Cursus MSW-Logo

Hfdst 1: De schildpadwereld

- *Recursie*

Def. Recursie: recursie is het oproepen van dezelfde functie of procedure binnen de functie of procedure

Regelmatige vierhoeken

Voorbeeld in Logo:

```
TO VIERKANT
        REPEAT 4 [FD 100 RT 90]
        VIERKANT
END
```

In deze procedure wordt vierkant opgeroepen in de procedure vierkant. Hierbij zullen we echter in een oneindige lus terecht komen, want als we de procedure vierkant oproepen, zal deze opnieuw de procedure vierkant oproepen en opnieuw, en opnieuw… Er wordt op geen enkel moment een einde gemaakt aan de lus.

Hoe kunnen we dit probleem verhelpen?
Gedachtengang: We kunnen het vierkant groter en groter maken, totdat we een zekere grote bereikt hebben. Bij deze welbepaalde grootte wordt uit de procedure gesprongen.

Implementatie:
We zullen eerst het vierkant groter maken:

```
TO VIERKANT :STAP
        REPEAT 4 [FD :STAP RT 90]
        VIERKANT :STAP + 5
END
```

We geven in deze procedure een variabele mee die de begingrootte van ons vierkant bepaald. Deze procedure is echter terug een oneindige procedure, omdat we nergens een stop voorzien.
Oplossing voor de oneindigheid van onze procedure:
We zullen gebruik maken van IF
IF <voorwaarde> <wat moet gebeuren bij een ware voorwaarde>

In ons voorbeeld zullen we aannemen dat getopt moet worden bij een lengte van een zijde groter dan 150. We voegen dus de volgende regel toe aan ons programma:
IF :STAP > 150 [STOP]
Waar moeten we deze regel toevoegen?

Twee mogelijkheden:

1)
```
TO VIERKANT :STAP
        IF :STAP > 150 [STOP]
        REPEAT 4 [FD :STAP RT 90]
```

```
        VIERKANT :STAP + 5
    END
```

In deze procedure zal NOOIT een vierkant groter dan 150 getekend worden. Als we dus als beginargument 200
opgeven, zal geen enkel vierkant getekend worden.

2)
```
    TO VIERKANT :STAP
        REPEAT 4 [FD :STAP RT 90]
        IF :STAP > 150 [STOP]
        VIERKANT :STAP + 5
    END
```

In deze procedure zal precies één vierkant groter dan 150 getekend worden. Als we dus als beginargument 200
opgeven, zal een vierkant met lengte van de zijde 200 getekend worden en zal direct daarna gestopt worden.

Regelmatige figuren

Het tekenen van regelmatige figuren kan op de volgende manier gebeuren:
```
TO FIGUUR :AANTAL
    REPEAT :AANTAL  [FD 100 RT 360/:AANTAL]
END
```
Deze procedure tekent een regelmatige figuur waarbij het aantal hoeken meegegeven wordt.

We proberen via recursie de volgende doelstelling te bereiken:
Wanneer we in de commando regel het volgende schrijven:
FIGUUR 10
Dan krijgen we als output: een regelmatige 10,9,…,3-hoek.

We moeten dus aan de procedure een stop-regel toevoegen en een recursieve oproep. We moeten de procedure terug oproepen, met een getal die 1 kleiner is dan het opgegeven getal. Dus:
FIGUUR :AANTAL – 1
Om de procedure te stoppen moeten we de variabele :AANTAL controleren. Als deze variabele kleiner is dan 3 moeten we stoppen. We moeten opletten waar we deze regel gaan toevoegen. We moeten deze regel invoegen voordat we tekenen, anders kan het gebeuren dat we toch proberen een regelmatig 2-hoek (???) proberen te tekenen. De regel is de volgende:
IF :AANTAL < 3 [STOP]

Als we dit alles tesamen voegen krijgen we het volgende programma'tje:
```
TO FIGUUR :AANTAL
    IF :AANTAL < 3 [STOP]
    REPEAT :AANTAL  [FD 100 RT 360/:AANTAL]
    FIGUUR :AANTAL – 1
END
```

Tekenen van een boom

We kunnen op een recursieve manier een boom tekenen:

```
TO BOOM :STAP
        IF :STAP < 18 [STOP]
        FD :STAP
        RT 45
        BOOM (4 * :STAP / 5)
        LT 90
        BOOM (4 * :STAP / 5)
        RT 45
        BK :STAP
END
```

