

SOFTWARE OG TEKNOLOGI

SOM NOGET NYT, KAN SOFTWARE VÆRE ET SELVSTÆNDIGT PRODUKT



KRAV TIL PRODUKT

- omhu og professionalisme ved fremstilling
- dokumentations- og kommunikationsværdi, herunder overskuelighed, sammenhæng, kildehenvisninger og teknisk dokumentation

FRA BEKENDTGØRELSEN:

- Ved programmering af elektroniske komponenter eller PC udarbejdes både flowdiagram eller pseudokode over programmets overordnede struktur og funktion, og selve programkoden med kommentarer.
- Programmeringssprog og anvendte softwareelementer som biblioteker og kommunikationsprotokoller gennemgås.

Overblik

Der findes forskellige ting til at danne overblik eksempelvis det er beskrevet i bekendtgørelsen: pseudokode og flowchart.

Men **inden** skal man, som ved alle andre produkter, tænke på:

1. Krav til produktet, som ved andre typer af produkter.
2. Test af opfyldelse af krav.

Så for hvert eneste krav skal man kunne teste om kravet er opfyldt. Det bør allerede ske fra begyndelsen.

Krav og test

Et eksempel kunne være at man ønskede at udvikle en app, hvor man kan få en alarm X minutter før en bus kører. Ideen er at når man stiger af bussen scanner man en QR kode og app'en finder ud af hvornår den sidste bus kører og giver en alarm 10 min før den kører. Et krav kunne være:

Krav	Test
Det skal være muligt at angive den tid der går fra alarmen til den sidste bus kører.	<p>#1 Indtast 10 min. Se at der kommer en alarm 10 min før sidste bus.</p> <p>#2 Indtast -5 og se at der kommer en fejlbesked: "Ugyldig tid den skal være mellem 1 min til 20 min"</p> <p>#3 Indtast 30 min og se der kommer en fejlbesked: "Ugyldig tid den skal være mellem 1 min til 20 min"</p> <p>#4 Vælg en bus, hvor den sidste kører efter middag og gentag test #1.</p> <p>Osv.</p>

Kilde: Jan Boddum Larsen

25-09-2018

Overblik

Der findes forskellige ting til at danne overblik eksempelvis det er beskrevet i bekendtgørelsen:

