SB Console Deel 2

Inleiding

In deze les is het vervolg van les 1 Small Basic Console. In deze les zullen de volgende onderwerpen besproken worden:

- Iteraties
- Arrays
- Constanten
- Initialisatie
- Functies – subroutines

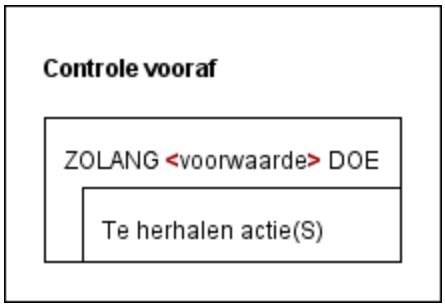
Iteratie (herhaallussen)

Een verzameling opdrachten worden aantal keer herhaald zolang een bepaalde **criteria** (voorwaarde) voldoet.

Door herhaallussen kunnen we een computer een verzameling opdrachten laten herhalen zo vaak als wij dat willen. Tevens wordt de broncode korter en overzichtelijker.

Er zijn drie herhaallussen:

De herhaal lus met **controle vooraf**. Deze wordt vooral gebruikt als je niet weet hoeveel keer iets gedaan moet worden maar alleen weet wanneer het met de herhaling moet stoppen.



Zolang de voorwaarde geldt worden de instructies binnen de lus herhaald.

In Small Basic is de opdracht voor het maken van een iteratie lus met controle vooraf in de afbeelding te zien.

```
1 A = 0
2 While A < 10
3   TextWindow.WriteLine("A = " + A)
4   A = A + 1
5 EndWhile
6
```

De opdracht bestaat uit twee delen.

1. **While** <voorwaarde>
2. **EndWhile**

LET OP! Oneindige lus
Zonder de opdracht **A = A +1** zal het programma oneindig door gaan, daar de waarde van A nooit 10 of groter zal worden.

SB Console Deel 2

Ooggetuigenverslag:

- Om inzicht te krijgen in de werking en voor controle is het noodzakelijk om de opdrachten na te lopen.
- Doel van een ooggetuigenverslag is tweedelig:
  - Als de werking van het PSD bekend is: controle of het PSD juist is.
  - Als de werking van het PSD onbekend is: onderzoeken wat het PSD doet.

Stel dat bij de PSD hiernaast de variabelen de volgende waarden hebben:

- A = 4
- B = 5

Voor A = 4 en B =5 geldt dat de functie wordt:
Y = AX + B ➡ Y = 4X + 5

Het ooggetuigen verslag is dan als volgt:

X	Y
0	5
1	9
2	13
3	17
4	21
...	
10	45

PSD Iteratie oef 2

```
SCHRIJF "Dit programma berekent de Y voor"
SCHRIJF "voor X is 0 to en met 10."
SCHRIJF "Voor de functiue Y = AX + B"
LEES "Voer een getal in voor de variabele A:"; A
LEES "Voer een getal in voor de v ariabele B:";B
X := 0
ZOLANG X <= 10 DOE
    Y = A * X + B
    SCHRIJF "Voor X = ";X;" geldt dat Y = "; Y
    X := X + 1
```

SB Console Deel 2

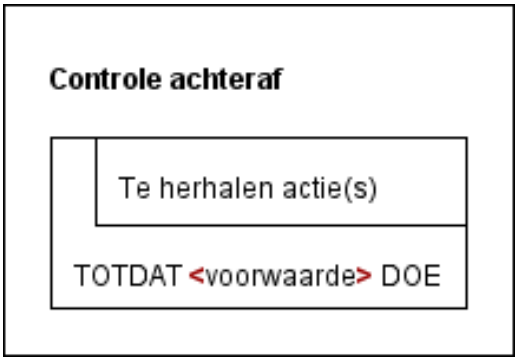
Vooruit lezen

Bij de iteratie structuur met controle vooraf moet er twee keer om invoer gevraagd worden.

