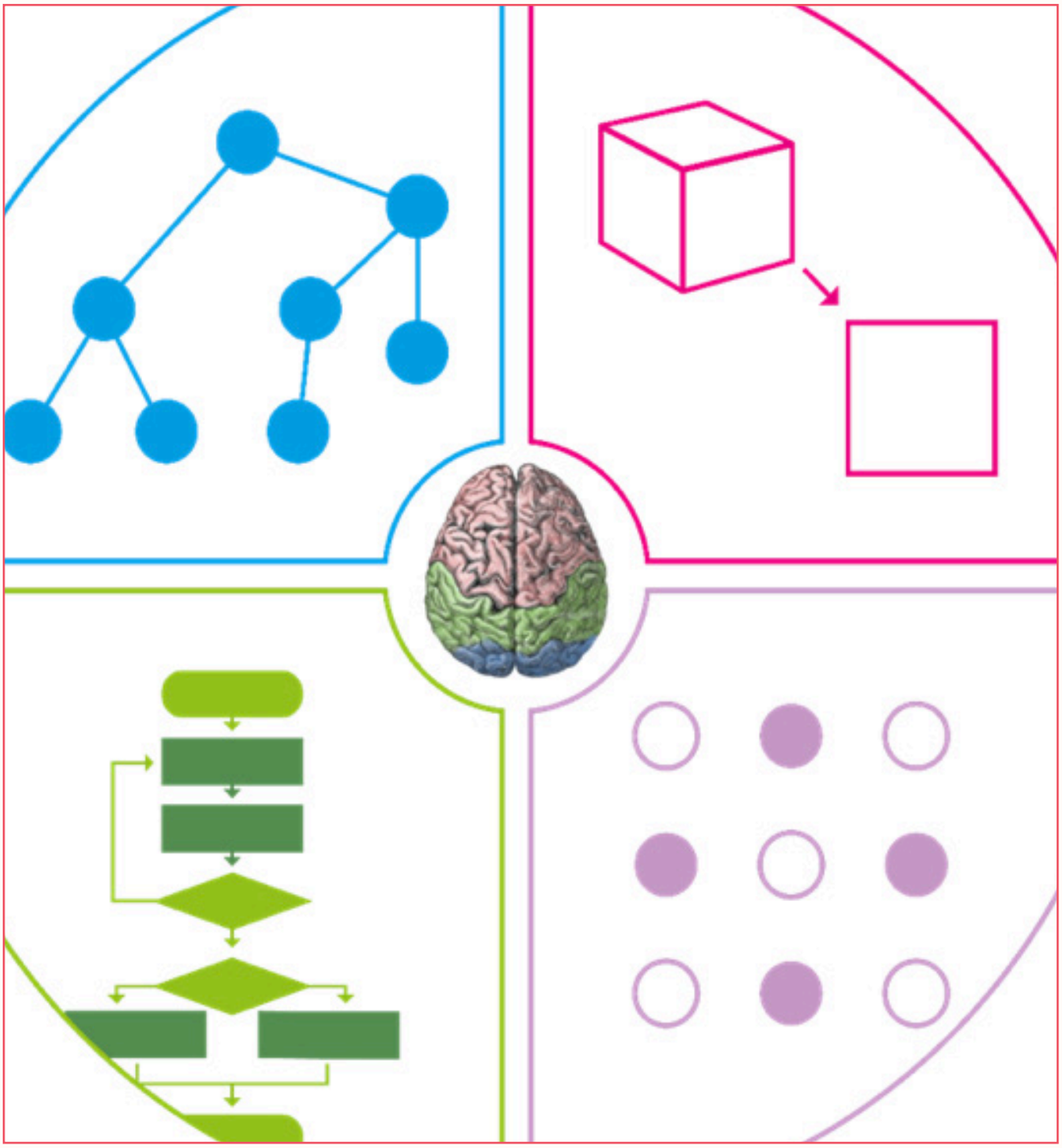
COMPUTATIONEL TÆNKNING



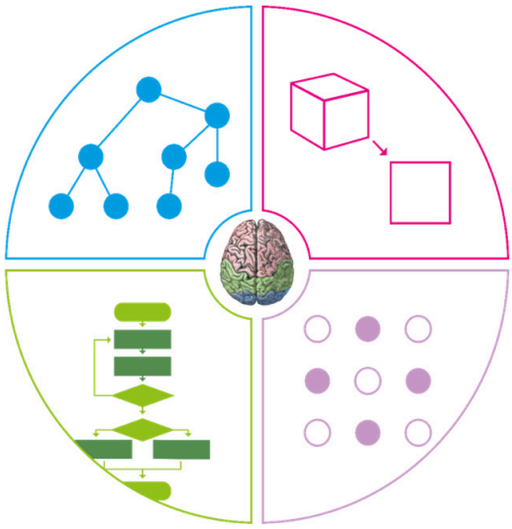
GRUNDEJERNES INVESTERINGSFOND
CIU Center for it i undervisningen
TEKNOLOGISK INSTITUT

Computationel tænkning

Computationel tænkning er nogle teknikker, som gør det nemmere for os at omsætte et behov eller en ide til en IT-løsning. Metoden omfatter 4 hovedteknikker:

Dekomposition

Abstraktion og generalisering



Algoritme design

Mønster genkendelse

DEKOMPOSITION

Dekomposition er nedbrydning af en opgave i delopgaver. Ved at nedbryde opgaven i små overskuelige delopgaver, kan vi på en struktureret måde komme rundt omkring alle elementerne i løsningen og få diskuteret alle detaljer – uden at miste overblikket. Til dette kan man benytte teknikker såsom struktur-diagrammer og flowdiagrammer.

ABSTRAKTION

Abstraktion og generalisering handler om at fokusere på de væsentlige elementer. Hvis man skal vide alt om binære talsystemer, programmering for at lave en prototype af et regneark, så kommer man aldrig i gang. Heldigvis er der lavet generelle løsninger, der gør, at udviklerne kan abstrahere fra dette, så de kun skal koncentrere sig om designet. Gennem abstrakt tænkning kan vi fjerne alt det unødvendige (eller bare det vi ikke behøver at løse lige her og nu), og så bør vi ellers forsøge at lave løsningerne generelle, så samme løsning kan bruges flere steder. I forbindelse med programmering laver vi abstraktion gennem brug af funktioner.

MØNSTER GENKENDELSE

Mønster genkendelse er når ting går igen. Er der noget computeren er god til, så er det at lave gentagelser (løkker). Derfor skal vi analysere problematikken og kigge efter gentagelser, for derefter at skrive koden ved brug af gentagelser.

ALGORITME DESIGN

