Vergelijkingen Aldfaer sjabloontaal

De sjabloon- of scripttaal is ontwikkeld om in Aldfaer* rapporten te schrijven die in een willekeurige browser, zoals Firefox, Edge of Opera, getoond kunnen worden. Dus alles is erop gericht om **HTML output** te genereren en te bewaren.

Er zijn de volgende vier basis vergelijkingen te onderscheiden:

|	alternatief
[: :]	**OF** blok
[& &]	**EN** blok
[? ?]	**TEST** blok

Alternatief **|** los en in samenwerking met **[: :]**

```
_BIRTDATE()|_BAPTDATE()
```

In de eerste plaats wordt gekeken of de geboortedatum **TRUE** is, zo ja, dan wordt die afgedrukt, zo niet, dan wordt pas gekeken of de tweede voorwaarde **TRUE** is.
Als beiden **FALSE** zijn, wordt er niets afgedrukt.
De haakjes `()` worden (helaas) ook mee geprint, die kunnen dus beter achterwege blijven.

```
[:*_BIRTDATE():]|[:~_BAPTDATE():]
```

Wanneer de eerste voorwaarde **TRUE** is, wordt de geboortedatum mét symbool afgedrukt: `* 12-12-1965`. Als alleen de tweede voorwaarde **TRUE** is, zien we de volgende output: `~ 22-12-1965` en in geval beiden **FALSE** zijn, dan is er géén output.
De haakjes `()` worden hier niet meegeprint!

OF blok **[: :]**

```
[:_SURN is in _BIRTPLAC geboren.:]
```

Binnen de **OF** haken wordt alles geëvalueerd, m.a.w. is het gevraagde **TRUE** of **FALSE**.
Het is **TRUE** zodra **óf** de naam **óf** de geboorteplaats bestaat in de database.
Wanneer alleen de achternaam **TRUE** is en de geboorteplaats **FALSE**, krijg je het volgende: *Francke is in geboren* en dat is geen duidelijke zin om in een rapport op te nemen.
Daarom is het niet verstandig twee voorwaarden in een **OF** blok op te nemen, kijk dan of er een betere constructie is die wel goede output genereert.
`[:Dit is een test:]` is **FALSE** want er gebeurt niets, er is nl. geen tag aanwezig om te evalueren.
Let op de niet aaneengesloten haken en spatie in de volgende code:
`[: _VAROUT(x, "0"):]` geeft geen waarde van 'x' terug omdat de parser bij gebruik van de **OF** haken een extra spatie leest vóór de `_VAROUT(x)`.
Schrijf altijd aaneengesloten `[:_VAROUT(x, "0"):]` om output te krijgen!

EN blok **[& &]**

Alles binnen deze haken wordt geprint zodra **alle** tags **TRUE** zijn.
De volgende code geeft: `[& _NAME is geboren op _BIRTDATE &]`
alléén output als zowel `_NAME` en `_BIRTDATE` **TRUE** zijn.

TEST blok **[? ?]** in samenwerking met **[: :]**

Alles binnen deze haken wordt **nooit** geprint. Er wordt gekeken of de gebruikte tag(s) **TRUE** is/zijn of **FALSE**

```
[:[?_BIRTDATE?]De geboortedatum is ingevuld.:]
```

Geeft alléén de output: *De geboortedatum is ingevuld.* als de datum ingevuld c.q. **TRUE** is.

Zoeken met het **[? ?]** blok in samenwerking met **|** en **[& &]**

```
[?_BIRTDATE?]|[&De geboortedatum is niet ingevuld&]
```

Twee blokken gescheiden door het alternatiefteken.
Eerst wordt getest of de geboortedatum is ingevuld, indien **TRUE** dan wordt het tweede testblok **niet** geprint, want het alternatief is **niet** nodig. Is de datum **FALSE** dan wordt het **EN** blok geëvalueerd en dat is altijd **TRUE**, want de tags ontbreken en de output is daarom *De geboortedatum is niet ingevuld*

In plaats van de vergelijkingsblokken te gebruiken kan de 'zoekfunctie' ook op *if then else* methode worden geschreven:

```
[?_BIRTDATE?]?
      _BEGI
      _ELSE
            De geboortedatum is niet ingevuld
      _END
```

Vergelijken met het **[? ?]** blok in samenwerking met **[::]**

```
[?_NAME._UPPE < B?]
```

Een testblok met binnen de haken een vergelijking met het *kleiner dan* teken. Er wordt dus gekeken of naam alfabetisch kleiner is dan B, zo ja dan wordt een naam beginnend met een **A** geprint. Door het filter `_UPPE` worden zowel de onderkast **b** als de bovenkast **B** in het vergelijk meegenomen. (**b** is alfabetisch groter dan **a** of **a** is alfabetisch kleiner dan **b**)
`[?_NAME._LOWE < b?]` geeft hetzelfde resultaat, de naam wordt nu in kleine letters geschreven.
_UPPE staat voor *upper case* = hoofdletters *(ik zou voor _UPPCA gekozen hebben)*
_LOWE staat voor *lower case* = kleine letters *(ik zou voor _LOWCA gekozen hebben)*

```
[:[?_CHILAMOU == 1?]Kind:]|[&Kinderen&]
```

print, ervan uitgaande dat er één of meer kinderen zijn, het woord *'Kind:'* indien er slechts één kind is en *'Kinderen:'* indien er meer kinderen zijn.
Wanneer er geen kinderen zijn, kiest de bovenstaande formule het alternatief, omdat de eerste tag **FALSE** is.

```
[:[?_CHILAMOU == 1?]Kind:]|[:[?_CHILAMOU > 1?]Kinderen:]|[&geen kinderen&]
```

