

16 januari 2012

DN1212 NumProg för T1, VT2012

Föreläsning 6: Funktioner

Kap 5 i PEng

Idag:

Varför ska man dela upp program i funktioner?

Egna funktioner

Inparametrar

Utparametrar

Åtkomstregler

Hur fungerar ett anrop

Funktionsparametrar

Antal parametrar: nargin och nargout

Reklam

Enklare testning

Koden går att återanvända

Begränsar sidoeffekter

Exempel: bibliotek

```
% bibliotek.m
% skrivet av linda 050126
% Biblioteksprogram med funktionsanrop
lasInBocker()
fortsatt=1;
while fortsatt
    visaMeny()
    val = input('Vad vill du göra? ', 's');
    switch(val)
        case {'L', 'Låna', 'låna'}
            lana()
        case {'R', 'Returnera', 'returnera'}
            returnera()
        case {'A', 'Avsluta', 'avsluta'}
            fortsatt=0;
            avsluta()
        otherwise
            disp('Felaktigt val, försök igen.')
    end
end
```

UPPGIFT: Vilka funktioner kan man tänka sig behövs i t ex ett program för bokning av teaterbiljetter, en DVD-spelare eller ett textredigeringsprogram?

Skriva egna funktioner

Din m-fil ska ha samma namn som funktionen

Filen inleds med funktionshuvud:

```
function [utparameter1, ...] = funktionsnamn(inparameter1,...)
```

Sen skriver man dokumentationskommentarer (för lookfor och help)

Därefter satserna som ska utföras vid anropet

Utparametrar ska ges värden

Kommer inparametrarna att vara tal, vektorer, matriser?

Exempel i filen krokfunktion.m:

```
function [f] = krokfunktion(x)
% Beräknar x*sin(x)
    f = x.*sin(x);
```

UPPGIFT: Skriv en funktion som beräknar funktionen $x^3 + \sin(x)$ och dess derivata.

Exempel på inparametrar vid anrop av Matlabfunktioner

Ett värde som inparameter: $y = \sin(0.7)$;

En variabls värde som inparameter: $y = \sin(x)$;

Ett uttryck som inparameter: $y = \sin(x.*x/2)$;

Två inparametrar till plot: $\text{plot}(x,y)$;

Tre inparametrar till plot: $\text{plot}(x,y,'r')$;

Exempel på utparametrar vid anrop av Matlabfunktioner

Ett värde som utparameter: $\text{maxvarde} = \text{max}(v)$;

Två värden som utparametrar: $[\text{maxvarde}, \text{maxindex}] = \text{max}(v)$;

Åtkomstregler

Parametrar är lokala dvs oberoende av variabler med samma namn utanför funktionen

Detsamma gäller övriga variabler inuti funktionen

Exempel:

```
function y = funk(x)
    z = 2*x;
    y = z+5;
```

Vi provkör funktionen och försöker sedan komma åt x:

```
funk(0.5)
ans =      6
>> x
??? Undefined function or variable 'x'.
```

Undantagen är globala och persistenta variabler

En variabel som deklarerats global är densamma både i funktioner och i omgivningen

En variabel som deklarerats persistent går bara att komma åt inifrån funktionen men behåller sitt värde till nästa anrop

Exempel:

```
function underlig
global summa;
persistent antal;
if isempty(antal)
    antal = 1
else
    antal = antal+1
end
summa = summa+antal
```

Nu utför vi följande satser:

```
>> clear functions, clear all
>> global summa
>> summa=0;
>> underlig
antal =      1
summa =      1
>> underlig
antal =      2
summa =      3
>> summa
summa =      3
>> antal
??? Undefined function or variable 'antal'.
>>
```

Hur fungerar ett anrop?

Anropsinparametrarnas värde kopieras till motsvarande inparameter.

Funktionens satser utförs. Funktionen har exekverat färdigt när sista satsen har utförts...

... eller när kommandot return har utförts

När funktionen exekverats färdigt kopieras utparametrarna till motsvarande anropsutparametrar.

Det är OK att anropa med färre utparametrar än i funktionsdeklarationen.

Exempel:

```
function [y,termAntal] = egenSin(x)
% egenSin beräknar sin(x) mha Taylorutveckling
%-----
% egenSin använder Taylorserie för att beräkna sin(x) = x - x^3/3! + x^5/5! - x^7/7! + ...
% Utparametrar: y = sin(x); termAntal = antal summerade termer i serien
%-----
yold = 0;
y = x;
potens = 1;
fakultet = 1;
tecken = 1;
termAntal = 1;
while abs(y-yold) > eps
    potens = potens + 2;
    fakultet = fakultet*(potens-1)*potens;
    tecken = -tecken;
    yold = y;
    y = y + tecken*(x^potens)/fakultet;
    termAntal = termAntal + 1;
end
```

UPPGIFT: Föreslå någon förbättring av funktionen egenSin(x)

Funktioner som parametrar

Matlabfunktionen QUAD beräknar integralen av en funktion

`Q = QUAD(FUN,A,B)`

Utparametern Q är integralens värde.

Inparametrar är funktionen FUN som ska integreras och integrationsgränserna A och B

FUN kan anges på tre(fyra) olika sätt:

Som en sträng: `svaret = quad('x.*sin(x)',0,3);`

Med namnet på en m-fil (filnamnet utan '.m'): `svaret = quad('krokfunktion',0,3);` där krokfunktion.m ser ut så här

```
function f = krokfunktion(x)
f = x.*sin(x);
```

Med en referens till en m-fil (function handle): `svaret = quad(@krokfunktion,0,3);` där krokfunktion.m ser ut som ovan.

Som en inlinefunktion: `kroki = inline('x.*sin(x)'); svaret = quad(kroki,0,3);`

Om man själv vill ha en funktionsparameter då?

Använd feval för att evaluera funktionen i önskade punkter.

Exempel:

```
function svar = blirIe(funk)
% Beräknar funk:s värde i e
e=exp(1);
svar = feval(funk,e);
```

UPPGIFT: Vet du något problem som man kan tänka sig att lösa med en funktion som tar en funktion som parameter?

Antal parametrar: nargin och nargs

Med nargin kan man undersöka hur många inparametrar funktionen anropats med

På samma sätt fungerar nargs för utparametrar

```
function k = kvot(n,t)
if nargin == 2
    k = n/t;
else
    disp('Ge nämnare och täljare som parametrar.')
end
```

Körexempel:

```
>> kvot()
Ge nämnare och täljare som parametrar.
>> kvot(1)
Ge nämnare och täljare som parametrar.
>> kvot(1,2)
ans =
    0.5000
```

Lokala funktioner

Om flera funktioner deklarerar i samma m-fil blir funktionerna efter den första lokala funktioner som bara kan anropas i samma m-fil. Det här kan användas för att dela upp en stor funktion i flera mindre, vilket förbättrar läsligheten.