Flowchart generelt, Pseudokode, Algoritmer, Flowcharts, Nassi-Shneiderman

Flowchart

Frank Gilbert udviklede flowcharts-diagrammer tilbage i 1921som et analyseredskab.

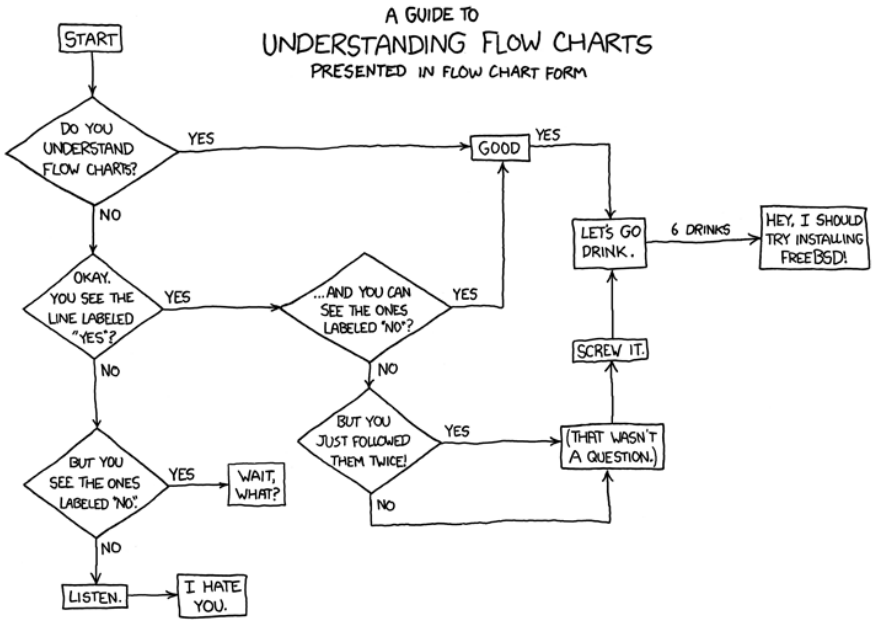
Det blev først præsenteret for American Society of Mechanical Engineers (ASME). Og i 1930'erne overførte industrialisten Allan Mogensen flowcharten til industri- og forretningsverdenen. Se kilde:

Et flowchart bruges til grafisk at beskrive et forløb, der er tidsmæssigt struktureret, og evt. afhængig af ja-nej-beslutninger. Det kan fx være en microcontrollers program-afvikling.

Til at beskrive et program-forløb bruges forskellige "standardiserede" kasser - rektangler, der hver især illustrerer en handling. I kasserne kan der skrives noget tekst, som forklarer hvad der sker i kassen.

Derudover bruges spørgekasser, der er firkanter sat på spidsen. De kan have to udgange, en "ja" og en "nej" udgang.

Eksempel på et flowchart:



Kilde: http://xkcd.com/518/

Der findes nogle gode programmer til at tegne flowcharts med, men de skal købes. Fx EDGE-Diagrammer, eller Smartdraw.

Men der findes også et hav af gratis online-muligheder. Fx https://www.draw.io/

Eller tegn pænere flowcharts online på https://creately.com/app/ eller på https://app.diagrams.net/

Faktisk ret god og let at bruge!! Diagrammer kan gemmes hvis der oprettes et login!!

Men før man kan lave et ” *rigtigt* ” flowchart, kan man med fordel først beskrive et program med en mellemting, såkaldt pseudokode.

Pseudokode:

Inden man skal i gang med at skrive kode, skal man ha et overblik over, hvad programmet skal.

Dvs. i designfasen. Her kommer Pseudokode til sin ret.

Det er nemmere at overskue og forstå 10 linjer Pseudokode end 100 linjer rigtig kode.

Pseudokode er ikke noget der kan oversættes / kompileres af en PC. Der findes ingen programmer, som kan forstå det og oversætte det til maskinkode. Men det er heller ikke meningen med Pseudokoden.

Pseudokode er en simpel - tekstlig – menneskelig forståelig – og i daglig sprog forståelig måde at beskrive et programs funktion. Man skriver i normal dagligt sprog, hvad det er, programmet skal lave, og i hvilken rækkefølge det skal udføres. Hvad skal ske først, dernæst osv.

På en sådan måde, at ikke-programmører kan forstå hvad programmet laver.

Beskrivelsen af de handlinger, som skal til for at løse en opgave, skal være både generel og letforståelig. Derfor bruges almindelige danske sætninger i beskrivelsen.

På den måde undgår man at spilde tid på at overveje særlige detaljer i programmeringssproget, som f.eks. om der i syntaksen skal bruges komma eller semikolon mellem variable, osv.

Men samtidig med at man bruger danske ord, skriver man disse ord i en opsætning, som minder om programmeringssprog. Derfor navnet pseudokode.

Ideelt set kan man skrive sit program direkte i et programmeringssprog ud fra Pseudokoden.

Pseudokode kunne også opfattes som en ” Drejebog ” over koden.

Kan hjælpe med at se hvilke delfunktioner, der kan placeres i Subrutiner / Funktioner / Procedurer

En gennemarbejdet Pseudokode gør det meget lettere at skrive selve koden.	What is pseudocode? Pseudocode is a ***step-by-step written outline of your code*** that you can gradually transcribe into the programming language.

Bl.a. fra <http://htx-elev.ucholstebro.dk/wiki/index.php?title=Pseudokode>

Eksempler på pseudokode:

```
      BEGIN                          // her begynder programmet
hop1                                 //  Mærk efter om du er sulten
      IF sulten                      // her spørges
        THEN spis spaghetti
        ELSE vent en time
     GOTO hop1                       // her hoppes fire linjer op i programmet
     END                             // programmet er færdig
```

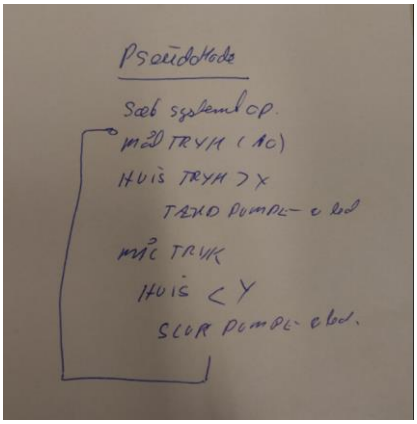
Fra <http://informatik-gym.dk/glossary/pseudokode/>

Eksempel:

Pseudokode til et Arduino-program til et måleprojekt:

Sæt systemet op

Gentag:
  Mål tryksignal på A0
  Hvis tryk > x
    Tænd Pumpe & LED
  Mål tryksignal på A0
  Hvis tryk < y
    Sluk Pumpe & LED
End-gentag



Eksempel:

Definerer variable og pins
Inkluder SoftwareSerial biblioteket på pin 2 & 3
Klokfrekvens variabel = 50 (Hz)

Kør 1 gang {
  Slå interrupts fra
  Nulstil kontrolregistre for timer interrupts
  Sæt timer prescaler
  Indstil til interrupt ved timer overflow
  Slå interrupts til

  Start serielle port, baudrate på 9600
  Sæt pin 13-4 til OUTPUT
}

Loop(){

```
If(seriel data modtaget){

  Aflæs en datapakke fra serielbufferen
  Vent på næste pakke
  Når modtaget:

  Switch(aflæste datapakke){
    Case 202:  //
      Do stuf
      Break;

    Case 203:  //
      Do stuf
      Break;
    }
Else(){
  Sluk begge motorpins
} }
```

