

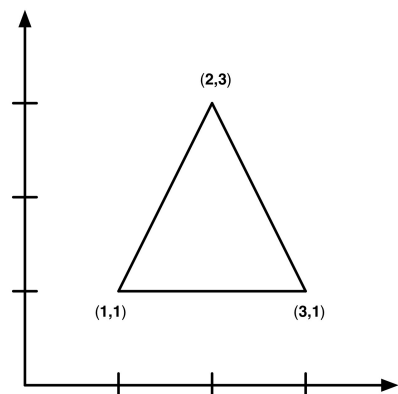
## TDT4110 Informasjonsteknologi grunnkurs:

### Eksempler

Amanuensis Terje Rydland  
Kontor: ITV-021 i IT-bygget vest (Gløshaugen)  
Epost: [terjery@idi.ntnu.no](mailto:terjery@idi.ntnu.no)  
Tlf: 735 91845

## Mangekanter

- Må finne en måte å **representere** mangekanter
  - Finne egnet datastruktur
- En liste med hjørnepunkter
  - En vektor med punkt-strukturer
  - En liste med x-koordinatene og en liste med y-koordinatene
  - En matrise med x- og y-verdiene i første og andre kolonne
  - Andre forslag?



## Tre alternativer

```
clear, clc

% Trekant representert med vektor med struct-er
mk_s(1) = struct('x', 1, 'y', 1);
mk_s(2) = struct('x', 2, 'y', 3);
mk_s(3) = struct('x', 3, 'y', 1);

% Trekant representert med 2 parallelle lister
xListe(1) = 1; yListe(1) = 1;
xListe(2) = 2; yListe(2) = 3;
xListe(3) = 3; yListe(3) = 1;

% Trekant representert med 2-dimensjonal tabell
mk_m(1,:) = [1,1];
mk_m(2,:) = [2,3];
mk_m(3,:) = [3,1];
```

## Om krets av ”struct-mangekant”

- Skriv kode for de andre alternativene selv.

```
function omkrets = mkOmkrets_struct(mk)

antallPunkter = length(mk);

omkrets = 0;
for p = 1:1:antallPunkter
    if p < antallPunkter
        kant = sqrt((mk(p).x - mk(p+1).x)^2 + ...
                    (mk(p).y - mk(p+1).y)^2);
    else
        kant = sqrt((mk(p).x - mk(1).x)^2 + ...
                    (mk(p).y - mk(1).y)^2);
    end % if
    omkrets = omkrets + kant;
end % for

end % function
```

## return og break

- Terminering av funksjoner og løkker (før tiden)
- return
- break

```
>> help return
```

```
return Return to invoking function.  
return causes a return to the invoking function or to the keyboard.  
It also terminates the KEYBOARD mode.
```

```
Normally functions return when the end of the function is reached.  
A return statement can be used to force an early return.
```

```
>> help break
```

```
break Terminate execution of WHILE or FOR loop.  
break terminates the execution of FOR and WHILE loops.  
In nested loops, break exits from the innermost loop only.
```

```
break is not defined outside of a FOR or WHILE loop.  
Use RETURN in this context instead.
```

## Problem: Søke etter del-tabell

- Dersom del-tabell finnes i tabell, returnere
  - true + rad og kolonne for ”øvre-venstre-hjørne”
- Ellers returneres
  - false + 0 for både rad og kolonne
- Funksjonssignatur:
  - function [finnes, rad, kol] = finnMonster(T, m)
- Problemer? Løsning?

