NTNU

Kunnskap for en bedre verden

TDT4110 Informasjonsteknologi grunnkurs:

Datastrukturer (kap. 8)

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

Læringsmål og pensum

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

Datastrukturer

- Datatype:
  - Datatypen til en variabel er de verdiene som variabelen kan ha
  - Har ofte et sett operasjoner som kan utføres på variabler av den aktuelle datatypen.
  - single, double, int8, int16, int32, int64, char, logical
- Datastruktur:
  - Samling variabler, gjerne av ulik datatype, forbundet på ulike måter
  - For eksempel: Vektorer og matriser
    - Alle elementer av samme type
    - Fast struktur (alle rader har like mange kolonner)

Eks

Initialiserer nysetning til en tom vektor - da blir elementene tall - ascii-verdiene til bokstavene som legges inn.

Siden nysetning ble initialisert som en tom vektor, er den nå blitt en vektor bestående av ASCII-verdiene til de tegnene som ble lagt inn.

```
function ord = fjernIkkeBokstav(setning)
    nysetning = [];
    tegnNr = 1;
    for i=1:length(setning)
        if isletter(setning(i))
            nysetning(tegnNr) = setning(i);
            tegnNr = tegnNr+1;
        end%if
    end%for
    ord=nysetning;
end%function
```

Initialiserer nysetning til en tom tekst - da blir elementene som legges inn tegn.

Siden nysetning ble initialisert som en tom streng, er den nå blitt en vektor bestående av bokstaver/tegn.

```
function ord = fjernIkkeBokstav(setning)
    nysetning = '';
    tegnNr = 1;
    for i=1:length(setning)
        if isletter(setning(i))
            nysetning(tegnNr) = setning(i);
            tegnNr = tegnNr+1;
        end%if
    end%for
    ord=nysetning;
end%function
```

Problem

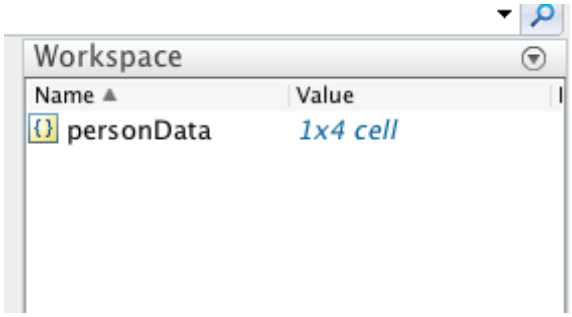
- Vi ønsker å legge inn data av forskjellig type i en variabel
  - Samle info om en person på et sted i koden
    - Navn (tekst), fødselsår (tall), telefonnr (tall), by (tekst)
  - En vektor kan bare inneholde en datatype

```
>> personData = ['Nils' 1982 95725634 'Trondheim']
Warning: Out of range or non-integer values truncated
during conversion to character.

personData =

Nils□□Trondheim
```

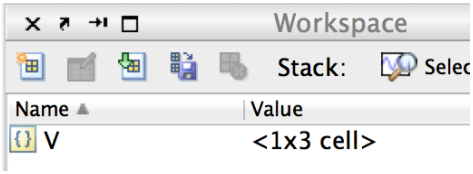

Løsning

- Vi ønsker å legge inn data av forskjellig type i en variabel
  - Samle info om en person på et sted i koden
    - Navn (tekst), fødselsår (tall), telefonnr (tall), by (tekst)
  - **Cell Array**

```
>> personData = {'Nils' 1982 95725634 'Trondheim'}

personData =

    'Nils'    [1982]    [95725634]    'Trondheim'
```



Cell Arrays



- Vektorer/Matriser der elementene er cells
  - CellArray kan lagre data av ulike typer
- Indekseres som ordinære vektorer/matriser
- Bruker *sløyfe/krøll-parenteser:* {}
  - V = {1910, 'NTNU', 'Trondheim'}
- Innholds-indeksering:
  - V{1} = 1910 (tall, double)
  - V{2} = 'NTNU' (string, 4 tegn lang)
  - V{3} = 'Trondheim' (string, 9 tegn lang)
- Celle-indeksering
  - V(1) = [1910] (en celle som inneholder 1910)

