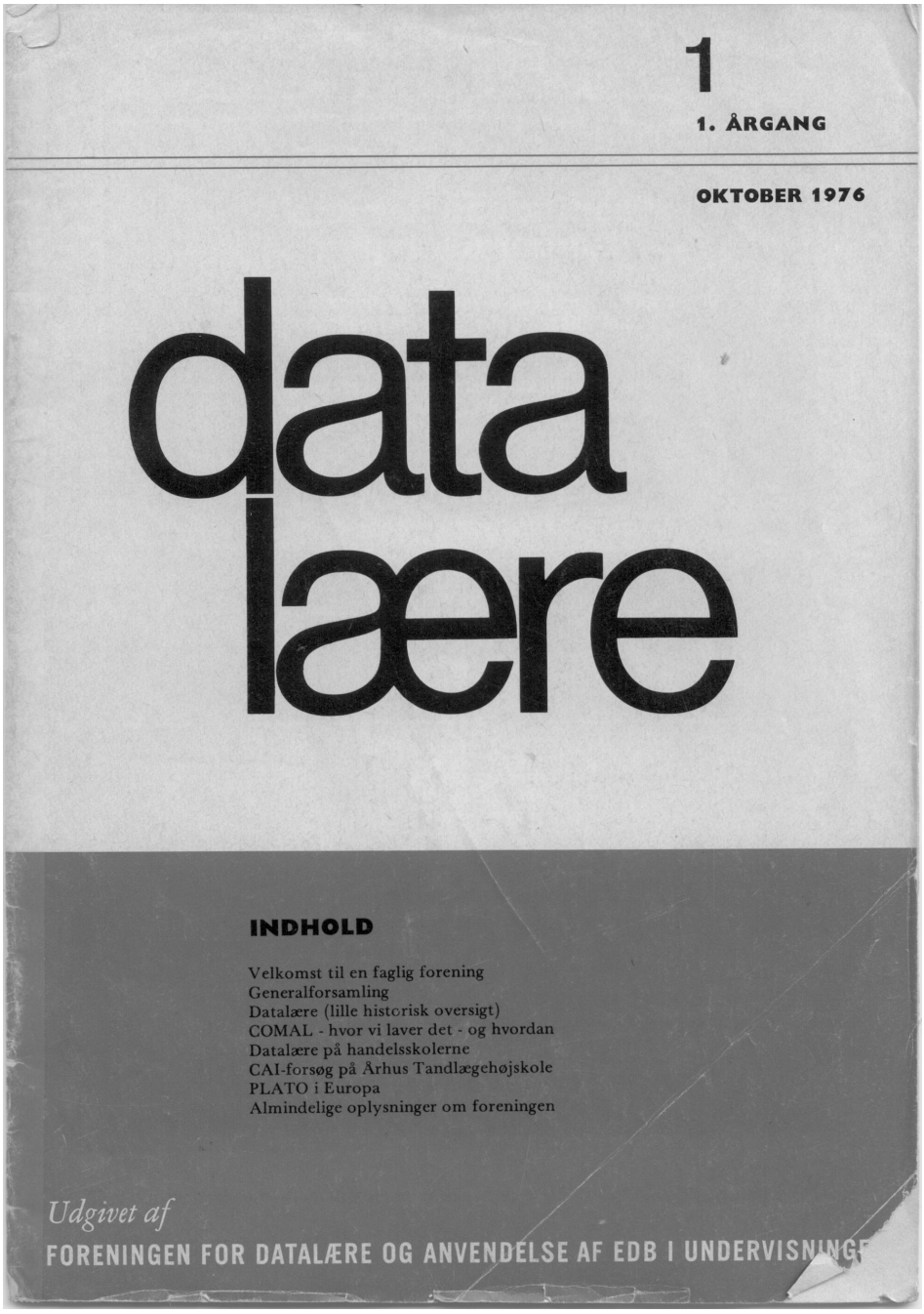
Dansk Datahistorisk Forening

COMAL fylder 40 år

Jubilæumsarrangement på Tapeten
Lørdag den 28. februar 2015



Genoptryk af nogle udvalgte artikler om COMAL m.v. fra de første numre af Datalæreforeningens blad **'datalære'**

Datalæreforeningen

Datalæreforeningen - eller som det rette navn var - 'Foreningen for datalære og anvendelse af edb i undervisningen' - startede sit virke den 12. marts 1976 ved en stiftende generalforsamling på Danmarks Lærerhøjskole i København.
Der var på dette tidspunkt grøde i tingene med edb i undervisningen. Betænkning nr. 666 om edb i det offentlige uddannelsessystem (kaldet 'Johnsen-rapporten') var udsendt i 1972 og anbefalede indførelse af datalære på alle trin i uddannelsessystemet. Det var i øvrigt Johnsen-udvalget, der opfandt begrebet 'datalære' som en bredere betegnelse end det videnskabsfaglige 'datalogi'.
Foreningen var bred i to dimensioner: Gennem fraktioner for de forskellige skoleområder (folkeskoler, gymnasier, tekniske skoler og handelskoler, højere uddannelser m.v.) søgte man bevidst samarbejde på tværs af skoleformerne om de opgaver, der var nye for alle. Og ud over det nye fag datalære havde man også fra starten øje for mulighederne med at bruge edb som værktøj og medie i undervisningen.

Bladet 'datalære'

I 1976 havde man hverken internet eller WWW, så man søgte hurtigst muligt at få lavet et blad for foreningen, der sammen med forskellige møder, kurser, foredrag, studieture m.v. kunne sikre kommunikationen mellem foreningens medlemmer.
Bladet udkom normalt med 4 numre om året, men lejlighedsvis kunne der være 6 numre i en årgang. I maj 1989 skiftede bladet navn fra 'datalære' til 'informatik og undervisning' og i juni 2001 endnu engang til 'IT og Undervisning'.
Det udgives fortsat af Danmarks IT-vejlederforening, som er den fortsættende forening på folkeskoleområdet.

Artikler fra de første numre af 'datalære' medtaget her

Kort om de udvalgte artikler fra perioden 1976 til 1982

Ved udvælgelsen af artikler er der selvfølgelig fokuseret på COMAL. Derfor er hovedparten også forfattet af Børge Christensen, som var en flittig skribent i bladets første år.
Ud over artiklerne om selve COMAL skrev Børge Christensen også en række artikler om algoritmer og anvendelsen af COMAL, f.eks. en simulation af kassekøerne i et supermarked og en række artikler om sortering inspireret af Niklaus Wirths bog 'Algorithms + Data Structures = Programs'
Artiklerne om sortering (5 i alt) blev peppet op med anskuelighed af principperne bag algoritmerne med levendegørelse og inddragelse af kendte personer. I 'Quicksort' er det f.eks. Teddy (Lang Petersen), Emil (Pedersen), Torsten (Alf Jensen), Erling (Schmidt) og Torben (Høirup), der skal løbe om kap fra en kulkran i Odense havn mod Ejby overdrev. Men den bedste historie gemmer sig i den medtagne artikel 'Teddy og Torben flytter skilte', der handler om 'straight selection' og 'straight exchange'.
Algoritmen anskueliggøres ved en historie om to mænd, som flytter nogle afstandsskilte, der er sat op for at fange motorcyklister, der kører for hurtigt. 'Teddy' er selvfølgelig Teddy Lang Petersen, der var redaktør af 'Datalære', og 'Torben' er Torben Høirup, der var kasserer i Datalæreforeningen og leder af DOS - Datacenteret ved Odense Skolevæsen.
Men hvad måske ikke alle ved, var, at den unavngivne motorcyklist, der var sur på skiltene og lavede rod i dem, må have været Børge Christensen, som i en moden alder kastede sig ud i et selvrealiseringsprojekt og anskaffede en nogenlunde stor japansk motorcykel.
Engang på en tur fra Tønder til møde i Odense blev Børge nemlig stoppet af politiet, der efter sigende blev noget forbløffet over, at der i stedet for en ung, fartglad lømmel dukkede en pæn, ældre lektor op, da hjelmen kom af. Det skulle dog ikke have reduceret bøden …

Professor H.B. Hansen var også hyppig bidragsyder til bladet i de første år. Bl.a. kommer han i 1977 de betrængte rutediagrammer til undsætning med en artikel, hvor han argumenterer for, at de er et udmærket problemløsningsværktøj, og i 1978 skriver han om et grafisk system i COMAL, som de på Roskilde Universitetscenter har udviklet til RC7000 med en tilkoblet Tektronix grafisk skærm og plotter.
Her er medtaget H.B. Hansens behandling af 'klassikeren' om Tårnet i Hanoi. Men på dette tidspunkt kan COMAL endnu ikke håndtere rekursive procedurer og må lide den tort, at H.B. Hansen omtaler det således: "Med så primitive sprog som Basic og Comal er dette imidlertid sjældent muligt, og man må derfor omskrive den rekursive algoritme til en iterativ algoritme."
H.B. Hansen løser så problemet i COMAL ved at lave sin egen stak, der kan holde orden på tingene og konkludere, at Tårnet i Hanoi "er tilstrækkelig kompliceret til, at man ikke umiddelbart kan se løsningen, men samtidig ikke mere kompliceret, end at løsningen virker indlysende når man får den forklaret' - og dermed har stor pædagogisk værdi.
Som en slags afrunding af H.B. Hansens artikel om Tårnet i Hanoi er lige medtaget et program i COMAL 80, som løser problemet med de rekursive procedurer, der blev indført her. Siden er mærket 'annonce', for Regnecentralen bragte i en årrække et 'blad i bladet' som en annonce.
I de første år hed dette RC7000 ÅREN, men skiftede fra november 1979 navn til RC-INFO.

Torben Høirups artikel om anvendelse af edb ved Odense kommunale Skolevæsen giver ud over et tidsbillede af pionertiden også i afsnittet 'COMAL på DOS' et kig ind i tilblivelsen af COMAL 75 til RC7000.

Fritz G. Knudsen refererede fra Datalæreforeningens seminar på Esbjerg Seminarium, hvor man ville se på de pædagogiske og praktiske muligheder med de nye 'mikroer'. Titlen på indlægget indikerer, at 'mikroerne' *var* nye - alene betegnelsen 'Mikroprocessorbaserede smådatamater' - signalerer, at sprogbrugen om de kommende PC'er ikke var faldet på plads.

COMAL - hvorfor vi lavede det - og hvordan

Rutediagrammer?

Det hele begyndte for ca. 3 år siden, da jeg en dag gik ind på Benedict Løfstedts kontor på Datalogisk Institut, Aarhus Universitet. Benedict underviser bl.a. begyndere i datalære på Aa.U. Jeg kom egentlig blot for at hente mig nogle »opgavefiduser». Alle, der har forsøgt at lave opgaver i »algoritmer» eller »datalære» for hf, gymnasiet eller seminariet, ved, hvor svært det er at lave nogle, der ikke er enten for banale eller for svære. På det tidspunkt havde jeg mest lavet opgaver af den slags, hvor man skal tegne et rutediagram for en eller anden proces, og derpå skrive et BASIC-program efter diagrammet. Jeg havde også opdaget, at rutediagrammer let bliver uoverskuelige, hvis de beskrevne processer ikke er helt banale, og at denne uoverskuelighed afspejler sig i BASIC-programmerne; men jeg blev alligevel noget forbløffet, da Benedict ligeud erklærede, at jeg overhovedet ikke burde bruge rutediagrammer - og i hvert fald slet ikke på den måde, jeg hidtil havde gjort.

Hvorfor ikke?

Jeg spurgte ham forsigtigt - jeg stod overfor en mand af professionen, og ville helst ikke tage mig for dum ud - hvad der var i vejen med rutediagrammer? Svaret på dette spørgsmål overraskede mig også. Det gik nemlig ud på, at man kan lave alt for meget med rutediagrammer! Man kan blive ved med at lappe på et rutediagram ved at føje nye veje og kasser til det, og det er netop denne strukturløshed, som på en gang gør rutediagrammer både forførende og farlige. Man kan i ordets bogstaveligste forstand blive »lokket på afveje», når man arbejder med dem. Nu bagefter, når jeg tænker på det, burde det egentlig være indlysende for en matematiker, at det er sådan. I næsten alle de emner, vi har med at gøre, bestræber vi os på at beskrive tingene ved hjælp af bestemte strukturer. Tænk blot på gruppeteorien. Diagrammernes strukturløshed afspejler sig naturligvis i BASIC-programmerne, som med den nævnte arbejdsform ikke bliver andet end verbale repræsentationer af rutediagrammer.

Det nævnte møde gav anledning til mange samtaler og megen skriven frem og tilbage, og efterhånden opstod tanken om at udvikle et bedre beskrivelsesmiddel for algoritmer og programmer end rutediagrammer og BASIC. Et beskrivelsesmiddel, som - med Edsger Dijkstras ord - ville kunne »egne sig til at befordre den menneskelige tanke».

BASIC + PASCAL

Næsten fra begyndelsen blev det klart for mig, at et sådant sprog måtte bygge på ideerne i det af Wirth beskrevne sprog PASCAL. På den anden side ville det sikkert være klogt at beholde nogle af faciliteterne i BASIC. For det første eksisterede BASIC - en kendsgerning, der ikke var til at overse - og for det andet har dette sprog visse faciliteter, som jeg nødigt vil undvære i begynderundervisningen. Jeg tænker her først og fremmest på den konverserende tekst-editor, som findes i BASIC-fortolkere, og den simple måde, på hvilken man kan håndtere ind- og uddata.

Bedre styrestrukturer

Jeg koncentrerede mig derfor om at få skabt nogle bedre styrestrukturer end de meget primitive, som findes i BASIC. Som læseren formentlig ved, er BASIC et linjeorienteret sprog: Fortolkeren udfører programmet linje for linje, og tegnet »vognretur» danner skilletegn mellem de enkelte sætninger i programmet. Den eneste blokstruktur i BASIC er FOR..NEXT-løkken, hvor FOR-sætningen danner blokkens begyndelsessætning, mens NEXT-sætningen afslutter blokken. Man har ikke som i fx. PASCAL, ALGOL eller PL/M nogle bestemte ord, der kan afgrænse visse programdele, som hver for sig består af flere sætninger (i PASCAL og ALGOL er det ordene: BEGIN og END). Løsningen på dette problem fik jeg af Benedict, som foreslog mig at indføre de par af styreord, der nu findes i COMAL (altså IF .. (ELSE)..ENDIF, WHILE..ENDWHILE, REPEAT ..UNTIL og SELECT..ENDCASES). Sammen med de relevante Boolske udtryk skulle disse par af ord kunne styre udførelsen af så store programdele, man måtte ønske, på lignende måde som FOR..NEXT gør det. Ligeledes måtte man naturligvis kunne indlejre blokkene i hinanden på samme måde, som man kan lægge FOR..NEXT-

løkker inden i hinanden (i det mindste til en vis dybde).

Første version

Herefter beskrev jeg en første version af sproget, som jeg døbte COMAL (COMmon Algorithmic Language) i en artikel, der bl.a. blev trykt i International World of Computer Education. Hvilken lykke! Nu måtte firmaerne da stå på nakken af hinanden for at få skrevet COMAL-fortolkere til deres minier! Og frem for alt det udmærkede danske firma Regnecentralen, der er kendt for sine fornemme traditioner i udvikling af programmel. Det tør nok siges, at jeg kom ned på jorden igen. Bevares, man fandt mit forslag »interessant» - og gik derpå over til dagsordenen.

Jeg er i skrivende stund aldeles overbevist om, at hvis ikke den række af heldige omstændigheder, som jeg skal redegøre for i det følgende, var indtruffet, var COMAL aldrig kommet længere end til de tryksager, som jeg kolporterede til sagesløse personer i tide og - efter nogens mening ganske givet - i utide.

Begyndelsen

Da vi fik vort datamatiske system her til Tønder Statsseminarium, blev det udstyret med den version af BASIC, som er udviklet af den Amerikanske fabrikant af NOVA'en (RC7000), Data General. Brugere af dette programmel vil vide, at det - mildt sagt - er behæftet med visse fejl og mangler! En stor del af disse fejl er blevet rettet af den danske forhandler, A/S Regnecentralen, men dog langtfra alle. I 1974 begyndte to af vore brugere, lærerstuderende Knud Christensen og gymnasieelev Per Christiansen, at interessere sig for at skrive maskinprogrammer for NOVA'en, og de udviklede efterhånden så godt et kendskab til denne datamat, at jeg i begyndelsen af 1975 bad RC om nogle af kildeteksterne til BASIC-fortolkeren, så vi kunne forsøge at rette nogle af de fejl, der stadig hjemsøgte vort system endog ved højlys dag.

COMAL til SOS

Dette ønske blev imødekommet, og i foråret 1975 fik vi de fleste af teksterne til fortolkeren. Den ene af de studerende - Per Christiansen - begyndte at studere de fremsendte tekster, og efter et par ugers forløb var han kommet til den konklusion, at det ikke blot skulle være en overkommelig opgave at rette de fleste af fejlene, men at det heller ikke skulle være umuligt at ændre programmerne, så vi kunne komme til at køre COMAL. Han tog fat på arbejdet med en utrolig energi og stædighed, og arbejdede med projektet hele sommerferien. I begyndelsen af august kom Knud Christensen til, og i midten af august kørte det første COMAL-program. Sammen fortsatte de arbejdet med fortolkeren resten af efteråret, og samtidig med at COMAL blev udbygget, blev de fleste af fejlene i den oprindelige fortolker rettet. Resultatet var, at vi ved juletid havde en COMAL-fortolker kørende, og at denne fortolker var betydelig mere robust end den oprindelige BASIC-fortolker. Vi kaldte denne version COMAL I, og nogle af læserne vil kende den, som den første vi sendte ud til venner og bekendte. I COMAL I manglede stadig væsentlige dele af det oprindeligt definerede COMAL, bl.a. muligheden for at bruge lange variabelnavne. Denne og en række andre faciliteter, der i blandt nogle meget bekvemme kommandoer, blev føjet til i løbet af vinteren 1976, i marts måned var den version, COMAL II, som de fleste brugere af RC 7000 kender, færdigudviklet. Jeg skylder de to studerende at fortælle, at dette udviklingsarbejde er foregået på en lille datamat uden nogen form for baggrundslager. Professionelle programmører, som har erfaret om sagen, har på det nærmeste nægtet at tro, at noget sådant kan lade sig gøre. Stillet overfor kendsgerningerne, har de naturligvis måttet tro om, det kan lade sig gøre - for nogen!

COMAL til RDOS

Det skal også nævnes, at samtidigt med at stand-alone versionen er blevet udviklet i Tønder, har Knud og Per været dels i København hos RC og dels i Odense hos DOS for at udvikle COMAL-fortolkeren til kørsel under RDOS.

Siden julen-75 har vi haft et meget fint samarbejde med lederen af DOS, Torben Høirup, som har ofret mange timers arbejde på at afprøve systemet og har ydet væsentlige bidrag til aflusningen af RDOS-versionen.

Hvor er vi nu

Regnecentralen har nu overtaget hele projektet, og siden 1. juli i år har man arbejdet på at få COMAL-fortolkeren til at køre under RC's eget MUS-operativsystem. Jeg venter mig meget af denne version, idet MUS-operativsystemet er kendt som et af verdens bedste.

Er lærerne nøjsomme?

Nu går jeg så og undrer mig! Hvordan kan det egentlig være, at så mange lærere er så nøjsomme? Man køber en minidatamat i dyre domme,

og så fylder man BASIC i den! Med de erfaringer, vi har høstet her i Tønder, er BASIC nærmest at betragte som en fornærmelse mod både datamaten og brugeren. Jeg synes næsten, det vigtigste resultat af vort arbejde er, at vi har bevist, man ikke behøver nøjes med så lidt. Jeg er begyndt at arbejde med lommedatamater i den sidste tid, og jeg er ikke langt fra at mene, at de bedste af disse er BASIC overlegne. I hvert fald til rene numeriske algoritmer. De er forresten også betydelig mere nøjagtige og prisforskellen er påfaldende på en lommedatamat og en mini. Skal en investering i en mini have mening, må man efter min opfattelse forlange adskilligt mere end det, man kan få ud af et BASIC-system.

Når man tænker på den udvikling, der har fundet sted på dette område, må man have lov at sige, at det er meget længe siden, BASIC var sagen, og den eneste grund til, at det stadig overlever, må søges i den brugerskare, for hvem datamater er noget forholdsvis nyt og imponerende, og som endnu ikke har fundet ud af, hvad man bør kunne forlange af et system til et par hundrede tusinde kroner. Misforstå mig nu ikke. COMAL er ikke hele svaret på alle den elementære undervisnings datalære-problemer. Et sådant svar findes næppe nogensinde. Udviklingen fortsætter. Men jeg er ikke mere ene om at mene, at det er langt bedre end BASIC til vore formål.

For øvrigt er jeg begyndt at spekulere over, om man ikke - nå, nej, det kan være til en anden gang.

Børge.

For dem, der vil orienteres om COMAL, findes der et lille skrift kaldet: COMAL II, Addendum to Extended Basic Users Manual. Dette skrift kan erhverves mod indsendelse af kr. 1,50 i frimærker til:

Seminarielektor Børge Christensen
Tønder Statsseminarium
6270 Tønder.

DATALÆRE på handelsskolerne

Uddannelsens form.

HX og HHX.

