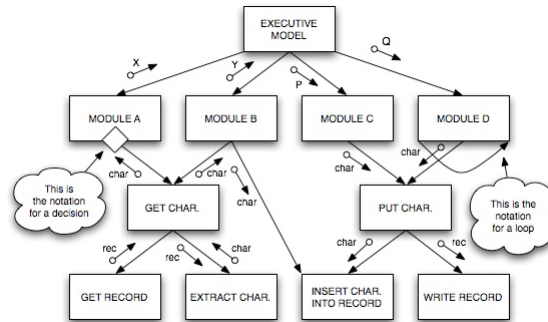
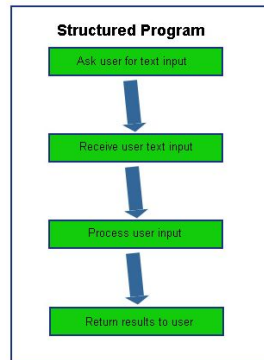


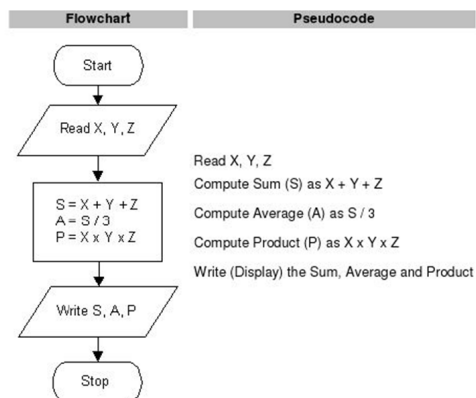
Programming in MATLAB

- Structured programming



Programming in MATLAB

- Build a framework of your data structure, e.g. by a flow chart
 - to get an overview of what data is required in the various subparts of your program
- Transform the flow chart to a pseudocode
 - A further explanation of the data flow in your program
- Write the MATLAB code

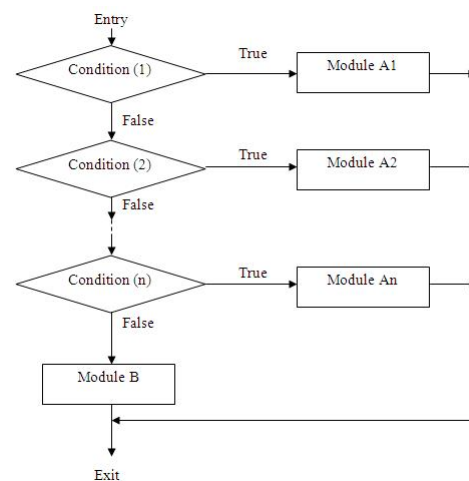


Programming in MATLAB

- Structured programming
 - Decisions
 - `if ... else` structures
 - `switch` structures
 - Loops
 - `for ... end` loops
 - `while ... end` loops

Structured programming

- Decisions
 - `if ... else` structures
 - `switch` structures



Structured programming – Decisions

- `if ... else` structure

- Execute a set of instructions if a logical condition is true.

```
if condition
    statements
end
```

- Example:

A programme to evaluate whether a grade is passing the exam

```
function passing_exam(grade)
% Determines whether the grade is passing
% input: grade=numerical value of grade (0-100)
% output: Displayed message
if grade >= 60
    disp('Passing grade')
end
```

Structured programming – Decisions

- `if ... else` structure

- Example 2

A programme to evaluate the sign of a number

```
function sgn = my_sign(x)
% my_sign(x) returns 1 if x is greater than 0
%                -1 if x is less than zero
%                0 if x is equal to zero
if x > 0
    sgn = 1;
elseif x < 0
    sgn = -1;
else
    sgn = 0;
end
```

Structured programming – Decisions

- `switch` structure

- This decisional structure is similar to the `if ... elseif` structure, but rather than testing individual conditions, the branching is based on the value of a single test expression. Depending on the value of its value, different blocks of code are implemented.

