Programmatielogica

Arrays en Strings

Webleren

School je gratis bij via het internet. Waar en wanneer je wilt.

www.vdab.be/webleren

7/04/2015

Inhoud

Hoofdstuk 1. Inleiding

1.1. Algemene informatie

1.1.1. Waarom?

Waarom leren programmeren? De belangrijkste redenen op een rijtje:

- Programmeren leert je een probleem-oplossende manier van denken aan. Je leert **analytisch denken**.
- Als je kan programmeren kun je een hoop dingen **automatiseren** en ben je waarschijnlijk ook handig met andere ict-gerelateerde zaken.
- Je leert werken met **gegevens**. Informatica is een studie over gegevens en informatie. Bij programmeren leer je hoe je met gegevens om moet gaan en wat je ermee kunt doen.
- Websites maken is waardevol tegenwoordig. Bijna elk bedrijf of project heeft een **bijhorende site** die moet worden onderhouden (Javascript, PHP,…).
- …

In deze cursus **programmatielogica** leer je gestructureerd programmeren, dwz dat je leert zelfstandig problemen op te lossen met de computer. Je gebruikt Nassi-Shneiderman diagrammen. Deze leggen de basis voor het echte programmeerwerk in één of andere taal.

De cursus programmaticalogica bestaat uit verschillende delen.

Het eerste deel, **Basisstructuren**, heb je nu achter de rug. Blijf wat je daar geleerd hebt ook in dit deel toepassen. We beschouwen dit als reeds verworven kennis.
In dit deel, **Arrays en Strings**, leer je werken met tabellen en tekst.

Als je het onderdeel **Procedures en functies** nog niet hebt doorgenomen, kan je dat na dit deel nog doen.
Heb je dan nog steeds zin in meer, kan je nog verder verdiepen in de delen **Sorteren** en/of **Bestanden**.

De nadruk in alle delen ligt op het probleemoplossend denken. Na het tekenen van de diagrammen gaan we deze schema's ook omzetten naar echte programmacode om ze uit te testen.

1.1.2. Werkwijze

In deze cursus gebruiken we een tekentool om Nassi-Shneiderman diagrammen, PSD's of Programma Structuur Diagrammen te tekenen (**Structorizer**). Je kan alle diagrammen ook maken met pen en papier. We gebruiken ook het programma **Lazarus** om onze code uit te testen. Later meer over beide programma's.

In deze cursus zijn heel wat oefeningen voorzien. Je kan ze onderverdelen in twee categorieën:

- *Gewone oefeningen:*
  Dit is het elementaire oefenmateriaal dat noodzakelijk is om de leerstof onder de knie te krijgen.
  Bij deze oefeningen kan je online ook altijd een modeloplossing (van het PSD) terug vinden.

- *Opdrachten voor de coach:*
  Per hoofdstuk is er een opdracht voor de coach voorzien. Deze kan als 'test' voor dat hoofdstuk dienen. Dit is een samenvattende oefening van de voorgaande leerstof. Voor deze opdrachten zijn er geen modeloplossingen voorzien.

Als je deze cursus volgt binnen een **competentiecentrum**, spreek je met je **instructeur** af hoe de opdrachten geëvalueerd worden.

Anders stuur je je oplossingen van de opdrachten voor de **coach** door aan je coach ter verbetering. Als je een probleem of vraag hebt over één van de gewone oefeningen, kan je ook bij je coach terecht. Stuur steeds zowel het .nsd-bestand als het .pas-bestand dat je gemaakt hebt door. Dit maakt het voor je coach makkelijker om je vraag snel te beantwoorden.

Let op!
Het is niet de bedoeling om met Structorizer te leren werken, wel om een probleem te leren analyseren en in een gestructureerd diagram weer te geven. Structorizer en Lazarus zijn enkel hulpmiddelen om het tekenen en uittesten van die diagrammen te vereenvoudigen.

Hoofdstuk 2. Arrays

2.1. Statische arrays

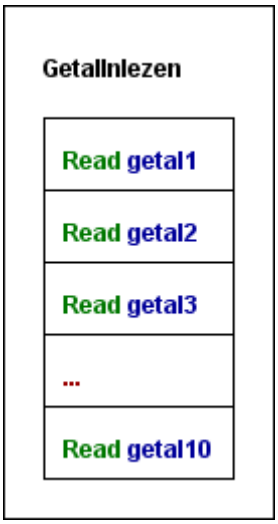
In bepaalde gevallen zal het gebruik van de vier elementaire gegevenstypen (logisch, geheel, reëel en karakter) **ontoereikend** zijn.

Je wil bijvoorbeeld een programma maken dat een tiental getallen inleest en die afdrukt van klein naar groot. Om te kunnen bepalen welk getal het kleinste is, welk het tweede kleinste enz., moet je eerst alle getallen kennen. Dit programma bestaat uit drie fasen:

- Lees 10 getallen in.
- Sorteer de 10 getallen.
- Druk de 10 getallen af.

Op dit moment kunnen we de tweede fase – het effectieve sorteren – even negeren. In het deel **Sorteren** zullen we nog uitgebreid ingaan op verschillende methodes om gegevens snel te sorteren en op alle bijhorende voor- en nadelen.

Het inlezen van de 10 getallen zouden we kunnen beschouwen als een sequentie van 10 opdrachten:



Dit betekent dat we tien verschillende, losstaande variabelen aanmaken, die elk verwijzen naar een eigen geheugenplaats, en die vooral ook elk een eigen naam hebben. Dit kan als volgt voorgesteld worden :

2.1.1. Definitie

Deze manier van werken levert belangrijke nadelen op:

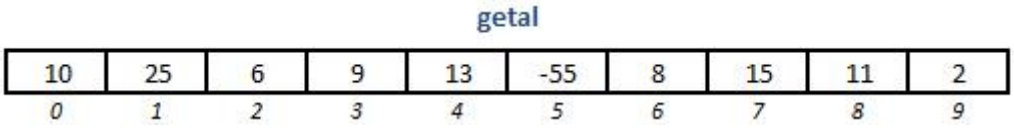
- het inlezen van de gegevens vergt veel dezelfde programmacode (een aparte leesinstructie per variabele);
- er zijn veel verschillende variabelen, en dus ook variabele-namen nodig.

