Array Basisprincipes

Tot nu toe heb je met variabelen gewerkt om losse stukjes informatie op te slaan. Maar je kunt nog meer geweldige programma's maken door veel informatie in één variabele op te slaan! In Small Basic doe je dat door een ***array*** te gebruiken.

Een ***array*** is een ingebouwd gegevenstype waarmee u met groepen gegevens kunt werken.

Small Basic heeft twee soorten arrays;

1. **indexed arrays**
   Er wordt verwezen naar de stukjes gegevens, *data*, in een *geïndexeerde array* met behulp van een integer-index, zoals score[1], naam[3], enzovoort.
2. **Associative arrays**
   Er wordt naar de elementen van een associatieve array verwezen met behulp van een tekenreeksindex, zoals price["apple"] of address["John"].

Stel dat je een programma wilt schrijven dat vier testscores van een gebruiker neemt en deze vervolgens samen met hun gemiddelde waarde weergeeft. Op basis van wat je tot nu toe hebt geleerd, kun je een programma schrijven zoals hieronder.

```
1  'Average1.sb
2  TextWindow.WriteLine("Voer 4 toetscijfers in op basis van 100 (85, 55, etc.")
3  TextWindow.WriteLine("Druk op <ENTER> na elk cijfer.")
4  s1 = TextWindow.ReadNumber()   'Leest 1ste cijfer
5  s2 = TextWindow.ReadNumber()   'Leest 2de cijfer
6  s3 = TextWindow.ReadNumber()   'Leest 3de cijfer
7  s4 = TextWindow.ReadNumber()   'Leest 4de cijfer
8  avg = (s1 + s2 + s3 + s4) / 4  'Berkend het gemiddelde
9  TextWindow.WriteLine("Cijfers: " + s1 + ", " + s2 + ", " + s3 + ", " + s4)
10 TextWindow.WriteLine("Het gemiddelde is: " + avg)
```

Start Small Basic en voer de code in. Voer het programma uit en bekijk de code.

Stel dat je niet vier cijfers maar 100 zou moeten invoeren. Hoe zou je de code anders kunnen schrijven met behulp van arrays dat het efficiënter wordt. De code hieronder geeft aan hoe het met 10 cijfers zou kunnen werken.

Array's

```
For N = 1 To 10
  TextWindow.Write("Voer cijfer " + N + " in: ")
  score[N] = TextWindow.ReadNumber()
EndFor
TextWindow.WriteLine(score)
```

In plaats van 10 variabelen te maken, zoals `s1`, `s2` enzovoort tot `s10`, maakt je één array variabele met de naam `score`. Om naar elk stuk data in de scorereeks te verwijzen, gebruikt u de syntaxis `score[N]`, waarbij N een variabele is die de waarden 1 tot en met 10 aanneemt. Het schrijven van `score[N]` is als `score[1]`, `score[ 2]`, ... `score[10]`, en de for-lus verhoogt N voor jou.

Start Small Basic en tik de bovenstaande programma code in. Voer het programma uit en bekijk het resultaat.

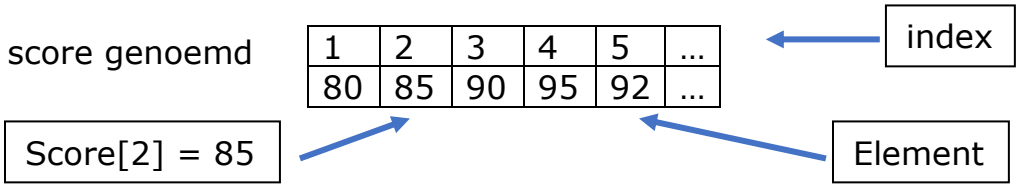
Arrays kunnen je helpen jouw gegevens zo te organiseren dat de gegevens veel gemakkelijker te wijzigen en te gebruiken zijn. De naam van een array volgt dezelfde regels en richtlijnen die je gebruikt voor het benoemen van variabelen.

Array Basics

Elk stukje informatie in een array wordt een element genoemd. Om toegang te krijgen tot een element in een array, gebruik je deze syntaxis:

arrayName[index]

De variabele *arrayName* is de naam van de array en index is een identifier, een getal of een tekenreeks, die een element in de array identificeert (zie de onderstaande afbeelding). Deze syntaxis staat bekend als een *indexed variabele* of een *subscripted variabele*. De index, die tussen vierkante haken wordt geplaatst, identificeert op unieke wijze één element in de array.



