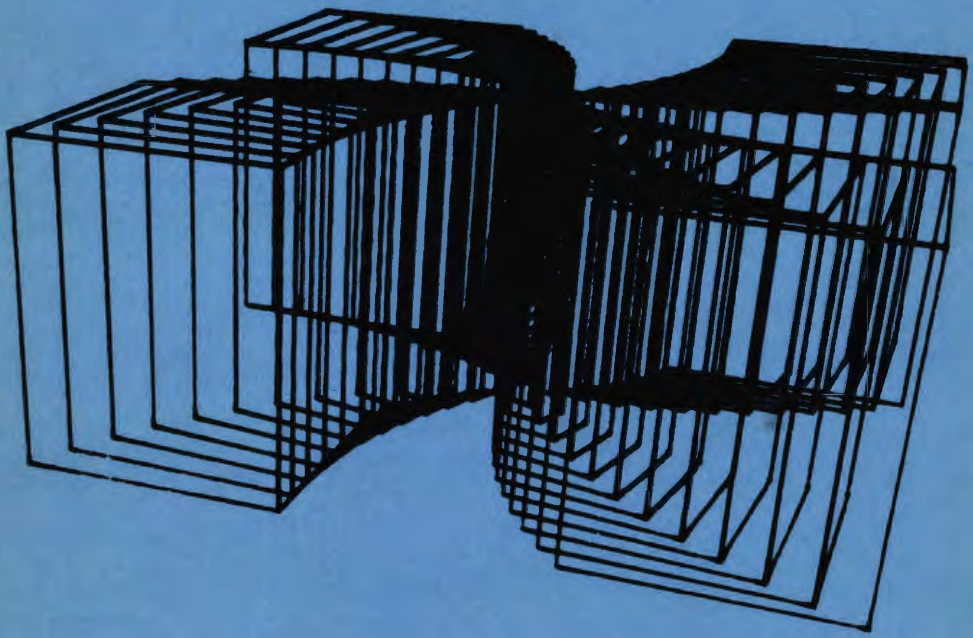
Dats

Datalære ved de tekniske skoler



Oktober 1984 - nr. 3 og 4

Til læserne.

Hermed fremsendes et nyt nummer af DATS.

I indeværende skoleår forventes yderligere 2 numre, nemlig 15. februar 1985 og 15. maj 1985.

Samlet pris for disse tre numre, incl. forsendelse er kr. 45,00, som venligst bedes indbetalt inden 15. november 1984 på gironummer 7 00 50 08, hvorefter fremsendelse af de næste 2 numre vil ske.

Med venlig hilsen

Teknisk
Informatik
Center
Aarhus teknisk Skole

Redaktion:
Ole Karmark

Ekspedition:

DaTS/TICA
Aarhus tekniske Skole
Halmstadgade 6
8200 Århus N

Tegning på forsiden
er fremstillet ved
hjælp af Monster.

INDHOLD

Forord

Det er lykkedes endnu engang at udsende datalærebladet for de tekniske skoler!

Den store mængde stof, vi har modtaget, har gjort, at vi har valgt at udsende Dats som et dobbelt nummer.

Dats indledes med en spændende og lærerig artikel om emnet: CAD/CAM. Artiklen giver en grundig introduktion til dette omfattende og aktuelle område af datateknologien. Den efterfølgende artikel drejer sig om, hvordan SKF82-programmet er designet og om hvorledes, det er tænkt anvendt. Et interessant bidrag til diskussionen om programstrukturer. Problemstillingen om programdesign videreføres i artiklen om dokumentation af programmer.

Enhver undervisers tilbagevendende opgave er planlægning og tilrettelæggelse af sin undervisning. Vi er glade for at kunne bringe et undervisnings- og materialeforslag til hvorledes man kan gennemføre den indledende undervisning i datalære på efg-basisår. Materialet kan rekvireres i A4-format hos forfatteren.

Det afsluttende indlæg handler om et projektforslag om styring af en firtaktsmotor.

Dats rundes også denne gang af med nogle boganmeldelser, bl.a. anmeldes Bogen om MONSTER.

Ved udsendelse af næste nummer af Dats overvejer vi at få etableret en egentlig abonnementsordning. Men herom mere i næste nummer.

God læselyst.

Redaktionen

CAD/CAM

Hvad er det?
Hvordan gør man?
Hvad får man ud af det?

Af Ulrich Lysdal Jensen.

Resume

Denne artikel er den første i en række på 2 eller 3, som skal behandle det meget aktuelle emne Computer Aided Design (CAD. I denne første artikel forklares først, hvad CAD er samt delvis gennem et kort historisk tilbageblik, hvorfor CAD er så aktuelt i dag. Dernæst beskrives, hvorledes en datamat kan indgå i designarbejdet. Efter en kort diskussion af de grundlæggende krav, der må stilles til det udstyr, der indgår i et CAD-system, afsluttes artiklen med en beskrivelse af, hvorledes en velgennemtænkt indførelse af CAD i en virksomhed kan bidrage til bedre dokumentations- og informationsmateriale overalt i virksomheden. I næste artikel vil emnet CAD-tegneprogrammer blive behandlet.

Bogstavforkortelsen CAD/CAM er en af de evindelige edb-forkortelser, man i vore dage vil støde på, blot man har den mindste berøring med databehandlingens verden.

Lad os straks opklare betydningen af denne edb-jargon:

CAD = Computer Aided Design
CAM = Aided Manufacturing

eller på godt dansk: Hvorledes anvender man en datamat i henholdsvis design- og produktionsfasen for et produkt?

Der findes også danske (og andre engelske) betegnelser for disse meget interessante discipliner.

DAK = DatamatAssisteret Konstruktion
DAP = DatamatAssisteret Produktion

CAE = Computer Aided Engeneering (CAD + CAM)

Hvorfor CAD/CAM netop nu?

I første sætning blev det antydet, at CAD/CAM er et populært emne i dag. For at belyse årsagen til dette, kunne vi tage et historisk tilbageblik over indførelsen af elektronisk databehandling inden for forskellige områder.

Da datamaten blev udviklet, havde man en ide om, at den var velegnet til at udføre store beregninger med, dvs. den var velegnet for teknisk-videnskabelige opgaver. Ikke destro mindre blev det inden for administrative opgaver, at datamaten hurtigst blev udbredt. Den takt, hvori administrative opgaver blev løst ved brug af datamater kan kort beskrives således:

LØNNINGSREGNSKAB:
Helt naturligt var denne opgave en af de første, man tog fat på og fik løst på fuldstændig vis. Grunden er naturligvis, at det er en veldefineret opgave at beregne folks løn. Ind- og uddate er veldefinerede. Det samme er de formler, der bruges til at beregne uddata på basis af inddata.

FAKTURERING:
Denne opgave er kun en lidt mere kompliceret end lønningsregnskab.

DEBITORREGNSKAB:
En naturlig følge af fakturering.

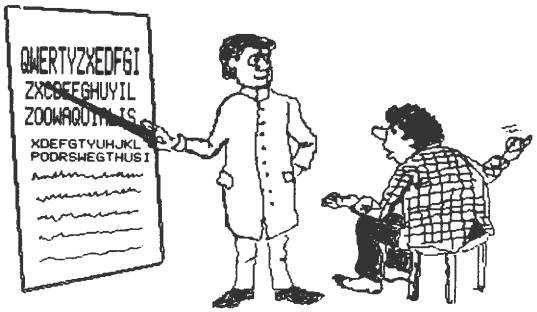
FINANSREGNSKAB:
Hvorfor ikke tage kreditorer og de andre komponenter, der indgår i det totale finansregnskab med?

LAGERREGNSKAB:

Det er en væsentlig men ikke helt simpel opgave at overføre lagerregnskab og -styring til databehandling. Ud over at holde regnskab med den aktuelle lagerbeholdning på basis af til- og afgange omfatter opgaven også at kunne få oplyst (automatisk):

- genbestillingstidspunkt for hver vare.
- genbestillingsmængde for de enkelte varer.
- lagerlokationen hvor varen opbevares.
- og meget meget mere.

TEKSTBEHANDLING:
At kunne fremstille breve, rapporter, dokumenation m.v. med udnyttelse af datamatens faciliteter.



BEKLAGER, DOKTOR, JEG KAN SLET IKKE NOGEN FREMMEDE SPROG

PRODUKTIONSSTYRING:
Denne opgave omfatter at få en datamat til at udarbejde en plan for anvendelsen af en virksomheds resourcer (mennesker, maskiner osv.) baseret på oplysninger om eksisterende/forventede ordrer. Produktionsstyring har mange flere aspekter, men alene de ovenfor nævnte antyder, at opgaven har en vis kompleksitet.

Som det fremgår af ovenstående overordentlig overordnede oversigt over indførelsen af databehandlingen, har man taget fat på de enkelte opgaver i en rækkefølge, der svarer til opgavernes stigende kompleksitet. Tiden er nu til at tage fat de opgaver, som er mere komplekse end produktionsstyring. Sådanne opgaver er:

INFORMATIONSBEHANDLING:
Integration af tekstbehandling med andre datasystemer, således at man i tekstbehandlingen kan anvende de informationer, der findes og opdateres i de andre systemer.

DET AUTOMATISKE KONTOR:
Udover at udføre tekst- og informationsbehandling også at udføre opgaver som:

- brev- og telexforsendelser
- arkivering/fremhentning af dokumenter
- føre kalender
- behandle post

og mange andre rutineopgaver ved hjælp af dataskærme og datamater, der kommunikerer indbyrdes.

Og sidst men ikke mindst CAD/CAM.

På dette sted er det på sin plads at anføre, at produktionsstyring er CAM, men CAM er andet og mere, nemlig ting som numerisk styring af værktøjsmaskiner, anvendelse af robotter mm. Iøvrigt skal vi i denne artikel ikke yderligere beskæftige os med CAM, men straks koncentrere os omkring CAD.

Naturligvis er der også andre årsager end opgavens kompleksitet til at CAD-systemer er så meget på tale netop nu:

- De produkter, der udvikles i vore dage er meget komplekse, og der skal ofte foretages meget store beregninger, som det

måske ligefrem er umuligt at udføre ved håndkraft.

- Af konkurrencemæssige årsager må en virksomhed sikre sig, at de produkter, der sendes på markedet, lever op til markedets forventninger med hensyn til faciliteter og økonomi. Dette kræver en optimering af forbrug af råvarer, produktionstid, vedligeholdelsesprocedurer m.v.
- Dokumentation af moderne produkter er ofte så omfattende, at produktudviklerne har behov for et hjælpemiddel til at holde orden på de mange informationer.

Lad os nu se lidt nærmere på, hvad begrebet CAD dækker.

Først og fremmest bør man bemærke sig ordet "Aided" (=assisteret) i "Computer Aided Design". Heraf kan man se, at der ikke er tale om at lade datamaten udføre hele konstruktionsopgaven. Derimod skal datamaten være et hjælpeværktøj for konstruktøren, som stadig er den, der bidrager med ideer, kreativitet, utraditionel tankevirksomhed osv. Man skal altså ikke kunne gå hen til en datamat og sige, at man ønsker en gearkasse med 5 fremadgående gear, hvoraf et skal være et overgear, samt et bakgear, og så forvente at få alle tegninger og beskrivelser af den ideelle gearkasse ud af datamaten. Derimod kan konstruktøren under arbejdet med med at fremstille en ny gearkasse lade datamaten udføre nogle komplicerede styrkeberegninger, beregne antal tænder i tandhjul for at opnå de rette omsætningsforhold osv.

Vil man som konstruktør anvende datamaten som hjælpeværktøj under konstruktion af et nyt produkt, kan man gøre dette på to måder:

1. Datamaten kan anvendes til analyse af dele af konstruktionen efterhånden som arbejdet skrider frem. Analyser kan være styrkeberegninger, simulering af elektriske kredsløb, beregning af effektforbrug o.l. CAD-programmer til analyseopgaver er mere eller mindre selvstændige programmer, som kan foretage komplekse beregniner, som det vil være meget tidskrævende (eller umuligt at udføre) manuelt.

2.

2. Man kan lade datamaten udarbejde løsningsforslag - syntese - baseret på specifikationer for det ønskede produkt. Indenfor visse områder (f.eks. elektronik) er det faktisk muligt at syntesisere visse delkonstruktioner ved hjælp af en datamat. Arbejdsformen er sædvanligvis den, at konstruktøren til et program i datamaten opgiver nogle specifikationer for det produkt, der skal konstrueres. Datamatprogrammet kan da udarbejde et løsningsforslag, som vurderes af konstruktøren - evt. med anvendelse af en datamat som analyseværktøj. Konstruktøren vil da give nye specifikationer til datamaten, som vil udarbejde et nyt og bedre forslag osv. I en sådan vekselvirkning mellem datamaten (som kan udføre de komplekse beregninger og holde styr på mange data og oplysninger på samme tid) og konstruktøren (som kan udnytte sin erfaring (en sådan har datamaten ikke) og menneskelige hjernes iderigdom), kan man fremstille 6o - 8o% af en ny konstruktion. Den resterende del må tilføjes alene af konstruktøren. I nogle enkelte tilfælde (f.eks. ved udlægning af ikke for komplicerede elektroniske printkort) kan datamaten udføre hele konstruktionsopgaven.

Oftest vil man naturligvis anvende en kombination af 1 og 2 for at opnå en effektiv og sikker produktudvikling.

Totale CAD-systemer

Lad det straks være sagt, at der i mange år har eksisteret dataprogrammer til såvel analyse som syntese i forbindelse med designopgaver. Kun i meget få tilfælde har sådanne programmer imidlertid udgjort et samlet hele - et CAD-system. Generelt kan man også sige, at de større CAD-systemer, der findes på markedet i dag, prismæssigt falder i en klasse, som kun gør dem interessante for meget store virksomheder.

Vore dages datateknologi er imidlertid så udviklet, at man nu vil kunne udvikle nye CAD-systemer, som vil kunne afvikles på prisbilligt (mikro)datamat-udstyr, således at udnyttelsen af de muligheder, der ligger i CAD, kommer indenfor rækkevidden af langt flere virksomheder. Dette er ikke mindst interessant i et land som Danmark, hvor industrien består af mange mindre virksomheder.

At udarbejde et CAD-system er kunsten at sammenstille en række programmer, der kan arbejde sammen, dvs. f.eks. anvende de samme data og assistere konstruktøren i en given opgave. Man vil næppe nogensinde se et generelt CAD-system, som kan anvendes indenfor alle fagområder. Der findes CAD-systemer for maskinbranchen, CAD-systemer for elektronik konstruktører, CAD-systemer til den petrokemiske industri osv.

Fordelene ved at anvende et CAD-system er mange. Lad os her nævne nogle få:

- Når man analyserer sine konstruktioner ved hjælp af en datamat, opnår man større sikkerhed for, at produktet vil fungere tilfredsstillende og pålideligt i det tiltænkte miljø og under de forventede belastninger.

- Omvendt sikrer man sig imod at overdimensionere sit produkt, hvilket igen sikrer, at det produkt, man fremstiller bliver (prismæssigt) konkurrencedygtigt på markedet.

- Lader man en datamat udføre tidskrævende rutineopgaver i konstruktionfasen, sparer man tid, og det medfører, at man hurtigere kan starte salget af det nye produkt og dermed hurtigere få udbytte af de penge, man har investeret i produktudvikling.

Når man taler om datamat assisteret konstruktion, er der to områder, man især skal fokusere på:

a. Hvorledes meddeler konstruktøren sig til (kommunikerer med) datamaten? og omvendt?

b. Dokumentation af de færdige produkt.

Medens det største spørgsmål er et problem, der opstår som følge af, at man introducerer en datamat i konstruktionsfasen, så er det andet problem et mere generelt problem, som man, som vi senere skal se, får langt bedre muligheder for at løse tilfredsstillende, når man har indført datamatassisteret konstruktion som et totalt CAD-system.