7

```

>> T = randi(2,10,10) - 1;
>> T
T =
     1     1     0     1     1     0     1     0     0     1
     1     1     1     0     0     0     1     0     1     1
     0     1     0     0     0     0     1     1     0     0
     1     1     0     0     1     1     1     1     1     1
     0     1     1     0     0     1     1     1     1     1
     1     1     0     1     1     1     0     0     1     0
     1     1     0     0     1     1     0     1     0     0
     1     0     0     1     1     0     1     1     1     0
     0     1     0     1     0     1     0     1     0     0
     0     0     0     1     1     0     0     1     0     0

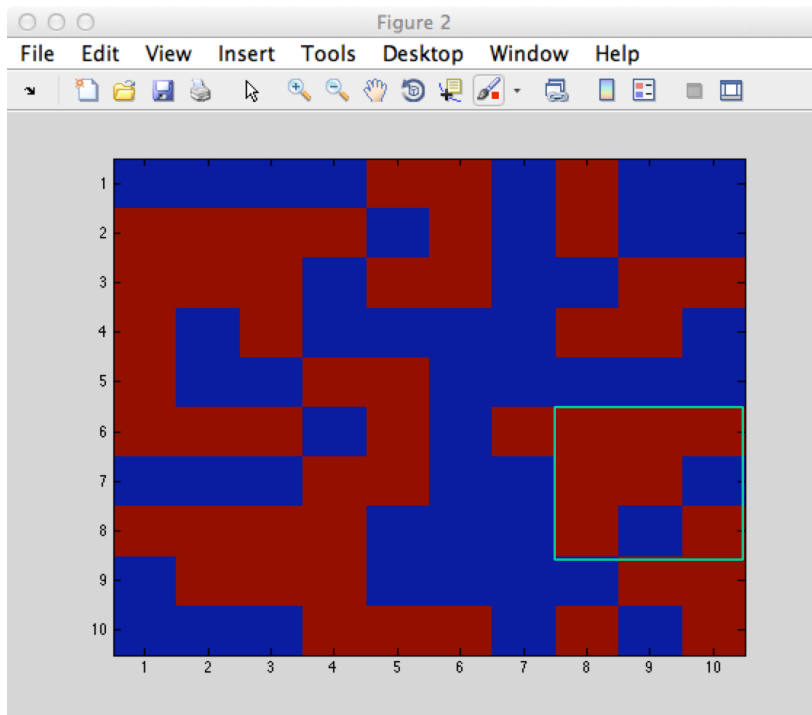
>> m = [1 1 1; 0 1 0; 0 1 0]
m =
     1     1     1
     0     1     0
     0     1     0

>> [finnes, rad, kol] = finnMonster(T, m)
finnes =
     1
rad =
     8
kol =
     7
>>

```

8

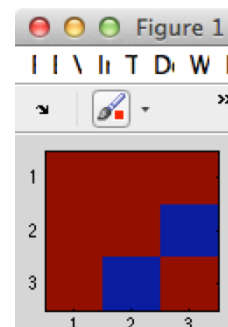
## Mønster-gjenkjenning



```

>> image(T*100)
>> image(m*100)
>>

```



## Pseudokode

Hvis testmønsteret er større enn målmønsteret avsluttes funksjonen

Start i øverste venstre hjørne

Fortsett til man har funnet en match, eller vi har sjekket hele mønsteret

Plukk ut en 3x3 matrise og sammenlign med målmatriksen

Hvis de er like

Marker hvor den ble funnet

Ferdig - Returner resultat

ellers

Flytt testmatrisen en posisjon

Returner at mønsteret ikke ble funnet

```
function [finnes, rad, kol] = finnMonster(T, m)
% Leter etter m i T, returnerer indeks til første forekomst

[TRader, TKolonner] = size(T);
[mRader, mKolonner] = size(m);

% sjekker om mønster er større enn tabellen
if (mRader > TRader) || (mKolonner > TKolonner)
    finnes = false;
    rad = 0;
    kol = 0;
    return
end % if

% leter etter mønster
for rad=1:1:(TRader-mRader+1)
    for kol=1:1:(TKolonner-mKolonner+1)

        if isequal(m, T(rad:(rad+mRader-1), kol:(kol+mKolonner-1)))
            finnes = true;
            return
        end % if

    end % for
end % for

% fant ikke mønster
finnes = false;
rad = 0;
kol = 0;

end %function
```

## return og break, forts

- Overstyrer den normale programflyten
- Gir av og til enklere kode som er lettere å forstå
- Bør brukes sparsomt
- NB! **break** avslutter bare den (innerste) løkken den står i