- General syntax

```
switch testexpression
case value_1
    statements_1
case value_2
    statements_2
    ...
    ...
otherwise
    statement_otherwise
end
```



Structured programming – Decisions

- Example 3

- A small program to return a message depending on the value of the string variable `grade`

```
grade = 'B';
switch grade
case 'A'
    disp('Excellent')
case 'B'
    disp('Good')
case 'C'
    disp('Mediocre')
case 'D'
    disp('Whoops')
otherwise disp('??')
end
```



Structured programming – Decisions

- Relational operators in MATLAB

```
x == 0   Equal to
x ~= 0   Not equal to
x < 0    Less than
x > 0    Greater than
x <= 0   Less than or equal to
x >= 0   Greater than or equal to
```

Note in particular:

- ~ (not) Performs a logical negation on an expression
`~expression`
- & (and) Performs a logical conjunction on two expressions
`expression1 & expression2`
- | (or) Performs a logical disjunction on two expressions
`expression1 | expression2`

Structured programming

- **Exercise**

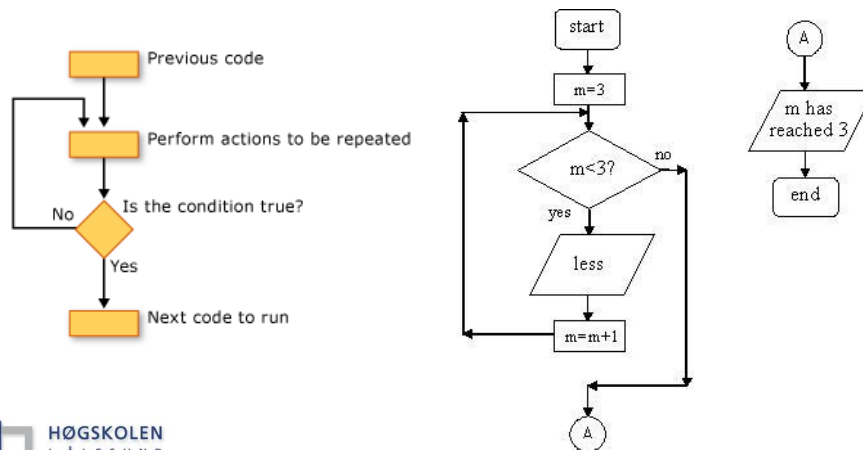
Create a code that, given an age, it checks whether you can buy alcohol in Norway:

- Age below 18 – Can not buy alcohol
- Age between 18 and 20 – Can buy alcohol below 22%
- Age above 20 – Can buy all kinds of alcohol in Norway

```
age = 20;
if age < 18
    disp('You are not allowed to buy alcohol')
elseif age >= 18 & age < 20
    disp('You can buy alcohol below 22%')
else
    disp('You can buy all sorts of alcohol')
end
```

Structured programming - Loops

- Loops
A structure to perform operations repetitively



Structured programming - Loops

- Two types of loops exists
 - `for ... end` Repeats statements a specific number of times
 - `while ... end` Repeats statements as long as a logical condition is true

- `for ... end` structure; the general syntax is

```
for index = start:step:finish
    statements
end
```

- `while ... end` structure; the general syntax is

```
while condition
    statements
end
```

Structured programming - Loops

- **The `for` structure**

This structure allows statements to be repeated a specific number of times

```
for i = 1:4
    i
end
```

Result →

```
1
2
3
4
```

Pseudo code:

```
"for i from 1 to 4"
"show i"
"stop"
```



Structured programming - Loops

- **Example 4**

A simple function to calculate the factorial of a number

```
function fout = factor(n)
% factor(n)
% Computes the product of all the integers from 1 to n
x = 1; % in order to provide the correct result of 0!
for i = 1:n
    x = x*i;
end
fout = x;
end
```



Structured programming - Loops

- **Exercise:** Collect data from all students in a class.
 - Get names and age from all students
 - Sum all the ages
 - Find average age

- **MATLAB code:**

```
% Collect names in the vector "all_names"  
% Collect ages in the vector "all_age"  
all_names = {'aa' 'bb' 'cc'};  
all_ages = [21 23 27];  
sum_age = sum(all_ages);  
num_stud = length(all_ages)  
average = sum_age/num_stud
```

- Who wants to explain the MATLAB code?