Handelsskolerne tilbyder en treårig uddannelse i forlængelse af den obligatoriske skolegang. Af en årgang unge vælger cirka 20% en handelsskoleuddannelse. De fleste forlader handelsskolerne efter et års uddannelse med handelseksamen (HX) og går ud til en lærlingekontrakt i erhvervslivet. På nogle handelsskoler findes et handelsgymnaisum og elever med handelseksamen kan supplere med den toårige uddannelse til højere handelseksamen (HHX). Nogle steder findes desuden en etårig HHX for studenter og HF'ere. Ud over at være en erhvervskompetencegivende eksamen giver HHX adgang til videregående uddannelser på handelshøjskoler, universiteter, edb-assistentskoler m.m.

EDB på HX.

På HX tilbydes et 80-timers edb-grundkursus. Det indgår ikke i det normale skema, men tilbydes som ekstra fag. Hvor mange elever der vælger edb-grundkursus på den enkelte skole afhænger meget af, hvilken måde faget bliver tilbudt på. Nogle steder vælger over halvdelen, andre steder næsten ingen edb. I gennemsnit vælger cirka 40% edb-grundkursus. I planer for fremtiden indgår , at edb skal være obligatorisk med mulighed for ekstra tilvalg.

EDB på HHX.

På HHX findes datalære som fag på andet skoleår med 4 timer ugentlig. På den økonomisk-administrative gren er det obligatorisk, på den rene økonomiske gren og på den sproglige gren kan datalære vælges som det ene fag, der skal vælges ud over de rent obligatoriske. I gennemsnit har cirka 20% af en årgang datalære.

EDB-udstyr.

Ved starten af skoleåret 1976-1977 har samtlige handelsskoler edb-udstyr. Minimum er en TTY (skrivemaskine + papirbåndsterminal). Nogle har en terminal med skærm, stregmarkeret kortlæser og linieskriver, enkelte har eget edb-anlæg. Terminalerne kører på andre skolers anlæg, på Kommunedata eller andre steder.

Der har været forlydender fremme om, at det i fremtiden kunne blive muligt at køre på de anlæg, der står på edb-assistentskolerne i Ålborg,

Teddy og Torben flytter skilte

På en vejstrækning i Odense har politimesteren ladet opstille en række skilte med fra 25 til 50 m mellemrum. Disse skilte skal benyttes til målinger, som skal gøre det lettere for politiet at idømme motorcyklister fartbøder. En person, der er uvenlig stemt overfor denne plan, har i en sen nattetime ombyttet skiltene på en del af strækningen, så de nu optræder i denne rækkefølge:

225 285 400 315 425 500 475 200 350 260

Som tilsigtet af gerningsmanden afstedkommer denne ombytning en del forvirring i politiets målinger, og efter at en hel mc-klub er sluppet uantastet igennem, beslutter man straks at sende to medarbejdere fra Fyns amt, Torben og Teddy, ud med det opdrag at føre skilterækken tilbage til dens oprindelige status.
De to udmærkede medarbejdere beslutter sig for følgende fremgangsmåde: De slår en lille, hvid pæl i jorden ved det første skilt (225), og Teddy bliver stående ved det og venter. Torben skriver tallet på skiltet på en seddel og cykler derpå ud ad vejen. Hver gang, han kommer til et nyt skilt, sammenligner han det tal, der står på skiltet, med tallet på sedlen. Hvis tallet på skiltet er mindre end tallet på sedlen, vinker han til Teddy, der cykler ud til ham. I vort eksempel sker dette ved skiltet med påskriften 200. Teddy venter nu ved dette skilt, mens Torben cykler videre med en ny seddel, hvorpå tallet på det nye skilt er skrevet(200), for at se efter, om der skulle være skilte med endnu lavere talværdier. Da det viser sig ikke at være tilfældet, bytter Teddy og Torben det skilt, der er afmærket med den hvide pæl, om med det skilt, Teddy står vagt ved. Resultatet bliver, at skiltene nu står sådan:

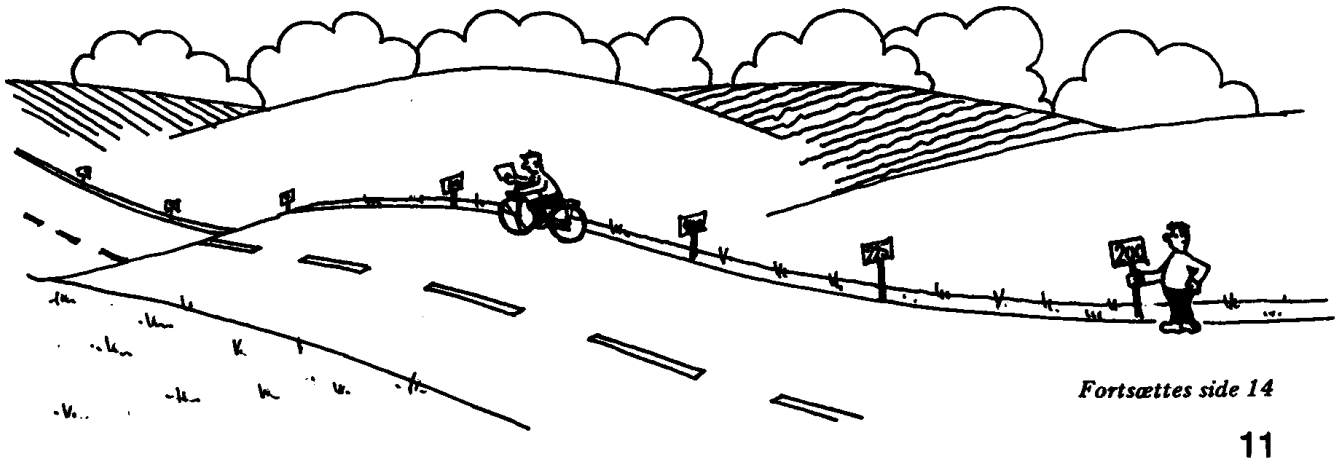
200 285 400 315 425 500 475 225 350 260

Den hvide pæl bliver nu flyttet hen til skiltet med påskriften 285, og Teddy venter ved dette skilt, mens Torben cykler ud ad vejen med en seddel, hvorpå tallet 285 står skrevet. Ved skiltet med påskriften 225 standser han, fordi 225 er mindre end 285, og vinker til Teddy. Teddy cykler ud til dette skilt, mens Torben cykler videre med en ny seddel, hvorpå der står 225. Resten af turen giver ikke noget nyt resultat, og skiltet med den hvide pæl (285) ombyttes med det, Teddy står vagt ved (225). Resultatet er følgende:

200 225 400 315 425 500 475 285 350 260

Næste gang sættes den hvide pæl ved skiltet med påskriften 400, og læseren kan selv overbevise sig om, at vagtposten denne gang kommer til at flytte plads to gange: først ud til 285 og derpå til 260, og at 400 sluttelig bliver ombyttet med 260:

200 225 260 315 425 500 475 285 350 400



Fortsættes side 14

11

Fortsat fra side 11

Således fortsætter de to flittige medarbejdere ved Fyns amt, og da dagen går på hæld, står skiltene, som de skal:

200 225 260 285 315 350 400 425 475 500

Politimesteren kan atter betragte sit værk med tilfredshed, motorcyklisterne har endnu en grund til at udvise lovmedholdelig adfærd, og læseren kan gå igang med at studere den algoritme, Teddy og Torben har benyttet.

Øvelse
Gennemfør de resterende trin af sorteringsprocessen, som fører frem til slutresultatet ovenfor. Læg mærke til, hvor "vagten" står, når den sidste ombytning udføres.

Der er tale om ialt 10 skilte, og vi kan tænke os, at deres pladser er nummererede 1, 2, ..., 10. Tallet på skiltet, der står på plads nr. *i*, vil vi betegne *tal(i)*, og de to personer vil vi i fortsættelse af gammel tradition for terminologi i forbindelse med teoretiske vejarbejder kalde *A* og *B*. Fremgangsmåden ovenfor kan derefter beskrives således:

```
for i:=1 til 9
  hvid pæl sættes ved nr. i
  A stiller sig op ved nr. i
  B noterer tal(i) på sin seddel
  for j:=i+1 til 10
    hvis tal(j) er mindre end tallet på sedlen så
      A flytter hen til nr. j
      B noterer tal(j) på sin seddel
    slut
  skift j
  skiltet ved nr. i byttes med det skilt, A står ved
skift i
```

Den således beskrevne proces til sortering af en række tal, kan man meget let programmere. Idet vi tænker os, at ialt *n* tal er tildelt vektoren *tal*, får vi ved at benytte algoritmen ovenfor følgende program:

```
for i:=1 til n-1          //hvid pæl ved skiltene//
  a:=i                    //A ved skilt nr. i//
  b:=tal(i)               //B skiver op//
  for j:=i+1 til n        //B kigger på de følgende skilte//
    hvis tal(j)<b så      //er tal på skilt mindre end
                            tal på seddel?//
      a:=j; b:=tal(j)     //så flyt A hen til j og
                            lad B skrive en ny seddel//
    slut
  skift j                 //B til næste skilt//
  tal(a):=tal(i)          //byt det skilt, A står ved,
  tal(i):=b               //med skilt ved hvid pæl//
skift i                   //flyt hvid pæl//
```

Under processen foretages for hvert trin udvælgelse af et element (det hidtil mindste), som derpå ombyttes med det, der nu står forrest (afmærket med den hvide pæl ved nr i). Algoritmen har derfor fået navnet *straight selection*, og herunder er vist en procedure til udførelse af den sammen med et program, der aktiverer proceduren:

```
0010 PROC STRSELEC
0020 FOR I=1 TO N-1
0030   LET A=I; B=TAL(I)
0040   FOR J=I+1 TO N
0050     IF TAL(J)<B THEN DO
0060       LET A=J; B=TAL(J)
0070     ENDIF
0080   NEXT J
0090   LET TAL(A)=TAL(I); TAL(I)=B
0100 NEXT I
0110 ENDPROC   STRSELEC
0120 REM //--------------//
0130 REM ** HOVEDPROGRAM **
0140 INPUT "ANTAL ELEMENTER: ",N
0150 DIM TAL(N)
0160 FOR K=1 TO N
0170   LET TAL(K)=INT(RND(0)*900+100)
0180   PRINT TAL(K);
0190 NEXT K
0200 EXEC STRSELEC
0210 PRINT
0220 FOR I=1 TO N
0230   PRINT TAL(I);
0240 NEXT I
0250 END
```

Øvelse
Herunder er vist en procedure, der på mange måder minder om straight selection. Den går sædvanligvis under navnet "bubblesort", men kaldes også *straight exchange*. Prøv at sortere skiltene ved hjælp af bubblesort. Hvorfor kan skiltesorteringen udføres af én mand, når bubblesort bruges?

```
0010 PROC BUBLSORT
0020 FOR I=2 TO N
0030   FOR J=N TO I STEP -1
0040     IF TAL(J-1)>TAL(J) THEN DO
0050       LET X=TAL(J-1); TAL(J-1)=TAL(J); TAL(J)=X
0060     ENDIF
0070   NEXT J
0080 NEXT I
0090 ENDPROC   BUBLSORT
```

Vedr. manuskripter til Datalære

Maskinskrevne manuskripter er velsete. Håndskrevne manuskripter må være letlæselige, og kun den ene side af papiret bør forsynes med tekst.
Ønske om bibeholdelse af afvigelser fra den "gængse" retskrivning og/eller tegnsætning bedes angivet på manuskriptet.

OBS! OBS!

Stof til næste nummer af bladet skal være redaktionen i hænde senest mandag den 16. oktober 1978.

Tårnet i Hanoi

analyse af en rekursiv algoritme

af H. B. Hansen

Mange problemer forklares lettest hvis man udtrykker dem rekursivt. Tænk f. eks. på den binære søgning. Den handler om at finde et bestemt element i en mængde af elementer, der er ordnet efter størrelse. Algoritmen kan udtrykkes således:

1. Prøv om det er det midterste element i mængden. Hvis ja er man færdig, og ellers går man videre med pkt. 2.

2. Hvis det element man søger er mindre end det midterste element, så brug binær søgning på forreste halvdel af datamængden, ellers brug binær søgning på bageste halvdel.

Processen "binær søgning" kan altså bekvemt udtrykkes ved *sig selv*, altså rekursivt. Det er klart at processen konvergerer, eftersom den mængde man søger videre i, hele tiden halveres. Man kan endda meget hurtigt se at det maksimale antal gange man skal prøve en sammenligning, må være totalslogaritmen til antallet af elementer i tabellen, da dette antal så vil være reduceret til 1.

Nu ville det være lykkeligt hvis det programmeringssprog man anvender, tillod én at nedskrive og udføre en rekursiv algoritme. Med så primitive sprog som Basic og Comal er dette imidlertid sjældent muligt, og man må derfor forsøge at omskrive den rekursive algoritme til en iterativ algoritme. For binær søgning er dette ikke svært, se fig. 1.

De to algoritmer på fig. 1 viser det væsentlige ved en omskrivning fra rekursiv til iterativ form, nemlig at en IF-konstruktion i den rekursive algoritme erstattes af en WHILE-konstruktion (eller evt. en REPEAT-konstruktion) i den iterative algoritme. Jeg vil i denne artikel vise et eksempel på hvordan man mere generelt kan omforme en rekursiv algoritme til iterativ form, samt hvordan man kan analysere sådanne algoritmer.

Stakken

Det værktøj man anvender ved realiseringen af rekursive algoritmer, kaldes en *stak*. Herved forstås en mængde af elementer hvorom Jesu ord: "De sidste skal blive de første", gælder helt bogstaveligt. Man kan nemlig putte elementer ind i en stak, og man kan tage elementer ud igen, men det element man får ud, er altid det der sidst blev puttet ind. Af denne grund kaldes en stak også tit en LIFO-kø (Last In - First Out).

I et programmeringssprog som Comal kan man tænke sig stakken realiseret som et array, DIM A (N), hvor N er stakkens maksimale *dybde*. Hvert element består i så fald af et tal. Der kan selvfølgelig også være tale om at man vil gemme tekster i stakken, hvis det passer bedre til problemstillingen, eller om et todimensionelt array, DIM A(N,M), hvor M så er det antal tal der skal til at beskrive et enkelt element i stakken. Det afhænger alt sammen helt af problemstillingen.

```
PROC BINSØG (A, FRA, TIL)
  IF FRA< TIL THEN
    LET K= INT((FRA+TIL)/2)
    IF X = A(K) THEN GOTO FUNDET
    IF X < A(K) THEN
      EXEC BINSØG(A,FRA,K-1)
    ELSE
      EXEC BINSØG(A,K+1,TIL)
    ENDIF
  ENDIF
ENDPROC
EXEC BINSØG(A,1,N)
```

```
LET FRA= 1; TIL= N
  WHILE FRA< TIL DO
    LET K= INT((FRA+TIL)/2)
    IF X=A (K) THEN GOTO FUNDET
    IF X<A (K) THEN
      LET TIL= K-1
    ELSE
      LET FRA= K+1
    ENDIF
  ENDWHILE
```

Fig. 1. Binær søgning opskrevet i "menneskecomal", dels som rekursiv algoritme (til venstre), og dels som iterativ algoritme med samme virkning (til højre). Der søges efter X *i talsættet* A *med* N *elementer. Man ser hvordan de to algoritmer svarer nøje til hinanden. Hvis sætningen* GOTO FUNDET *bliver udført er* X *fundet i* A*; Hvis* X *ikke findes i* A *fortsættes efter sidste sætning i de to algoritmer.*

```
PROC STAK
   LET S= S+1; A(S)= X
ENDPROC

PROC AFSTAK
   LET X= A(S); S= S-1
ENDPROG
```

Fig. 2. Comalprocedurer til stakning og afstakning af elementet X. Den variable S kaldes stakpegepinden. Den peger hele tiden på det øverste element i stakken A.

De to fundamentale operationer, at putte et element ind i stakken, og at tage et element ud, kan nu programmeres som to procedurer, se fig. 2. Disse to procedurer viser det helt væsentlige i stakmekanismen, nemlig at "de sidste skal blive de første", men som man ser er der overhovedet ingen kontrol på om man stakker et element i en fuld stak, eller om man afstakker et element fra en tom stak. Grunden er at det i de fleste tilfælde må betragtes som en fejl hvis dette sker, en mangelfuld analyse af algoritmen.

Tårnet i Hanoi

Fig. 3 viser det enmandsspil der går under navnet "Tårnet i Hanoi", vist nok fordi det fortælles at nogle buddhistiske munke i et tempel i Hanoi

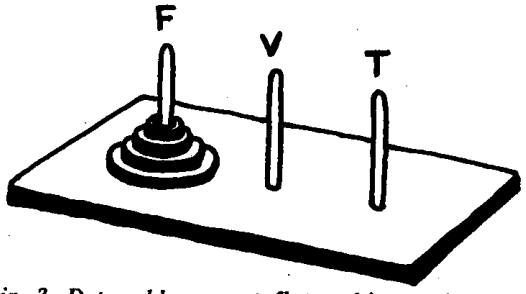


Fig. 3. Det gælder om at flytte skiverne fra pinden F *(for* FRA*) til pinden* T *(for* TIL*). Man må kun flytte 1 skive ad gangen, og man må aldrig placere en større skive oven på en mindre. Hjælpestangen* V *(for* VIA*) må gerne bruges undervejs. Opgaven lyder altså: flyt tårnet fra* F *til* T *via* V *med kun 1 skive ad gangen, og aldrig en større over en mindre skive.*

går og flytter rundt på sådanne skiver - en skive pr. dag - og at sagnet siger at verden vil gå under når spillet ender.

Hvis der kun er 1 skive på F er løsningen indlysende:

F = > T,

Men også for 2 skiver lader opgaven sig løse let:

F = > V
F = > T
V = > T.

Allerede med 3 skiver kommer mange i vanskeligheder, og for større antal, f. eks. 100, er man ret fortabt - ja, mange vil måske tvivle på at det overhovedet kan lade sig gøre når der kun er tre pinde.

Et eksistensbevis

Jeg vil nu vise at opgaven lader sig løse for N skiver, ved simpelthen at nedskrive en algoritme for fremgangsmåden. Dette er en ofte overset form for eksistensbevis, som imidlertid atter er kommet til ære og værdighed indenfor Datalæren.

Lad mig antage at jeg allerede har løst problemet, at jeg altså råder over en procedure:

PROC FLYT(N,F,V,T)

som flytter et tårn med højden N skiver fra stang F via V til T. Jeg ved godt at man ikke kan have parametre til procedurer i Comal, men lad mig foreløbig se stort på det. I mit Comal- med-parametre-sprog kan løsningsproceduren f. eks. se ud som vist på fig. 4.Først flyttes N-1 skiver fra F via T til V; derefter flyttes 1 skive fra F via V til T (det er jo blot operationen F = > T for den nederste skive, eftersom de N-1 overliggende lige er flyttet til V, og man får derfor ikke brug for at lægge den nederste store skive ovenpå de N-1 skiver der befinder sig på V). Nu ligger den største skive på T, og man kan derfor flytte de N-1 skiver på V via F til T - og så er opgaven løst!

```
PROC FLYT(N,F,V,T)
   EXEC FLYT(N-1,F,T,V)
   EXEC FLYT(1,F,V,T)
   EXEC FLYT(N-1,V,F,T)
ENDPROC
```

Fig. 4. En rekursiv løsningsalgoritme.

Vi ser altså at problemet kan beskrives rekursivt. Men betyder det nu at der altid er en løsning? Nej, kun hvis det rekursive program er en ægte algoritme, dvs. hvis det altid før eller siden stopper. Men det må det jo gøre, for vi ser at løsningen består af delløsninger med *færre* skiver, og disse kan igen nedbrydes til delløsninger med endnu færre skiver, indtil man til sidst står med en flok delløsninger, hvor der kun skal flyttes 1 skive i hver - og det er jo let. Der eksisterer derfor en løsning for ethvert N.

Et Comalprogram for tårnet i Hanoi

Men hvordan skriver vi nu PROC FLYT i virkeligt Comal? Heldigvis er der jo ikke så mange dikkedarer i det rekursive program; det består udelukkende af kald af "sig selv" med ændrede parametre. En beskrivelse af et sådant kald kan ske blot ved at angive parametrene, dvs. ialt 4 størrelser pr. kald.

Jeg vil bruge en stak til at gemme beskrivelser

Fortsættes side 13

Fortsat fra side 8

af de kald der endnu ikke er udført. Hvert element i stakken består derfor af 4 tal, og det vil være naturligt at erklære stakken som f. eks. DIM A(M,4), hvor M er den maksimale stakdybde.

Det første der skal stå i stakken er følgende:

N, F, V, T.

Tallenes rækkefølge angiver deres betydning: første tal er antallet af skiver der skal flyttes; andet tal angiver den pind de er på; tredie tal angiver hvilken pind der må bruges som hjælpepind; og sidste tal angiver den pind de skal flyttes til. Man kan nu benytte følgende fremgangsmåde:

1. Udfør skridt 2 til 4 sålænge stakken ikke er tom.

2. Afstak det øverste element i stakken; navngiv de fire tal således:

   N = første tal
   F = andet tal
   V = tredie tal
   T = fjerde tal

3. Hvis N = 1 så udfør flytningen F => T, ellers ellers gå videre til skridt 4.

4. Stak følgende tre elementer:

   N-1, V, F, T (kommer nederst i stakken)
   1, F, V, T (kommer næstnederst i stakken)
   N-1, F, T, V (kommer øverst i stakken).