Eksempel, Pseudokode for Program til trådløs kommunikation

```
Inkluder bibliotek til software-seriel

Definer pins

Udfør 1 gang {
  Definer indbygget serial port (UART) til 9600 Baud
  Definer HC12-serial til 9600 Baud
}

Gentag{
  Hvis(der er data fra indbygget UART) {
    Send data via HC12
  }

  Hvis (der er data at hente fra HC12) {
    Send data via UART
  }
}
```

Funktion bubblesort (array af heltal)

```
For ( i=0 til længden af array )
  For ( j=0 til længden af array-i )
    Hvis array [ j ] > [ j + 1 ]
    Swap array [ j ] og array [ j + 1 ]
  End for
```

End for
End funktion

Fra <http://pcfly.info/software/2012/02/introduktion-til-pseudokode.html>

Der er ingen regler for hvordan man bør skrive en pseudokode, men så alligevel. Pseudokoden skulle jo gerne skrives, så andre kan følge en tankegang og forstå koden, så her er nogle retningslinjer:

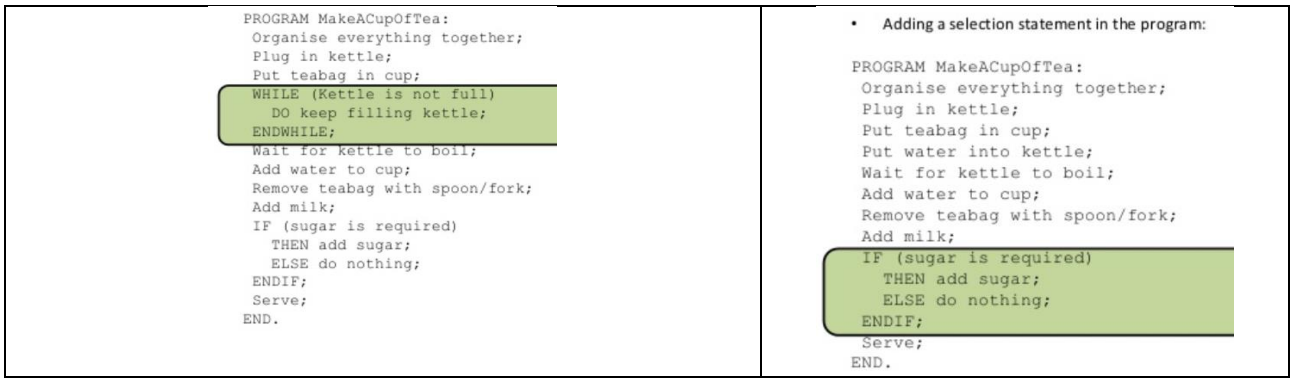
Regel:	Forklaring
Én statement pr linje	Beskriv hver statement på sin egen linje
Hold fokus på meningen.	Beskriv **hvad** programmet skal gøre. Ikke hvordan.
Følg standard programmerings-strukturer	Så vidt muligt følges programmerings-strukturer, hvilket gør det lettere at læse og efterfølgende skrive et program.
Gør brug af Blokke"	Gruppér så meget som muligt i blokke, som kan give yderligere overblik over programmet, og også gøre selve programskrivning lettere.
Lav gerne flere udgaver, - niveauer / lag. Fra generel til mere og mere detaljeret.	De første udgaver af Pseudokoden er mere generel / overordnet, - med få delhandlinger. I 2. udgave deles komplicerede delhandlinger yderligere op, og sådan fortsætter man, indtil man umiddelbart kan skrive kildetekst til valgte programmeringssprog. Variabelnavne og typer må gerne optræde i Pseudokoden, og der må gerne suppleres med tegninger / blokke.

Se evt. en wikiHow her:

Algoritmer:

For at kunne forstå og skrive programmer, er man nødt til at kende til algoritmer, og løkke-strukturer. Altså Principper for at afvikle programmer, programstrukturer, løkker osv.

Sammenhængende løkkestrukturer holdes i (Arduino) C-kode sammen med Tuborg-parenteser. I Pseudokode kan man i stedet bruge KEYWORDS, som fx ENDWHILE eller ENDIF:	



```
PROGRAM MakeACupOfTea:
 Organise everything together;
 Plug in kettle;
 Put teabag in cup;
 WHILE (Kettle is not full)
   DO keep filling kettle;
 ENDWHILE;
 Wait for kettle to boil;
 Add water to cup;
 Remove teabag with spoon/fork;
 Add milk;
 IF (sugar is required)
   THEN add sugar;
   ELSE do nothing;
 ENDIF;
 Serve;
END.
```

- Adding a selection statement in the program:

```
PROGRAM MakeACupOfTea:
 Organise everything together;
 Plug in kettle;
 Put teabag in cup;
 Put water into kettle;
 Wait for kettle to boil;
 Add water to cup;
 Remove teabag with spoon/fork;
 Add milk;
 IF (sugar is required)
   THEN add sugar;
   ELSE do nothing;
 ENDIF;
 Serve;
END.
```

Her er vist nogle Algoritme-Keywords:

Start… end,
For … Endfor, eller Close For
While...Endwhile
Do...Enddo
If...ENDIF
If…ELSE…ENDIF
CASE…ENDCASE
Call;
When;
Input, Læs,
Beregn, Initier

Her et eksempel på, hvordan man kan tilføje " Close " I kommentarer for at vise, hvor strukturen lukkes:

```
for(int i = 0; i < 5; i = i + 2){

   pinMode(MyArray[i], OUTPUT);

}  //close for, eller EndFor
```

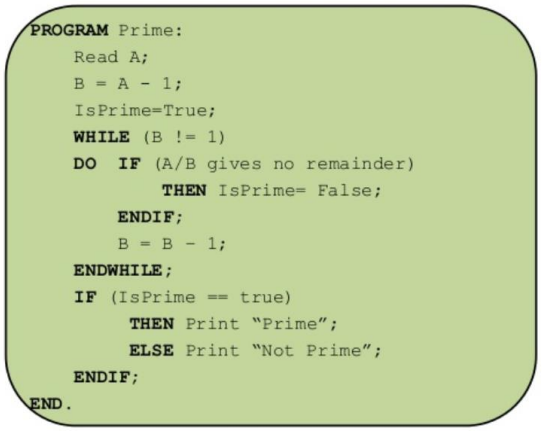
Flere eksempler:

```
IF ( Maden mangler salt )
            THEN tilsæt Salt
            ELSE tilsæt ikke salt
ENDIF
```

```
INPUT color
CASE color of
   red:     PRINT "red"
   green:   PRINT "green"
   blue:    PRINT "blue"
OTHERS
            PRINT "Please enter a valid color"
ENDCASE
```

Fra: https://medium.com/@ngunyimacharia/how-to-write-pseudocode-a-beginners-guide-29956242698

Her et eksempel mere:



```
PROGRAM Prime:
    Read A;
    B = A - 1;
    IsPrime=True;
    WHILE (B != 1)
    DO  IF (A/B gives no remainder)
            THEN IsPrime= False;
        ENDIF;
        B = B - 1;
    ENDWHILE;
    IF (IsPrime == true)
         THEN Print "Prime";
         ELSE Print "Not Prime";
    ENDIF;
END.
```