Je kunt een geïndexeerde variabele net als een gewone variabele behandelen door de juiste syntaxis te gebruiken. De volgende instructies

initialiseren en tonen bijvoorbeeld de eerste drie elementen van de scorereeks in de onderstaande afbeelding.

```
score[1] = 80
score[2] = 85
score[3] = 90
TextWindow.WriteLine(score[1] + ", " + score[2] + ", " + score[3])
```

De uitvoer is als volgt:

```
80, 85, 90
Press any key to continue..
```

Als je de eerste score wilt wijzigen, kun je deze code regel schrijven:

```
score[1] = score[1] + 5
```

Deze regel code voegt vijf toe aan de eerste score bij index 1. als je de waarde van `score` nu zou weergeven, zou je zien dat `score [1] = 85` is. Je zou de volgende regel code kunnen gebruiken om de twee elementen bij indices te vermenigvuldigen 1 en 2.

```
score[1] = score[1] + score[2]
```

Als `score [1]` 80 is en `score [2]` 85, worden ze vermenigvuldigd om 6800 te krijgen, die weer wordt opgeslagen in `score [1]`. Hoogste score!

Arrays initialiseren

Voordat je een array in jouw programma gebruikt, moet je deze vullen, *initialiseren,* met wat gegevens. In Small Basic kun je dit op twee manieren doen:

Stel dat je een array wilt maken met vier scores. Dit is de directe manier om dat te doen:

```
score[1] = 80
score[2] = 85
score[3] = 90
score[4] = 95
```

Je kunt ook een ***string initialzer*** gebruiken, waarmee je de vier waarden in slechts één regel als volgt kunt instellen: `score = "1=80;2=85;3=90;4=95;'`

Deze string-initialisator heeft vier tokens (of velden), die worden afgesloten met puntkomma's.
Merk op dat er geen spaties voor of na het = teken staan.

index=value;

Opmerking:

Met Small Basic kunt je alle gewenste getallen voor indices kiezen. Je kunt zelfs negatieve en decimale getallen gebruiken, en de indices hoeven niet opeenvolgend te zijn.

Arrays vullen met een For-lus

Vaak moet u de elementen van een array vullen met een constante waarde, een willekeurige (*random*) waarde, een waarde berekend op basis van een formule of een waarde die door een gebruiker is ingevoerd. Laten we elk scenario bekijken.

- **Constante initialisatie**

  Het volgende codefragment laat zien hoe u de eerste 10 elementen van een array initialiseert met een constante waarde van nul.

```
For N = 1 To 10
  score[N] = 0
EndFor
```

- **Willekeurige (*random*) initialisatie**

  Je kunt de elementen van een array ook vullen met willekeurige getallen, zoals deze:

```
For N = 1 To 10
  score[N] = Math.GetRandomNumber(5)
EndFor
```

- **Formule-initialisatie**

  Je kunt de elementen van een array ook initialiseren met behulp van een formule. In dit voorbeeld stel je het N-de element van de score array in op N * 8; deze code slaat de vermenigvuldigingstabel van acht op in de array:

```
For N = 1 To 10
  score[N] = N * 8
EndFor
```

- **Gebruikersinitialisatie**

  Het volgende programma vraagt de gebruiker om vijf cijfers in te voeren en na elk cijfer op <ENTER> te drukken.

```
TextWindow.WriteLine("Voer 5 cijfers in. Druk op <ENTER> na elk cijfer.")
For N = 1 To 10
  score[N] = TextWindow.ReadNumber()
EndFor
```

Deze techniek is erg handig voor het opslaan van veel gegevens van een gebruiker.

Arrays weergeven

Laten we zeggen dat we een array hebben met de naam age die de leeftijden van drie broers als volgt bevat:

```
age[1] = 14
age[2] = 15
age[3] = 16
```

Je kunt de inhoud van deze array op twee manieren weergeven. De eerste en makkelijkste manier is om de naam van de array door te geven aan de `WriteLine()` methode, als volgt:

```
TextWindow.WriteLine(age)
```

De uitvoer is dan:

```
1=14;2=15;3=16;
Press any key to continue...
```

Deze instructie geeft de elementen van de array op een enkele regel weer, gescheiden door puntkomma's. Elk token in de tekenreeks toont de index en de waarde van het element van de array bij de index.