Stakningen af de tre elementer i skridt 4 afspejler

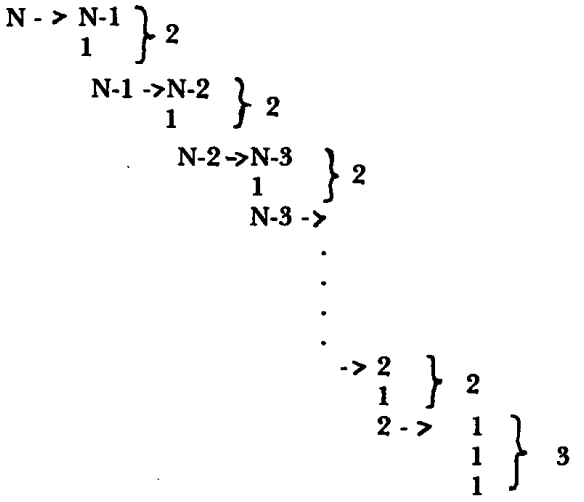


Fig. 5. Stakkens udvikling. Kun første parameter (antal skiver der skal flyttes) er vist. Tegnet -> betyder "afstakkes og erstattes af". Man ser altså f. eks. at det første N afstakkes og erstattes af de tre tal N-1, 1 *og* N-1, *hvoraf det sidste igen afstakkes, osv. indtil alle tal er reduceret til* 1. *Tallene efter klammerne er forøgelsen i stakdybden.*

nøje funktionen af den rekursive algoritme FLYT, idet de elementer der stakkes først i skridt 4 vil være dem der afstakkes sidst i skridt 2.

Tilbage er kun at beregne hvor dyb stakken kan blive. Hvis vi nøjes med at skrive det første af de fire tal i hvert stakelement, kan vi af ovenstående algoritme se at stakken vil udvikle sig som vist på fig. 5. Heraf ses at den maksimale længde bliver:

$$2 + 2 + 2 + 2 + \ldots + 2 + 2 + 3$$

Antallet af 2-taller i denne sum må være N-2, thi man kan trække 1 fra N ialt N-2 gange før resultatet bliver 2, som sluttelig forvandles til de tre 1-taller til sidst. Staklængden bliver altså:

$$2*(N-2) + 3 = 2*N - 1$$

```
0010 DIM T$(21)
0020 INPUT "ANTAL SKIVER= ",N0
0030 DIM A(2*N0-1,4)
0040 PRINT "TÅRNET I HANOI MED";N0;"SKIVER"
0050 PRINT
0060 LET S=0; F0=1; V0=2; T0=3
0070 LET T$="VENSTREMIDTEN HØJRE  "
0080 EXEC STAK
0090 REPEAT
0100   EXEC AFSTAK
0110   IF N1>1 THEN DO
0120     LET N0=N1-1; F0=V1; V0=F1; T0=T1
0130     EXEC STAK
0140     LET N0=1; F0=F1; V0=V1; T0=T1
0150     EXEC STAK
0160     LET N0=N1-1; F0=F1; V0=T1; T0=V1
0170     EXEC STAK
0180   ELSE
0190     PRINT T$(7*F1-6,7*F1);" => ";T$(7*T1-6,7*T1)
0200   ENDIF
0210 UNTIL S=0
0220 STOP
0230 PROC STAK
0240 LET S=S+1; A(S,1)=N0; A(S,2)=F0
0250 LET A(S,3)=V0; A(S,4)=T0
0260 ENDPROC
0270 PROC AFSTAK
0280 LET N1=A(S,1); F1=A(S,2); V1=A(S,3)
0290 LET T1=A(S,4); S=S-1
0300 ENDPROC
0310 END
```

Fig. 6. Comalprogram for Tårnet i Hanoi. T$ er en tekst der bruges til udskrift af resultatet, idet pind nr. 1 kaldes VENSTRE, pind nr. 2 MIDTEN og pind nr. 3 HØJRE. Tårnet flyttes fra VENSTRE til HØJRE. Størrelsen S er stakpegepinden i stakken A.

På fig. 6 ses et Comalprogram der løser problemet. Det er lavet sådan at de størrelser der stakkes (linie 230 - 250) kaldes N0, F0, V0 og T0, mens de størrelser der afstakkes (linie 260 - 280) hedder N1, F1, V1 og T1. Selve programløkken findes som linie 90 - 210. Starten af programmet sørger for initialisering af stakken, og sætter en tekst op der gør udskriften fra en kørsel nogenlunde forståelig. Programmets løsning af problemet med 4 skiver ses på fig. 7.

Analyse af algoritmen

Bestemmelsen af den maksimale stakdybde var en del af analysen af algoritmen på fig. 6. Et andet og måske nok så interessant spørgsmål er

```
TÅRNET I HANOI MED 4 SKIVER

VENSTRE => MIDTEN
VENSTRE => HØJRE
MIDTEN  => HØJRE
VENSTRE => MIDTEN
HØJRE   => VENSTRE
HØJRE   => MIDTEN
VENSTRE => MIDTEN
VENSTRE => HØJRE
MIDTEN  => HØJRE
MIDTEN  => VENSTRE
HØJRE   => VENSTRE
MIDTEN  => HØJRE
VENSTRE => MIDTEN
VENSTRE => HØJRE
MIDTEN  => HØJRE

STOP I LINIE 0220
◆
```

Fig. 7. Resultatet af en kørsel med programmet fra fig. 6, idet antal skiver blev indlæst som 4 i linie 20.

hvor længe programmet egentlig kører, når man starter med N skiver. Et naturligt mål for køretiden er antallet af kald af proceduren STAK. Dette må være lig med antallet af kald af proceduren AFSTAK, eftersom man slutter når stakken er tom. Ved at måle udførelsestiden for disse to procedurer, og beregne hvor mange gange de kaldes, kan man altså få en fornemmelse af programmets køretid.

Hver værdi af N giver anledning til tre nye stakninger, nemlig N-1, 1 og N-1. Man kan derfor tegne et træ som vist på fig. 8. Antallet af stakninger er det samme som antallet af knudepunkter i dette træ.

Hvert knudepunkt på fig. 8, som ikke er en kvist, deler sig i tre grene, hvoraf den ene altid er en kvist. Når roden er delt N gange er alle knudepunkter kviste. Derfor må man have:

antal knudepunkter

$$=1+1*(2+1)+2*(2+1)+2^2*(2+1)+...+2^{N-2}*(2+1)$$

$$=(1+2+2^2+2^3+...+2^{N-1})+(1+2+2^2+...+2^{N-2})$$

$$=2^N - 1+2^{N-1} - 1 = (3*2^N - 4)/2$$

Beregningen bygger på at summen kan spaltes i to kvotientrækker, som vist. Man ser at algoritmens køretid vokser eksponentielt med N. Beregninger med meget store skiveantal er derfor ikke praktisk gennemførlige.

Ud fra fig. 8 kan man også finde antallet af skiver der skal flyttes; det er nemlig lig med antallet af kviste i træet, da hver kvist repræsenterer en flytning. Man ser let at antallet af kviste (1-taller) fordobles for hvert niveau, og da den første kvist optræder på niveau 2 må dette antal være:

$$1+2+2^2+2^3+...+2^{N-1} = 2^N - 1$$

Ovenstående ræsonnement gælder ikke for N=1, men vi ser at formlen også kan anvendes i dette tilfælde. Som en ekstra kontrol kan vi se at antallet af flytninger på fig. 7 er 15, hvilket netop er $2^4 - 1$.

Afslutning

Hvis man ønsker at undervise i emner der lettest forklares ved hjælp af rekursive algoritmer, så kan der let brede sig en vis mystifikation blandt

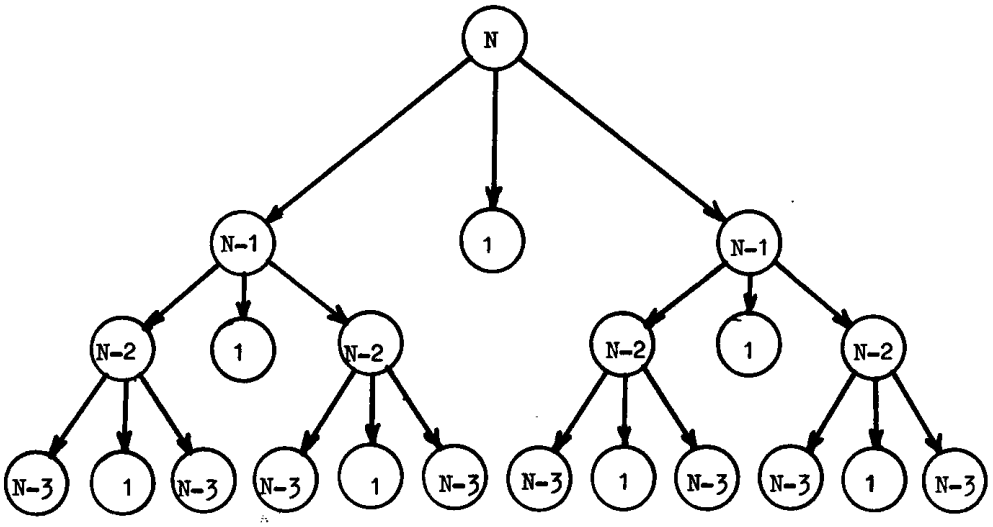


Fig. 8. Hvert knudepunkt (cirkel) symboliseret en værdi der stakkes. Når den afstakkes igen giver den anledning til de nye stakninger, som knudepunktet peger på. Kun kvistene i det fremkomne træ (cirkler med 1-taller) giver ikke anledning til nye stakninger. På figuren er kun vist fire niveauer af ialt N niveauer.

eleverne, udsprunget af en skepsis med hensyn til om sådanne algoritmer overhovedet kan laves i virkeligheden. Her kan et eksempel som Tårnet i Hanoi, der er tilstrækkeligt kompliceret til at man ikke umiddelbart kan se løsningen, men samtidig ikke mere kompliceret end at løsningen virker indlysende når man får den forklaret, være af stor pædagogisk værdi.

Endvidere viser eksemplet at Datalæren også kan bruges som illustration i brugen af matematiske beregninger. I dette eksempel er der tale om kombinatoriske overvejelser af ret avanceret natur, som gennem eksemplet får mening. Jeg tror at meget datalogisk stof vil kunne udnyttes på denne måde.

PROGRAMMERING OG PROBLEM-LØSNING – GRUNDBOG I DATALÆRE

H. B. Hansen, Ole Caprani, Frank Jensen
Gyldendal 1978.

Med udgivelsen af denne bog i grundlæggende datalære er der kommet en virkelig god lærebog til faget datalære i gymnasiet og hf. Bogen opfylder de krav, der stilles i Johnsen-udvalgets betænkning, hvad angår den grundlæggende del af stoffet.

Forfatterne har på en fin måde forstået at vise samspillet mellem problemløsning og algoritmeopbygning. Rutediagrammer anvendes i stor udstrækning til beskrivelse af algoritmer, uden at det virker hæmmende og uoverskueligt på indlæringsprocessen.

Bogen betjener sig af programmeringssproget BASIC, som gennemarbejdes grundigt med eksempler og opgaver. Ved denne gennemgang er det en absolut forudsætning, at eleverne har adgang til en datamat.

Som noget meget vellykket har forfatterne i kapitlerne 5 og 6 søgt at skabe en oversigt over de mest almindelige opgaveløsningsteknikker ved at anvende disse teknikker på en række eksempler. Jeg er overbevist om, at mange elever ved at gennemarbejde disse afsnit vil få mange ideer og impulser til opgaveløsning, som ikke blot vil være af værdi for faget datalære.

Endelig skal det bemærkes, at bogen ikke gør meget ud af systembeskrivelse (databeskrivelse). Dette skal ikke beklages, idet en dybere forståelse for denne disciplin kræver et kendskab til de systemer, der skal beskrives, et kendskab, der ligger langt udover, hvad der kan forventes af elever i gymnasiet og hf.
Jeg kan meget anbefale denne bog.

N. Stubbe Solgaard

●●●

SIKKE DOG ET NAVN!

I programkomiteen for IFIP's 8. verdenskongres sidder en række notabiliteter fra mange forskellige lande.
Men ved valget af én af personerne viser det sig tydeligt, at nok er Tønder og specielt Børge verdensberømt i Danmark, men dette strækker sig altså ikke til IFIP.
De har nemlig placeret en person ved navnet GOTO i programkommiteen (!)

●●●

DATALÆRE I AALBORG

På forsiden af Aalborg Stiftstidende d. 13. 1. 79 kunne man læse om en ny sygdom blandt skoleelever. EDB-feber.
I forbindelse med skolekommisionens anbefaling af fortsættelse af forsøget med datalære på fire Aalborgskoler, havde en journalist været på besøg på Sofiendalskolen samtidig med at denne skole havde det rejsende anlæg på besøg.
Aktiviteternes omfang og elevernes store interesse fik så avisen til at finde på betegnelsen EDB-feber.
Interessen har også været overvældende; eleverne møder ofte kl. 7 om morgenen for at få en time ved terminalerne i fred og ro, og de sidste bliver smidt ud af pedellen kl. 22.30 om aftenen.
I et enkelt tilfælde har en elev iøvrigt sneget sig ind på skolen, mens den havde lukket, og så bag nedrullede gardiner sat sig til at arbejde ved terminalen.
Så meget trækker datalære og terminalerne altså i eleverne.

Alle programmer, som er skrevet i version 1.05, vil umiddelbart kunne anvendes i den nye version 1.06. Nuværende COMAL 80 brugere vil for en beskeden pris kunne få opdateret de "gamle"systemer, og samtidig få et eksemplar af den reviderede manual og referencekort.

RC 700 PICCOLO som generel STYRE- og MÅLEDATAMAT.

Undervisningsministeriet har fornylig offentliggjort en rapport over typer og antal af datamaskiner, som finder anvendelse i folkeskolen, gymnasier og seminarier.
Det fremgår heraf, at et stort flertal af disse undervisningsinstitutioner anvender microdatamatsystemet RC 700 PICCOLO fra A/S Regnecentralen af 1979. Systemerne anvendes i mange sammenhænge, og antallet af anvendelsesområder udvides stadig.
Den nyeste anvendelse er blevet til i et samarbejde mellem adjunkt Martin Lund fra Sct. Knuds Gymnasium og elektronikfirmaet Micro Technic i Odense: et interface til RC 700, således at denne kan anvendes i undervisningen til dataopsamling, processtyring m.v.
Interface-udstyret produceres og markedsføres af Micro Technic under navnet Micronixsystemet. Det vil uden problemer kunne tilkobles ethvert RC 702 microdatamatsystem, altså også hidtil leverede anlæg, og naturligvis uden indskrænkninger i eksisterende garanti- og/eller serviceforhold.
Micronix måle-/ styreinterface for RC 702 PICCOLO består i grundversionen af følgende enheder:

1 stk. Expansionskabinet Z 80 - 607
1 stk. PICCOLO interface Z 80 - 605
1 stk. Kabel Z 80 - 606

Grundsystemet kan bestykkes med op til 4 stk. I/O-moduler, hvert med et antal digitale eller analoge ind- og/eller udgange. Micronix tilbyder et stort antal forskellige I/O- kort, med hvilke brugeren kan løse størstedelen af sit signalbehov til RC 702 PICCOLO.
Endvidere har Micro Tecnic med adjunkt Martin Lund som pædagogisk konsulent påbegyndt udvikling og markedsføring af "pakkeløsninger" af små projekter med henblik på undervisning i automatiseret måling og proccesstyring. Hver "pakke" omfatter de til opgaven fornødne komponenter og øvrige løsdele, monteringsvejledning, funktionsbeskrivelse samt et dokumenteret program med kildeteksten på minidiskette. Dokumentationen vil endvidere omfatte idéer og konkrete forslag til ændringer og/eller videreudvikling af projektet. Brugeren får således rådighed over et solidt basismateriale, som kan varieres og udbygges afhængigt af individuelle evner og behov.
Nærmere oplysninger om Micronixsystemet og "pakkeløsningerne" fås ved henvendelse til:

Ingeniørfirmaet Micro Technic ApS.
Sdr. Boulevard 58 D
5000 Odense C
Telefon (09) 13 74 13

REKURSIVE PROCEDUREKALD

Anvendelsen af rekursive procedurekald, d.v.s. procedurer som kalder sig selv et antal gange, kan ofte resultere i at programmerne bliver væsentligt kortere. For dygtige programmører er der ofte sport i at gøre programmerne så korte som muligt. Dette bevirker dog også at de bliver svære at forstå for andre end den, som har udarbejdet programmet.
Det klassiske programmeringsproblem "Tårnet i Hanoi", har ofte været genstand for ovennævnte dispositioner. Tidligere har disse korte versioner været forbeholdt PASCAL programmører, men ved fremkomsten af rekursive procedurer i COMAL 80 udfoldes der også en stor fantasi her. Som eksempel på hvor elegant det kan gøres, bringes her listningen af programmet. Programmet bringes ukommenteret, idet det jo er klart for enhver, hvad der sker i programmet! ! !.

```
0010 PROC flyt(n,tårn1,tårn2,tårn3)
0020   IF n>1 THEN EXEC flyt(n-1,tårn1,tårn3,tårn2)
0030   PRINT "FLYT SKIVE ",n," FRA TARN ",tårn1,
0040   PRINT " TIL TARN ",tårn2
0050   IF n>1 THEN EXEC flyt(n-1,tårn3,tårn2,tårn1)
0060 ENDPROC flyt
0070 INPUT "HVOR MANGE SKIVER: ": i
0080 EXEC flyt(i,1,2,3)
```

Referat af:

Seminaret vedrørende brug af Mikroprocessorbaserede smådatamater i undervisnings-sektoren

✻ Esbjerg 28.-29. sept. 1979.

Det følgende et er stærkt koncentreret referat af det meget fyldige program.

I et par timer før den officielle åbning og under hele seminaret var det muligt at arbejde med 8 firmaers materialer. De deltagende firmaers navne vil fremgå af det følgende.
Selve seminaret, der afholdtes i Esbjerg Seminariums yderst velegnede lokaler, blev åbnet fredag eftermiddag med velkomst af seminariets rektor, Palle Schmidt. Rektor Schmidt glædede sig over de op mod to hundrede deltagere, og hans indledning blev fulgt af J. Ditlev Nielsen fra SUC, der i sin indledning fortalte, at Ribe Amtsprojektet var udsat et år, men at der alligevel var mange ting på vej.
Endelig bød Torsten Alf Jensen som repræsentant for de øvrige arrangører, Statens erhvervspædagogiske Læreruddannelse og Datalærerforeningen, velkommen til seminaret, og han introducerede derefter den første taler:

Roy Atherthon, Bulmershe College of Higher Education, Reading, UK

R. A., der har forbindelse med uddannelse af lærere til Secondary Schools, beskrev forholdene for datalære i England, og han kunne oplyse, at der i 1978 var 26.419 elever, der aflagde prøve i datalære på forskellige niveauer.
Af foredraget, der iøvrigt blev fremlagt i dansk oversættelse, kan fremdrages nogle enkeltheder:
Når lærerne ikke engang kan bruge OHP, lysbilledapparat eller filmapparat, kan der blive problemer med at få dem til at anvende datamater som hjælpemiddel.
Det skal være simpelt at starte et program.
Strukturerede sprog som f. eks. COMAL er afgørende vigtige.
Arbejde med datafiler er vigtigere og vigtigere, mens behovet for at kunne løse matematiske opgaver er mindre.
Mikrodatamaterne vil snart kunne tælles i tusindvis i skolerne.
Det er vigtigt med en god og omfattende læreruddannelse, f. eks. 480 timer som deltidsstudium for en lærer til datalære.
Datamaskinen anvendes både i England og i Scotland som hjælpemiddel i mange fag. Denne brug vil være stigende.
R. A. opstillede "de 7 dødssynder":

1. Ikke-standard software
2. Ikke-standard hardware
3. Mangel på standard for programskrivning
4. Dårlig afprøvning af programmer
5. For optimistiske tidsplaner
6. Ukontrollerede programmodifikationer
7. Dårlig dokumentation

Efter Roy Atherton fulgte næste foredragsholder:

Børge Christensen, Tønder Statsseminarium

BC, der har defineret COMAL, gennemgik 70'ernes udvikling i Danmark m.h.t. maskinel og programmeringssprog. Han gjorde opmærksom på, at brugen af strukturerede programsprog havde medført brug af nye metoder for undervisning i datalære. Eksempelvis strukturdiagrammer i stedet for rutediagrammer.
BC fastslog det synspunkt, at det måtte være væsentligt at undervise i programmodifikation fremfor i programkonstruktion. Dette vil desuden give pædagogiske og metodiske fordele, lige som det ville give mindre skrivetid for den enkelte.
BC fremlagde eksempler på et undervisningsmateriale, der udnyttede disse synspunkter.
BC talte desuden om vanskelighederne for læreruddannelse, om den kommende udvikling i brug af smådatamater, og om nødvendigheden af at finde et beskrivelsesmiddel, der kunne lette udvekslingen af programmer fra et system til et andet.

Præsentation

Efter middagsspisningen i seminariets kantine fortsatte seminaret med en mundlig præsentation af de 8 firmaer, der deltog i udstillingen.
Der var fremlagt brochurematerialer for alle firmaer på udstillingen. I et meget kort sammendrag beskrives her de enkelte maskiner i grundversion:

APPLE II: Personlig datamat med ROM Basic-fortolker. Med 16kb RAM lager, men uden skærm og ydre lager ca. 11.500 kr. Gode muligheder for udbygning, men ingen planer for et udbygget regionalt system.