Cell Arrays, fortsatt

```
>>V = {1910,'NTNU','Trondheim'}
V =
    [1910]    'NTNU'    'Trondheim'

>>V{1}
ans =
        1910

>>V(1)
ans =
    [1910]

>>V(2) = 'UiO'
Conversion to cell from char is not possible.

>>V{2} = 'UiO'
V =
    [1910]    'UiO'    'Trondheim'
```

Cell Arrays, fortsatt

- Pre-allokering
  - cell(<rader>, <kolonner>)
  - Oppretter en cell array med tomme elementer ([]).
- length() / size()
  - Som for alminnelige tabeller
- Referere til deler av cell arrays
  - Som for tabeller
  - V{1:2}
    - To første celle-verdiene i V
- celldisp()
  - Skriver ut alle elementene

```
>> a=cell(3,4)

a =

    []    []    []    []
    []    []    []    []
    []    []    []    []
```

```
>> V{2}='UiO'
V =
    [1910]    'UiO'    'Trondheim'
>> V{1:2}
ans =
        1910
ans =
UiO
>> celldisp(V)
V{1} =
        1910
V{2} =
UiO
V{3} =
Trondheim
>>
```


Smart: Lagring av tekststrenger

- Kan lagre tekststrenger av *ulik* lengde i samme datastruktur.

```
>> bilmerker = {'Audi', 'BMW', 'Ford', 'Toyota', 'VW'}

bilmerker =

    'Audi'    'BMW'    'Ford'    'Toyota'    'VW'

>> bilmerker{3}

ans =

Ford

>> bilmerker{3}(2)

ans =

o

>>
```

Eksempel - enkel persondatabase

```
function data = personData()
    % Legger inn noen utgangsdata
    data = cell(3,4);
    data{1,1} = 'Terje'; data{1,2} = 1977; data{1,3} = 54736527;
    data{1,4} = 'Trondheim';
    data{2,1} = 'Nils'; data{2,2} = 1983; data{2,3} = 87235412;
    data{2,4} = 'Namsos';
    data{3,1} = 'Kari'; data{3,2} = 1989; data{3,3} = 36457398;
    data{3,4} = 'Bergen';
end %function
```

```
>> test = personData()

test =

    'Terje'    [1977]    [54736527]    'Trondheim'
    'Nils'     [1983]    [87235412]    'Namsos'
    'Kari'     [1989]    [36457398]    'Bergen'
```


Eksempel - enkel persondatabase

```
function skrivUtData(personData)
    [rad,kol] = size(personData);
    fprintf('\n\nNavn\tFødt\tTlf\t\tBy\n')
    fprintf('————————————————————-\n')
    for i=1:rad
        fprintf('%s\t',personData{i,1})
        fprintf('%d\t',personData{i,2})
        fprintf('%d\t',personData{i,3})
        fprintf('%s\n',personData{i,4})
    end %for
end %function
```

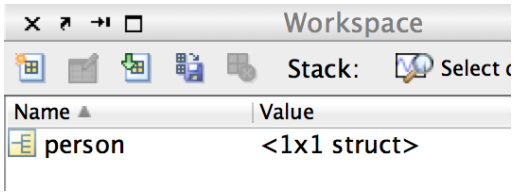
```
>> skrivUtData(test)


Navn  Født  Tlf        By
————————————————————————————
Terje 1977  54736527   Trondheim
Nils  1983  87235412   Namsos
Kari  1989  36457398   Bergen
```