Als je de array in een gemakkelijker leesbare indeling wilt weergeven, kun je een `For`-lus gebruiken om elk element van de array in zijn eigen rij weer te geven:

```
1 age[1] = 14
2 age[2] = 15
3 age[3] = 16
4 For N = 1 To 3
5   TextWindow.WriteLine("age[" + N + "] = " + age[N])
6 EndFor
```

Hier is de uitvoer voor de lus:

```
age[1] = 14
age[2] = 15
age[3] = 16
Press any key to continue...
```

Arrays verwerken

Veel programma's omvatten het verwerken van de elementen van een array, zoals het toevoegen ervan en het vinden van hun gemiddelde, minimum, maximum, enzovoort.

Een politieagent genaamd Buchi wil weten hoeveel geld hij heeft gered van 10 overvallers in zijn bario. Met het volgende programma kan Buchi de bedragen die hij heeft gered invoeren in een array met de naam `placaDebolbi`. Het programma vindt de som van alle elementen van deze array:

```
totaal = 0
TextWindow.WriteLine("Vul 10 bedragen in die terug gegeven zijn:")
For N = 1 To 10
  placaDebolbi[N] = TextWindow.ReadNumber()
  totaal = totaal + placaDebolbi[N]
EndFor
For N = 1 To 10
  TextWindow.Write("$"+ placaDebolbi[n])
  TextWindow.WriteLine(" gered van overvaller " + N)
EndFor
TextWindow.WriteLine("$" + totaal + " was gered door Butchi!")
```

Om het totaal te vinden, begin je met het initialiseren van de variabele `totaal` op 0. Vervolgens voer je een `For`-lus uit om elk element van de `placaDebolbi`-array te lezen en toe te voegen aan de variabele `totaal`. Wanneer de lus eindigt, start je een andere lus om te laten zien hoeveel er van elke overvaller is gered, en vervolgens geef je het totale geretourneerde bedrag weer.

De uitvoer ziet er als volgt uit:

```
Vul 10 bedragen in die terug gegeven zijn:
10
76
3
2
38
48
90
26
36
8
$10 gered van overvaller 1
$76 gered van overvaller 2
$3 gered van overvaller 3
$2 gered van overvaller 4
$38 gered van overvaller 5
$48 gered van overvaller 6
$90 gered van overvaller 7
$26 gered van overvaller 8
$36 gered van overvaller 9
$8 gered van overvaller 10
$337 was gered door Butchi!
Press any key to continue...
```

Opdracht 1:
Stel dat je met negen van je goede vrienden concurreert om te zien wie de meeste vrienden heeft op Instagram. Gebruik het volgende codefragment om de grootste waarde te vinden in een array met de naam vrienden.

Gebruik string waarden om de array `friends` te initialiseren me de volgende waarden: `1 = 10, 2 = 30, 3 = 5, 4 = 10, 5 = 15, 6 = 45, 7 = 25, 8 = 13, 9 = 6, 10 = 11`

De uitvoer moet er zo uitzien:

```
De meeste vrienden aantal is 45.
Press any key to continue...
```

String waarden gebruiken in arrays

Array's zijn niet beperkt tot getallen. Je kunt ook arrays gebruiken om strings op te slaan. Stel dat je bijvoorbeeld een array wilt maken om de namen van de boeken in jouw verzameling op te slaan. Je zou deze array kunnen initialiseren:

```
book[1] = "Autonomous"
book[2] = "Ready Player One"
book[3] = "The Hobbit"
```

Opmerking:
Je zou ook een string-initialisatieprogramma kunnen gebruiken om de boekarray als volgt te initialiseren (zorg ervoor dat de hele instructie op één regel staat!):

```
book = "1=Autonomous;2=Ready Player One;3=The Hobbit"
```

Records opslaan
Je kunt verschillende gegevenstypen binnen één array combineren. Je kunt getallen, zowel hele als decimaal, en tekenreeksen opslaan als verschillende elementen in dezelfde array. De volgende array is bijvoorbeeld geldig:

```
arr[1] = "Vondellaan"
arr[2] = "#14"
arr[3] = 5822005
```

Geïndexeerde arrays gebruiken
Het voorbeeld in deze sectie laat zien hoe u willekeurige elementen uit een array kunt selecteren.

Laten we zeggen dat we een zak hebben met 10 ballen genummerd van 1 tot 10, en we willen er vijf willekeurige ballen uit halen.

We zullen een programma schrijven dat willekeurig vijf ballen selecteert en vervolgens hun nummers toont.

