



Practicum Programmeerprincipes 2006-2007

cderoove@vub.ac.be

REEKS 4

WERKEN MET TABELLEN IN PICO

Bestudeer het volgende transcript om Pico's beschikbare tabelbewerkingen op te frissen. Merk op dat n-dimensionale tabellen in Pico voorgesteld worden als tabellen waarvan de elementen opnieuw tabellen zijn. Een 3x4-matrix is bijvoorbeeld een 2-dimensionale tabel.

```
> rij_1 : [1,1,1,1]
<table>
> rij_2[4] : 2
<table>
> rij_3[4] : 3
<table>
> matrix : [rij_1, rij_2, rij_3]
<table>
> display(matrix)
[[1, 1, 1, 1], [2, 2, 2, 2], [3, 3, 3, 3]]
> size(matrix)
3
> display(matrix[1])
[1, 1, 1, 1]
> size(matrix[1])
4
> matrix[1,4]
1
> i:0
0
> m[3,4]: i:=i+1
<table>
> display(m)
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
> size(m)
3
> size(m[2])
4
> t : m[2]
<table>
> display(t)
[5, 6, 7, 8]
> m[2,1]
5
> m[2,1]:=10
<table>
> display(m)
[[1, 2, 3, 4], [10, 6, 7, 8], [9, 10, 11, 12]]
> m[2,1]
10
> t
<table>
> display(t)
[10, 6, 7, 8]
>
```

Bewerkingen op tabellen

Oefening 1. Schrijf een functie `numbers_i(b, e)` die een tabel teruggeeft met alle getallen in `[b,e]` en een functie `numbers_e(b, e)` die een tabel teruggeeft met alle getallen in `]b,e[`.

Oefening 2. Schrijf een functie `member(element, tabel)` die `true` teruggeeft wanneer `element` in een 1-dimensionale `tabel` voorkomt.

Oefening 3. Schrijf een functie `alleenklinkers(tekenreeks)` die voor een gegeven `tekenreeks` een nieuwe tekenreeks met alleen de klinkers uit de originele tekenreeks teruggeeft. Maak hiervoor gebruik van de `member`-functie uit de vorige oefening. (HINT: definieer een tabel met alle klinkers uit het alfabet en kijk ook nog eens naar oefening 2.3).

Oefening 4. Onderstaand transcript demonstreert n-dimensionale tabellen in Pico.

```
> m[2,3,4]:0
<table>
> display(m)
[[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]], [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]]
> t : m[1]
<table>
> display(t)
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
> z : t[1]
<table>
> display(z)
[0, 0, 0, 0]
> z[1]
0
> m[1,1,1]
0
> size(m)
2
> size(t)
3
> size(z)
4
>
```

Bestudeer het transcript nauwkeurig en voorspel vervolgens de waarde van de volgende Pico-expressie:

```
{ i : 0; m[3,1,2,3]: i:=i+1 }.
```

Deze expressie bindt in het woordenboek een tabel van grootte ... aan de naam `m`. De ... elementen van `m` zijn zelf tabellen van grootte ... waarvan elk element opnieuw een tabel is van grootte Deze laatste tabellen bestaan uit ... elementen. Na de evaluatie van de expressie is `i` gebonden aan de waarde De tabel gebonden aan de naam `m` ziet er dus als volgt uit:

Oefening 5. Schrijf een functie die de gemiddelde lengte van alle woorden in een zin teruggeeft. Hint: je kan gebruik maken van de ingebouwde functie `explode(tekst)` en de gemiddelde lengte van alle woorden berekenen uitgaande van het aantal spaties in de zin.

Oefening 6. Schrijf een functie `palindroom(woord)` die bepaalt of een tekenreeks een palindroom is. Schrijf hiervoor eerst een functie `omgekeerde(woord)` die het omgekeerde van een tekenreeks teruggeeft.

Oefening 7. Schrijf een functie `copydouble(tabel)` die een tabel als argument neemt en een nieuwe tabel teruggeeft waarbij elk element het dubbel is van het overeenkomstige element in de originele tabel. De originele tabel mag hierbij niet aangepast worden:

```
<function copydouble>
> t : [1,2,3,4]
<table>
> copy : copydouble(t)
<table>
> display(copy)
[2, 4, 6, 8]
> display(t)
[1, 2, 3, 4]
> |
```

Oefening 8. Schrijf een functie `map(t, f)` die op elk element van de tabel `t` de functie `f` toepast en de resultaten van deze functie-toepassing in een nieuwe tabel teruggeeft.

Definieer nu de functie `map(t, f(x))` met de functionele parameter `f(x)` en druk daarmee de functie `copydouble(tabel)` uit.

Oefening 9. Schrijf een functie `destructivedouble(tabel)` die de elementen van de originele tabel destructief vermenigvuldigt met twee:

```
<function destructivedouble>
> t : [1,2,3]
<table>
> result : destructivedouble(t)
<table>
> display(result)
[2, 4, 6]
> display(t)
[2, 4, 6]
> |
```

Oefening 10. Schrijf een functie `is_sorted(t)` die bepaalt of een tabel gesorteerd is volgens `<`.

Oefening 11. Schrijf een functie `somtabellen(t1, t2)` die 2 tabellen optelt. Indien de tabellen niet van gelijke lengte zijn, kan je 0 als neutraal element gebruiken.

```

<function somtabellen>
> t : somtabellen([1,2,3],[4,5,6,7])
<table>
> display(t)

```

[5, 7, 9, 7]

```

>

```

Oefening 12. Doe hetzelfde voor matrices van dezelfde grootte.

Funcies met een staartje

Oefening 13. Schrijf een functie `accumulate@args` die een variabel aantal argumenten neemt en deze optelt. Test deze functie als volgt: `accumulate(1,2,3,4)`.

De `apestaart` wordt in Pico niet alleen gebruikt voor het definiëren van functies met een variabel aantal argumenten. Het is ook mogelijk deze te gebruiken om een functie met een variabel aantal argumenten toe te passen op een tabel. Je kan de functie `accumulate` dus ook als volgt testen: `accumulate@[1,2,3,4]`. Op deze manier is het mogelijk om dit soort functies toe te passen op expressies waarvan het aantal niet op voorhand gekend is.

Oefening 14. Schrijf een functie `sumbetween(b, e)` die gebruik maakt van de functie `numbers_i(b, e)` en `accumulate@args` om de som van alle getallen in `[b,e]` te berekenen.