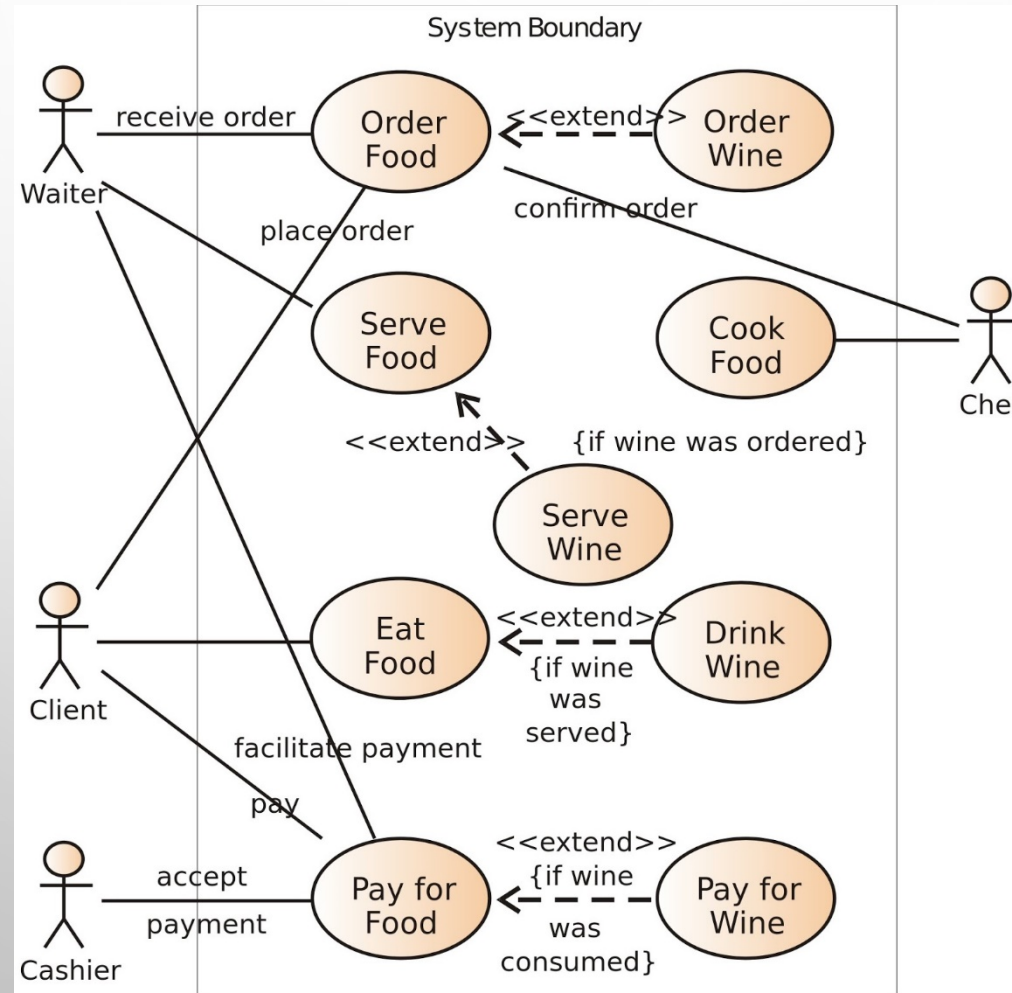
- Pseudokode
- Flowchart

Af andre værktøjer kan nævnes:

- "Use Case"
- Blokdiagrammer

Overblik

Hvis softwareren skal betjenes af flere brugere kan **"USE CASE"** være med til at danne et overblik:

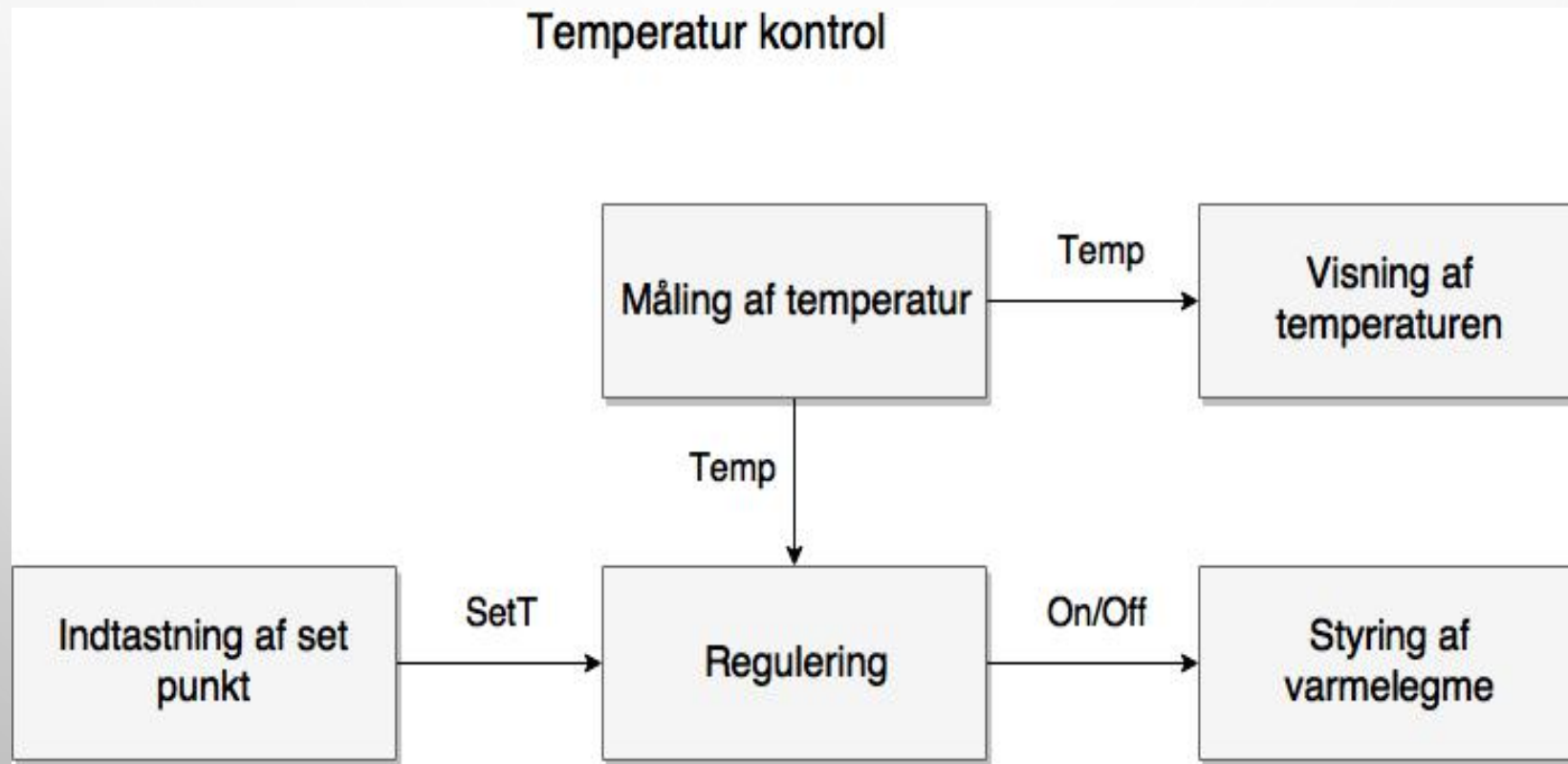


Overblik

Inden egentligt design, kan et blok diagram bruges til at danne overblik.

Hvad laver processen?

Hvilke data er input / output og til hvilke bokse?





DOKUMENTATION AF PROGRAMMET

EN DEL AF UDVIKLINGEN AF PROGRAMMET.

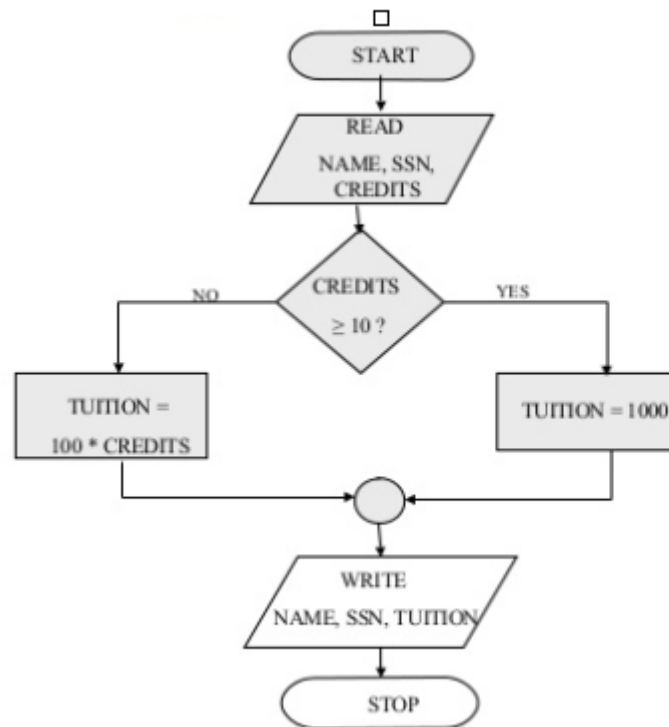
FLOWCHART

PSEUDOKODE

EN SPROGLIG BESKRIVELSE AF KODEN



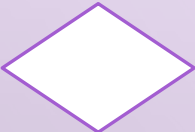


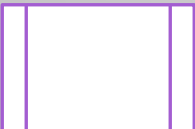
```
If Elevens antal point er mere eller lig med 60  
  Print "Bestået"  
else  
  Print "Dumpet"
```