Tik de code in Small Basic in en voer het programma uit.

```
'RandomSelect.sb
For N = 1 To 10 'Plaatst de 10 ballen in een array
  ball[N] = N
EndFor

For N = 1 To 5                          'Lus om 5 ballen te selecteren
  idx = Math.GetRandomNumber(10)        ' Kies een willekeurg nummer voor een ball
  While (ball[idx] = 0)                 ' Bal is al geselecteerd
    idx = Math.GetRandomNumber(10)      'Neem een ander nummer
  EndWhile

  TextWindow.WriteLine(ball[idx] + ", ")  'Toond de geseleteerde ball
  ball[idx] = 0                           'Markeerd de bal als uit de zek genomen
EndFor
TextWindow.Write("")
```

Opdracht 2: Magic 8 ball

In deze opdracht moet je een programma schrijven dat een Magic 8 Ball-spel simuleert. Een gebruiker stelt een ja of nee vraag, en de computer antwoordt. De uitvoer van het programma ziet er zo uit:

```
Ask me a yes-no question. Do it!
Am I smart?
Let me consult my Magic 8 Ball... It says yes.

Ask me a yes-no question. Do it!
```

Gebruik de volgende tekstregels voor de antwoorden van het programma:

1. "It is certain. Like really, really certain."
2. "It is decidedly so. By me. I decided."
3. "Without a doubt. Maybe one doubt."
4. "Yes, definitly. Isn't it obcious?"
5. "Very doubtful. The doubt is very full."
6. "Maybe. Depends on the horse race."
7. "No. Wait, yes. Wait, no. Yes, it's no."
8. "Let me consult my Magic 8 Ball... It says yes."
9. "Outlook not so good. Restart Outlook."
10. "Try again. It's funny when you shake things."

Opdracht 3: Dobbelsteen
Schrijf een programma dat het rollen van een dobbelsteen simuleert. Laat het programma 10.000 keer een dobbelsteen gooien en houd bij hoe vaak elk nummer voorkomt. Een voorbeeld van de uitvoer van het programma is als volgt:

```
Num  Count  Probability
1    1669   0.1669
2    1633   0.1633
3    1652   0.1652
4    1626   0.1626
5    1706   0.1706
6    1714   0.1714
Press any key to continue...
```

Hint: gebruik een array met de naam dobbelstenen die zes elementen heeft. Als je 1 in een worp krijgt, verhoog je dobbelstenen [1]. Als je er 2 krijgt, verhoog je dobbelstenen [2], enzovoort.)

Gegevens opslaan met associatieve arrays

Naast geïndexeerde arrays ondersteunt Small Basic andere typen arrays die veel programmeertaken kunnen vereenvoudigen. In dit volgende deel leer je over ***associatieve arrays***. Dan leer je over het ***Array***-object.

Associatieve arrays gebruiken

In de vorige lessen heb je geleerd hoe je een integer-index gebruikt om toegang te krijgen tot de elementen van een array. Maar in Small Basic kan de index van een array ook een string zijn. Door strings geïndexeerde arrays worden ***associative arrays***, ***maps*** of ***dictionaries*** genoemd.

Net als een geïndexeerde array, kan een associatieve array waarden van elk type opslaan. Je kunt een associatieve array gebruiken om een koppeling te maken tussen een reeks sleutels (*string indexen*) en een reeks waarden, wat een *map* of ***key-value pairs*** (sleutel-waardeparen) wordt genoemd.

De volgende code toont een eenvoudig voorbeeld van een associatieve array in actie. Het is een lijst met docentencodes op basis van hun afkortingen van twee letters:

```
docentcode["ES"] = "Ellis"
docentcode["FP"] = "Fernandez-Perna"
docentcode["VS"] = "van Schuilenburg"
```

Om de naam van de leraar weer te geven, gebruikt je gewoon de bijbehorende sleutel en de juiste syntaxis. Om bijvoorbeeld `Fernandez-Perna` weer te geven, kunt je deze instructie schrijven:

```
TextWindow.WriteLine(doventcode["FP"])
```

Een associatieve array werkt als een opzoektabel die sleutels toewijst aan waarden (Excel lookup functies); als je de sleutel kent, kun je de waarde ervan heel snel vinden.

