2
3. ÅRGANG

FEBRUAR 1979

data lære

INDHOLD

Udgivet af
FORENINGEN FOR DATALÆRE OG ANVENDELSE AF EDB I UNDERVISNINGEN

ITT 3343

markedets bedste dataterminal, nu også med dansk karaktersæt

Den ideelle dataterminal til forsknings- og undervisningsmiljøer. Den utroligt driftsikre og stabile opbygning sikrer lang levetid med minimal vedligeholdelse.

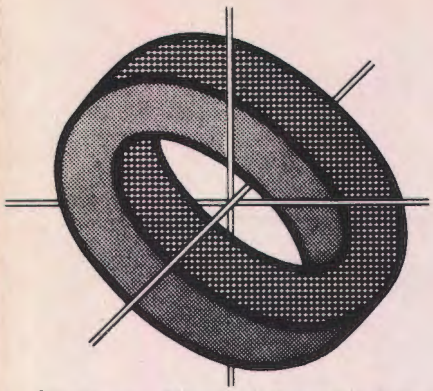
• 132 karakterers skrivebredde • almindeligt papir • Buffer sikrer maksimal printhastighed • Margin kan sættes såvel fra linien som fra tastaturet • 7 x 9 matrix • Akustisk kobler til transmission over telefonnettet.

Nu: incl. danske tegn og akustisk kobler kr. 15.300,-
undervisningsrabat 1.530,-

kr. 13.770,- excl. moms.

ITT data

Standard Electric Kirk A/S, Fabriksparken 31, 2600 Glostrup. Telefon 02 45 00 45
Standard Electric Kirk A/S, Ane Stauningsvej 21, 87 00 Horsens. Telefon 05 62 33 11



Der var mødt ca. 35 medlemmer til generalforsamlingen og de dermed arrangerede faglige møder i Aalborg den 24. - 25. november 1978.

Valg af dirigent.

W. Kjellberg valgt.

Formandens beretning.

Erling Schmidt (ES) udtrykte glæde over det ret store fremmøde trods det noget afsides sted, man befandt sig.
Det havde været et ret stille år, hvor der ligesom var sket en modning af foreningen og dens aktiviteter.
ES udtrykte en vis form for skuffelse over, at der kun havde været én henvendelse fra medlemsskaren i årets løb.
Der havde ikke været direkte formaliserede aktiviteter, men vi havde dog deltaget i den store kongres i Odense for nordiske matematik-, fysik- og kemilærere i gymnasiet.
Det skyldes måske nok, at vor medlemsskare med undtagelse af København er meget spredt, og man skulle måske overveje at give arrangementerne en anden struktur, som f. eks. et større samlet arrangement i en week-end for derved at kunne få større lødighed.
Hvad angør datalæres placering er der endnu ingen lysning for folkeskolens vedkommende, og gymnasiet venter stadig på sin reform. På handelsskolerne er faget derimod et kraftigere og kraftigere obligatorisk fag.
Anvendelsen af EDB i undervisningen ser derimod ud til at vinde støt og roligt frem.
Formanden sluttede med håbet om, at flere medlemmer ville tage mere aktivt del i foreningens arbejde.

Debat

Peter B. Yde (PY) spurgte, om der ikke snart kom en programmelcentral.
Kjellberg (KJ) omtalte de programmer, der er udgivet fra RUC på gymnasieniveau. De var tænkt som et idéoplæg, men fremtidige idéer bør komme fra lærerne. Hvis der kunne skaffes penge, var der mulighed for at lave en form for programmelcentral for gymnasierne på RUC.
PY roste foreningen for dens bestræben for at få datalære ind som valgfag. Måske bør foreningen nu også arbejde for reduktion ved pasning af anlæg og udarbejdelse af programmer til anvendelse i undervisningen.
KJ.: Det tager de faglige foreninger sig af. Der pågår pt en undersøgelse af tidsforbruget ved udplukkede anlæg. Resultaterne heraf vil derefter indgå i overenskomstforhandlingerne.
ES: Vi har peget på problemet overfor de respektive faglige foreninger.
PY ønskede ikke honorar men reduktion.
Claes Holmgren: Vil gerne påtage sig arbejdet med programcentralen, dvs. katalogisere og sende ud.
PY ønskede efteruddannelseskurser for EDB-lærere i andre fag for af få indsigt i disse, og derved være i stand til at udarbejde programmer til disse.
KJ: Der vil komme kursus af den slags i efteråret 1979.
Grethe Illum: Det er de andre lærere, der bør lære at udtrykke sig klart om, hvad de vil have lavet.
Torben Høirup og Slemming nævnte, at for folkeskolens vedkommende kan enhver henvende sig til DLH og foreslå kurser oprettet.