Stel je eens voor wat dat betekent als je niet 10 maar 1000 getallen wil inlezen. Je programma wordt dan een enorme sequentie van lees- en schrijfinstructies. Terwijl toch eigenlijk elk in te lezen getal op dezelfde manier behandeld wordt, behalve dan dat het in een andere variabele wordt gestopt.

Precies daarvoor worden arrays gebruikt: om **gelijkaardige** gegevens (gegevens die eenzelfde soort functie vervullen) te gaan groeperen in één variabele. Zo'n variabele verwijst dan (met één naam) gewoon naar een reeks van geheugenplaatsen. Wil je één van de afzonderlijke geheugenplaatsen gebruiken, dan geef je met een **index** aan welk element je bedoelt.

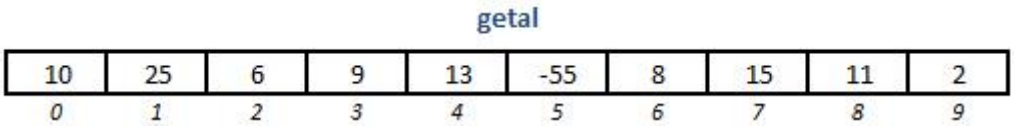
DEFINITIE
Een **array** is een variabele die verwijst naar een reeks van geheugenplaatsen van hetzelfde type, die elk een eigen inhoud kunnen krijgen.

Belangrijk in deze definitie is **van hetzelfde type**. Dat betekent dat er geen numerieke en alfanumerieke waarden samen in dezelfde array kunnen bewaard worden. Het is dus niet toegelaten om bijv. in het eerste array-element een naam te bewaren en in het tweede array-element een loon.



2.1.2. Gebruik en declaratie

Gebruik



In de meeste programmeertalen is de eerste index standaard 0.

Om bijvoorbeeld het 3de element uit de array getal aan te spreken, gebruiken we de notatie getal[2]; voor het 7e getal, geldt de notatie getal[6].
(let op de vierkante haakjes)

- getal[2] bevat de waarde 6,
- getal[6] bevat de waarde 8.

Declaratie

Ook een array moet gedeclareerd worden.

```
var getal:array[0..9] of integer;
```

Dit is een ééndimensionale array met 10 rijen (of plaatsen). Elk item in de array moet van het type integer zijn.

```
var letters:array[5..11] of char;
```

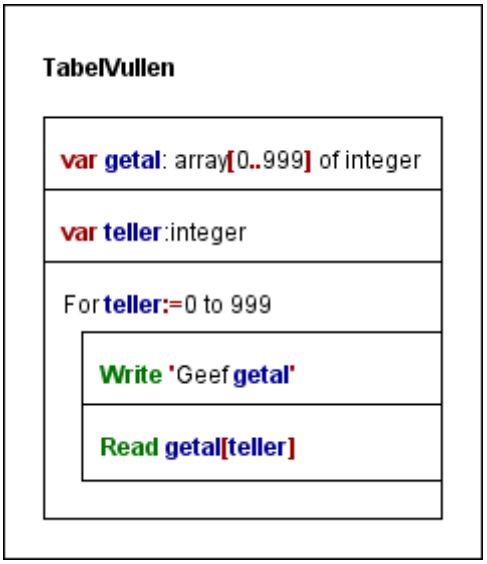
Dit is een ééndimensionale array met 7 rijen (of plaatsen). Elk item in de array moet van het type char zijn.

2.1.3. Lezen en schrijven

Door het gebruik van een array, wordt het PSD om de 10 getallen in te lezen en in het geheugen op te slaan, veel korter dan de 10 regels die we eerst hadden:

We laten dus een lus lopen om aan de 10 array-elementen een waarde te geven. De lus loopt van 0 tot 9 omdat de indexen van onze array ook van 0 tot 9 gaan.

Om dan 1000 getallen in te lezen en in het geheugen op te slaan in plaats van 10, wijzigt er niet veel aan bovenstaand PSD:



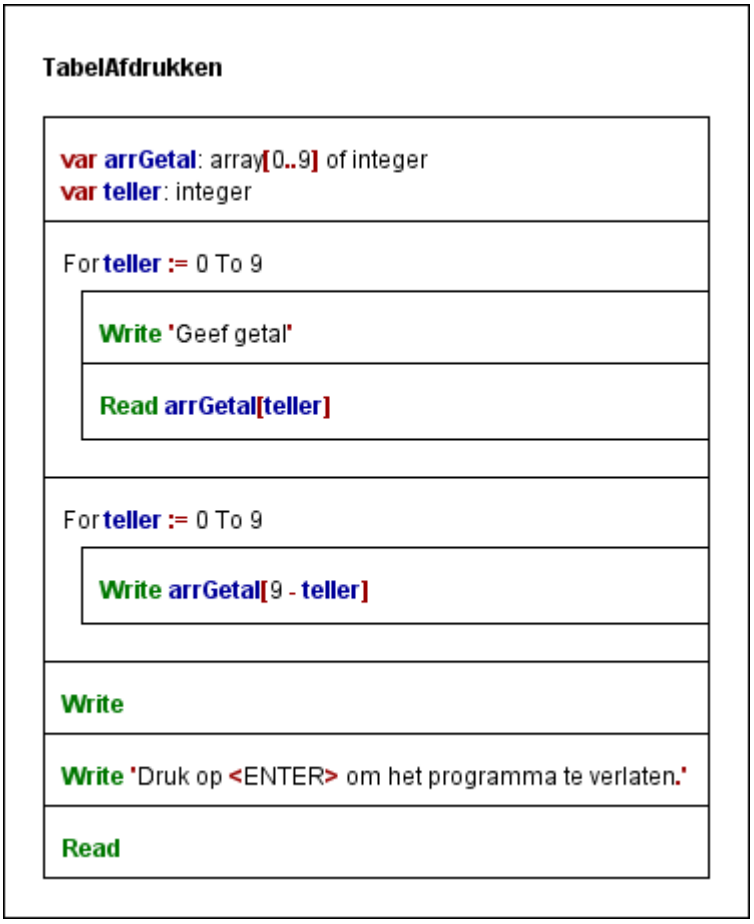
Om gebruik te kunnen maken van een iteratie, kunnen we elk array-element niet aanspreken met een constante index (bijv. **getal[0]**, **getal[1]**, **getal[2]**, enz.), maar gebruiken we hiervoor een index-variabele **teller** die we achtereenvolgens de waarden gaande van 0 tot en met 9 of van 0 tot en met 999 laten aannemen.

