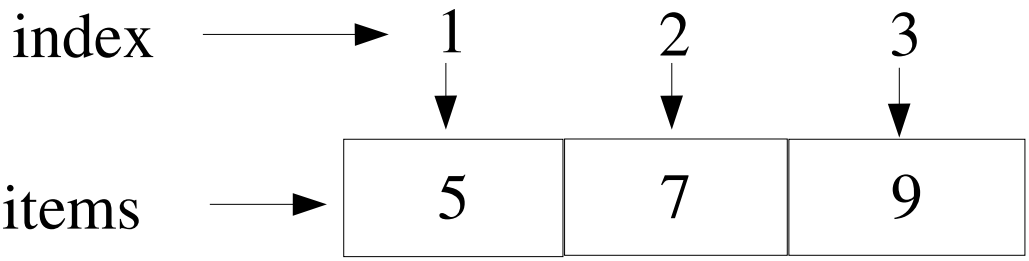
Arrays in LOGO

Een array Is één variabele die verschillende items bevat. Iedere item heeft een waarde. De items staan in een bepaalde volgorde. Het is dus een **rij** van waarden. De positie van iedere item in die rij noemen we de **index**.



In LOGO heeft de eerste item van de array standaard index 1.

Notatie
Zo maken we een array:

```
Make “array1 {7 8 5 1}    ;vier bepaalde elementen
Make “array2 (array 100)  ;honderd onbepaalde elementen
```

Zo vragen we de waarde van een item op

```
(item 3 array1)          ;geeft item met index 3 (=5)
```

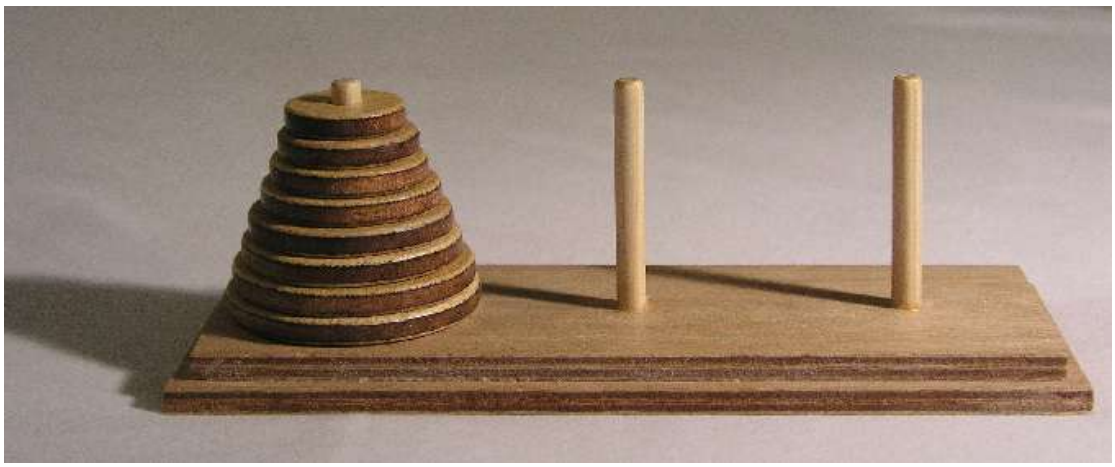
Zo veranderen we de waarde van een item

```
setItem 50 array2 100    ;zet item 50 op 100
```

De Torens Van Hanoi

1. Inleiding: Geschiedenis en beschrijving van het probleem

Het probleem van de Toren van Hanoi is uitgevonden door de Franse Wiskundige Edouard Lucas in 1883. Het is een spel of een puzzel dat hij beschreef in zijn boek "Récréations Mathématiques". Men heeft er ook een echt spelletje van gemaakt voor kinderen, dat verkocht werd in Frankrijk (zie foto).



Lucas spreekt over een "Legende". De legende vertelt over een Hindoe-tempel waarvan de priesters, de Brahmanen, zich bezighouden met het verplaatsen van een toren van 64 gouden schijven. Volgens de legende komt de wereld tot een einde als het werk af is.

Er staan drie identieke staven naast elkaar. Over een van de staven zit bij de aanvang van het spel een aantal schijven van afnemende diameter met een gat in het midden als een piramide op elkaar gestapeld. (De grootste schijf ligt onder.)