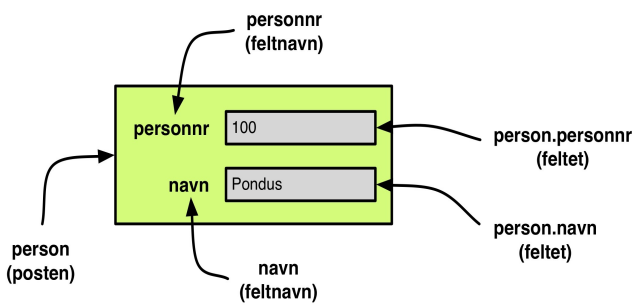
Eksempel - enkel persondatabase

```
function data = leggTilData(personData)
    [rad,kol] = size(personData);
    mereData = true;
    nyRad = rad + 1;
    fprintf('\n\n')
    while mereData
        fprintf('Vennligst oppgi:\n\n')
        personData{nyRad,1} = input('     navn: ','s');
        personData{nyRad,2} = input('fødselsår: ');
        personData{nyRad,3} = input('    tlfNr: ');
        personData{nyRad,4} = input('       by: ','s');
        nyRad = nyRad + 1;
        videre = input('Skal du legge inn flere personer (ja eller nei): ','s');
        mereData = strcmp((lower(videre)),'ja')
    end %while
    data = personData;
end %function
```


Structures

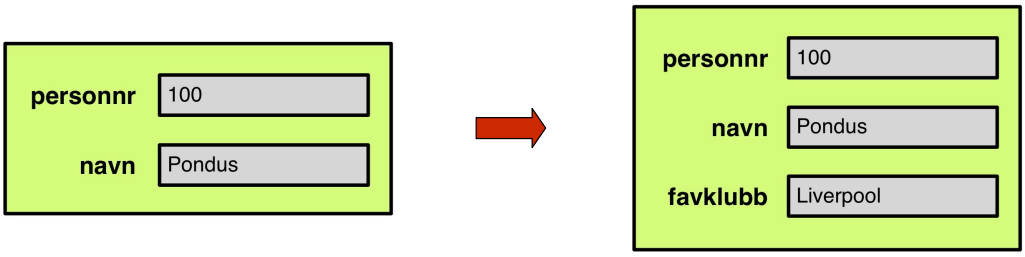
- Datastruktur som grupperer sammen verdier som hører sammen
- Navngitte felter (fields)
- person = struct('personnr',100,'navn','Pondus')
- person.personnr = 100
- person.navn = 'Pondus'





Structures, forts.

- Å referere til feltene
  - <struktur-variabel>.<feltnavn> (kalles **dot operator**)
  - person.favklubb = 'Liverpool'
  - Legger til et nytt felt i strukturen
- Pre-allokering er lurt: struct(felt1, verdi1, felt2, verdi2…)
  - Legg inn det siste elementet først! (da preallokeres en vektor med nok elementer)
- rmfield(<struct-variabel>,<feltnavn>) fjerner felt



Eksempel - enkel persondatabase med strukturer

```
function data = personDataStruct()
    % Legger inn noen utgangsdata. Legger inn element 3 først – dette
    % setter av plass til de to første elementene også.
    data(3) = struct('Navn','Kari','Aar',1989,'Tlf',36457398,'By','Bergen');
    data(1) = struct('Navn','Terje','Aar',1977,'Tlf',54736527,'By','Trondheim');
    data(2) = struct('Navn','Nils','Aar',1983,'Tlf',87235412,'By','Namsos');
end %function
```

```
>> struktur = personDataStruct()

struktur =

1x3 struct array with fields:

    Navn
    Aar
    Tlf
    By
```

```
>> struktur(2).Tlf

ans =

    87235412
```

Eksempel - enkel persondatabase

```
function skrivUtDataStruct(personData)
    lengde = length(personData);
    fprintf('\n\nNavn\tFødt\tTlf\t\tBy\n')
    fprintf('————————————————————\n')
    for i=1:lengde
        fprintf('%s \t%d \t%d \t%s\n',personData(i).Navn,...
            personData(i).Aar, personData(i).Tlf,personData(i).By)
    end %for
end %function
```

```
>> skrivUtDataStruct(struktur)


Navn        Født  Tlf         By
————————————————————————————————————
Terje       1977  54736527    Trondheim
Nils        1983  87235412    Namsos
Kari        1989  36457398    Bergen
```