In al deze voorbeelden gaat het om **ééndimensionale arrays**: d.w.z. dat een element in de array kan aangesproken worden wanneer men één index kent.

2.1.4. Voorbeeld 1

Voorbeeld: Gegevens inlezen en afdrukken

Het PSD kan vrij snel aangepast worden om de ingelezen getallen in omgekeerde volgorde af te drukken:



De index kan een constante of een variabele zijn, maar ook een bewerking of expressie, bijv. **teller**+1 of zoals in het bovenstaande voorbeeld 9-**teller**.

In elk geval moet de waarde steeds een geheel getal zijn.

Opdracht

Maak bovenstaand psd en voer het uit in Lazarus.

2.1.5. Opmerkingen

In sommige programmeertalen worden andere notaties gebruikt:

- bijv. **getal(3)**, met ronde haken i.p.v. vierkante haken.
- bij sommige talen spreekt men van subscript in plaats van index.

- heel wat talen verplichten je om op voorhand plaats te reserveren voor elke array, d.w.z. je moet eerst aangeven hoe groot de array zal zijn, alvorens je de array kan gebruiken.
  Bij sommige talen is het zelfs mogelijk om de grootte van een array achteraf nog aan te passen. Dit noemt men **dynamische arrays** (zie verder in de cursus). In het andere geval spreekt men van **statische arrays**.
- wanneer er met de array-elementen gerekend dient te worden (bijv. een som maken van alle elementen), is het belangrijk dat de tabel vooraf **geïnitialiseerd** wordt.
  Dit initialiseren dient te gebeuren door een lus te laten lopen en alle array-elementen één voor één een initiële waarde te geven. In sommige programmeertalen bestaan hier kortere statements zodat je de array zonder lus kan initialiseren.

2.1.6. Voorbeeld 2

Soms is het handig om de index een betekenis te geven. We tonen dit aan aan de hand van een voorbeeld.

Voorbeeld: Frequentie bepalen

De bedoeling is om een onbepaald aantal getallen in te lezen tussen 11 en 20 om hiervan de frequentie te bepalen, maw hoe vaak komen de getallen 11 t/m 20 voor?

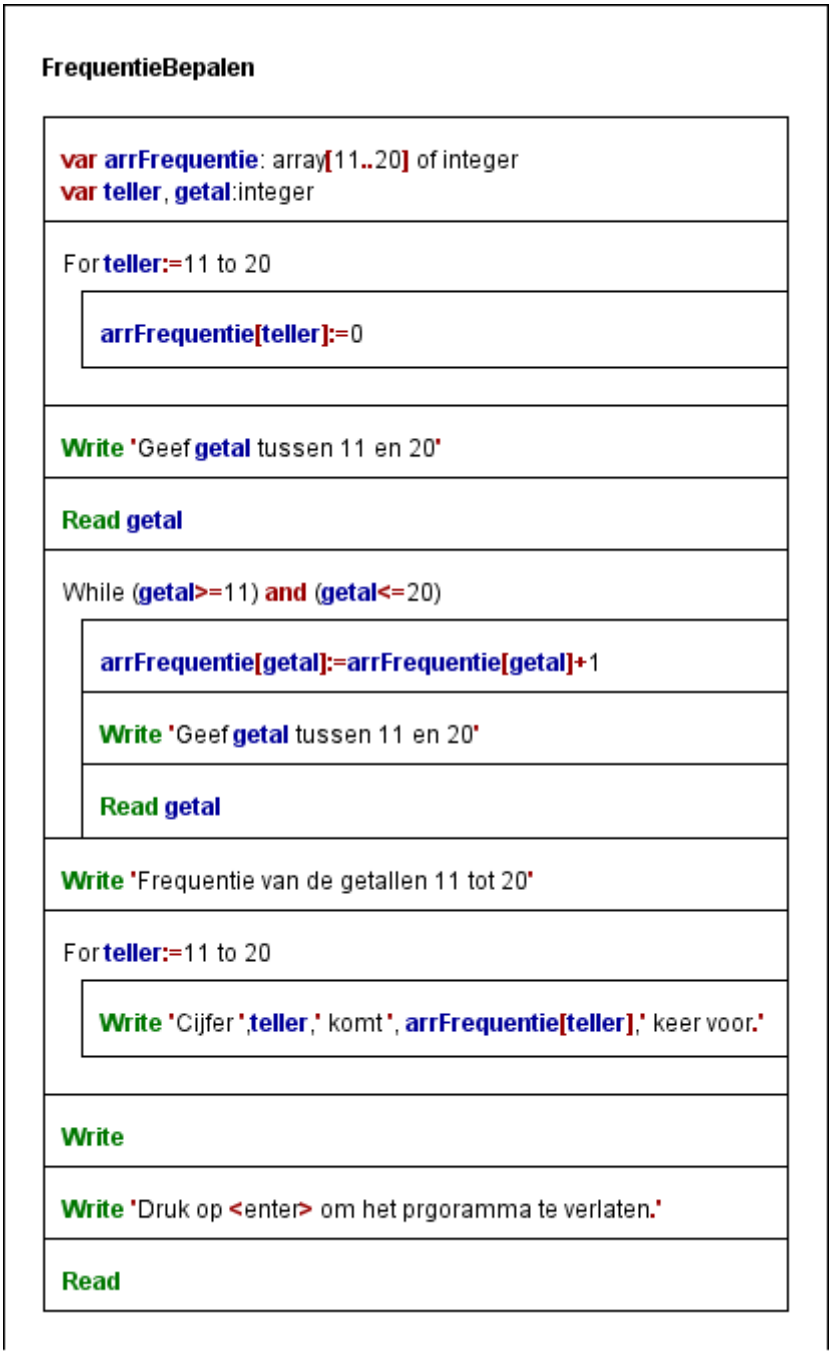
Dus er dienen getallen ingegeven te worden tussen 11 en 20 en er dient vervolgens bijgehouden te worden hoe vaak het cijfer 11, 12, t/m 20 is ingegeven.