Akademisk Boghandel: Fremførte planer for den kommende Texas 99/4, som skulle have meget fine specifikationer. Men der blev ikke fortalt noget om et eventuelt struktureret sprog eller planer om udbygningsmuligheder. Prisen skulle blive 5-6000 kr.
Endvidere omtaltes PET med BASIC og 16 kb lager, kassettebåndstation men ingen udbygning. Pris 9800 kr.

RC 701 En ikke færdigudviklet smådatamat med COMAL-fortolker, der indlæses fra diskette. 16 kb lager og ydre lager på disketter med 180 kb. Pris 16.000 kr. De første 50 kan leveres inden jul, og der bliver mulighed for opkobling med de kendte RC datamater, der i forvejen findes. Firmaet lagde sig tæt op af Edb-kapacitets udvalgets modeller og indbød til lærersamarbejde.

ICL COMET En næsten færdig smådatamat med COMAL-fortolker, der indlæses fra hurtigt kassettebånd (6000 baud). 16 kb lager og ydre lager på kassettebånd. Pris 10.000 kr. Leveringstid 6 uger. Der søges opbygget et overordnet system, som det kendes fra Ribe Amts model.

DDE En fuldt færdig smådatamat med COMAL-fortolker, der indlæses fra minidiskette. 16 kb lager og ydre lager på 90 kb minidiskette. Hardware-matematik og meget stabil opbygning af kabinetter. Pris 19.500 kr. Ingen udbygningsplaner, men dette kan laves til andre systemer. Kan trække op til 8 terminaler.

Poly-data En personlig datamat med BASIC-fortolker, der indlæses fra kassettebånd. Pris 10.500 kr.
Kan udbygges, men ingen planer om et overordnet system.

ABC 80 En salgsklar personlig datamat med Basic-fortolker i ROM. 16 kb lager og kassettebånd (800 baud). Pris 8.900 kr. Kan udbygges med tilbehør, men foreløbig intet overordnet system.

TRS-80 En personlig datamat med Basicfortolker. Ingen nærmere oplysninger om udbygning lokalt eller i større system. Pris 8.000 kr.

Efter yderligere prøver på udstillingen sluttede det officielle arrangement fredag kl. 22.

Lørdagens program indledtes med et foredrag af

Max Bramer, Open University Computer-assisted Learning Research group

MB gik i sin indledning ind på eksplosionen i udvikling af digitale kredse, og anførte dette som en væsentlig grund for at undervise alle i datalære. Der omtaltes problemet vedrørende den manglende læreruddannelse, og MB understregede ligeledes det uheldige i, at kun matematikere beskæftigede sig med datamaskiner i undervisningssituationer. MB pointerede kraftigt, at undervisere skulle styre udviklingen af programmel. Vigtigheden af at undgå fejl, der kan tåles af en specialist, men som generer en lægmand, blev betonet.
MB mente, at Computer-assisted Learning ville kunne udføres på smådatamater i nær fremtid. Maskinomkostninger ville ikke længere være et problem, men der mangler stadig meget i programdokumentation.
Iøvrigt blev der omtalt forskellige materialer fra England, og der blev fremlagt en adresseliste.
Afslutningsvis blev der givet eksempler på kommercielle systemer, der var helt uden påvirkning fra lærerside.
Efter foredraget viste MB et par film om de omtalte emner.

Efter 2 timers arbejde med de udstillede materialer, og efter frokosten, fortsatte møderækken med

Tom Østerby, Danmarks Tekniske Højskole

TØ talte om de tekniske muligheder indtil 1985, og han delte emnet op i Teknologi, Microprocessorer, Hovedlagre og Baggrundslagre.

Der blev givet en populær fremstilling af de faktorer, der spillede en rolle for de enkelte emner, og derud fra drog TØ, hvad han selv mente, forsigtige konklusioner.
Microprocessoren ville få 20-30 gange så stor "styrke", arbejdslageret ville blive 4-16 gange så stort og også de ydre lagre ville blive mere tilgængelige for den almindelige bruger.
TØ kunne således konkludere, at smådatamaten ville få datamatisk styrke som en større datamat i dag.
Alle TØ's slutninger blev underbygget med talrige eksempler, og der blev således givet grundlag for at tro, at de tekniske muligheder ville give grundlag for selv det mest ambitiøse projekt.

Panel diskussion

Efter foredraget fortsatte seminaret med en paneldiskussion, hvori der deltog som panel:
Paul Bjørnum,
Direktoratet for erhvervsuddannelser (handel)
Asger Svane,
Direktoratet for Erhvervsuddannelser (teknisk)
Niels Bandholm,
Amtsgymnasiet Viby J
Børge Christensen,
Tønder Statsseminarium
Søren Ravn,
Danmarks Lærerhøjskole
Tom Østerby,
Danmarks Tekniske Højskole
Arne Jepsen,
Tandlægehøjskolen, Aarhus
Torsten Alf Jensen,
Statens Erhvervspædagogiske Læreruddannelse
Erling Schmidt,
Folkeskolen og Datalæreforeningen (ordstyrer)

Meget kort om de enkelte indlæg:
Bjørnum søgte en overordnet målsætning, mens Svane ønskede en belysning af de humane aspekter i forbindelse med f. eks. numeriske styringers indflydelse på arbejdspladsen.
Bandholm fortalte om en metode til at opnå indflydelse på (udvalg), og han efterlyste humanister som deltagere i udviklingen. Han efterlyste endvidere kurser og folk, der kunne skrive i medierne.
TAJ påpegede, at udviklingen skulle ske på skolerne, ikke centralt, og han advarede mod at bruge datamaterne på områder, hvor andre metoder var bedre.
BC omtalte en model for datamatforsyning af mellemniveauskolerne og ønskede et sprog og et udvekslingsmedie (ét og kun ét).
Ravn omtale de forskellige krav til de forskellige brug af datamaskinen, og han plæderede for en uddannelse af alle lærere for at skabe et milieu, der forstår begrundelsen for brug.
Østerby omtalte muligheden for, at smådatamater kunne blive så små, at de kunne medtages til f. eks. en prøve. Han mente, at fremtidige operativsystemer ville blive mindre og begyndende standardiseret.
Jepsen fortalte, at man prøver at gå videre med brugen af datamaten, således at den også kunne anvendes til problemløsning på højt plan. Man havde fået bevilling til et større system. Samarbejde med Jysk Telefon. Materiel: RC 8000 og RC 701.

Efter den indledende runde kom der kommentarer fra auditoriet: Vejle Gymnasium spurgte, om man ville anbefale at udsætte køb af smådatamater et år? TAJ betonede nødvendigheden af, at man kom i gang for at få erfaringer. I den følgende diskussion blev der spurgt fra salen, om man overhovedet kunne lægge pres på firmaerne? BC mente, at det var muligt.
Senere omtalte TAJ, at projekter ofte standses i "velvilje".
Søren Ravn blev spurgt om, hvor mange lærere der fik kursus, men han sagde, at hvis et kursus var bare få år gammelt, var det til ingen nytte.
Gerd Belhage kunne fra salen oplyse, at 400 elever i København modtog undervisning i datalære.
Herning Amtscentral spurgte, om der kunne ventes anbefalinger fra "oven", og om der skulle oprettes programbiblioteker på amtsplan eller lignende. Tom Østerby mente, at det var bedre at udveksle ideer end programmer. Arne Jepsen sagde, at alle systemer fra Tandlægehøjskolen var generelt anvendelige, og de kunne fås gratis. TAJ sagde, at man måtte undgå lukkede løsninger (program/maskinel).
Rud. Christensen, Edb-rådet påpegede lovgivningens manglende målsætning. Han omtalte desuden Edb-undervisningens vigtighed for minimalisering af vanskelighederne ved indførelse af Edb. Han mente iøvrigt, at man i skolekredse burde sige stop for videre arbejde, indtil ordentlige forhold forelå.
Det sidste punkt besvarede Børge Christensen med henvisning til, at han (og andre) beskæftigede sig med disse ting, fordi han ikke kunne lade være. Det var ikke et spørgsmål om at strejke, men om at udføre et stykke arbejde.
Videre efterlyste Arne Jepsen dokumentation af programmer, hvilket fik Grete Illum fra salen til at beskrive en arbejdssituation for en ansat i folkeskolen.
I samme forbindelse ønskede TAJ at undgå for præcise undersøgelser med specifikke resultater.
Til slut blev der fra salen ytret ønske om, at man fortsatte arbejdet med at undersøge undervisningssektorens fremtidige virke på dette felt.

Erling Schmidt modtog dette ønske på datalæreforeningens vegne, og efterlyste samtidig reaktioner fra foreningens medlemmer.

Torsten Alf Jensen afsluttede derefter møderækken med et løfte om fælles anskaffelse af engelske materialer, og omtalte en studietur til England i december 79.
Efter dette var der endnu en time til arbejde med udstillingen, hvorefter seminaret sluttede kl. 17.

FGK

Om datalæreundervisning

af Finn Haahr Kristiansen

DATALÆRE er nu udkommet i 3 år, et tidsrum, som vel er langt nok til, at man kan begynde at vurdere, hvad der så er kommet ud af det! I det følgende vil jeg tillade mig at betragte de behandlede emner som udtryk for, hvad der foregår i forbindelse med datalære hos vore medlemmer rundt om i landet.
Desværre må jeg sige, at bladet for mig stort set har været kedeligt og ikke alt for opløftende læsning. De emner, som er morsomme, taknemmelige og relativt nemme at gå til for en datalærer med matematisk baggrund (hvilket er langt de fleste af os), har været behandlet i rigt mål; mens de emner, som er mere problematiske som objekter for en forsvarlig undervisning, næsten ikke har været berørt.
De to altdominerende emner har været noget, der kunne kaldes "datamaskinen som værktøj" og "programmering". Ved det første forstår jeg emner, som f. eks. CAI, datamaskinens anvendelse i forskellige skolefag, administrative anvendelser m. m. Ved programmering forstår jeg programmering som disciplin indenfor faget datalære, og omfatter både algoritmisering og afvikling af kørsler. For at undgå misforståelser vil jeg pointere, at jeg naturligvis anser begge emner for vigtige, men at de altså er groft overrepræsenterede.
Et par ord skal de dog have med på vejen. Som underviser i datalære er det af disse to emner især programmering, som interesserer mig, og jeg må indrømme, at det er med bekymring, at jeg konstaterer, at programmeringssprog er lig med BASIC for langt de fleste af DATALÆRE's bidragydere. Det burde efterhånden være dokumenteret, at når BASIC er det første programmeringssprog, der stiftes bekendtskab med, og derfor er med til at danne de programmeringsvaner, man uvægerligt slæber med sig, er dét sprog intet mindre end en pædagogisk katastrofe. At det alligevel stadig har den store udbredelse skyldes jo nok, at det ofte er det eneste flerbrugersystem, og at inertiens lov jo desværre ikke kun gælder i den newtonske mekanik. Det skal såmænd nok vise sig, at den sidste grund bliver langt den vanskeligste at komme til livs!
På seminaret i Esbjerg for nylig om mikrodatamater i undervisningen var der dog en vis bevægelse at spore på det punkt. Men underligt nok ser det ud til, at BASIC's afskaffelse bliver et resultat af, at datamaskinfabrikanterne efterhånden ved, at nu er det COMAL, der er sagen, hvis man vil ind på markedet, og ikke et resultat af pression fra en bred front af undervisere i datalære. Det er selvfølgelig Børge Christensen, der skal have æren af at have overbevist disse fabrikanter. Desværre var der i Esbjerg al for lidt lejlighed for deltagerne til at komme til orde, så det er lidt vanskeligt at vurdere, om BASIC for alvor er ved at blive afvist af vore medlemmer; men jeg frygter, at der er et stort tavst flertal, som vil formå at holde liv i BASIC i mange år endnu. Et lille suk, når talen er om COMAL: nu har vi så fået de kontrolstrukturer, vi manglede (while, repeat, m. m.); men det at kunne udtrykke sig naturligt i et programmeringssprog er også et spørgsmål om at kunne organisere sine data naturligt - et forhold, som desværre ofte overses. Kraftige datastrukturer á la PASCAL's må altså blive det næste (så hvorfor egentlig ikke PASCAL?).

Så meget om programmering. Men datalære er jo meget mere end programmering, som jo kun er et af mange emner, som skal dækkes. Til trods herfor har disse emner næsten ikke været berørt i noget indlæg i DATALÆRE. Hvad er mon grunden? Jeg tror selv, at det er den enkle, at de fleste undervisere i datalære er folk med matematisk/fysisk baggrund, som er vant til at udtrykke sig præcist i et formaliseret sprog - og det er lige netop, hvad man kan i et programmeringssprog, som ydermere har den for matematikere fascinerende egenskab, at det lader sig bruge til at udtrykke dynamiske sammenhænge, i modsætning til matematik, der holder sig til de statiske.
Desuden er det taknemmeligt at undervise i pro-

Anvendelsen af EDB i undervisningen - herunder datalære - ved

Odense kommunale Skolevæsen

Gennem denne og følgende artikler - indtil nu er der planlagt yderligere 2 - vil jeg prøve at beskrive, hvad der sker nu, hvad der er sket i et "historisk" tilbageblik,og hvad der er planlagt i den nærmeste fremtid på datalære-fronten og i anvendelsen af datamaskinen i undervisningen ved skolevæsenet. Artiklerne vil delvis være summariske, idet pladsen i bladet ikke tillader for f. eks. datalæres vedkommende at gå ind i en metodisk beskrivelse i alle detaljer. For dette områdes vedkommende såvel som for anvendelsesområdet må jeg henvise til vore rapporter, der forefindes på Amtscentralerne for undervisningsmidler.
Denne artikel vil prøve at ridse nogle historiske linier op fra starten og indtil i dag, hvor der er ved at blive indledt en ny spændende epoke på hardwaresiden med fremkomsten af microdatamaterne.

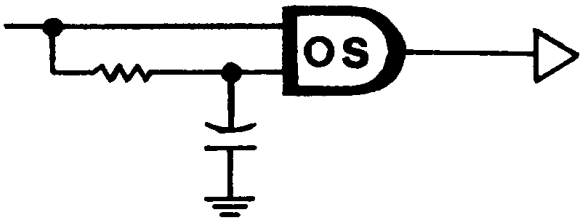
Starten

Odense Kommunale Skolevæsen er landets 3. største skolevæsen med 42 folkeskoler, altså et meget stort elevgrundlag. Da vi allerede i 1972 begyndte at tumle med planer om EDB i undervisningen, i første omgang som datalære, stod det os klart, at der måtte findes løsninger, der kunne tilgodese alle skoler og ikke blot en enkelt. Det var derfor væsentligt, at den lærergruppe, der havde nogle idéer herom, kontaktede skoleforvaltningen i allerførste fase. Dette skete i 1972, og viceskoledirektør Emil Pedersen fornemmede straks, at vi her stod overfor et medie med en kolossal betydning for den enkelte og for samfundet i alle dets aspekter i den nærmeste fremtid og derfor måtte tages op som emne i folkeskolen. Dette underbyggedes yderligere ved fremkomsten af betænkning nr. 666 (Johnsen rapporten) i 1973, som gjorde meget ud af folkeskolens rolle i denne sag, og som derfor indgik i beslutningsgrundlaget. Dertil kom, at folkeskolens læseplansudvalg i 1974 udgav vejledende læseplaner for valgfaget datalære (hæfte nr. 24). Det rent faglige fundament var nu til stede, der kunne skrides til handling!
Skolevæsenets meget store velvilje resulterede i, at vi i april 1974 kunne indlede et samarbejde med Odense Tekniske Skole, der rådede over et dengang meget avanceret dataanlæg bestående af en RC7000, 16 kw time-sharing maskine uden disk. Odense skolevæsen etablerede yderligere 16 kw lager, multiplexer for ialt 8 linier og en 512 kw fasthoveddisk på Teknisk Skoles anlæg, så det nu var rimeligt slagkraftigt. Alt dette udstyr blev leaset ud fra den beslutning, at vi ville starte på forsøgsbasis med de 4 skoler, der hver blev forsynet med en indkøbt TTY.
Vi var i gang - og det var mageløst!
Det var en oplevelse i 1974 at se sine kolleger på skolen stå i fortabelse (beundring eller afmagt?) over dette af-sig-selv-skrivende-og-larmende uhyre af en Teletype. Jeg inviterede dengang en ældre, meget fynsk kollega til at sætte sig ved terminalen. Men med udslåede arme kom det på syngende fynsk: "Næj, de' dævelskav ve' j'ette røre ve'!". Det var Mads Jørgensen dengang. Han ændrede senere sit synspunkt og er nu en central person i vore aktiviteter, men det er en anden historie, som jeg senere vil komme ind på.

Fortsættelsen

I løbet af 1974 og til begyndelsen af 1975 skete der en eksplosiv udvikling, da vi fik mulighed for at indkøbe 6 brugte TTY'er fra Kommunedata/ØK-data for en rimelig pris. Parallelt hermed havde Danmarks Lærerhøjskole dengang en meget høj aktivitet omkring kurser i datalære. Dette medførte, at mange skoler i begyndelsen af 1975 ønskede at få en terminal til datalære. Fremgangsmåden for en skole for at få en terminal var dengang som nu, at skolen fremsendte en ansøgning til skolevæsenet om at måtte få en terminal. I det omfang, der var terminaler til rådighed, fik skolen da leveret én med tilhørende modem og telefon, kvit og frit. Skolens fremtidige udgift ville kun blive driftsudgiften til telefon (abonnement + samtaler) og lejeafgiften til P&T for modem. Skoler, der eventuelt lå udenfor laveste telefontakst eller havde et meget højt timeforbrug, kunne dog få en fast linie. I foråret 1975 stod vi derfor pludselig med 7 ansøgninger om terminal, Teknisk Skole forudså også terminaludvidelser, hvilket alt i alt indebar, at den fælles RC7000 på Teknisk Skole ville være for lille ved starten på det nye skoleår pr. august 1975. Odense skolevæsen valgte derfor at opbygge en RC7000 datamat med placering på Højstrupskolen.
I maj 1975 afholdtes på Højstrupskolen et større arrangement med et meget stort antal personer med interesse for datamaskinen i undervisningen. Det var ved dette arrangement, vi så Cauchis fantastiske hjemmesløjd udi logiske modeller og half-

addere, og det var vel også ved dette arrangement startsignalet til Datalæreforeningen blev givet. Ved denne lejlighed fik "stedet" på Højstrupskolen også sit navn DOS. På tegningen herunder ses vort symbol, bag dette og navnet DOS ligger der en lille historie, der senere er kommet tilbage til os fra det tyske i en noget forstørret form - men det er vel heller ikke for ingenting, at vi bor i H. C. Andersens fødeby, hvor én fjer bliver til 5 høns.



Dette er historien: RC7000 med tilbehør kom i spændende papkasser, der behørigt og andægtigt blev pakket ud. Den 21. maj begyndte RC på samlingen af stumperne og den 22. kl. 11.55 var man nået så vidt, at en TTY var forbundet til I/O-kortet på RC7000 på korrekt vis, som jeg selv senere kunne konstatere på diagrammet. Fra maskinens bagside blev jeg af RC's tekniker bedt om at dreje maskinens nøgle på ON, der skulle spænding på - OG DET KOM DER! Med brag og røg stoppede RC7000 enhver form for aktivitet og en dybt chokeret tekniker dukkede frem af mørket. Hvad var der sket? Diagrammet var forkert. I stedet for 6 V DC fik I/O-kortet 115 V AC. Enhver med lidt kendskab til kondensatorers opførsel overfor store spændinger kender resultatet, det kan næsten bruges nytårsaften. Der blev smadret virkelig meget, idet de 115 V var kommet langt omkring: til lager, til diskkontroller, til CPU etc.

Men barnet var født: en gate med OS i - DOS blev til Datacentret ved Odense Kommunale Skolevæsen, hvortil vor vakse sekretær replicerede: "Det er da godt,det ikke ligger i Assens!"

En forklaring på hvad der var sket, fandt vi også senere. Diagrammet var et resultat af lemfældig omgang med en kopimaskine. Der var intet at laste den pågældende RC-tekniker for.

Den opstillede RC7000 var dengang forsynet med 512 kw fasthoveddisk, 2 stk. Diablo movingheaddiske á 2.4 MB, printer, kortlæser, strimmellæser og -huller, og vi så fremtiden fortrøstningsfuldt i møde, indtil der skete 3 ting, der fik afgørende betydning:

- terminalantallet så ud til at ville øges kraftigt.
- Børge Christensen ringede en højhellig 1.juledag
- Mads Jørgensen blev "omvendt"

Det første taler for sig selv, det gør det andet for så vidt også for dem der kender omtalte Børge Christensen.

COMAL på DOS

Men han ringede alligevel denne dag i 1975, hvor man nød en afslappet juleferie uden ret megen tanke for edb, og spurgte om han måtte komme over 2. juledag, mens han stadig mumlede om noget nyt og mærkværdigt: COMAL. Jo, det var noget, man gik og rodede med nede i Tønder på en single-user maskine, og det var vist da også meget godt, men man ville nu gerne have en diskmaskine at rode på, for at blive fri for strimlerne. Måske kunne man (Per Christiansen, en meget kvik gymnasieelev og Knud Christensen, en meget lidt talende, men dygtig lærerstuderende) gøre "noget" ved et disksystem. Grundet fødselsdag i familien måtte jeg bede BC udsætte besøget til 3. juledag, hvor jeg så blev vidne til en fantastisk forestilling med 2 personer, der talte "sort" og engang imellem i mere klart sprog bad BC holde mund. (Kan man forstå det skulle være nødvendigt!), alt imens denne påstod, at han ikke forstod en pind af, hvad de lavede, men på den anden side kunne løfte lidt af sløret for, hvad COMAL var.