### TDT4110 Informasjonsteknologi grunnkurs:

#### Filbehandling

Amanuensis Terje Rydland  
Kontor: ITV-021 i IT-bygget vest (Gløshaugen)  
Epost: [terjery@idi.ntnu.no](mailto:terjery@idi.ntnu.no)  
Tlf: 735 91845

## Læringsmål og pensum

- Læringsmål
  - Datastrukturer:
    - Cell Arrays
    - Structures
  - Problemløsning og litt repetisjon
  - return og break – overstyring av normal programflyt
- Pensum
  - Kapittel 8 Data Structures

## Læringsmål/pensum

- Problemløsning/programmering
  - Finne ”mønster” i større tabell
  - Lagre mangekanter vha. struct
- Filbehandling
  - Mål:
    - Forstå forskjell på tekstlig og binær form
    - Forstå hvordan data organiseres i en fil
    - Kunne overføre data til og fra filer i Matlab
  - Pensum i Matlab-boka
    - 3.6 (2.6 gammel bok) «Introduction to File Input/Output»
    - 9 Advanced File Input and Output

## Filer: Motivasjon

- Kan ikke alltid gjenskape data eller regne ut ting på nytt
- Ofte behov for å lagre verdier som skal leses inn eller er beregnet i et program for senere bruk
- Data kan være av samfunnsmessig interesse (værvarsling) og påkrevet lagret for senere tilgang
- Mange beregninger på realistiske systemer lager store datamengder – må lagre for å kunne studere data med andre verktøy, f.eks. visualisering
- Overføre data mellom programmer

## Tekstlig og binær form

- Mindre datamengder kan med fordel lagres som (ascii) tekst så vi kan se hva verdiene er
- Dette kalles å lagre data på **tekstlig form**
  - Tallverdier lagres vanligvis som binært kodede tall
    - Hvert tall (double) bruker 8 bytes i minnet (RAM)
  - Ved lagring på tekstlig form oversetter Matlab fra binær form til en tekststreng
    - Med tegnene: '0', '1', ..., '9', '.', '+', '-', 'e'
    - Hvert tegn (enkelt-bokstav) bruker 2 bytes
- Lagring på **binær form**
  - Praktisk for datamaskiner, veldig upraktisk for mennesker!



## Binærfiler versus tekstfiler

- Ulemper med tekstfiler
  - Oversetting til/fra tekstlig form
  - Tar (vanligvis) større plass
- Fordeler med tekstfiler
  - Kan lese innholdet (så lenge filen er liten)
  - Standardisert, kan utveksle data mellom programmer
  - Kan skrive inn nytt eller endret innhold i en teksteditor
- Filer kan bli store, veldig store
  - Data fra værberegning kan være 4 GB (4 milliarder byte)
  - Effektivitet og raske oppslag er viktig
  - Lesbarhet (for mennesker) er irrelevant
- Vi skal se på behandling av tekstfiler

## Lagring av matriser

- Matlab gjør det enkelt å lagre matrisedata
  - Samme datatype
  - Regelmessig form (samme antall kolonner i alle rader)
- `save <filnavn> <variabel> -ascii`
  - Lager variabel i tekstfil med filnavn
- `save <filnavn> <variabel> -ascii -append`
  - Legger variabels innhold til i tekstfilen
- `load <filnavn>`
  - Henter data fra fil <filnavn> til variabel <filnavn>
- `<variabel> = load('<filnavn>')`
  - Henter inn data fra fil til oppgitt variabel
- `type <filnavn>`
  - Skriver ut innholdet i filen

## Eksempel 1

```
>> v = [1 2 3 4 5];
>> save 'testfill.txt' v -ascii;
>> type 'testfill.txt'

1.0000000e+00  2.0000000e+00  3.0000000e+00  4.0000000e+00  5.0000000e+00

>> save 'testfill.txt' v -ascii -append;
>> save 'testfill.txt' v -ascii -append;
>> save 'testfill.txt' v -ascii -append;
>> save 'testfill.txt' v -ascii -append;
>> type 'testfill.txt'

1.0000000e+00  2.0000000e+00  3.0000000e+00  4.0000000e+00  5.0000000e+00
1.0000000e+00  2.0000000e+00  3.0000000e+00  4.0000000e+00  5.0000000e+00
1.0000000e+00  2.0000000e+00  3.0000000e+00  4.0000000e+00  5.0000000e+00
1.0000000e+00  2.0000000e+00  3.0000000e+00  4.0000000e+00  5.0000000e+00
1.0000000e+00  2.0000000e+00  3.0000000e+00  4.0000000e+00  5.0000000e+00
```