In geval een getal ingegeven wordt buiten deze grenzen, wordt het resultaat afgedrukt en eindigt het programma. Hiervoor wordt een ééndimensionale array gebruikt van 10 array-elementen. Deze wordt vooraf op 0 geïnitialiseerd.

De array wordt in dit geval gedefinieerd met index 11 tot 20 in plaats van met index 0 tot 9. We gebruiken het ingelezen getal ook als index voor de array.

Vervolgens wordt de ingave gevraagd van de getallen. Wanneer een ingave gebeurt buiten de waarden 11 t/m 20, stopt de verwerking en wordt de inhoud van de array afgedrukt.

Het psd



Opdracht

Maak het psd en test het uit in Lazarus.

2.1.7. Opgave: Tabel van 10

Vul een tabel van 10 elementen met willekeurige getallen tussen 1 en 1000 (random). Nadat de hele tabel gevuld is, druk je de hele tabel af.

Tot slot bereken je de som van de tien elementen en ook die waarde druk je af.

2.2. Dynamische arrays

In praktijk is het niet altijd mogelijk om bij de start van je programma reeds te weten hoe groot je je array moet maken.

Als je de grootte van je array wil aanpassen tijdens je programma, spreken we van een **dynamische array**.

Voorbeeld

Stel je voor dat je een array wil gebruiken om waarden die je uit een bestand leest op te slaan.

Wanneer je zeker weet dat er bijvoorbeeld altijd 50 waarden in het bestand zullen staan, kan je dus veilig een statische array aanmaken van 50 elementen groot.

Wat vaker voorkomt in het 'echte leven' is dat je geen idee hebt hoeveel waarden er in het bestand zullen staan. Je kan dan kiezen om gewoon een statische array te gebruiken die erg groot is, zodat je vrij zeker weet dat alle waarden er wel in gaan passen. Maar de nettere oplossing is om een dynamische array te maken die 'meegroeit' met het aantal waarden wat je er in opslaat.

2.2.1. Gebruik - Hoe maak je een dynamische array?

Declareer je array zonder index:

```
var arrGetallen: array of integer
```

Pas de lengte van je array aan met de instructie

```
SETLENGTH(arrGetallen,lengte)
```

Opmerkingen:

Je kan de lengte van je array aanpassen voor elk item dat je toevoegt. Op die manier heb je een array die exact de juiste grootte heeft. Het nadeel van deze manier van werken is dat je array erg verspreid bewaard wordt in het geheugen.

Je kan ook een startlengte meegeven. Stel je bent zeker dat je altijd minimaal 50 gegevens in je array zal hebben. Als de limiet bereikt is, kan je dan je array vergroten met bijvoorbeeld 10 extra elementen in plaats van telkens één element toe te voegen. Op deze manier geraakt je array minder versnipperd.

Met SETLENGTH() bepaal je niet enkel de grootte van je array, maar je legt ook de indices al vast, nl van 0 tot lengte - 1.

De kleinste en grootste index van je array kan je opvragen met:

```
LOW(arrGetallen)
HIGH(arrGetallen)
```

2.2.2. Voorbeeld: Namen inlezen

Je wil de namen van een willekeurig aantal studenten inlezen. We voeren minstens 10 namen in. Nadien gaan we, indien nodig, de array telkens vergroten met 5 plaatsen.

Als je als naam "" ingeeft, moet de invoer stoppen.

Nadien vraag je de hoeveelste naam de gebruiker wil zien.

Het psd

Opdracht

Maak het psd en test het uit in Lazarus.

2.2.3. Opgave: Grootste van de klas

Via het scherm worden de lengtes van de leerlingen van de klas ingelezen. Een klas heeft minstens 15 studenten.
Bij ingave van een 0 voor de lengte, stopt het programma.
Geef de gemiddelde lengte van die klas.
Geef de grootste lengte en de kleinste lengte en zeg waar die staan in de tabel.

2.3. Multidimensionale arrays

Tot nu hebben we alleen ééndimensionale arrays gezien. Het is echter ook mogelijk arrays met 2 of meer dimensies aan te maken. Het maximale aantal dimensies is afhankelijk van de programmeertaal.

We beperken ons in deze cursus tot tweedimensionale arrays. In volgend onderdeel gaan we hier verder op in.

2.3.1. Waarom?

Stel dat we een array maken met de lonen van een aantal werknemers voor een bepaalde maand (we noemen deze tabel bijvoorbeeld **lonenJanuari**). Daarna willen we ook de gegevens van de volgende maand februari bewaren, dan maken we een tweede array aan en eventueel voor maart een derde array, enz. . Dit kan je voor de volgende maanden verderzetten.

	lonenJanuari
1	1140,31
2	1016,36
3	1437,78
4	1090,73
5	1388,2
6	1214,68
7	1115,52
8	1016,36

	lonenFebruari
1	1189,89
2	1338,63
3	1264,26
4	1214,68
5	1264,26
6	1090,73
7	1016,36
8	1239,47

	lonenMaart
1	1264,26
2	1487,36
3	1189,89
4	1264,26
5	1065,94
6	1065,94
7	1363,41
8	1090,73

Als je de lonen voor een gans jaar op deze manier zou willen bijhouden, heb je 12 arrays nodig.

2.3.2. Gebruik

Als we alle lonen van een heel jaar in één array willen stoppen, dan maken we beter gebruik van een tweedimensionale array.