Structured programming - Loops

- **Exercise (cont)**
For all students, find the number of days alive
- Pseudo-code: get every age, multiply by the number of days in a year

- **MATLAB code:**

```
for i = 1:num_stud  
    all_ages(i)*365;  
end
```

- Who wants to explain this MATLAB code?
- How can we save our number of days in another vector, e.g. called `days`
- How can we calculate the average of days lived?



Structured programming - Loops

- **Loops and vectorization**

The `for` loop is easy to implement and understand. However, for MATLAB, this may be an inefficient way of representing data.

- A cosine-function can be built with a for loop

```
i = 0;
for t = 0:0.02:50
    i = i + 1;
    y(i) = cos(t);
end
```

- A more efficient way to establish this is in the vectorized way

```
t = 0:0.02:50;
y = cos(t);
```

Structured programming - Loops

- **Loops to make many plots**

The `plot` command can be used inside a loop to make several plots e.g. with varying parameters.

- Harmonic functions can be built with varying frequency

```
t = 0:pi/10:10*pi;
for w = 0.2:0.2:0.6
    y = cos(w*t);
    plot(t,y)
    hold on
end
```

Structured programming - Loops

- The **while** structure

```
while condition
    statements
end
```

The statements between the while and the end are repeated as long as the conditions is true

- A simple example

```
x = 8;
while x > 0
    x = x - 3;
    disp(x)
end
```

Structured programming - Loops

- We can also introduce a new element into the **while** loop, to stop the repetition. This is called the **while ... break** structure

```
while (1)
    statements
    if condition, break, end
    statements
end
```

- A simple example

```
x = 8;
while (1)
    x = x - 5;
    if x < 0, break, end
end
```

Structured programming - Loops

- **Exercise:**
Find the number of days alive, using the `while` structure
- Pseudo-code: While the number of remaining students is non-zero, get the age of the next student, multiply by the number of days in a year, stop when all students have been processed.

```
i = 1;
while i <= num_stud
    all_ages(i)*365
    i = i + 1;
end
```

- Who wants to explain the code?

Structured programming – Combining decisions and loops

- **Exercise:**
For all students in a class, check if they are allowed to buy alcohol or not
- Pseudo-code: For each of the students in the class, get the age of the student, check his/her allowance wrt. buying alcohol

```
all_ages = [19, 20, 27, 22, 23, 25];
N=length(all_ages);
for i = 1:N
    if all_ages(i) < 18
        all_ages(i)
        disp('Not allowed to buy alcohol')
    elseif all_ages(i) >=18 & all_ages(i) < 20
        all_ages(i)
        disp('Can buy alcohol below 22%')
    else
        all_ages(i)
        disp('can buy all sorts of alcohol')
    end
end
end
```

Structured programming – Combining decisions and loops

- The code can be further modified:

```
all_names={'Per', 'Kari', 'Ola', 'Hans','Mari'}
all_ages = [17, 27, 19, 21, 25];
N1=length(all_names);
N2=length(all_ages);
if N1 ~= N2
    disp('Error in length of vectors')
    fprintf('N1=%d and N2=%d \n',N1,N2)
    return
end
underage='is not allowed to buy alcohol';
beer_and_wine='can buy alochol below 22%';
liqor='can buy all sorts of alcohol';
```

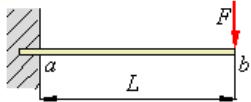
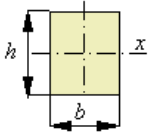
Structured programming – Combining decisions and loops

- ...cont...

```
for i = 1:N
    if all_ages(i) < 18
        status{i} = underage
        text = {char(all_names(i)), ' ', char(status{i})};
        disp(text)
    elseif all_ages(i) >=18 & all_ages(i) < 20
        status{i} = beer_and_wine
        text = {char(all_names(i)), ' ', char(status{i})};
        disp(text)
    else
        status{i} = liqor
        text = {char(all_names(i)), ' ', char(status{i})};
        disp(text)
    end
end
end
```

