Perusasioita

2. helmikuuta 2005

Yleistä filosofiaa ja erityisesti Numsym05-kurssin tarpeita palvellee parhaiten, jos esitän asian rinnakkain MAPLE:n ja MATLAB:n kannalta.

Ohjelmien kirjoittaminen merkitsee systeemin laajentamista omilla funktioilla, jotka periaatteessa toimivat aivan kuten systeemin (MAPLE,MATLAB, ym.) omat funktiot.

Sekä MAPLE että MATLAB sisältävät suuren määrän kirjastofunktioita, joiden koodi on avointa. Hyvä tapa saada ohjelmointioppia on tutkia ohjelman valmistajan omia koodeja. Sitä ennen kannattaa harjoitella lukemaan ja kirjoitamaan helppoja ja riisuttuja muutaman rivin funktioita.

Tässä samaistamme termit "ohjelma", "proseduuri" ja "funktio". Sekä MAPLE että MATLAB ovat *funktionaalisia* ohjelmointikieliä, eli yleensä ohjelmat "ovat funktioita".

No, asiaan:

Maple

Olemme jo harrastaneet ohjelmointia kirjoittaessamme funktiomäärityksiä.

Tyyppiä `f:=x->x^2` olevat määritykset ovat jo tuttuja.

```
> type(f,procedure);
                                    true
```

Toden totta, kirjoitimme jo ohjelman.

Vastaavasti voidaan määritellä useamman muuttujan funktio vaikkapa $f(x,y,z) = x^2 + y^2 + z^2$ näin

```
> F:=(x,y,z)->x^2+y^2+z^2;
```

Jos halutaan lokaaleja muuttujia ja yleensä laajempaa koodimäärää, kirjoitetaan määrittely käyttäen `proc`-syntaksia.

Edellä oleva F voitaisiin määritellä tätä pitempää syntaksia käyttäen näin:

```
> F:=proc(x,y,z)
  x^2+y^2+z^2
  end:
```

Tällaisessa `proc`-määrittelyssä palautetaan tuloksena viimeiseksi suoritetun komennon tulos (ellei `RETURN`-lauseella toisin määrätä).

Käytännön neuvo*:* Kun kirjoitat koodirivejä MAPLE- työarkille, käytä `SHIFT-ENTER`- yhdistelmää rivinvaihdoissa!

Proseduuri on MAPLE-objekti siinä, missä yksinkertainen muuttuja, lista, matriisi jne. Se siis sijoitetaan arvoksi jollekin symbolille, jos sitä halutaan nimellä kutsua.

Proseduurin yleinen muoto on tällainen

```
proc(argjono)
   local muuttujajono1;
   global muuttujajono2;
   options muuttujajono3;
```

```
      komento_1;
      komento_2;
      ...
      komento_n
   end:
```

Sanoilla `local, global` ja `options` alkavat rivit eivät ole pakollisia.

Viimeiseksi laskettu tulos on se, jonka **proseduuri palauttaa**, ellei `RETURN`- lauseella toisin määrätä. Kaikki lauseet päätetään puolipisteeseen, paitsi viimeinen, `end`-sanaa edeltävä (siihenkin saa laittaa puolipisteen). Loppumerkkinä on normaaliin tapaan puolipiste tai kaksoispiste.

Matlab

Sama MATLAB:llä:

1) ns. inline-määrittely tehdään MATLAB-istunnossa:

```
>> F=inline('x^2+y^2+z^2','x','y','z')

F =

     Inline function:
     F(x,y,z) = x^2+y^2+z^2
>> F(1,2,3)

ans =

    14
```

MATLAB:ssa MAPLE:n proc:ia vastaava funktiomääritys kirjoitetaan suoraan teksititedostoon, ei siis MATLAB-istuntoon. Tiedosto on tyyppiä `.m` ja sen tulee sijaita joko oletushakemistossa (jonne voidaan siirtyä MATLAB:n sisällä -komennolla `cd`). Yleispätevämpi tapa on käyttää `addpath`-komentoa, jolla MATLAB:n hakupolkuun lisätään kyseinen hakemisto.