Ved hjælp af telefonsamtaler med Tønder og mange aftener og week-ends på DOS blev den berømte version til RDOS systemet færdiglavet på DOS i løbet af perioden januar-april 1976. Det er vel ikke at tage munden for fuld, når jeg vil hævde, at dette stykke software - COMAL75 - er noget af det mest stabile, der har kørt på en RC7000. Det var også med blødende hjerte, at vi i december 1978 sagde farvel hertil, men det var bl. a. Mads Jørgensens skyld.

På trods af påbud fra sin læge om at skippe alt ekstraarbejde grundet for højt blodtryk, lykkedes det Emil Pedersen og undertegnede at overtale Mads Jørgensen i dennes 62. år til at gå ind i et større projekt: anvendelse af datamaskinen i dansk- og regneundervisningen i hjælpeskolen. Mads er skolekonsulent for en del af skolevæsenets specialklasser (elever med generelle indlæringsvanskeligheder) som vi på Højstrupskolen pr. gammel tradition kalder hjælpeklasser, hvor der stilles meget store krav til individualiseret undervisning.

Forsøgsundervisning

Mads havde i foråret 1976, hvor COMAL75 var færdiglavet, en 2. specialklasse, og vi havde med COMAL et godt værktøj til at lave rimeligt godt programmel. Vi var så heldige, at Mads lidt tøvende sagde: "Jah, lad mig prøve". På mindre end en uge var han overbevist, det lidt betingede ja, blev nu et ubetinget JA. Ovenpå hvirvelstormen fra Tønder, kom vi nu ind i en ny serie hvirvelstorme, Folkeskolens Forsøgsråd sagde ja til at gå ind i et forsøgsarbejde på to fronter: anvendelse af datamaskinen som hjælpemiddel i to hjælpeklasser samt forsøg med datalære i 8. og 9. klasse. Gennem datalæreforsøget håbede vi på, at data-

lære kunne komme ind som valgfag i folkeskolen. Det må her erindres, at på trods af betænkning 666, hæfte 24 fra læseplansudvalget og hele diskussionsoplægget til den nye skolelov, der blev vedtaget i 1975, var datalære ikke med som valgfag i den nye lov. Men datalære var for forsøgsrækken allerede indplaceret på de lokale undervisningsplaner som en deldisciplin af regning/matematik eller samtidsorientering eller fysik over et tillæg af 2 timer pr. uge til det pågældende fag. Det vil sige, at datalære i praksis fungerede og fungerer som et valgfag for elever på 8.-10. kl. Ved skolevæsenet er der i indeværende skoleår ca. 40 datalærehold på de 18 skoler, der har terminal. Elevtallet på datalæreholdene varierer fra 7-8 elever til 22 (!). Om holdene med det meget store antal elever, kan jeg sige, at de ikke fra starten er tilstræbt så store, men at specielle forhold har spillet ind, f. eks. kan en lærers afgang til anden stilling have medført, at to hold er blevet slået sammen til ét. På Højstrupskolen er der 4 hold med ialt 46 elever. Forsøget med datamaskinens anvendelse i hjælpeklasserne faldt således ud, at vi måtte påregne en yderligere terminalvækst for at klare efterspørgslen. RC7000 ville snart være for lille på multiplexersiden (16 indgange), på disksiden og på CPU-siden (svartiderne). I det sidste år med RC7000 (1978) afvikledes vore programmer i et lagerområde på ialt 8860 bytes, og vi havde ofte alle 16 indgange i brug samtidig. Endvidere har undervisningsprogrammer med en eller anden form for database erfaringsmæssigt en uhyggelig tendens til at vokse og vokse, man kan ikke bare smide opgaverne væk, når undervisningen skal differentieres. Der blev, som det fremgår, efterhånden stillet større og større krav til disk-plads og organisering af databaser og programmel. Det komplekse system skulle i undervisningssituationen, for læreren, der absolut intet kender til EDB, være lige så nemt at gå til, som det at tage en bog på biblioteket eller en båndoptager med tilhørende bånd. Vi mener selv, at dette efterhånden er ved at være almindeligt, idet læreren - for at køre en serie vilkårligt valgte opgaver - skal kunne 4 ting for at "køre"dansk i sin klasse. Den første af disse 4 ting er: læreren skal kunne tænde/slukke for dataskærmen! Det videre forløb vil jeg senere komme ind på i en artikel om anvendelsen af datamaskinen i undervisningen.

Udbygning til RC8000

Den påtænkte udvidelse skulle efter planen være køb af en RC8000 med installation i december 1978.

Pr. 1./8. 79 rådede vi over 28 terminaler, 1. december 79 steg dette tal til 31 og er i skrivende stund vokset med yderligere 5 til ialt 36 - en håbløs opgave for RC7000. Store var derfor vore forventninger til RC8000. Men den software (BASIC/COMAL) vi fik leveret i dec. 78 - jan. 79 svarede ikke til vore forventninger. Vi klarede os på trods af dette igennem foråret 1979. Her kom arbejdet med COMAL75 og Tønder-folkene til virkelig stor hjælp.

Den 26. april 1979 trådte Regnecentralen i likvidation. Vi klarede os rimeligt igennem de økonomiske problemer, der i denne forbindelse opstod, og indgik i et fornuftigt samarbejde med A/S Regnecentralen af 1979 efter reorganiseringen, så det tidligere samarbejde kunne fortsætte.

BASIC/COMAL må i dag siges at være rimeligt godt kørende på RC8000. Der er absolut ting der skal rettes, men dette er allerede i gang. RC8000 indfrier derfor vore forventninger. Vi har nu ofte 20 terminaler i gang samtidigt og kan sikkert øge dette antal væsentligt, når, de sidste "lus" er pillet af BASIC/COMAL. Den "gamle" RC7000 er blevet til en RC3600 frontend til RC8000, der p.t. har 128 kw, hertil hører 2 diske á 66 MB, 2 stk. 2.4 MB diablodiske plus alt det løse udstyr fra RC7000.

I løbet af februar 1980 udvides RC8000 yderligere med 64 kw halvlederlager, bl. a. med det formål at kunne eksperimentere med tilkobling af microdatamater ved siden af den kørende BASIC-proces i maskinen, idet der givetvis skal være en form for software-interface mellem værtsmaskinen og microdatamaterne.

Endvidere etableres en magnetbåndstation for at kunne tage backup-kopier af systemet lettere, end vi kan nu, og for at gøre leveringen af software fra RC lettere og billigere.

Fremtiden

Vi forventer os meget af den nærmeste fremtid både i forholdet til RC af 1979 og til et større samarbejde på landsplan omkring både programmel og maskinel.

De første 3 microdatamater er allerede bestilt hos Regnecentralen. Vi kan således i løbet af kort tid gå ind i forsøg med anvendelse af disse lokalt på den enkelte skole og som opkoblingsmulighed til vor RC8000 her på Højstrupskolen, hvorfra dele af de store undervisningsprogrammer kan hentes ud til microdatamaten for lokal afvikling.

Jeg må til slut i denne artikel slå fast, at RC8000 udelukkende anvendes til undervisningsformål og ikke til administrative formål. Den daglige drift og vedligeholdelse foretages af undertegnede og 2 kolleger fra Bolbro skole (Poul Erik Hubert og Karl Johan Jørgensen) i samarbejde med en kontaktlærer på hver af de skoler, der har terminal(er). Herved kan information fra DOS relativt enkelt distribueres til alle interesserede kolleger.

Kontaktlærerordningen har i de forløbne år stået sin prøve som garant for decentraliseringsprocessen - ikke mindst i "det forsømte forår" i 1979.

Torben Højrup

COMAL 80 - hvorfor og hvordan

af Børge Christensen

I det første nummer af DATALÆRE (oktober 1976), skrev jeg en artikel om COMAL (COMAL - hvorfor vi lavede det - og hvordan). Jeg sluttede artiklen med ordene: "for øvrigt er jeg begyndt at spekulere over, om man ikke - nå, nej, det kan være til en anden gang."

Et tilbageblik

Det, mine overvejelser drejede sig om, havde at gøre med den kendsgerning, at COMAL på et vigtigt punkt aldrig var blevet gjort helt færdig. I efterskriften til den nævnte artikel stod den daværende COMAL version omtalt som COMAL II, og der fandtes allerede på dette tidspunkt en COMAL III, som indeholdt én af de faciliteter, som burde have været med i alle COMAL-versioner, idet den tillod at man anvendte *parametre* i forbindelse med procedurer. Der var ganske vist tale om et noget primitivt parameterbegreb, men det fungerede fejlfrit, og i hænderne på dygtige programmører blev der skrevet ret store programmer i COMAL III. Herunder bragte man i erfaring, at selv primitive parametermekanismer kan være uhyre nyttige ved skrivning af programmer, der er så store, at de får systemkarakter. Det kan i forbifarten bemærkes, at den tidligste definition af COMAL fra 1974 også indeholdt et forslag til parametre i forbindelse med underprogrammer.

Det lykkedes ikke at overtale leverandøren af vor minidatamat til at implementere den nævnte facilitet i den udgave af COMAL, som fra midten af 1977 blev den mest udbredte. Denne COMAL - som for øvrigt slet ikke blev kaldt COMAL - indeholdt til gengæld så store forbedringer af anden art, at vi alligevel gik over til at bruge den, og parametrene blev gemt, *men ikke glemt* til bedre tider.

The Micro Revolution

I 1978 kom der igen gang i udviklingen. Navne, som slet ikke lyder "computeragtige", som fx. PET, APPLE eller TRS-80, i forbindelse med sære science-fiction-klingende betegnelser, som fx. 8080, 6502 eller Z80, indvarslede "The Micro Revolution" og dermed, at minidatamaternes tid i den elementære undervisning snart ville være forbi. Ved et sært spil af omstændigheder blev jeg i 1978 inddraget i udviklingen af en mikrodatamat til skolebrug, og jeg så, at der nu var mulighed for at få realiseret nogle af de idéer om COMAL, vi tidligere havde haft og i mellemtiden havde gennemtænkt til større modenhed.

Den første "mikro-COMAL" blev imidlertid slet ikke skrevet i Tønder, men på DTH, hvor Tom Østerby tog initiativet til at få udviklet en COMAL-version til den mest udbredte af mikrodatamaterne, Intel 8080. Denne version vil være kendt af mange af læserne, idet den er implementeret på RC701 og - dog i en noget udvidet form - på SPC/1. ID-COMAL tillader for øvrigt brugen af parametre i forbindelse med procedurekald.

COMAL 80!

I løbet af foråret 1979 skrev jeg definitionen på en COMAL-version, som jeg nogen tid forinden havde døbt "COMAL 80". Der har været nogen debat mellem visse firmaer om "retten" til at bruge navnet COMAL 80. Jeg har blandet mig så lidt som muligt i denne noget mærkværdige disput, men jeg vil tillade mig at benytte denne lejlighed til at slå fast, at navnet "COMAL 80" kom til verden på RC-computers hovedkontor i Hannover tirsdag den 26. september 1978 sidst på eftermiddagen! Og det blev ved den lejlighed udtalt "COMAL-achtzig".

Starten

Der var altså efterhånden tale om mindst tre COMAL-versioner, og - som bevis på, at miraklernes tid ikke er forbi - blev der i oktober 1979 etableret en arbejdsgruppe bestående af H. B. Hansen, Tom Østerby og undertegnede samt repræsentanter for de firmaer, som er interesseret i COMAL. Tom Østerby blev sekretær for gruppen og har i denne egenskab udarbejdet en rapport, som definerer den endelige version af COMAL 80. Rapporten er for tiden til udtalelse hos interesserede parter, bl. a. datalæreforeningen. Den har også været en tur i England, og de eneste væsentlige kommentarer til rapporten er kommet fra John Hammond, der er lektor i compilerteori ved City University of London. Det skal dog bemærkes - og blev også understreget af John Hammond - at der kun er tale om kritik af visse detaljer og påpegning af et par regulære fejl i rapporten. Den overvejende del af COMAL 80-rapporten synes at kunne stå for en nærlæsning.

Comal 80 eller kaos?

Desværre blev gruppen ikke færdig med at definere forslag til filsystemer i forbindelse med COMAL 80, og da en række firmaer enten allerede har eller meget snart vil have færdigimplementeret COMAL 80, må det nok forudses, at der på dette vigtige område bliver visse afvigelser fra version til version. Men det totale kaos er blevet undgået, så COMAL 80 programmer, der er skrevet på én datamat med ingen eller ganske få ændringer vil kunne køre på en hvilken som helst

anden, der råder over en COMAL 80 fortolker. Og det bør betones, at disse ændringer aldrig vil berøre væsentlige dele af programmet som fx. strukturen eller nøgleordene. Der kan altså aldrig blive tale om, at man skal rette SELECT til CASE og CASE til WHEN for nu at nævne et eksempel, som nok vil gøre indtryk på nogle af læserne, og man vil heller aldrig komme ud for, at sætninger skal flyttes eller afsnit helt omskrives, fordi strukturer i den ene COMAL-fortolker afviger fra strukturer i den anden eller helt mangler.

Hvordan udveksler vi?

Det er i denne forbindelse værd at bemærke, at et betydeligt større problem kun har været flygtigt berørt i og slet ikke er løst af gruppen. Det drejer sig om et medie, der kan bruges til overførsel af programmer fra én maskine til en anden. Det vil naturligvis være problemløst at flytte et COMAL-80-program fra en COMET i Esbjerg til en COMET i København eller fra en RC702 i Odense til en RC702 i Århus, men hvordan med at flytte et program fra en COMET i Tønder til en RC702 i Ålborg? Eller omvendt - man kunne jo få en lys idé i Ålborg! Det forlyder ganske vist at både ICL og RC vil bruge CP/M-operativsystemet, men det gælder med sikkerhed ikke for DDE, som har bandlyst CP/M fra deres enemærker i almindelighed og SPC/1 i særdeleshed (bl. a. fordi de forlængst har udviklet deres eget). De gode, gamle dage, da vi blot kunne sende hinanden papirstrimler med vore nyeste genistreger - eller patentprogrammer - er vel forbi. Jeg kan ikke i skrivende stund pege på nogen løsning af den art, som er rimelig billig og hurtig. Man kan selvfølgelig stadig benytte skrivere med ASR-udstyr, men dels er de meget langsomme, og dels findes de langt fra alle steder, og jeg går ud fra, at ingen vil købe en teletype blot for at kunne udveksle programmer. Ser man på erfaringerne fra udlandet, vil det eneste fornuftige vær at benytte *musikkassetter*. De er ganske vist ikke super-hurtige, men de er billige og pålidelige og en kombineret læser/skriver koster 300-400 kr. Umådelige mængder af programmer udveksles i dag over hele verden ved hjælp af musikkassetter. De er nemmere at håndtere og mindst lige så holdbare som papirstrimler, og den format-problematik som plager de andre og mere avancerede medier, lader sig let løse for kassetternes vedkommende. Man kan blot vælge den samme overføringsteknik, som bruges ved telefonmodem og så optage hyleriet på bånd i stedet for at sende det i øret på sagesløse modtagere. Det er tilforladeligt og uhyre billigt at installere. Datalæreforeningen bør reagere hurtigt på dette problem. Det er let og billigt at løse nu og måske komplet umuligt om et år. COMAL 80 har oven i købet indbygget særlige sætninger til håndtering af simple medier, men det vil jeg vende tilbage til senere i denne fremstilling.

(fortsættes næste nr.)

Rapport fra arbejdskonference

fra vor udsendte medarbejder

I april 1980 blev der afholdt en arbejdskonference i Sevres om "Microcomputers in secondary education". Der blev holdt 4-6 foredrag om dagen hvert efterfulgt af diskussion. Alt foregik på engelsk (ingen tolkning), hvilket var hård kost for nogle af de deltagende, og hvilket samtidigt gav et lidt skævt indtryk af, hvor der foregik noget spændende, idet diskussionerne blev stærkt præget af de personer, der havde engelsk som modersmål. Men en masse fik man hørt om de mildest talt forskellige måder, man greb sagen an på, når man ville indføre datamaskinen i skolen enten til datalære eller som hjælpemiddel i undervisningen.

Frankrig

Den mest radikale metode var nok den franske. Man havde i årene fra 1970-76 afprøvet 2 typer maskiner (fransk fabr.) og 1 sprog på enkelte skoler (15-18-årige elever). Sproget var LSE (Langage Symbolique d'Enseignement), og de har kørt med 3 udgaver af dette.

I 1976 fik lærere fra forskellige dele af landet 1 års fuldtids-kursus i brugen af maskinerne, og samtidig skulle de prøve at fabrikere undervisningsprogrammer. Efter dette år måtte de arbejde på egen hånd ude på de efterhånden 58 skoler, der deltog i forsøget. I 1979 installeredes 416 microcomputere på 219 "secondary-schools", og 36 af lærerne fra 1-års kurset fik et ekstra 3 ugers kursus og skulle så fremover fungere som kursusledere i deres landsområde, hvor lærere fra skolerne, der havde fået maskiner, fik et 3-4-dages kursus. Kursuslederne syntes, at det allerede nu var klart, at dette var et for tyndt grundlag for fornuftig brug af maskinerne.

De nye programmer, der igennem årene var blevet lavet, blev samlet centralt, afprøvet og korrigeret

COMAL 80 - hvorfor og hvordan

af Børge Christensen

(Fortsat fra sidste nr.)

Rekursive procedurer

I slutningen af juli modtog jeg omsider den første kørende version af COMAL 80, og de programeksempler, som ledsager artiklen, er skrevet og kørt på denne version. Jeg vil starte med et eksempel, som både demonstrerer nogle af de helt simple og nogle af de meget "dyre" udvidelser i forhold til de hidtil anvendte COMAL-versioner. Der er samtidig tale om et program, som læserne kender - eller let kan komme til at kende - teori og baggrund for. Der er nemlig tale om en *rekursiv* udgave af *quicksort*, og skulle man have glemt teorien for denne berømte algoritme, kan man passende grave DATALÆRE, nummer 2, 1979, frem af dyngen. På side 21 i dette udmærkede skrift begynder en af publikationens stjerneartikler under overskriften QUICKSORT (i det følgende henvises til denne artikel med: QS). Som man vil erindre eller kunne forvisse sig om beror quicksort på følgende grundidé: En vektor, som skal sorteres, opdeles i to delvektorer på en sådan måde, at den ene del med sikkerhed ikke indeholder komponenter, der kommer før komponenterne i den anden del, og derpå udføres quicksort på de to delvektorer hver for sig. Processen er altså født rekursiv, og som gennemgangen i den ovenfor nævnte artikel viser, kræver det en del raffineret bogholderi af få processen omskrevet, så den kan udføres iterativt, altså ved hjælp af løkker. COMAL 80 udgaven af quicksort er vist i fig . 1. Opdelingen af vektoren foregår efter den samme algoritme, som er beskrevet i artiklen fra 1979 (QS, side 22), og i det nye program udføres den af det programafsnit, der indbefatter linjerne 100-220 (læseren opfordres *kraftigt* til at sammenligne algoritmen og programafsnittet). Idet vi angiver hele opdelingsprocessen med ordene:

udfør opdeling

får vi følgende rekursive beskrivelse af quicksort:

proc quicksort
  udfør opdeling
  hvis der er en venstre delrække så
    udfør quicksort på denne delrække
  hvis der er en højre delrække så
    udfør quicksort på denne delrække
endproc quicksort

En sammenligning af denne algoritme med den tidligere givne (QS, side 29) viser straks, hvor forbløffende simpel den nye er. Hvordan kan det

```
0010 //* PROGRAM: QUICKSORT *//
0020 //* SKREVET I COMAL-80, COMET VERSION *//
0030 //* EFTER N. WIRTH: ALGORITHMS+DATASTRUCTURES ECT. *//
0040 //* AF BØRGE R. CHRISTENSEN *//
0050 //* DATO. TØNDER, DEN 29. JULI, 1980 *//
0060 //
0070 PROC QUICKSORT(V, H)
0080  EXEC SNAPSHOT(V,H,A,N)
0090  I:=V; J:=H
0100  X:=A((V+H) DIV 2) //SÆT PIVOT//
0110  REPEAT //OPDEL VEKTOR//
0120   WHILE A(I)<X DO
0130    I:=I+1 //RYK PILEN TIL HØJRE//
0140   ENDWHILE
0150   WHILE X<A(J) DO
0160    J:=J-1 //RYK PILEN TIL VENSTRE//
0170   ENDWHILE
0180   IF I<=J THEN
0190    W:=A(I); A(I):=A(J); A(J):=W //BYT A(I) OG A(J)//
0200    I:=I+1; J:=J-1 //OG SÆT PILE TIL NÆSTE POSITION//
0210   ENDIF
0220  UNTIL J<I
0230  IF V<J THEN EXEC QUICKSORT(V,J) //SORTER VENSTRE-DELRÆKKE//
0240  IF I<H THEN EXEC QUICKSORT(I,H) //SORTER HØJRE-DELRÆKKE//
0250 ENDPROC QUICKSORT
0260 //
0270 RANDOM
0280 INPUT "HVOR MANGE KOMPONENTER? ": N
0290 DIM A(N) //VEKTOR TIL SORTERING//
0300 TAB:=4
0310 FOR I:=1 TO N DO
0320  A(I):=RND(1,2*N)
0330 NEXT I
0340 FOR I:=1 TO N DO
0350  PRINT A(I),
0360 NEXT I
0370 PRINT
0380 SELECT LP:
0390 EXEC QUICKSORT(1,N)
0400 PRINT
0410 SELECT DS:
0420 FOR I:=1 TO N DO
0430  PRINT A(I),
0440 NEXT I
0450 END
0460 //
0470 PROC SNAPSHOT(VENST, HØJ, REF VEKTOR(), N) CLOSED
0480  VENST:+1; HØJ:+1
0490  PRINT " ",
0500  FOR I:=1 TO N DO
0510   PRINT VEKTOR(I),
0520  NEXT I
0530  PRINT
0540  PRINT TAB((VENST-1)*4);"▬";TAB((HØJ-1)*4);"▬"
0550  PRINT
0560 ENDPROC SNAPSHOT
```

Fig. 1.

