F6: Filhantering (kap. 8)

Läsning från tangentbordet, skrivning på skärmen
Binär skrivning och läsning med save och load
Skrivning på och läsning från textfil med save och load
Användning av filreferenser, öppna, stänga, filslut
Binär skrivning och läsning med filreferens
Skrivning och läsning med formatstyrning
Direktfiler

Läsning, skrivning, tangentbord, skärm

Det finns många sätt att hämta och spara data.
In från tangentbordet:

- med tilldelningskommando
- `input(`*ledtext*`)`, `input(`*ledtext*`,'s')`

Ut på skärmen:

- kommando utan ; variabelns namn skrivs också ut
- `disp(`*uttryck*`)` inget variabelnamn
- `fprintf(`*formatsträng*`,`*uttryck*`,…)` hur uttryckens värden skrivs styrs av formatkoder m.m. i formatsträngen

Tabell på skärmen

Skriv en tabell med rätvinkliga trianglar med heltalssidor <= 100.

```
fprintf('\n    a    b    c\n');
for c=1:100
  c2 = c^2;
  for a=1:floor(sqrt(c2/2))
    a2 = a^2;
    b = round(sqrt(c2-a2));
    if a2+b^2==c2
      fprintf('%5d%5d%5d\n',a,b,c);
    end
  end
end
```


Tabell på skärmen, test

Resultatet är monterat i 3 kolumner.

```
    a    b    c       12   35   37       16   63   65
    3    4    5       15   36   39       25   60   65
    6    8   10       24   32   40       33   56   65
    5   12   13        9   40   41       39   52   65
    9   12   15       27   36   45       32   60   68
    8   15   17       14   48   50       42   56   70
   12   16   20       30   40   50       48   55   73
    7   24   25       24   45   51       24   70   74
   15   20   25       20   48   52       21   72   75
   10   24   26       28   45   53       45   60   75
   20   21   29       33   44   55       30   72   78
   18   24   30       40   42   58       48   64   80
   16   30   34       36   48   60       18   80   82
   21   28   35       11   60   61       13   84   85
```

```
36   77   85      39   80   89      65   72   97
40   75   85      54   72   90      28   96  100
51   68   85      35   84   91      60   80  100
60   63   87      57   76   95
```


save

Ut på fil:

- `save` [*variabellista*] alla variabler i arbetsplatsen (eller angivna) binärt till filen `matlab.mat`
- `save` *filnamn* [*variabellista*] binärt till filen *filnamn*`.mat`
- `save` *filnamn.suffix* [*variabellista*] binärt till filen *filnamn*.suffix
- `save -ascii` [`-double`] [`-tabs`] som text till filen `matlab.mat`
- `save` *filnamn* [*variabellista*] `-ascii` [`-double`] [`-tabs`] som text till filen *filnamn*`.mat`
- `save` *filnamn.suffix* [*variabellista*] `-ascii` [`-double`] [`-tabs`] som text till filen *filnamn.suffix* med 8 siffror, `-double` ger 16 siffror, `-tabs` ger tabulatorseparerad utskrift

load

In från fil:

- `load` [*variabellista*] alla variabler (eller angivna) från `matlab.mat` binärt
- `load -ascii` alla data från `matlab.mat` som text till variabeln `matlab`
- `load` *filnamn* [*variabellista*] alla variabler (eller angivna) från *filnamn*`.mat` binärt
- `load` *filnamn* `-ascii` alla data från *filnamn*`.mat` som text till variabeln filnamn
- `load` *filnamn.suffix* [*variabellista*] [`-ascii` | `-mat`] alla variabler (eller angivna) från *filnamn.suffix* värdena läses normalt binärt, med `-ascii` eller med *suffix* annat än `mat` läses alla data till variabeln *filnamn* (då används ej variabellistan) från filen som text

Load forts.

Med

```
>> type in.dat

2 3
4 5 6
6 7
```

får man

```
>> load in.dat
??? Error using ==> load
Number of columns on line 2 of ASCII file in.dat
must be the same as previous lines.
```

Om vi tar bort `" 6".`

```
>> load in.dat
>> in
in=
```

```
    2     3
    4     5
    6     7
```


Skillnaden mellan »binärt» och »text»?

