Unit 4: Lussen | **Oefenblad 3: De Loop-lus**

In deze les leer je over de universele **Loop…EndLoop**-structuur.

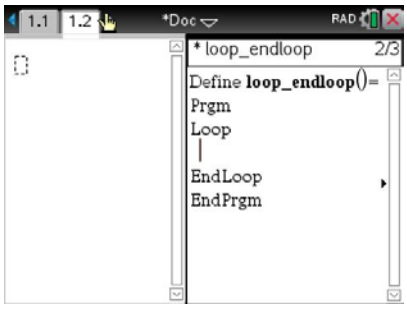
Doelen:

- De **Loop…EndLoop**-structuur gebruiken
- De opdracht **Exit** gebruiken om een lus te verlaten

Docenten Tip: De meeste traditionele programmeerdocenten houden er niet van om de **Loop…EndLoop**-structuur te gebruiken omdat deze slechte programmeergewoontes aankweekt die leiden tot 'spaghetti code' (programma's die steeds heen en weer springen door het gebruik van **GoTo**-opdrachten). De **Loop…EndLoop**-structuur wordt hier behandeld met de bedoeling dat deze op gepaste wijze wordt aangepakt: gebruik slechts één **Exit**-opdracht en gebruik onder geen enkele omstandigheid de opdracht **GoTo**!
Een voordeel van de **Loop…EndLoop**-structuur is de *mogelijkheid* om meerdere Exit punten te maken; dit is echter een zeldzaamheid en er is altijd een alternatieve manier om deze situatie te vermijden. De opdracht **Exit** forceert de programmabesturing om de eerste opdracht na **EndLoop** te verwerken, er is dus geen verwarring over de richting waarin het programma verder gaat.

Wat is zo'n Loop…EndLoop eigenlijk?

De **Loop…EndLoop**-structuur laat ons een meer flexibele lus-structuur creëren. Het voert opdrachten in de kern van de lus herhaaldelijk uit. Merk op dat de lus eindeloos zal worden uitgevoerd, tenzij er ergens in de kern van de lus een **Exit**-instructie wordt uitgevoerd.



Als het programma geen **Exit**-opdracht tegenkomt dan zal de lus een oneindige lus zijn. Als je per ongeluk in een oneindige lus terecht komt op de rekenmachine, druk dan op **ON/HOME** en houd dit ingedrukt totdat het programma onderbroken wordt.

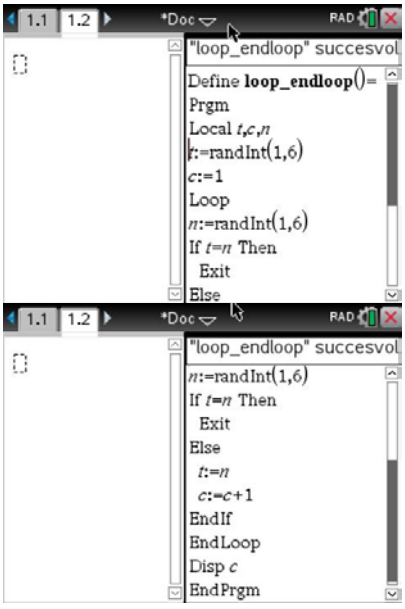
De **Loop…EndLoop**-structuur wordt minstens één keer verwerkt aangezien er geen voorwaarde is waar aan voldaan moet worden om de lus in te gaan.

Exit forceert het programma om de opdracht meteen na **EndLoop** te verwerken.

Voorbeeld: Hoeveel toevalsgetallen van 1 tot en met 6 kun je genereren voordat twee opeenvolgende getallen gelijk zijn aan elkaar?

Bekijk in het programma hier rechts het gebruik van **Loop…EndLoop** met een *voorwaardelijke* **Exit-opdracht**. Wanneer de toevalsgetallengenerator *randInt*(1,6) twee opeenvolgende identieke waarden produceert dan, verlaat het programma de lus om de **Disp**-opdracht aan het end van het programma te verwerken.

Je vindt **Exit** in menu>Overzendingen



Hetzelfde effect kan worden gerealiseerd met een **While**-lus.

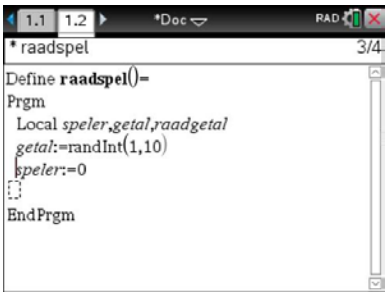
Opmerking over de Syntax: Let in het programma hierboven op het gebruik van de dubbele punt (:) om twee opdrachten op dezelfde regel te scheiden. Dit is alleen gedaan om het hele programma op het scherm te laten passen. Het 'gereserveerde' woord **Exit** is te vinden in menu>Overzendingen.