Drie codeblokken gescheiden door het alternatiefteken **|**
Het eerste blok test of er een(1) kind is, indien **TRUE** dan is *Kind;* de output, de andere blokken zijn **FALSE**. Is het eerste blok **FALSE** dan komt het alternatief in actie, meer kinderen dan 1, indien **TRUE** dan is de output *Kinderen:* het alternatief is niet aan de orde. Zijn de eerste twee blokken **FALSE** dan is het alternatief van blok drie aan de beurt en geeft als output: *geen kinderen*

NOT vergelijking *(zo overgenomen uit de help, kan hier niets mee)*
Een **NOT** kun je maken door te vergelijken met niets.

```
[&[?_DIALEXP(X, 3) == ?][?_DIALEXP(X, 2)?][:functie doeiets:]&]
```

Hier wordt de functie aangeroepen als 3 leeg is, en 2 gevuld.

`[?_FIRS==?]` zal een **TRUE** opleveren als het **NOT** de eerste is binnen een set en is eigenlijk dus hetzelfde als `[?_FIRS("1;0")==?]`

Operator en operand

In de wiskunde en in computerprogrammering is een **operator** een *teken* dat een specifieke wiskundige of logische actie of proces vertegenwoordigt. Bijvoorbeeld **x** is een **rekenkundige** operator die vermenigvuldiging aangeeft, terwijl **&&** een logische operator is die de logische **AND**-functie in programmeren vertegenwoordigt.
In de wiskunde en programmering is een operand een *invoerwaarde* voor een operator
In 4 + 6 = 10 zijn de getallen **4** en **6** de operanden voor de operator **=**

In Aldfaer kunnen we de volgende operatoren onderscheiden:
== → gelijk aan
! → ongelijk aan
> → groter dan
< → kleiner dan
>= → groter of gelijk aan
<= → kleiner of gelijk aan

Deze operatoren kunnen in een vraagstelling worden gebruikt bijvoorbeeld:

```
als Jan groter is dan Piet dan
     koopt Jan een broek
anders
    gebruikt Jan de broek van Piet
```

```
if Jan > Piet then
begin
    koopt Jan een broek
else
    gebruikt Jan de broek van Piet
end
```

---

1) in de Aldfaer sjabloontaal op de *standaard* manier:

```
[:[?Jan > Piet?]koopt Jan een broek:]|[&gebruik broek van Piet&]
```

---

2) en met *if then else* methode:

```
[? Jan > Piet ?]?
_BEGI
    koopt Jan een broek
_ELSE
    gebruik broek van Piet
_END
```

De functie verder uitgewerkt zou hetvolgende kunnen zijn:

```
_FUNCBEGI(fBroek)
    :. maten in centimeters
   _VARDEF(iJan, 120)
   _VARDEF(iPiet, 110)
   [:[?_VAROUT(iJan, 0) > _VAROUT(iPiet, 0)?] koop broek :]|[&gebruik oude broek&]
   _VARDEL(iJan)
   _VARDEL(iPiet)
_FUNCEND
```

f staat voor functie fBroek
i staat voor integer of getal iJan en iPiet

Het is ook mogelijk de lengten van Jan en Piet in de functieaanroep door te geven en dat gaat als volgt:

```
_FUNCBEGI(fBroek, iJan, iPiet)
   _VARDEF(iLengteJan, [:iJan:])
   _VARDEF(iLengtePiet, [:iPiet:])
   [:[?_VAROUT(iLengteJan, 0) > _VAROUT(iLengtePiet, 0)?] koop broek :]|[&gebruik oude broek&]
   _VARDEL(iLengteJan)
   _VARDEL(iLengtePiet)
_FUNCEND
```

De lengten iJan en iPiet worden aan de variabelen `iLengteJan` en `iLengtePiet` doorgegeven en daar kan verder mee gerekend worden.

Het doorgeven van de waarden kan ook zo:

```
 _VARDEF(iLengteJan, iJan)
 _VARDEF(iLengtePiet, iPiet)
```

Jan en Piet zijn respectievelijk 130cm en 115cm lang en dan wordt de functie als volgt aangeroepen:

```
fBroek(130, 115)
```

uitkomst: omdat Jan **langer** is dan Piet, koopt hij een broek.

of

Jan en Piet zijn respectievelijk 120cm en 135cm lang en dan wordt de functie als volgt aangeroepen:

```
fBroek(120, 135)
```

uitkomst: omdat Jan **kleiner** is dan Piet, gebruikt hij de oude broek van Piet.

Personenset Aldfaer sjabloontaal

De scripttaal is ook nogeens hoofdlettergevoelig, **benz** is dus wat anders dan **BenZ** en de functies `_SETFILL` etc. dienen altijd met hoofdletters geschreven te worden en vergeet de *underscore* aan het begin niet. Als u Notepad++ gebruikt met de language Aldfaer aan, krijgen de functies e.d. verschillende kleuren afhankelijk van hun functie en doel, hier maak ik ze voor de duidelijkheid allemaal `blauw` of `groen` sjf

---

Een **set** is een *verzameling* van personen, een dergelijke set wordt ook wel een *personenset* genoemd.
Met de functie `_SETDEF` wordt een set aangemaakt, de complete syntax is:
`_SETDEF(mijnFamilie)`, mijnFamilie is de naam van de personenset, de set is nog leeg.

Vullen van een set.

Er zijn twee manieren om de set te vullen met personen uit uw database c.q. stamboom.