Fra: https://www.slideshare.net/DamianGordon1/pseudocode-10373156

Pseudokode	Og tilhørende kode

FOR i = 0 to 9
PRINT i
	for(i=0, i<10, i++)
	{
	Serial.println(i);
	}

Side med pseudokode og flowcharts:
http://www.computing.outwood.com/website/NEA/algorithms.html

Se dokument om løkker: http://vthoroe.dk/Teknologi/Arduino/Loops%20mm.pdf

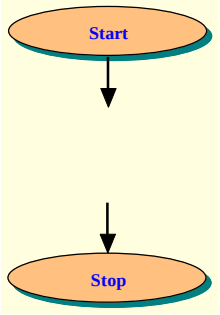
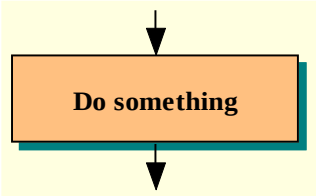
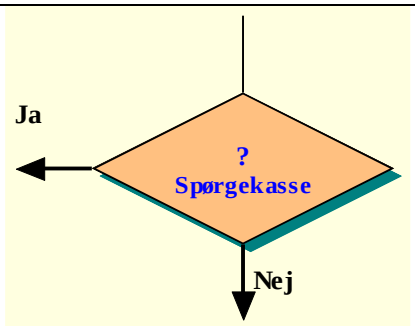
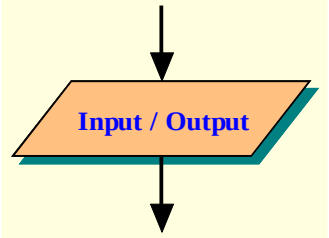
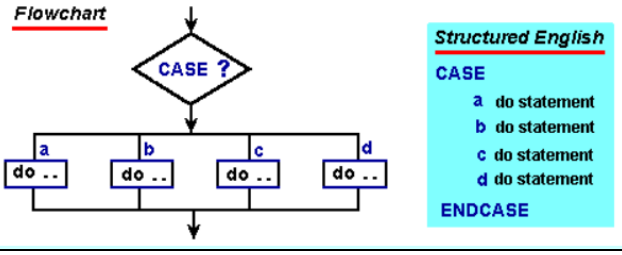
Flowcharts:

Når selve koden skal dokumenteres, skal der laves flowcharts.

Der tegnes forskellige kasser til forskellige handlinger.

Der findes mig bekendt 2 måder, man kan "tegne" et program på. Den "normale, klassiske", og " Nassi Shneidermann ".

Først gennemgås her den "klassiske" måde at lave flowcharts.

Start og slut Elipsen angiver start af programmet eller afslutningen	
Handling: En handling er en normal operation i et program. Noget, der sker. I kassen skrives, hvad der sker på dette sted i programmet!! Det kan fx være en proces, eller en beregning !! En proces har kun 1 indgang, og 1 udgang !!	
Spørgekasse I en spørgekasse foretages et valg. En valg-kasse har 1 indgang, normalt for oven, og 2 udgange, nedad eller til siden. Den ene udgang har betegnelsen Ja, og den anden Nej !! I kassen kan skrives et spørgsmål !!, hvis udfald så afgør hvilken vej, der gås ud af kassen.	
Input / Output I en input-kasse kan der fx ventes på input fra knapper mm.	
Case I en Case-struktur udføres 1 af flere mulige funktioner afhængig af fx en variabels værdi Kilde her:	

Funktion: Kassen illustrerer en subrutine / funktion der er beskrevet med et selvstændigt flowchart.	

Forbindelseslinjer:

Mellem kasserne tegnes forbindelseslinjer, altid i samme retningen som handlingen. Linjerne har pil i den ene retning. Man går altså altid fremad i pilens retning.

Regler for tegning af flowcharts:

Veltegnede flowcharts er lette at læse. Overholdes disse regler, skulle det være muligt, at tegne gode flowcharts, så man let kan aflæse et programforløb!!

Alle flowcharts har én start symbol og evt ét stopsymbol.

Flowet i et flowchart er normalt fra toppen af en side og nedad. Dette kan dog variere med loops, hvor pile fører tilbage til et indgangspunkt.

Der skal altid bruges pile på forbindelseslinjer

Der er kun 1 flowchart pr side (A4)

En side skal have et sidenummer og en titel

Et flowchart på én side bør ikke fortsættes på en anden side!!

Et flowchart bør ikke have mere end ca. 15 symboler, eksklusiv START (og STOP)

Her følger et par eksempler på et flowchart:

Delay-program

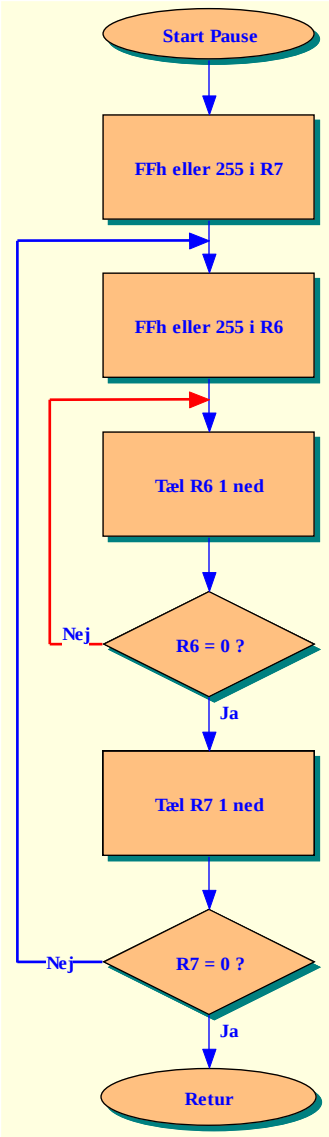
Flowchartet viser, et pauseprogram til en uC.

En processor kan ikke ” lave ingenting”, så hvis man vil have en tid til at gå, kan bare lade den tælle!!

Den indre løkke kører 256 gange, og den ydre løkke kører 256 gange.

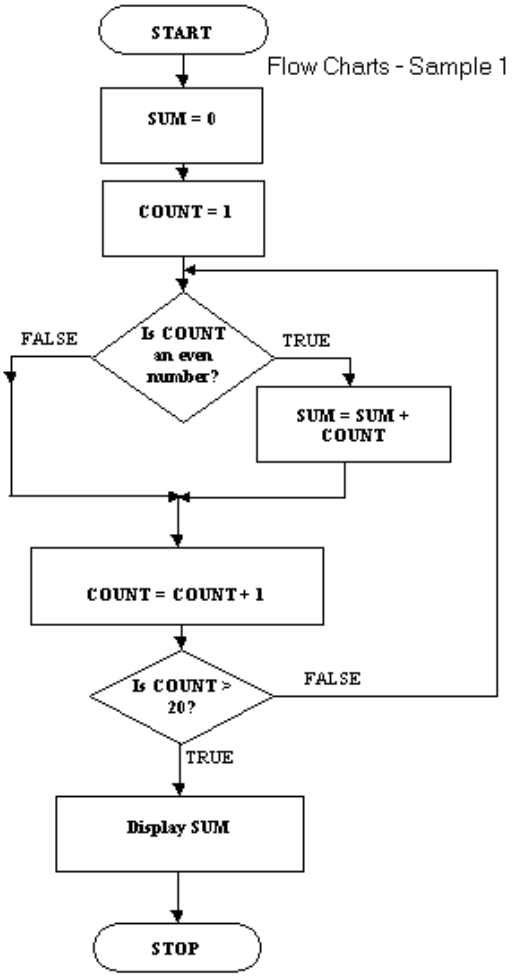
Dvs. 65.536 ”omgange.