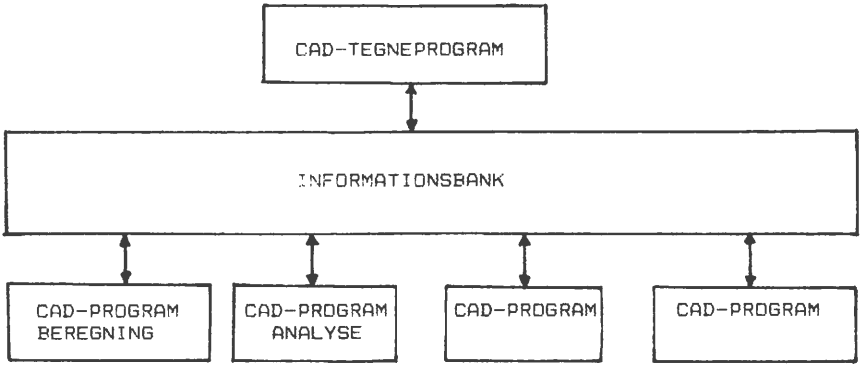
Lad os først se på kommunikationen konstruktør (-) datamat.

De fleste konstruktører arbejder med papir og blyant, når de meddeler sig til omverdenen. Datamater har indtil for ganske nylig krævet, at man instruerede og informerede dem ved at taste tekst og data ind fra et skrivemaskinetastatur. Resultaterne af datamatens behandling af de indtastede informationer er blevet præsenteret som udskrifter på printer eller dataskærme i form af bogstaver og tal - i meget sjældne tilfælde som tegninger.

Den teknologiske udvikling har imidlertid ført til, at man nu har muligheden for at vise tegniner med tilfredsstillende nøjagtighed (opløsning) på en dataskærm. Og hvad der er nok så vigtigt: konstruktøren kan kommunikere med datamaten på en for ham mere naturlig måde, nemlig ved at tegne og pege frem for

at indtaste tal og bogstaver. Sådanne muligheder har naturligvis først været til rådighed på større og dermed dyrere datamater, men nu kan mulighederne også stilles til rådgihed på datamater helt ned til mikro-klassen, og dermed bliver der langt flere muligheder for at indføre CAD-systemer.

Senere skal vi se på teknikken bag at fremstille tegninger ved hjælp af (mikro)datamater, men lad os for indeværende blot konstatere, at det er muligt at kommunikere med en datamat via tegninger - lidt populært sagt: brugeren kan tegne for datamaten og datamaten kan tegne for brugeren. Dette fører til, at vi kan anskue et CAD-system fra den synsvinkel, at det centrale i et sådant system bør være et tegneprogram. Til langt de fleste produkter (selv kemiske produkter), der udvikles, hører en række tegninger. Disse tegniner kan fremstilles med datamaten som hjælpeværktøj. Arbejdet kan som anført ovenfor foregå på en for konstruktøren velkendt måde, nemlig i det store og hele som med papir og blyant. Dog assisterer datamaten konstruktøren under tegnearbejdet, således at det færdige resultat bliver mere nøgagtigt og hurtigere færdigt. Tegninger fremstillet på en datamat skal naturligvis kunne gemmes på datamatens baggrundslager (f.eks. disketter), således at man let kan fremhente og rette/ajourføre dem. Alene denne sidste ting giver så store fordele, at den i sig selv berettiger indførelsen af et CAD-system.



Via tegninger beskriver konstruktøren sit produkt (eller en detalje deraf) overfor datamaten. Denne beskrivelse gemmes som sagt på datamatens baggrundslager, og det er så naturligt at supplere et CAD-tegneprogram med en række andre CAD-programmer, som kan udføre beregninger m.v. baseret på de oplysninger, der indsamles (inddateres) og lagres via tegneprogrammet. Vort CAD-system får således en fælles informationsbank (database), som rettes og ajourføres samtidigt med, at tegningerne rettes og ajourføres. Dette indebærer en række fordele, som leder os frem til det andet væsentlige punkt, der blev nævnt tidligere, nemlig:

Dokumentation:

Et produkt skal dokumenteres (beskrives) i tekst og tegning til brug for produktion, markedsføring, salg, installation, betjening, vedligeholdelse m.v. Produktdokumentationen er oftest det sidste, der bliver fremstillet i forbindelse med udviklingen af et nyt produkt, og så har produktudvikleren ofte vanskeligt ved at opretholde den rette begejstring for det produkt, der for ham nu er historie, til at udarbejde en tilstrækkelig god dokumentation . Ofte anvender den, der har udviklet produktet, en dokumantationsform og et sprog som markedsføringsfolkene ikke kan forstå/anvende og som er alt for teknisk til at indgå i installations-, vedligeholdelses- og betjeningsvejledninger. Resultat: Når et produkt er færdigudviklet vil man ofte se at flere forskellige personer i virksomheden hver for sig udarbejder deres egen produktdokumentation til forskellige formål:

- Produktionsafdelingen laver beskrivelser, der passer til de (måske numerisk styrede) maskiner, der anvendes til produktionen.
- Marketingsafdelingen beskriver produktet med henblik på annonce og reklamemateriale.
- Salgsafdelingen laver produktbeskrivelser til brug for sælgerne.

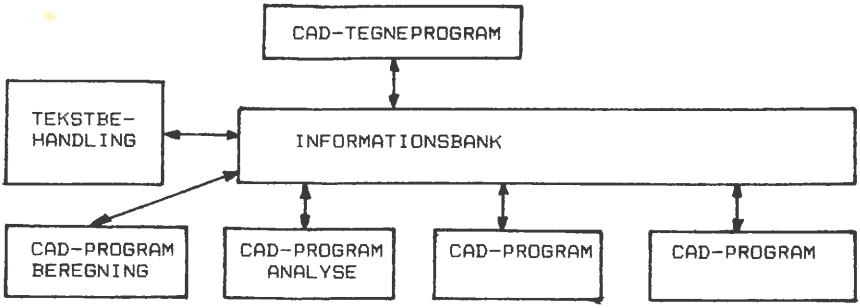
- en eller anden afdeling udarbejder betjeningsvejledning til brugerne.
- Vedligeholdelsesafdelingen laver installations- og vedligeholdelsesinstruktioner til det tekniske personale.

Det kan meget vel forekomme, at man 6 til 7 steder i en virksomhed udarbejder egne beskrivelser af et nyt produkt.

Et andet problem med produktdokumentation er vedligeholdelsen. Alle (eller næsten alle) produkter ændres og justeres hen ad vejen. Dette medfører, at tegninger og tekst i produktdokumentationen skal rettes. Dette er et tidskrævende og ikke særligt attraktivt arbejde, som ydermere indebærer en stor risiko for at blive udført forkert og ufuldstændigt, når der er tale om flere uafhængige sæt beskrivelser, der skal tilrettes.

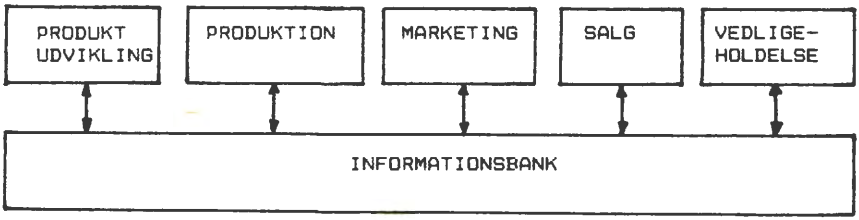
Hele dokumentationsproblematikken kan forenkles drastisk, hvis man baserer sin produktudvikling på en central informationsbank som beskrevet ovenfor. Informationsbanken skal indeholde såvel de tegninger som den tekst, der beskriver et produkt. Et godt CAD-system skal derfor også omfatte et tekstbehandlingssystem til at:

- lagre tekst i informationsbanken i et format, der kan anvendes af de andre programmer i CAD-systemet.
- behandle tekst og tegninger (!), der findes i informationsbanken, og som måske er skabt af andre programmer i systemet.



Straks fra det tidspunkt, hvor ideen til et nyt produkt opstår, og gennem hele produktudviklingsfasen samt i hele produktets levetid, skal man tilsikre at alle tegninger og beskrivelser vedrørende produktet registreres i den centrale informationsbank. Alle rettelser og ajourføringer, der forekommer, skal naturligvis også registreres her.

Informationsbanken skal være opbygget således, at man kan give andre afdelinger end produktudviklingsafdelingen adgang til at benytte og lagre informationer i den centrale bank. For at kunne tillade dette, må der naturligvis findes visse sikkerhedsforanstaltninger, som gør det muligt at begrænse adgangen til informationsbanken. Nogle afdelinger skal måske kun have lov til at hente tegninger og beskrivelser fra banken, medens andre afdelinger også må rette i (nogle af) informationerne.



Udstyrer man nu de relevante afdelinger med udstyr, som gør det muligt at benytte informationsbanken, kan man eleminere en masse dobbeltarbejde og samtidig opnå en langt bedre dokumentationsstandard:

- I en marketingsafdeling kan man lade en tekstforfatter via en tekstbehandlingsterminal bearbejde de beskrivelser, produktudviklerne har forfattet, således at teksterne kan indgå i brochurer, annoncer osv.

- Salgsafdelingen kan i informationsbanken hente de oplysninger, der skal bruges til prisberegninger.
- Produktionsafdelingen kan via informationsbanken automatisk få fremstillet de programmer, der skal benyttes til numerisk styrede værktøjsmaskiner og robotter, der anvendes i produktionen.
- Produktionsafdelingen vil også langt hurtigere og mere præcis blive informeret om ændringer i produktet, og dermed i produktionen.
- Indkøbsafdelingen kan konsultere informationsbanken for at finde ud af, hvilke råvarer, der skal indkøbes.
- Vedligeholdelsesafdelingen får fra informationsbanken oplysninger om hvilke reservedele, der er behov for.
- Vedligeholdelsesafdelingen kan lave fejlstatistikker i informationsbanken. Disse statistikker er en tilbagemelding til så vel produktudviklerne som produktionsafdelingen om, hvilke justeringer der eventuelt skal foretages.
- Mange af de tegninger, der der fremstillet under udviklingen af et produkt, kan med enkelte modifikationer anvendes i installations-, betjenings- og vedligeholdelsesvejledninger.

Det ideelle system er det gode systems værste fjende.

Nu må man naturligvis passe på, at man ikke med baggrund i overvejelser af ovenstående art læner sig tilbage og venter med at indføre datamatassisteret konstruktion i sin virksomhed indtil man på markedet kan købe et totalt CAD-system, der kan løse alle problemstillinger i virksomheden. Det eneste man opnår herved er, at man aldrig får indført datamatassistance i produktudviklingen.

Den konklusion, man skal drage af ovenstående er, at man ved indførelsen af datamatassisteret konstruktion skal bygge dette op omkring:

- brugervenligt udstyr, som det falder naturligt at bruge.
- en central informationsbank, som kan være til gavn og glæde for hele virksomheden.

Med dette in mente, kan der være megen fornuft i at indføre datamatassisteret konstruktion i flere faser. I en virksomhed vil der som oftest være et eller to områder, hvor det er særligt fordelagtigt at tage en datamat med i produktudviklingsfasen. Ved at koncentrere indsatsen omkring sådanne enkelte områder, vil investeringerne i dataudstyr være overkommelig, og man vil indhøste nogle erfaringer, som vil være guld værd, når aktiviteterne udvides til at omfatte andre dele af virksomheden.

I næste artikel vil vi gå i detaljer med, hvilket dataudstyr der findes til brug ved datamatassisteret tegningsfremstilling, ligesom vi vil se på CAD tegneprogrammer og opbygningen af informationsbanker.

Hvordan SKF82 blev til

ved J. Meilgård

Undervisningssituationen

I undervisningen sker der en faglig indlæring af de emner, som kursusplanerne anbefaler, enten gennem indlæring af teoristof med efterfølgende øvelser, eller gennem indlæring af en bestemt problemløsningsteknik.

Her opstår så de første problemer.

Forskellige metoder

Den metodik som bruges i mange af de applikationsprogrammer, der kan købes er meget forskellig fra den form, hvorunder eleverne har arbejdet med stoffet. Der bruges en anden terminologi, der bruges andre fremgangsmåder, som ikke harmonerer med den måde teoristoffet er gennemgået på, og derved opstår der kommunikationsproblemer, netop som eleverne skal til at opnå en begyndende rutine.

Naturligvis vil eleverne i erhverssituationen kunne omstille sig til det tilbudte programmel, fordi de her formentlig har den nødvendige tilpasningstid.

Tilpasning og samarbejde

En løsning for de tekniske skoler må være at programmerne tilpasses en undervisningssituation, som erhvervene selv er gået ind for, ved at etablere et samarbejde mellem lærerne ved de tekniske skoler og programmelhusenes forfattere, eller ved at vi selv udarbejder det nødvendige programmel. Softwarehusene kan lære os, hvordan velfungerende programmer designes.

Strategi

En løsningsstrategi for et konstruktionsprogram kan være:

1. Hvad er målet?
2. Modelstruktur.

3. Algoritmer og testdata.
4. Betjeningsvejledning

Målbeskrivelse

Her kan der hentes inspiration i kursusplanernes emnemålbeskrivelser. De vil normalt være en tydeliggørelse af bekendtgørelsernes indhold.

Citat:
Emne: Dimensionering af transmissionselementer

Formålet med undervisningen er at sætte eleven i stand til på baggrund af opgivne data, selvstændigt at gennemføre dimensionsberegniner ved hjælp af firmakataloger, håndbøger og/eller lærebøger.

Altså eleven skal ud fra en række opgivne data selvstændigt kunne vælge det mest velegnede design.

Begrænsning

Traditionelt gennemregnes kun ganske få muligheder, idet tiden og undervisningssituationen ikke muliggør en optimering af designet, fordi det kræver mange ressourcer at gennemføre en løsning.

Dimensions forståelse

Her giver så microdatamaten en ny dimension, idet programmerne på meget kort tid giver en række valg, som elev/lærer så kan diskutere slutløsning ud fra. Derved kommer eleven over i en mere relevant situation omkring problemløsningsteknik og valg af løsning.

Samtidig opnås en dimensionsforståelse hos eleven, som kun lang tids praktik omkring dimensionering kan give.

Konstruktøren får bedre mulighed for at få en dialog med kunden, frem for at bruge arbejdstid op trivielle løsninger, og derved opnås en større tilfredsstillelse for kunden og samtidig forbedres konstruktionsafdelingens produktivitet en del.

Modelstruktur

Modellers frihedsgrad

Beregningsmodellers struktur er stort set fastlagt fra producenternes side, når der anvendes standardkomponenter. Der mangler måske blot en ststematisering af modellen.

Andre mænder

Hvor der ikke anvendes katalogkomponenter har lang tids konstruktionserfaring lært os hvordan problematikken løses, når der anvendes traditionel løsningsmetodik.
Hvem siger, at dette nye værktøj blot slavisk skal arbejde efter de samme fremgangsmåder?

Analyse

Hvor det er muligt at lette brugers adfærd og derved øge programmellets anvendelighed, må konstruktøren anstrenge sig, så der opnås større sikkerhed og en rimelig tidsreduktion.

Eks.: Hvor opslag i tabeller og derfra inddata med rimelighed kan erstattes af numeriske søgemetoder eller numerisk analyse, bør de numeriske metoder foretrækkes.

Kontrol og vedligeholdelse

Modellen skal også muliggøre en ordentlig dokumentation af fremgangsmåde og resultat, så kontrollørens overblik ikke forsvinder.

Endelig skal modellen tillade en rimelig let vedligeholdelse af en bruger af hvem man ikke kan forlange særlig høj programforståelse.