Docenten Tip: Een voordeel van een **Loop…EndLoop**-structuur is de mogelijkheid om meerdere **Exit**-punten aan te brengen, dit onderwerp moet echter worden vermeden, totdat de leerlingen met meer oefenbladen hebben gewerkt, omdat het kan leiden tot slordig programmeren.

Programma: Raad mijn getal

Om het gebruik van **Loop…EndLoop** te demonstreren**,** ontwikkelen we een spel voor 2 spelers waarin het gaat om het raden van een willekeurig getal tussen 1 en 10. Wanneer een speler het getal raadt, dan eindigt het programma met een bericht aan de winnaar. Een voorbeeld van de uitvoer van zo'n programma zie je hier rechts.

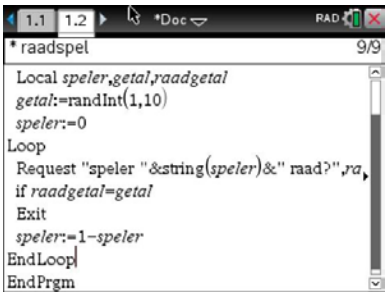
1. Begin met een nieuw programma genaamd **raadspel**.
2. Start met het instellen (initialiseren) van het spel. Identificeer drie variabelen: Het nummer van de speler, het getal van de computer en de gok van de speler, het raadgetal.



De computer kiest een willekeurig getal tussen 1 en 10. We beginnen met de speler met het nummer 0. Dit maakt het eenvoudiger om het spelersnummer te veranderen zoals we snel zullen zien!

Docenten Tip: Denk eraan om **lokale** variabelen te gebruiken, zodat deze de huidige opgave niet vol laten lopen met onnodige variabelen.

3. Bouw vervolgens de lus waarin we eerst de speler vragen om een gok in te voeren. De volledige opdracht is:



Request "Speler "&string(*speler*)&" raad?", *raadgetal*

- Het symbool '**&**' is bedoeld voor het *aaneenschakelen* van tekenreeksen (strings). Het gevraagde deel van een **Request**-opdracht moet een string zijn daarom wordt de *speler*-variabele omgezet in een string met de functie **string( )** die te vinden is in de catalogus.

Docenten Tip: Herhaling: aaneenschakeling is het proces waarmee twee tekenreeksen (strings) worden gecombineerd tot één. De tekens van de tweede tekenreeks woorden toegevoegd aan het eind van de tekens van de eerste reeks.
Je moet de functie **string( )** gebruiken om een numerieke variabele om te zetten in een stringwaarde voordat je deze aaneenschakelt met een andere tekenreeks. De functie **string( )** is alleen te vinden in de de catalogus.

4. Bouw vervolgens de **Exit**-voorwaarde. Wanneer een speler het getal raadt, dan verlaat het programma de lus. Gebruik hier de eenvoudige If-opdracht:

   If raadgetal=getal
   Exit

Bedenk dat **If** zonder **Then** de volgende opdracht uitvoert wanneer de voorwaarde waar is; in andere gevallen wordt de volgende opdracht overgeslagen.

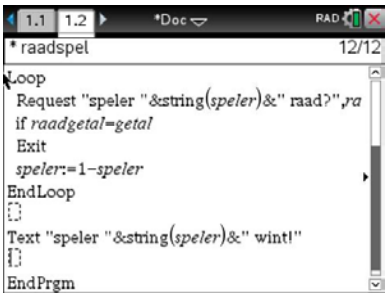
5. Om te wisselen van speler zullen we een slimme opdracht gebruiken:

   speler* := 1 - *speler

deze unieke opdracht schakelt de waarde van *speler* tussen 0 en 1. Dat wil zeggen: wanneer *speler* is 0 dan verandert de opdracht dit in 1 en wanneer *speler* is 1 verandert de opdracht dit in 0.

6. Voeg tenslotte een opdracht toe na de lus, om de winnaar te feliciteren:

   Text "speler "&string(speler)& "wint!"



Tip: Als je het niet prettig vindt om 'speler 0' en 'speler 1' te zien, tel dan gewoon 1 op bij de variabele *speler* in deze opdracht en in de opdracht **Request**. De gebruiker ziet dan een 1 of een 2 hoewel de computer 0 en 1 gebruikt voor de spelers!

7. Bewaar het document voordat je het programma uit gaat voeren voor het geval je vastraakt in een oneindige lus! Zodra het programma begint, moet je doorspelen totdat er een winnaar is. Je kunt niet ontsnappen aan de Request-opdracht.

Docenten Tip: Als je programma's schrijft is het, zoals met elk document, belangrijk om het document regelmatig op te slaan zodat de veranderingen niet verloren gaan als er iets fout gaat!