Med et 12 MHz krystal tager det ca. ¼ sekund.



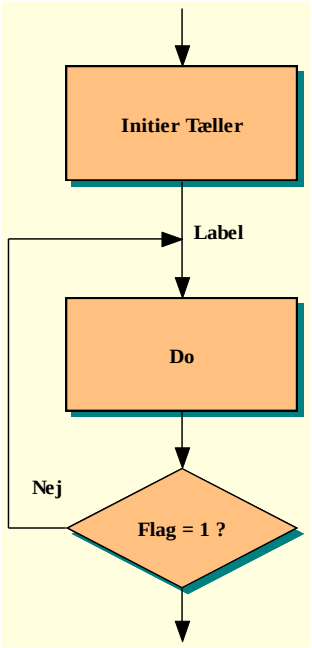
Et eksempel

Her er vist et eksempel.



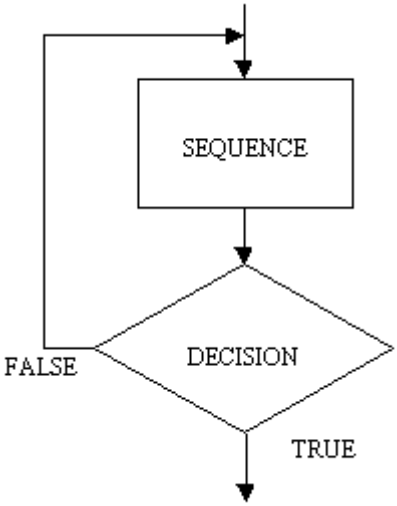
Her er nogle eksempler på loops.

Repeat Until Loop:



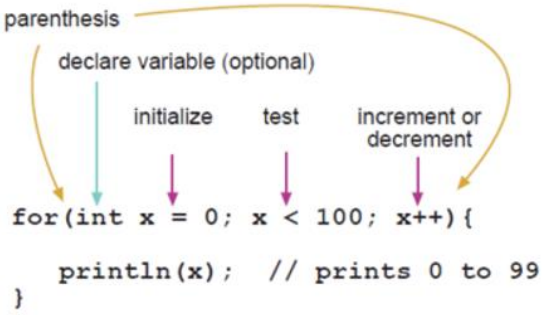
Her et eksempel på en handling, der udføres indtil en given betingelse er opfyldt.

Handlingen udføres mindst 1 gang.



En For-loop kan godt opfattes som en Repeat until-loop.

I en For-loop Udføres " noget " et antal gange - eller indtil

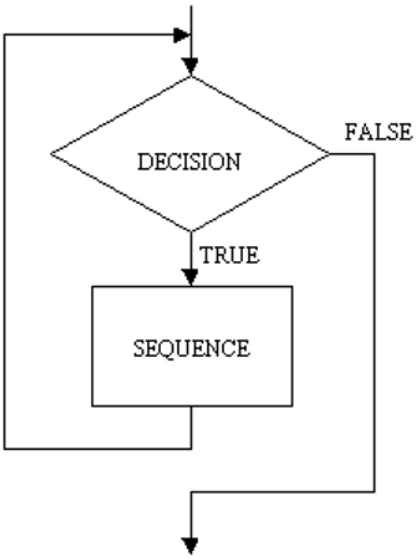



```
// analogWrite

// Værdi fra 0 til 255 !!!

void loop()
{
   int x = 1;
   for (int i = 0; i > -1; i = i + x){
      analogWrite(PWMpin, i);
      if (i == 255) x = -1;             //
switch direction at peak
      delay(10);
   } // Close for
}
```

While Loop

I en While – loop tjekkes, om en handling skal udføres, - før den udføres.

Handlingen behøves således ikke at udføres.

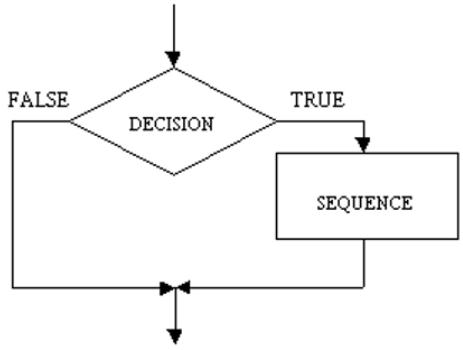


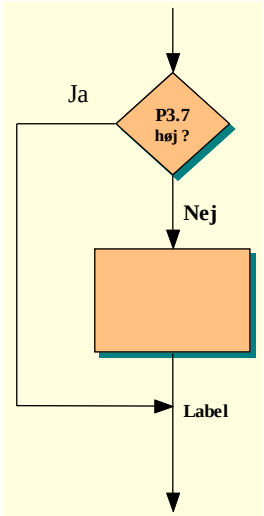

```
var = 0;
while(var < 200){
  // do something repetitive 200 times
  var++;
}  // EndWhile
```

```
void loop()
{

while( digitalRead(5) == 1 ) //while the
button is pressed
    {
    //blink
    digitalWrite(3,HIGH);
    delay(1000);
    digitalWrite(3,LOW);
    delay(1000);
    }  // EndWhile
}
```

If Then

Hvis en betingelse er opfyldt, udføres handlingen, ellers springes handlingen over!!





Tjek, om et ben er høj. Hvis ja, hop udenom til Label

```
//  If - Else:   Eksempel:

if (x > 120){
  digitalWrite(LEDpin1, HIGH);
```

```
  digitalWrite(LEDpin2, HIGH);
}  // EndIf
```

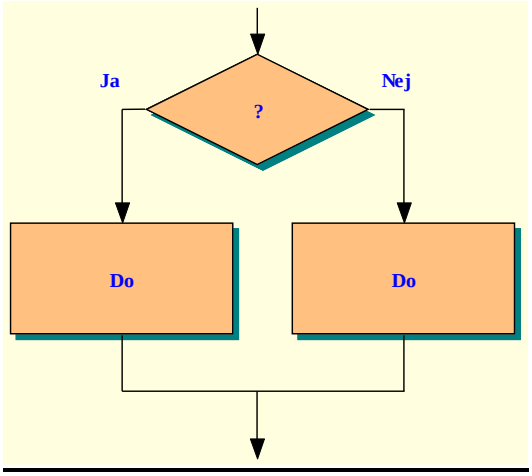
Eks. på assemblerkode:

Jb P3.7, Her
Mov P1, #03h

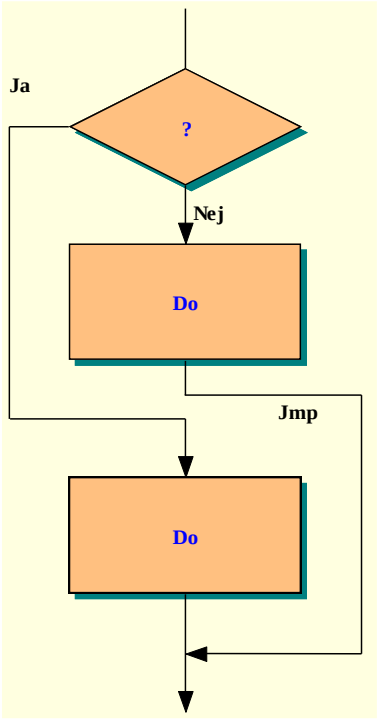
If Then Else Loop

Her udføres enten den ene proces, eller den anden.

```
//  If else

if (pinFiveInput < 500)
{
  // action A
}
else
{
  // action B
}
```



