# Dette bilag er en del af inspirationsmaterialet til faget [Erhvervsinformatik](https://emu.dk/eud/erhvervsinformatik/digital-myndiggorelse) på emu.dk. Artiklen stiller skarpt på kompetenceområdet ”[Teknologisk handleevne og computationel tankegang](https://emu.dk/eud/erhvervsinformatik/teknologisk-handleevne-og-computationel-tankegang)” og giver konkrete eksempler på elementer, der kan inddrages i undervisningen, fx mønstergenkendelse og abstraktion, flowcharts til understøttelse af algoritmisk tænkning og blokprogrammering. *Skrevet af* Lars Kristian Clausen.

Computationel tankegang

Computationel tankegang handler om analyse, modellering og strukturering af data og dataprocesser [[1]](#footnote-1)

Computationel tankegang er tankegang, der kan bruges til at oversætte den fysiske verden til en verden en computer kan forstå og manipulere.

Noget så, for mennesker, simpelt som at spise suppe, kan være en stor udfordring for en robot/computer. Der er en lang række faktorer, som skal måles og analyseres for at kunne få suppen op af den dybe tallerken, og derefter løfte den op til munden. Hvis du tænker at det kan da ikke være så svært, kan du blot udbygge tanken med at det skal foregå på en cykel.

I dag ligger computere inde med utrolige mængder af databehandlingskraft, og kan derfor let fremstå som overlegne i forhold til mennesker. Men hvis data ikke passer ind i de algoritmer en computer kender til, er det svært eller tilmed umuligt for den at løse opgaven.

For at kunne fortælle en computer (en processor) hvad den skal gøre for at løse udfordringen, skal mennesket kunne kommunikere i et sprog den forstår.

Omsætning af virkelighed til virtuel, er computationel tankegang

For at styre menneskets tankegang i den rigtige retning, er der en enorm mænge værktøjer, der kan hjælpe. I denne vejledning ser vi på simple og let tilgængelige værktøjer, og eksempler på brugen af dem. De værktøjer der er i denne vejledning er ikke de rigtige, eller for den sags skyld forkerte, at bruge. De er eksempler på at kunne komme i gang og derefter er det op til den enkelte at udvide værktøjskassen.

I forbindelse med faget ”Teknologiforståelse” i Folkeskolen, er der lavet en kort video, der forklarer begrebet computationel tankegang <https://www.emu.dk/grundskole/teknologiforstaelse/computationel-tankegang>

## Mønstergenkendelse & Abstraktion

Der stilles en mængde data til rådig. Eleverne skal så genkende mønstre. Dataene kan være af alle mulige slags, og kan relativt let rettes mod forskellige fagområder.

Fx følgende data til Mønstergenkendelse.:

Observationer i en stald/et anlæg:

|  |  |  |  |
| --- | --- | --- | --- |
|  | °C | %Rf |  |
| Dato/klokkeslæt | Inde | Ude | Inde | Ude | Bemærkning |
| 1/1 10:25 | 22,5 | 18,2 | 64,5 | 71,2 |  |
| 2/1 10:22 | 22,6 | 18,5 | 64,3 | 71,0 |  |
| 3/1 07:35 | 19,0 | 15,1 | 72,2 | 76,3 |  |
| 4/1 10:00 | 22,5 | 16,5 | 65,1 | 71,1 |  |
| 5/1 10:12 | 22,4 | 17,5 | 64,2 | 71,2 |  |
| 6/1 10:15 | 22,6 | 17,5 | 64,3 | 71,2 | Dyrene nyser |

Eleverne skal se på datasættet og genkende mønstre. Fx hvordan hænger ind- og udetemperatur sammen?

Abstraktion er den [tankevirksomhed](https://da.wikipedia.org/wiki/Tankevirksomhed), hvor man koncentrerer sig om hovedsagen og ser bort fra mindre væsentlige detaljer, eller hvor man finder fælles egenskaber ved en række begreber eller ting.