- Voor dat de lus in gegaan wordt.
- Aan het einde van de lus.

Controle Achteraf

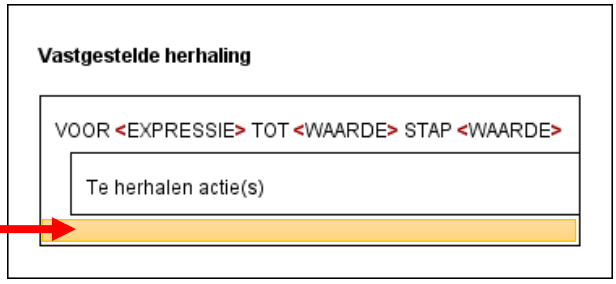
Bij deze herhaal-structuur worden de opdrachten eerst uitgevoerd en daarna wordt er een controle uitgevoerd of de lus herhaald moet worden.
Deze structuur zorgt vaak voor problemen in programma's daarom wordt het zelden of nooit gebruikt.
In moderne talen komt het niet meer voor.



Vastgestelde herhaling

Deze herhaal-structuur herhaalt de opdrachten een vaste aantal keren.

```
VOOR x=1 TOT 10
          Herhaal acties
Einde VOOR
```



Dan worden de acties 10 keer herhaald.

SB Console Deel 2

In SB is de opdracht voor het maken van een iteratie lus met vastgestelde herhaling in de afbeelding te zien.

De opdracht bestaat uit twee delen:

1. For <start> To <stop>
2. EndFor

```
For A =  1 To 10
  TextWindow.WriteLine("A = " + A)
EndFor
```

Als men de code van de controle vooraf lus vergelijkt met die van de vastgestelde lus, dan is te zien dat bij de laatste de instructie A = A + 1 niet meer aanwezig is.

```
A = 0
While A < 10
  TextWindow.WriteLine("A = " + A)
  A = A + 1
EndWhile
```

Deze instructie wordt automatisch uitgevoerd door de compiler (vertaalprogramma) wat als voordeel heeft dat de lus altijd een keer zal eindigen.

```
For A =  1 To 10
  TextWindow.WriteLine("A = " + A)
EndFor
```

Iteratie met een stap

Doormiddel van de stap (step) opdracht is het mogelijk om het aantal keren dat de lus wordt uitgevoerd aan te passen.

- Een positieve waarde bij de step zorgt dat de lus zoveel stappen overslaat en verder telt.

```
For A =  1 To 10 Step 2
  TextWindow.WriteLine("A = " + A)
EndFor
```

- Een negatieve waarde doet hetzelfde, echter wordt er nu afgeteld

```
For A =  10 To 1 Step -1
  TextWindow.WriteLine("A = " + A)
EndFor
```

SB Console Deel 2

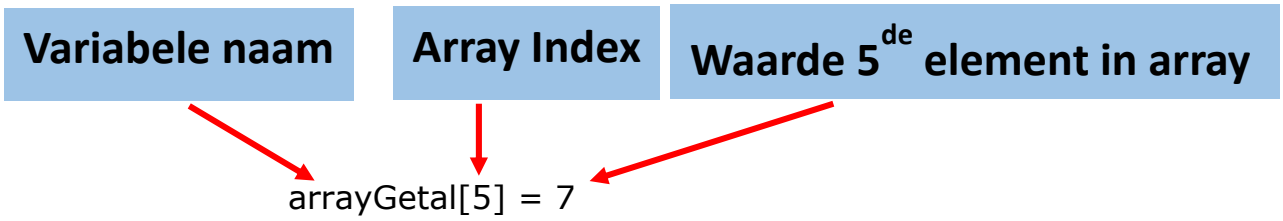
De instructie RANDOM

Voor het genereren van willekeurige getallen gebruik je in de PSD structuur de opdracht:

- RANDOM(waarde)
  - Voor waarde vul je een geheel getal in van nul tot groter.
  - Het getal dat voor waarde wordt ingevuld bepaald het grootste willekeurige getal dat gegenereerd kan worden.
    - Dus RANDOM(10) genereerd een willekeurig getal tussen 0 en 10.