En algoritme er en opskrift på, hvorledes man løser et konkret problem. Det kan eksempelvis dokumenteres i et flowdiagram, hvor løsningen beskrives trin for trin. Det kan også dokumenteres ved brug af formler og regler. (bemærk at der i små opgaver er en del sammenfald mellem dekomposition og algoritme design).

De fire teknikker er nogle metoder, vi kan bruge, når vi skal angribe et problem (gælder ikke kun IT systemer). Der er ikke nogen bestemt rækkefølge de skal anvendes i, men ofte er det en god ide at starte med dekomposition.

Udvikling af et regneark til spånteknisk beregning

Inspiration og materialer af Lars Skjærbæk UCH på emu.dk, danner grundlag for dette materiale:

emu.dk/eud/erhvervsinformatik/teknologisk-handleevne-og-computationel-tankegang/udvikling-af-kasseapp

Eleverne skal igennem forløbet udvikle en prototype på et regneark, der kan beregne spåntekniske data (et regneark, der kan bruges ved opstilling og kørsel med træbearbejdnings maskiner). Dette omfatter foruden computationel tænkning også design af brugergrænseflader, basal programmering, anvendelse af formler og betingelser, samt afprøvning og fejlretning.

Værktøjer til programmering i erhvervsinformatik

Se Lars Skjærbæks video på emu.dk med gode råd til undervisning i softwareudvikling og programmering.

Målgruppe: Erhvervsinformatik på F niveau · **Tidsforbrug:** 12 timer inkl. evaluering

OM FORLØBET

Formålet med materialet er at lære eleverne, hvorledes vi gennem computationel tænkning kan udvikle en prototype på et regneark til spånteknisk beregning.

I forløbet forklares begreberne omkring computationel tænkning med eksempler og øvelser. Derefter går vi i gang med at lære eleven om basal blokprogrammering via www.code.org. Prototypen udvikles i Microsoft Excel.