Uitleg van de procedure:
->De lengte van de stam wordt meegegeven bij het aanroepen van de procedure. Deze variabele zal gebruikt worden om de lengte van de takken te bepalen
->We controleren of de lengte van de variabele minimaal 18 is. Is dit niet zo, dan wordt uit de procedure gesprongen.
->We tekenen de eerste tak (hier stam). Op het einde van de procedure moeten we diezelfde lengte ook terugkeren, zodanig dat de turtle terug in zijn beginpositie staat.
->We draaien 45° naar rechts. Dit om op onze eerste stam een tak te tekenen.
->We tekenen de takken aan de rechtse kant van de stam via recursie. Onze tak die we op de stam zetten zal 4/5 bedragen van onze oorspronkelijke lengte.
->We draaien 90° naar links, en tekenen via recursie de volledige linker kant van onze boom.
->We draaien 45° naar rechts, zodat onze turtle terug naar zijn oorspronkelijke richting kijkt.

Dit is een niet zo gemakkelijke procedure en vraagt enig denkwerk om deze volledige procedure te doorgronden. We kunnen dit best doen door deze procedure op te roepen met een niet te groot getal als lengte voor de stam en door ons programma te lopen via de step functie die MSW-Logo aanbiedt.

Het invoegen van random-variabelen:
Met RANDOM kunnen we een willekeurig getal bepalen tussen 0 en het opgegeven getal.
Vb.:
RANDOM 10: geeft een getal terug tussen 0 en **9**.
Wanneer we nu een getal wensen tussen 1 en 10 zullen we dit op de volgende manier bereiken:
(RANDOM 10) + 1
Dit lijkt misschien zeer eenvoudig, maar heeft vele nuttige toepassingen.

We zullen nu terug onze boom tekenen, maar we zullen de lengte van de takken laten afhangen van random-variabelen. Laat ons zeggen, dat de tak die via recursie getekend moet worden ofwel 4/5 van de oorspronkelijke tak moet zijn, ofwel 3/5. Ons programma wordt dan op de volgende manier aangepast:

```
TO BOOM :STAP
        IF :STAP < 18 [STOP]
        FD :STAP
        RT 25
        BOOM (((RANDOM 2)+3) * :STAP / 5)
        LT 50
```

```
        BOOM (((RANDOM 2)+3) * :STAP / 5)
        RT 25
        BK :STAP
END
```

Omdat de boom duidelijk zou zijn, werden de coördinaten van de roatatie naar rechts en links ook aangepast. Dit kunnen we doen zolang de som 0 is. Dus (RT 25)+(LT 50)+(RT 25)=0; zo-ook (zoals in het eerste voorbeeld:) (RT 45)+(LT 90)+(RT 45)=0.

- *Pen- en kleuropdrachten*

Inleiding

Bij pen- en kleuropdrachten kunnen we gebruik maken van een kleurvector of een kleurgetal.

Een kleurvector ziet er als volgt uit:
Zwart: [000][000][000]
Wit: [255][255][255]
Rood: [255][000][000]
Groen: [000][255][000]
Blauw:[000][000][255]
We kunnen ook combinaties maken om zo andere kleuren te bekomen.

Een kleurgetal ziet er als volgt uit:
0 -> [000 000 000]
1 -> [000 000 255]
2 -> [000 255 000]
3 -> [000 255 255]
4 -> [255 000 000]
5 -> [255 000 255]
6 -> [255 255 000]
7 -> [255 255 255]
8 -> [155 96 59]
9 -> [197 136 18]
10 -> [100 162 64]
11 -> [120 187 187]
12 -> [255 149 119]
13 -> [144 113 208]
14 -> [255 163 000]
15 -> [183 183 183]

Om de achtergrond van kleur te veranderen kunnen we typen:
        SETSCREENCOLOR kleurgetal
        SETSCREENCOLOR [kleurvector]
Of de verkorte notatie:
        SETSC kleurgetal
        SETSC [kleurvector]
Om de pen van kleur te veranderen kunnen we de volgende opdracht geven:
        SETPENCOLOR kleurgetal
        SETPENCOLOR [kleurvector]