Denne søgen efter fællestræk betyder at abstraktion er en modsætning til den konkrete iagttagelse af en enkelt ting eller et enkeltbegreb.[[2]](#footnote-2)

På baggrund af Mønstergenkendelsen laves en Abstraktion, der fokuserer på de væsentligste ting og sorterer det tilfældige fra. Fx kan vi se en variation i temperaturen (inde) som har sammenhæng med luftfugtigheden (inde). Ud over den sammenhæng kan temperaturen 19,0° (inde) måske være grunden til at dyrene nyser 3 dage efter.

Ud fra ovenstående Abstraktion kan man bevæge sig over mod andre faktorer - som ikke registreres endnu - der også kunne have en indvirkning på fx sygdom. Her kunne det være vandforbrug eller fødeoptaget, der varierer efter et genkendeligt mønster.

## Algoritmisk tænkning

Et godt værktøj til at komme videre fra Abstraktionen til Algoritmisk tænkning, er at konstruere et flowchart. Et flowchart (også kaldet et rutediagram eller et flowdiagram) viser forskellige hændelsesmuligheder, fx baseret på input fra brugeren eller en sensor (se Figur 1 & Figur 2). Diagrammet hjælper til at få et overblik over hvordan den senere kode skal struktureres, for at funktionerne virker.

Diagrammet kan fx laves mere tilgængeligt ved at printe sider med de enkelte former af bokse, disse kan lamineres og eleverne kan skrive noter/funktioner på med *white board marker*. Den nemmeste måde at lave de forskellige figurer på, er ved at bruge Microsoft Word®, Microsoft Publisher® eller lignende programmer. I Microsoft® Office-serien vælger man [Indsæt], [Figurer] under ”Rutediagram” findes de forskellige typer af bokse (der findes flere bokse end brugt i dette dokument). Den engelske Wiki-side om flowcharts (<https://en.wikipedia.org/wiki/Flowchart>) indeholder en god gennemgang af de forskellige bokse.

 Et flowchart begynder ét sted ved et aflangt symbol med afrundede ender, der ofte er mærket Start. En pil leder herfra til et symbol for den første aktivitet. Det kan f.eks. være en rombe, der angiver indlæsning eller udlæsning. Teksten i symbolet angiver, hvad der skal læses ind eller ud. En pil leder videre til det næste symbol, der f.eks. kan være et rektangel, der angiver en behandling. Teksten i symbolet angiver igen den ønskede behandling. En pil leder nu f.eks. til en ”rude”, der er et valg. Det eneste symbol hvorfra mere end én pil kan gå ud. Symbolet indeholder spørgsmålet og de forskellige pile ud er mærket med de forskellige svarmuligheder. En pil kan f.eks. føre tilbage til en tidligere behandling, der nu udføres igen. Der kan altid gå flere pile ind i et symbol, men kun én ud – bortset fra ”ruden”. Særlige konnektorsymboler muliggør spring fra den ene ende af diagrammet til den anden, eller til en anden side. Processen afsluttes når man når et symbol med teksten Slut. Det er magen til Start-symbolet. [[3]](#footnote-3)

Figur 2 er en omsætning af det at beregne arealet (A) af en cirkel. I faget regning vil man sjældent opleve at en opgave i at beregne arealet af en cirkel, vil tage udgangspunkt i at radius af cirklen er 0. Men i et program til en computer, vil det være meget hensigtsmæssigt at kontrollere radius forud for en beregning.

Hvis man ikke benytter sig af principperne om computationel tankegang, kunne man vælge ikke at tage højde for det – for i den virkelige verden lykkedes det ikke at få et brugbart resultat, men i den virtuelle verden, kan det fastfryse et program.

I Figur 1 er det en styring af temperatur. Brugeren angiver minimums- og maksimumstemperatur, hvorefter algoritmen beregner hvilken af tre processer der skal eksekveres. Rutediagrammet er fejlbehæftet, da det forventer to brugerindput hver gang der tændes for varmen eller kølingen. I rutediagrammet er der også indlagt mulighed for at ændre temperaturene eller afslutte programmet – det er dog kun muligt hvis der er tændt for varme eller køling.