Array's

Tot nu toe hebben we gewerkt met variabelen die maar één waarde kunnen bevatten. Soms wil je echter een variabele hebben die een groep waarden kan onthouden. Zo'n variabele heet een **Array**.

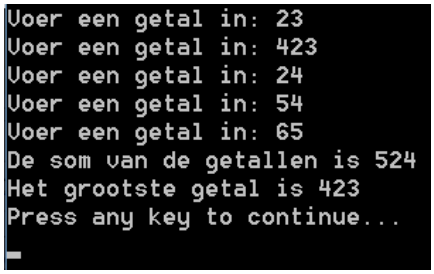


Een Array is een lijst van waarde met dezelfde naam.
Er wordt toegang tot elke waarde van de array gekregen door de index van de array. De Array index begint standaard (default) bij nul te tellen.
Arrays kunnen getallen of tekst bevatten.

Arrays doorlopen

In de afbeelding hiernaast zie je de uitvoer van een programma.

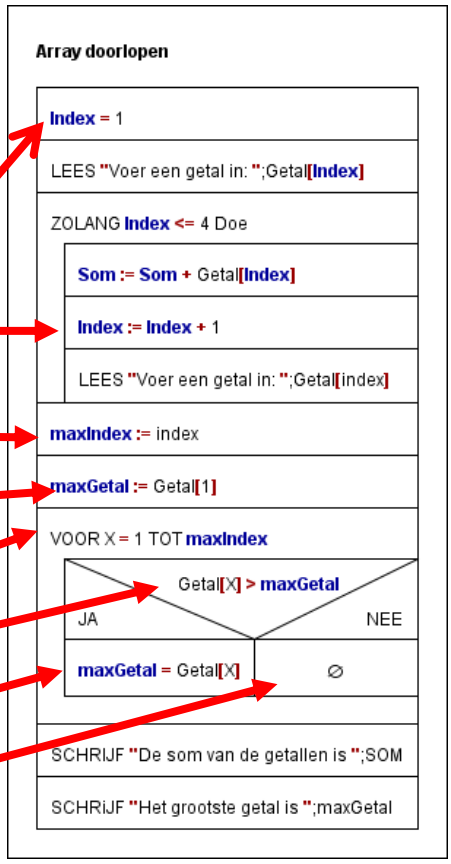
- Het programma vraagt de gebruiker om 5 willekeurige getallen in te voeren.
- Daarna Drukt het programma de som van de 5 getallen af op het scherm.
- Tevens bepaald het welk getal het grootste was en drukt deze af op het scherm.



SB Console Deel 2

De index

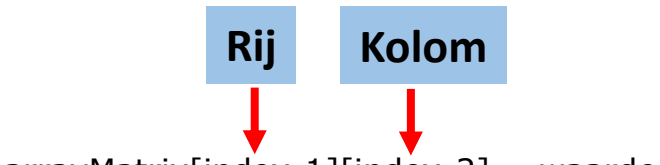
De index van een array wordt gebruikt om een array te doorlopen en elk individueel element in de lijst te benaderen.



- De index geeft het element van de array aan. We beginnen bij 1.
- De index moet iedere keer verhoogd worden.
- Het aantal elementen moet worden bijgehouden. Dit gebeurt in maxIndex.
- maxGetal krijgt de waarde van het eerste element in de lijst.
- We gebruiken een voor-lus om de lijst te doorlopen tot en met maxIndex.
- We gebruiken een select om te bepalen welke het grootste getal is.
- Alleen als het element groter is dan maxGetal moet maxGetal de waarde krijgen van dat element.
- Anders gebeurt er niets.

Matrix Array (meerdimensionale array)

Naast de gewone array kunnen ook array's met meerder dimensies gemaakt worden, of wel matrix array's.