Het gaat erom de toren van schijven te verplaatsen naar een andere staaf, waarbij de volgende regels gelden:

1. er mag slechts 1 schijf tegelijk worden verplaatst
2. nooit mag een grotere schijf op een kleinere rusten

Om praktische redenen heeft de toren meestal 8 schijfjes, omdat dit aantal binnen een minuut of 6 op te lossen is. Ieder schijfje meer verdubbelt de minimale oplostijd (we zullen later zien waarom).

Aannemend dat de priesters 1 schijf per seconde zouden verplaatsen, zou

het ongeveer 2^{64}, dat is ongeveer 1.8×10^{19} seconden duren om de puzzel af te maken. Dit komt overeen met ongeveer 580.000.000.000 jaar, ruwweg veertig maal langer dan de geschatte leeftijd van het universum.

Op de computer kunnen we probleem voor een beperkt aantal schijven vrij snel oplossen. Het spel kan intuïtief gespeeld worden, maar om de oplossing te programmeren moeten we een patroon zoeken in onze handelingen.

2. Het recursieve algoritme

In de klas hebben we het spel eerst intuïtief gespeeld met vier schijven. Het is ons toen opgevallen dat we eerst een torentje van twee hebben gemaakt. Vervolgens hebben we dat zelfde torentje nog eens verplaatst om een torentje van drie te maken. Om uiteindelijk het torentje van vier terug op te bouwen moesten we het torentje van drie verplaatsen, en om dat nog eens opnieuw te doen moesten we weeral twee keer het torentje van twee verplaatsen... Je kan misschien wel reeds "aanvoelen" waarom het dubbel zo lang duurt om één schijf méér te verplaatsen...

Hierboven worden onze intuïtieve handelingen chronologisch beschreven, maar je kan ze ook op een andere manier bekijken: Om een torentje van vier te verplaatsen, hebben we tweemaal een torentje van drie verplaats, om een torentje van drie te verplaatsen, hebben we tweemaal een torentje van twee verplaatst. Als je het zo bekijkt, zie je in dat het algoritme recursief werkt: om een torentje van n schijven te verplaatsen, hebben we telkens tweemaal een torentje van n-1 schijven verplaatst. Met telkens als doel de onderste schijf ook mee te kunnen verplaatsen.

We kunnen ons algoritme als volgt samenvatten (in pseudocode):

```
To VerplaatsToren :n
     VerplaatsToren :n-1 naar hulpstaaf
     Verplaats onderste schijf naar doelstaaf
     VerplaatsToren :n-1 van hulpstaaf naar doelstaaf
end
```

Hierbij werd aangenomen dat n>1. Als n=1 kunnen we de schijf immers verplaatsen in één stap! Het is nu gemakkelijk in te zien dat de tijd ongeveer verdubbelt als we n met één verhogen.

3. Implementatie van Grafische Voorstelling in LOGO

De grafische voorstelling van dit probleem in LOGO is complexer dan het recursieve algoritme alleen. Daarom maken we een analyse van ons probleem door het op te splitsen in deelproblemen.

- We willen de staven en schijven op het scherm tekenen. (De beginsituatie)
- We willen een willekeurig aantal staven kunnen verplaatsen.
- We willen één staaf kunnen verplaatsen. Hiervoor moeten we ze kunnen tekenen, maar ook uitwissen.

We komen uiteindelijk tot het volgende schema: (zie volgende pagina)