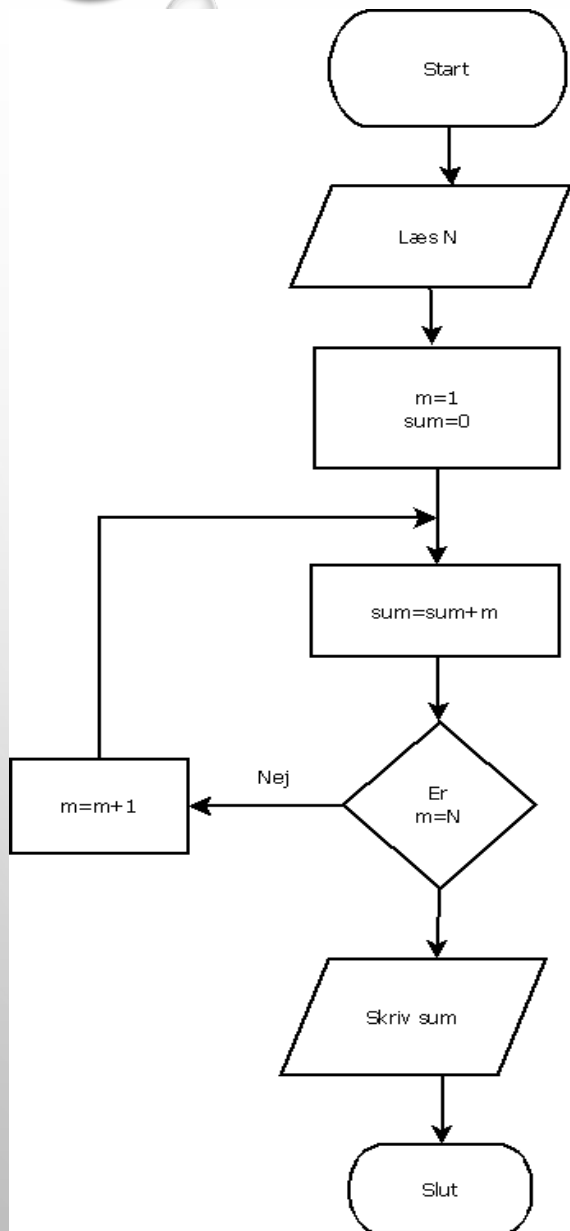
GRAFISK OVERBLIK FLOWCHART



Pseudocode for Tuition problem
Start
Read NAME, SSN, CREDITS
IF CREDITS >= 10 THEN
TUITION = 1000
ELSE
TUITION = 100 * CREDITS
ENDIF
Write NAME, SSN, TUITION
Stop

FLOWDIAGRAM SYMBOLER

Symbol	Hvad betyder symbolet
	Terminal punkt. Angiver start eller slut på processen.
	Proces. En række af handlinger som skal udføres.
	Beslutning. Her kan vælges mellem en proces som udføres hvis betingelsen er sand og en anden proces hvis betingelsen er falsk.
	Forbindelse. Dette symbol bruges hvis diagrammet fylder mere end en side.
	Input / output I denne blok hentes data ind fra omverden til programmet, eller data bringes ud fra programmet. Kunne være indlæsning af en tast, udskrift til skærm eller anden hardware
	Forud defineret proces, kald af en funktion.



Start

Aflæs et antal N

Klargør til løkke. Alle indgående størrelser stilles i kendt status.

m (som er 1 første gang) lægges til variablen sum og gemmes i sum.

Beslutning hvis m er noget op på det indtastede tal N gå ned til næste trin. Hvis den ikke er kommet op på N, læg 1 til m og gem den i m og gentag.

Udskriv sum til et display eller på skærm til bruger.