Eksempel - enkel persondatabase

```
function data = leggTilDataStruct(personData)
    nesteElement = length(personData)+1;
    mereData = true;
    fprintf('\n\n')
    while mereData
        fprintf('Vennligst oppgi:\n\n')
        personData(nesteElement).Navn = input('     navn: ','s');
        personData(nesteElement).Aar = input( 'fødselsår: ');
        personData(nesteElement).Tlf = input( '    tlfNr: ');
        personData(nesteElement).By = input(  '       by: ','s');
        nesteElement = nesteElement + 1;
        videre = input('Skal du legge inn flere personer (ja eller nei): ','s');
        mereData = strcmp((lower(videre)),'ja')
    end %while
    data = personData;
end %function
```

Strukturer

- Kan nøste strukturer inni strukturer
- Kan ha vektorer og matriser med strukturer som elementer
- Kan overføre (hele) strukturer til funksjoner
- Kan returnere strukturer fra funksjoner
- Fordeler:
  - Navngitte felt
  - Ulike datatyper
- Ulemper:
  - Kan ikke indeksere feltene i en løkke etc. like lett som elementene i vektorer og matriser.

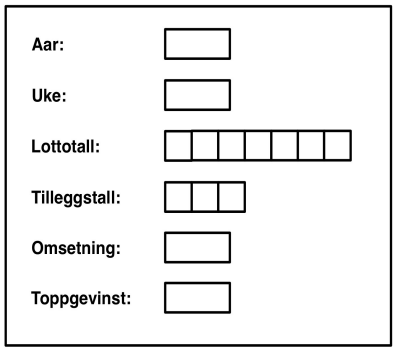
(Database-) Poster

- Lottospill
- Holde data for en uke

Aar:	2012
Uke:	42
Lottotall:	12 15 19 22 27 28 33
Tilleggstall:	9 29 31
Omsetning:	72023064
Toppgevinst:	6248000

Eksempel

```
clear, clc

lottoUke.aar = 2011;
lottoUke.uke = 41;
lottoUke.lottotall = [2 3 7 8 19 22 32];
lottoUke.ttall = [15 21 24];
lottoUke.toppgev = 3929655;
lottoUke.omsetning = 70172420;

disp(lottoUke)
disp(lottoUke.toppgev)
```




```
          aar: 2011
          uke: 41
    lottotall: [2 3 7 8 19 22 32]
        ttall: [15 21 24]
      toppgev: 3929655
    omsetning: 70172420

     3929655

>>
```

NTNU

Alternativ - cell array

```
>> lottoRekke = cell(1,6)
lottoRekke =
    []    []    []    []    []    []
>> lottoRekke{1} = 2011
lottoRekke =
    [2011]    []    []    []    []    []
>> lottoRekke{2} = 41
lottoRekke =
    [2011]    [41]    []    []    []    []
>> lottoRekke{3} = [2 3 7 8 19 22 32]
lottoRekke =
    [2011]    [41]    [1x7 double]    []    []    []
>> lottoRekke{4} = [15 21 24]
lottoRekke =
    [2011]    [41]    [1x7 double]    [1x3 double]    []    []
>> lottoRekke{5} = 70172420
lottoRekke =
    [2011]    [41]    [1x7 double]    [1x3 double]    [70172420]    []
>> lottoRekke{6} = 3929655
lottoRekke =
  Columns 1 through 5
    [2011]    [41]    [1x7 double]    [1x3 double]    [70172420]
  Column 6
    [3929655]
>> celldisp(lottoRekke)
lottoRekke{1} =
        2011
lottoRekke{2} =
    41
lottoRekke{3} =
     2     3     7     8    19    22    32
lottoRekke{4} =
    15    21    24
lottoRekke{5} =
    70172420
lottoRekke{6} =
     3929655
>>
```