Onderstaand voorbeeld toont de array met de lonen van 8 werknemers voor de 12 maanden van het jaar:

lonen

	1	2	3	4	5	6	7	8	9	10	11	12
0	1140,31	1189,89	1264,26	1140,31	1189,89	1264,26	1140,31	1189,89	1264,26	1264,26	1140,31	1189,89
1	1016,36	1338,63	1487,36	1016,36	1338,63	1487,36	1016,36	1338,63	1487,36	1487,36	1016,36	1338,63
2	1437,78	1264,26	1189,89	1437,78	1264,26	1189,89	1437,78	1264,26	1189,89	1189,89	1437,78	1264,26
3	1090,73	1214,68	1264,26	1090,73	1214,68	1264,26	1090,73	1214,68	1264,26	1264,26	1090,73	1214,68
4	1388,2	1264,26	1065,94	1388,2	1264,26	1065,94	1388,2	1264,26	1065,94	1065,94	1388,2	1264,26
5	1214,68	1090,73	1065,94	1214,68	1090,73	1065,94	1214,68	1090,73	1065,94	1065,94	1214,68	1090,73
6	1115,52	1016,36	1363,41	1115,52	1016,36	1363,41	1115,52	1016,36	1363,41	1363,41	1115,52	1016,36
7	1016,36	1239,47	1090,73	1016,36	1239,47	1090,73	1016,36	1239,47	1090,73	1090,73	1016,36	1239,47

Deze array bestaat uit rijen en kolommen.
Elk element uit de array wordt nu aangeduid met de rij en de kolom. De cel die grijs gekleurd is in het voorbeeld, is de cel **lonen[3,6]**.

Om een element uit een **tweedimensionale array** aan te spreken, heb je dus **2 indices** nodig, nl. een **rijaanduiding** en een **kolomaanduiding**!
De eerste index is steeds de rijaanduiding en de tweede index is de kolomaanduiding.

2.3.3. Declaratie

Ook een tweedimensionale array moet gedeclareerd worden.

```
var getal:array[0..7,1..12] of real;
```

Dit is een tweedimensionale array met 8 rijen en 12 kolommen. Elk item in de array moet van het type real zijn.

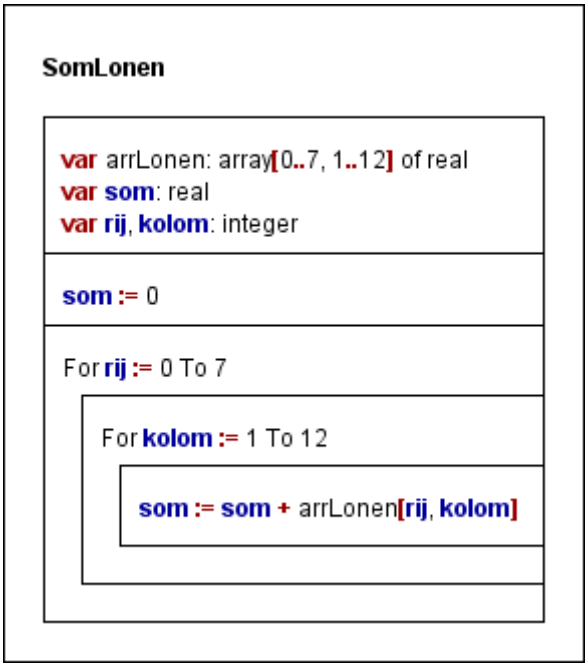
Als je een dynamische tweedimensionale array voor letters wil declareren, gebruik je volgende instructie:

```
var letters:array of array of char;
```

2.3.4. Voorbeeld 1

Om alle elementen van een tweedimensionale array in een iteratie te lezen, heb je dus ook **twee tellers** (indices) nodig, nl. één teller voor elke dimensie, oftewel één teller voor elke index (één voor de rij-index en één voor de kolom-index).

Om bijvoorbeeld de som van alle elementen (alle lonen) van deze array **lonen** te berekenen zou je volgend PSD kunnen gebruiken:



In dit PSD wordt de som van de lonen gemaakt door de array rij per rij te overlopen: eerst worden alle kolomelementen van de eerste rij bij elkaar opgeteld, daarbij worden alle kolomelementen van de tweede rij geteld, vervolgens de derde rij, enz.

Ook bij tweedimensionale arrays kan de index uiteraard een constante of een variabele zijn, maar ook weer een bewerking of expressie.

2.3.5. Voorbeeld 2

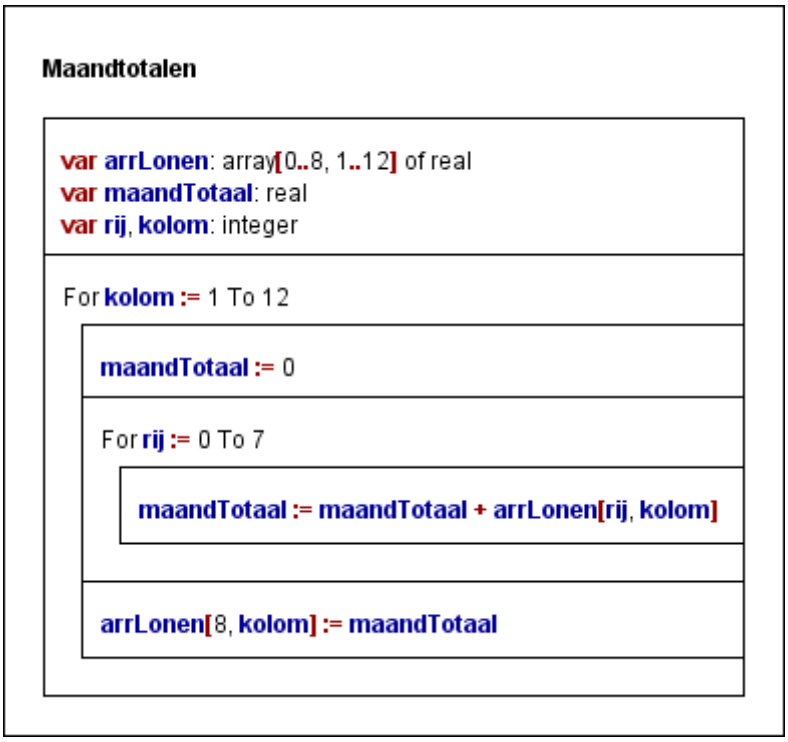
Typisch voor het werken met tweedimensionale arrays is het gebruik **van 2 in elkaar geplaatste lussen**: één om de rijen te overlopen en één om de kolommen te overlopen. Deze structuur wordt vooral gebruikt indien we elk element achtereenvolgens wensen te gebruiken, het is geen verplichte structuur.