Slutning på program

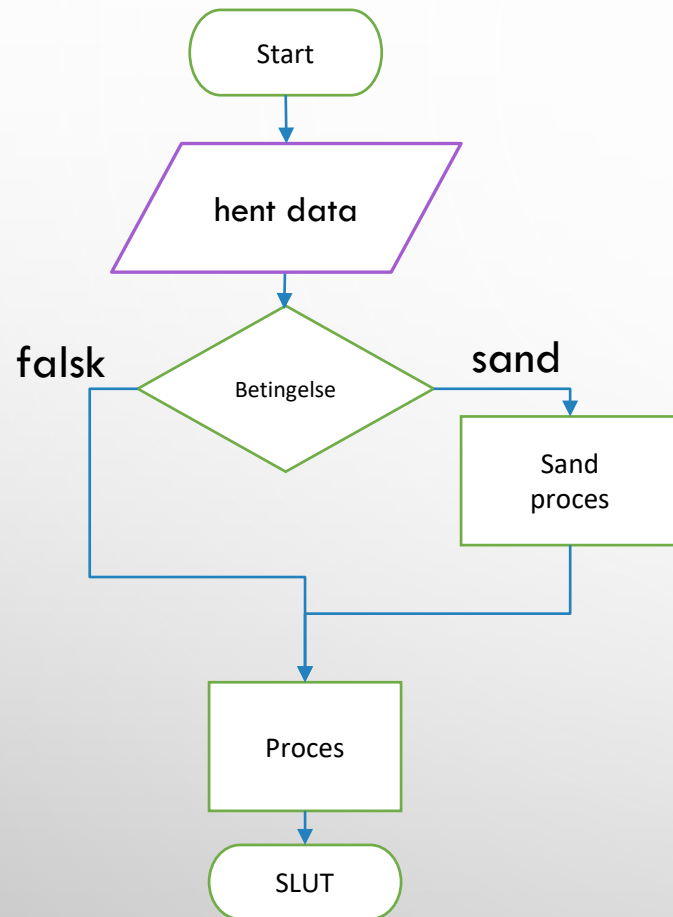
Frit efter: Jan Boddum Larsen

KONTROL STRUKTURER

- Der findes en del forskellige strukturer som man kan anvendes i programmering.
- Disse strukturer kan alle visualiseres ved hjælp af flowdiagrammer.

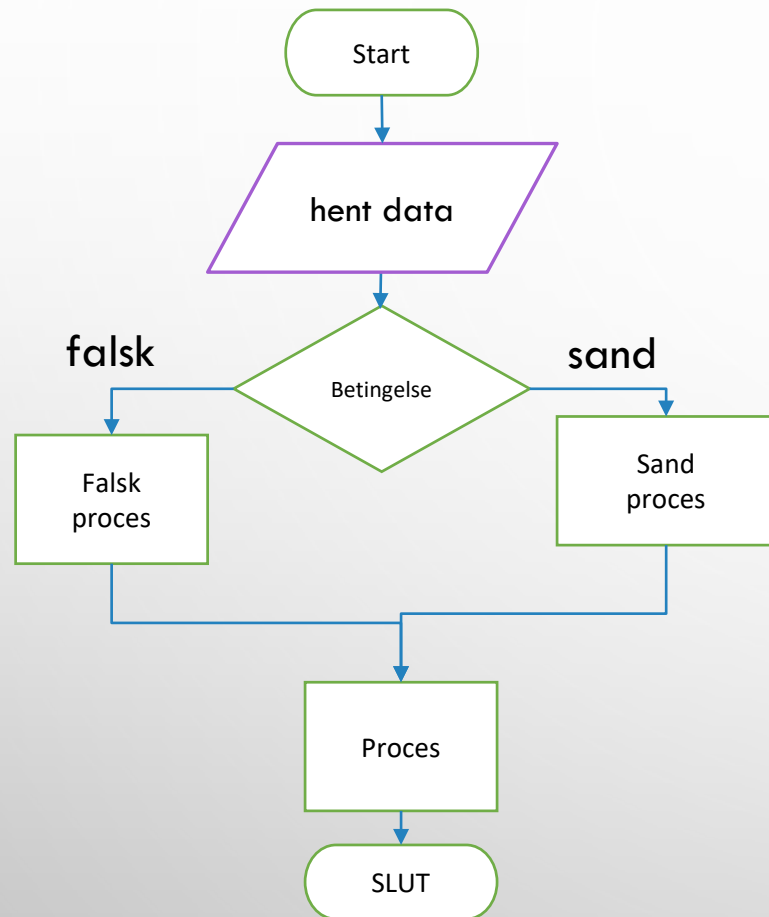
- **If**
- **If ... else**
- **Switch case (eller blot case)**
- **While**
- **Do ... While**
- **For**
- **Function**