1) `_SETADD` voegt de actieve persoon in de database toe aan de set.
   De syntax is: `_SETADD(naam)` of `_SETADD(naam, _PERS)`
   `_PERS` is niet beslist noodzakelijk.

   voorbeeld: `_SETADD(mijnFamilie)`

2) `_SETFILL(mijnFamilie, _FAMIFILE)` zo wordt iedereen in de set opgenomen.
   De volledige syntax is: `_SETFILL(naam, bereik, selectie)`

   naam is bijv. mijnFamilie, sAllemaal, broers of BenZ
   bereik een selectie uit de stamboom:
   `_ASCE` voorgeslacht (kwartierstaat) van de huidige persoon;
   `_DESC` nageslacht (genealogie) van de huidige persoon;
   `_FAMIFILE` iedereen uit de stamboom;
   `_FAMITREE` vóór- en nageslacht van de huidige persoon.
   selectie:
   `_MALE` alleen mannen;
   `_FEMA` alleen vrouwen;
   `_SEXUN` alleen onbekende sexe;
   `_FAMI` alleen mannen en *ongehuwde* vrouwen.
   functie:
   `_FUNCBEGI(naamfunctie) broncode _FUNCEND`

   voorbeeld:
   `_SETFILL(mijnFamilie, _DESC, _FEMA)` -> de set mijnFamilie wordt gevuld met alle *vrouwelijke personen* uit het *nageslacht* van de *actieve persoon*.

   `_SETFILL(sAllemaal, _FAMITREE)` -> de set sAllemaal wordt gevuld met alle *personen* uit het *voor- en nageslacht* van de *actieve persoon*.

Sorteren van een set.

Het sorteren van een set gaat d.m.v. van de functie **_SETSORT** of te wel *sorteer set*.

De syntax is als volgt: **_SETSORT**(*naam, sleutel1, sleutel2, sleutel3*)

naam de naam van de te sorteren set
sleutel1 verplicht in te vullen bijv. **_SURN**

voorbeelden: **_SETSORT**(sAllemaal, **_SURN**)
_SETSORT(sAllemaal, **_SURN**, **_BIRTDATE**)
_SETSORT(sAllemaal, **_SURN**, **_NAMEFIRS**, **_BIRTDATE**)
_SURN staat voor *surname*, achternaam
_NAMEFIRS staat voor *first name*, voornaam
_BIRTDATE staat voor *date of birth*, geboortedatum

Het is mogelijk de volgorde van de sortering te bepalen, **a..z** of **z..a** / **0..9** of **9..0**

_SETSORT(*naam, sleutel1, sleutel2, sleutel3*)
is gelijk aan
_SETSORT(*naam, sleutel1, 0, sleutel2, 0, sleutel3, 0*)
De **0** behoort bij de voorgaande sleutel en staat voor default sortering: **a**..**z** of **0**..**9**

_SETSORT(*naam, sleutel1, 1, sleutel2, 1, sleutel3, 1*)
De **1** behoort bij de voorgaande sleutel en draait de sortering om: **z**..**a** of **9**..**0**

voorbeeld: **_SETSORT**(sAllemaal, **_SURN**, **_NAMEFIRS**, **_BIRTDATE**)
_SETSORT(sAllemaal, **_SURN**, 0, **_NAMEFIRS**, 0, **_BIRTDATE**, 0)
_SETSORT(sAllemaal, **_SURN**, 0, **_NAMEFIRS**, 0, **_BIRTDATE**, 1)
gesorteerd op achter- en voornaam en van oud naar jong in leeftijd

opmerking:
De functie **_SETREVSORT** is hiermee overbodig geworden
_SETREVSORT staat voor *reverse the sorting of the set*, m.a.w. keer de sortering om

Een set kan gesorteerd worden op **nageslacht** per generatie of tak, daar zijn de volgende sleutels voor:
_DESCGENE sorteer op nakomelingen per generatie
_DESCBRAN sorteer op nakomelingen per tak in de stamboom
_DESCGENE staat voor *descendant generation*, nakomelingen generatie
_DESCBRAN staat voor *descendant branch*, nakomelingen per tak

voorbeeld: **_SETSORT**(sAllemaal, **_DESCGENE**)
_SETSORT(sAllemaal, **_DESCBRAN**)
voorwaarde: de set moet eerst gedefinieerd, gevuld en op nakomelingschap gesorteerd zijn:
_SETFILL(sAllemaal, **_DESC**)

Een **set** kan, voor alle duidelijkheid, ook op **voorgeslacht** worden gesorteerd met **_ASCE**.
De syntax is **_SETFILL**(naam, **_ASCE**, selectie)
_ASCE staat voor ascendant, stijgend gerekend vanaf de actieve persoon.
selectie:
_MALE alleen mannen;
_FEMA alleen vrouwen;
_SEXUN alleen onbekende sexe;
_FAMI alleen mannen en *ongehuwde* vrouwen;

voorbeeld:
_SETFILL(mijnFamilie, **_ASC**, **_FEMA**) -> de set mijnFamilie wordt gevuld met alle *vrouwelijke personen* uit het *voorgeslacht* van de *actieve persoon*.

Verwijderen uit een set.

Er zijn vier functies die een persoon of personen uit een set verwijderen:

```
_SETSUB          verwijdert de huidige persoon uit de set
_SETTRUNNEXT     verwijdert alle personen ná de huidige persoon
_SETTRUNPREV     verwijdert alle personen vóór de huidige persoon
_SETCLR          verwijdert alle records
```

_SETSUB staat voor *subtract set*, verwijder de actieve persoon uit de set;
_SETTRUNNEXT staat voor *truncate next in set*, verwijder de volgende personen ná de actieve persoon;
_SETTRUNPREV staat voor *truncate previous in set,* verwijder de personen voorafgaande aan;
_SETCLR staat voor *clear set,* maak de set schoon of leeg.