NTNU

Lotto-database

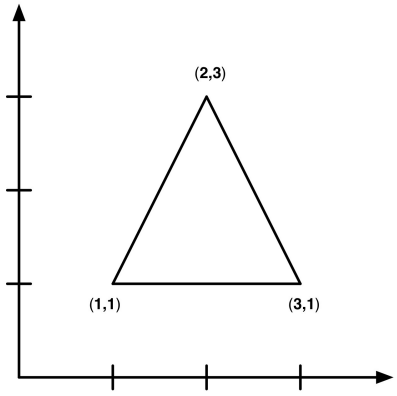
- Vektor av Lotto-poster

	1	2	3	4	5
Aar:	2012	2012	2012	2012	2012
Uke:	38	39	40	41	42
Lottotall:	13 15 16 23 25 26 32	7 8 9 19 20 21 33	13 14 22 24 27 28 30	3 6 13 21 22 26 33	12 15 19 22 27 28 33
Tilleggstall:	17 21 29	2 5 23	4 25 26	16 17 30	9 29 31
Omsetning:	70707140	77286364	70914244	70928940	72023064
Toppgevinst:	12267685	2681835	2460720	3076540	6248000

- Databasefunksjoner
  - Sett inn post, endre post, slette post
  - Maks gevinst, gjennomsnittlig omsetning, "form-tall", …
- Utfordringer?

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];
```


Omkrets av ”struct-mangekant”

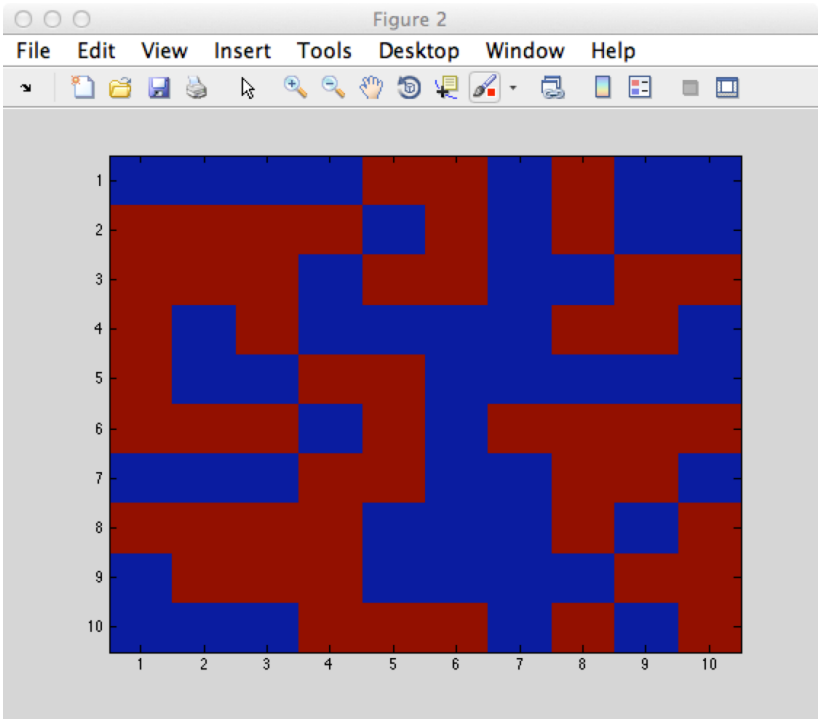
- Skriv kode for de andre alternativene selv.

```
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
```

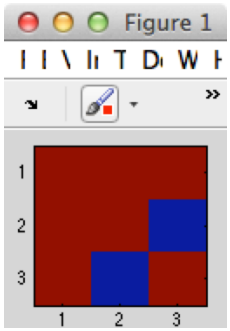
Problem: Søke etter del-tabell

- 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?


```
>> 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
>>
```




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



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.
```

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

Eksamen 2013 og konte 2014

- Oppgave 4 (2013) er egnet for Cell Arrays
- Oppgave 4d (2014kont) er egnet for Cell Arrays