## Eksempel 1 (forts.)

- save og load er den enkleste måten å bruke filer
- Forutsetter matrisedata, ikke alltid det passer

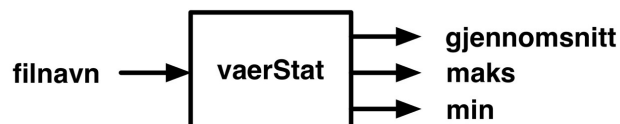
```
>> w = load('testfill.txt')

w =

     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
```

## Eksempel: Værdedata

- Middeltemperaturer fra Voll i september:
  - voll\_sept\_2011.txt
  - Hver linje: dag mnd år temp
- Ønsker funksjon som beregner:
  - Gjennomsnittstemperatur
  - Laveste temperatur
  - Høyeste temperatur



voll_sept_2011.txt			
1	9	2011	12.4
2	9	2011	13.1
3	9	2011	14.5
4	9	2011	16.8
5	9	2011	15.5
6	9	2011	14.7
7	9	2011	14.1
8	9	2011	11.2
9	9	2011	10.7
10	9	2011	9.5
11	9	2011	14.1
12	9	2011	14.3
13	9	2011	12.3
14	9	2011	10.6
15	9	2011	8.5
16	9	2011	8.1
17	9	2011	11.2
18	9	2011	10.4
19	9	2011	11.3
20	9	2011	9.5
21	9	2011	10.1
22	9	2011	8.5
23	9	2011	10.0
24	9	2011	10.3
25	9	2011	10.7
26	9	2011	10.7
27	9	2011	9.0
28	9	2011	11.4
29	9	2011	16.1
30	9	2011	14.1

## vaerStat.m

```

function [snittT minT maxT] = vaerStat(filnavn)
% Leser inn værdedata, finner snitt-, min- og maxtemp

vaerData = load(filnavn);

snittT = mean(vaerData(:,4));
minT = min(vaerData(:,4));
maxT = max(vaerData(:,4));

end % function
  
```

## Eksempelkjøring

```
>> [Te Mi Ma] = vaerStat('voll_sept_2011.txt')
Te =
    11.789999999999999
Mi =
     8.100000000000000
Ma =
    16.800000000000001
>>
```

## Lavnivå filbehandling

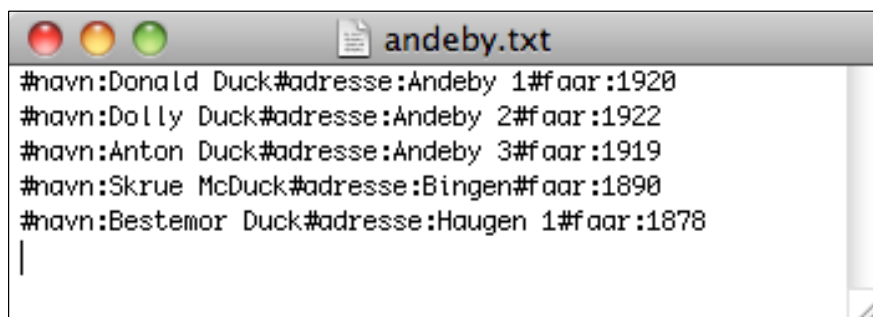
- Åpner filer (etablerer kobling til filen)
  - `<fil-id> = fopen('<filnavn>', '<tilgangstype>');`
  - `fil-id` er en filpeker (referanse til filen)
  - `fil-id` settes til -1 ved problemer
  - Tilgangstyper:
    - `r` – lese (fra starten)
    - `w` – skrive (sletter gammelt fil-innhold først)
    - `a` – legge til (skrive på slutten av filen)
- Lukker filen når vi er ferdig med den
  - `<status> = fclose(<fil-id>)`
  - Returnerer 0 når det går bra, -1 ellers