IF STRUKTUR



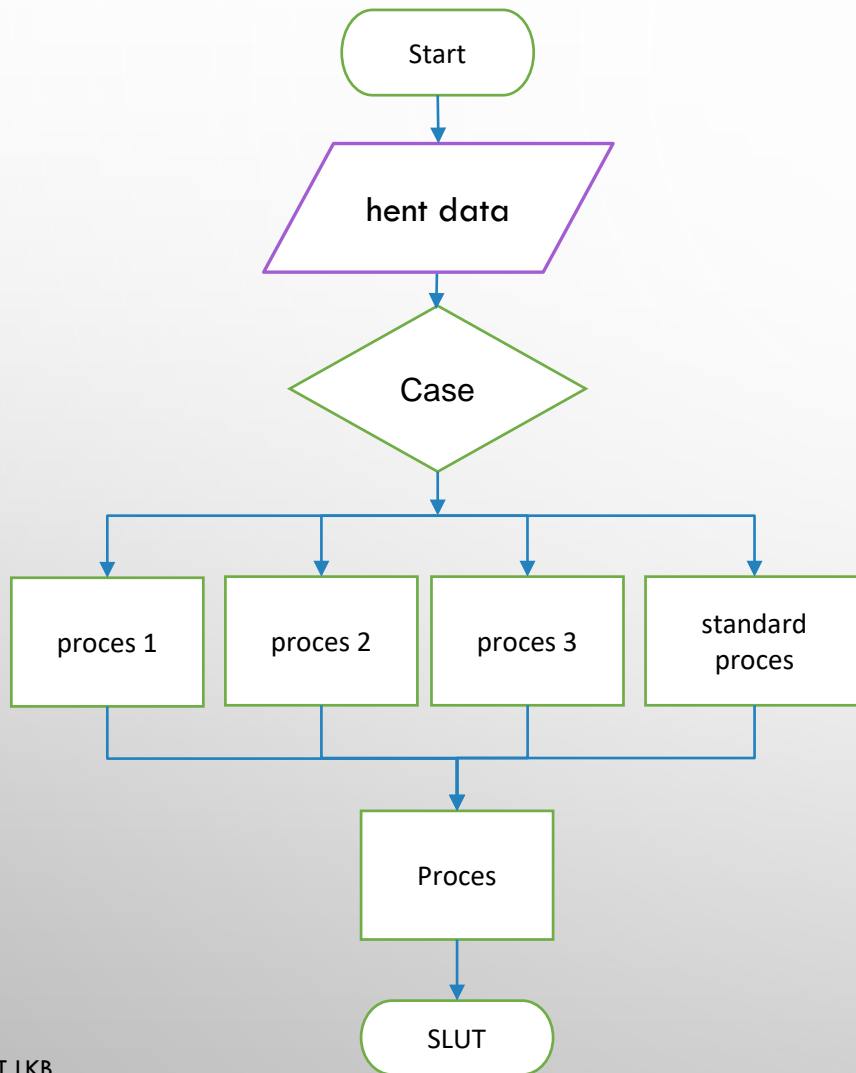
```
.  
.   
.   
if(betingelse)  
{  
    Her skrives koden til den sande proces  
}  
.   
.
```

