Ohjelmointia Maplella

- Maplessa on useita valmiita *rakenteisia tietotyyppejä*, esimerkiksi taulukko, lista ja joukko
- Maplesta löytyvät myös tavalliset ohjausrakenteet, kuten if, for ja while
- Maplessa työarkilla komennot jaetaan *suoritusryhmiin (execution group)*: Kun suoritusryhmän sisällä painetaan Enter-nappulaa, ryhmän komennot suoritetaan
- Suoritusryhmän sisällä seuraavalle riville siirrytään painamalla *shift+enter*

Rakenteiset tietotyypit

- *Jono (sequence)* muodostetaan luettelemalla peräkkäin lausekkeita pilkulla erotettuna, esimerkiksi `jono:=Int(x,x=0..1),2,Maple,diff(x^2,x);`
- Jonoja voidaan muodostaa myös komennolla `seq`, esimerkiksi 10 ensimmäistä alkulukua saadaan komennolla `seq(ithprime(i),i=1..10)`, jonka tulos on jono `2, 3, 5, 7, 11, 13, 17, 19, 23, 29`
- Lisäksi Maplessa on jono-operaattori $, joka toistaa tiettyä lauseketta useita kertoja, esimerkiksi `x$4` tuottaa jonon `x,x,x,x`
- Jonon indeksiä n vastaavaan alkioon voidaan viitata hakasulkujen avulla, esimerkiksi `jono:=seq(ithprime(i),i=1..10); jono[4];` antaa tuloksena luvun 7
- *Joukko* määritellään Maplessa antamalla jono alkioita kaarisulkujen { ja } sisällä, esimerkiksi `joukko:={1,2,3,4,5}`
- Joukossa kukin alkio esiintyy vain kerran, siis esimerkiksi joukot `J1:={1,2,3,4}` ja `J2:={1,1,2,2,3,3,4,4}` ovat samoja
- Joukon alkioita ei ole järjestetty, joukosta voidaan kuitenkin poimia alkioita hakasulkujen avulla, esim `J1[3]` antaa tuloksena luvun 3,
- Alkioiden lukumäärä saadaan komennolla `nops`, esimerkiksi `nops(J1)` on 4. Joukon määrittelevä jono saadaan komennolla `op`, esimerkiksi `op(J1)` tuottaa jonon `1,2,3,4`
- Joukko-operaatioita Maplesta löytyy `union`, `minus` ja `intersect`, esimerkiksi `A:={3,2,1},B:={3,4,5}`

  - `A union B`; tulos `{1,2,3,4,5}`
  - `A minus B`; tulos `{1,2}`
  - `A intersect B`; tulos `{3}`

- *Lista* määritellään Maplessa antamalla jono alkioita hakasulkujen [ja] sisällä, esimerkiksi `lista:=[2,1,5,3,4]`
- Listassa sama alkio voi esiintyä useamman kerran
- Alkioiden järjestys säilyy listassa, listasta poimiminen tapahtuu hakasulkujen avulla, esimerkiksi komennon `lista[3]` tulos on 5
- Listasta voidaan poimia tiettyä indeksiväliä vastaavat alkiot, esimerkiksi `lista[2..4]` palauttaa listan 2,3 ja 4 alkion, siis listan `[1,5,3]`
- Listoilla toimivat komennot `op` ja `nops` samalla tavoin kuin joukoilla

Maple-aliohjelmat

- Nykyaikaisten ohjelmointikielien tapaan myös Maplessa voi kirjoittaa aliohjelmia (procedure), uusia komentoja
- Aliohjelma kirjoitetaan *yhden* suoritusryhmän sisään
- Maple-aliohjelma aloitetaan sanalla `proc`

```
proc (argumentit)
    local L;
    global G;
    options O;
    description D;

    komennot
end;
```

- Huomaa, että Maple ei automaattisesti sisennä komentoja
- **Esimerkki** Maple-aliohjelmasta

```
f:=proc (x,y)
   x^2+y^2;
end;
```

Aliohjelmalle annetaan nimeksi `f` sijoituslausekkeella `:=`, ohjelmalla on kaksi argumenttia `x` ja `y`, ja se laskee funktion $f(x, y) = x^2 + y^2$ arvon.

- Aliohjelmaa kutsutaan kuten aiemminkin funktioiden yhteydessä, esimerkiksi `f(1,2)` = 5 ja `f(t,sin(r))` = $t^2 + \sin(r)^2$
- Jos aliohjelman runko koostuu useasta lausekkeesta, palautetaan aliohjelman arvon viimeisen lausekkeen arvon, palautettava arvo voidaan määritellä myös `RETURN`-komennon avulla
- **Esimerkki**: Aliohjelma

```
f:=proc (x,y)
   x^2+y^2;
   sin(x)+cos(y);
end;
```

laskee funktion $f(x, y) = \sin(x) + \cos(y)$ arvon, kun taas aliohjelma

```
f:=proc (x,y)
   RETURN(x^2+y^2);
   sin(x)+cos(y);
end;
```

palauttaa funktion $f(x, y) = x^2 + y^2$ arvon

Maplen ohjausrakenteet

if-lause

Yleinen lauseke	**Esimerkki**

`if expression then`
`  statements`
`elif expression then`
`  statements`
`else`
`  statements`
`fi;`
	`if a < 0 then`
	`  -1`
	`elif a=0 then`
	`  0`
	`else`
	`  1;`
	`end`

Saman voi kirjoittaa myös yhdelle riville: `if a<0 then -1; elif a=0 then 0; else 1; fi;`

while-lause

Yleinen lauseke	**Esimerkki**

`while expression do`
`  statements`
`od;`
	`while x > 1 do`
	`  x:=x/2;`
	`od;`

Yhdelle riville kirjoitettuna: `while x > 1 do; x:=x/2; od;`

for-lause

Yleinen lauseke

`FOR name from expr by expr to expr do`
`  statements`
`od;`

Esimerkki

`for i from 1 to 10 do`
`  i^2;`
`od;`