## Lesing/skriving

- `fgetl(<fil-id>)`
  - Leser inn en linje
  - Returnerer en tekststreng med linjens innhold
- `feof(<fil-id>)`
  - **usann** så lenge det er mer data igjen i filen (som ikke er lest)
  - **sann** når vi har kommet til slutten av filen
- `fprintf(<fil-id>, <formatstreng>, <verdier>)`
  - Skriver til filen med `<fil-id>`
  - Som til skjerm
- Mange flere (spesialiserte) muligheter, se læreboka og hjelp i Matlab.

## Eksempelfil: Personregister

- Data om en person på hver linje
- Feltindikatorer: `#navn: #adresse: #faar:`
- Leser inn en og en linje og plukker ut data



```
#navn:Donald Duck#adresse:Andeby 1#faar:1920
#navn:Dolly Duck#adresse:Andeby 2#faar:1922
#navn:Anton Duck#adresse:Andeby 3#faar:1919
#navn:Skrue McDuck#adresse:Bingen#faar:1890
#navn:Bestemor Duck#adresse:Haugen 1#faar:1878
|
```

## lesAndebyFil.m

```

clear, clc

% aapner filen
fid = fopen('andeby.txt', 'r');

% sjekket at aapningen gikk bra
if fid == -1
    disp('Kunne ikke aapne filen')
else
    % leser alle linjene
    while ~feof(fid)
        % leser en linje
        filLinje = fgetl(fid);
        % skriver ut linjen
        fprintf('%s\n', filLinje)
    end % while

    lukkStatus = fclose(fid);
    if lukkStatus == 0
        disp('Lukket filen')
    else
        disp('Kunne ikke lukke filen')
    end
end
end

```

## Eksempelkjøring

- Leser en og en linje
- Skriver ut hele linjen

```

#navn:Donald Duck#adresse:Andeby 1#faar:1920
#navn:Dolly Duck#adresse:Andeby 2#faar:1922
#navn:Anton Duck#adresse:Andeby 3#faar:1919
#navn:Skrue McDuck#adresse:Bingen#faar:1890
#navn:Bestemor Duck#adresse:Haugen 1#faar:1878
Lukket filen
>>

```

## Plukker ut data: lesAndebyFil2.m

```

% leser alle linjene
while ~feof(fid)
    % leser en linje
    filLinje = fgetl(fid);
    % finner #-ene
    navnStart = 1;
    adresseStart = strfind(filLinje, '#adresse:');
    faarStart = strfind(filLinje, '#faar:');
    % finner linjelengden
    linjeLengde = length(filLinje);
    % plukker ut data
    navn = filLinje(7:adresseStart-1);
    adresse = filLinje(adresseStart+9:faarStart-1);
    faar = filLinje(faarStart+6:linjeLengde);
    fprintf('Navn:   %s\n', navn)
    fprintf('Adresse: %s\n', adresse)
    fprintf('Faar:   %s\n', faar)
end % while

lukkStatus = fclose(fid);
if lukkStatus == 0
    disp('Lukket filen')
else
    disp('Kunne ikke lukke filen')
end

```

## Kjøreeksempel

```

Navn:   Donald Duck
Adresse: Andeby 1
Faar:   1920
Navn:   Dolly Duck
Adresse: Andeby 2
Faar:   1922
Navn:   Anton Duck
Adresse: Andeby 3
Faar:   1919
Navn:   Skrue McDuck
Adresse: Bingen
Faar:   1890
Navn:   Bestemor Duck
Adresse: Haugen 1
Faar:   1878
Lukket filen

```

## Andeby-”database”

- Leser inn persondata til en vektor av strukturer
- Personpost (struktur):

Navn:	<input type="text" value="Donald Duck"/>
Adresse:	<input type="text" value="Andeby 1"/>
Faar:	<input type="text" value="1920"/>