De syntax is eenvoudig:
_SETSUB(naam) etc., naam is de set

In de help van Aldfaer staat een voorbeeld hoe deze functies te gebruiken om te bepalen of je broers hebt en of ze ouder zijn dan de actieve persoon in de stamboom.

```
_USES(DutchDateFormat, ";;;;"):. nodig voor geboortedatum [:DD-MM-YYYY:]
[?
_VARDEF(iTeller):. iTeller voor aantal broers
_SETDEF(sBroers):. set om broers in op te slaan
_FATHBEGI
      _CHILBEGI _MALE?:. is kind van het mannelijke geslacht?
            _BEGI:.
                  _SETADD(sBroers):.    voegtoe aan set
                  _VARADD(iTeller, 1):. telop door iTeller met 1 op te hogen
            _END
      _CHILEND
_FATHEND
_SETORDE(sBroers)?:.     is er een actieve persoon in de set?
_BEGI
      _SETSUB(sBroers):. zo ja verwijder haar/hem dan
_END
?]
```

De set sBroers wordt gedefinieerd en ook de variabele tellller, iTeller wordt gebruikt om het aantal broers te bepalen: **_VARADD**(iTeller, 1) en de set wordt tegelijkertijd gevuld met de diverse broers indien aanwezig.
iTeller is niet beslist noodzakelijk, het aantal broers kan ook met **_VAROUT**(sBroers, 0) bepaald worden.
Daarna wordt met **_SETORDE**(sBroers) naar de index van de actieve persoon gezocht en met **_SETSUB**(sBroers) wordt de actieve persoon uit set verwijderd.

```
:. (Vervolg programmacode)
_SETAMOU(sBroers, "0") > 0?]?:. is aantal broers groter dan 0?
_BEGI
      Er zijn _SETAMOU(sBroers) broers:
      _SETBEGI(sBroers)
            (_NUMB) _BIRTDATE([:DD-MM-YYYY:]) _NAME[: _NAMEPREF:] _SURN
      _SETEND
      _SETSORT(sBroers, _BIRTDATE):. sorteer op geboortedatum
      _SETTRUNNEXT(sBroers):. verwijder personen na actieve persoon
      Na de verwijdering zijn er nog _SETAMOU(sBroers) oudere broers in de set.
      _SETBEGI(sBroers)
            (_NUMB) _BIRTDATE([:DD-MM-YYYY:]) _NAMEFIRS[: _NAMEPREF:] _SURN
      _SETEND
_ELSE:.   aantal broers is nul
      [&[?_VAROUT(iTeller, "0") == 0?] Er zijn geen jongens in dit
      gezin.&]|[&[?_VAROUT(iTeller, "0") == 1?] Ik heb geen broers.&]
_END:.    einde _SETAMOU
```

Doorlopen of evalueren van een set.

Een **loop** of **lus** in een programma is programmacode dat telkens wordt herhaald.
In de Aldfaer sjabloontaal wordt tussen `_SETBEGI` en `_SETEND` elk element uit een personenset sequentieel of opeenvolgend geëvalueerd. De elementen in de set worden sequentieel of opeenvolgend benaderd.
Dat houdt in dat per geëvalueerde persoon zijn of haar gegevens beschikbaar zijn en opgevraagd kunnen worden, zoals daar zijn: geboortedatum, -plaats, ouders, kinderen en partner etc., dus alles wat er aan een persoon is gekoppeld in de database.

Zie hieronder een voorbeeld sjabloon hoe de set **sVoorGeslacht** wordt geëvalueerd en de output - *achternaam, voornaam etc.* - in een HTML tabel wordt wordt gezet.

Hetzelfde kan gezegd worden van `_FATHBEGI` en `_FATHEND`, `_CHILBEGI` en `_CHILEND`
De Aldfaer help zegt: *„begin iteratie vader en begin iteratie kinderen*"

Iteratie is **herhaling**. Deze begrippen uit de wiskunde en de informatica geven aan dat met een bepaald, zich herhalend proces een berekening of evaluatie kan worden uitgevoerd.
Bijvoorbeeld in een personenset kan worden gezocht naar de kinderen die geadopteerd zijn en die weer te bewaren in een set sAdoptieKind:

```
_CHILBEGI(_ADOP)
      _SETADD(sAdoptieKind):. voegtoe aan set sAdoptieKind
_CHILEND
```