Stel dat in de array **lonen** een 13e kolom voorzien is voor de rijtotalen (dus de jaarlonen), dan kan deze als volgt ingevuld worden:

Voor elke rij wordt de variabele **jaarloon** opnieuw geïnitialiseerd op 0. Per rij worden de kolomelementen opgeteld en tot slot wordt dit **jaarloon** in de 13e kolom van die rij bewaard.

Naar analogie van de rijtotalen kunnen ook kolomtotalen gemaakt worden. Een kolomtotaal in de eerste kolom stelt een totaal van het loon voor van alle werknemers voor de maand januari. Het kolomtotaal in de tweede kolom is het totaal van het loon van alle werknemers voor de maand februari, enz.

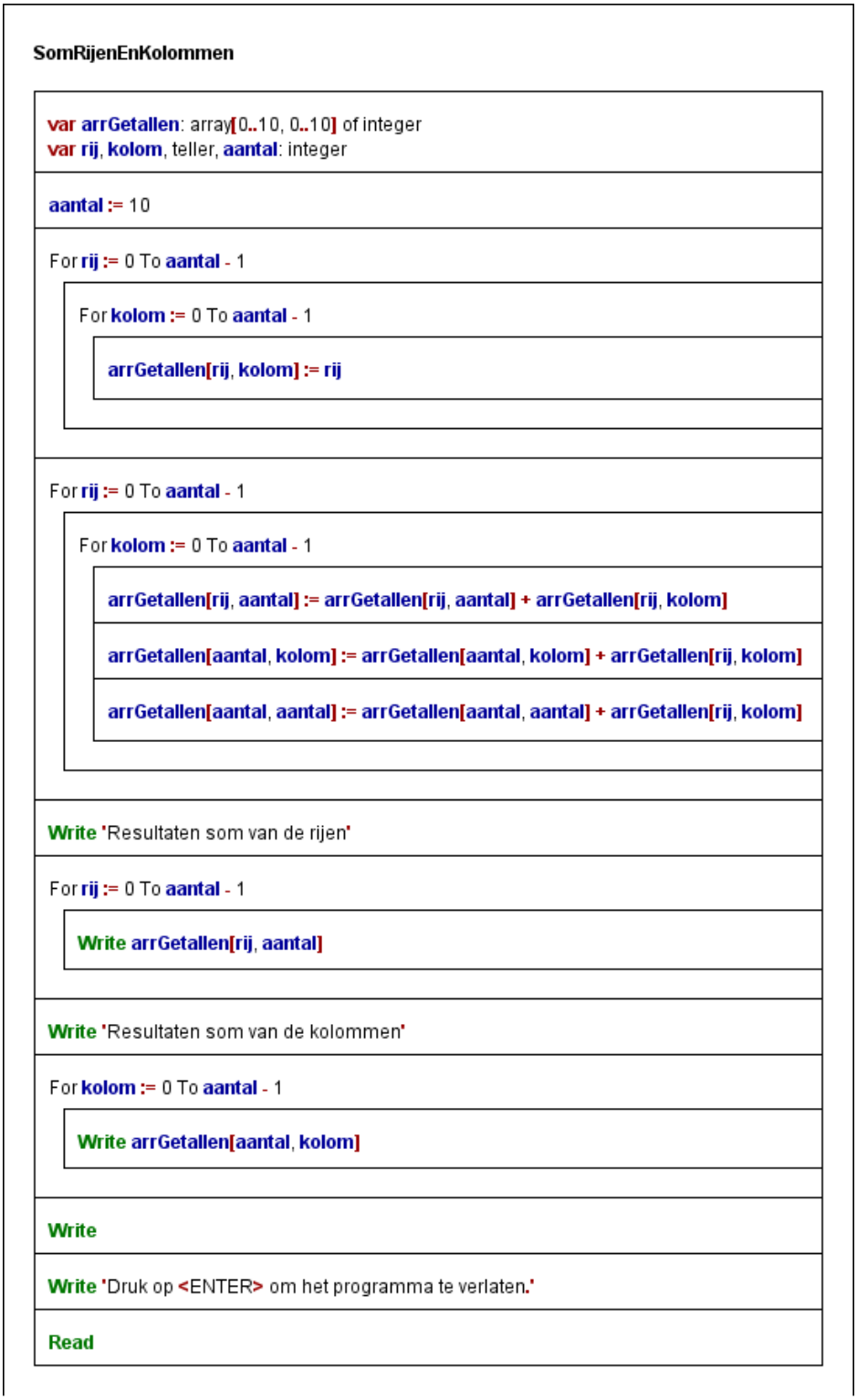
Toegepast op ons voorbeeld geeft dit:



2.3.6. Voorbeeld 3: Som van rijen en kolommen

Gegeven een array van 11 op 11, waarvan de eerste 10 rijen en kolommen met getallen zijn gevuld. Vul de tabel verder aan: in de 11de kolom komt telkens de som van de rij, in de 11de rij de som van de bovenliggende kolom.
Het element in de 11e rij en 11e kolom bevat de som van alle getallen.

Bekijk het PSD



Opdracht

Maak het psd en test het uit in Lazarus.

2.3.7. Opgave: Arrays optellen

Declareer 2 arrays met hetzelfde aantal rijen en kolommen. Vul de arrays met willekeurige getallen tussen 0 en 10.
Tel de twee arrays bij elkaar op.
Toon op het einde de twee arrays en de resultaatsarray.
Zorg ervoor dat je bij de start van het programma het aantal rijen en kolommen nog kan bepalen.

2.4. Opgaven

2.4.1. Opgave 1: Even getallen

Ontwerp een programma waarmee je aan de gebruiker 10 getallen opvraagt. Je toont als resultaat enkel de even getallen.