- Een Array is een lijst van waarde met dezelfde naam.
- Er wordt toegang tot elke waarde van de array gekregen door de index van de array.
- Arrays kunnen getallen of tekst bevatten.
- Verder gelden dezelfde regels als bij normale variabelen.

SB Console Deel 2

Constanten

- Constanten zijn variabelen die nooit van waarde veranderen
    - In de wiskunde bijvoorbeeld het getal π (Pi = 3.14159…)
- Schrijf constanten altijd helemaal in hoofdletters
- Constanten moeten altijd helemaal aan het begin worden gedefinieerd van het programma.
- Wijzig je de waarde van de constante dan geldt deze wijziging voor het gehele programma.

Initialisatie

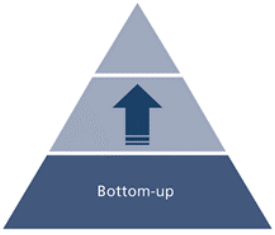
Wanneer programma's groter worden is het belangrijk om structuur in de bron code aan te brengen.

- Initialisatie
    - Helemaal aan het begin van het programma.
    - Variabelen worden hier gedefinieerd met een initiële waarde.
- Voordelen
    - Fouten voorkomen en opsporen gaat beter.
    - Programma wordt beter leesbaar.

Top ⇒ Down en Bottom ⇒ Up

Wanneer een programma wordt ontworpen wordt er op twee manieren gewerkt.

- Top ⇒ Down
    - Van boven af wordt naar het totale probleem gekeken.
    - Er vindt verfijning van het probleem plaats naar beneden tot het totale probleem opgelost is.
- Bottom ⇒ Up
    - Van beneden naar boven worden de verschillende delen ontwikkeld.

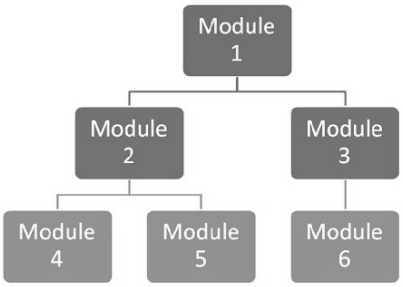


SB Console Deel 2

Functies – Subroutines – Methoden

In alle programmeertalen zijn er structuren om kleinere stukken programma te schrijven die deel uitmaken van een programma.

- Functies – Subroutines - Method
  - Kleinere stukken bron code waarmee structuur wordt aangebracht in een programma.
  - Voor het oplossen van een opdracht in kleinere makkelijker op te lossen deel opdrachten.
  - Een verzameling code onder één naam voor uit uitvoeren van een bepaalde functie.

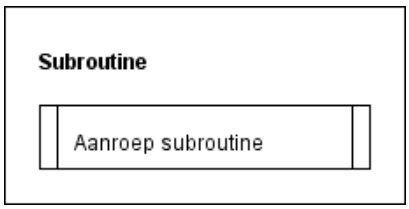


- Voordelen
  - Structuur in programma
  - Her bruikbaar
    - Ze kunnen meerder keren worden aangeroepen waardoor bron code korter wordt en overzichtelijker
  - Flexibeler
    - Bron code makkelijker aan te passen
  - Debuggen
    - Fouten sneller op te lossen

Functie aanroep in PSD en SB

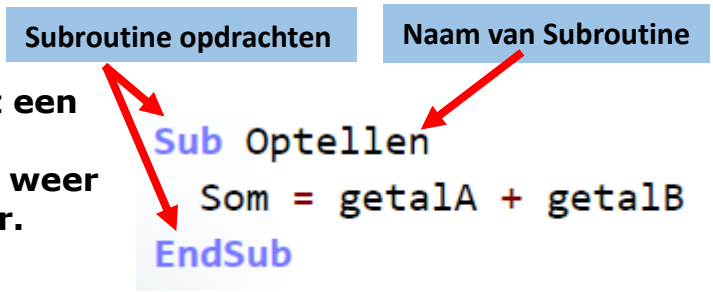
Een functie/Subroutine aanroep in een PSD ziet er als volgt uit:

- In het vak staat de naam van de subroutine die aangeroepen moet worden.