Både datorns minne och filer innehåller följder av bitar, grupperade som bytes. Bitmönstren kan tolkas på många sätt. Med Unix-kommandot `od` kan man visa en fils innehåll på valfritt sätt. `–t u1` betyder bytevis decimalt och `–c` betyder som tecken.

```
>> od -t u1 -c in.dat
0000000    50  32  51  10  52  32  53  32
             2       3  \n   4       5
0000010    54  10  54  32  55
             6  \n   6       7
```

```
% Test av binär lagring
d_17 = 17;
i2_17 = int16(17);
i1_17 = int8(17);
s_17 = '17';
```



```
ofr = fopen('bintest','w');
fwrite(ofr,d_17,'double');
fwrite(ofr,i2_17,'int16');
fwrite(ofr,i1_17,'int8');
fwrite(ofr,s_17,'char');
fclose(ofr);
```

```
>> od -t u1 -c bintest
0000000    64  49   0   0   0   0   0   0
             @   1  \0  \0  \0  \0  \0  \0
0000010     0  17  17  49  55
           \0 021 021   1   7
```

Byte 1–8 innehåller 17 i flyttalsformat: (–1)^s*t*2^e.
Bit 1 är s, här 0.
Bit 2–12 är e+1 023, här e = 4.
Bit 13–64 är taldelen–1, här t = 1+1/16. Dvs. talet är 17.
Det finns detaljer som vi hoppar över.

Filreferenser

Användning av filreferens. Med en filreferens håller Matlab reda på filen, vad man får göra med den och var man jobbar i den.

- *filreferens* = `fopen(`*filnamn*`,`*tillstånd*`)`
  Tillstånd kan vara `'r'`, `'r+'`, `'w'`, `'w+'`, `'a'`, `'a+'`. Om öppningen misslyckas blir *filreferens* –1 och ett felmeddelande lagras i variabeln `msg`.
- `fclose(`*filreferens*`)` Ev. utskrift avslutas. Filen stängs. Om det går bra återsänds 0 annars –1.
- `feof(`*filreferens*`)` sant om inga data återstår

Binär skrivning och läsning med filreferens

Skrivning binärt med filreferens:

- `fwrite(`*filreferens*, *uttryck*, *precision*`)`
  där precision kan vara `'char'`, `'int16'`, `'float32'` m.m.

Läsning binärt med filreferens:

- [*variabel*,*antal*] = `fread(`*filreferens*, *storlek*, *precision*`)`
  antal är antalet lästa värden, *storlek* är ett tal, `Inf` eller [*m n*] och *precision* som ovan

Fileread

`fileread(`*filnamn*`)` ger filens innehåll som en sträng.

Skrivning och läsning med formatstyrning

Skrivning som text med filreferens:

- `fprintf(`*filreferens*,*formatsträng*,*uttryck*,…`)`
- `sprintf(`*formatsträng*, *uttryck*,…`)` returnerar en sträng

Läsning från textfil:

- [*variabel1*, *variabel2*,…,*variabeln*] = `textread(`*filnamn*,*formatsträng*,*radantal*`)` läser kolumnuppdelade data

Läsning från textfil med filreferens:

- *variabel* = `fscanf(`*filreferens*,*formatsträng*,*storlek*`)`
- [*variabel*,*antal*] = `fscanf(`*filreferens*,*formatsträng*,*storlek*`)`
  texten i filen tolkas som värden enligt *formatsträng*
- [*variabel*,*antal*] = `sscanf(`*sträng*,*formatsträng*,*storlek*`)` läser från sträng

- *rad* = `fgetl(`*filreferens*`)` texten fram till radslut, om filen står vid radslut återsänds -1
- *rad* = `fgets(`*filreferens*`)` texten fram till och med radslut, om filen står vid radslut återsänds -1

Med

```
>> type in.dat
2 3
4 5 6
6 7
8 9 10
```

får man

```
>> [a, b, c]=textread('in.dat');
[a';b';c']
ans =
     2     4     6     8
     3     5     7     9
     0     6     0    10
```

Med

```
>>type s.dat
ab c

def
```

ger

```
s = fscanf(fopen('s.dat'),'%c')
s =
ab c

def
```

och

```
abs(s)
ans =   97   98   32   99   10   10  100  101  102
```


Formatsträng

En formatsträng består av formatkoder, specialtecken och annan text. En formatkod har formen `%`[–|+|0][#.#]*k*. `%` och *k* måste sättas ut.

`-` vänsterställ
`+` sätt alltid ut tecken
`0` fyll ut med inledande nollor
`#.` totalt antal tecken
`.#` antal siffror/antal decimaler
`c` ett tecken
`d` decimalt heltal
`e` exponentform med e
`E` exponentform med E
`f` decimalbråk
`g` kortaste av e och f
`G` kortaste av E och f
`o` oktalt heltal

`s` teckensträng
`u` decimalt heltal utan tecken
`x` sedecimalt (basen 16) med gemena siffror
`X` sedecimalt (basen 16) med versala siffror

Specialtecken är `%%` (representerar ett procenttecken), `\'` (apostrof), `''` (apostrof), `\\` (inverterat snedstreck), `\f` (sidmatning), `\r` (vagnretur), `\b` (blanktecken), `\t` (tabulatortecken), `\n` (ny rad).