Laten we, om te leren hoe je associatieve arrays gebruikt, een programma schrijven dat de leeftijden van je vrienden bij naam bijhoudt. Tik de programma code hieronder in Small Basic in. Sla het op met de naam AssociativeArray.sb en voer het programma uit.

```
1 'AssociativeArray.sb
2 leeftijd["Sil"] = 16
3 leeftijd["Jayke"] = 14
4 leeftijd["Aeon"] = 17
5 leeftijd["Gideon"] = 15
6 TextWindow.Write("Tik de naam van een vriend in: ")
7 naam = TextWindow.Read()
8 TextWindow.Write(naam + " is [")
9 TextWindow.WriteLine(leeftijd[naam] + "] jaren oud.")
```

De uitvoer ziet er dan als volgt uit:

```
Tik de naam van een vriend in: Aeon
Aeon is [17] jaren oud.
Press any key to continue...
```

```
Tik de naam van een vriend in: aeon
aeon is [17] jaren oud.
Press any key to continue...
```

Merk op dat de sleutel niet hoofdlettergevoelig is: het maakt niet uit of je `leeftijd["Aeon"]`, `leeftijd["aeon"]` of zelfs `leeftijd["AEON"]` invoert. Als de array een sleutel met de naam `Aeon` bevat, ongeacht hoofdletter gebruik, Small Basic retourneert de waarde voor die sleutel.

Stel dat je bent vergeten welke namen van vrienden je in de array hebt opgeslagen en je probeert toegang te krijgen tot de leeftijden van iemand die je bent vergeten op te nemen:

```
Tik de naam van een vriend in: Jeanrick
Jeanrick is [] jaren oud.
Press any key to continue...
```

Als de array een bepaalde sleutel niet bevat, retourneert Small Basic een lege string, daarom is leeftijd["Jeanrick"] leeg.

```
'AssociativeArray2.sb
TextWindow.Write("Tik de naam van een vriend in: ")
naam = TextWindow.Read()
If (naam = "Sil") Then
  leeftijd = 16
ElseIf (naam = "Jayke") then
  leeftijd = 14
ElseIf (naam = "Aeon") Then
  leeftijd = 17
ElseIf (naam = "Gideon") then
  leeftijd = 15
Else
  leeftijd = ""
EndIf
TextWindow.WriteLine(naam + " is [" + leeftijd[naam] + "] jaren oud.")
```

Hoewel dit programma lijkt op het vorige. De twee hebben één belangrijk verschil:

> Hier is stringvergelijking hoofdlettergevoelig. Als je `gideon` [*met kleine letter g*] invoert, geeft het programma de volgende uitvoer weer:

```
Tik de naam van een vriend in: gideon
gideon is [] jaren oud.
Press any key to continue...
```

Het array-object

Het array-object in de Small Basic-bibliotheek kan je helpen belangrijke informatie over arrays in jouw programma's te vinden. We zullen dit object in detail onderzoeken en enkele voorbeelden bekijken over hoe het te gebruiken. Laten we beginnen met het invoeren van de volgende code om het `Array`-object te verkennen:

```
naam = "Bart"              'Een gewone variable
leeftijd["Homer"] = 18     'Een associatieve array met twee elementen
leeftijd["Marge"] = 17
score[1] = 90              'Een geïndexeerde array met één element
```

Deze code definieert een gewone variabele genaamd naam, een associatieve array genaamd leeftijd die twee elementen heeft, en een geïndexeerde array

genaamd score die één element heeft. Je gebruikt deze arrays in de volgende voorbeelden. Wat kan het `Array`-object je vertellen? Laten we het uitzoeken!

Is het een Array?
Denk je dat Small Basic weet dat naam een gewone variabele is en dat leeftijd en score arrays zijn? Voer het onderstaande programma uit om erachter te komen.

```
1 'IsArray.sb
2 naam = "Bart"                'Een gewone variable
3 leeftijd["Homer"] = 18       'Een associatieve array met twee elementen
4 leeftijd["Marge"] = 17
5 score[1] = 90                'Een geïndexeerde array met één element
6 ans1 = Array.IsArray(naam)                    'Retourneert "False"
7 ans2 = Array.IsArray(leeftijd)                'Retourneert "True"
8 ans3 = Array.IsArray(score)                   'Retourneert "True"
9 Textwindow,WriteLine(ans1 + ", " + ans2 + ", " + ans3)
```

Hoe groot is een Array?
Het Array-object kan je ook vertellen hoeveel elementen er in jouw arrays zijn opgeslagen. Voer het onderstaande programma uit.

```
1 'GetItemCount.sb
2 naam = "Bart"
3 leeftijd["Homer"] = 18
4 leeftijd["Marge"] = 17
5 score[1] = 90
6 ans1 = Array.GetItemCount(naam)          'Retourneert: 0
7 ans2 = Array.GetItemCount(leeftijd)      'Retourneert: 2
8 ans3 = Array.GetItemCount(score)         'Retourneert: 1
9 Textwindow,WriteLine(ans1 + ", " + ans2 + ", " + ans3)
```

De methode `GetItemCount()` retourneert het aantal items in de opgegeven array. Merk op hoe `GetItemCount(naam)` 0 retourneert, omdat naam geen array is.