Simple mechanical problem – A cantilever beam

- Use simple mechanical formulas

Cross section	Bending
<p>Elementary equations for uniform beams subjected to bending:</p> 	$M_x = S_x \sigma_b, \quad S_x = \frac{I_x}{y_{\max}}$ $I_x = \int y^2 dA$
	$I_x = \frac{1}{12} b h^3$ $S_x = \frac{1}{6} b h^2$

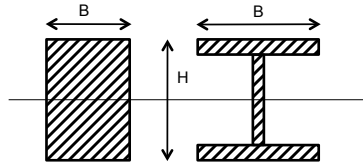
Simple mechanical problem – A cantilever beam

- Problem at hand
 - Given a cantilever beam with length L
 - Select appropriate cross-section properties
 - Establish the deflection at the end of a cantilever beam when subject to a point load P at the end
 - Define an acceptable deflection and compare whether the beam meets the criterion or not
 - Calculate the maximum bending stress for the beam and check whether the allowable stress is exceeded.
 - Yield stress of the material is $\sigma_y = 355 \text{ MPa}$, use a material factor of $\gamma = 1.3$ to determine the allowable stress

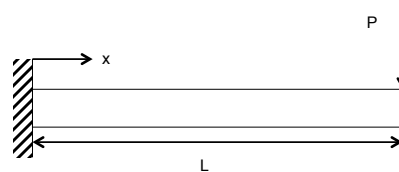
Simple mechanical problem – A cantilever beam

- Pseudo-code

- Define a cross-section,
 - Height (H) and width (B)
 - Moment of inertia (I)



- Define a beam length
- Define a point load



- Define a function which calculates the vertical displacement $y_{max} = \frac{PL^3}{3EI}$
- Define a function which calculates the bending stress $\sigma_b = -\frac{M_b(x)y}{I}$

Simple mechanical problem – A cantilever beam

- Pseudo-code cont.

- Define an allowable deflection
- Define the yield stress, the material factor and the allowable stress
 - Allowable stress is given as $\sigma_a = \frac{\sigma_y}{\gamma}$
- Maximum stress is found at $x = 0$
 - Bending moment at $x = 0$ is given as $M = P \cdot L$
 - Moment of inertia of a rectangular cross-section is given as $I = \frac{B \cdot H^3}{12}$
 - Bending stress at $x = 0$ becomes $\sigma_b = \frac{6PL}{BH^2}$
- Compare the deflection with the allowable deflection
- Compare the maximum stress in the beam with the allowable stress

Simple mechanical problem – A cantilever beam

- MATLAB code

```
H=20;      % Height (mm)
B=300;    % Width (mm)
L=5000;   % Length (mm)
P=800;    % Point load (N)
E=2.1E5   % E-modul [MPa]
% Define allowable deflection
y_a=100;  % (mm)
% Establish allowable stress
sigma_y=355; % Yield stress
gamma=1.3; % Material factor
sigma_a=sigma_y/gamma
%
y=def(P,L,E,H,B)
sigma=bending(P,L,H,B)
```

- Own files (`def.m` and `bending.m`):

```
function [nedb]=def(Force,...
    Lbeam,Young,Height,Width)
    Inrt=Width*Height^3/12
    nedb = Force*Lbeam^3/...
        (3*Young*Inrt)
end

function [spenning] = ...
    bending(Force,Lbeam,...
        Height,Width)
    Inrt=Width*Height^3/12
    spenning = Force*Lbeam*...
        Height/(2*Inrt)
end
```



Simple mechanical problem – A cantilever beam

- This code can be modified to test a lot of options

```
H=10:10:200;      % Height (mm)
B=100:100:1000;  % Width (mm)
L=4000:500:10000; % Length (mm)
P=600:50:1200;   % Point load (N)

%Loop over all variations
for i=1:length(H)
    for j=1:length(B)
        for k=1:length(L)
            for m=1:length(P)
                y(i,j,k,m)=...
                def(P(m),L(k),E,H(i),B(j))
                sigma(i,j,k,m)=...
                sigma(P(m),L(k),H(i),B(j))
            end
        end
    end
end
```