Ved at lade eleverne gennemkøre de to flowcharts, opdager de forhåbentligt fejlene.



Figur 1 Eksempel med flere hændelser (fejlbehæftet!)

Figur 2 Eksempel med få hændelser

## Datatyper

Se dokumenterne under overskriften ”værktøjer og ressourcer til underviseren” på emu-siden [Teknologisk handleevne og computationel tankegang](https://emu.dk/eud/erhvervsinformatik/teknologisk-handleevne-og-computationel-tankegang)

## Programmering

Nu er vi nået til del af processen, der måske kan virke lidt skræmmende, men når forarbejdet er lavet er programmeringen blot det næste trin i processen.

Programmering er rigtigt mange forskellige ting – mange tænker den tanke at det er bunker af linjer med uforståeligt tekst. Men det behøver det på ingen måde at være. Hvis man tænker tilbage på rutediagrammet, så er det faktisk en form for programmering. Formen hedder pseudokode. En meget simpel form for kodning, der ikke tager hensyn til et programmeringssprog.

Der findes et væld af programmeringssprog, der strækker sig fra meget simple til meget komplekse. Ud over pseudokode, ser vi på *blokprogrammering*, hvilket er et visuelt programmeringssprog. Det smarte ved blokprogrammering, er at man bygger sit program op af blokke, der så (automatisk) oversættes til kodelinjer. Det giver eleverne mulighed for nemt at bygge programmer op, og efterfølgende se hvordan koden ser ud. De elever, der er visuelt orienterede kan bliver ved med at bruge blokprogrammering og de elever, der er mere matematisk rettede, kan skifte over til det bagvedliggende kodesprog.

På siden <https://blockly-games.appspot.com/> er der en række øvelser i at analysere sig frem til en forståelse for hvordan en computer ”tænker” og giver et spædt indblik i blokprogrammering. Øvelserne kræver meget lidt introduktion, er selvrettende og kan løses i forskellige tempi.

## Blokprogrammering

Der findes en del forskellige platforme, der tilbyder blokprogrammering. Både løsninger, der skal installeres og vedligeholdes af lokalt, samt onlineløsninger, der er designet til undervisning.

### Scratch

Scratch er udviklet til undervisning, og er en onlineversion. Målgruppen er børn, men kan sagtens bruges til introduktion til programmering.

Siden drives af Massachusetts Institute of Technology (bedre kendt som MIT). Siden er oversat til dansk – vejledningssiderne til undervisere er dog kun delvist oversat.

Siden lægger op til at programmere historier eller spil, som kan deles. Der findes et væld af allerede oprettede programmer, der kan inspirere.

<https://scratch.mit.edu/educators/faq>

### Blockly

Blockly er udviklet til programmering, og er en version der skal installeres i et webmiljø. Målgruppen er udviklere (både begyndere og erfarne).

Siden drives af Google.

<https://developers.google.com/blockly/>

### App Inventor

App Inventor er udviklet til programmering af apps til mobiltelefoner/tablets (kun Android), og er en onlineversion.

Siden drives af Massachusetts Institute of Technology (MIT)

<http://appinventor.mit.edu/explore/ai2/beginner-videos.html>

### Node-RED

Node-RED er udviklet til programmering, og er en version der skal installeres i et webmiljø. Målgruppen er udviklere (både begyndere og erfarne). Det installeres nemt på en Raspberry Pi.

Programmet er udviklet af IBM, og bruges i mange forskellige systemer.

<https://nodered.org/>

1. https://www.emu.dk/grundskole/teknologiforstaelse/computationel-tankegang [↑](#footnote-ref-1)
2. https://da.wikipedia.org/wiki/Abstraktion [↑](#footnote-ref-2)
3. https://da.wikipedia.org/wiki/Flowchart [↑](#footnote-ref-3)