Tehdäänpä tuo yllä sanottu nyt tällä tyylillä: Tiedostoon `F.m` kirjoitetaan jollain tekstieditorilla (luontevimmin MATLAB:n omalla editorilla) rivit:

```
function w=F(x,y,z)
w=x^2+y^2+z^2;
```

Tässä on vastaava MATLAB-istunto:

```
>> pwd

ans =

/mnt/kiekko/home/apiola

>> addpath opetus/materiaali03/maple
>> F(1,2,3)

ans =

    14
```

Peruserot:

- MAPLE:n proc:ssa ei tarvitse sijoittaa funktion palauttamaa arvoa millekään muuttujalle. (Haluttaessa sen voi tehdä, jolloin kyseinen muuttuja kirjoitetaan `local`-sanan perään paikallisten muuttujien jonoon.
- MATLAB:n funktiomääritys alkaa aina varatulla sanalla `function`, jonka jälkeen kirjoitetaan otsikkoriville paluumuuttujan nimi (esimerkissa `w`. Se on vapaasti valittavissa (ei tietenkään saa olla sama kuin jokin argumenttilistan jäsen). Viimeiseksi tälle muuttujalle sijoitettu arvo on se, jonka funktio palauttaa.

Vastaavasti kuin MATLAB:ssä, MAPLE-proseduurit kannattaa kirjoittaa tekstitiedostoon, kun niitä alkaa kertyä. Niiden hakeminen eri työarkeilta käy pian tuskalliseksi. Toisin kuin MATLAB:ssä, MAPLE-funktiot kannattaa koota samaan tekstitiedostoon, joka luetaan MAPLE:een tarvittaessa. Esim: tiedosto `omatohjelmat.mpl`

```
# Tiedosto omatohjelmat.m  1.2.2005
# Tänne alan kerätä omia hengentuotteitani.
# Aloitetaan vaikka tällä:

minmax:=proc(lista)
#Haetaan listan pienin ja suurin alkio, palautetaan
# tulos listana muodossa [m,M]
local jono;
jono:=op(lista);
[min(jono),max(jono)];
end:
```

Maple-istunto

Luetaan nyt äskeinen tiedosto MAPLE:en ja kokeillaan siinä olevan (toistaiseksi ainoan) funktion toimintaa.

```
> read("/home/apiola/opetus/materiaali03/maple/omatohjelmat.mpl");
> print(minmax);

proc(lista)
local jono;
    jono := op(lista); [min(jono), max(jono)]
end proc

> minmax([seq(i,i=-10..10)]);

                          [-10, 10]
> minmax([seq(sin(i),i=-10..10)]);

                          [-sin(8), sin(8)]
```

1. OHJAUSLAUSEITA

Tulkkaavassa järjestelmässä, kuten molemmat MAPLE ja MATLAB voidaan käyttää ohjauslauseita suoraan istunnossa, tarvitsematta aina kirjoittaa funktioita.

`for do end do`- rakenteen yleinen muoto on

```
|for <name>| |from <expr>| |by <expr>| |to <expr>| |while <expr>|
                        do <statement sequence> end do;
tai

  |for <name>| |in <expr>| |while <expr>|
                        do <statement sequence> end do;
```

Näissä kunkin pystyviivaparin rajaama osa on valinnainen. Esimerkiksi

```
for i to 10 do a[i]:=x^i end do;
```

tarkoittaa, että käytetään oletusarvoja `from 1 by 1` ja jätetään `while`-osa pois.

Huomaa `do <statement sequence> end do;`, missä voidaan ajatella, että `do` on alkusulku ja `end do` loppusulku. Vastaava pari on `if` — `end if` .