In SB is dit:

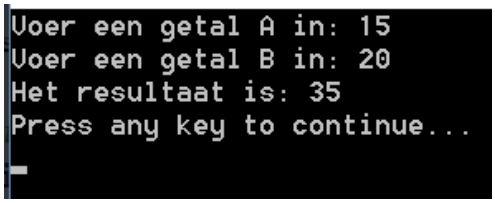
- Opdracht **Sub**
- Gevolgd door een **naam**
  - **Begin de naam met een hoofdletter en elk volgend woord ook weer met een hoofdletter.**
- Bron code
- Opdracht **EndSub**



SB Console Deel 2

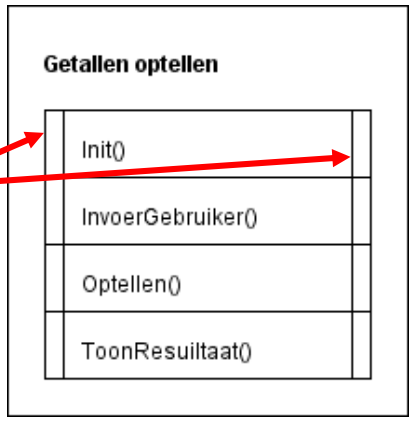
Voorbeeld gebruik subroutines

Het programma Getallen optellen van *Les 1.*



Programma vraagt een gebruiker om twee getallen in te voeren, telt deze op en toont daarna het resultaat op het scherm.

Als we dit programma aanpassen door gebruik te maken van subroutines dan ziet het hoofd programma er nu uit zoals in de PSD hier naast. De Call structuur wordt nu gebruikt, herkenbaar door de *dubbele strepen* links en rechts van een standaard instructie structuur.



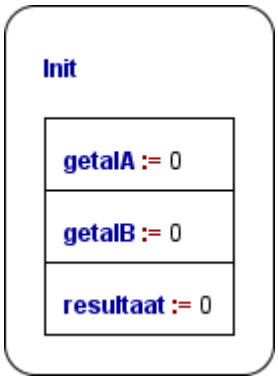
Het hoofdprogramma is te verdelen in 4 subroutines

- Init()
- InvoerGebruiker()
- Optellen()
- ToonResultaat()

In Small Basic wordt de broncode als volgt:

```
1  'Hoofdprogramma
2  Init()
3  InvoerGebruiker()
4  Optellen()
5  ToonResultaat()
```

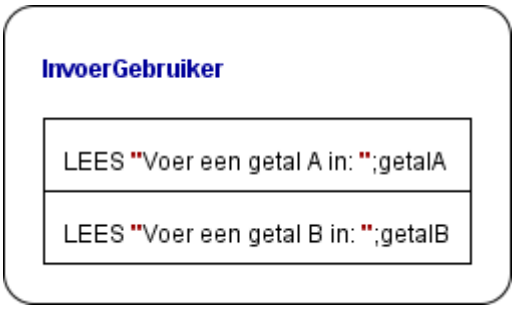
De routine Init() initialiseert de variabelen die in het programma gebruikt worden. De PSD en broncode is als volgt:




```
7  'Subroutines
8  Sub Init
9    getalA = 0
10   getalB = 0
11   resultaat = 0
12 EndSub
```

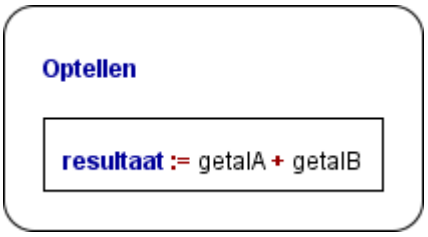
SB Console Deel 2

De functie InvoerGebruiker() vraagt de gebruiker om twee getallen in te voeren en bewaart deze in de twee variabelen getalA en getalB. De PSD en broncode voor deze routine zien er zoals in de afbeeldingen hieronder uit:




```
14 Sub InvoerGebruiker
15   TextWindow.Write("Voer een getal A in: ")
16   getalA = TextWindow.ReadNumber()
17   TextWindow.Write("Voer een getal B in: ")
18   getalB = TextWindow.ReadNumber()
19 EndSub
```