S K F 82

Modellen er menu-orienteret, interaktiv og hvor det har været muligt er inddata erstattet af numerisk analyse. Uddata er valgfrit på skærm eller printer.
Som noget relativt nyt er rapporterne ikke blot en tabbellarisk opstilling, men del-beregninger og rækkefølge er vist for at tilgodese kontrollen.

Segmenter

Modellen er segmenteret i 3 dele:

- SKF 2, som foretager inddata, inddatakontrol, beregninger og til slut lagrer resultaterne i en binær datafil.
- RUSKÆRM, som giver det ønskede uddatabillede på skærm.
- RUPAPIR, giver det ønskede uddatabillede på rapportform.

Modellen er oprindeligt skrevet til et flerbruger-anlæg, men foreligger både i enkeltbruger og flerbrugerversion.

COMAL-8o kernen

Programmeringssproget er COMAL-8o VI.6c, og forfattarne har gjort sig stor anstrengelse for at holde sig til COMAL-8o kernen, hvor omskrivning til andre sprogversioner vil være meget let.

Algoritmer og testdata

De anvendte algoritmer fremgår af rapporten TA2:SKF82, som mange tekniske skoler har købt.

Her skal blot gives et eksempel på den numeriske analyse. Side 2o i rapporten. Tabeldata fra side 115 i SKF katalog 32ooo Reg. 8oooo.1981-1o.

Beregningsfaktorerne X og Y.

Beräkningsfaktorerna X och Y för spårkullager

	Normalt lagerglapp					Lagerglapp C3					Lagerglapp C4				
		$F_a/F_r \leq e$		$F_a/F_r > e$			$F_a/F_r \leq e$		$F_a/F_r > e$			$F_a/F_r \leq e$		$F_a/F_r > e$	
F_a/C_0	e	X	Y	X	Y	e	X	Y	X	Y	e	X	Y	X	Y
0,025	0.22	1	0	0.56	2	0.31	1	0	0.46	1.75	0.4	1	0	0.44	1.42
0.04	0.24	1	0	0.56	1.8	0.33	1	0	0.46	1.62	0.42	1	0	0.44	1.36
0.07	0.27	1	0	0.56	1.6	0.36	1	0	0.46	1.46	0.44	1	0	0.44	1.27
0.13	0.31	1	0	0.56	1.4	0.41	1	0	0.46	1.3	0.48	1	0	0.44	1.16
0.25	0.37	1	0	0.56	1.2	0.46	1	0	0.46	1.14	0.53	1	0	0.44	1.05
0.5	0.44	1	0	0.56	1	0.54	1	0	0.46	1	0.56	1	0	0.44	1

e:= funktion af F_a/C_o

Den almindeligste fremgangsmåde har altid været at der blev interpoleret liniært hvis F_a/C_o antog værdier, som ikke direkte kunne aflæses af tabellen.

Analyse

En analyse afslørede, at den almindelige liniære interpolation var ikke så lidt forkert, særligt for høje værdier af F_a/C_o.

Ved en forsøgsvis ændring af aksernes inddeling, så de blev dobbelt logaritmiske opnåedes en meget nær tilnærmelse til den lineære funktion. Den lineære funktion er bekendt for en mellemtekniker, dog ikke med logaritmiske akser.

Ved lineær regressionsanalyse findes følgende værdier for korrelations-koefficienten:

:R:= o.97lo576
l:R:= o.944o686
2:R:= o.998loo7 og denne værdi er acceptabel, hvorved e kan bestemmes som:

e:= alnx (o2332llxlnx(F_a/C_o)-o.674oo2l), med definitionsmængden o.o25 = F_a/C_o = o.5o

I min oprindelige analyse havde jeg brugt den briggske logaritme, men den funktion findes ikke i METANICs COMAL-8o version. Der kaldes LOG for den naturlige logaritme.

I det interval som F_a/C_o forekommer i, er de to karakteristikker meget nær sammenfaldende, så det giver blot en anden værdi for det stykke, der afskæres på y-aksen.

Testdata

Konstruktøren skal sammen med bruger opstille de nødvendige testdata, så designet bliver så sikkert som muligt.

Dette opnås ved grænseværditest af inddata samt de nødvendige logik-tests for at undgå matematikfejl. Om der skal gennemføres sandsynlighedstests af uddata, må bero på data's videre forarbejdning. I mange tilfælde skal der ske en oprunding til nærmeste standard dimension, og det vil i sig selv lægge en dæmper på beregningsnøjagtigheden.
Det vil være en fordel, som bruger kan oplyse definitionsmængden samt sandsynlige grænser for værdimængden.

Betjeningsvejledning

Jo lettere tilgængelig man gør adgangen for brugere uden forkundskaber, jo tungere bliver systemet at arbejde med for den vante bruger.

Tolerante vejledninger

Der burde derfor være nogle alternative dialogformer, hvor den enkelte bruger, alt efter sin erfaring, hurtigt får adgang til systemet. Dog må man huske, at hvis programmellet ikke med jævne mellemrum anvendes, vil ens hukommelse miste færten, og man er på begyndelsesniveau igen.

Hvilke krav man kan stille til dokumentation af den færdige løsning, vil jeg belyse i en sidste artikel.

Eksemplarer af rapporten TA2:SKF82 kan købes for kopieringsprisen 25,- kr. ved henvendelse til forfatteren.

Dokumentation af små applikationsprogrammer ved J. Meilgård

Efterhånden som mængden af applikationsprogrammer øges vil det være en fordel, om der blev beskrevet nogle rimelige krav til den dokumentation, som må følge med programmet og disketten til brugere, som i mange tilfælde er helt uden kendskab til programmering - men som kan anvende sin datamat, når de første betjeningskurser er overstået.

1: Hvad skal løsningen indeholde (dokumentation)?

2: Hvad skal man forlange af selve programmet?

3: Hvordan vedligeholdes programmel?

4: Hvordan distribueres programmellet?
Vedligeholdelsestid/forældelsestid)

5: Hvordan fremskaffes programmel og copyright?

Ad 1.

1.1 Indholdsfortegnelse.

1.2 Betjeningsvejledning:
- Opstart af programkompleks, må kun kræve en minimun-indsats af brugeren.
- Daglig brug af programmerne.
- Testdata.
- Hovedblokdiagram e.lign.

1.3 Fejlsituationer:
- Fejl som opfanges af programmet, som sandsynlige grænsedata.
- Kritiske fejl, som program ikke tager højde for, og som bruger derfor må tage højde for

1.4 Skærmbilleder:
- Dump af inddata (evt. foto
- Uddata skærmbilleder
- Zoneinddelinger
- Fejlfelter

1.5 Rapporter:
- Uddata på papir
- Evt. Printerindstillinger

1.6 Beregningsforskrifter som:
- Detailblokdiagrammer
- Pseudosprog
- Warnier Orr diagrammer
- Strukturdiagrammer
- Anden relevant beregnings-dokumentation

1.7 Eventuelt filindhold:
- post nr.
- postindentifikation
- indhold

1.8 Eventuelt variabeloversigt:
- Identifikationer
- Indhold

1.9 Programlistninger.

1.1o Diskette med følgende indhold:
- Filnavne og pladskrav
- Type:
  - ASCII filer af hensyn til langtidslagring
  - BINÆRE filer af hensyn til program segmentering
- Forbrug af lagerplads og restplads
- Autostart af diskette.

Ad 2: Programmets kvaliteter:

2.1 Brugerrelation:
- Brugervenligt (skal kunne betjenes uden brug af kodeark og manualer)
- Konverserende (interaktivt)
- Skal kunne rettes før beregning
- Selvkontrollerende, hvor det er muligt.
- Alle indlæste data skal kunne udskrives, både på skærm (selvkontrol) og på papir (brugerkontrol. Måske mellemresultater (pædagogisk kontrol)

2.2 Program relation:

- Struktureret og segmenteret.
- Vedligeholdelsesvenligt:
  - Procedure orienteret.
  - Konstanter og andre normdata bør samles i et vedligeholdelsesafsnit.
  - Data overføres som parametre til procedurer.
  - Procedurer som "lånes" af omverdenen, bør være lukkede, så der ikke opstår kolissions-problemer.
- Oversættelsesvenlige:
  - Bør holde sig til "kernen" af sproget, så det let kan oversættes til andre sprog.

    Al for meget "smart" programmering vil stavns-binde programmellet.
  - Resultater opsamles i ASCII-filer, som kan læses af andre programmer.

Ad 3: Vedligeholdelse:

3.1 Forfatterens vedligeholdelsespligt.
- Omfang og brugeroplæring
- Tid og forældelse
- Copyright

3.2 Brugers vedligeholdelse af programmel
- Mulighed for ændringer
- Brugers egne ønsker i relation til forfatters garanti.
- Evt. brugerkvittering for at ændringer er foretaget.

3.3 Maskinleverendørens vedligeholdelse
- Ændring af hardware
- Ændring af software
  - Operativsystemets dele.
  - Sprogændringer og systemets opførsel

Ad 4: Programdistribuering:

4.1 Standard Programmer:

- Matematik
- Statistik
- Økonomi m.fl.

som "bog" pakker med program-listninger.

4.2. Standard Programmer:

- Produktionsstyring
- Lagerstyring
- Netværksplanlægning

som købt software incl. diskette

4.3. Brugerudviklede programmer:

- Beregning
- Nivellering
- Dimensionering
- Simulering

Som brugerudviklet pro - grammel incl. kopierings-diskette.

4.4 Brugerudviklede programmer

- Som T P projekter
- Som afgangsprojekter i.f.m. læreruddannelse (på baggrund af brugerønske-katalog).

Ad 5: Fremskaffe programmer:

5.1 Software houses.
5.2 Samarbejde teknisk skole og software house
5.3 Edb rådgiver tilknyttes en teknisk skole
5.4 Beskæftigelsesprojekter i TS regi
5.5 Samarbejde mellem industri og teknisk skole
5.6 T P Projekter
5.7 Oversættelse og omskrivning af bogprogrammer
5.8 Frihed til produktudvikling i arbejdstiden.
5.9 Micro laboratorier på flere tekniske skoler
5.1o Opbygning af programorganisation på teknisk skole
5.11 Årets edb-opgave på teknisk skole med præmie.

Introducerende EDB-kursus - et undervisningsforløb.

Ved Kaj Thrysøe

Indledning

Denne artikel består af to dele.
Første del redegør for mit formål med at offentliggøre dette undervisningsmateriale, de overvejelser der ligger bag materialets form og indhold samt mine praktiske erfaringer hermed. Anden del udføres af det materiale kursisterne fik udleveret i forbindelse med kurset.

Formål med artiklen

Undervisning er en stadig udviklingsproces. Læreren gør sig nye erfaringer i den daglige undervisning og justerer til stadighed undervisningens form og indhold. Således gennemgår også undervisningsmaterialet en bestandig forandring.

Nærværende undervisningsforløb prætenderer altså ikke at være materialet, - det findes vel næppe heller!?

Formålet med denne artikel er først og fremmest at medvirke til en større offentlighed omkring den daglige undervisning i edb på de tekniske skoler.

Ved at offentliggøre gode og dårlige erfaringer kan viforhåbentlig højne den faglige og pædagogiske udvikling hos den enkelte lærer og dermed udvikle og fastholde en mere kvalificeret undervisning i edb på de tekniske skoler.

Kursets målgruppe

Nærværende undervisningsforløb er skrevet til et introducerende edb-kursus for faglærere ved Aarhus tekniske Skole. - Det er kørt igennem for lærere indenfor jern- og metalområdet, og det drejer sig om folk indenfor maskin- og værktøjsfaget, smedefaget og autofaget.

De tre faggrupper kørte hver for sig, hvilket gav en optimal faglig homogenitet.

Kurset fordrede ingen edb-kundskaber hos deltagerne. - Nogle af kursisterne havde deltaget i SEL's edb-kurser, men det gav ikke anledning til undervisningsmæssige problemer. Det har imidlertid givetvis opvirket det sociale klima på holdet (jo større edb-viden, jo større social status) og det har sikkert medført en større spredning i det faglige udbytte af undervisningen.

Kursets indhold og omfang

Kurset er planlagt til 2o lektioner og består af tre blokke. I den første blok præsenteres maskinen, COMET 3ooo og 34oo. Der instrueres i indlæsning af operativsystem (CP/M) og fortolker (COMAL-8o) og de vigtigste tastfunktioner og styrekommandoer gennemgås.

Denne blok skal give kursisterne en vis fortrolighed i at anvende maskinen. Kendskab til tastfunktioner og styrekommandoer indøves ved hjælp af færdige programmer.
Database og CPR-programmerne fremkalder næsten automatisk en debat omkring anvendelsen af personregistre i det offentlige og på arbejdspladsen. De mindre programeksempler lægger op til en mere begrebsmæssig indlæring af programkommandoer i blok II. Man kunne tro, at programeksemplerne, der er af overvejende administrativ karakter, ville blive oplevet som mindre relevante i en teknisk sammenhæng, men det blev ikke tilkendegivet særligt tydeligt af kursisterne.

Blok I skal imidlertid suppleres med en gennemgang af maskinarkitektur og især skal samspillet talvariable og regneenheden trækkes tydeligt op i forbindelse med en senere forklaring af tildelingssætningen.

I blok II gennemgås de mest elementære programkommandoer og en algoritmebeskrivelsesmetode i pseudokodesprog. Her præsenteres endvidere en problemløsningsmetode, som jeg finder nyt-

tig i arbejdet med udvikling af programmer.

Det kan i starten være svært for kursisterne at forstå meningen med algoritmebeskrivelsen, men senere i forløbet - i blok III - bliver det i al fald lysende klart for de fleste, hvilket formål en sådan beskrivelse tjener.

Blok III er en overbygning på blok I og II og er udformet som en praktisk Case omkring processtyring. - Det væsentligste her er algoritmebeskrivelse og procedurebegrebet. Denne Case er overordentlig engagerende for kursisterne, hvilket nok skyldes to forhold: De kan forbinde programmeringsarbejdet med noget konkret, og de oplever et færdigt velafgrænset stykke arbejde.

Materiel (håndteringsmaskine) og programmel hertil er udviklet af Helge Jensen, Viborg tekniske Skole, og er brugt i SEL's kursusrække - Anvendt mikroprocessorteknik. Programmellet er ændret en lille smule af Ole Karmark, Aarhus tekniske Skole. Hensigten med denne ændring er rent pædagogisk og betyder, at kursisterne ikke behøver at gå ind i hard-ware(porte) på dette tidlige tidspunkt.

Pædagogiske overvejelser

Dette undervisningsmateriale er delvis selvinstruerende uden dog at lægge op til programmeret undervisning. Materialet giver mulighed for differensieret undervisning ud fra den betragtning, at ikke alle når lige langt i indlæringsmæssig dybde. Kursisterne kan stort set arbejde i deres egen rytme og ud fra deres personlige forudsætninger. De kan arbejde i gruppe eller individuelt alt efter temperement. Læreren giver instruerende oplæg og gennemgår centrale begrebsmæssige størrelser i plenum. I den resterende tid fungerer han som konsulent, som kursisterne kan trække på efter behov.

Stofmængden er valgt og tilrettelagt således, at kursisterne ikke bliver "kvalt" i starten, samtidig med at der hele tiden er et "håndtag" indbygget i stoffet, så den enkelte kursist ikke mister balancen og dermed selvtilliden.

Formålet med dette kursus er at afmystificere og vække interesse for edb - ikke som det ofte er set - at føre tingene mere indviklet end de behøver at være.

Edb er ikke kun et spørgsmål om maskiner, det er også i høj grad et spørgsmål om mennesker.