Eksempel på udskrift

```
 16  18  11   8  17  15  17  10   9  11
 ▬                                   ▬

 16  11  11   8   9  15  10  17  17  18
 ▬                  ▬

  8  11  11  16   9  15  10  17  17  18
     ▬                  ▬

  8  11  11  10   9  15  16  17  17  18
     ▬                  ▬

  8   9  10  11  11  15  16  17  17  18
     ▬   ▬

  8   9  10  11  11  15  16  17  17  18
         ▬                  ▬

  8   9  10  11  11  15  16  17  17  18
             ▬   ▬

  8   9  10  11  11  15  16  17  17  18
                 ▬   ▬

  8   9  10  11  11  15  16  17  17  18
                     ▬   ▬

  8   9  10  11  11  15  16  17  17  18
                         ▬   ▬

  8   9  10  11  11  15  16  17  17  18
                             ▬   ▬

  8   9  10  11  11  15  16  17  17  18
                                 ▬   ▬
```

være? De processer, der skal udføres, må da i grunden være de samme. Svaret er, at selvfølgelig udføres de samme processer, som beskrevet tidligere, men en væsentlig del af *styringen* af disse er nu *overladt til COMAL-fortolkeren*. Hele hemmeligheden ligger i, at *proceduren kan kaldes rekursivt*, eller sagt på en anden måde: proceduren kan "kalde sig selv". Sætningen:

hvis der er en venstre delrække så
udfør quicksort på denne delrække

er i programmet implementeret således (linje 230):

```
IF V<J THEN EXEC QUICKSORT(V,J)
```

At V er mindre end J, betyder netop at der er en delrække til venstre for pivoten, som skal sorteres. Denne delrække er angivet ved de to *parametre* V og J, som står i parentesen efter procedurekaldet. Parametrene V og J kaldes i denne sammenhæng for de *aktuelle* parametre, og deres værdier angiver indeks for hhv. første og sidste komponent i delrækken. Værdien af disse to parametre overtages af de to *formelle* parametre V og H, som står anført i procedurehovedet (linje 70):

```
PROC QUICKSORT(V,H)
```

dvs. V og H har efter kaldet de samme værdier som V og J. Men pas på! Det V, der står i procedurehovedet, og det V, der står som aktuel parameter i kaldet, bliver af systemet opfattet som *to forskellige variable*, og det er netop en af hemmelighederne ved det hele. Hver gang, proceduren kaldes, opretter systemet en *ny udgave* af de to formelle parametre V og H, og det er denne række af udgaver af V og H, i hvilket de enkelte elementer kan have helt forskellige værdier, der erstatter den *stak*, som vi selv måtte opretholde i den iterative version af quicksort (se QS, side 23 - 28). Det er naturligvis ikke nogen helt simpel sag at få fortolkeren til at overtage bogholderiet, men det kan brugeren være ligeglad med. Den dertil svarende hovedpine er overstået én gang for alle med skrivningen af fortolkeren.

I samme nummer af DATALÆRE, som jeg har henvist til i det foregående, findes i øvrigt en udførlig gennemgang af begrebet: rekursiv procedure. Artiklen hedder "Tårnet i Hanoi" og er skrevet af H. B. Hansen. Den begynder på side 7 i det omtalte nummer.

Brug af parametre

Parametre kan naturligvis også bruges, uden at der behøver at være tale om rekursion, og i langt de fleste anvendelser er der tale om simple tilfælde, hvor en procedure kaldes fra hovedprogrammet eller *en anden* procedure. Når jeg alligevel har valgt at bruge en rekursiv procedure som eksempel, hænger det sammen med, at dels er det i sådanne tilfælde, et system skal stå sin prøve, og dels, at jeg tidligere har gennemgået algoritmen i datalærebladet. Jeg forstår følgelig selv dette eksempel og kan på bekvemmeste vis henvise læserne til et arbejde, jeg har udført.

Programlisten er ledsaget af en udskrift, som er fremkommet ved kørsel af programmet. Denne udskrift administreres af proceduren SNAPSHOT (linje 470 - 580). I forhold til procedure QUICKSORT er der altså tale om et "assisterende program" (nyeste eufemisme for underprogram), og jeg vil ikke gennemgå den i sin helhed. Den indeholder imidlertid et par detaljer, som er karakteristiske for COMAL 80 og væsentlige for forståelsen af de idéer, der ligger bag denne nye udvidelse af COMAL. Hovedet af SNAPSHOT findes i linje 470 og ser sådan ud:

```
PROC SNAPSHOT(VENST, HØJ, REF VEKTOR (), N) CLOSED
```

og kaldet til SNAPSHOT findes i linje 80 og ser sådan ud:

```
EXEC SNAPSHOT(V,H,A,N)
```

De to formelle parametre VENST og HØJ får ved kaldet tildelt de samme værdier, som de aktuelle parametre V og H har. Noget tilsvarende gælder den formelle parameter N, der får samme værdi som den aktuelle parameter N, men der er igen grund til at understrege, at de to N'er af systemet opfattes og behandles som to *forskellige variable*. På listen over formelle parametre finder man også denne: REF VEKTOR(). Ordet "REF" er en forkortelse for "REFerence", der som bekendt betyder "henvisning". I forhold til tidligere COMAL-versioner er der tale om et helt nyt nøgleord. Når dette ord er anført foran en formel parameter, betyder det, at den pågældende parameter *henviser til* det *samme datafelt*, som den aktuelle parameter. Lad os se lidt nærmere på den her givne reference-parameter VEKTOR(). Den tomme parentes efter navnet betyder, at parameteren henviser til *én-dimensional vektor*, og ser vi efter i kaldet, kan vi konstatere, at den tilsvarende aktuelle parameter netop er navnet på en sådan én-dimensional vektor (A). Navnet VEKTOR bliver derved blot et *nyt navn for A*, sålænge der arbejdes *indenfor* rammerne af SNAPSHOT. Man kan sige, at VEKTOR er et "øgenavn" for A, som det er tilladt SNAPSHOT at bruge. Hver gang SNAPSHOT foretager sig noget med VEKTOR, er det i virkeligheden A, det "går ud over". - En dreng, der hedder Ole,

bliver kaldt Wolle af sin bror, så hver gang, Wolle er uvenner med sin bror, er det Ole, det går ud over.

I visse kredse, hvor man ynder at gøre sig ubegribelig på fremmede tungemål, siger man, at vektor bliver "called by reference", og om den anden type parametre, at de bliver "called by value". De tidligere nævnte parametre i COMAL III var alle henvisningsparametre. Jeg vil ikke på dette sted give flere og mere dybtgående eksempler på brugen af henvisningsparametre, men overlade til læseren selv at gøre sine erfaringer med dette redskab. For forfatteren er der imidlertid ikke nogen som helst tvivl om, at dersom der skulle vælges mellem værdi-parametre og henvisningsparametre, ville valget ubetinget falde på de sidste.

Lukket procedure

Procedurehovedet afsluttes med ordet "CLOSED". Tilstedeværelsen af dette nøgleord bevirker, som ordet selv siger, at proceduren er *lukket*. Det vil sige, at alle variable, der optræder i procedurens krop, er *lokale* for proceduren. Det betyder fx., at tælleren I, som optræder i linje 500 - 520, intet har at gøre med den tæller I, der optræder som pil i QUICKSORT. De to I'er opfattes af systemet som to forskellige variable. Lukkede procedurer er ganske vist ikke med i COMAL-80-kernen, men da de findes i både COMET-COMAL 80, RC-COMAL 80 og ID-COMAL 80, har jeg fundet det betimeligt at nævne begrebet her. I undervisningen kan de blive af meget stor betydning, idet man nu for første gang kan skrive en række biblioteksprocedurer som lukkede procedurer. Indholdet af disse "pakker" behøver ikke at være kendt i detaljer af brugeren, der blot skal vide, hvilke parametre, der kræves ved kaldet af dem. Vedkommende behøver ikke at bekymre sig om, hvad de variable i den lukkede procedure hedder, idet disse ikke på nogen måde kan komme til at påvirke de variable, som anvendes i det program, fra hvilket proceduren kaldes. Vi kan fx. tænke os, at man vil lukke proceduren QUICKSORT. Man skal da blot udstyre den med dette hoved:

```
PROC QUICKSORT(V, H, REF A()) CLOSED
```

Det er klart, at man er nødt til at anbringe henvisnings-parameteren A() på parameterlisten, idet den vektor A, der optræder i proceduren, nu er en rent lokal vektor, der kun kan knyttes til hovedprogrammets A ved hjælp af henvisningen. Man behøver ikke længere forklare, hvad QUICKSORT er for noget, men kan nøjes med at give besked om, at dersom en tal-vektor skal sorteres, kan man blot føje en kopi af QUICKSORT - som naturligvis står på programbiblioteket - til sit program og anvende et kald af formen:

```
EXEC QUICKSORT(FØRST,SIDST,VEKTOR)
```

For første gang er det virkelig muligt at bygge et helt COMAL-program op af "pakker", og begrebet "biblioteksprogrammer" får en helt anden og langt videre betydning, end det hidtil har haft i denne sammenhæng. Som man vil kunne se, er vi med begrebet lukket procedure meget nær ved at kunne benytte os af rigtige *externe procedurer*, og COMAL-gruppen har gjort sig overvejelser i den retning, men en række tekniske problemer er endnu ikke tænkt helt igennem.

Andre faciliteter

I øvrigt kan man i programmet se eksempler på følgende faciliteter i COMAL 80:

Kommentarer kan anbringes efter alle slags sætninger, og indledes med symbolet "//". Ordet REM kan stadig benyttes umiddelbart efter linjenumre, men vil af systemet blive ændret til "//", når programmet listes.

Der findes nu et rigtigt *tildelingstegn* i COMAL, nemlig det fra Algol og Pascal kendte ":=". Man kan stadig skrive fx. "A=5" eller "LET A=5", men systemet vil selv ændre disse to til "A:=5". Denne detalje kan måske synes betydningsløs, men det har vist sig, at brugen af det samme symbol for tildelingstegnet og lighedstegnet - som intet har med hinanden at gøre - ofte har ført til begrebsforvirring, især hos begyndere. Og hvad betyder i grunden følgende:

```
FUNDET=A=X(I)
```

En COMAL 80 fortolker ændrer selv dette til:

```
FUNDET:=A=X(I)
```

Der er endnu en grund til at bruge et særskilt tildelingstegn. I såvel COMET-COMAL 80 som RC-COMAL 80 kan man skrive fx.:

```
SALGSPRIS:+MOMS
```

og denne sætning har samme virkning som:

```
SALGSPRIS:=SALGSPRIS+MOMS
```

På tilsvarende måde kan ":-" bruges. Da "+" bruges ved *sammenkædning* af tekster (strenge), kan man også skrive fx.:

```
TEKST$ :+LINJE$
```

i betydningen:

```
TEKST$ :=TEKST$ + LINJE$
```

I programmets linje 480 har jeg benyttet ":+". Det skal dog bemærkes, at denne facilitet ikke er kommet med i COMAL 80 kernen. Man kan altså ikke forvente at finde den i alle COMAL 80 versioner. Når jeg alligevel har nævnt muligheden, er det fordi det har indgået i de overvejelser, der er gået forud for valget af et egentligt tildelingstegn.

I næste artikel vil jeg fortælle om bl. a. de nyheder der findes på områderne: vektorer, matricer og tekster.

Børge R. Christensen

Almindelige oplysninger om foreningen

Bestyrelsens sammensætning:

Formand: ERLING SCHMIDT
Revlingebakken 40, II, 9000 Ålborg, tlf. (08) 18 53 66.

Næstformand: WILLY KJELLBERG CHRISTENSEN
Strandpromenaden 32, 4900 Nakskov, tlf. (03) 92 30 34.

Sekretær: FRITZ G. KNUDSEN
Kollerupvej 17, 8900 Randers, tlf. (06) 43 49 04.

Kasserer: TORBEN HØIRUP
Karl Withsvej 2, 5000 Odense C, tlf. (09) 14 33 53.

HUGO JØRGENSEN
Olivenvej 11, Helsted, 8900 Randers, tlf. (06) 42 37 91.

GERD BELHAGE
Slettebjergvej 7, 2750 Ballerup, tlf. (02) 97 10 46.

TORSTEN ALF JENSEN
Langemarken 27, 5762 Vester Skerninge, tlf. (09) 24 22 35.

Henvendelser til foreningen:

Indmeldelser, adresseændringer o.l. til kassereren:

FORENINGEN FOR DATALÆRE OG ANVENDELSE AF EDB I UNDERVISNINGEN
Rismarksvej 80, 5200 Odense V, tlf. (09) 16 86 50.

eller til privatadressen.

Årskontingent: 90 kr. incl. blad. Studerende 45 kr.

Øvrige henvendelser til formanden.

BLADET:
Ansvarshavende redaktør:

TEDDY LANG PETERSEN
Holstedvej 7, 5200 Odense, tlf. (09) 16 90 56.

Henvendelser vedr. annoncer/stof:

Til redaktøren.

COMAL 80 - hvorfor og hvordan

af Børge Christensen

(Fortsat fra sidste nr.)

Selv om forfatteren nødigt vil indrømme det, findes der faktisk situationer, hvor en GOTO-sætning kan være en god udvej. Derfor er GOTO-sætninger heller ikke bandlyst i COMAL-80, selv om de kun bør benyttes i helt specielle tilfælde. For at gøre GOTO-sætninger lettere at anvende og forstå, er der indført etiketter (labels) i COMAL-80. Som eksempel på brugen af en etikette tager jeg et program, som H. B. Hansen har vist i DATALÆRE, nr. 2, 79, side 7. Programmet hedder BINSØG og er skrevet i "menneskecomal". Jeg glæder mig naturligvis over, at COMAL-80 åbenbart også er "menneskecomal". Det er lige netop det, der er meningen med det. Her er programmet:

```
0010 DIM A(100)
0020 FOR I:=1 TO 100 DO A(I):=I*3
0030 REPEAT
0040   INPUT "HVAD SØGES? ": X
0050   EXEC BINSØG(1,100)
0060   IF FUNDET THEN
0070     PRINT "FUNDET SOM NUMMER ",K
0080   ELSE
0090     PRINT "FINDES IKKE PÅ LISTEN"
0100   ENDIF
0110 UNTIL X=0
0120 //
0130 PROC BINSØG(FRA,TIL)
0140   IF FRA<=TIL THEN
0150     K:=(FRA+TIL) DIV 2
0160     FUNDET:=X=A(K)
0170     IF FUNDET THEN GOTO HALT
0180     IF X<A(K) THEN
0190       EXEC BINSØG(FRA,K-1)
0200     ELSE
0210       EXEC BINSØG(K+1,TIL)
0220     ENDIF
0230   ENDIF
0240 HALT:
0250 ENDPROC BINSØG
```

Læseren vil observere, at jeg har sat BINSØG ind i en testramme, så det kan prøvekøres. I linje 240 står der

```
HALT:
```

og det er netop en etikette, der virker som hopadresse for den betingede GOTO-sætning i linje 170. En etikette i COMAL-80 har samme form som et variabelnavn efterfulgt af tegnet kolon (:). En linje med en etikette kan afsluttes med en kommentar, men må ikke indeholde nogen dynamiske sætninger. Under listningen springer etiketterne i øjnene, idet de ikke bliver indrykket sammen med de øvrige sætninger. Dette viser følgende lille eksempel:

```
0010 REPEAT
0020   INPUT "HVAD SØGES? ": NAVN$
0030   FOR I:=1 TO MAXNR DO
0040     IF NAVN$=ELEV$(I) THEN GOTO FUNDET
0050   NEXT I
0060 FUNDET:
0070   PRINT ELEV$(I)
0080 UNTIL FALSE
```

Læg også mærke til, at det er tilladt at hoppe ud af en struktur. Kønt er det ikke, men det virker.

Væsentlige faciliteter til programmering af processer med tekster er vist i det program, der findes som bilag til artiklen. Programmet hedder TEKSTSTAT og i overskriften til det står forklaret, hvad dets formål er. I linje 70 og 80 er vist erklæringer af en række tekstvariable. Man vil se, at erklæringerne er af denne form:

```
DIM CIFRE$ OF 10
```

Grunden til, at længden af den variable har fået sit eget nøgleord OF, er et ønske om at adskille begrebet en variabels maksimale længde fra begrebet en tabels dimensioner. I tidligere COMAL-versioner (og i de fleste BASIC-versioner) ville følgende sætning:

```
DIM A$(30),B(30)
```

kunne give anledning til alvorlige misforståelser. At B(30) betyder erklæring af en tal-tabel med 30 komponenter, er der ingen tvivl om. Men hvad betyder A$(30)? I nogen tilfælde kan det betyde, at der erklæres en teksttabel med op til 30 komponenter. Det gælder faktisk for Microsoft BASIC, der er den mest benyttede BASIC overhovedet. I RC-7000 BASIC betyder det derimod, at der erklæres en tekstvariabel med plads til højst 30 tegn. Rigtigt galt bliver det, når man erklærer således:

```
DIM A$(100,30),B(100,30)
```

Her er B åbenbart en dobbelt tal-tabel med 100 gange 30 komponenter, mens B$ kan være en enkelt teksttabel med 100 komponenter hver på op til 30 tegn, eller en dobbelt teksttabel med 100 gange 30 komponenter, alt afhængig af, hvilken COMAL eller BASIC, man har fået fat i. I COMAL-80 er der ingen tvivl:

```
DIM A$(100) OF 30
```

betyder det første, mens

```
DIM A$(100,30) OF 30
```

betyder det sidste. Jeg skal senere vende tilbage til DIM-sætningen og dens definition i COMAL-80, samt de udvidelsesmuligheder denne definition rummer.
I linje 310 finder man sætningen:

```
IF TEGN$ IN BOGSTAVER$ THEN
```

Nøgleordet IN angiver en ny relationsoperator, og det Boolske udtryk: TEGN$ IN BOGSTAVER$ antager værdien sand, hvis værdien af TEGN$ findes som en deltekst i værdien af BOGSTAVER$. Hvis ikke det er tilfældet, antager udtrykket værdien falsk. Operatoren IN kan dog også bruges til at stedfæste, hvor delteksten evt. er fundet. Således vil den variable STED i sætningen:

```
STED:="OLSEN" IN "ANTON OLSEN"
```

få tildelt værdien 7, når sætningen udføres, idet delteksten "OLSEN" af "ANTON OLSEN" begynder med det 7. tegn. Operatoren IN er brugt flere steder i programmet, og læseren kan få et godt indtryk af dens anvendelighed ved at studere det nøjere.

I linje 180 - 210 finder man en REPEAT-løkke, der slutter med:

```
UNTIL EOD
```

Her betegner EOD (End-Of-Data) en Boolsk funktion, der får værdien falsk, når læsningen i en data-kø påbegyndes, og værdien sand, når det sidste element i køen er læst af READ-sætningen (190). Funktionen kan lettest jævnføres med den gammelkendte Boolske funktion EOF (End-Of-File), der benyttes i forbindelse med sekventielle filer.

I linje 270 finder man et eksempel på endnu en ny sætningskonstruktion i COMAL-80:

```
FOR I:=1 TO LEN(TEKST$)
         DO EXEC ANALYSE(TEKST$(I))
```

altså en FOR-løkke på én linje. Denne form af FOR-løkken har opbygningen:

```
FOR "var":="beg.værdi" TO "slutværdi"
                          DO "sætning"
```

og er naturligvis konstrueret i analogi med den gammelkendte:

```
IF "Boolsk udtryk" THEN "sætning"
```

En lignende udvidelse er sket for WHILE-løkkens vedkommende. I COMAL-80 kan man fx. skrive:

```
WHILE X(I)<I DO I:=I+1
```

i stedet for:

```
WHILE X(I)<I DO
   I:=I+1
ENDWHILE
```

Det første er unægtelig noget mere elegant og overskueligt end det sidste. For REPEAT-løkken findes ikke noget tilsvarende.

I linje 520 står der:

```
ZONE:=30
```

Når denne sætning udføres, sættes bredden af skrivezonen lig med 30 tegn. Hvis ZONE ikke eksplicit får tildelt nogen værdi, sætter systemet den til 0, hvilket betyder, at skilletegnet komma (,) er helt neutralt. Virkningen af zoneværdien kan ses af udskriften til programmet. Til gengæld giver skilletegnet semikolon (;) altid anledning til udskrift af et mellemrum, også når det adskiller tekster. Begge tegn er altså særdeles veldefinerede i COMAL-80. Skulle man få en version af COMAL-80, der ikke overholder disse normer for skilletegnene i PRINT-sætninger, henvender man sig blot til leverandøren og forklarer vedkommende, at han har misforstået noget.