2.4.2. Opgave 2: Even getallen bis

Ontwerp een programma waarmee je aan de gebruiker getallen opvraagt. Je toont als resultaat enkel de even getallen.
Zorg dat de invoer stopt als je het getal 0 ingeeft.

2.4.3. Opgave 3: Som van de diagonalen

Initialiseer een tabel met evenveel rijen als kolommen. Het aantal rijen moet je ingeven bij de start.
Bereken de som van de diagonalen.

Hoofdstuk 3. Strings

Tekstvariabelen of strings (bijv. een naam) worden vaak beschouwd als een soort ééndimensionale array: elke cel bevat dan een karakter (d.w.z. een ASCII-waarde).

Voorbeeld:

- de string bevat “vdab”
- de eendimensionale tabel ziet er dan als volgt uit

v	d	a	b

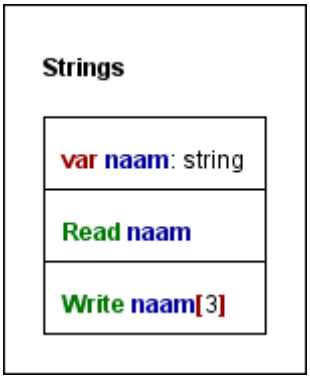
Let op! In tegenstelling tot arrays, heeft de eerste letter index 1!

Dit betekent dat je volgende taken kan uitvoeren:

een bepaalde letter van de string aanspreken

Met onderstaande code kan je de derde letter van de naam afdrukken

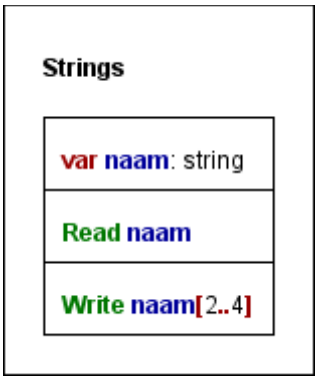
Vb bij de naam 'Ruben' zal de letter 'b' getoond worden.



een deel van de string afdrukken

Met onderstaand voorbeeld halen we de 2de, 3de en 4de letter uit de tekst.

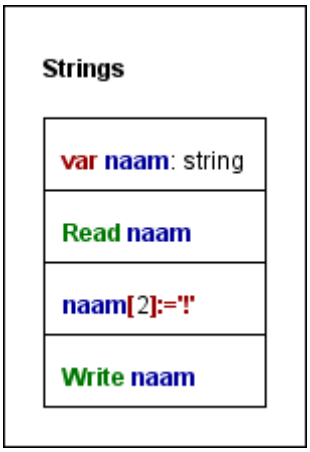
Vb bij de tekst 'Ruben' geeft dit 'ube'



een bepaalde letter veranderen

In volgend voorbeeld wordt de waarde van de tweede letter veranderd in een '!'.

Vb de naam 'Ruben' wordt dan 'R!ben'.

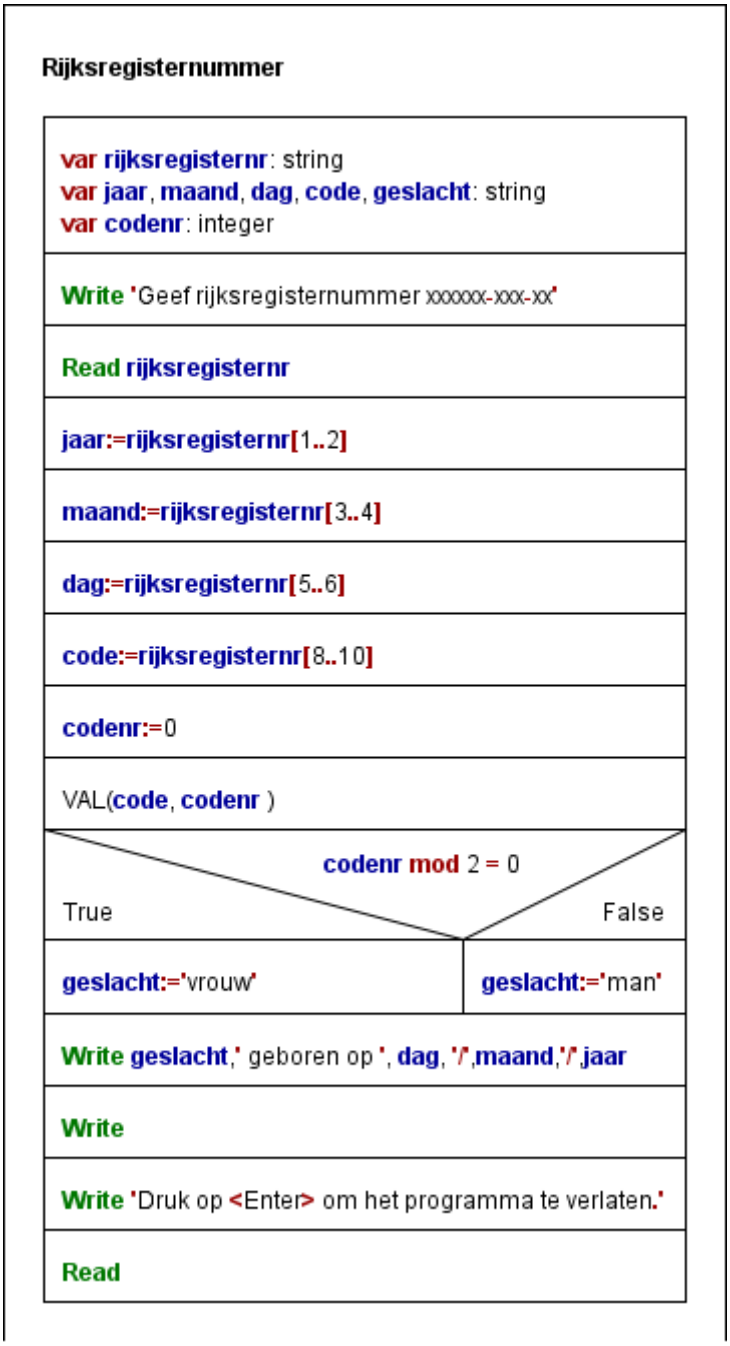


3.1.1. Voorbeeld: Rijksregisternummer

Neem als voorbeeld het volgende rijksregisternummer: 720202-900-81

- De eerste 6 cijfers stellen de geboortedatum voor (in het formaat JJMMDD).
- De volgende 3 cijfers geven aan de hoeveelste geboorte van die dag, waarbij de even getallen gebruikt worden voor vrouwen en de oneven getallen voor mannen.
- De laatste 2 cijfers vormen het controlegetal.