Heeft het een bepaalde index?
Je kunt het Array-object ook gebruiken om erachter te komen of een van jouw arrays een bepaalde index bevat. Om te zien hoe, voer het programma hieronder uit.

```
1  'ContainsIndex.sb
2  leeftijd["Homer"] = 18
3  leeftijd["Marge"] = 17
4  score[1] = 90
5  ans1 = Array.ContainsIndex(leeftijd, 1)          'Retourneert: "False"
6  ans2 = Array.ContainsIndex(leeftijd, "homer")    'Retourneert: "True"
7  ans3 = Array.ContainsIndex(leeftijd,"LISA")      'Retourneert: "False"
8  Textwindow,WriteLine(ans1 + ", " + ans2 + ", " + ans3)
9
10 ans1 = Array.ContainsIndex(score, "1")           'Retourneert: "False"
11 ans2 = Array.ContainsIndex(score, 1)             'Retourneert: "True"
12 ans3 = Array.ContainsIndex(score, 2)             'Retourneert: "False"
13 Textwindow,WriteLine(ans1 + ", " + ans2 + ", " + ans3)
```

Regel 6 laat zien dat het zoeken naar index hoofdletter-*ongevoelig* is, wat de reden is waarom het zoeken naar index `homer` "True" retourneert. Ook het zoeken in de `score`-array naar index "1" (*als een string*) of index 1 (als een getal) levert beide "True" op.

Heeft het een bepaalde waarde?
Het object Array biedt ook een methode die controleert of een array een bepaalde waarde bevat.

```
1  'ContainsValue.sb
2  leeftijd["Homer"] = 18
3  leeftijd["Marge"] = 17
4  score[1] = 90
5  ans1 = Array.ContainsValue(leeftijd, 18)         'Retourneert: "Truee"
6  ans2 = Array.ContainsValue(leeftijd, 20)         'Retourneert: "False"
7  ans3 = Array.ContainsIndex(score, 90)            'Retourneert: "True"
8  Textwindow,WriteLine(ans1 + ", " + ans2 + ", " + ans3)
```

Opmerking: In tegenstelling tot de methode `ContainsIndex()` is de methode `ContainsValue()` hoofdlettergevoelig. Dus het is het beste om consistent te zijn met je teksten!

Geef me alle indexen
Een andere handige methode van het `Array`-object is `GetAllIndices()`. Deze methode retourneert een array die alle indices van een bepaalde array heeft. Het eerste element van de geretourneerde array heeft een index van 1. Om

te begrijpen hoe deze methode werkt, voer je de onderstaande code uit en bestudeer je deze.

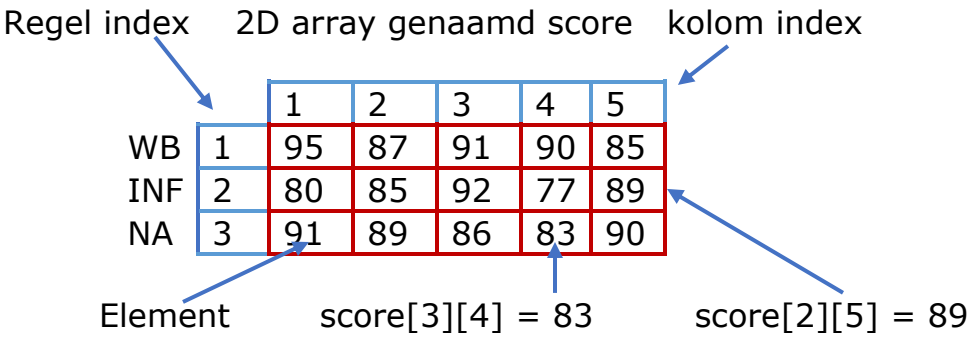
```
1 'GetAllIndeices.sb
2 leeftijd["Homer"] = 18
3 leeftijd["Marge"] = 17
4 namen = Array.GetAllIndices(leeftijd)
5 Textwindow.WriteLine("De indexen av de leeftijd array zijn:")
6 For N = 1 To Array.GetItemCount(namen)
7   Textwindow.WriteLine("Index " + N + " = " + namen[N])
8 EndFor
```