Of de verkorte notatie:

```
SETPC kleurgetal
SETPC [kleurvector]
```

Tekenen van een kubus

We tekenen een kubus waarbij het eerste argument de lengte van de ribbe is en het tweede argument de kleurvector of kleurgetal:
Vb.: KUBUS 100 0
Zal een kubus tekenen met ribbe 100 in het kleur 0 (=zwart).

```
TO KUBUS :S :K
        SETPC :K
        REPEAT :S [FD :S SETX XCOR + 1 BK :S]
END
```

We kunnen deze procedure verkorten door gebruik te maken van fill. Waarbij SETFC de kleur opgeeft, waarmee we de kubus zullen inkeuren.

```
TO KUBUS :S :K
        SETPC :K
        REPEAT 4 [FD :S RT 90]
        RT 45 FD :S
        SETFC :K
        FILL
END
```

Deze procedure heeft exact dezelfde oplossing wanneer we starten met een leeg scherm. Het verschil tussen de twee procedures is, dat we bij de eerste 704 calls hebben en bij de tweede 15 calls (en 18 wanneer we ook wensen dat onze turtle terug keert naar onze beginpositie). We moeten echter wel opletten met deze vereenvoudiging. Voeren we deze tweede procedure twee maal na elkaar uit, dan zal dit niet het gewenste resultaat hebben. Verifieer dit eens.

Meerdere turtles

We kunnen ook werken met meerdere turtles. We kunnen een tweede trurtle aanroepen door:
SETTURTLE 1
Onze eerste turtle is eigenlijk turtle 0. Deze tweede turtle zal op onze beginpositie verschijnen. We kunnen meerdere turtles gebruiken om een tekening mooier in uitvoering te maken.

- *Samenvatting*

We kunnen nu al deze aangebrachte methoden en commando's combineren in het volgende programmaatje:

```
TO MEERDERE :AANTAL :GROOTTE
        IF :AANTAL < 1 [STOP]
        SETTURTLE :AANTAL SETX :AANTAL * 100 SETPC :AANTAL BOOM
:GROOTTE
```

```
    MEERDERE (:AANTAL – 1) :GROOTTE
END
```

MEERDERE 5 50 zal dus 5 bomen tekenen in de kleuren 1,2,3,4,5. Hierbij zal gebruik gemaakt worden van de reeds eerder geïmplementeerde procedure BOOM waarbij het argument :GROOTTE wordt meegegeven.

Hfdst 2: De taalwereld

Inleiding

In de taalwereld hebben we enkele manieren om woorden uit te printen:
-PRINT OF PR
-TYPE
-FIRST
-LAST
-BUTFIRST of BF
-BUTLAST of BL

Om iets naar ons scherm te schrijven kunnen we de volgende twee methoden hanteren:
PR “Hallo
PR [Hallo]

We krijgen twee maal dezelfde uitvoer, maar de eerst uitvoer is woord-georiënteerd en de tweede is lijst-georiënteerd. Om een spatie af te drukken door gebruik te maken van “, zullen voor de spatie een \ zetten; dus:
PR “Hallo\ allemaal
Zal een correcte uitvoer geven, wat niet het geval is, zonder de \.
We zullen ook gebruik maken van \ om een [en] af te drukken in lijsten. Dus:
PR [Hallo \[allemaal]
Zal als output:
Hallo [allemaal
geven
Dit kan soms nuttig zijn voor bepaalde doeleinden.

We kunnen ook het commando type gebruiken om uitvoer naar het scherm te produceren. Het is wel zo dat TYPE niet rechtstreeks naar het scherm schrijft, maar wacht op een commando PR om zijn volledige buffer naar het scherm uit te schrijven.

Bv.:
TYPE “Hallo\ allemaal
OUTPUT:
PR [gelukkig nieuwjaar]
OUTPUT: Hallo allemaal gelukkig nieuwjaar

Kleine toepassing

Een laatste voorbeeldje, waarbij we recursie toepassen in de taalwereld is het volgende: we proberen een procedure te schrijven, waarbij het woord omgekeerd wordt uitgeschreven. Bijvoorbeeld: ACHTERUIT “logo wordt: ogol

We maken in de procedure gebruik van EMPTYP om te controleren of dat de invoer wel effectief leeg is.

```
TO ACHTERUIT :INVOER
        IF EMPTYP :INVOER [PR " STOP]
        TYPE LAST :INVOER
        ACHTERUIT BL :INVOER
END
```