Derefter blev beretningen enstemmigt godkendt.

Regnskab

Kassereren forelagde regnskabet med bemærkninger om, at porto er blevet en belastende post, at 137 ud af 431 medlemmer havde fået ændret i stamoplysningerne i årets løb, og at mange glemte at meddele disse ændringer i rette tid.
Regnskabet enstemmigt godkendt.

Fastsættelse af kontingent

Uændret 90 kr.

Valg

Formandsvalg:
Erling Schmidt genvalgt.
Styrelsesmedlemmer: W. Kjellberg, Torben Høirup, Frits G. Knudsen blev valgt.
Suppleant: K. Slemming valgt.
Revisor: Karl Johan Jørgensen genvalgt.
Revisorsuppleant: Jens Damgård valgt.

Eventuelt

Der blev foreslået nedsat en lille gruppe, der skulle udforme en skrivelse om at få datalære ind i vort uddannelsessystem. Det blev vedtaget på den måde, at bestyrelsen supplerer sig med 2-3 medlemmer.

Om kæder af hele positive tal

Efterhånden som adgangen til datamaskiner bliver mere og mere udbredt på alle uddannelsestrin, føler mange undervisere behov for at kunne fremvise "virkelige" problemer for deres elever, dvs. opgaver, som ikke specielt er konstrueret til datamaten, og som endnu ikke er løst fuldtud. Samtidig må problemet forholdsvis let kunne formuleres og vække interesse hos eleverne. I det følgende skal jeg gøre rede for et sådant problem, som mig bekendt kun er undersøgt ganske overfladisk.

Betragt et vilkårligt naturligt (dvs. helt, positivt) tal A. Ud fra dette tal dannes rekursivt en følge af naturlige tal, som vi vil kalde en talkæde, på følgende måde: Hvis A er lige, er næste element i talkæden 1/2 A. Hvis A er ulige er næste element 3A +1. Formelt kan vi altså skrive definitionen således:

a_o : vilkårlig

$$a_n = \begin{cases} 1/2\, a_{n-1} & \text{hvis } a_{n-1} \text{ er lige} \\ 3 \cdot a_{n-1} + 1 & \text{hvis } a_{n-1} \text{ er ulige} \end{cases}$$

Vælger vi fx. startværdien a_o = 24, får vi talkæden:

24 12 6 3 10 5 16 8 4 2 1.

Her standser vi, idet de følgende led nu blot bliver: 1 4 2 1 4 2...

Som endnu et eksempel betragter vi a_o = 25, og vi får talkæden:

25 76 38 19 58 29 88 44 22 11 34 17
52 26 13 40 20 10 5 16 8 4 2 1

Vi kan allerede af disse to eksempler se, at forskellige startværdier giver vidt forskellige talkæder, som også har forskellig længde (dvs. antal elementer). Tilsyneladende er der ikke noget gennemgående system i antallet af elementer i en sådan talkæde.

Det er ikke særlig vanskeligt at skrive et simpelt BASIC program, der giver talkæden for en vilkårlig startværdi. For at spare maskintid, er det kun nødvendigt at begynde med startværdier, der er ulige, idet lige startværdier blot i de følgende led i kæden vil blive halveret en eller flere gange, indtil et ulige tal fremkommer.

Undersøges tallene mellem 2 og 100 som startværdier, får vi talkæder, hvis længder er angivet i nedenstående tabel.

Af startværdierne under 100 er det altså 97, der giver anledning til den længste talkæde, nemlig 118 led inden tallet 1 nås. Denne lange kæde stiger lidt efter lidt op til sin største værdi på 9232 i led nr. 85 og den næsthøjeste værdi på 7288 nås i led nr. 75.

Det er naturligvis nødvendigt at have en datamaskine til rådighed, hvis man vil udforske tal-

a_o	ldg.	a_o	ldg.	a_o	ldg.	a_o	ldg.	a_o	ldg.	a_o	ldg.	a_o	ldg.
2	1	16	4	30	18	44	16	58	19	72	22	87	30
3	7	17	12	31	106	45	16	59	32	73	115	88	17
4	2	18	20	32	5	46	16	60	19	74	22	89	30
5	5	19	20	33	26	47	104	61	19	75	14	90	17
6	8	20	7	34	13	48	11	62	107	76	22	91	92
7	16	21	7	35	13	49	24	63	107	77	22	92	17
8	3	22	15	36	21	50	24	64	6	78	35	93	17
9	19	23	15	37	21	51	24	65	27	79	35	94	105
10	6	24	10	38	21	52	11	66	27	80	9	95	105
11	14	25	23	39	34	53	11	67	27	81	22	96	12
12	9	26	10	40	8	54	112	68	14	82	110	97	118
13	9	27	111	41	109	55	102	69	14	83	110	98	25
14	17	28	18	42	8	56	19	70	14	84	9	99	25
15	17	29	18	43	29	57	32	71	102	85	9	100	25
										86	30		