Regel 4 roept `GetAllIndices()` aan om alle indices van de `namen`-array te vinden. Deze methode retourneert een array, die wordt opgeslagen in de `namen`-ID. De code start dan een lus die loopt van het eerste naar het laatste element in `namen`. Merk op hoe de code de methode `GetItemCount()` gebruikt om deze waarde te achterhalen. Hier is de uitvoer van deze code:

```
De indexen av de leeftijd array zijn:
Index 1 = Homer
Index 2 = Marge
Press any key to continue...
```

Tweedimensionale arrays

Een 2D-array heeft twee dimensies: rijen en kolommen. Je kunt een 2D-array zien als een tabel. Bijvoorbeeld in de onderstaande afbeelding een 2D-array genaamd `score` die de testscores van een student opslaat van drie vakken.



De eerste rij bevat scores voor wiskunde B toetsen, de tweede rij bevat de scores voor Informatica toetsen en de volgende rij slaat Natuurkunde scores op. Deze 2D-rangschikking van elementen wordt ook wel een ***matrix*** genoemd (het meervoud is ***matrices***).

Om toegang te krijgen tot de afzonderlijke elementen van een matrix, heb je twee indices nodig: één voor de rijen en de andere voor de kolommen. Hier zijn enkele voorbeelden:

```
score[1][1] = 95   'Regel 1, kolom 1
score[1][2] = 87   'Regel 1, kolom 2
score[2][1] = 80   'Regel 2, kolom 1
score[2][3] = 92   'Regel 2 kolom 3
```

Een klant genaamd MI6 wilt je hulp om wachtwoorden voor veiligheidssloten te genereren. Het onderstaande programma maakt een matrix met de naam `mat` die bestaat uit willekeurige getallen. De matrix bevat drie rijen en vier kolommen, wat een 3x4 (lees als: 3 bij 4) matrix is, of een 3x4 array.

```
'Random2Darray.sb
For r = 1 To 3        '3 regels
  For c = 1 To 4      '4 kolommen
    Math[r][c] = Math.GetRandomNumber(9)
  EndFor
EndFor

'Toon de matrix om de inhoudt te zien
For r = 1 To 3        '3 regels
  For c = 1 To 4      '4 kolommen
    TextWindow.Write(Math[r][c] + " ")
  EndFor
  TextWindow.WriteLine("")
EndFor
```

Het is tijd om je programma over te dragen aan jouw MI6-client. Hier is een voorbeelduitvoer van dit programma.

```
7 3 7 7
7 5 2 4
4 1 5 1
Press any key to continue...
```

Gebruik van string indexen

De vorige voorbeelden gebruikten integer-indexen om toegang te krijgen tot de elementen van een matrix. Dit volgende voorbeeld leert je hoe je strings voor indexen kunt gebruiken. Je onderzoekt een applicatie die de scores van studenten in verschillende vakken bijhoudt.

Welkom in de klas van juf Tromp. De klas heeft momenteel maar drie leerlingen: Maria, Juan en Pedro. De school leert slechts drie vakken: wiskunde, informatica en natuurkunde. Schrijf een programma dat de gebruiker vraagt om de naam van een student in te voeren en vervolgens de gemiddelde score van de student weergeeft.

Tik de code in Small Basic in en voer het programma uit.

```
'StudentAvg.sb
score["Maria"]["WI"] = 92
score["Maria"]["INF"] = 90
score["Maria"]["NA"] = 87
score["Juan"]["WI"] = 85
score["Juan"]["INF"] = 82
score["Juan"]["NA"] = 92
score["Pedro"]["WI"] = 85
score["Pedro"]["INF"] = 95
score["Pedro"]["NA"] = 99

TextWindow.Write("Voer een student naam in: ")
naam = TextWindow.Read()
totaal = score[naam]["WI"]
totaal = totaal + score[naam]["INF"]
totaal = totaal + score[naam]["NA"]
avg = Math.Round(totaal / 3)
TextWindow.WriteLine(naam + " gemiddelde score = " + avg)
```

De uitvoer ziet er zo uit:

```
Voer een student naam in: Maria
Maria gemiddelde score = 90
Press any key to continue...
```

Opdracht 4:

Laten we eens kijken hoe we de scores van de leerlingen van de gebruiker kunnen krijgen in plaats van ze hard te coderen in het programma zoals in de vorige lijst. Wijzig het eerdere programma om het gewenste resultaat te bereiken. Gebruik daar bij de zelfde studenten namen en vakken. Hint, je hebt twee lussen nodig om de namen en onderwerpen van de studenten te herhalen.