Til understøttelse af dette er der lavet nogle eksempler, som kan guide såvel lærere som elever igennem udviklingen. Forløbet omfatter også teori og øvelser i udvikling af flowdiagramer, indsamling af inddata, beregninger og betingelser, der giver de ønskede uddata , samt afprøvning og fejlretning.

Der arbejdes således med følgende faglige mål:

6. har kendskab til computationel tankegang til løsning af problemstillinger i en erhvervsfaglig kontekst
7. kan via simpel programmering genkende basale strukturer i et programmeringssprog
8. har kendskab til datatyper og registrering af data

Følgende mål fra kernestoffet dækkes:

- Programmering: funktioner, variable, sekvenser, løkker og forgreninger
- Afprøvning og fejlretning
- Omsætning af problemløsninger til algoritmer gennem dekomposition, abstraktion, mønstre og generaliseringer
- Datatyper

PLANLÆGNING/OVERVEJELSER

Forløbet er bygget op som et projekt med indlagt teori og øvelser. Eleverne skal arbejde sammen i par. Dels er det vigtigt, at alle får "hands on" på programmeringsdelen, men det er også vigtigt, at man hele tiden har nogle at sparre med. Dan grupperne i starten af projektet.
Det beskrevne forløb er bare en af mange muligheder. Regnearket til beregning af spåntekniske data er et bevidst valg for at gøre eksemplet så simpelt som muligt. Hvis du synes, den er for nem, kan du udvide den med mere avanceret funktionalitet, eller du kan definere en helt anden opgave. Teorien kan stadig benyttes.

Forløbets opbygning

Forløbet strækker sig over 12 lektioner

1. INTRODUKTION. HVAD ER COMPUTATIONEL TÆNKNING (1 TIME):

- Underviseren gennemgår teori omkring computationel tænkning.
- Kombiner med Øvelser til computationel tænkning.
- Slides med kombineret teori, eksempler og øvelser.

2. BRUG AF STRUKTURDIAGRAMMER OG FLOWDIAGRAMMER (1 TIME):

- Gennemgå teori og øvelser.
- Slides med teori og øvelse.

3. INTRODUKTION TIL PROGRAMMERING (1 TIME):

- Til dette benyttes www.code.org.