Jeg tror imidlertid det er vigtigt at starte med maskinen, fordi arbejdet hermed giver nogle vigtige erkendelser, der så senere kan udfoldes omkring de egentlige edb-baserede systemer. Med denne tilgangsvinkel tror jeg desuden vi kan undgå mange ørkesløse diskussioner (hvor angsten ofte bliver det dominerende) og få en mere kvalificeret debat omkring de samfundsmæssige konsekvenser af anvendelsen af edb. Det overordnede perspektiv her må være bestemt af forandringsproblematikken. Forandring

- på arbejdet,
- i uddannelsessystemet,
- i lokalsamfundet osv.

Det anvendte programmeringssprog er COMAL, og det giver mulighed for et interaktivt arbejde med datamaten. Kursisterne får en hurtig respons på eventuelle fejltagelser, hvilket helt givet fôrøger motivationen for det videre arbejde.

Litteratur og programmer

Dette materiale har foruden den daglige undervisning især hentet inspiration fra følgende fremstillinger:

Lone Verner Nielsen m.fl.: Grafo-projektet. Århus 1983

Børge Christensen: COMAL. Problemløsning og programmering. 1 og 2. Bogika 198o/81.

Poul Østergård: Programmering i COMAL-8o. Teknisk forlag 1982.

A. Egebjerg Johansen m.fl.: Datalære 1. EDB i hverdagen. Systime 1983.

Programmerne DATABASE og CPRNR er udarbejdet af Frede Dybkjær og kan erhverves hos Forlaget Systime.

Programmerne MANUAL og MASKIN er udarbejdet af Helge Jensen, Viborg tekniske Skole, og programmet MASKIN er i dette undervisningsforløb modificeret af Ole Karmark, Aarhus tekniske Skole.

Introducerende EDB-kursus

Indledning

Formål

Kursets formål er at give deltagerne kendskab til følgende:

- hvordan bruges afdelingens mikrodatamat
- hvad er et program
- hvordan konstrueres programmer
- hvordan anvendes edb indenfor fagområdet - og i samfundet iøvrigt.

Afvikling

Den konkrete afvikling af kurset er tænkt opdelt i 3 blokke.

I. Hvordan bruges afdelingens datamat.

- systemets enheder
- opstart af datamaten
- kørsel af database- og CPRprogram
- indlæsning og kørsel af udleverede programmer, herunder indøvelse af styrkekommandoer og tastfunktioner.

II. Konstruktion af programmer

- problemløsningsmetode
- indøvelse af programkommandoer: Print, Input,: = (tildelinger) - simple talvariable, If-Then-Endif, If-Then-Else-Endif, tekstvariable og While-Endwhile.

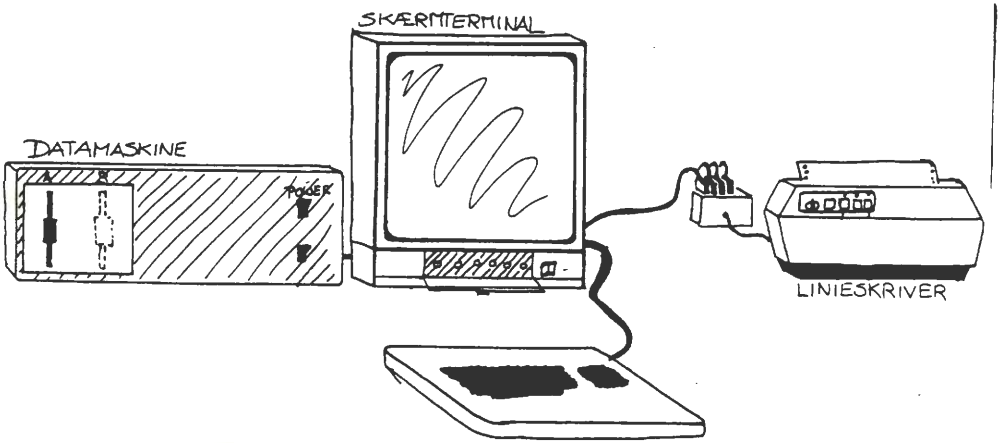
III. Programmering af kran og transportbånd

- et tilfælde af processtyring
- studering af arbejdsprocessen
- beskrivelse af arbejdsprocessen
- udarbejdelse af program
- efterkontrol af resultat.

Afslutning

- evaluering af kursusforløbet.

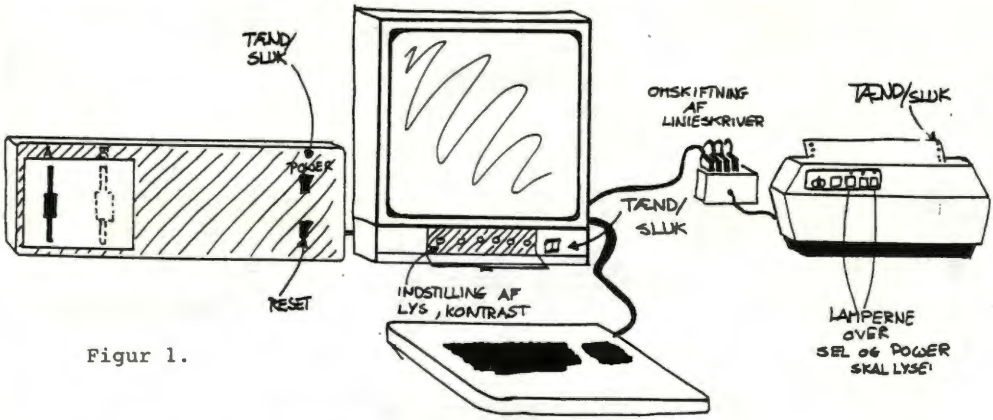
Systemets enheder



På tegningen ses datamaskinen og de enheder, der er knyttet til den. Selve datamaskinen er den orange kasse, her har bearbejdningsenheden og arbejdslageret. De to sprækker foran er til at sætte disketter i, dvs. vi har en datamaskine med indbygget diskettestation. Disketterne er baggrundslagre, og vi kan kun få datamaskinen til at udføre de programmer, der er lagret på disketten, hvis vi først sørger for at få den læst ind i arbejdslageret.

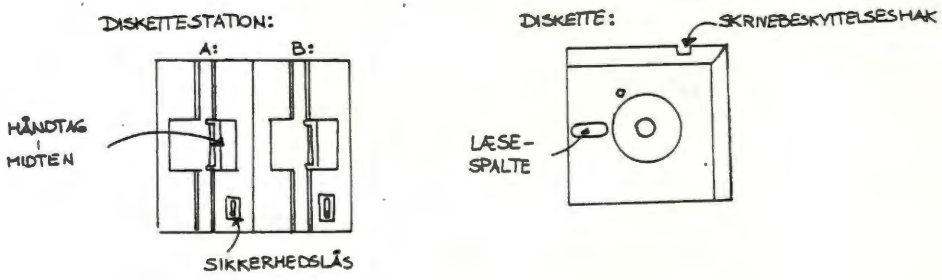
Foran datamaskinen ses tastaturet. På tastaturet skriver man både de ordrer, som datamaskinen skal udføre og de oplysninger (data), som datamaskinen skal bruge. Når man skriver på tastaturet, kan det man skriver ses på skærmen. Skærmen og tastaturet udgør tilsammen en kombineret indlæse-/udskriveenhed, dvs. vi kan kommunikere med datamaskinen gennem skærmen og tastaturet. Skærm og tastatur kaldes tilsammen en skærmterminal. Til venstre ses linieskriveren, som er en udskriveenhed. På linieskriveren kan man få udskrevet sine programmer og evt. de resultater, som datamaskinen kommer frem til under udførelsen af programmet.

Opstart af datamaskinen



Figur 1.

1. Tænd for strømmen ved stikkontakten.
2. Tænd for datamaskinen ved "POWER"-knap, som derefter lyser (se figur 1).
3. Tænd for skærmen ved "OFF-ON" (se figur 1). Derefter lyser lampen på skærmen. Hvis ikke der efter kort tid er tekst på skærmen, trykkes "RESET"-knappen (se figur 1).
4. Tænd for linieskriveren ved kontakten bagpå. Derefter lyser "POWER"-knappen foran på linieskriveren. Tænd også for kontakten her.



Figur 2.

5. Indsæt disketten i diskettestationen A (se figur 2).
   Disketten indsættes på følgende måde: Disketten skydes lodret ind med læsespalten indad og skrivebeskyttelseshakket opad. Det midterste håndtag skubbes mod venstre, og sikkerhedslåsen skydes op (Comet MPS 3ooo). Ellers drejes håndtaget 9o° (Comet 34oo).

6. Tast 1 på tastaturet.
   Herefter kommer en udskrift på skærmen, der slutter med:

   A>

7. Tast COMAL-8o | RETURN | på tastaturet.
   Herved læses det program, som bruges til at fortolke vores programmers COMAL-ordrer ind i arbejdslageret (COMAL-fortolkeren). Når COMAL-fortolkeren er læst ind, kommer følgende udskrift på skærmen:

   Ønskes tekster ved fejlmeldinger (J/N)?

   Tast J på tastaturet.
   Herefter kommer der en stjerne (*) på skærmen, og systemet er nu klar til brug.

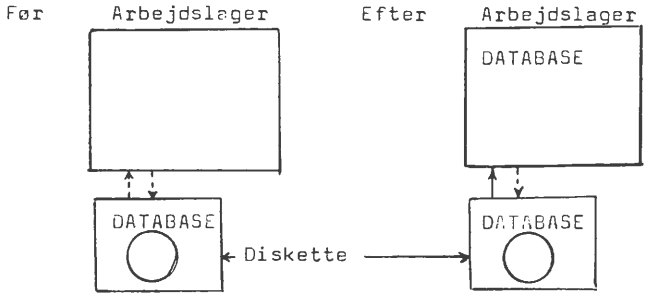
   HUSK. Hver ordre afsluttes med at trykke på | RETURN | -tasten. Hvis ordren, der gives til datamaskinen ikke accepteres, fordi man har skrevet forkert, eller det ikke er en korrekt COMAL-ordre, skrives den ud på skærmen sammen med en fejlmeddelse. Man kan så rette fejlen i ordren eller trykke på | ESC | - tasten og skrive ordren om.

Database

En database er en samling af registre på en fysisk lokalitet. En databank er en samling af databaser fra flere forskellige lokaliteter.

Hent programmet database, der ligger på disketten og læs det ind i arbejdslageret.
Det gøres ved følgende kommando: LOAD DATABASE RETURN
LOAD er styrekommandoen.
DATABASE er programnavn
Udførelse af RETURN betyder, at kommandoen nu er afgivet.

Der sker nu følgende



NB! Vi har taget en kopi af programmet DATAbase og hentet det ind i arbejdslageret.

* betyder at systemet er klar til at modtage ordrer.
Udfør programmet ved at skrive RUN RETURN
RUN er en styrekommando, der "kører" programmet igennem.
Udførelse af RETURN betyder, at kommandoen nu er afgivet.

Der opstår nu problemer med at komme ind i databasen. - Vi skal have en "nøgle" for at komme ind. Programmet er med vilje gjort mere besværligt for at gøre beskyttelsen (eller magten!!) mere effektiv.

Følgende fremgangsmåde skal anvendes:

1. Afgiv en generel kode for at komme ind i databasen.
2. Afgiv en personlig kode til hvert af de registre vi vil se.
3. Tast tallet, der svarer til det aktuelle register.

Generel kode:
82XY35 (bemærk, der bruges store bogstaver. 82xy35 kan ikke bruges).

Personlig kode:

AB34X → 1. personregister
BC87Y → 2. hospitalsregister
CD89Z → 3. økonomireg.
DE35P → 4. straffereg.
EF17V → 5. uddannelsesreg.
FG22S → 6. militærreg.
HI35R → 7. ikke flere data

Eksempel

Vi vil kikke i register 6
1. Tast 82XY35 RETURN
2. Tast FG22S
3. Tast 6

Opgave

Hvordan fik vores registerperson en spiritusdom ?

- var det fordi han røg ind i politiets rutineundersøgelse ?
- var det hans fraskilte kone, der meldte ham ?
- var det et færdselsuheld, der afslørede ham.

Svar: _____

CPR-nummer
Hent programmet CPRPROG og kør det.
Det gøres ligesom før: LOAD CPRPROG RETURN
* RUN RETURN

Opgave

Kontroller dit eget CPR-nummer.
Prøv at konstruere et falsk CPR-nummer.

Beregning af CPR-nummer

Der findes flere beregningsmåder til beregning af nummersystemer.
I forbindelse med cpr.numre har man valgt den såkaldte Modulus 11-metode, der bygger på følgende: De seks første cifre kaldes klassefikationscifre (de er givet). De næste tre cifre er et løbenummer (de vælges) og det sidste ciffer er et kontrolciffer (det beregnes).
Se f.eks. på følgende personnummer

13 07 45 2215
dato md. år. lbnr. kontrolciffer

Kontrolcifret beregnes ved hjælp af nogle faste vægte.

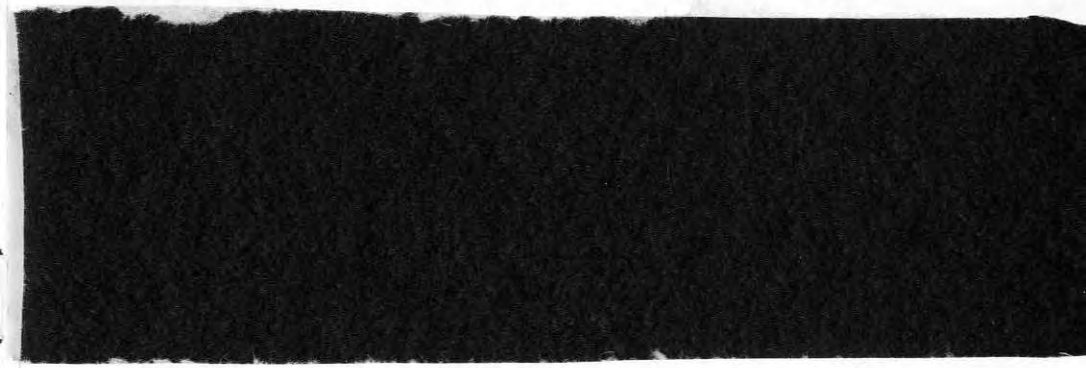
1. Første ciffer	ganges	med	4 :	1x4 = 4
Andet -	-	-	3 :	3x3 = 9
Tredje -	-	-	2 :	ox2 = o
Fjerde -	-	-	7 :	7x7 = 49
Femte -	-	-	6 :	4x6 = 24
Sjette -	-	-	5 :	5x5 = 25
Syvende	-		4 :	2x4 = 8
Ottende	-	-	3 :	2x3 = 6
Niende	-	-	2 :	1x2 = 2

2. Disse multiplikationer adderes : 127
3. Dette tal divideres med 11 : 127/11 og herved fremkommer et helt tal og en rest : 11 + rest 6.
4. Denne rest trækkes fra 11 : 11-6=5 og dette er vores controlciffer.

Ulige kontrolcifre tildeles en dreng og lige tildeles en pige.

Denne beregningsmetode kan overføres til et program, hvorved datamaskinen kan kontrollere om det er et lovligt personnummer, der er læst ind i maskinen.