en met `_SETAMOU(sAdoptieKind,"0")` is te zien hoeveel adoptiekinderen er zijn gevonden.

```
_Sjabloon _SJABNAME versie 0.1
_METAPATH([:_FILEPATH[::]tmp\:], "g"):.  wegschrijfpad vh rapport
_FILEMODE("html", ifNewer):.
_FILENAME("voorgeslacht_sjf.html")
_SETDEF(sVoorGeslacht):. s staat voor set
_SETFILL(sVoorGeslacht, _ASCE):. vul met voorgeslacht vanaf actieve persoon
_SETSORT(sVoorGeslacht, _SURN, _NAMEFIRS, _BIRTDATE)
_FUNCBEGI(fHoofdPersoon):. f staat voor functie
      _NAMEFIRS[:[?_NAMEPREF?] _NAMEPREF:] _SURN:.
_FUNCEND
:. --------------  begin van HTML rapport inclusief CSS code -------------------
_WRITBEGI
<!doctype html>
<html lang="nl">
<head>:.
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />:.
<meta charset="utf-8">:.
<meta http-equiv="X-UA-Compatible" content="IE=edge">:.
<meta name="viewport" content="width=device-width, initial-scale=1.0">:.
<title>voorgeslacht</title>:.
<style type="text/css">:. begin CSS code
  body {font-family:Verdana,Tahoma,Sans Serif,Serif,Arial;
      background-color:#FFFFFF}:.
  table{border-collapse:separate;border-spacing:1px;margin-bottom:
      12px;width:100%;background-color:silver}:.
  caption{border-radius:4px;color:#5C6976;background:#FFCD81;font-size:
           1.5rem;padding-left:15px;padding-top:7px;padding-bottom:7px}:.
  th{border-radius:4px;position: -webkit-sticky;position:sticky;top:0;font-size:
      1.0rem;letter-spacing:2px;color:#585858;background:#FFE1A6;padding-top:
      4px;padding-bottom:4px}:.
  td{padding-left:0.50rem}:.
  td:nth-child(1){text-align:right;padding-right:0.50rem;font-size:0.8rem}:.
  td:nth-child(2){font-weight:bold}:.
  td:nth-child(3){font-style:italic}:.
  tr:nth-child(odd){background:#FFF2D7}:.
  tr:nth-child(even){background:#FFFAF0}:.
  tr:hover{background:#FFE1A6;color:black}:.
</style>:. einde CSS code
</head>
  <table>:. begin tabel met gegevens
    <caption>voorgeslacht fHoofdPersoon met _SETAMOU(sVoorGeslacht)
      personen</caption>
   <tr>:.
      <th>nr</th>:.
      <th>achternaam</th>:.
      <th>voornaam</th>:.
      <th>geboortedatum</th>:.
      <th>geboorteplaats</th>:.
    </tr>
      _SETBEGI(sVoorGeslacht)
        <tr>:.
            <td>_NUMB</td>:.
            <td>_SURN </td>:.
            <td>_NAMEFIRS </td>:.
            <td>_BIRTDATE() </td>:.
            <td>_BIRTPLAC</td>:.
        </tr>
      _SETEND
   </table>:. einde tabel met gegevens
  </html>
_WRITEND:. einde rapport
```

Het bovenstaande programma komt uit de Aldfaer help, ik heb het hier en daar aangepast en verbeterd, o.a. door de regel met kolomkoppen en de CSS code toe te voegen. De naam van de personenset heeft een gerichte naam gekregen.
In de eerste 10 regels code, wordt het voorgeslacht, van de actieve persoon, uit de stamboom gehaald. Daarna volgt het output gedeelte wat uit twee delen bestaat, de CSS code en de HTML tabel.

Een HTML **<table> </table>** tabel zoals hierboven is gebruikt, bestaat uit minimaal twee onderdelen; het opschrift en de regels met kolomkoppen en data.

1) een **<caption> </caption>** ook wel het opschrift genoemd
2) **<tr> </tr>** de regels — `table row`
3) **<th> </th>** de kolomkoppen — `table header`
4) **<td> </td>** cellen met data — `table data`

```
<table>
  <caption>het opschrift boven de tabel</caption>
  <tr>
      <th>kolomkop 1</th>
      <th>kolomkop 2</th>
      <th>kolomkop 3</th>
  </tr>
  <tr>
      <td>datacel 01</td>
      <td>datacel 02</td>
      <td>datacel 03</td>
  </tr>
  <tr>
      <td>datacel 11</td>
      <td>datacel 12</td>
      <td>datacel 13</td>
  </tr>
</table>
```

In het **CSS** gedeelte worden de `tabel`, het `opschrift`, de `kolomkoppen`, `regels en datacellen` beschreven hoe ze eruit zullen gaan zien qua kleur, lettergrootte, vet of schuin, achtergrondkleur, ruimte tussen de letters enzovoorts.

`tr:hover{background:#FFE1A6;color:black}` geeft aan wat de achtergrond- en letterkleur wordt wanneer de muiscursor over een regel in de tabel gaat;

`td{padding-left:0.50rem}` geeft de ruimte rondom de tekst in een datacel aan;

`td:nth-child(2){font-weight:bold}` bepaalt dat in de tweede kolom de tekst **vet** wordt weergegeven.

Dit alles is te vinden op de site van het World Wide Web w3schools

Stringset Aldfaer sjabloontaal

Een **set** met een **verzameling** strings of stukjes tekst wordt ook wel een *stringset* genoemd in het Aldfaer jargon.

> Een string is een rij karakters of tekens, de string "Hallo, wereld!" bestaat uit 14 karakters waaronder de letters "H", "e" en tekens als " " (spatie) en "!" Elk karakter wordt in het computergeheugen opgeslagen als een getal.
> Een stringset werkt nagenoeg hetzelfde als een personenset: *records* respectievelijk *strings*.
> De strings worden los van elkaar opgeslagen in de set en zijn d.m.v. een index bereikbaar. Een string wordt bij invoer door **" "** - dubbele aanhalingstekens - omsloten of **[::]** gebruiken, wat m.i. minder duidelijk is en ook niet aansluit bij andere programmeertalen.

Met de functie **_SETDEF** wordt een stringset aangemaakt, de complete syntax is:
_SETDEF(cNewText), cNewText is de naam van de stringset, de set is nog leeg en dient gevuld te worden om ermee te kunnen werken of,
_SETDEF(cNewText, "Appingedam (Gr)") is een mogelijkheid om cNewText meteen te vullen met de string: Appingedam (Gr)

Vullen van een stringset

1) **_SETADD** voegt een string toe aan de set.

De syntax is: **_SETADD**(*naam*, string)
voorbeeld:

```
_SETADD(cNewText, "J.Janssens")
_SETADD(cNewText, "Dwarsstraat 65")
_SETADD(cNewText, "Appingedam")
```

Nu is de stringset cNewText gevuld met drie strings, gescheiden door een separator die bepaald werd door het programma zelf en dus voor de gebruiker onbekend. Met een volgende _SETADD wordt een volgende string toegevoegd, aansluitend op de voorgaande.