Het programma dat op basis van een ingevoerd rijksregisternummer, de volgende tekst op het scherm brengt: “Vrouw/Man geboren op dd/mm/jj”, ziet er als volgt uit:



3.1.2. Opgave: Letters tellen

Lees een woord in. Tel hoe vaak elke letter voorkomt.
We houden geen rekening met het verschil tussen hoofdletters en kleine letters.

Voor deze oefening mag je ervan uit gaan dat het woord volledig in kleine letters ingegeven wordt.

Hoofdstuk 4. Opgaven

4.1. Opgave 1: Product

Je declareert 3 arrays met telkens 5 waarden. De eerst 2 arrays vul je op met willekeurige getallen tussen 0 en 10. De derde array vul je op met het product van overeenkomstige velden uit array 1 en 2. Je toont als resultaat de 3 arrays.

4.2. Opgave 2: Temperaturen

De bedoeling is om gedurende één week 3 temperaturen per dag bij te houden, nl. een ochtendtemperatuur, een middagtemperatuur en een avondtemperatuur. Hiervoor kan een array gebruikt worden van 7 rijen (7 dagen) en 3 kolommen (3 temperaturen).

Tijdens de ingave van alle temperaturen wordt de gemiddelde temperatuur per dag berekend. Achteraf wordt de gemiddelde weektemperatuur berekend.

4.3. Opgave 3: Aantal woorden in een zin

Lees een zin in. Tel het aantal woorden in de zin.

4.4. Opgave 4: Hoe vaak komt elke letter voor in een zin?

Tel hoe vaak elke letter voorkomt in een zin.
Hou rekening met hoofdletters en kleine letters. 'E' en 'e' staan voor dezelfde letter.

Toon op het einde alle letters die voorkomen en het aantal keer dat die letter voorkomt.

4.5. Opgave 5 Coderen en decoderen

Om een tekst te versleutelen kiest men eerst een geheim sleutelwoord, bijvoorbeeld ‘vpw’. Dit sleutelwoord mag geen spaties bevatten.

Het sleutelwoord wordt herhaald totdat een string verkregen wordt met dezelfde lengte als de oorspronkelijke tekst die moet gecodeerd worden.

Vervolgens telt men de corresponderende letters uit de originele tekst en het sleutelwoord bij elkaar op. Bij deze optelling worden de letters A tot Z beschouwd als de getallen 1 tot 26, een spatie krijgt de waarde 0. De optelling wordt uitgevoerd modulo 27 (het aantal letters uit het alfabet plus de spatie).

Het ontcijferen van een boodschap gebeurt analoog. Men trekt de waarde van de overeenkomstige codeletter af van de gecodeerde letter.

Maak een programma dat de keuze geeft tussen coderen of decoderen. Werk alleen het coderen uit.

Voorbeeld uitvoer:

4.6. Opdracht voor coach: Statistiek

Van de lengtes van een groep personen (max. 99, stoppen met lengte = 0), dient het volgende bepaald te worden:

- gemiddelde lengte
- standaardafwijking
- het aantal personen wiens lengte ligt binnen éénmaal de standaardafwijking van het gemiddelde
- het aantal personen wiens lengte ligt tussen éénmaal en tweemaal de standaardafwijking tov het gemiddelde
- het aantal mensen buiten het bereik

Ter info: de formule van de standaardafwijking:

$$\sqrt{\frac{\sum (X_i - AVG)^2}{n}}$$

waarbij:

AVG	=	gemiddelde
Xi	=	het ide element
n	=	aantal elementen
Σ	=	de som van hetgeen volgt

Of meer gedetailleerd uitgelegd: je berekent het gemiddelde van alle elementen en dat noem je AVG. Vervolgens ga je alle elementen af, en voor elk element neem je het verschil tussen dat element en AVG, en van dat verschil neem je het kwadraat. Al die kwadraten tel je dan op, en je deelt het eindresultaat door het aantal elementen. Tenslotte neem je van dat getal de vierkantswortel.

Stuur je oplossing van deze oefening ter verbetering aan je **coach**. Geef je bericht als onderwerp "**Statistiek**".

Hoofdstuk 5. Einde cursus

5.1. Eindoefening

Bij deze cursus hoort ook een **eindoefening**. In die eindoefening komen de belangrijkste zaken van deze cursus terug aan bod.

Stuur een bericht aan je **coach** met als onderwerp "**Opdracht eindoefening Arrays en Strings**".

5.2. Wat nu?

Je hebt nu het Programmatie logica - Arrays en strings afgerond.

Heb je het deel **Procedures en functies** nog niet doorgenomen, dan kan je dat nu doen.

Als je na deze drie delen nog zin hebt in meer, kan je nog verder verdiepen in de delen

- **Sorteren**: sorteren van gegevens
- **Bestanden**: omgaan met bestanden

Na het doornemen van **Basisstructuren**, **Procedures en functies** en **Arrays en strings** kan je voor de opleiding Programmatielogica een getuigschrift met resultaatsvermelding krijgen. Hiervoor leg je een eindtest af in één van de VDAB-opleidingscentra. De nodige informatie over de verschillende centra vind je terug in de module **Intro Informatica**.

Ligt je focus meer op **webapplicaties** en wil je graag al beginnen aan een concrete taal, dan zijn onze cursussen **Javascript** of **Inleiding tot PHP** misschien wel interessant voor jou. Je neemt dan best eerst nog het onderdeel **Procedures en functies** door.