3.1 Staaf: Tekenen van een staaf

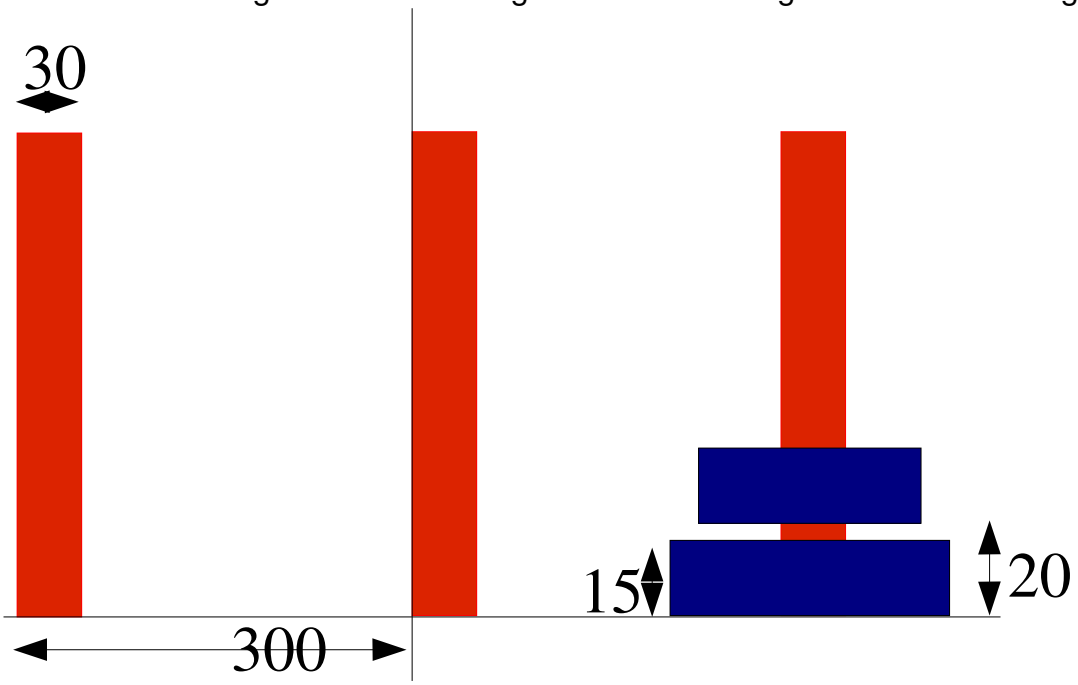
to Staaf :staaf :aantalschijven

Overzicht van de parameters
Staaf is een waarde 1,2 of 3 (bepaalt de positie),
aantalschijven bepaalt de hoogte van de staaf.

Opmerking: Voor het practische gemak maken we gebruik van devolgende procedure die we reeds in een andere les hebben geprogrammeerd:

```
to positie :x :y
     pu
     setxy :x :y
     pd
end
```

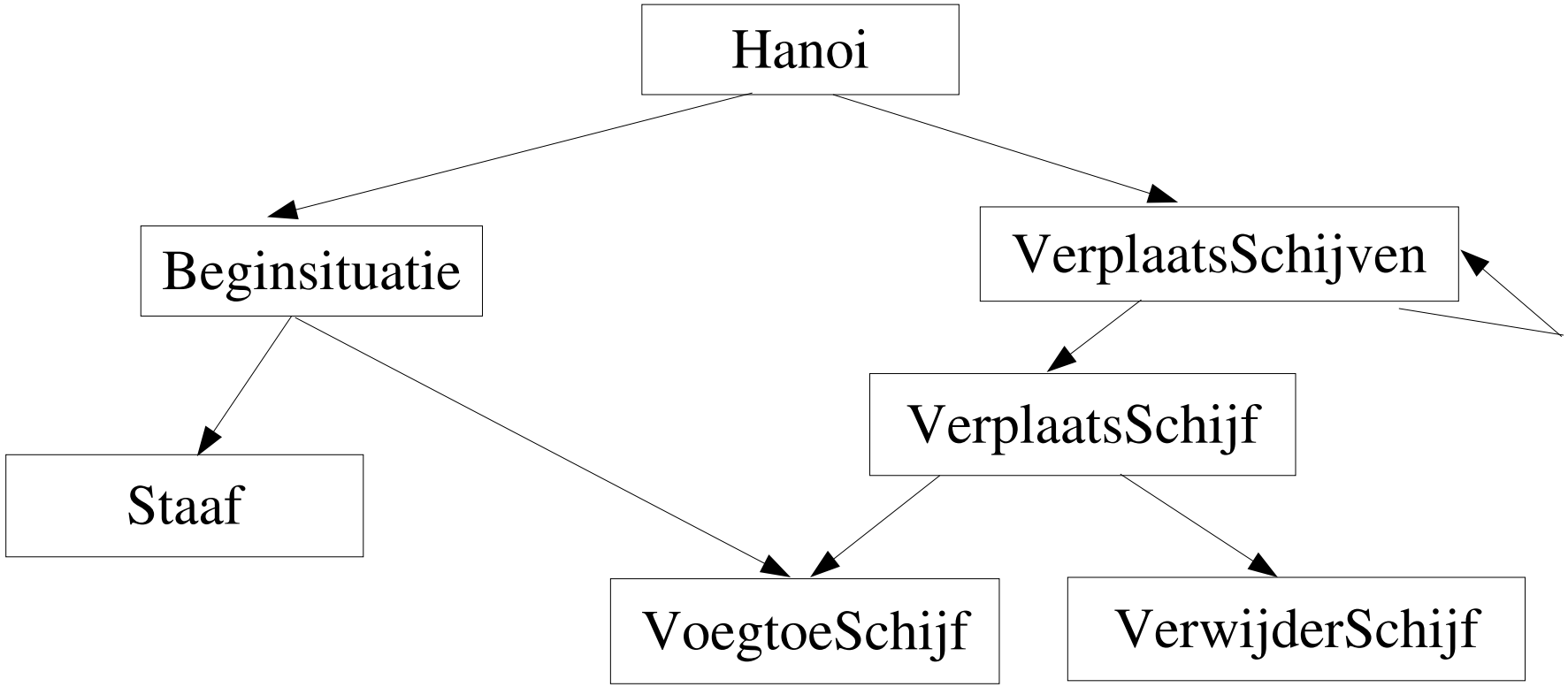
Ziehier de afmetingen die we zullen gebruiken voor de grafische voorstelling:



We bekomen devolgende code:

```
to staaf :staaf :aantalschijven
  setpc [255 0 0] setfc [255 0 0]
```

```
  positie 300*(:staaf-2) 0
Anni  repeat 2 [fd 20+:aantalschijven*20 rt 90 fd 30 rt
90]
  rt 10 fd 5 fill lt 10
end
```


Hanoi
Beginsituatie
VerplaatsSchijven
Staaf
VerplaatsSchijf
VoegtoeSchijf
VerwijderSchijf

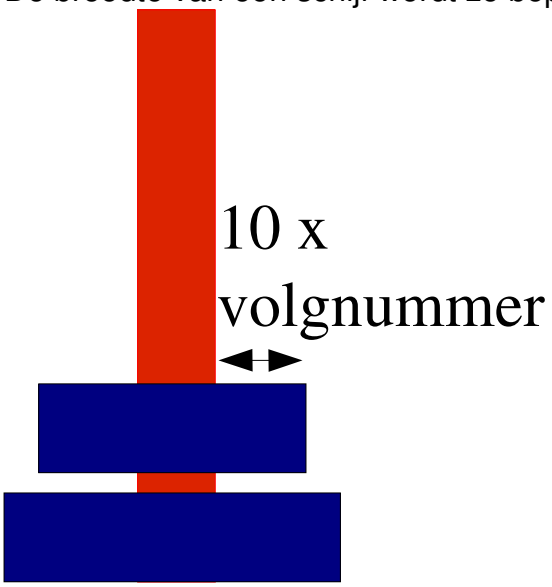
3.2 VoegtoeSchijf: toevoegen van een schijf:

to voegtoeSchijf :staaf :schijf

Overzicht van de parameters
De procedure moet weten op welke staaf de schijf moet geplaatst worden! Dit is een nummer van 1 tot 3. De procedure moet ook weten welke schijf (=grootte) er moet geplaatst worden. 1 is de kleinste, 2 is groter, enz...

We moeten ook weten op welke positie de schijf moet getekend worden. We moeten dus weten hoe hoog de toren op de staaf al is, want de volgende schijf moet daar bovenop getekend worden. Als we een procedure schrijven om een schijf te verplaatsen, dan moet die werken op ieder moment, voor om het even welk aantal schijven op de staven zitten. We moeten de hoogtes van de torens dus bijhouden in variabelen.
E r zijn drie staven. We zouden dan drie variabelen kunnen gebruiken. Wanneer we dan een schijf bij een willekeurige staaf toevoegen, moeten we drie if-structuren schrijven om de juiste variabele op te vragen! Als we de waarde dan weer veranderen, moeten we weer drie if-structuren gebruiken! Daarom kunnen we beter gebruik maken van een array van drie items.

De breedte van een schijf wordt zo bepaald:



We bekomen devolgende code:

```
to voegtoeSchijf :staaf :schijf
  setpc [0 0 255] setfc [0 0 255]
  positie 300*(:staaf-2)-(10*:schijf)
              (item :staaf :staven)*20
  repeat 2 [fd 15 rt 90 fd 30+20*:schijf rt 90]
```

```
  ; Er moeten drie vakken ingevuld worden (links van de
  ; staaf, binnenin en rechts)
  rt 10 fd 5
  fill
  positie 300*(:staaf-2)+5 (item :staaf :staven)*20+5
  fill
  positie 300*(:staaf-2)+35 (item :staaf :staven)*20+5
  fill
  lt 10
  setItem :staaf :staven (item :staaf :staven)+1
end
```

Merk op dat om deze procedure te kunnen uitvoeren we eerst de array staven {0 0 0} moeten aanmaken en de staven tekenen. Dit zullen we laten gebeuren in onze beginSituatie.