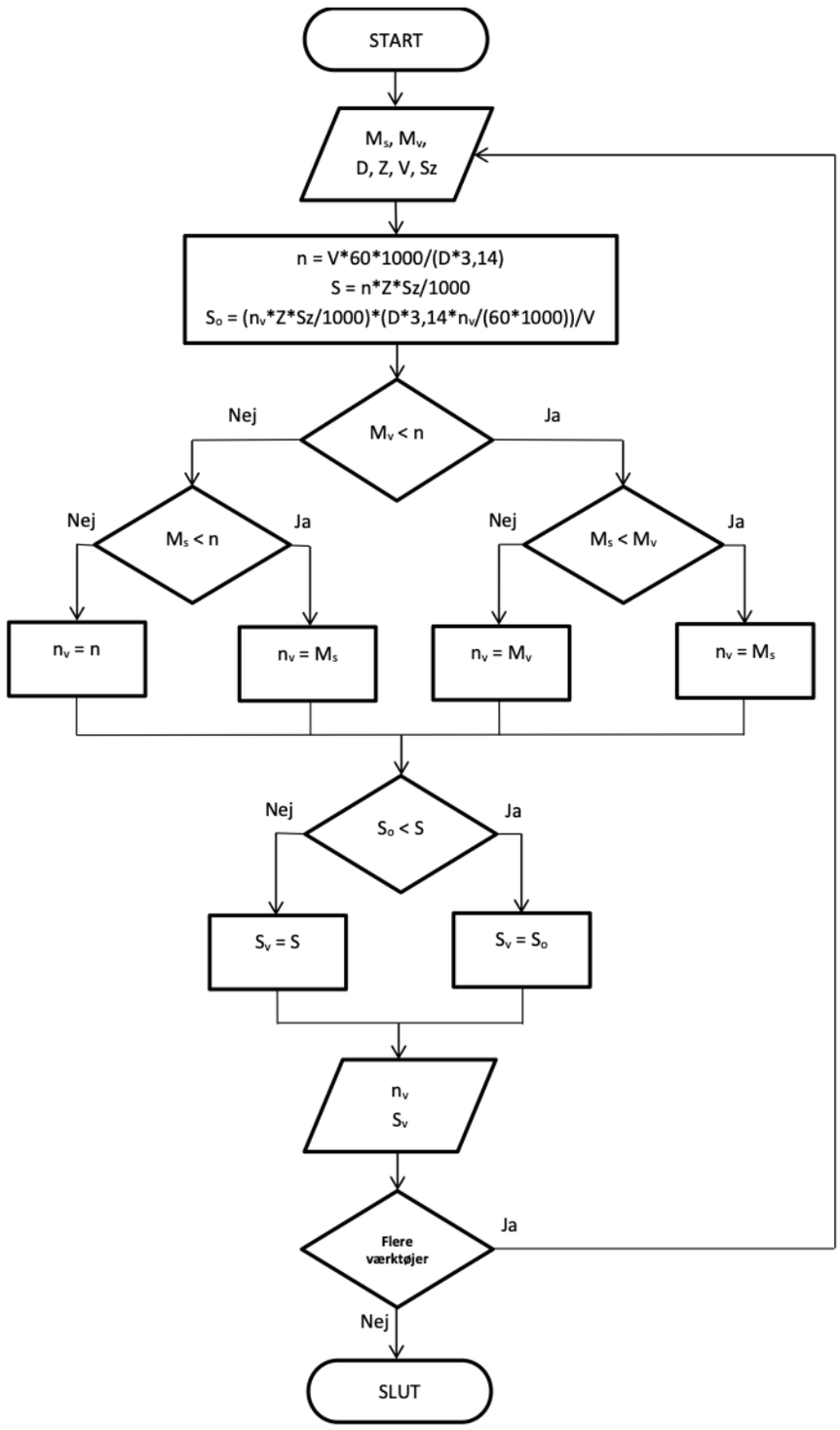
Se video: Start kodning på code

Der etableres et kursus (Klassisk labyrint) og nøglen sendes til eleverne. Disse instrueres i, hvorledes de logger på code.org, og hvor de indtaster nøglen. Læreren kan således følge op på elevernes progression.

Se video: Opfølgningsværktøjer til code

Efter udførelsen af opgaven (eller i starten af den følgende time) forklares de basale kodningselementer: Variable, sekvenser, løkker, forgreninger, funktioner.

Spånteknisk beregning – Flowdiagram



	A	B
1	**Spånteknisk beregning**	
2		
3	**Vejledende hastighed** (uddata)	
4	n_v = Vejledende spindel omdrejninger pr. min.	=HVIS(B9<B16;HVIS(B8<B9;B8;B9);HVIS(B8<B16;B8;B16))
5	S_v = Vejledende fremføringshastighed i m. pr. min.	=HVIS(B18<B17;B18;B17)
6		
7	**Oplysninger** (inddata)	
8	M_s = Maksimum spindelomdrejninger pr. min.	24000
9	M_v = Maksimum værktøjsomdrejninger pr. min.	22000
10	D = Værktøjsdiameter i mm.	4
11	Z = Antal skær	2
12	V = Ideel skærehastighed i m. pr. sek.	50
13	Sz = Fremføring pr. skær i mm.	0,25
14		
15	**Optimalt** (beregnet)	
16	n = Spindel omdrejninger pr. min.	=B12*60*1000/(B10*3,14)
17	S = Fremføringshastighed i m. pr. min.	=B16*B11*B13/1000
18	S_o = Optimeret fremføringshastighed i m. pr. min.	=(B4*B11*B13/1000)*(B10*3,14*B4/(60*1000))/B12

Opgave

Fremstil en prototype på et regneark, der kan beregne spåntekniske data (et regneark, der kan bruges ved opstilling og kørsel med træbearbejdnings maskiner). Dette omfatter foruden computationel tænkning også design af brugergrænseflader, basal programmering, anvendelse af formler og betingelser, samt afprøvning og fejlretning.

Det kan eksempelvis være automatisering af formler til beregning af:

- Vejledende spindel omdrejninger (tag højde for max. hastighed på værktøj og spindel)
- Vejledende fremføringshastighed (tag højde for værktøjer med diametre mindre end 40 mm)

Formler og tilhørende forklaringer kan findes i Maskin-snedkerbogen, i afsnittet Spåntekniske beregninger.

Fremlæg fremgangsmåde og resultat for klassen.