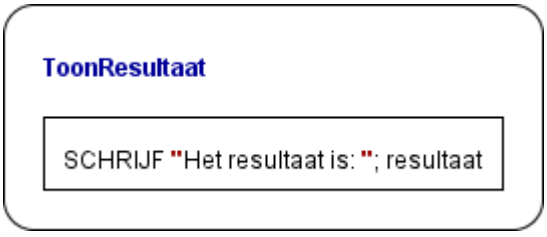
De routine optellen() heeft de volgende PSD en broncode:




```
21 Sub Optellen
22   resultaat = getalA + getalB
23 EndSub
```

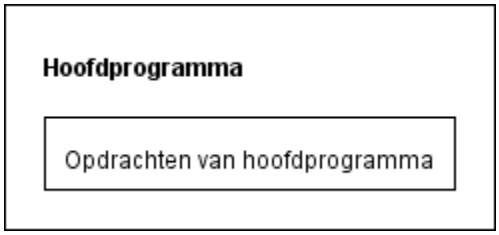
Het resultaat van de optelling wordt bewaard in een variabele resultaat.

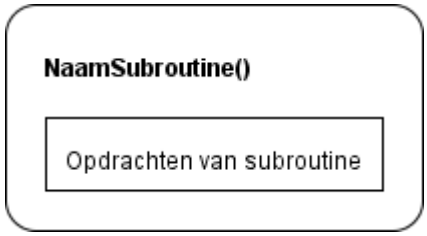
De routine ToonResultaat() drukt het resultaat op het scherm af zodat de gebruiker deze ziet. De PSD en bron code zijn als volgt:




```
25 Sub ToonResultaat
26   TextWindow.Writeline("Het resultaat is: " + resultaat)
27 EndSub
```

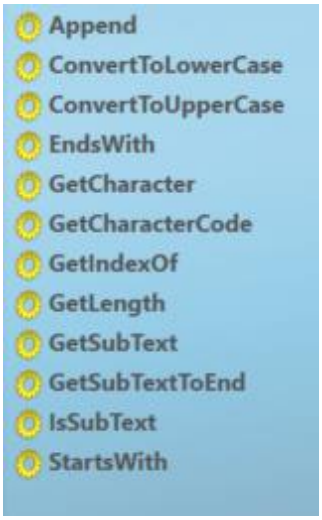
In een PSD wordt het hoofdprogramma in een rechthoek met hoeken van 90^0 graden weergegeven en een subroutine heeft ronde hoeken.





SB Console Deel 2

Het Text object



Small Basic is een object georiënteerde taal en elk object heeft 1 of meerdere eigenschappen en of Methodes.
Enkele veel gebruikte Methoden van het object Text zijn:

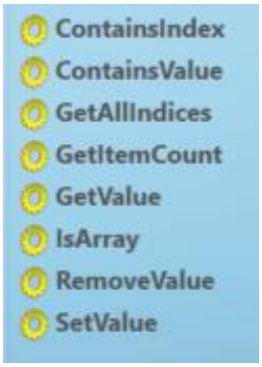
Append:
Voegt invoer van twee teksten samen en geeft het resultaat als een andere tekst. Deze operatie is vooral handig bij het omgaan met onbekende tekst in variabelen die per ongeluk zou kunnen worden behandeld als getallen en worden opgeteld, in plaats van te worden samengevoegd.

ConvertToLowerCase/UpperCase:
Converteert de gegeven tekst naar kleine letters of in het andere geval naar hoofdletters.