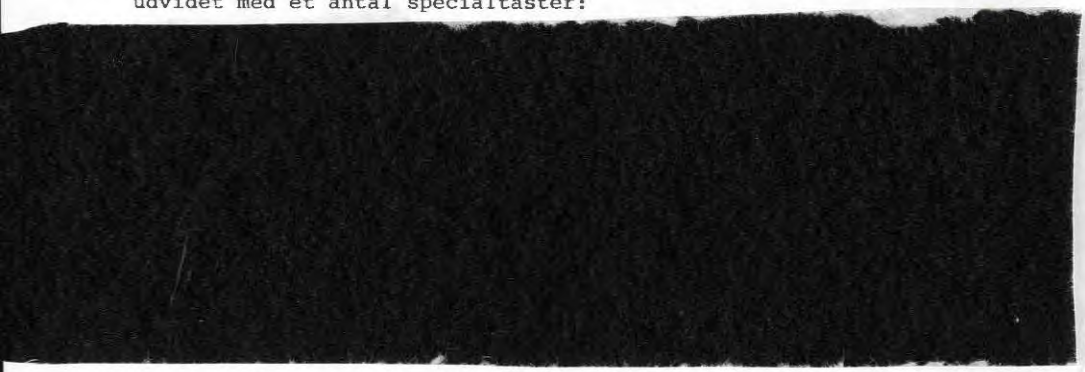
Modulus 11-metoden kan fange de fleste inddateringsfejl, men ikke dem alle. Vi kan faktisk godt konstruere et falsk cpr. nr. ved hjælp af modulus 11-metoden ved at bytte om på cifrene. I sådan et tilfælde vil datamaskinen acceptere nummeret som værende gyldigt.



Skærmterminalen

Skærmterminalen består af et tastatur til indtastning af data samt en skærm, hvorpå resultatet af indtastningen kan ses. På skærmen vises en lysende firkant - kaldet markøren. Denne markerer, hvor på skærmen det næste indtastede tegn vil blive placeret. Når et tegn tastes ind fra tastaturet, placeres det på den position, markøren angiver, og markøren rykker derefter en plads til højre.

Tastaturet ligner et almindeligt skrivemaskinetastatur, som er udvidet med et antal specialtaster:



Bemærk, at der er forskellige taster til tallet 0 og bogstavet o/O, og til tallet og bogstavet l.

Nedenfor beskrives nogle af de taster, vi kommer til at benytte på tastaturet:

SHIFT — Benyttes til at skifte til de øverste tegn på tasterne, f.eks. til store bogstaver.

ALFA LOCK — Benyttes til at skifte fra store til små bogstaver eller små til store bogstaver. På den måde kan man opnå, at al teksten skrives med store bogstaver, uden at det er nødvendigt samtidig at trykke på SHIFT - tasten.

RETURN — Benyttes til at afslutte en linie med, dvs. markøren flytter til første position i næste linie. Datamaskinen starter først med at tolke betydningen af de indtastede tegn fra en linie, efter at der er trykket på RETURN - tasten.

ESC — Benyttes til at afbryde datamaskinens udførelse af en ordre.

 — Benyttes til at flytte markøren en position til højre. Hvis der står et tegn på denne position i forvejen, vil dette ikke blive slettet.

 — Benyttes til at flytte markøren en position til venstre. Hvis der står et tegn på denne position i forvejen, vil dette ikke blive slettet.

INS — Benyttes til at skaffe en fri position ved markøren. Eks. skafe - efter tryk på INS - tasten vil der stå ska fe. Dvs. man kan presse et tegn ind mellem andre ved at bruge INS .

DEL — Benyttes til at fjerne en position ved markøren. Eks. fjjerne - efter tryk på DEL - tasten vil der stå fjerne. Dvs. tegnet bliver fjernet og resten af tegnede skubbes sammen.

De sidste fire taster, der beskrives ovenfor bruges til at rette eventuelle indtastningsfejl i en linie, inden der er trykketpå RETURN - tasten. "Pilene" bruges til at flytte markøren hen til den position, hvor der er indtastningsfejl, og INS - og DEL - tasterne til at slette eller indføje tegn med.

Når man er færdig med at rette i en linie, trykkes på RETURN uanset, hvor på linien markøren er placeret.

Indlæsning af et program, der kan udskrive en tekst

Indlæsning af programmet

```
oolo Print "XXXXXXXXXXXXXXXXXXXX"        Return
oo2o Print "X                  X"        -
oo3o Print "X                  X"        -
oo4o Print "XXXXXXXXXXXXXXXXXXXX"        -
```

Kørsel af programmet

Run Return

Opgave

Kør programmet

Skriv dit navn i feltet

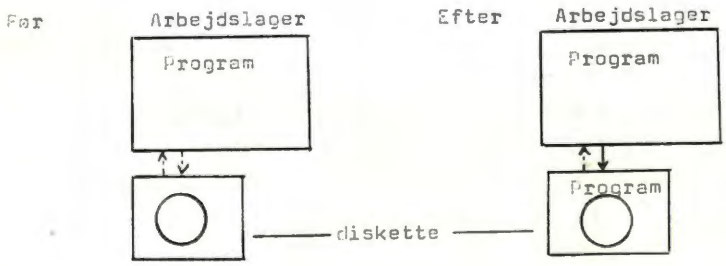
- du kan kalde programmet frem på skærmen med LIST RETURN
- du kan skrive programmet ud på skriveren med LIST LP -
- du kan rette i programmet med EDIT - og bruge de relevante tastfunktioner → ← INS DEL
- du kan lagre dit program sådan

  INIT RETURN

  SAVE Programnavn RETURN Find selv på navnet.

  - Det må højst fylde 8 tegn.

Ved lagringen sker der følgende:



Indlæs nedenstående program og kør det

Det er lettere at få maskinen til selv at udskrive linjenumre.
Det gøres sådab

```
* AUTO    RETURN
oolo PRINT
oo2o PRINT
oo3o PRINT
  .
  .
  .
```

onnn ESC Når programmet er indlæst, skal man ud af den automatiske linjegenerator. Dette gøres ved et tryk på funktionstasten ESC .

Et program, der kan indlæse variableværdier og udskrive dem.
Indlæsning af programmet

```
oolo Print "Antal"                          Return
oo2o Input Antal                            -
oo3o Print "Beløb"                          -
oo4o Input Beløb                            -
oo5o Print "Antal" = ",Antal                -
oo6o Print "Beløb" = ",Beløb                -
```

Bemærk: Print-sætningen i linje oolo og oo3o er vejledende for inputsætningen i linje oo2o og oo4o.
Kommaet i linje oolo bevirker, at linje oolo og oo2o skrives i samme linje under kørslen af programmet. Dvs. at systemet skriver Antal ? - Hvis vi ikke havde brugt kommaet, ville systemet skrive Antal
?
under hinanden

Kørsel af programmet

```
Run          Return
```

Indlæs og kør følgende program:
Et program, der udregner en ny variabel værdi og udskriver resultatet.

Indlæsning af programmet

```
oolo Længde:=1o, Bredde:=35                    Return
oo2o Areal:=Længde*Bredde                      -
oo3o Print "Arealet er :     ",Areal           -
```

Kørsel af programmet

```
Run          Return
```

Prøv at rette i værdierne i linje oolo
Prøv at erstatte navnet Areal med navnet Orla. Hvad sker der?

Følgende program indeholder faciliteter fra de 3 foregående programmer. - Indlæs programmet og kør det.
Prøv herefter at ændre programmet, så det får en momsprocent på 22.

Et program, der kan indlæse forskellige variabelværdier, udregner flere nye variabelværdier og udskriver en standardregning, der kan indeholde forskellige mængder og priser.

Indlæsning af programmet

```
oolo Print "Antal"                             Return
oo2o Input Antal                               -
oo3o Print "Pris"                              -
oo4o Input Pris                                -
oo5o Salgspris:=Antal*Pris                     -
oo6o Moms:=salgspris*25/loo                    -
oo7o Udpris:=salgspris+moms                    -
oo8o Print "De har modtaget "",Antal," Stk."   -
oo9o Print "Til ",Pris, "Kr.pr.stk."           -
oloo Print "Det bliver incl. 25%moms: ",Udpris," Kr., -
ollo Print "som De bedes betale snarest belejligt."   -
```

Kørsel af programmet

```
Run          Return
```

Et program, der gør forskel på folk

- Hvis det købte antal er mindre end 2o, så skal der beregnes et ekspeditionsgebyr på 5% af salgsprisen.
Er det købte antal større end eller lig med 2o, skal der ikke beregnes noget gebyr.

Indlæsning af programmet

```
oolo Print "Antal enheder          ",          Return
oo2o Input Antal                                -
oo3o Print "Enhedspris             ",          -
oo4o Indput Pris                                -
oo5o Print                                      -
oo6o Print                                      -
oo7o Salgspris:=Antal*Pris                      -
oo8o IF Antal  2o Then                          -
oo9o    Salgspris:=Salgspris*lo5/loo            -
oloo    Print "der er lagt ekspeditionsgebyr til"  -
ollo    Print "salgsprisen. Dette gebyr er på 5 pct."  -
ol2o    Print "ved køb af mindst 2o stk."       -
ol3o    Print "kan De spare gebyret."           -
ol4o EndIF                                      -
ol5o Print "Salgspris :          ",Salgspris,"Kr." -
```

Kørsel af programmet

```
Run        Return
```

Ret i programmet sådan at
- antallet skal være over loo og
- ekspeditionsgebyret sættes til lo%.

Resume:

Vi har set på systemets enheder.
Rent logisk opfatter vi enhederne som selvstændige adskilte størrelser, selv om de i praksis virker sammen. Enhederne benævnes:
indlæseenhed (tastatur), udlæse-/udskriveenhed (skærm/linjeskriver), centralenhed (der hvor databehandlingen foregår) og ydre enheder (diskettestation, plotter m.v.)

Vi har foretaget opstart af datamaten.

Vi har arbejdet med følgende styrekommandoer:

LOAD - henter et program fra diskette til arbejdslager (centralenhed.

SAVE - gennem et program fra arbejdslager på disketten.

INIT - foretages inden SAVE, og er en besked til diskettestationen om at vi nu vil skrive på disketten.

RUN - kører programmet.

LIST - henter programmet frem på skærmen.

LIST-LP - skriver programmet ud på linieskriveren.

EDIT - kalder programmet frem linie for linie, så vi kan rette i det.

AUTO - skriver selv linienumre ud.

Hertil kan tilføjes to nye nyttige styrekommandoer.

NEW - sletter indholdet i arbejdslageret.

CLAR - sletter skærmbilledet, men bevarer indholdet i arbejdslageret.

Vi har arbejdet med forskellige tastfunktioner.
De to vigtigste er ESC og RETURN

Vi har følgende
Regnetegn
+ til addition
- til subtraktion
* til multiplikation
/ til division
og

Sammenligningstegn

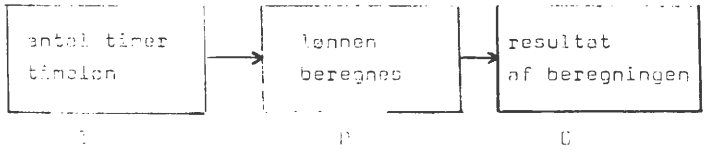
< betyder mindre end
> betyder større end
<= betyder mindre end eller lig med
<> betyder forskellig fra
>= betyder større end eller lig med

Programkonstruktion - et eksempel

1. Problem:
   Udarbejd et COMAL-program, der kan foretage en simpel lønberegning og udskrive resultatet.

2a. Udtænkning af en plan:
   For at kunne løse problemet må vi vide
   - hvor mange timer der arbejdes
   - hvor stor timelønnen er
   - kunne foretage en beregning af vores data
   - kunne udskrive resultatet

   Vi kan illustrere vores løsningsforslag i et diagram:



   I=input(inddata),P=proces (databehandlingsproces) og O=output(uddata) til løsning af vores problem.

2b. Beskrivelse af planen:
   Vi kan nu formulere en algoritme(en plan), der beskriver hvordan vi helt konkret kan løse vores opgave.

   Den algoritmebeskrivelse vi vil anvende kaldes pseudokode, og den er kun en ud af flere muligheder (beskrivelsesmetoder).
   Pseudokoden er en slags overgangsbeskrivelse mellem naturligt sprog og programmeringssprog.

   Algoritme:pseudokode

   skriv indtast timetal (ledetekst)
   indlæs timetal

skriv indtast timeløn (ledetakst)
indlæs timeløn
beregn samlet timeløn
skriv samlet løn
skriv løn (værdien)

De understregede ord kaldes nøgleord, og angiver de handlinger vi må udføre for atløse vores problem

3. Udførelse af planen

For at udføre vores plan på en datamaskine må vi konstruere og indlæse et program i et sprog datamaskinen kan "forstå". Vi anvender programmeringssproget COMAL-8o.

Programmet får følgende udseende:

COMAL-8o program(sammenlign med algoritmen)

```
1o // Simpel lønberegning ----------> linjen er kun til bruge-
                                       rens orientering.
2o PRINT "Tast timetal", ----------> vi får en ledetekst ud
                                       på skærmen under program-
                                       udførelsen.
3o INPUT timetal ----------> vi opretter variablen:
                             timetal, og kan tildele den
                             forskellige værdier.
4o PRINT "Tast timeløn", ----------> som linje 2o
5o INPUT timeløn                     som linje 3o variablen
                                     hedder her:timeløn
6o løn:=timetal*timeløn ----------> vi beregner løn og pla-
                                     cerer værdien i variablen:
                                     løn
7o PRINT "Samlet løn, kr." ----------> ledetekst til skærmen
8o PRINT løn ----------> vi skriver værdien af løn
                         ud på skærmen.
9o END ----------> her slutter programmet.
```

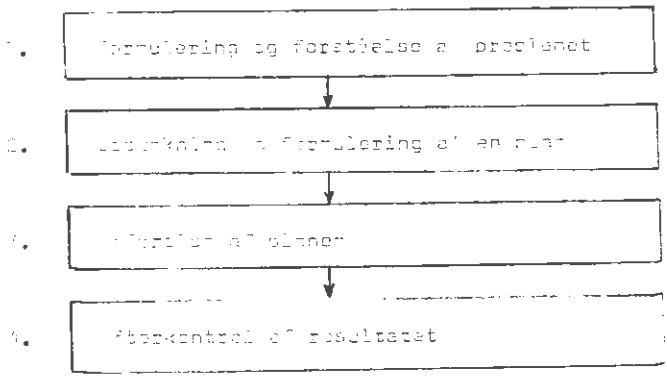
4. Efterkontrol af resultatet

Kør programmet og se om det virker. (RUN). Det gør det som

regel ikke første gang!!
Er der fejl i programmet, så få eventuelt en udskrift på linjeskriveren (LIST LP:).
Analysér programmet, lokalisér fejlen og ret den (EDIT linjenr.)!

Problemløsningsmodel

Vores problemløsningsmetode indeholder fire faser, der teoretisk ligger efter hinanden i følgende orden



I praksis foregår det imidlertid ikke så striks som figuren kan give indtryk af. Der er i virkeligheden tale om en vekselvirkning imellem de forskellige faser, ligesom man kan vægte betydningen af de enkelte faser forskelligt. Følger man metoden, der til tider kræver en hel del tålmodighed, ender man som regel med et godt resultat.

Bemærkning
Ovennævnte metode kan godt virke overflødig ved udvikling af midre programmer, men når man kommer op på lidt større og mere komplicerede programmer, kan man ikke arbejde uden en metode.
- Det kan simpelt hen ikke lade sig gøre at have det hele i hovedet.

Opgave - simpel lønberegning

1) Indtast COMAL-programmet.
2) Kør programmet og udregn nedenstående lønninger:

	Timetal	x	Timeløn	= Samlet løn
a)	4o	x	12,25	_____
b)	4o	x	27,85	_____
c)	4o	x	69,3o	_____
d)	4o	x	152,25	_____
e)	1o	x	152,25	_____

Programstruktur: Sekvens. PRINT, INPUT,:=,Simple talvariable

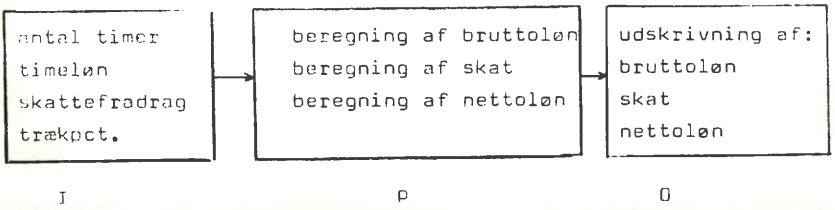
Opgave - løn- og skatteberegning

1. Problem
   Udarbejd en COMAL-program, der beregner en vilkårlig løn og skatten heraf og udskriver bruttoløn, skat og nettoløn.

