0→T** | C staat voor de telling (het aantal) |
| **Prompt G** | T staat voor het totaal |
| **While G≠-999** | vraag om het eerste cijfer (**G**) |
| **C+1→C** | zolang het niet gelijk is aan -999 |
| **T+G→T** | tel 1 op bij de telling **C** |
| **Prompt G** | tel het cijfer (**G**) op bij het totaal (**T**) |
| **End** | vraag om nog een cijfer (**G**) |
| **Disp "Total =",T** | |
| **Disp "Count =",C** | |




De **While**-lus hierboven blijft tellen en de **G'**s bij elkaar tellen (accumuleren) zoals als -999 niet wordt ingevoerd. Wanneer -999 wordt ingevoerd stopt de lus en worden de resultaten weergegeven.

**Docenten Tip:** Wijs op de twee **Prompt** opdrachten in de programmacode hierboven. De eerste haalt het eerste cijfer op en de laatste haalt de rest van de cijfers op. De laatste **Prompt** staat aan het *eind* van de lus om de terugkeer naar de opdracht **While** voor te bereiden om de waarden van G opnieuw te testen.

**Uitbreiding**

Als onderdeel van je input-routine kun je een controle uitvoeren om er zeker van te zijn dat de waarde die is ingevoerd een geldig cijfer is (tussen 0 en 10) en om een passende actie uit te voeren wanneer dat niet zo is.

**Docenten Tip:** Deze uitbreiding vraagt om een nieuwe lus om de invoer-opdrachten heen om er zeker van te zijn dat de ingevoerde waarde geldig is. Het voorbeeld met programmacode hieronder stopt simpelweg het programma wanneer er een ongeldige waarde is ingevoerd. Leerlingen kunnen een betere oplossing bedenken !

**Voorbeeldantwoord in het Engels (het inspringen is alleen gedaan voor de duidelijkheid):**

```
prgmWARNINGZ
ClrHome
Ø  →A
Ø  →C
Ø  →T
Input "ENTER A GRADE:",A

While A    ≠999
  If A<Ø or A>1ØØ
  Then
    Disp "OUT OF RANGE!"
    Stop
  Else
    C+1  →C
    T+A  →T
  End
  Input "ANOTHER? (OR -999):",A
End
T/C  →M

ClrHome
Output(1,1,"NUMBER OF GRADES:")
Output(2,1," TOTAL OF GRADES:")
Output(4,1,"AVERAGE:")
Output(1,19,C)
Output(2,19,T)
Output(4,1Ø,M)
If M<6Ø
Then
  Output(5,9,"FAILING")
Else
  If M<7Ø
  Then
    Output(5,9,"MARGINAL")
  Else
    Output(5,9,"PASSING")
  End
End
Pause
ClrHome
```

Wanneer je een programma test probeer dan de uitzonderlijke gevallen uit, zoals het invoeren van -999 als het eerste cijfer en het ergens invoeren van negatieve waarden. Als het programma dan een foutmelding geeft, heeft de programmeur geen rekening gehouden met alle mogelijke scenario's. Het programma hierboven zal falen wanneer -999 wordt ingevoerd als eerste cijfer, omdat het zal proberen 0 te delen door 0, wat niet kan. De berekening van het gemiddelde, M, zou in een test moeten worden 'verpakt' om zeker te weten dat er minstens een cijfer is ingevoerd. Hieronder staat een mogelijke oplossing om dit te repareren:

```
If C>0
Then
 T/C→M
Else
 Disp "NO GRADES ENTERED!"
 Stop
End
```

De opdracht **Stop** wordt in dit programma gebruikt om een programma *onmiddellijk* te beëindigen. Deze opdracht kan overal in een programma worden toegevoegd zodat de uitvoering wordt onderbroken en de boodschap 'Klaar' van het beginscherm verschijnt.