GetLength:
Bepaald de lengte van een geven tekst.

GetSubText:
Bepaald een sub-tekst uit een gegeven tekst.

Het Array object



We hebben nu geleerd met array variabelen te werken. Er is ook een Array object dat verschillende methoden kent voor het werken met array variabelen.

ContainsValue:
Bepaald of de array de opgegeven waarde bevat. Dit is erg handig bij de beslissing of de waarde van de array werd opgeslagen in sommige index.

GetItemCount:
Hiermee wordt het aantal elementen die in een array zijn opgeslagen bepaald.

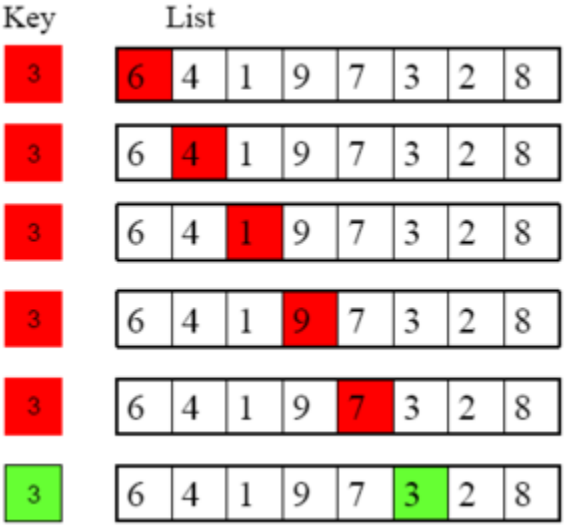
SB Console Deel 2

Zoeken

Computers kunnen snel informatie zoeken in een grote hoeveelheid informatie. Ook kunnen ze snel informatie sorteren van hoog naar laag of omgekeerd.

Wanneer er gezocht moet worden in informatie die niet gesorteerd is dan gebruik je het algoritme dat heet ***lineair zoeken***.

1. Zet K=1 loc=0
2. Herhaal stap 3 en 4 zolang K<=N
3. Als Key=LA[K], dan zet Loc=K
   Schrijf Loc en Stop
4. Zet K=K+1
5. Als Loc=0 dan schrijf element in lijst.
6. Stop



Dat wil zeggen je doorloopt alle informatie tot je gevonden hebt wat gezocht wordt of tot dat je alles hebt bekeken en het dus niet hebt gevonden.

Bijvoorbeeld je hebt een zak met allemaal groene knikkers en één rode knikker. Je moet de rode knikker uit de zak halen zonder dat je in de zak kijkt en je mag iedere keer maar 1 knikker uit de zak pakken.

Dit kan heel snel gaan, als je bij de eerste poging al de groene knikker pakt. Of heel lang duren als je pas bij de laatste knikker de groene uit de zak vist.

Bekijk de onderstaande broncode:

```
knikkers[1] = "ROOD"
```

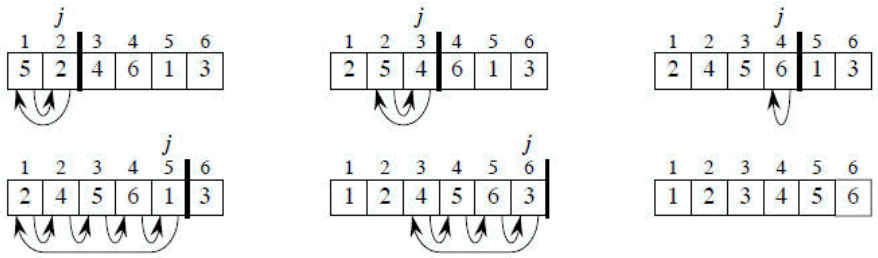
SB Console Deel 2


```
knikkers[2] = "ROOD"
knikkers[3] = "ROOD"
knikkers[4] = "ROOD"
knikkers[5] = "ROOD"
knikkers[6] = "GROEN"
knikkers[7] = "ROOD"
knikkers[8] = "ROOD"
knikkers[9] = "ROOD"
knikkers[10] = "ROOD"
MAX_KNIKKERS = 10
FALSE = 0
TRUE = 1
gevonden = FALSE
index = 0

Zoeken()
If gevonden = TRUE then
  TextWindow.WriteLine("Groene knikker is gevonden")
  TextWindow.WriteLine("De index is " + index)
Else
  TextWindow.WriteLine("Geen groene knikker gevonden!")
EndIf

Sub Zoeken
  For X =  1 To MAX_KNIKKERS
    If knikkers[X] = "GROEN" Then
      gevonden = TRUE
      index = X
      X = MAX_KNIKKERS
    EndIf
  EndFor
EndSub
```