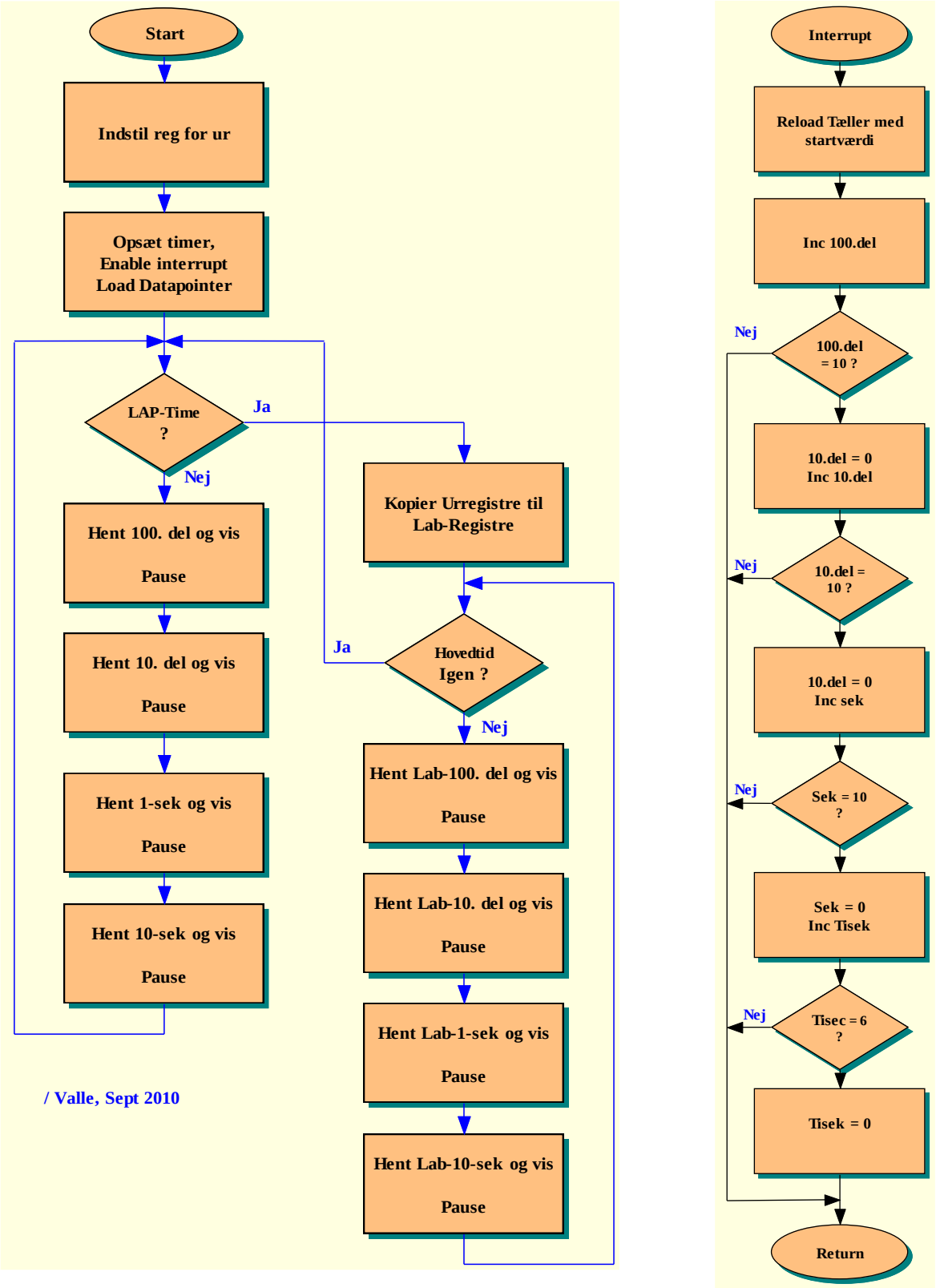
En If – Then – Else struktur er sværere at programmere, men det kan lette, hvis strukturen tegnes som ”1 streng”, som vist til højre.



Stopurs-kit:

Følgende flowchart viser et program-forløb for vores lille stopurskit.

Programmet har faktisk to funktioner. Det ene er hele tiden at skrive tiden i displayet, - og det andet er, at holde øje med, hvor lang tid, der er gået. Denne del er lagt ud i en interrupt-rutine.



Ovenstående er et program, der skal programmeres og overføres til en microcontroller. For at lette programskrivning, bør flowchartet tegnes om til 1 lang søjle, så det bliver lettere at programmere.

Nassi Shneiderman

I forbindelse med søgningen efter en god design-metode og en god dokumentationsmetode til brug ved struktureret programmering er der udviklet flere forskellige typer teknikker. En af dem er Nassi Shneiderman Charts.

Metoden kaldes også struktureret Flowcharts.

N-S Charts er meget anvendelig i struktureret "Top-Down" programmering. Det er faktisk svært at lave ustrukturerede programmer ud fra N-S.

Et Nassi Shneiderman diagram starter med en rektangel på en hel A4 side. Kasserne laves inde i dette rektangel, som vist her efterfølgende:

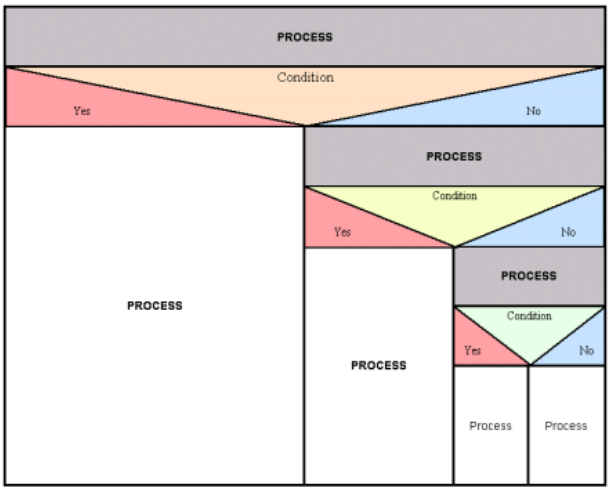
Som tegningen skrider frem, vil de mulige blokstørrelser efterhånden blive meget små.. Men enhver rektangulær blok af et N-S diagram kan fjernes, og flyttes til et nyt – stort – ark. De mærkes så sådan, at det er muligt entydigt at finde frem til næste side.

Hvad er Nassi-Shneiderman Diagrammer?