2. Udtænkning af plan

   For at kunne løse problemet må vi vide
   - hvor mange timer, der arbejdes
   - hvor stor timelønnen er
   - hvor stort skattefradraget er
   - hvor høj trækprocenten er, og vi må
   - kunne foretage en beregning af data
   - kunne udskrive resultatet

Oversigt over vores løsningsforslag i et IPO-diagram



2b. Beskrivelse af planen

skriv	indtast timeløn (vejledende tekst)
indlæs	timeløn
skriv	indtast timetal (vejledende lekst)
indlæs	timetal
skriv	indtast skattefradrag (vejledende tekst
indlæs	fradrag
skriv	indtast træprocent (vejledende tekst)
indlæs	trækprocent
beregn	bruttoløn=timetalxtimeløn
beregn	trækgrundlag=bruttoløn-fradrag
beregn	skat=trækgrundlagxtrækprocent/loo
beregn	nettoløn=bruttoløn-skat
skriv	bruttoløn i kr. (vejledende tekst) og bruttoløn (værdien)
skriv	skatten udgør i kr. (vejledende tekst) og skat (værdien)
skriv	nettoløn i kr. (vejledendetekst) og nettoløn (værdien)

3. Udførelse af planen

```
COMAL-program
oolo // løn- og skattesystem
oo2o  // programmet er udarbejdet af:
oo3o //
oo4o Print "Indtast timeløn",
oo5o Input timeløn
oo6o Print
oo7o Print
oo8o
oo9o
oloo
ollo
ol2o
ol3o trækgrundlag:=bruttoløn-fradrag
```

```
o14o skat:=trækgrundlag*trækprocent/1oo
o15o
o16o
o17o
o18o
o19o
```

4. Efterkontrol af resultatet

Afprøv programmet og se om det virker!
Test eventuelt programmet ved at lave kontrolberegninger med papir og blyant!!

Programstruktur: Betingelser tekstvariabel, IF-THEN-ENDIF,
IF-THEN-ELSE-ENDIF.

Programeksempel:

Et program der kan beregne løn med eller uden tillæg.
- Programmet er en udvidelse af simpel lønberegning.

```
oolo // lønberegning med eller uden tillæg.
oo2o DIM svar$ OF 5
oo3o Print "Tast timetal",
oo4o Input timetal
oo5o Print "Tast timeløn",
oo6o Input timeløn
oo7o løn:=timetal*timeløn
oo8o Print "Indtast fast tillæg JA/NEJ"
oo9o Input svar$
oloo       IF svar$="JA" THEN
ollo       løn:=løn+79,5o
ol3o Print "Deres løn udgør, kr.   ", løn
ol4o End
```

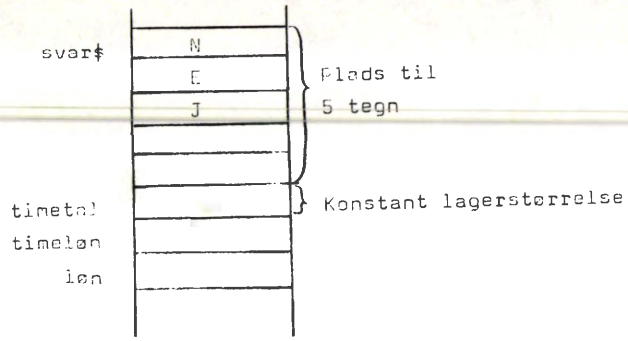
Forklaring af nye programfaciliteter

Simple talvariable og tekstvariabel

Simple talvariable oprettes automatisk når de nævnes i programmet, f.eks. i Input timeløn eller løn, og lagerstørrelsen er konstant.

Lagerstørrelsen ved tekstvariable skal erklæres særskilt. DIM svar$ OF 5 betyder, at vi skal dimentsonere (erklære/oprette) en tekstvariabel (svar$) der kan indeholde 5 tegn.

I arbejdslageret ser det således ud:



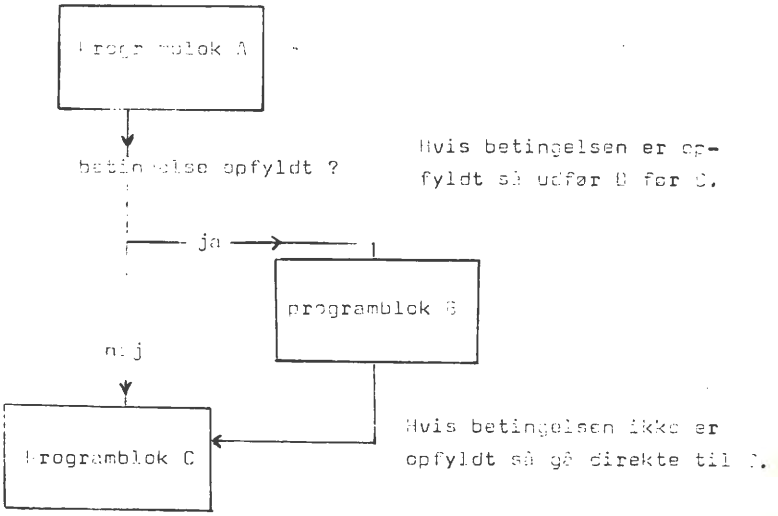
IF-THEN-ENDIF

Med denne programstruktur kan vi indbygge en valgmulighed i programmet:

Hvis udsagnet svar$="ja" er sandt (og det er det, når svars indeholder værdien ja), så skal vi beregne tillæg.

Hvis udsagnet derimod er falsk (og det er det, når svar$ ikke indeholder værdien ja), så skal vi ikke beregne tillæg og skriver blot en oprindelige løn ud.

Vi kan illustrere anvendelsen af betingelsessætningen på følgende måde



Opgave

1) Indtast programmet og kør det.
2) Indtast følgende værdier

	Timetal	Timeløn	Tillæg	Løn
a)	4o	79,5o	nej	_____
b)	39	79,5o	ja	_____

c) Find selv på andre eksempler.

3) Udarbejd en algoritmebeskrivelse (pseudokode).

Nye COMAL-ordrer	nye pseudokodenøgleord
DIM svar$ OF 5	erklær en tekstvariabel på 5 tegn
IF THEN	hvis så
ENDIF	sluthvis

Algoritmebeskrivelse for program: lønberegning m/u tillæg.
erklær en tekstvariabel (5 tegn)

hvis der skal betales tillæg så
beregn læg tillæg til lønnen
sluthvis

Eksempel: Billetprogram.

Problem:

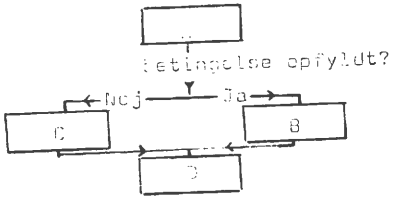
Programmet skal beregne og udskrive en vilkårlig billetpris
og sætte prisen for
voksne=fuld pris
børn=halv pris. Børn er dem under 12 år.

Algoritmebeskrivelse	COMAL-program
	oolo //Billetprogram
skriv billetpris (vejl.)	oo2o Print "Billetpris",
indlæs pris	oo3o Input pris
skriv alder (vejl.)	oo4o Print "Alder",
indlæs alder	oo5o Input alder
hvis alder < 12 så	oo6o If alder < 12 Then
børnebillet=pris/2	oo7o beløb:=pris/2
skriv pris børnebil.(vejl.)	oo8o Print "Børnebillet koster"
skriv børnebillet (værdi)	oo9o Print beløb
ellers	oloo Else
skriv pris voksenbil.(vejl.)	ollo Print "Voksenbillet koster"
skriv voksenbillet (værdi)	ol2o Print pris
sluthvis	ol3o Endif
	ol4o End

Pseudokodenøgleordet ellers ≈ COMAL - ordren Else

Forklaring af IF-TEHN-ELSE-ENDIF.

IF-THEN-ELSE-ENDIF er en udvidelse af IF-THEN-ENDIF.
Programmet tester på udsagnet alder< 12.
Er udsagnet sandt beregner og udskriver vi en børnebillet.
Er udsagnet falsk beregner og udskriver vi en voksenbillet.
Vi er altså i stand til at udføre to forskellige processer,
inden vi eventuelt går til en tredie proces. - Vi kan illustrere strukturen med følgendetegning



Opgave: Program overtimer.

Problem:

Udarbejd en algoritmebeskrivelse og et COMAL-program, der beregner løn og tager hensyn til overtimer. Den aktuelle timeløn er 71,25 kr. for de første 4o timer. Overtidsbetalingen er 71,25 + 5o%.

Algoritmebeskrivelse		COMAL-program	
		oo1o // Overarbejde	
skriv		oo2o	
indlæs		oo3o	
skriv		oo4o	
indlæs		oo5o	
beregn	overarbejde=timetal-4o	oo6o	
hvis		oo7o IF	
beregn		oo8o	
ellers		oo9o	
beregn		o1oo	løn:=timetal*timeløn+overarbejde*(71,25*5o/1oo)
sluthvis		o11o	
skriv		o12o	
		o13o	

Eksempel: En "smart" anvendelse af IF-THEN-ELSE-ENDIF.

Problem:

STATSSKATTEN for 1982 beregnes på følgende måde

- skattegrundlaget er den skattepligtige indkomst
- af de første 95500 kr. betales 14,4%
- af de næste 68600 kr. betales 28,8%, og
- af resten betales 39,6%

Der ønskes et program, der
- indlæser den skattepligtige indkomst for 1982
- beregner statsskatten og
- udskriver skattepligtig indkomst og statsskat.

COMAL-program

```
0010// Statsskat
0020 Print "Tast skattepligtig indkomst i kr.:",
0030 Input inkomst
0040 If indkomst<= 95500 Then
0050    statsskat:=0,144*indkomst
0060 Else
0070    If indkomst<=95500+68600 Then
0080       statsskat:=0,144*95500+0,288*(indkomst-95500)
0090    Else
0100       statsskat:=0,144*95500+0,288*68600+0,396*(indkomst-
           95500-68600)
0110    Endif
0120 Endif
0130 Print "Af en skattepligtig indkomst på ",indkomst," kr."
0140 Print "skal der betales  ", statsskat," kr. i statsskat."
0150 End
```

Det smarte er, at man kan indbygge en IF-sætning i en IF-sætning ..... og det kan man gøre mange gange!

Programstruktur: gentagelser. WHILE-DO-ENDWHILE.

Programeksempel:

Vi ønsker et program, der kan udregne løn (intet tillæg) for flere personer. For hver person indlæses navn, og antallet af timer og timeløn. Efterhånden udskrives navne og lønninger på linjeskriveren samtidig med, at vi kan se, hvad der foregår på skærmen.

Algoritmebeskrivelse	COMAL-program
	0010 // lønberegning for flere personer
erklær	0020 DIM navn$ OF 10
skriv	0030 Print "indtast dine data og afslut med STOP'"
skriv	0040 Print "Navn"
inslæs	0050 Input navn$
sålænge der er flere navne udfør	0060 While navn$<>"STOP" DO
skriv	0070 Print "Antal timer",
indlæs	0080 Input timetal
skriv	0090 Print "Timeløn",
indlæs	0100 Input timeløb
beregn	0110 løn:=timetal*timeløn
udskriv på linjeskriver	0120 Select Output "LP:"
skriv værdi af tekstvariablen	0130 Print navn$,
skriv s og løn (vejl.)	0140 Print"s løn: ",
skriv løn (værdi)	0150 Print løn
udskriv på dataskærmen	0160 Select Output "DS:"
skriv navn på næste person	0170 Print "Navn"
indlæs	0180 Input navn$
slutsålænge	0190 Endwhile
	0200 End

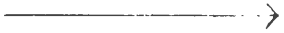
Nye pseudokodenøgleord		Nye COMAL-ord	
sålænge	udfør	While	Do
slutsålænge		Endwhile	
udskriv		Select Output (LP:angiver linjeskriver) (DS: angiver dataskærm)	

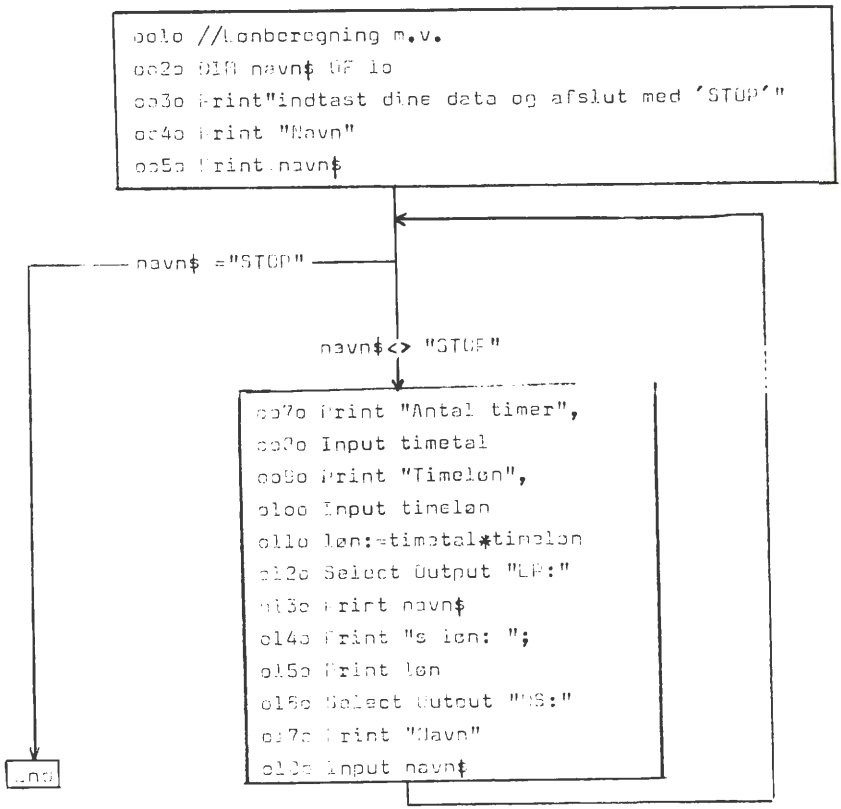
; bevirker, at linjerne o13o, o14o og o15o udskrives som een linje.

Forklaring af program: Lønberegning for flere personer.

Vi lægger en betingelse ind i programmet, og så længe betingelsen er opfyldt udføres det, der står under betingelsen.

Vores betingelser er opfyldt, sålænge værdien af navn$ er forskellig fra STOP. - Udsagnet navn$<>STOP er sandt.
Når vi indtaster STOP i stedet for et navn, får variablen navn$ værdien STOP, og så er betingelsen ikke længere opfyldt.
- Udsagnet navn$<>STOP er falsk, fordi STOP=STOP !!
Vi kan illustrere, hvordan programmet virker med følgende tegning




0010 //Lønberegning m.v.
0020 DIM navn$ OF 10
0030 Print"indtast dine data og afslut med 'STOP'"
0040 Print "Navn"
0050 Print navn$
navn$ ="STOP"
navn$<> "STOP"
0070 Print "Antal timer",
0080 Input timetal
0090 Print "Timeløn",
0100 Input timeløn
0110 løn:=timetal*timeløn
0120 Select Output "LP:"
0130 Print navn$
0140 Print "s løn: ";
0150 Print løn
0160 Select Output "CS:"
0170 Print "Navn"
0180 Input navn$
end

Opgave:

1. Indtast programmet og kør det nogle gange.
2. Udfyld Algoritmebeskrivelsen.
3. Indtast navnet Bartholomæussen. Hvad sker der ? - Ret programmet, så vi kan indlæse navnet!