I linje 330 findes et eksempel på brugen af det særlige tildelingstegn :+:

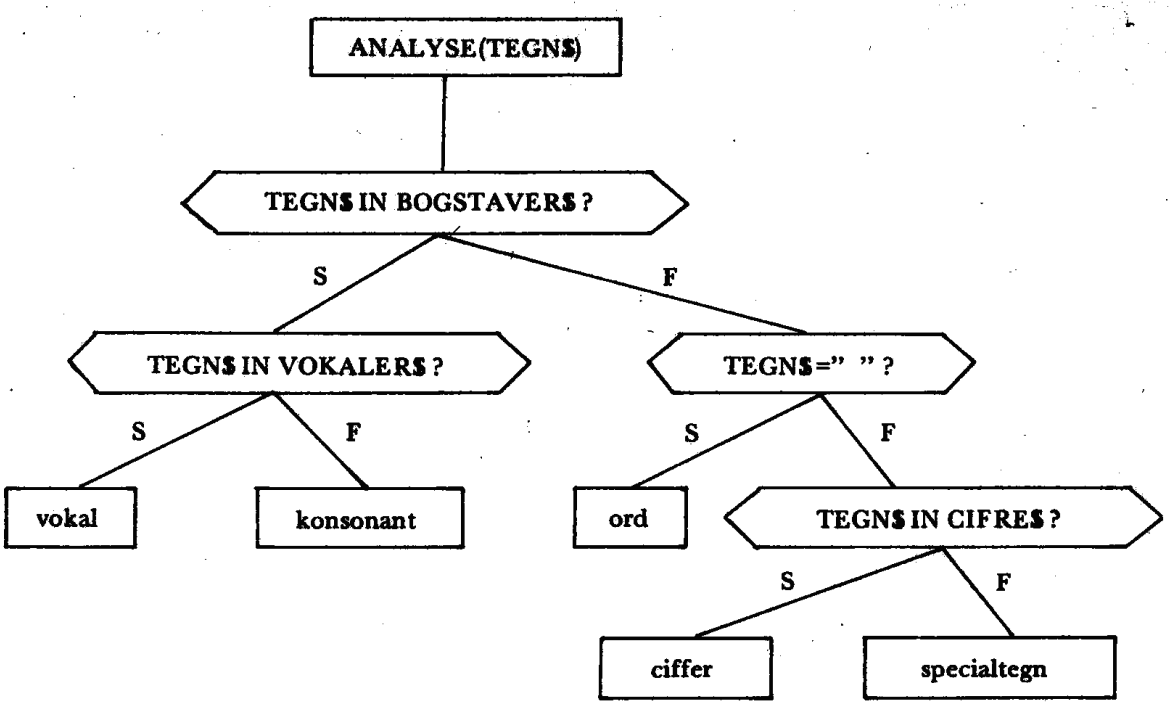
```
ANTALVOKALER:+1
```

der har samme virkning som:

```
ANTALVOKALER:=ANTALVOKALER+1
```

Denne facilitet er dog ikke standard for COMAL-80 og findes muligvis ikke i alle versioner.

Til slut viser jeg et strukturdiagram over proceduren
ANALYSE:




```
0010 //PROGRAM: TEKSTSTAT
0020 //TÆLLER ANTAL ORD, VOKALER, KONSONANTER,
0030 //CIFRE OG SPECIALTEGN I EN FORELAGT TEKST.
0040 //SKREVET I CBM COMAL-80.
0050 //BØRGE R. CHRISTENSEN, DEN 14. JANUAR 1981.
0060 //
0070 DIM CIFRE$ OF 10, TEKST$ OF 500, LINJE$ OF 70
0080 DIM BOGSTAVER$ OF 30, VOKALER$ OF 10
0090 VOKALER$:="AEIOUYÆØÅ"; CIFRE$:="1234567890"
0100 BOGSTAVER$:="ABCDEFGHIJKLMNOPQRSTUVWXYZÆØÅ"
0110 //
0120 EXEC INDTAST
0130 EXEC TÆLOP
0140 EXEC UDSKRIV
0150 END  //SLUT PROGRAM: TEKSTSTAT//
0160 //
0170 PROC INDTAST
0180   REPEAT
0190     READ LINJE$
0200     TEKST$:=TEKST$+LINJE$+" "
0210   UNTIL EOD
0220 ENDPROC INDTAST
0230 //
0240 PROC TÆLOP
0250   ANTALORD:=0; ANTALVOKALER:=0; ANTALKONS:=0
0260   ANTALCIFRE:=0; ANTALSPECIAL:=0
0270   FOR I:=1 TO LEN(TEKST$) DO EXEC ANALYSE(TEKST$(I))
0280 ENDPROC TÆLOP
0290 //
0300 PROC ANALYSE(TEGN$)
0310   IF TEGN$ IN BOGSTAVER$ THEN
0320     IF TEGN$ IN VOKALER$ THEN
0330       ANTALVOKALER:+1
0340     ELSE  //MÅ VÆRE KONSONANT//
0350       ANTALKONS:+1
0360     ENDIF
0370   ELSE  //IKKE ET BOGSTAV//
0380     IF TEGN$=" " THEN //MELLEMRUM?//
0390       ANTALORD:+1 //GODT ORD IGEN//
```

```
0400      ELSE  //CIFFER ELLER SPECIALTEGN//
0410        IF TEGN$ IN CIFRE$ THEN
0420          ANTALCIFRE:+1
0430        ELSE  //SPECIALTEGN//
0440          ANTALSPECIAL:+1
0450        ENDIF
0460      ENDIF
0470    ENDIF
0480 ENDPROC ANALYSE
0490 //

0500 PROC UDSKRIV
0510   SELECT OUTPUT LP
0520   ZONE:=30
0530   PRINT
0540   PRINT "ANTAL ORD:",ANTALORD
0550   PRINT "ANTAL VOKALER:",ANTALVOKALER
0560   PRINT "ANTAL KONSONANTER:",ANTALKONS
0570   PRINT "ANTAL CIFRE:",ANTALCIFRE
0580   PRINT "ANTAL SPECIALTEGN:",ANTALSPECIAL
0590   SELECT OUTPUT DS
0600 ENDPROC UDSKRIV
0700 //
0800 DATA "EFTERHANDEN, SOM MÆNGDEN AF PROGRAMMEL VOKSER, BLIVER DET"
0810 DATA "MERE OG MERE NØDVENDIGT, AT PROGRAMMERNE BLIVER SÅ LETTE"
0820 DATA "AT LÆSE, SOM MULIGT. KOMMUNIKATIONEN MENNESKE TIL MENNESKE"
0830 DATA "BLIVER OGSÅ PÅ DETTE OMRÅDE VIGTIGST AF ALT OG LANGT VIGTIGERE"
0840 DATA "END KOMMUNIKATIONEN MENNESKE-MASKINE. DERFOR ER COMAL-80 BEDRE"
0850 DATA "END BASIC, OG DERFOR MÅ UDVIKLINGEN AF ENDNU BEDRE SPROG IKKE"
0860 DATA "STANDSE. FORSTÅR VI IKKE DETTE, VIL VI BYDE VORE BØRN OG"
0870 DATA "BØRNEBØRN AT ÆDE STENE FOR BRØD."
```

```
ANTAL ORD:              71
ANTAL VOKALER:          141
ANTAL KONSONANTER:      226
ANTAL CIFRE:            2
ANTAL SPECIALTEGN:      12
```

Seminar på AUC

Torsdag d. 26. - fredag d. 27. marts afholdes et seminar på AUC med titlen: "Hvilken indflydelse kan EDB, audiovisuelle medier og telekommunikation forventes at få på undervisning?"
Der er lagt op til et spændende program, hvor man virkelig prøver at komme rundt til alle emneområder, og der vil i forbindelse med seminaret også blive en udstilling og mulighed for praktisk arbejde i en workshop, hvor bl. a. CDC (PLATO), Regnecentralen, Bellevue studierne og Jysk Telefon (teledata) er med.
Seminargebyr, der inkluderet forplejning, er 775 kr. og tilmelding skal være foretaget inden 17. feb. 1981.
Yderligere oplysninger og tilmeldingsblanket kan fås fra:
Hans Hessellund
CBE-gruppen, AUC
Aalborg Universitetscenter
Langagervej 6
9220 Aalborg Ø.

●●●

Microdatamat-anmeldelser

Matematisk Institut på Danmarks Lærerhøjskole har altid været aktiv og positiv overfor datalære og anvendelser af edb i undervisningen.
DISK-projektet, hvor der med jævne mellemrum kommer nye programmer og opgavesæt, er et eksempel på én af aktiviterne. En anden er, at man "anmelder" forskellige microdatamater og hidtil har man haft ABC80, PET 2001, RC701 og APPEL II under behandling. Der er ikke tale om en egentlig test, men netop en anmeldelse eller beskrivelse, omend der i en konklussion gives udtryk for en helhedsvurdering af maskinen. Men også i disse subjektive vurderinger er der relevante oplysninger at hente, så alt i alt er det nyttige informationsblade.

COMAL 80 - hvorfor og hvordan

af Børge Christensen

(Fortsat fra sidste nr.)

De versioner af COMAL-80, som hidtil er blevet implementeret, er alle udstyret med et simpelt, men ret effektivt fil-håndterings-system. Det lykkedes desværre ikke for arbejdsgruppen at nå frem til et endeligt forslag til et filsystem, skønt man i grunden var tæt på at være enige. På grundlag af allerede eksisterende idéer og notaterne om disse, findes der dog nu en faktisk standard, udtrykt i de allerede kørende systemer. Fælles for disse er, at man kan skrive såvel program- som datafiler i det ydre lager, og at dette kan ske enten i tekst-format (ASCII-format) eller i internt format (binær-format). Man synes også at være enige om, at

```
0010 DIM NAVN$ OF 20, ADR$ OF 20, BY$ OF 20
0020 REPEAT
0030   PRINT CHR$(147) //CLEAR SCREEN//
0040   FOR I:=1 TO 10 DO PRINT  //10 LIN. NED//
0050   PRINT "1 = OPRET NY LISTE"
0060   PRINT "2 = INDSÆT ADRESSÆT"
0070   PRINT "3 = SLET ADRESSÆT"
0080   PRINT "4 = SKRIV LISTE"
0090   PRINT "5 = STOP"
0100   PRINT
0110   PRINT
0120   INPUT ">>  ": JOB
0130   //
0140   CASE JOB OF
0150   WHEN 1
0160     EXEC OPRET
0170   WHEN 2
0180     EXEC INDSÆT
0190   WHEN 3
0200     EXEC SLET
0210   WHEN 4
0220     EXEC LISTE
0230   WHEN 5
0240     STOP
0250   OTHERWISE
0260     //GENTAG JOBKALD//
0270   ENDCASE
0280 UNTIL FALSE
0290 //
0300 PROC OPRET
0310   OPEN 2,"ADRESSER",RANDOM 80
0320   INPUT "NAVN.:.....": NAVN$
0330   I:=1
0340   WHILE NAVN$<>"" DO
0350     INPUT "ADRESSE.....": ADR$
0360     INPUT "POSTNR. BY.....": BY$
0370     I:=I+1
0380     WRITE FILE 2,I: NAVN$,ADR$,BY$
0390     INPUT "NAVN.....": NAVN$
0400   ENDWHILE
0410   MAX:=I
0420   WRITE FILE 2,1: MAX
0430   CLOSE
0440 ENDPROC OPRET
0450 //
0460 PROC INDSÆT
0470   //OVERLADES TIL LÆSEREN//
0480 ENDPROC INDSÆT
0490 //
0500 PROC SLET
0510   //OVERLADES TIL LÆSEREN//
0520 ENDPROC SLET
0530 //
0540 PROC LISTE
0550   PRINT CHR$(147) //CLEAR SCREEN//
0560   PRINT "1 = SKÆRM"
0570   PRINT "2 = PRINTER"
0580   PRINT
0590   INPUT ">>  ": ENHED
0600   IF ENHED=2 THEN SELECT OUTPUT "LP"
0610   OPEN 5,"ADRESSER",RANDOM 80
0620   READ FILE 5,1: MAX
0630   FOR I:=2 TO MAX DO
0640     READ FILE 5,I: NAVN$,ADR$,BY$
0650     PRINT NAVN$
0660     PRINT ADR$
0670     PRINT BY$
0680     FOR LN:=1 TO 3 DO PRINT  //3 LIN.//
0690     IF ENHED=1 THEN INPUT BY$ //VENT PÅ RETURN//
0700   NEXT I
0710   CLOSE
0720   SELECT OUTPUT "DS"
0730 ENDPROC LISTE
0740 //
```

LIST "filnavn"

skriver et program ind i det ydre lager i tekst-format, mens

ENTER "filnavn"

omvendt læser et program ind fra det ydre lager, når dette program står skrevet i tekst-format. Hvis et program skrives i det ydre lager med kommandoen:

SAVE "filnavn"

bliver det repræsenteret i internt format og kan kun læses ind i arbejdsområdet med kommandoen:

LOAD "filnavn"

For brugeren er det tilstrækkeligt at vide, at SAVE og LOAD kommandoerne normalt giver anledning til hurtigere operationer end LIST og ENTER kommandoerne, men at LOAD kommandoen sletter det nuværende indhold af arbejdslageret, hvad ENTER kommandoen ikke gør. Hvis man skal sammenflette programmer, er man altså henvist til at bruge LIST/ENTER-parret.

CHAIN

Et program, som står skrevet i en fil, kan også kaldes op og startes af et andet program med sætningen:

CHAIN "filnavn"

Et program, der på denne måde skal bruges som „ydre procedure", skal stå skrevet i internt format, dvs. det skal være skrevet ind med LOAD-kommandoen. I et enkelt af de systemer, der kører COMAL-80, nemlig CBM's, kan CHAIN også bruges som kommando.

LÆSNING OG SKRIVNING

Hvis man ønsker at skrive eller læse i en datafil, skal denne først åbnes med en sætning, som denne

OPEN 'numerisk udtryk'," 'filnavn' ",'måde'

Sætningen knytter en kanal med nummeret, givet ved 'numerisk udtryk' (en konstant, en variabel eller en formel) til den fil, der er angivet ved 'filnavn'. Filen kan åbnes i forskellige måder, nemlig

READ: læsning i en sekventiel fil
WRITE: skrivning i en sekventiel fil
APPEND: skrivning i forlængelse af allerede oprettet sekventiel fil.
RANDOM 'postlængde': skrivning eller læsning i en fil med direkte tilgang.

Det bemærkes, at postlængden altid skal angives efter nøgleordet RANDOM (antal bytes).

Eksempel:

```
OPEN 2,"MEDLEMMER", RANDOM 80
```

BEMÆRKNING. 'filnavn' kan i nogle versioner indeholde oplysninger om den ydre enhed, i hvilken filen skal skrives eller læses. I RC 702-COMAL-80 og CBM-COMAL-80 kan man endvidere forsyne filnavnet med et præfix, som bevirker, at evt. eksisterende udgave af filen slettes, når der skrives i den pånv. I RC 702-COMAL-80 bevirker således sætningen:

```
OPEN 2,"NEW. MEDLEMMER", WRITE
```

at filen MEDLEMMER kan overskrives uden videre, og noget tilsvarende bevirker i CBM-COMAL-80 sætningen:

```
OPEN 2,"@0: MEDLEMMER", WRITE
```

Der er planer om at ændre CBM-COMAL-80, så den tillader helt samme skrivemåde som RC 702-COMAL-80.

Ved skrivning i sekventielle filer kan man benytte sætningerne:

```
PRINT FILE 'numerisk udtryk': 'feltliste'
WRITE FILE 'numerisk udtryk': 'feltliste'
```

PRINT FILE sætningen skriver data på tekstform, mens WRITE FILE sætningen skriver data på binær form. Det numeriske udtryk angiver kanalnummeret på den fil, man ønsker at skrive i. Den anførte 'feltliste' er en liste med de værdier (angivet ved konstanter, variable eller udtryk), som skal skrives i filen.

Ved skrivning i filer med direkte tilgang kan man benytte sætningen:

```
WRITE FILE'numerisk udtryk','postnummer': 'feltliste'
```

Eftersom der er tale om filer med direkte tilgang, skal der naturligvis angives postnummer, og det sker med en (numerisk) konstant, en variabel, eller et udtryk.

Eksempler:

```
FOR I:=1 to MAXNR DO WRITE FILE 2: N$(I), K(I)

FOR I:=1 TO MAXPOST DO
  WRITE FILE 5, I: MEDLNR(I), NAVN$(I), AFD$(I), LK(I)
NEXT I
```

Ved læsning i sekventielle filer benytter man sætningerne:

```
INPUT FILE 'numerisk udtryk': 'felter'
READ FILE 'numerisk udtryk': 'felter'
```

Her er 'felter' en liste med variable, der får tildelt de indlæste værdier.

Ved læsning i filer med direkte tilgang benytter man sætningen:

```
READ FILE 'numerisk udtryk','postnummer': 'felter'
```

Eksempler:

```
I:=1
WHILE NOT EOF (5) DO
  READ FILE (5), I: MEDLEM$(I)
  I:= I+1
ENDWHILE

READ FILE 2, 1: MAX
FOR I:= 2 TO MAX DO
  READ FILE 2, I: N$(I), T$(I)
NEXT I
```

LUKNING

Når man er færdig med at læse eller skrive i en fil, lukker man den med en af sætningerne:

```
CLOSE
CLOSE 'numerisk udtryk'
```

Når den første bruges, lukkes alle filer, der måtte være åbnet. Når den anden bruges, lukkes kun den fil, hvis kanalnummer er angivet ved det numeriske udtryk.

Eksempel:

```
CLOSE 3
```

SLETNING

I ICL-COMAL-„næsten 80" og i RC 702-COMAL-80 kan man slette filer af alle typer ved at bruge:

```
DELETE "filnavn"
```

der kan optræde både som kommando og som

sætning. I ICL-COMAL findes der nogle særlige regler for filnavnet, når det bruges i en DELETE-kommando eller -sætning.

AFSLUTNING

Ovenstående korte gennemgang gør naturligvis ikke krav på at være udtømmende, men skulle blot give en nogenlunde almen beskrivelse af de vigtigste fil-sætninger og -kommandoer i eksisterende versioner af COMAL-80. Læseren bør hæfte sig ved sætningernes opbygning mere end ved den omstændighed, at én version har et komma her og måske mangler et kolon hist. Som altid er der kun ét sted, man kan forvente - forhåbentligt - at få alle detaljer oplyst, nemlig i firmaernes manualer. Det har dog været meget opmuntrende for undertegnede, at man har kunnet enes så langt, som tilfældet er.

Som eksempel på brugen af filer i COMAL-80 er vedlagt et lille program, der kan bruges til skrivning af adresselister.

Den næste artikel om COMAL-80 hedder: Hvad der ikke kom med i denne omgang.

★★★★★★★★★★★★★★★★★★★★★★★★

MANUSKRIPTER

Redaktøren begynder igen at mangle stof. Går du med én eller anden idé, har du prøvet noget indenfor faget datalære eller anvendelsen af edb, er der et af de tidligere indlæg, du godt kunne tænke dig at svare på eller er der andet du godt ville indvie dine kollegaer i, så fat pennen eller skrivemaskinen og send det ind til redaktøren. Du har hele sommeren til det, dead - line for næste nummer er først midt i august.

★★★★★★★★★★★★★★★★★★★★★★★★

● ● ●

HP-85

Af Peter B. Yde

HP-85 er en handy mikrodatamat fra Hewlett-Packard. I en enhed (kasse), der vejer 8 kg., er samlet en centralenhed, tastatur, skærm, printer og en kassetteoptager, der kan sammenlignes med en diskstation. Den er programmérbar i en udbygget BASIC-version med bl.a. en fin grafik og gode fejlretningsmuligheder. Hele herligheden koster omkr. 30.000 kr. med moms.

HARDWARE

Datamatens dimensioner er 45 x 42 x 16 cm. Og som nævnt vejer det hele kun 8 kg.

Centralenheden indeholder 16 Kbytes, der kan udvides til 32 Kbytes. Efter mine målinger er maskinen en lille smule langsommere end visse andre mikrodatamater, men kun lidt. Den arbejder til gengæld med 12-cifrede tal. Cifferlængden kan dog afskæres til 5 v. hj. a. erklæringen SHORT.

Til forskellige tidspunkter har jeg haft forskellige eksemplarer af HP-85 (jeg har vist arbejdet med 6 maskiner), og jeg har haft lejlighed til at teste den grundigt. Ikke på noget tidspunkt har jeg været ude for, at en HP-85 udviste funktionsfejl. Overhovedet! I 1979 og 1980 har jeg testet en halv snes mikrodatamater, og kun HP-85 har jeg ikke på en eller anden måde haft problemer med, når jeg under problemer foruden funktionsfejl indregner fejlanvisninger i manualer og bøvl med at forbinde enheder og at få dem til at fungere. Da mine undersøgelser (der har været bragt i „Elektronik" og „Populær Radio") omfatter markedets vigtigste selvstændige, fuldt udbyggede mikrodatamater, vil det dog være rimeligt at nævne, at der er flere maskiner, jeg i denne forstand har meget lidt at udsætte på. F.eks. fandt jeg på ABC-80 blot en enkelt fejl.