3.3 BeginSituatie

We zullen de schijven initiëel toevoegen aan de middenste staaf.

```
to beginSituatie :aantalschijven
  cs ht
  make "staven {0 0 0}
  staaf 1 :aantalschijven
  staaf 2 :aantalschijven
  staaf 3 :aantalschijven
  repeat :aantalschijven [
     voegtoeSchijf 2 :aantalschijven
     make "aantalschijven :aantalschijven-1
  ]
end
```

3.4 VerwijderSchijf: verwijderen van een schijf.

to verwijderSchijf :staaf

Het zal steeds de bovenste schijf zijn die van de staaf verwijderd wordt. We maken dus ook gebruik van de array {staven}.

[Overzicht van de parameters
De procedure moet slechts weten van welke staaf de bovenste schijf moet verwijderd worden.

We bekomen devolgende code:

```
to verwijderSchijf :staaf
  setItem :staaf :staven (item :staaf :staven)-1
  ; Eerst wissen we de schijf in het wit.
```

```
  setfc [255 255 255]
  positie 300*(:staaf-2) (item :staaf :staven)*20+5
  fill
  ; Dan moeten we nog het uitgewiste stuk van de paal
  ; terugkleuren
  positie 300*(:staaf-2) (item :staaf :staven)*20
  setpc [255 0 0] setfc [255 0 0]
  repeat 2 [fd 15 rt 90 fd 30 rt 90]
  rt 10 fd 5 fill lt 10
end
```

3.5 VerplaatsSchijf: het verplaatsen van een schijf.

to verplaatsSchijf :bronstaaf :doelstaaf :schijf

Overzicht van de parameters
We willen een schijf van een bronstaaf naar een doelstaaf verplaatsen. Bronstaaf en doelstaaf zijn nummers van 1 tot 3. Schijf is de volgnummer (grootte) van de schijf.

We zullen dit doet door eerst de bovenste schijf van bronstaaf te verwijderen. Vervolgens voegen we de schijf toe aan doelstaaf.

Opmerking:
Let er op dat deze procedure er zomaar vanuit mag gaan dat de bovenste schijf van bronstaaf dezelfde grootte heeft als :schijf. De gebruiker van de procedure is hiervoor verantwoordelijk. De procedure controleert bovendien ook niet of het wel over een “goede” verplaatsing is (geen grotere op een kleinere). Als we ons recursief algoritme toepassen is dit echter geen enkel probleem!

We bekomen devolgende code:

```
to verplaatsSchijf :bronstaaf :doelstaaf :schijf
  verwijderSchijf :bronstaaf
  voegtoeSchijf :doelstaaf :schijf
end
```

3.6 VerplaatsSchijven: het verplaatsen van meerdere schijven

to verplaatsSchijven :aantalschijven :bronstaaf :doelstaaf :hulpstaaf

Overzicht van de parameters
We willen een torentje van grootte aantalschijven verplaatsen van de bronstaaf naar de doelstaaf, met de behulp van een hulpstaaf.

Lettend op ons recursief algoritme uit paragraaf 2 bekomen we devolgende

code. Let wel op dat we if-structuren moeten aanleggen om niet in een oneindige lus te geraken! Wanneer aantalschijven één is, moeten en kunnen we niet meer recursief handelen, we moeten dan slechts één stap meer doen.

```
to verplaatsSchijven :aantalschijven :bronstaaf
                     :doelstaaf :hulpstaaf
  if :aantalschijven > 1 [
    verplaatsSchijven :aantalschijven-1 :bronstaaf
                      :hulpstaaf :doelstaaf
  ]
  verplaatsSchijf :bronstaaf :doelstaaf :aantalschijven
  if :aantalschijven > 1 [
    verplaatsSchijven :aantalschijven-1 :hulpstaaf
                      :doelstaaf :bronstaaf
  ]
end
```

3.7 Hanoi: de hoofdprocedure

```
to hanoi :aantalschijven
  beginSituatie :aantalschijven
  verplaatsSchijven :aantalschijven 2 3 1
end
```

Ons probleem is nu opgelost. Typ eens “hanoi 8” in in het commandovenster, en je computer zal het spel bliksemsnel tot een goed einde brengen..

Opmerking: je kan de verschillende stappen van de computer beter bekijken als je de procedure verplaatsSchijf vertraagt met het commando “wait”.

```
to verplaatsSchijf :bronstaaf :doelstaaf :schijf
  wait 10
  verwijderSchijf :bronstaaf
  voegtoeSchijf :doelstaaf :schijf
end
```