Kanon i forbindelse med undervisning i programmering

Tildelinger

Tildelinger anvendes, når variable får tildelt en værdi eller ændret værdien, og tildelinger sikrer dermed dynamikken i programmer. Med hertil hører vigtige begreber som variable, data, udtryk, datatyper, objekter mm, men ingen af disse ville have en signifikant rolle uden tildelinger

Kontrolstrukturer

Strukturen i alle programmer er bygget op ved hjælp af sekvenser, forgreninger og gentagelser, hvilket er en vigtig erkendelse, når man skal lære at programmere

Datastrukturer

Organisering af data i passende strukturer til de givne problemstillinger. Eksempler på datastrukturer er lister, køer, stakke, træer og grafer

Modularisering

Komplekse programmer opdeles i mindre, selvstændige enheder kaldet moduler, der kan udvikles, testes og vedligeholdes separat. Hvert modul udgør en specifik funktion og kommunikerer med andre moduler gennem veldefinerede grænseflader

Algoritmisk tankegang

Med algoritmisk tankegang udvikles pseudokoder, der er universelle og springbræt til udvikling af kode i et givent programmeringssprog. Unplugged programmering styrker kompetencen til at udvikle og forstå pseudokode

Abstraktion

Når man programmerer, handler det om at skabe en model af virkeligheden med fokus på det centrale. Abstraktion drejer sig præcist om at fokusere på den givne problemstilling og dermed frasortere alt andet

Mønstergenkendelse

Vigtigt at kunne genkende forskellige mønstre, når man programmerer. Selvom problemstillinger kan være nok så forskellige, er der alligevel mulighed for, at de samme algoritmeskabeloner kan bruges, hvilket gælder både i det enkelte programmeringssprog, men også på tværs af forskellige sprog og pseudokode

Stepwise improvement

Den iterative arbejdsmetode, hvor man trinvist forbedrer sit program i forbindelse med udvidelse, forfining og omstrukturering. Metoden er med til at sikre, at man kommer i mål med noget, der fungerer

Fejlfinding

Det er en illusion, at man altid laver programmer uden fejl. Derfor er det vigtigt med systematiske metoder til debugging

Programmeringslag

Forskellige arkitekturer som klient/server og trelagsarkitektur med præsentation, kode og data. Forskellige lag af computersprog fra maskinkode til højniveausprog og sammenhængen mellem lagene, kompilering, fortolkning mm

Kanon i forbindelse med undervisning i programmering

Intensionen med denne kanon er at beskrive og kategorisere de vigtigste ingredienser i forbindelse med undervisning i programmering. Der er tale om en universel beskrivelse, som skal kunne bruges af både undervisere og elever, hvor beskrivelsen er uafhængig af valg af programmeringssprog. Desuden skal den universelle tilgang sikre, at den kan bruges i alle situationer, hvor der undervises i programmering, og helt konkret betyder det, at denne kanon kan bruges af

- klasser i indskolingen med *Teknologiforståelse*
- gymnasieklasser i *Informatik* og *Programmering*
- hold på Masteruddannelsen i Informatikundervisning i kurset *Programmering*
- hold på læreruddannelsen i *Teknologiforståelse*
- hold på Datalogi i *Introduktion til programmering*
- ...

Det vil sige alle steder med undervisning i programmering – specielt begynderundervisning

I alle situationer vil der kunne udvikles en specifik Rubric, hvor de SOLO-taksonomiske niveauer kan beskrives, og som løbende i undervisningen kan bruges som formativ evaluering

	Meget svag	Svag	OK	Stærk	Meget stærk
Tildelinger					
Kontrolstrukturer					
Datastrukturer					
Modularisering					
Algoritmisk tankegang					
Abstraktion					
Mønstergenkendelse					
Stepwise improvement					
Fejlfinding					
Programmeringslag					