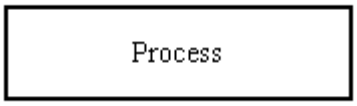
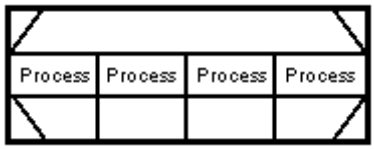
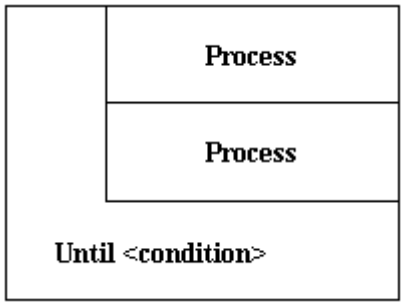
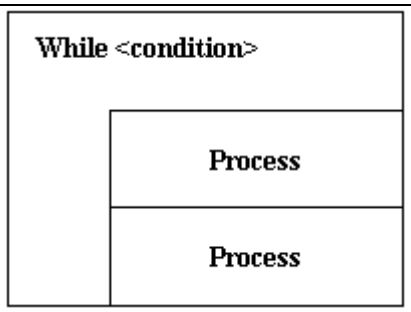
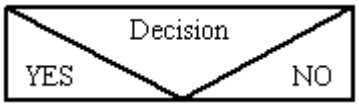
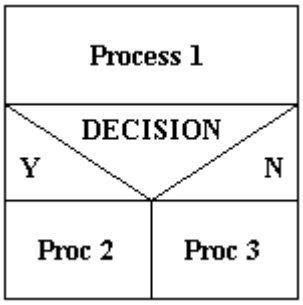
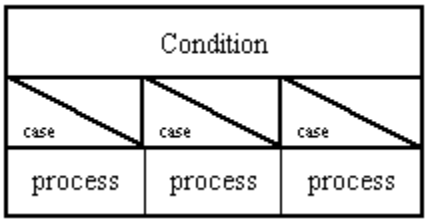
Nassi-Shneiderman (NS) diagrammer er udviklet af Ike Nassi og Ben Shneiderman
Hovedformålet med Nassi-Shneiderman diagrammer er at skabe logiske programstrukturer for et Pc-program.

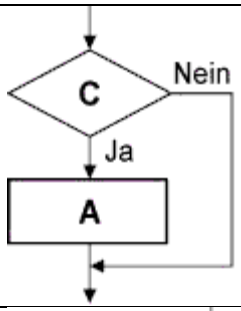
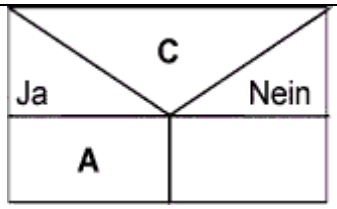
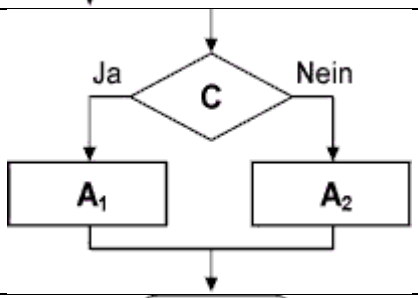
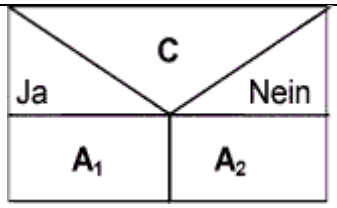
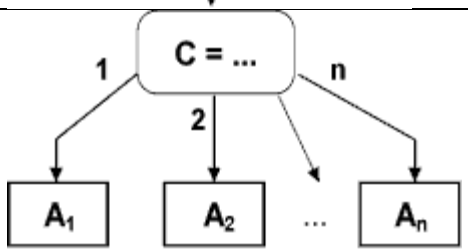
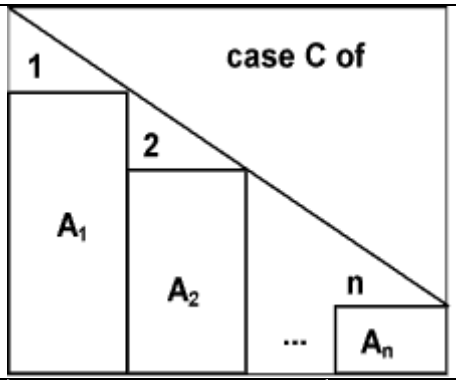
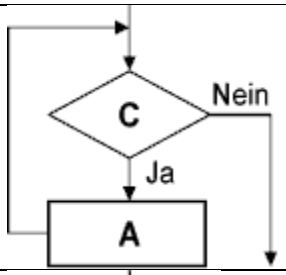
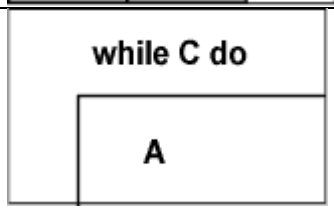
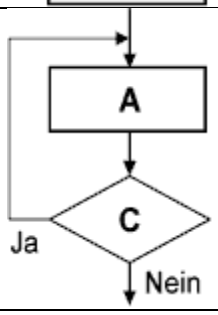
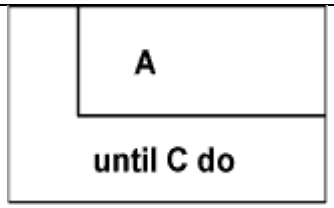
På tysk kaldes NS-diagrammer også for struktur-diagrammer.

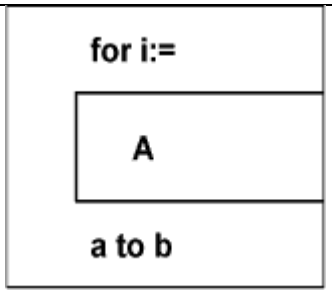
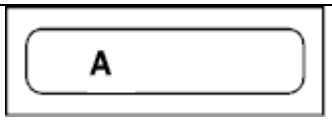
Her er et eksempel på, hvordan man tegner et program-forløb med et Nassi-Shneidermann diagram.



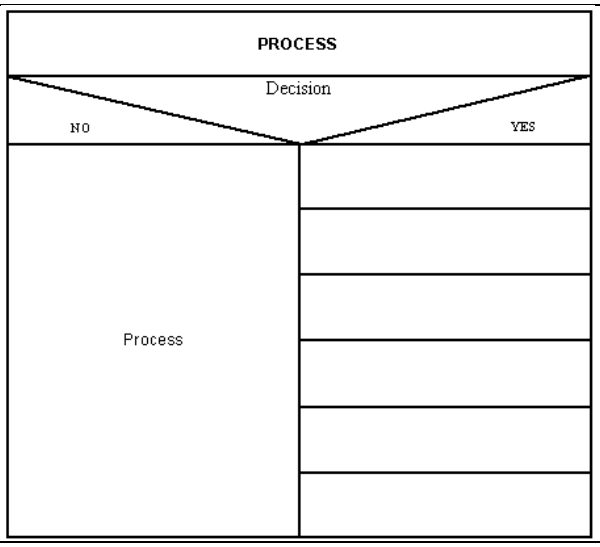
Som i det traditionelle flowchart-tegnemåde, - er der i Nassi Shneidermann-systemet en række forskellige kasser og symboler:

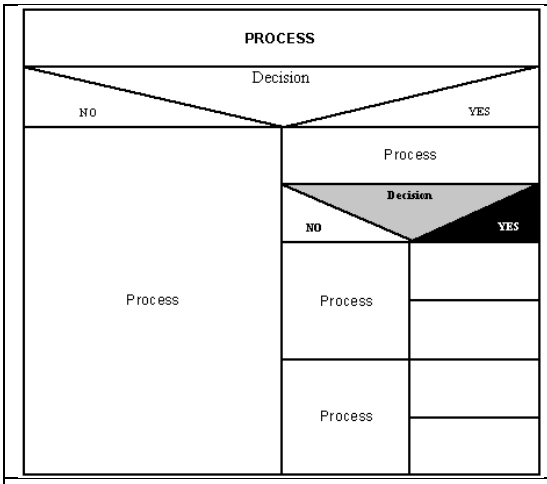
	En Proces En process beskriver en program-del som pseudocode. Dvs. som tekst-beskrivelse ! Man kan lave flere processer efter hinanden.
	Parallelle Processer Placer processesr, som udføres same tid I et trapez. Tegnes som vist !
	Loops Loop notationer bruges, når processer gentages indtil en bestemt tilstand er opnået. Repeat until:
	While
 	Beslutning / Decision Valg-symbolet er et rektangel delt I tre dele som vist. Skriv betingelsen eller spørgsmålet I den øverste trekant, og placer de to mulige udfald I hver deres side af beslutningen. De behøver ikke være af samme størrelse Diagrammet indikerer en algoritme med en process (Proces 1) efterfulgt af et valg. Hver udfald af valget fører til en process. Hvis valget er sand, udføres Proc 2, hvis falsk, proc 3.
	**Case statement** List flere cases ved siden af hinanden i table-format

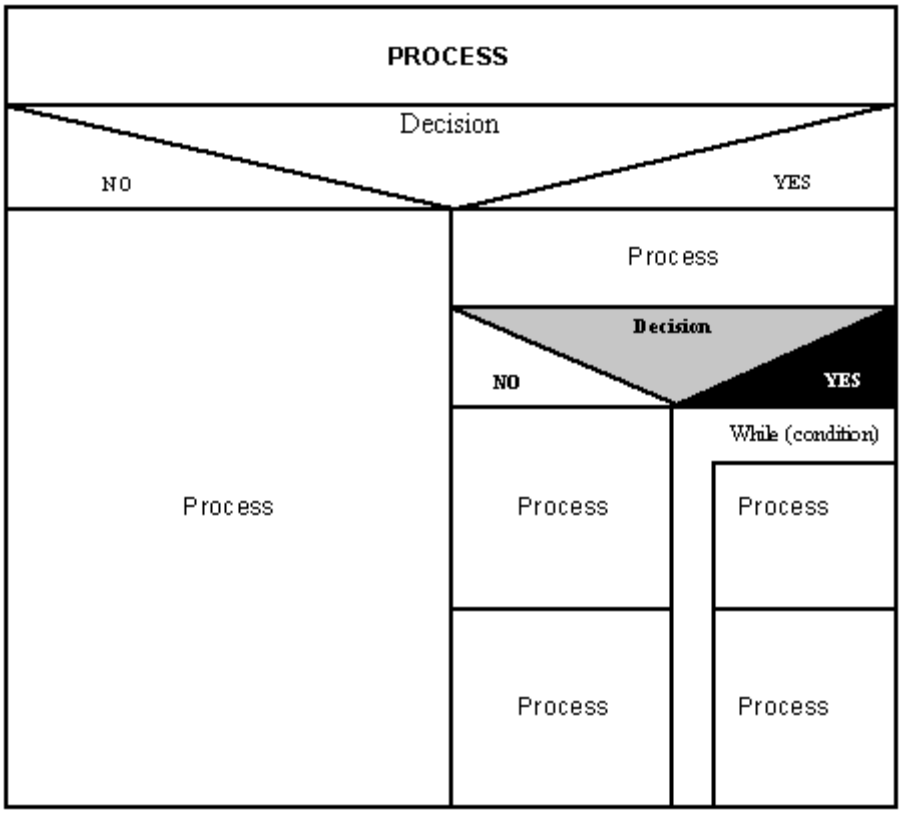
If-Struktur IF-struktur i normal flowchart og i NS.		
If Then Else		
Case-struktur		
Gentagelse (WHILE)		
Gentagelse (REPEAT)		

Som sidste form af gentagelsesstrukturen findes en For-struktur, der udføres et antal gange. Strukturen findes ikke i Flowchart !!		
Underprogram –kald i NSD-fremstilling Hyppigt anvendes i programmer bestemte delprogrammer flere gange. De er derfor smarte at placere i et underprogram, en procedure ! De kaldes så fra hovedprogrammet !'		

Eksempler:

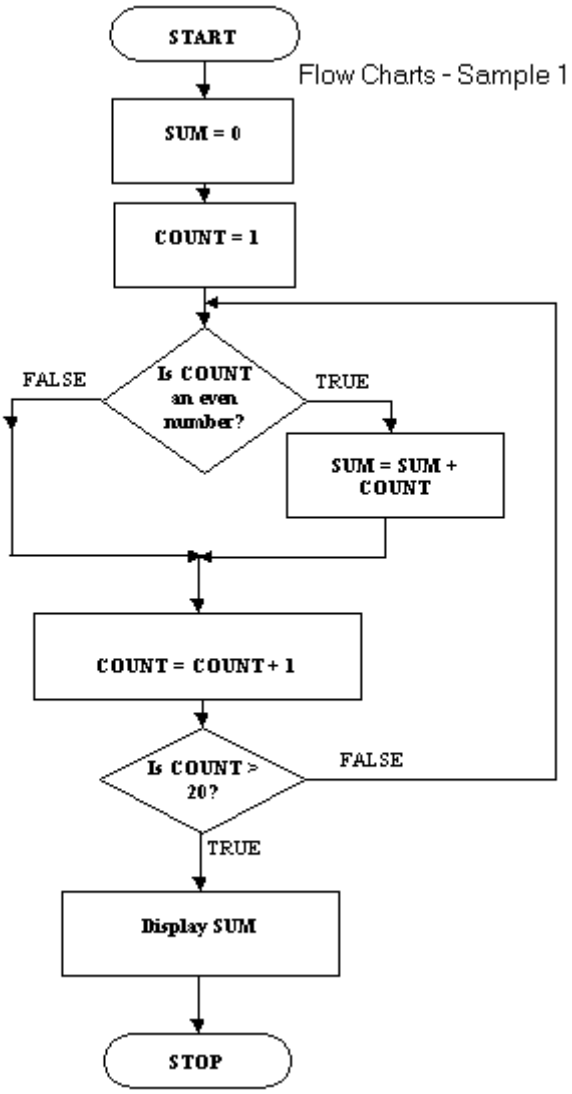
Til højre vises, at der godt kan tegnes flere kasser efter hinanden, for at illustrere flere efter hinanden følgende processer.	