IF ... ELSE STRUKTUR



```
.  
.   
if(betingelse)  
{  
    Her skrives koden til den sande proces  
}  
else  
{  
    Her skrives koden til den falske proces  
}  
.   
.   
.
```

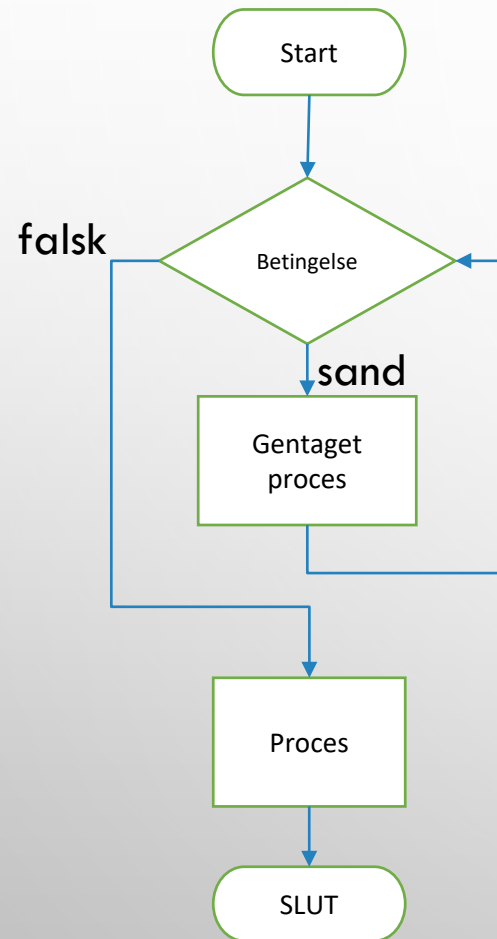

CASE STRUKTUR



```
int sensorReading = analogRead(A0);  
int range = map(sensorReading, sensorMin, sensorMax, 0, 3);
```

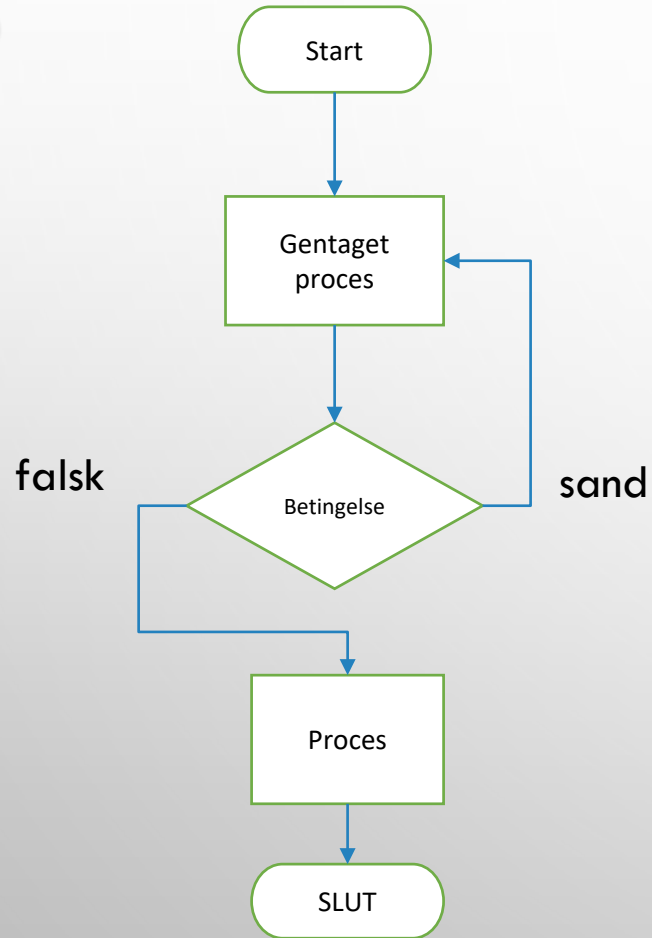
```
switch (range)  
{  
  case 0:  
    Serial.println("dark");  
    break;  
  case 1:  
    Serial.println("dim");  
    break;  
  case 2:  
    Serial.println("medium");  
    break;  
  case 3:  
    Serial.println("bright");  
    break;  
  default:  
    Serial.println("Ingen gyldig måling");  
    break;  
}
```