- Vektor av personer:

1	2	3	4	5
Navn: <input type="text" value="Donald Duck"/>	Navn: <input type="text" value="Dolly Duck"/>	Navn: <input type="text" value="Anton Duck"/>	Navn: <input type="text" value="Skruer McDuck"/>	Navn: <input type="text" value="Bestemor Duck"/>
Adresse: <input type="text" value="Andeby 1"/>	Adresse: <input type="text" value="Andeby 2"/>	Adresse: <input type="text" value="Andeby 3"/>	Adresse: <input type="text" value="Bingen"/>	Adresse: <input type="text" value="Haugen 1"/>
Faar: <input type="text" value="1920"/>	Faar: <input type="text" value="1922"/>	Faar: <input type="text" value="1919"/>	Faar: <input type="text" value="1890"/>	Faar: <input type="text" value="1978"/>

## lesAndeby.m

```
function data = lesAndeby
% leser persondata fra fil til vektor

% aapner filen
fid = fopen('andeby.txt', 'r');

% sjekket at aapningen gikk bra
if fid == -1
    exit('Feil i lesAndeby: Kunne ikke aapne filen')
else
    % leser alle linjene
    personNr = 1;
    while ~feof(fid)
        % leser en linje
        filLinje = fgetl(fid);
        % finner #-ene
        navnStart = 1;
        adresseStart = strfind(filLinje, '#adresse:');
        faarStart = strfind(filLinje, '#faar:');
        % finner linjelengden
        linjeLengde = length(filLinje);
        % plukker ut data
        data(personNr).navn = filLinje(7:adresseStart-1);
        data(personNr).adresse = filLinje(adresseStart+9:faarStart-1);
        data(personNr).faar = str2num( filLinje(faarStart+6:linjeLengde) );
        % neste personnr
        personNr = personNr + 1;
    end % while

    lukkStatus = fclose(fid);
    if lukkStatus == 0
        disp('Lukket filen')
    else
        disp('Kunne ikke lukke filen')
    end
end
```



## Kjøreeksempel

```
>> dataVektor = lesAndeby;
Lukket filen
>> dataVektor(1)

ans =

      navn: 'Donald Duck'
   adresse: 'Andeby 1'
      faar: 1920

>> dataVektor(5).navn

ans =

Bestemor Duck

>>
```

## Å legge til en karakter

- Lager funksjon som registrerer ny Andeby-karakter
- Tar person-post som inn-parameter
- Åpner filen for å legge til ('a')
- Lager tekstlinje ut fra person-posten
- Skriver tekstlinje til fil
  - fprintf(fid, '%s\n', linje);
- Lukker filen

## regAndebyKarakter.m

```
function regAndebyKarakter(karakter)
% tar inn karakter og legger til i datafilen

% aapner filen for tillegg av data
fid = fopen('andeby.txt', 'a');

% sjekket at aapningen gikk bra
if fid == -1
    exit('Feil i lesAndeby: Kunne ikke aapne filen')
else
    linje = ['#navn:' karakter.navn];
    linje = [linje '#adresse:' karakter.adresse];
    linje = [linje '#faar:' num2str(karakter.faar) ];
    fprintf(fid, '%s\n', linje);
end

% lukker filer
lukkStatus = fclose(fid);
if lukkStatus == 0
    disp('Lukket filen')
else
    disp('Kunne ikke lukke filen')
end

end % function
```

## Kjøreeksempel

```
>> p.navn = 'Guffen';
>> p.adresse = 'Haugen 1';
>> p.faar = 1895;
>> regAndebyKarakter(p)
Lukket filen
>> lesAndebyFil
#navn:Donald Duck#adresse:Andeby 1#faar:1920
#navn:Dolly Duck#adresse:Andeby 2#faar:1922
#navn:Anton Duck#adresse:Andeby 3#faar:1919
#navn:Skrue McDuck#adresse:Bingen#faar:1890
#navn:Bestemor Duck#adresse:Haugen 1#faar:1878
#navn:Guffen#adresse:Haugen 1#faar:1895
Lukket filen
>>
```