PROCESS
Decision
NO
YES
Process
Decision
NO
YES
Process
Process
Process

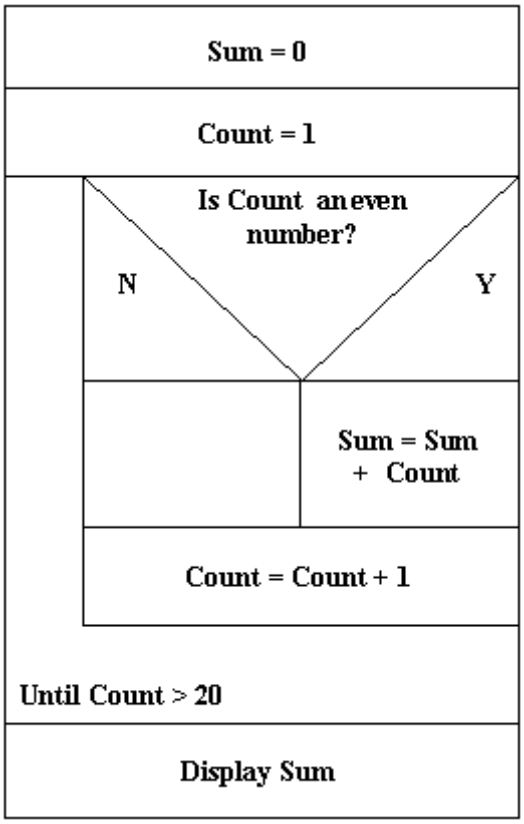

PROCESS
Decision
NO
YES
Process
Decision
NO
YES
While (condition)
Process
Process
Process
Process
Process

Herunder ses en sammenligning mellem Flowchart og NS

Traditionel flowchart



NS Chart:

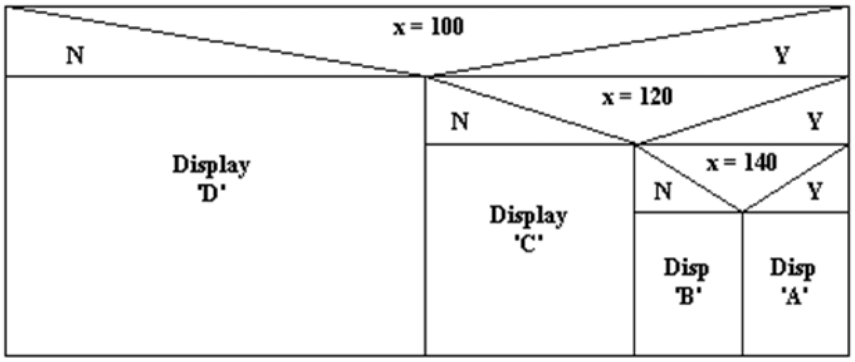


Som I traditional flowchart er sekvens-retningen I NS også fra toppen og nedad.

Her følger et par sammenhørende pseudokoder og deres NS diagram.

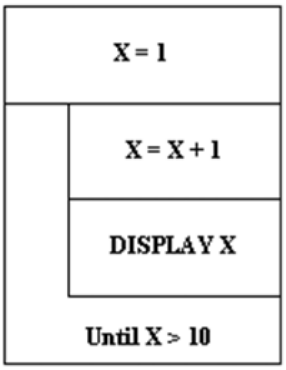
```
IF x = = 100
 THEN IF x == 120
   THEN IF x == 140
          THEN DISPLAY 'A'
          ELSE DISPLAY 'B'
   ELSE DISPLAY 'C'
 ELSE DISPLAY 'D'
ENDIF.
```

Dette er et eksempel på en " multiway selection ".



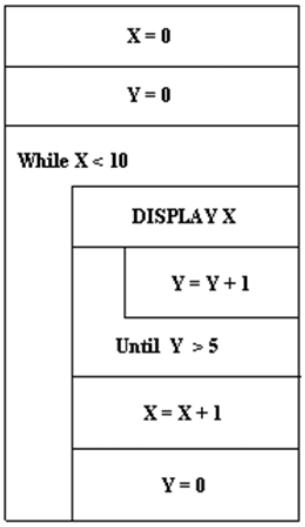

```
X = 1
REPEAT
 X = X + 1
 DISPLAY X
UNTIL X > 10
```

Et eksempel på et repeat loop. Processen X = X + 1 og DISPLAY X vil blive gentaget indtil X er større end 10.

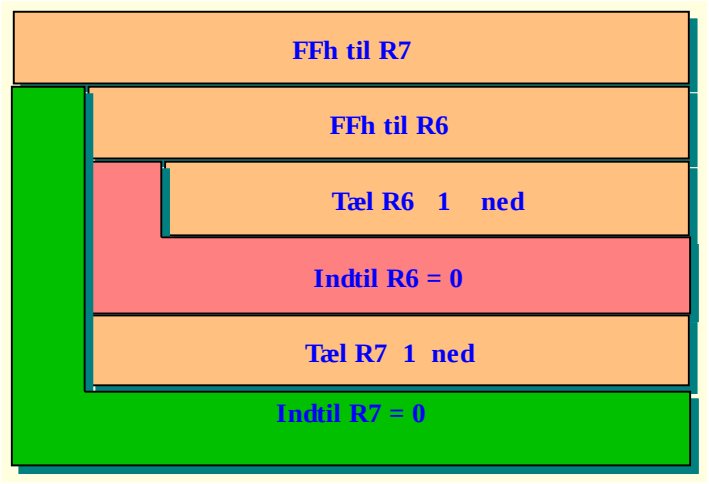



```
X = 0
Y = 0
DOWHILE X < 10
 DISPLAY X
 REPEAT
  Y = Y + 1
 UNTIL Y > 5
 X = X + 1
 Y = 0
END WHILE
```

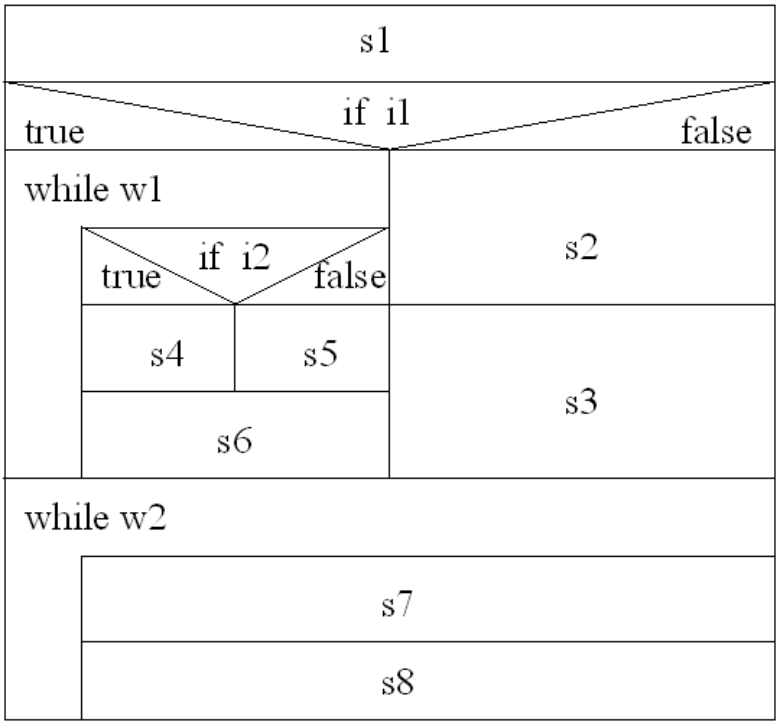
Et eksempel på en indlejret loop, en repeat-loop inden I en while loop.



Pauseprogram tegnet med Nassi Shneidermann



Et Andet eksempel:



Der findes nogle PC-programmer, der kan tegne Nassi-diagrammer ?? http://www.fz-juelich.de/jsc/nassi/#Downloads

Se pdf om Struktograaf. Der findes et program til Nassi Shneidermann

Diagramdesigner 1.23 (Gratis),

edraw, mm.

Se min hjemmeside / elektronik / flowcharts.

https://www.edrawsoft.com/Nassi-Schneiderman.php