WHILE



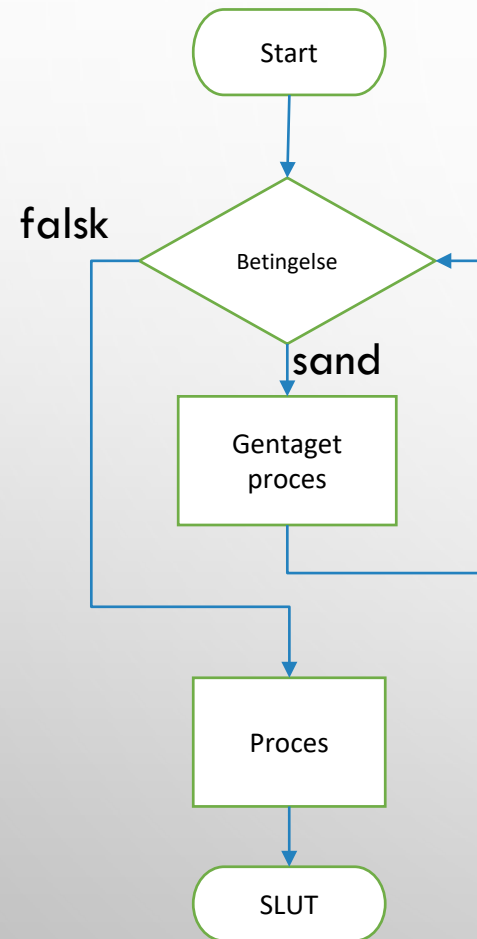
```
var = 0;
while(var < 200){
    // do something repetitive 200 times
    var++;
}
```

DO ... WHILE



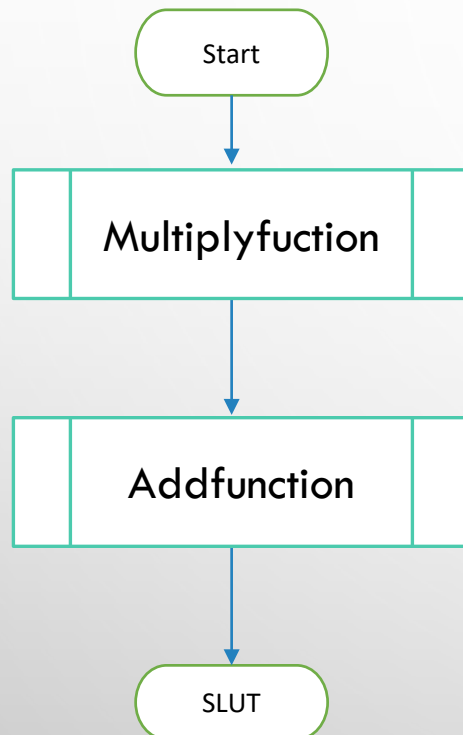
```
do
{
    delay(50);
    x = readSensors();
}
while (x < 100);
```

FOR



```
for (int i=0; i <= 255; i++)  
{  
    // skriv koden som skal udføres et bestemt antal gange  
}
```

FUNCTION



```
void loop()
{
  int produkt = MultiplyFunction(2, 3);
  int Sum = AddFunction(2,3);
}
```

```
int MultiplyFunction(int x, int y)
{
  int result;
  result = x * y;
  return result;
}
```

```
int AddFunktion(int x, int y)
{
  int Result;
  Result = x+y;
  return Result;
}
```