Om een string weer zichtbaar in een een rapport te krijgen is de volgende code nodig:

```
_SETEXP(naam, index, _SETSTRI)
naam    → _SETEXP(cNewText, 1, _SETSTRI)
straat  → _SETEXP(cNewText, 2, _SETSTRI)
plaats  → _SETEXP(cNewText, 3, _SETSTRI)
```

Naam, straat en plaats hebben de index 1 t/m 3 in de set cNewText
_SETEXP staat voor *export from set* exporteer een string met een bepaalde index uit de set
_SETSTRI staat voor *string in set* een string is tekst

_SETIN(naam, index, string) is een functie om een string met een bepaalde index te vervangen of tussen te voegen.

```
_SETIN(cNewTex, 2, "Kerkstraat 13a"):. adreswijziging doorgevoerd
```

2) **_SETFILL** is een andere functie om een stringset te vullen.

De syntax om te vullen met items gaat als volgt:

```
_SETFILL(naam, tekststring, separator)
```

voorbeeld:

```
_SETFILL(cNamen, "Jan;Piet;Klaas;Herman;", ";") of anders
_SETFILL(cNamen, [:Jan;Piet;Klaas;Herman;:], [:;:])
```

dit is één van de mogelijkheden van _SETFILL zie Aldfaer help voor meer mogelijkheden.

opmerking:

_SETFILL verwijdert de separator **na** Herman **niet**. Zo ontstaat er een laatste lege kolom, die in Excel zichtbaar wordt.

De volgende functie print getallen tot en met 10 als tekst; geen één twee etc

Eerst wordt de stringset *cGetallen* gedefinieerd en gevuld met `_SETFILL`, daarna volgt de vraagstelling; is het getal iGetal groter of gelijk aan 1 **EN** kleiner en gelijk aan 10 dan wordt het als *tekst* weergegeven, zo niet dan wordt er verder geëvalueerd, als iGetal nul is dan wordt de output *geen*, is iGetal geen nul **EN** groter dan 10 dan wordt iGetal geprint *zoals het is* ; bijv. *5*

De **c** in cGetallen staat voor character of letter en geeft aan dat het om een *stringset* gaat en dat het geen personenset is, die laat ik meestal beginnen met een **s**.
De **i** in iGetal staat voor integer of getal.

```
_FUNCBEGI(fGetal2Tekst, iGetal)
 _SETDEF(cGetallen):. stringset met getallen in letters
 _SETFILL(cGetallen, "één;twee;drie;vier;vijf;zes;zeven;acht;negen;tien;", ";")
 [& [? iGetal >= 1 ?][? iGetal <= 10 ?] &]?
 _BEGI
    _SETEXP(cGetallen, iGetal, _SETSTRI)._SUBS(1, -1)
 _ELSE
      [? iGetal == 0 ?]?
      _BEGI
         geen
      _ELSE
         iGetal
      _END
 _END
 _SETDEL(cGetallen)
_FUNCEND
```

Alternatief om de set te vullen met `[::]`

```
_SETFILL(cGetallen,[:één;twee;drie;vier;vijf;zes;zeven;acht;negen;tien;:],[:;:])
```

De functie is als volgt toe te passen:
iAantal is de uitkomst van een berekening n.l. 6.

```
Er zijn fGetal2Tekst([:_VAROUT(iAantal, 0):]) gevallen van
waterpokken geconstateerd.
```

Output: Er zijn zes gevallen van waterpokken geconstateerd.

De syntax om een string uit een stringset te benaderen is de volgende

`_SETEXP(naam, index, selectie)`

`_SETSTRI` geeft tekst uit de stringset terug en kan als selectie worden gebruikt in `_SETEXP`
`_SETEXP` staat voor *export from set* exporteer een string met een bepaalde index uit de set
`_SETSTRI` staat voor *string in set* een string is tekst
`_SETFILL` staat voor *fill a set* vul een stringset met item gesepareerd door een teken of separator

In het voorbeeld is iGetal de index en de telling van de items, de index begint bij **1** en niet bij **0** zoals in andere programmeertalen gebruikelijk is, dus 1 = één, 2 = twee … 10 = tien
De **0** (nul) wordt in de Aldfaer scripttaal gebruikt om de laatste uit een reeks aan te duiden.
Het voorbeeld laat zien dat na een `_ELSE` er weer een `_BEGI _ELSE _END` gebruikt kan worden.

Let op!

`_SETEXP` kan zonder `_SETBEGI` en `_SETEND` gebruikt worden.
Met `_SUBS(1, -1)` wordt de laatste letter - de separator - verwijderd, eigenlijk zou dat ‚vanzelf' moeten gaan, zoals voorzien is bij `_SETADD(naam, string)`

`_SUBS(x, y)` is een nieuw filter in versie 11 met zeer veel mogelijkheden om strings te bewerken.
Met de functie fMidString is het mogelijk om een bepaald stuk tekst in een string vanaf een bepaalde positie te isoleren, hierbij is dankbaar gebruik gemaakt van het filter `_SUBS(x, y)`

```
fMidString(string, vanaf, aantal)
fMidString("Amsterdam", 3, 4) → ster
```

```
_FUNCBEGI(fMidString, cString, iVanaf, iAantal)
      _SETDEF(cTekst, cString)
      _SETEXP(cTekst, 1, _SETSTRI)._SUBS(iVanaf)._SUBS(1, iAantal):.
      _SETDEL(cTekst)
_FUNCEND
```

met oude code, zonder `_SUBS(x,y)` zijn er aanzienlijk meer regels code nodig,
zie Aldfaer Forum