Resumé:

Vi har i dette afsnit præsenteret en problemløsningsmetode, der er velegnet til programudvikling.

Vi har gennemgået tre fundamentale programstrukturer: Sekvens - Betingelser - Gentagelser.

Vi har i hele kapitlet brugt følgende COMAL-begreber (program-kommandoer):

`Print "tekst"`	- udskriver en tekstkonstant
`Print variabel`	- udskriver værdien, der er lagunder navnet variabel.
`Input variabel`	- opretter en simpel talvariabel, og inviterer til at læse værdier til den.
`variabel (eks.løn)`	- opretter en simpel talvariabel.
`DIM variabel$ OF tal`	- opretter en tekstvariabel med plads til det antal tegn, som tal angiver.
`Input variabel$`	- inviter til at læse værdier ind til den allerede oprettede tekstvariabel.

`IF {betingelse} TEHN`
`  {ordre`
`ENDIF`
	- hvis betingelsen er opfyldt, så udføres ordren.

`IF {betingelse} THEN`
`  {ordre`
`ELSE`
`  {ordre 2}`
`ENDIF`
	- hvis betingelsen er opfyldt, så udføres ordre 1. Ellers udføres ordre 2.

`WHILE {betingelse} DO`
`  {ordrerække}`
`ENDWHILE`
	- sålænge betingelsen er opfyldt udføres ordrerækken

Indledning:

I dette afsnit skal vi - i en konkret opgave - forsøge at omsætte de færdigheder, den metode og de begreber, som vi har gennemgået i det foregående forløb.

Vi skal desuden bruge to "færdiglavede" programmer for at løse vores opgave, Begge programmer ligger på jeres diskette.

Det ene program hedder MANUAL, og det skal bruges til at studere den arbejdsproces, som I skal

- beskrive
- programmere og
- få til at virke rigtigt.

I kalder programmet frem sådan: LOAD MANUAL [RETURN] ,
og sætter det igang med RUN [RETURN] .

Det andet program hedder MASKIN, og det er det, I skal bruge til at løse jeres programmeringsopgave med. I kalder programmet frem med LOAD MASKIN [RETURN] =
Programmet har en række delprogrammer/procedurer, men mangler hovedprogrammet. Det bliver jeres opgave at udarbejde hovedprogram (styringsdelen. Betydningen af hovedprogram og delprogrammer/procedurer gennemgås i næste afsnit (Procedurebegrebet).

Det er en god ide at lagre jeres resultat (fra gang til gang) under et andet navn. Det gøres således

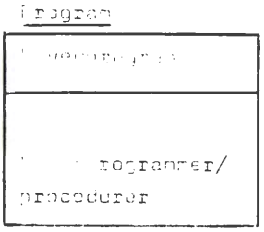
INIT [RETURN]
SAVE navn [RETURN]

Jeres program ligger nu på disketten, og det er en nem sag at kalde det frem igen.

Procedurebegrebet:

I større programmer er det en klar fordel at dele programmet op i mindre delprogrammer. - Det letter overskueligheden, og det muliggør udarbejdelse af nogel faste rutiner, som man kan udnytte flere steder i det samme program eller overføre til andre programmer.

Vi vil her dele programmet op i to områder:
Det ene område kalder vi for hovedprogrammet (styreprogrammet), og det er den aktive del af vores program.
Det andet område kalder vi for underprogrammer, eller procedurer, og det er den passive del af vores program.
Vi kan illustrere det med følgende tegning



Underprogrammerne/procedurerne har alle samme form, men forskelligt indhold. - De kan udføre forskellige ting, når de aktiveres fra hovedprogrammet.
Eks. på en procedure: PROC OPSTART
ordrer
ENDPROC OPSTART
Når vi skal kalde de forskellige procedurer i hovedprogrammet, gør vi det med samme kommando, nemlig EXEC.
EXEC er en forkortelse af det engelske ord execute, der betyder udfør. Udfør er den betegnelse vi vil anvende i algoritmebeskrivelsen (pseudokodenøgleord).
Eks. på procedurekald: EXEC OPSTART - vi henter proceduren OPSTART op i hovedprogrammet og udfører den.

Værktøjer til processtyring:

Vi har følgende procedurer til rådighed:

1. Opstart
2. Stepmotor (,)
3. DCmotor ( )
4. Magnet ( )
5. Forsinke ( )

og det er netop disse funktioner, vi har behov for til løsning af vores opgave.
Udover at procedurerne har forskellige navne (kan noget forskelligt), så skal de også tildeles nogle forskellige værdier (anføres inde i parantesen), når vi kalder dem.
- Kun proceduren Opstart skal ikke have tildelt nogen værdi.

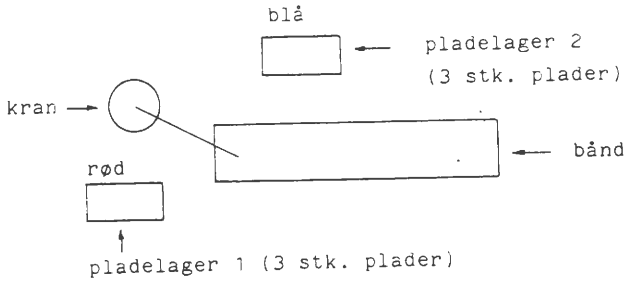
Det følgende giver en oversigt over de anvendte procedurer, som vi har til rådighed, og de værdier som vi kan tildele hver enkelt procedure.

Procedure	Programmering
1. Opstart ——→	EXEC opstart
2. Stepmotor -——→	EXEC Stepmotor (step, retning)
	step—→et heltal
	retning# —→ højre# ,venstre#
3. DCmotor ·····→	EXEC DCMotor (variabel)
	a)Kran: op# ,ned# , slut#
	b)Bånd: transportfrem ,slut
4. Magnet ——→	Exec Magnet (magnet#)
5. Forsinke — – →	Exec Forsinke (tal)

Eks. Vi vil have kranen til at dreje ca 90° til højre:
linjenummer EXEC Stepmotor (340,højre#)

Emne: Algoritmer

1. Opgaven går ud på at beskrive følgende proces:



Kranen skal skiftevis flytte en rød plade fra pladelager 1 til transportbåndet, som dernæst køres en sektion frem. Derefter flyttes en blå plade fra pladelager 2, og båndet køres en sektion frem.

Kranens udgangsstilling er som vist på skitsen placeret over transportbåndet og i en passende højde.
Du/I skal ved hjælp af de procedurervi har diskuteret beskrive ovennævnte proces på følgende måde.

1. Algoritmebeskrivelse med nøgleord.
2. Programskrivning
3. Indtastning af program
4. Afprøvning og fejlretning med håndteringsmodellen
5. Udskrivning af programmer på printer

Firtaktsmotor
Styret/kontrolleret af mikrodatamat

Ved Ole Karmark

1. Indledning

Det følgende er de væsentligste afsnit fra et projektforslag, fremsendt til Direktoratet for erhvervsuddannelserne.

Formålet med at offentliggøre dette forslag her i Dats, er et håb og forventning om at få reaktioner fra læserne. Vi håber på at få nogle kommentarer med på vejen. Derfor opfordres der til, at læserne kommer med forslag til ændringer, forbedringer eller kritiske bemærkninger. Sådanne kommentarer vil blive imødeset med stor spænding.

2. Projektarbejde med emnet motorstyring

Baggrund og indhold.

Begrundelsen for at sætte et arbejde om motorstyring iværk er den nye udvikling, der kan iagttages indenfor autoområdet. Udviklingen er karakteriseret ved en øget anvendelse

af elektroniske kredse, bl.a. en udtrakt brug af mikroprocessorer til motorstyrings formål, gør det muligt og åbenbart, at der må udvikles en fagdidatik, som kan anvise metoder og principper for behandlingen af denne sammenkobling: motor/mikroprocesser. En fagdidaktik, som fremlægger erfaringer og metoder, der udfolder hvorledes eleverne kan tilegne sig en fundamental viden og forståelse for de principper, der er grundlaget for processtyring, herunder motorstyrings eksemplet.

Grundlaget for at kunne kontrollere og styre en motor, dvs. den teori og de metoder, som en sådan motorstyring bygger på er hentet fra flere forskellige fagområder.

Fra naturlæren, den teoretiske forståelse for selve motorens virkemåde. Emner som mekanik, varmeteori, forbrændingsprocesser etc. fortæller tydeligt, at naturlæren er en forudsætning for en indsigt i motorens virkemåde.

Gennem matematikkens beregningsmetoder kan vi få kendskab til de kvantitative, et talmæssigt udtryk, for de sammenhænge mellem forskellige parametre, som er væsentlige for f.eks. god brændstofsøkonomi.

Fra elektronikken får vi viden om elektronikkomponenter, en viden, der er nødvendig for at kunne udforme elektroniske kredsløb. Disse kredsløb benyttes til at omforme, overføre og opsamle signaler fra motorens delsystemer.

Datalogien giver os metoder og teknikker, der sætter os i stand til hurtigt og effektivt at opsamle og sammenholde data fra motorsystemet. Ved hjælp af datamaten (mikroprocessoren) og matematiske beregningsmetoder fremlægges informationer om,hvilken fysisk tilstand motorsystemet befinder sig i. Ud fra programinstruktioner afgør datamaten om,der skal ændres ved nogle af motorens parametre. Afgørelserne træffes ud fra det synspunkt, at motoren skal arbejde optimalt.

3. Fagdidaktiske overvejelser

De didaktiske overvejelser over, hvorledes eleverne kan/skal arbejde med de problemstillinger, der er indeholdt i emnet kan foreløbig sammenfattes i følgende:

Organiseringen og struktureringen af undervisningsindholdet i emnet, tager udgangspunkt i at undervisningen skal have en problemorienteret profil. Eleverne skal have mulighed for at arbejde med fagområderne matematik/naturlære/datalære på en praktisk, eksperimenterende form, hvor undervisningsoplægget skal stimulere og motivere til induktive arbejdsformer. Formelagtig kan det udtrykkes som discovery metoden. Intentionen er at sætte eleverne i situationer, hvor konsekvenserne af deres handlinger i form at beslutninger om indstilling af tænding, justering af kaburator o.l. umiddelbart er iagttageligt som tilbagemeldinger fra de datamængder datamaten opsamler. I tilrettelæggelsen af sådanne eksperimenter koncentreres undervisningsindholdet delsom de centrale begreber og metoder i fagene matematik/naturlære/datalære, dels om den interesse og nysgerrighed eleverne kan forventes at have(få) for at afprøve deres hypoteser for, hvorledes en motor egentlig virker.

Projektet organiseres som et samarbejde mellem følgende fagområder:

Matematik
Naturlære
Datalære
Svagstrøm (elektrolinien)
Transportlinien

4. Målene med projektet

Projektets mål:

- at fremstille undervisningsmaterialer, der kan anvendes ved gennemførelsen af undervisningsforløb, hvor emnerne motorteori, naturlære, matematik og datalære indgår som en integreret helhed.

- at fremstille programmer til opsamling af informationer fra motorens væsentligste parametre, og programmer som anvendes til styring og kontrol af motoren.

- at fremstille elektroniske hjælpemidler, der kan videregive målinger af tilstande i motoren på en maskinlæsbar form.

- at gennemføre undervisningsforløb med det formål, at indhente erfaringer, som kan danne basis for en diskussion om en fagdidaktik for stofudvælgelse, tilrettelæggelse og gennemførelse af undervisningsforløb, hvor undervisningsmål fra de nævnte områder indgår.

5. Projektets forløbsfaser

Projektet opdeles i fire faser:

1. fase: Undersøgelse af de tekniske muligheder for implementering af et kontrol- og målesystem. Eksperimenter med løsningsmetoder for de opstillede krav til programmer og elektronisk udstyr.

2. fase: Udvikling af programmer og elektronisk udstyr, således at en egentlig styring af motoren muliggøres.

3. fase: Gennemførelse af undervisningsforløb, hvor der arbejdes med udvalgte eksempler.

4. fase: Formidling af projektarbejdet til skolerne.

I det følgende beskrives fase 1 og fase 2 nærmere. De to resterende faser beskrives ikke, da indholdet i disse er stærkt afhængig af hele projektets forløb og resultater.

6. Etablering af målesystemet

Indholdet i 1. fase er i hovedsagen følgende:

- etablering af procedurer og metoder for programmer og elektronisk udstyr.

- skitse mæssig beskrivelse af undervisningsøvelser og opgaver i fagene matematik, naturlære og datalære, set i relation til motorens virkemåde.

- udarbejdelse af præsentationsmåder for de indsamlede måledata.

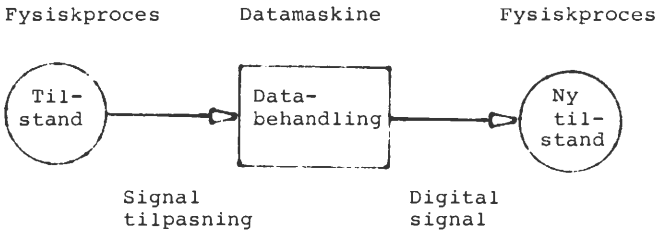
- vurdering af mulighederne for simulering af motorens virkemåde via datamaten (DFU).

Arbejdet består i at fremstille programmer og elektronisk udstyr, der gør det muligt at indhente måleresultater fra en række af motorens parametre:

1. Krumtapakslens position
   a. stemplets stilling
   b. omdrejning/tid
2. Kølevandstemperatur
3. Gasspjældets stilling
4. Trykket i indsugningsmanifolden
5. Trykket i den atmosfæriske luft
6. Indsugningsluftens temperatur
   a. luftfugtighed
7. Benzinforbrug
8. Motorbanken
9. Effekt - motorbremse
10. Udstødningsgassens sammensætning.

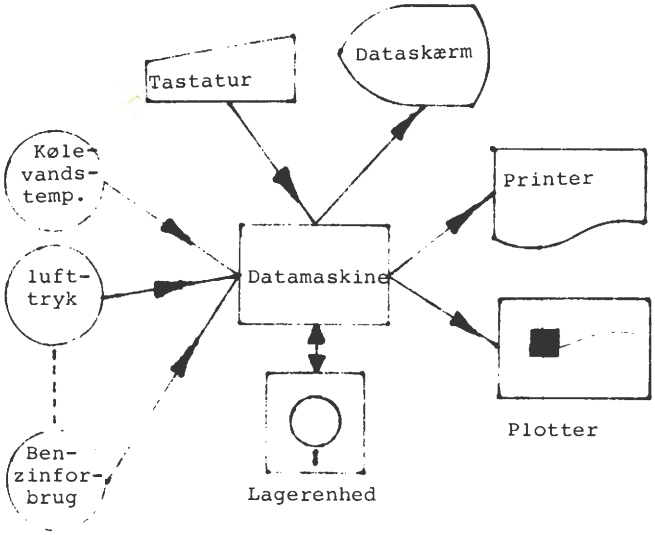
Nedenfor skitseres ved diagrammerne de planlagte systemer.

Figur 1



Figur 1 viser systemets principielle opbygning og virkemåde.

Figur 2



Figur 2 viser et systemdiagram med angivelse af de datamedier det færdige system vil kunne anvende.

På baggrund af de indsamlede måledata produceres informationer om tilstandene i de enkelte af motorens delsystemer. Informationerne om disse tilstande kan udskrives på de viste datamedier (fig. 2). Medierne vælges dels på baggrund af, hvordan informationerne kan fortolkes dels på baggrund af valgte præsentationsformer f.eks. kurver eller tabeller. Måledata udskrives efter ændringer eller justeringer af f.eks. tænding eller omdrejningsantal. Konsekvenserne af indgrebet kan derefter umiddelbart registreres på de udlæste uddata.