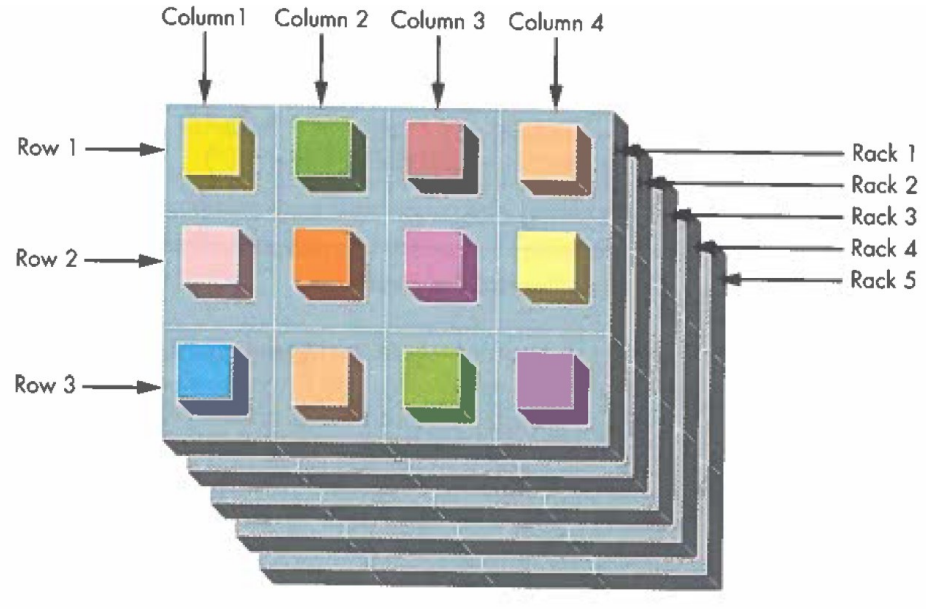
```
Maria's WI score: 55
Maria's INF score: 76
Maria's NA score: 80
Juan's WI score: 91
Juan's INF score: 54
Juan's NA score: 67
Pedro's WI score: 87
Pedro's INF score: 97
Pedro's NA score: 34
Voer een student naam in: Juan
Juan gemiddelde score = 71
Press any key to continue...
```

De uitvoer van het programma moet er als volgt uit zien:

Arrays van Drie of Meer dimensies

Je hebt geleerd dat het gebruik van 2D-arrays een handige manier is om een tabel of matrix weer te geven. Small Basic ondersteunt ook arrays met meer dan twee dimensies. Je kunt de syntaxis voor het maken van 2D-arrays uitbreiden om arrays met nog hogere afmetingen te maken.

Laten we werken met een open kast met vijf rekken. Elk rek heeft drie rijen en vier kolommen en elke positie op de plank heeft een doos met schroeven van een bepaalde grootte. Kijk naar de afbeeldingen en stel je voor in elke kolom en rij dozen met verschillende schroefgroottes (dat zijn 12 dozen). Stel je dan voor dat hetzelfde aantal dozen op alle kantoorrekken. Dat zijn in totaal 60 dozen!



Bestudeer het onderstaande programma dat elke doos vult met een willekeurig getal dat de grootte van de schroeven in die doos aangeeft.

```
'3DArrayDemo.sb
For rek = 1 To 5            'Voor elk rek
  For rij = 1 To 3          'Voor elke rij
    For kol = 1 To 4        'Voor elke kolom
      doos[rek][rij][kol] = Math.GetRandomNumber(9)
    EndFor
  EndFor
EndFor
```

Opdracht 5:
Schrijf een programma dat de uitvoer van de box-array weergeeft. Je uitvoer moet het volgende formaat hebben:

```
Rek 1
4 4 1 7
6 2 9 6
9 1 8 8

Rek 2
6 4 7 1
7 1 5 7
8 2 3 4
```