Esim. *Tsebyshevin polynomit* voidaan määritellä rekursiokaavalla

$$T_0(x) = 1, T_1(x) = x, T_n = 2xT_{n-1}(x) - T_{n-2}(x)$$

Tehtävänä on määritellä proseduuri näiden laskemiseksi. Huomautamme aluksi, että laskenta voidaan tehdä suoraan interaktiivisesti tähän tapaan:

```
> t0:=1;t1:=x;n:=5:
> for i from 2 to n do
  tn:=expand(2*x*t1-t0);t0:=t1;t1:=tn;end do;
> i;  # katsotaan i:n arvo for-silmukan jälkeen.
```

Interaktiivisessa käytössä `for - do - end do` -rakenteella on indeksimuuttujaan liittyvä sivuvaikutus, se jää arvoon *yläraja* + 1

Proseduuriksi kirjoitettuna toteutus voisi olla tällainen.

```
T:=proc(n::integer,x)
local t0,t1,tn,i;
t0:=1;t1:=x;
for i from 2 to n do
  tn:=expand(2*x*t1-t0);
  t0:=t1;t1:=tn;
end do;
tn;  # Tämä palautetaan tuloksena.
end:
```

Nyt voidaan kutsua vaikkapa `T(5,x);` Lisäsimme parametrille n tyyppitarkistuksen. Siten esimerkiksi kutsu `T(x,5);` tai `T(a,10);` antaa asianmukaisen virheilmoituksen.

Koska silmukkaindeksi `i` on määritelty lokaaliksi, proseduurin kutsu ei muuta sen arvoa. Tätä voi kokeilla vaikka näin `> i:='i':T(7,z);i;`

Sama MATLAB**:lla**

MATLAB:llä ei voida tehdä symbolista. Tehdään nyt oikean tyylinen toteutus. Alkukommenteissa näkyy funktion toiminta. MATLAB on siitä(kin) hyvä, että alkukommentit tulevat näkyvin tässä tapauksessa komennolla `help tseb`.

Tyypillisesti MATLAB-funktio rakennetaan toimimaan vektoriargumentilla. Kun lasketaan useiden polynomian arvoja annetun x-vektorin pisteissä, on luonnollista palauttaa tulos matriisina, jonka sarakkeen pituus on x ja kunkin sarakkeen arvot ovat ao. asteisen polynoin arvoja x-vektorin pisteissä. Tässä muodossa tulos voidaan samalla antaa heti `plot`:lle.

```
function Y=tseb(x,n)
% tseb  Tsebychevin poynomit
%   Y=tseb(X,n) laskee n ensimmäistä Tseb-polynomia
%   vektorin x pisteissä.
%   Tulosmatriisin Y k:s sarake antaa astetta k-1 olevan Teseb-
%   polynomin arvot.

Y=ones(length(x),n); % Ykkösmatriisi, n saraketta.
x=x(:);              % Varmsietaan, että x on sarakevektori.
  if n==1, return, end  % Pelkkä 1. Tseb. pol =1.

Y(:,2)=x;
for k=3:n
  Y(:,k)=2*x.*Y(:,k-1)-Y(:,k-2);
end
```

Tähän loppui funktion koodi. Huomaa, että `end` ei ole funktion loppusana, vaan se kuuluu `for`-silmukkaan. (Funktiolla ei ole mitään "lopputervehdystä".)

Nyt sitten innolla kustumaan:

```
>> help tseb

  tseb  Tsebychevin poynomit
    Y=tseb(X,n) laskee n ensimmäistä Tseb-polynomia
    vektorin x pisteissä.
    Tulosmatriisin Y k:s sarake antaa astetta k-1 olevan Teseb-
    polynomin arvot.

>> x=linspace(-1,1);
>> Y=tseb(x,4);
>> plot(x,Y)
```

Matriisin Y (100×4) sarakkeina ovat nyt asteita $0, 1, 2, 3$ olevien Tseb-polynomien arvot x-vektorin pisteissä.

Sekä MAPLE- että MATLAB- ratkaisut ovat omalla tavallaan tyylikkäitä. MAPLE:lla voidaan suorittaa symbolisesti polynomien käsittelyä. MATLAB:llä taas saadaan numeeriset arvot tehokkaasti käyttöön, ja numeerinen jatkokäsittely on äärimmäisen helppoa ja tehokasta.