7. Faglige begreber og emner

Følgende begreber og emner fra naturlæren er planlagt at skulle behandles i projektet:

- fysiske enheder og størrelser
- måleteknik
- bevægelseslære - kinematik
- kraft og bevægelse
- arbejde og energi
- kinetisk molekyleteori
- luftarternes tilstandsligning
- varmeteori
- emner fra el-læren

Fra matematikken indgår følgende hovedbegreber:

- koordinatsystemer
- analytiskgeometri
- beskrivelse af funktionssammenhænge, afhængig og uafhængige variable
- ikke-liniære funktioner
- hyperbler
- konstruktion af pv-diagrammer
- potensbegrebet
- metoder og begreber for behandling af statistisk materiale
- trigonometriske funktioner
- virkelighed og matematiske modeller.

Fra datalæren følgende begreber:

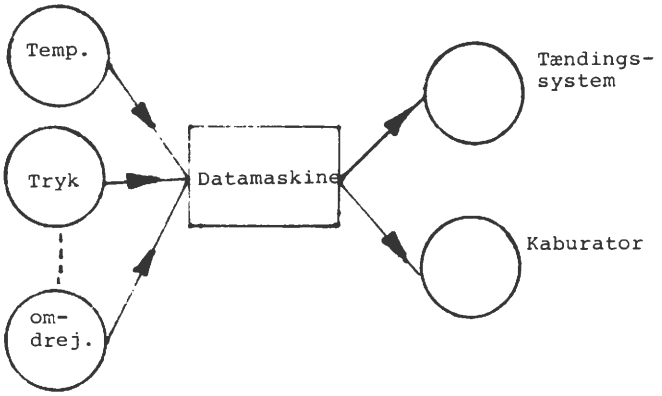
- databegrebet
- datastrukturer
- datamatiske modeller
- problemformulering
- algoritmer
- programmering
- beskrivelse af processystemer
- principper for kontrolsystemer
- måling af fysiske størrelser
- kørsel af færdige programsystemer
- tidstrodatabehandling

Motorteoretiske begreber er i høj grad bygget på naturlærens grundbegreber. Emner af central interesse - set ud fra et motorteoretisk synspunkt angives i det følgende:

- firtaktsmotorens arbejdsprincipper
- forbrændingsmotorens varmeteori
  - carnot-kredsprocessen
  - Otto-kredsprocessen
- forbrændingsmotorens nyttevirkning
- forbrændingsmotorens effekt
- forbrændingsprocessen
- motorens arbejdsproces
- brændstofsystemet
- tændingssystemet
- motorens lufttilførsel
- motorens belastningsforhold

8. Firtaktsmotor styret af datamat

Figur 3



Målet og arbejdet i denne fase er at nå frem til en egentlig automatisk styring af motoren (se fig. 3) og indeholder følgende opgaver:

1. Udvikling af systemet motor/datamat.
2. Dokumentation af systemet udarbejdes.
3. Beskrivelse af elevøvelser og elevopgaver, beskrivelse af undervisningsforløb.

Udfra de indhøstede erfaringer i første fase og de etablerede målepunkter udarbejdes procedurer for kontrol og styring af følgende variable:

1. Tændingssystemet
   - tændingstidspunktet.
2. Kaburatoren
   - forholdet mellem luft og brændstof

Målet med at lade datamaten overtage kontrollen af motoren, skal ses udfra et ønske om, under ethvert belastningsforhold, at optimere motoreffekten eller omvendt at minimere energiforbruget.

Der udarbejdes programmel, der kan overtage kontrollen og styringen af motoren. Herunder udarbejdes en bred dækkende dokumentation af systemets virkemåde, (realtidsprogrammering). De forudsætninger systemet bygger på beskrives ligeledes. Her tænkes først og fremmest på de matematiske beregningsmetoder, men desuden på de måleprocedurer der indgår.

De motorteoretiske aspekter, de problemer, der er forbundet med en automatisk styring af en motor beskrives.

A. Egebjerg Johansen m.fl.: Datalære 1-EDB i hverdagen. 168 sider.
Systime 1983

Anmeldt af Kaj Thrysøe.

Der er i dag en stor mangel på relevante grundbøger indenfor faget datalære. Litteratur findes der nok af, men meget af stoffet er behandlet ud fra en forældet teknik og metodik. Det er min opfattelse, at forfatterne til denne bog afhjælper en del af denne mangel. Allerede i bogens titel præciseres grundtemaet edb - "EDB i hverdagen" -, og det forekommer at være det mest relevante udgangspunkt for en kvalificeret undervisning i faget. Der spores en drejning væk fra en ensidig beskæftigelse med blokdiagrammer, systemdiagrammer m.v. over imod en mere direkte anvendelse af datamaten i undervisningen. Det synes at være en positiv udvikling al den stund, at eleverne først under et problemorienteret arbejde med datamaten udvikler et aktivt og engageret forhold til undervisningsindholdet i datalære.

Bogen består af 16 kapitler af varierende omfang. Hvert kapitel suppleres med opgaver i stoffet og afsluttes med et resumè. Bag i bogen findes et appendix, der giver en oversigt over nogle generelle størrelser (system- og programkommandoer, diagramtyper m.v.) og et stikordsregister.

Vi får præsenteret hard-ware og soft-ware på en let tilgængelig måde. Endvidere ser vi på anvendelser af edb ud fra forskellige synsvinkler: ergonomi og teknologiaftaler, arbejdsfunktioner før og efter indførelse af edb, datatransmission og dankort, det offentliges anvendelse af edb-registre og lovgivningen erom.

Til bogen er der udarbejdet en lærervejledning og et til undervisningsbrug ret omfattende og spændende applikationsprogramel

Der er et billetreservationsprogram, et faktureringsprogram, et lønsystemprogram og et databaseprogram. Desuden er der udarbejdet træningsprogrammer til programmeringsøvelser. Her indlæres programstruktur og simple programkommandoer til: indlæsning, udlæsning, beregning, betingelse og gentagelse.

Programmellet findes særskilt på en diskette, der kan erhverves til piccolo, SPC/L og COMET. Det må nok understreges, at bogen kun kan anvendes i begrænset omfang uden applikationsprogrammellet.

Bogen opererer - på virksomhedsplanet - stadig med de gammelkendte system-, black-box og mediediagrammer, men omfanget er ret behersket og eksemplerne virker rimelig nok.

Glædeligvis har bogens forfattere undladt blokdiagrammer i forbindelse med programmering, og i stedet gjort lidt ud af træstrukturer som systemmatiseringsværktøj.

Anvendelsen af træstrukturer synes jeg er relevant og nyttigt, fordi edb-maskinens specielle - lidt firkantede logik - indfanges meget sikkert med en sådan struktur. Det ville være ønskeligt, om forfatterne i en evt. fortsættelse af Datalære 1 udbygger anvendelsen af træstrukturer på andre områder, f.eks. struktureringaf et program i niveauer og moduler. Opbygningen af et program i moduler foretages relativt let, hvis det anvendte programmeringssprog har en procedurefacilitet til rådighed.
- Men det er jo bl.a. dette, der udmærker COMAL-sproget i forhold til det "gamle" BASIC-sprog.

Disse betragtninger skal imidlertid ikke overskygge det forhold, at Datalære 1-EDB i hverdagen, er en glimrende brugervenlig grundbog til indføring i edb-området.

Bogen er skrevet til handelsskolernes efg-basisår og er na-

turligvis rettet imod administrativ anvendelse af edb. Men dele af bogen kan sagtens anvendes indenfor andre undervisningsområder, f.eks. de tekniske skoler.

Mikrodatamaten som tegneredskab. Bogen om Monster.
af: Merete Barker, Per Jacobi, Ulrik Zimmermann.
(Borgens Forlag)

Anmeldt af Jørn Therkildsen.

I mit daglige arbejde anvender jeg ikke datamater og kender derfor heller ikke meget til dem. Mit personlige forhold til disse maskiner må nok betegnes som - hidtil køligt. Så får jeg udleveret denne her bog, læser og studerer den og ender med at blive positivt interesseret i maskinernes muligheder.

Hvordan gik det til?

Bogen er skrevet af tre forskellige mennesker med forskellige baggrunde. Per er arkitekt, Merete billedkunster og Ulrik ingeniør. Tre der til sammen fortæller, instruerer og giver gode råd i brugen af Monster.

Og hvad er Monster så? - Det er et edb-program, som kan tegne perspektivtegninger i mono eller stereo, hvis det ellers bliver betjent rigtigt. Et program, som kan spare sin bruger for megen tid og derfor også medvirke til fremstilling af tegninger/billeder, hurtigt og billigt. Det betyder ikke, ifølge Merete Barker, at man kan bruge Monster som en tegnemaskine, der automatisk kan udføre tegninger. Den er blot en programmerbar og hurtigt arbejdende "blyant", som ikke er særlig anvendelig, hvis brugeren ikke har kendskab til datamaskinen som tegneredskab og til tegning som visuelt udtryksmiddel.

Derfor er det også fint, at der er medtaget et afsnit om "form, farve og komposition". En let tilgængelig gennemgang af, hvordan vi mennesker opfatter billeder, som samtidig gør læseren lidt bedre til at udtrykke sig i billeder. Det afsnit har altså speciel informationsværdi for potentielle Monsterbrugere, som ikke tidligere har beskæftiget sig særlig bevidst med billedfremstilling. I et kapitel om Monster som tegneredskab er der indlagt et

regulært kursus i brugen af Monster. For mig at se, er det meget lettilgængeligt, og jeg ville ikke selv, med denne bog i hånden, være særlig nervøs ved at gå igang med en mikrodatamat som blyant.

Det jeg ikke umiddelbartkan vurdere er, hvordan det rent faktisk vil være at tegne med Monster. Når jeg bruger min blyant til at tegne med, går der en bestandig informationsstrøm, via mit syn, fra tegningen til min bevidsthed med bestandige korrektioner til følge. Tegner man med Monster, indskydes der en forsinkelse i denne kommunikation, idet jeg næppe kan korrigere midt i tegningsfremstilling.

På én måde er det altså en hurtig måde at tegne på, nemlig når det drejer sig om "slavearbejde" på en anden måde er det en utrolig "langsom" måde at arbejde på, idet man må vente på den færdige tegning, før en korrektion kan ske - og da i en ny tegning.

Jeg synes bogen med sin typografi og sine illustrationer er meget letlæselig. En spændende bog, som nok skulle kunne virke inspirerende - også i efg-undervisningssammenhænge.

4 * COMAL

af: Leif Pehrsson, Ester M. Christensen, Bjarne Aagaard.
(Forlaget Systime)

Anmeldt af Lone Verner Nielsen.

Jeg anmelder denne bog som underviser i datalære på efg, grafisk basisår på Aarhus tekniske Skole, og ser først og fremmest på bogen med disse øjne.

4 * COMAL er, som titlen siger en lærebog i programmering i COMAL-80 på gymnasieniveau eller tilsvarende. Enten til undervisningsbrug eller til selvstudium.

4-tallet i titlen må hentyde til, at der kan fås disketter til de følgende mikrodatamater: RC-piccolo, COMET/DITAMAT, New Brain og SPC/L.

Bogen er primært udarbejdet til mikrodatamaten RC-Piccolo, hvorfor alle konkrete oplysninger refererer hertil. Bagest i bogen findes en oversigt over de 3 øvrige mikrodatamaters tastatur samt systemkommandoer og COMAL-80 sætninger.

Anvender man imidlertid ikke en RC-Piccolo, men en COMET - som vi gør - kan man ikke bruge bogen direkte i sin undervisning. Man må som lærer først indsætte f.eks. de korrekte systemkommandoer, der hvor der er afvigelser fra Piccoloen.

Bogens indhold er koncentreret omkring de forskellige faciliteter, der findes i COMAL-80. Indledningsvis forklares, hvad en mikrodatamat er, på en letfattelig måde, rigt illustreret. Derefter følger et afsnit om operativsystemet med de mest anvendte systemkommandoer og forståelige forklaringer på deres betydning. Nu er vi kommet til side 23.

De sidste sider er en gennemgang af sætningerne i COMAL-80, med opgaver og eksempler. De fleste muligheder gennemgås.

Vi starter med indtastning af programmer, bestående af PRINT-sætninger. Princippet er - som ofte igennem bogen - "learning by doing". Jeg har selv erfaringer for, at eleverne bliver fanget, når stoffet præsenteres på denne måde.

Derefter skal vi køre programmer fra den tilhørende diskette. Her handler det også om systemkommandoerne LOAD og LIST.

Næste kapitel hedder "grundlæggende regler for programmering" og beskriver, hvordan man f.eks. ved hjælp af rutediagrammer udarbejder et program. Men "f.eks." kan slettes, idet man i det følgende udelukkende benytter sig af rutediagrammer.

Nu er vi kommet til side 38 og nu falder påstanden om, at resten af bogen bruges til præsentation af COMAL-80 i rækkefølgen: beregninger, variable, INPUT/PRINT, IF-sætninger, CASE-sætninger, procedurer, funktioner, løkkestrukturer, strenge, READ-DATA, filer, RANDOM-filer, procedurer (rekursive) og andre faciliteter som f.eks. CHAIN.

Igennem hele bogen er huskestoffet tydeligt markeret i rammer, ligesom kapitlerne afsluttes med en værdifuld oversigt over de introducerede faciliteter. Til nogle af kapitlerne er der desuden udarbejdet testprogrammer bestående af multiple choise opgaver i det aktuelle stof. Der lægges tydeligvis vægt på elevernes adfærd - de skal lære alle faciliteterne i COMAL-80. Der er et hav af eksempler, men hvad med forståelsen?

Sammenfattende kan bogen karaktereiseres om en god følgesvend igennem COMAL-80 manualen. Som i et supermarked - man skal udvælge de dele af programmeringssproget, som eleverne skal lære. De skal jo ikke være programmører.

Jeg mener, at programmeringen for eleverne er et middel til at få forståelsen af, hvordan datamaskinen arbejder. Og af programmeringssproget skal udvalgte dele præsenteres til brug ved løsning af problemer, der er relevante for elevernes hverdag eller

kommende arbejdssituation.

Hvis bogen skal anvendes i datalæreundervisningen bliver det som en bog, hvorfra læreren kan udplukke eksempelmateriale. Som sådan er den ganske udmærket, idet der er mange gode og anvendelige eksempler.

I selve datalæreundervisningen kan bogen ikke stå alene (hvilket forfatterne nok heller ikke forestiller sig.) Den tidligere på samme forlag udkomne "Datalære med mikrodatamater" af de samme forfattere (minus Knud Grosen) havde set fra dette synspunkt flere kvaliteter, idet den bl.a. også indeholdt stof til emnet "EDB og samfund".

Jeg kan ikke lade være med til slut at tænke på, hvor meget mere givende det kunne være at læse om andre læreres erfaringer med brug af en bestemt bog i et undervisningsforløb - fremfor en anmeldelse som denne. Lad dette være en opfordring!

Adresser

Kaj Thrysøe
TICA
Aarhus tekniske Skole
Halmstadgade 6
8200 Århus N

Jørn Therkildsen
Aarhus tekniske Skole
Halmstadgade 6
8200 Århus N

Lone Verner Nielsen
TICA
Aarhus tekniske Skole
Halmstadgade 6
8200 Århus N

Ulrich Lysdal Jensen
Klampenborgvej 232
2800 Lyngby

Ole Karmark
TICA
Aarhus tekniske Skole
Halmstadgade 6
8200 Århus N

Jørgen Meilgård
Skovtoften 9
Sundby
4800 Nykøbing Falster

TICA: Teknisk Informatik Center, Aarhus.