Verwijderen uit stringset

Het verwijderen van een string gaat in principe op dezelfde manier als bij een personenset. De interne functie `_SETSUB` wordt daarvoor gebruikt.
`_SETSUB(naam, string)` en daarbij moet string **exact** hetzelfde zijn als de te verwijderen string.

```
_FUNCBEGI(fDeleteFrom1):. vullen met _SETFILL
   _SETDEF(cNaam)
   _SETDEF(cNaam)
   _VARDEF(iIndex, 1):. begin bij 1
   _SETFILL(cNaam, "één;twee;drie;vier;vijf;zes;zeven;acht;negen;tien;", ";")
   _SETSUB(cNaam, "zes;"):. inclusief separator!!
   _SETBEGI(cNaam)
      _VAROUT(iIndex, 0) - _SETEXP(cNaam, iIndex, _SETSTRI)._SUBS(1, -1)
      _VARADD(iIndex, 1):. hoog iIndex met 1 op
   _SETEND
   8 - _SETEXP(cNaam, 8, _SETSTRI)._SUBS(1, -1):. string met index 8
   _SETDEL(cNaam)
   _VARDEL(iIndex)
_FUNCEND

_FUNCBEGI(fDeleteFrom2):. vullen met _SETADD
   _SETDEF(cNaam)
   _VARDEF(iIndex, 1)
   _SETADD(cNaam, "één")[::]_SETADD(cNaam, "twee")
   _SETADD(cNaam, "drie")[::]_SETADD(cNaam, "vier")[::]_SETADD(cNaam, "vijf")
   _SETADD(cNaam, "zes")[::]_SETADD(cNaam, "zeven")[::]_SETADD(cNaam, "acht")
   _SETADD(cNaam, "negen")[::]_SETADD(cNaam, "tien")
   _SETSUB(cNaam, "zes"):. nu zonder separator!!
   _SETBEGI(cNaam)
      _VAROUT(iIndex, 0) - _SETEXP(cNaam, iIndex, _SETSTRI)
      _VARADD(iIndex, 1)
   _SETEND
   8 - _SETEXP(cNaam, 8, _SETSTRI)
   _SETDEL(cNaam)
   _VARDEL(iIndex)
_FUNCEND
```

Opmerkingen over de code in fDeleteFrom1 en 2

1) spaties

`[::]` is nodig om spaties in de output te onderdrukken of beter gezegd niet weer te geven.
Wanneer `_setadd` strak achter elkaar worden gezet: `_SETADD(cNaam, "één")_SETADD(cNaam, "twee") etc` levert dat een waarschuwing op, de gewenste output wordt wel gegenereerd. Het is mi geen foute code, maar de parser is lichtelijk van slag af.
Een spatie er tussen geeft een verstoring van de output, er worden nl spaties toegevoegd waar je ze niet wilt

hebben. `_SETADD(cNaam, "één") _SETADD(cNaam, "twee") etc`
Allemaal onder elkaar kan natuurlijk ook, maar dan wordt de code minder overzichtelijk, vooral als het er veel zijn.
Daarom is gekozen voor het bovenstaande. `_SETADD(cNaam, "één")[::]_SETADD(cNaam, "twee")[::] etc`

2) index

De volgende opmerking in de Aldfaer help *"Het aardige is dat na verwijdering van records bij de overgebleven records het daarbij behorende oorspronkelijke nummer aanwezig blijft."* kan ik niet plaatsen.
"tien" heeft nu index 9, de strings schuiven gewoon een plaatsje op in set.
Blijft het aantal plaatsen in de set behouden?
Index 10 en hoger geven een lege string terug zonder een foutmelding of waarschuwing. Dit bewijst nog niet dat index 10 nog steeds bestaat, wel als bijv. index 15 een foutmelding zou geven.
Dit kan tot fouten in een sjabloon leiden, omdat bij er een te hoge index aanroep geen alarmbellen afgaan.
Zie de uitwerking in functie **fBackupAllen2**

3) verwarrend

Het is verwarrend dat de benadering van de strings in een set afhangt hoe de strings zijn toegevoegd, wel of niet de separator zelf verwijderen met `_SUBS(1, -1)` of zoeken inclusief separator met `_SETSUB` of zonder separator.

Kopiëren van stringset

`_SETFILL(DoelSet, BronSet)`
met bovenstaande syntax wordt set *BronSet* naar set *DoelSet* gekopieerd, beide sets moeten gedeclareerd of gedefinieerd zijn, om foutmeldingen te voorkomen.

voorbeeld:

```
_FUNCBEGI(fBackupAllen1)
   _SETDEF(cAllen)
   _SETDEF(cBackup)
   _VARDEF(iTeller, 1)
   _SETFILL(cAllen, "één;twee;drie;vier;vijf;zes;zeven;acht;negen;tien;", ";")
   _SETFILL(cBackup, cAllen):. maak backup van cAllen
   _SETSUB(cBackup, "zes;")
   _SETBEGI(cBackup)
      _VAROUT(iTeller, 0) - _SETEXP(cBackup, iTeller, _SETSTRI)._SUBS(1, -1)
      _VARADD(iTeller, 1)
   _SETEND
   _SETDEL(cAllen)
   _SETDEL(cBackup)
   _VARDEL(iTeller)
_FUNCEND

_FUNCBEGI(fBackupAllen2):. met _LOOPBEGI en _LOOPEND
   _SETDEF(cAllen)
   _SETDEF(cBackup)
   _VARDEF(iTeller, 1)
   _SETFILL(cAllen, "één;twee;drie;vier;vijf;zes;zeven;acht;negen;tien;", ";")
   _SETFILL(cBackup, cAllen):. maak backup van cAllen
   _SETSUB(cBackup, "zes;")
   _LOOPBEGI
      _VAROUT(iTeller, 0) - _SETEXP(cBackup, iTeller, _SETSTRI)._SUBS(1, -1)
      _VARADD(iTeller, 1)
       [? _VAROUT(iTeller, 0) > _SETAMOU(cBackup) ?]?:.voorwaarde om uit loop te breken
         _BEGI _BRK _END
   _LOOPEND
      iIndex 10: _SETEXP(cBackup, 10, _SETSTRI)._SUBS(1, -1):. geeft lege string
      iIndex 15: _SETEXP(cBackup, 15, _SETSTRI)._SUBS(1, -1):. geeft geen foutmelding
      iIndex  0: _SETEXP(cBackup, 0, _SETSTRI)._SUBS(1, -1):.  geeft de laatste string
      Aantal strings: _SETAMOU(cBackup):.                      geeft aantal strings
   _SETDEL(cAllen)
   _SETDEL(cBackup)
   _VARDEL(iTeller)
_FUNCEND
```