Uttrycken paras ihop med formatkoderna. När formatsträngen tar slut används den från början igen. När data tar slut används formatsträngen fram till nästa formatkod eller till sitt slut. Text och specialtecken skrivs ut resp. matchas mot tecknen i filen.

Läs, lagra och skriv en textfil.

```
fr = fopen('findgear.m','r')
fr = 3
t = fscanf(fr,'%c',Inf);
disp(t);
```



```
function [m,n,finalgear] = findgear(gear,min,max)
...
end
whos t
Name  Size   Bytes  Class
   t  1x865   1730  char array
```

Läs en nx3 numerisk matris från textfil. Fyll ut sista raden.

```
type t.dat
1 2 3 4
5 6 7 8
9 10
fr = fopen('t.dat','r')
fr = 3
t = fscanf(fr,'%d',Inf); size(t)
ans = [10 1]
% Fyll sista raden
```

```
t = reshape([t; zeros(mod(-numel(t),3))],3,[])'
% mod([-12 -11 -10],3) är [0 1 2]
t =  1 2 3
     4 5 6
     7 8 9
    10 0 0
```

Med `uigetfile` och `uisetfile` kan man öppna dialogrutor för att välja in- och utfil.

Man bör kontrollera om läsning och skrivning gick bra:

```
[felmed,felnr] = ferror(filreferens);
if felnr~=0
  disp(felmed);
end
```

Läs i boken och hjälpinformationen om filhantering.

Direktfiler

Programmet kan styra var i filen läsning/skrivning ska ske:

`frewind(`*filreferens*`)` sätter aktuell position till filbörjan

`ftell(`*filreferens*`)` ger aktuell position i byte från filbörjan

`fseek(`*filreferens*`,`*pos*`,`*origo*`)` sätter aktuell position till *pos* byte efter origo, *origo* –1 betyder filbörjan, 0 betyder aktuell position, 1 betyder filslut

Funktion som parameter (function functions)

I många tillämpningar har man en metod (function method) som man ska kunna använda med många olika funktioner.

1 Funktion med bestämt namn

Fil: `use1.m`

```
function use1
method1;
```

Fil: `f1.m`

```
function f1(x)
disp(['f1 x='
num2str(x)]);
```

Fil: `method1.m`

```
function method
for x=3:3:10
  f1(x);
end
```

```
>use1
f1 x=3
f1 x=6
f1 x=9
```


2a Uttryck som text

Fil: `use2a.m`

```
function use2a
f = 'disp([''f2 x=''
num2str(x)]);';
method2a(f);
```

Fil: `method2a.m`

```
function method2a(f)
for x=3:3:10
  % eval(f);
  f(x);
end
```

```
>use2a
f2 x=3
f2 x=6
f2 x=9
```

2b Funktionsanrop som text

Fil: use2b.m

```
function use2b
f = 'f2b';
method(f);
```

```
>use2b
f2b x=3
f2b x=6
f2b x=9
```

Fil: f2b.m

```
function f2b(x)
disp(['f2b x=' ...
num2str(x)]);
```

Fil: method2b.m

```
function method(f)
for x=3:3:10
  % feval(f,x);
  f(x);
end
```


3a Enkel funktionsreferens (function handle)

Fil: use3a.m

```
function use3a
f=@f3a;
method3a(f);
```

```
>use3a
f3a x=3
f3a x=6
f3a x=9
```

Fil: f3a.m

```
function f3a(x)
disp(['f3a x='
num2str(x)];);
```

Fil: method.m

```
function method3a(f)
for x=3:3:10
  % feval(f,x);
  f(x);
end
```


3b Funktionsreferens med parameter

Fil: use3b.m

```
function use3b
for y = [5 10]
  f = @(x)f3b(x,y);
  method3b(f); end
```

```
>use3b;
f3b x=3 y=5
f3b x=6 y=5
f3b x=9 y=5
f3b x=3 y=10
f3b x=6 y=10
f3b x=9 y=10
```

Fil: f3b.m

```
function f3b(x,y)
disp(['f3b x='
num2str(x) ' y='
num2str(y)]);
```

Fil: method3b.m

```
function method3b(f)
for x=3:3:10
  % feval(f,x);
  f(x);
end
```