HP-85's tastatur er velforsynet med taster. Der er taster for visse kommandoer som f.eks. NEW (der dog kaldes SCRATCH) og LOAD. Der er fine cursor- (markør-) betjeningsmuligheder. Anslaget af tasterne er behageligt. Maskinen er forsynet med mange tegn, også Æ, Ø og Å. Det kræver dog anvendelse af tre taster f.eks. at indtaste et Æ.

Skærmen er på 5 tommer diagonalt - dvs. lille. Den har 16 linjer á 32 tegn. Benyttes den til grafik, er der imidlertid 192 x 256 punkter på skærmen, hvilket er mange. Punkterne står pænt på skærmen, og det vil sige, at også tekst står meget klart. Man kan desuden ved betjening af en enkelt taste rulle de fire foregående skærmbilleder frem.

HEWLETT-PACKARD laver selv maskinerne fra ende til anden. Maskinens kassetteoptager er således en specialitet for firmaet. Den er praktisk talt lige så hurtig som en floppy diskstation. Det tager kun 4-5 sekunder at overføre et program fra hurtiglager til bånd eller omvendt. Desuden kan data lagres på båndet både sekventielt og som

Afsluttende artikel om COMAL-80

Hvad der ikke kom med denne gang

De fleste af mine læsere kender programstumper af formen:

```
I F  X<A(I) THEN
   I:=I+1
ENDIF
```

Formen virker noget tung, og de fleste vil sikkert foretrække at skrive:

```
IF X<A(I) THEN I:=I+1
```

Denne kortere og også mere overskuelige skrivemåde blev i COMAL-80 udvidet til også at gælde for FOR- og WHILE-løkker som vist i tidligere artikler.

Derimod nåede vi ikke frem til også at indføre en forkortet form af simple dobbeltforgreninger som denne:

```
I F  X<A(I) THEN
   I:=I+1
ELSE
   J:=J-1
ENDIF
```

I analogi med ovenstående kunne man tænke sig at skrive sådan:

```
IF X<A(I) THEN I:=I+1 ELSE J:=J-1
```

Der er altså tale om en "kort" IF-THEN-ELSE med følgende syntax:

```
IF "betingelse" THEN "sætning" ELSE
"sætning"
```

Grunden til, at den ikke uden videre er blevet indført, er, at den ikke er uproblematisk. For det første lider den af samme "svaghed" som de øvrige sammensatte sætninger: Man kan ikke have en sammensat sætning som en del af en sammensat sætning. Således vil følgende sætning normalt ikke være tilladt i COMAL-80:

```
IF X<A(I) THEN IF Q=5 THEN I:=I+1
```

Vor læser: "Ja, men det kan man da i mange BASIC'er". Vi: "Ja, men COMAL-80 er sandt for dyden ikke nogen BASIC".
Konstruktioner af sidstnævnte type kan hurtigt føre til sætninger, der er overordentligt uigennemskuelige, og man må trods mulige tab af "smartness" hellere skrive:

```
I F  X<A(I) THEN
   IF Q=5 THEN I:=I+1
ENDIF
```

Den helt korrekte syntax for den korte IF-THEN-ELSE forgrening kunne således være:

```
IF "betingelse" THEN "simpel sætn."
ELSE "simpel sætn."
```

hvor "simpel sætn." naturligvis er tilbørligt veldefineret.

Men der er mere endnu. Med lange variabelnavne og tilpas komplicerede Boolske udtryk kan man let nå ud over 80 tegn med en "kort" IF-THEN-ELSE forgrening, og i så fald vil linjen alligevel "knække", og hvor skal indrykningen på næste linje så sættes? Man kunne let få et tekstbillede, der strider mod et af grundprincipperne i COMAL: Tekstformatet skal afbilde programstrukturen (med tab af elegance i visse tilfælde). I Pascal programmer - hvor man selv kan bestemme - skriver jeg næsten altid den korte IF-THEN-ELSE på denne form:

```
IF X<A(I) THEN I:=I+1
          ELSEJ:=J+1
```

Dette og lignende formater var på tale i arbejdsgruppen, men det er klart, at det sidstnævnte giver tekniske problemer under listningen af programmet, hvor indrykningen af ELSE-delen påvirkes af længden af det Boolske udtryk i IF-delen. Jeg vil indtil videre konkludere, at der kan blive tale om to former:
Dels den førstnævnte på een linje og dels den sidstnævnte:

```
IF "betingelse" THEN "simpel sætning"
                ELSE "simpel sætning"
```

Jeg foretrækker den sidste, men den første er unægteligt den mest nærliggende at implementere.

I COMAL-80 skriver man funktioner på samme måde som procedurer. Dette kan lade sig gøre, fordi procedurer nu har uindskrænket parameteroverføring. Denne i sig selv elegante løsning - som jeg desværre ikke har æren for - efterlader imidlertid et grimt hul i den nuværende definition af COMAL-80: Man kan ikke definere funktioner af tekst-type. Det er let at indføre fx. en funktion VÆRDI, som kan bruges til at konvertere en cifferfølge i en tekst til en tal-type i sætninger som:

```
DAGNR:=VÆRDI(INDBUFFER$(8:7))
```

Den funktion - eller funktional-procedure om man vil - der bliver kaldt middelbart i denne sætning, må begynde nogenlunde således:

```
PROC VÆRDI (S$)
   . . .
```

Derimod kan man ikke definere den funktion, der er brug for i sætninger som:

```
CPRNR$:=TXT$(DATO)+"-"+TXT$(LBNR)
```

Funktionen TXT$ skulle være opbygget efter skemaet:

```
PROC TXT$(TAL) OF 80
   . . .
   TXT$:=. . .
ENDPROC TXT$
```

Som man vil se, er et af problemerne ved indførelse af tekst-funktioner forbundet med erklæringen af navnet, der også skal optræde som variabel. Arbejdsgruppen var inde på problemet, og det er i øvrigt een af grundene til, at dimensioneringen af tekster sker med brug af nøgleordet OF i COMAL-80. Funktioner af tekst-type kan dog forventes indført i CBM-COMAL-80 efter ovennævnte beskrivelse i nærmeste fremtid, idet denne version er forsynet med de "hægter", der er nødvendige for at indføre sådanne faciliteter.

En af COMAL's svagheder sammenlignet med fx. Pascal er manglen på datatyper og datastrukturer. Der findes i virkeligheden kun to typer, nemlig tal og tekster, og af strukturer kun tabeller og filer. En af de strukturer, der ofte savnes i COMAL - og helt sikkert i COMAL-80 - er poststrukturen. Jeg har foreslået, at man skal kunne definere en poststruktur som fx. denne:

```
REC PERSON
   NAVN$ OF 30, ADR$ OF 20, POSTDSTR$ OF 20
   CPRNR$ OF 11, AFDNR, CIVILST, UDD$(10) OF 10
ENDREC PERSON
```

Denne "model" skal derpå kunne bruges i DIM sætninger fx. således:

```
DIM ANSATð(10) OF PERSON, BUFFERð OF PERSON
```

Herefter er ANSATð en tabel, hvis enkelte komponenter består af poster, som defineret i PERSON, mens BUFFERð er en variabel af type PERSON. De 10 poster kan oprettes ved fx.:

```
FOR I:=1 TO 10 DO
   INPUT "NAVN .....": ANSATð(I).NAVN$
   INPUT "GADE HUSNR ...": ANSATð(I).ADR$
   INPUT "POSTNR. BY ...": ANSATð(I).POSTDSTR$
   . . .
   PRINT "ANGIV UDDANNELSER:"
   J:=0
   INPUT "UDD. NR. 1:": IND$
   WHILE IND$<>"" AND J<10 DO
      J:=J+1; ANSATð(I).UDD$(J):=IND$
      PRINT "UDD. NR. "J,
      INPUT ":": IND$
   ENDWHILE
NEXT I
```

Bemærk, at posten indeholder en tabel. Posterne skal naturligvis kunne indeholde alle strukturer incl. nye poster!

Man skal også kunne anvende posterne således:

```
I F  ANSATð(I).NAVN$ <ANSATð(J).NAVN$ THEN
   BUFFERð:=ANSATð(I)
   ANSATð(I):=ANSATð(J)
   ANSATð(J):=BUFFERð
ENDIF //OMBYTNING AF POSTERNE I OG J//
```

eller således

```
FOR I:=1 TO 10 DO WRITE FILE 3,I: ANSATð(I)
```

Ved samtidigt at indføre den fra Pascal kendte WITH-sætning kunne man også benytte følgende:

```
FOR I:=1 TO 10 DO
   WITH ANSATð(I) DO
      PRINT NAVN$
      PRINT ADR$
      PRINT POSTDSTR$
      . . .
   ENDWITH
NEXT I
```

Det er planen at indføre poststrukturer i CMB's COMMERCIAL-COMAL i løbet af 1982 til kørsel på de nye 96 kB datamater. Den vil være ret resource-krævende, og man skal sikkert ikke forestille sig at finde den på datamater med kun 64 kB arbejdslager.

Til slut vil jeg omtale en vigtig udvidelse, som mig bekendt ikke tidligere har været diskuteret i "COMAL-kredse": Externe procedurer og funktioner, skrevet i assembler kode, men kaldt fra et COMAL-program, som om der var tale om sædvanlige COMAL procedurer! Altså ikke noget med CALL, USR, SYS, PEEK, POKE eller andre snurrepiberier. Denne udvidelse er blevet foreslået af Mogens Kjær, som har skrevet CBM-COMAL-80 fortolkeren, og den er bl. a. interessant derved, at den allerede er lavet i den seneste version af CBM-COMAL-80 og havde verdenspremiere i Lausanne i slutningen af juli måned 1981. Læseren kan fx. forestille sig en række assembler programmer til styring af skærmbilledet med henblik på vektor-grafik. Lad os tænke os følgende: MOVETO(X,Y), TURNTO(X,Y), MOVE(X), TURN(X), PENCOLOR(X$), osv. Alle disse procedurer er programmeret i assembler kode og ligger fx. i en EPROM, som starter ved adressen 8x4096. I COMAL-80 programmet skriver man nu sætningen:

```
OPTION 8x4096
```

(sammenlign evt. med Pascals USES TURTLEGRAPHICS). Herefter kan alle de i assemblerkode skrevne underprogrammer bruges af COMAL programmet, som om de var sædvanlige procedurer, skrevet af brugeren selv. Man kan fx. skrive:

```
EXEC MOVETO(20,20)
EXEC TURNTO(90)
EXEC PENCOLOR("RED")
EXEC MOVE(100)
```

En anden nærliggende anvendelse er processtyring. Man kan tænke sig procedurer som fx. BIT(X,Y), SET(X,Y), READPORT(X), WRITEPORT(X,Y) anvendt i følgende:

```
IF BIT(3,RØGAFGANG) THEN EXEC SET(4,LUFTTILFØRSEL)
```

eller

```
TILSTAND1:=READPORT(VANDTEMP)
```

hvor RØGAFGANG, LUFTTILFØRSEL og VANDTEMP er variable, som har fået tildelt talværdier svarende til de porte på microprocessoren, som er anvendt i de aktuelle systemer. Man kan naturligvis også have underprogrammerne liggende i et ydre lager, hvorfra de kan hentes ind i arbejdslageret til samarbejde med et COMAL-program. Inden COMAL-programmet indlæses - eller indtastes giver man kommandoerne:

```
OBJLOAD "TURTLE"
RUN
```

Derpå er assemblerkoden lagt på plads og beskyttet og COMAL-programmet skal blot indeholde en passende OPTION-sætning.

Jeg anser dette for at være en fantastisk god idé, fordi vi nu ikke behøver at udvide sproget med flere nøgleord, og vi risikerer ikke en vild vækst som i fx. BASIC. Man kan tilpasse COMAL til alle mulige anvendelser ved blot at føje nye kodestumper til, og man kan lade faciliteterne afhænge af anvendelserne. På den måde kan vi have fx. en COMMERCIAL COMAL, en PROCES COMAL, en COMAL TURTLE, osv. uden at flytte et komma i grundversionen!

Tilbage er blot et - som sædvanligt fremragende - forslag fra mig til datalæreforeningens bestyrelse: Udskriv snarest en prisbelønnet opgave med titlen:

SKRIV ET PROGRAM I COMAL-80 ELLER PASCAL, SOM OVERSÆTTER ET COMAL-80 PROGRAM TIL UCSD PASCAL OG SKRIVER OVERSÆTTELSEN I EN PASCAL PROGRAM FIL, KLAR TIL OVERSÆTTELSE TIL P-KODE.

Jeg er til disposition med yderligere specifikationer.

Og lad være med at være for fedtede!

Børge R. Christensen

* * * * * * * * * * * * * * * * * * * *

OBS! OBS!

Stof til næste nummer af bladet skal være redaktionen i hænde senest mandag, den 19. oktober 1981.

NYT OM COMAL

Interessen for COMAL ser ud til at brede sig uden for landets grænser. Det meget dynamiske firma Metanic har efterhånden fået sin Metanic COMAL-80 solgt til adskillige store firmaer, sidst til firmaet Osborne. Commodore, som er det ene af de tre søstre: Apple, Commodore og Tandy, synes omsider at have opdaget, at de har et pragtdyr i stalden, nemlig JKL gruppens CBM COMAL-80. Denne sidste version bliver for tiden implementeret på den nye COMMODORE 64 og bliver formentlig den første COMAL-version med højopløsnings-farve-grafik, som vil blive solgt over hele verden på en datamat til en virkelig lav pris. Der bliver tale om en indbygget version, som starter så snart der tændes for datamaten.

Det engelske firma Grundy Business Systems er netop gået på markedet med deres nye datamat, NewBrain, som bliver udstyret med en COMAL-80, der er skrevet af Grundys egne programmører. Også her er der tale om en indbygget version, der starter sammen med datamaten. NewBrain bliver forøvrigt allerede markedsført her i landet af firmaet Semicap, som også importerer den noget større, meget professionelle GEMINI GALAXY datamat. Til denne sidste fås Metanic COMAL og et helt nyt dansk udviklet system med en over ordentlig interessant Pascal compiler (se andetsteds i bladet om COMPAS systemet). På Trinity College i Dublin gør man gode fremskridt med implementering af en version, der i første omgang skal anvendes på Apple, men i øvrigt sigter på de nye 16/32 bit mikrodatamater.

Hidtil har man haft COMAL fortolkere til systemer, der er baseret på de tre mest populære mikroprocessorer, 8080 (fx. DDE), Z80 (fx. Metanic, RC og Grundy) og 6502 (Commodore). Samme dag, som dette skrives, har jeg imidlertid modtaget det første COMAL program, som er skrevet til og udført på en datamat, der er bygget over MOTOROLA 6809 mikroprocessoren. Dette program er også af andre grunde interessant, idet det indeholder en funktion, der begynder med linjen

```
FUNC TABULATOR$(STED)
```

Der er altså tale om en funktion af tekst-type! Hvis ikke der er foregået helt uventede udviklinger andetsteds må denne nye 6809-COMAL være den første som tillader brugen af tekst-funktioner. Fortolkeren er skrevet af Mogens Kjær - K'et i JKL - som også skrev den første fortolker til 6502 (Commodore). Mikroprocessoren 6809 anvendes i mange af de allernyeste datamater fra både japanske, amerikanske og engelske producenter. Forhandlinger med flere firmaer er allerede i gang, men som bekendt bør man ikke snakke for meget om "sager, der er til forhandling". Det skal dog ikke skjules, at denne nye COMAL ser ud til at blive et meget spændende bekendtskab.

Det ser ud til, at det også blev JKL gruppen, som kom først med en COMAL version, der kan anvendes i forbindelse med højopløsnings-grafik. I forbindelse med COMAL-kortet til CBM datamaterne kan man nu få et grafik-kort, som gør det muligt at frembringe billeder på skærmen i en opløsning svarende til 512x256 punkter (pixels). Kortet indeholder både programmel til styring af grafikken og tilstrækkeligt lager til at kunne indeholde informationer om to hele skærmbilleder ("dobbelt skærmhukommelse"). Det er også bemærkelsesværdigt, at den del af programmellet, der indeholder de grafiske funktioner og procedurer, kan anvendes uden at man bruger så meget som en byte af arbejdslageret - der for øvrigt er på over 30 kB! Programmellet tillader brug af såvel relativ vektorgrafik - "turtle graphic" - som sædvanlig koordinatgrafik. Man kan anvende de fra bl. a. UCSD Pascal kendte procedurer: MOVETO(X,Y), TURNTO(V), MOVE(X), TURN(V), PENCOLOR(F), osv. Ved implementeringen har man anvendt den tidligere omtalte teknik, hvorved man kan kalde underprogrammer, skrevet i maskinkode, som om der var tale om sædvanlige COMAL-procedurer. Denne teknik bliver også anvendt i et helt nyt projekt, hvor man styrer en værktøjsmaskine - in casu en fræser - med COMAL programmer. Delprocesser, som skal kunne af-

vikles meget hurtigt og reagere omgående på impulser fra værktøjet, bliver programmet i maskinkode og gemt i PROM kredse i den tekniske snitflade til værktøjsmaskinen. Projektet har vakt stor interesse på bl. a. Teknologisk Institut.

De mange initiativer omkring COMAL havde gjort det ønskeligt at tage definitionen op til en revision, og den 7.-9. maj i år lykkedes det at få et møde i stand mellem repræsentanter for Metanic, Regnecentralen, JKL gruppen, Trinity College (Irland) og COMAL Users Group (USA). Mødet blev afholdt på Tønder Statsseminarium. På dette møde enedes man om en række justeringer i COMAL definitionen. Efter min vurdering repræsenterer disse justeringer vigtige forbedringer i forhold til den tidligere COMAL NUCLEUS. Den vigtigste ændring er, at procedurer og funktioner nu er to adskilte begreber i COMAL, således som det i øvrigt oprindeligt var tilfældet, og som det også var beskrevet i den første definition af COMAL-80, jeg offentliggjorde. Procedurebegrebet er ikke ændret, men funktioner skal fremtidigt defineres således:

```
FUNC <funktionsnavn><evt. parameter-
  liste>

FUNC <funktionsnavn><evt. parameter-
  liste><evt. CLOSED>
  <funktions-krop>
ENDFUNC <funktionsnavn>
```

Efter den nye definition er <funktionsnavn> det samme som navnet på en variabel, hvilket bl.a. betyder, at en funktion også skal kunne være af tekst-type, man kan altså have "$-funktioner". Man kan også sige det sådan, at man kan have funktioner af alle de i sproget tilladte datatyper. Funktionsværdien returneres ikke mere i funktionens navn, men i en RETURN-sætning. Som eksempel kan vi tage

```
FUNC SFD(X,Y)
   IF X MOD Y=0 THEN
     RETURN Y
   ELSE
     RETURN SFD(Y,X MOD Y)
   ENDIF
ENDFUNC
```

Bemærk, at der er tale om en funktion, som kalder sig selv rekursivt. Der er flere fordele ved at anvende denne definition, men en af de mest iøjnefaldende er, at en tekst-funktion kan erklæres uden at man behøver have en "OF-del" i hovedet, idet funktionsværdien ikke mere skal tildeles funktionens navn. Som eksempel kan vi bruge:

```
FUNC BAGLÆNS$(TEKST$) CLOSED
   L:=LEN(TEKST$)
   DIM INTERIM$ OF L
   FOR I:=L TO 1 STEP -1
      INTERIM$:=INTERIM$+TEKST$(I)
   NEXT I
   RETURN INTERIM$
ENDFUNC BAGLÆNS$
```

Endvidere er det fra nu af sådan, at deltekster udpeges efter forskriften

```
<navn på variabel>(<første tegn>:
   <sidste tegn>)
```

Hvis således NAVN$ har værdien "EMIL P.", så vil udtrykket

```
NAVN$(6:7)
```

have værdien "P.". Hvis en deltekst skal udtages af en komponent i en teksttabel, skal komponent og deltekst udpeges hver for sig. Således betyder

```
ELEV$(22) (11:20)
```

den del af teksten i komponent nr. 22, som indbefatter tegnene fra og med nr. 11 til og med nr. 20.

Syntax beskrivelsen af den nye COMAL KERNAL kan fås ved henvendelse til Dataafdelingen, Tønder Statsseminarium, Østergade 65, 6270 Tønder. Alle forespørgsler skal indeholde frankeret svarkuvert (4,00 kr.), da riget og dermed dets institutioner fattes penge.

Ved revisionen blev der taget vidtgående hensyn til den ældste og mest udbredte COMAL-80, nemlig Metanics version. Til gengæld opnåedes der enighed på alle væsentlige punkter, således at COMAL programmer, der overholder specifikationer i COMAL KERNAL, i praktisk taget uændret form kan overføres fra et system til et andet. Hvorvidt en sådan samstemmighed kan bevares i fremtiden, efterhånden som der kan blive tale om udenlandske implementeringer, er naturligvis svært at udtale sig om. Men indtil videre er Trinity College i hvert fald indforstået, og jeg er i nær kontakt med Grundy Business Systems, som synes meget interesseret i at følge den nye definition. Derimod må man regne med, at de forskellige COMAL versioner vil kunne være udstyret med forskellige udvidelser. På dette punkt må sproget nok finde sig i at dele skæbne med sit store forbillede Pascal.

Børge R. Christensen

(I næste nummer: Hvordan Arkimedes fik styr på skildpadden med COMAL).