SB Console Deel 2

Sorteren

In de afbeelding hier onder zie je een voorbeeld van lineair sorteren.



In de afbeelding hiernaast staat de uitvoer van een programma dat 10 willekeurige getallen genereerd en in een array plaatst. Daarna sorteert het de lijst en laat na elke stap de nieuwe lijst zien op het scherm.

```
Ongesorteerde lijst is:
stap 0 -> 67 24 27 72 64 68 91 53 53 1

Sorteer stappen:
----------------
stap 1 -> 1 67 27 72 64 68 91 53 53 24
stap 2 -> 1 24 67 72 64 68 91 53 53 27
stap 3 -> 1 24 27 72 67 68 91 64 53 53
stap 4 -> 1 24 27 53 72 68 91 67 64 53
stap 5 -> 1 24 27 53 53 72 91 68 67 64
stap 6 -> 1 24 27 53 53 64 91 72 68 67
stap 7 -> 1 24 27 53 53 64 67 91 72 68
stap 8 -> 1 24 27 53 53 64 67 68 91 72
stap 9 -> 1 24 27 53 53 64 67 68 72 91
Press any key to continue...
```

Bekijk de broncode hieronder:

```
'Initialisatie
AANTAL = 10
teller1 = 0
teller2 = 0
getal[0] = 0
dummy = 0

'Hoofdlus
GenereerOngesorteerdeLijst()
TextWindow.WriteLine("")
TextWindow.WriteLine("Sorteer stappen:")
TextWindow.WriteLine("----------------")
SorteerLijst()

' zet aantal willekeurige getallen in een array
'schrijf het array meteen op het scherm
Sub GenereerOngesorteerdeLijst
  TextWindow.WriteLine("Ongesorteerde lijst is: ")
  TextWindow.Write("stap "+ teller1 + " -> ")
  for teller = 1 To AANTAL
    getal[teller] = math.GetRandomNumber(100)
    TextWindow.Write(getal[teller] + " ")
  EndFor
  TextWindow.WriteLine("")
```

```
EndSub

'sorteer het array
Sub SorteerLijst
  'doorloop de getallen van 1 tot aantal-1
  For teller1 = 1 To AANTAL - 1
    'vergelijk deze met de getallen erna
    For teller2 = teller1 + 1 To AANTAL
      'verwissel ze indien nodig
      If getal[teller1] > getal[teller2] Then
        dummy = getal[teller1]
        getal[teller1] = getal[teller2]
        getal[teller2] = dummy
      Endif
    EndFor
    ToonLijst()
  EndFor
EndSub
```

Ga naar Small Basic en klik op de optie import. Tik daar de volgende code ***SORTVIZ*** in het venster in en klik op ok. Er wordt een programma geïmporteerd van het internet. Voer dit programma uit. Het geeft op een grafische manier sorteren weer op twee manieren.

Naast lineair zoeken en lineair sorteren bestaan andere algoritmes voor zoeken en sorteren die veel sneller zijn. Bijvoorbeeld **Binair zoeken** en **Bubble Sort**.

Zoek op het Internet op hoe deze algoritmen werken, maak een PSD en schrijf de broncode in Small Basic voor de implementatie van deze algoritmen.