Verantwoording

Door zelf een beknopte handleiding te schrijven, kom je achter de mogelijkheden en onmogelijkheden, de notatie van de coderegels, het gebruik van strings en variabelen, enzovoorts van de Aldfaer script- of sjabloontaal.

Bij het schrijven heb ik me beperkt tot de logische vergelijkingen, de operators en operanden, de personen- en stringset.

Ook heb ik enkele korte stukjes code geschreven om zaken voor mezelf duidelijk te krijgen én om de scripttaal beter te leren begrijpen. De sjabloontjes heb ik allemaal in Aldfaer 11.1 gedraaid.

De richtlijn bij het schrijven was de Aldfaer-help *(Help 11.1 13-11-2024)*. Sommige delen heb ik gewoon overgenomen en andere stukken heb ik hier en daar aangepast, aangevuld of gepoogd opnieuw te schrijven, voor zover ik begreep hoe de vork in de steel zit.

De Aldfaer-help is in Aldfaer via `menu` → `Help` → `Versiebeheer Aldfaer` te downloaden.

* Aldfaer is een stamboomprogramma van © Stichting Aldfaer

Hulpprogramma's

AldKleur is een door mij geschreven hulpprogramma om in Aldfaer de kleuren van de user interface aan te passen aan de wensen van de gebruiker, wat in Aldfaer tamelijk onoverzichtelijk gaat, zie: AldKleur

AldMat is een door mij geschreven hulpprogramma om in Aldfaer makkelijk niet gekoppelde bestanden, zoals plaatjes en documenten, te vinden en te koppelen en andere manipulaties met de materiaalbestanden, zie AldMat

AldTijd is een door mij geschreven hulpprogramma om in Aldfaer sommige ten onrechte gewijzigde wijzigingsdatums te corrigeren, zodat de wijzigingsdatum weer overeenkomt met de werkelijke veranderingen in de stamboom, zie: AldTijd

Enkele begrippen

tag volgens de Aldfaer-help:

„Programmacode
Om de Aldfaer data base te benaderen zijn een aantal codes ontwikkeld, de zogenaamde _TAGs. Als deze code in een sjabloon wordt opgenomen dan zal de _TAG gevuld worden met de inhoud van het data veld. De schrijfwijze is uitsluitend in hoofdletters voorafgegaan door een underscore. In Notepad++ heeft iedere Tag een herkenbare eigen groepskleur zoals bij het onderstaande overzicht getoond wordt.
Naast deze _TAGs kan ook Javascript en html gebruikt worden."

Zoals bekend is de personenset het uitgangspunt van Aldfaer. Een dergelijke set is een verzameling personen met een bepaald aantal eigenschappen, bijvoorbeeld alle overleden mannen met twee of meer partners.

Om die personen in een dergelijke set te verzamelen, zijn er de zgn. tags ontwikkeld, in dit geval zijn nodig; **`_RELA`** *test op het aanwezig zijn van relatie(s)*, **`_MALE`** *test/select het geslacht van de persoon (mannelijk)* en **`_DEAT()`** *test op overleden*. Hier moet het mee lukken om een personenset te vullen en om een rapport te produceren.
En in dat rapport kunnen de persoonlijke details worden opgehaald met bijvoorbeeld **`_BIRTDATE`** *geeft de geboortedatum*, **`_BURIPLAC`** *geeft de plaats van begraven/ter beschikking gesteld/vermist* enzovoorts.

Tussen de tags **`_WRITBEGI`** en **`_WRITEND`** wordt het rapport of stukken daarvan geschreven en als platte tekst, met de extensie die de gebruiker bepaalt: *rapport_janssen.html* of *overige-bijlagen.txt*, als een computerbestand weggeschreven. Zie een voorbeeld op pagina 9.

Platte tekst is opmaakloze tekst dat met een teksteditor gelezen en/of gewijzigd kan worden, ook de rapporten die als Excel-rapporten worden aangeduid. Het is in wezen platte tekst met een indeling als gebruikelijk bij een csv bestand.

Merk op - in het Wikipedia lemma csv bestand - dat het laatste veld niet wordt afgesloten met de separator, zoals dit wel met `_SETFILL` gebeurt, wat tot een lege laatste kolom resulteert bij het inlezen in Excel.

keyword volgens ‚internet'

Sleutelwoorden of keywords zijn speciale gereserveerde woorden die een specifieke betekenis en doel hebben en alleen voor die specifieke doeleinden kunnen worden gebruikt.
Enkele voorbeelden van **tags** die ik als **keyword** zou willen bestempelen:

```
_BEGI _ELSE _END
_RETURN
[::] [&&] | [??]
```

Sam Francke, 16-08-2025

e-mail: samfrancke@hccnet.nl