kæderne i større målestok og for store begyndelsesværdier. Længderne kan være, men er ikke nødvendigvis lange, hvis startværdien er stor. Fx. har vi følgende længder for et par "store" startværdier:

a_o	længde	a_o	længde	a_o	længde
999	50	10 002	66	10 005	30
1 000	112	10 003	66	1 000 007	259
10 001	180	10 004	30	1 000 008	114

Man kan formulere mange spørgsmål om disse talkæder. Fx. kender jeg ikke noget matematisk bevis for, at alle talkæder nødvendigvis må føre til tallet 1, men foreløbig tyder alle empiriske data på, at det er tilfældet. Kan man overhovedet tænke sig uendelig lange kæder, hvis led bevæger sig frem og tilbage i talrækken uden nogensinde at nå tallet 1? Findes der talkæder, der er periodiske?

Det er desuden påfaldende, at alle talkæder tilsyneladende slutter med elementerne 16, 8, 4, 2, 1. Vil det nødvendigvis altid være sådan?

Ved optælling i forskellige talkæder ser det ud til, at ca. en trediedel af tallene er ulige, mens resten er lige. Men dette er måske, hvad man kunne vente, eftersom et ulige tal altid efterfølges af et lige tal, mens et lige tal ikke altid efterfølges af et ulige men af og til af flere lige?

Den skitserede måde at danne talkæder på, er selvfølgelig blot et specialtilfælde af en langt mere generel rekursiv dannelse af talfølger af hele tal, men det er som om der indgår et element at tilfældighed i den ellers så veldefinerede talfølge således, at det er umuligt at forudsige dens opførsel udfra en given startværdi.

Måske er undersøgelsen af talkæder som de ovenfor nævnte et frugtbart projekt at gå i gang med for energiske 'dataryttere'. Dels foreligger der temmelig få facts om dem på forhånd, så det er 'jomfrueligt' land, hvor man må kunne opnå originale resultater, og dels udgør de et ikke ganske trivielt emne at tage fat på. Mon læserne kan komme med yderligere (beviste eller empiriske) fakta om disse mystiske talkæder?

Kilde: The Mathematics Teacher, Januar 1978.

Jens Carstensen
Tårnby Gymnasium

```
0010 REM: DETTE PROGRAM UDREGNER TALKÆDEN FOR
0020 REM: EN GIVEN ULIGE STARTVÆRDI
0030 PRINT "ANGIV NEDRE GRÆNSE A OG ØVRE GRÆNSE B"
0040 INPUT A,B,
0050 REM: STARTVÆRDIEN SKAL VÆRE ULIGE, TESTES I 0060
0060 IF A/2 = INT(A/2) THEN LET A = A + 1
0070 LET I = A
0080 PRINT
0090 PRINT I;
0100 LET I = 3*I + 1
0110 PRINT I;
0120 IF I/2 = INT(I/2) THEN GOTO 140
0130 GOTO 100
0140 LET I = I/2
0150 PRINT I;
0160 IF I = 1 THEN GOTO 0180
0170 GOTO 120
0180 PRINT
0190 LET A = A + 2
0200 PRINT
0210 IF A<B THEN GOTO 0070
```

Så kan man godt begynde at spare penge sammen...

I 1980, nærmere bestemt d. 6.-9. okt. i Tokyo og d. 14.-17. okt. i Melbourne, afholder IFIP den 8. verdenskongres om datamater.

Kongressen er som mange sådanne store kongresser, delt i flere spor, og et af disse hedder "Information Processing and Education", og ud fra undertitlerne skal det nok vise sig at være af interesse for flere af Datalæreforeningens medlemmer.

Nu er hverken Japan eller Australien jo lige om hjørnet, så det er desværre nok de færreste, der kan finde midler til at deltage, men fra foreningens side vil vi undersøge, om der er mulighed for at arrangere en fællesrejse, evt. i et samarbejde med de øvrige nordiske lande, således at det kan blive billigst muligt at deltage.

Mere om dette vil følge i kommende numre af Datalære, men er man særligt interesseret, kan man henvende sig til formanden.

OBS! OBS!

Stof til næste nummer af bladet skal være redaktionen i hænde senest mandag, den 23. april 1979.

Advanced Basic Computer – 80'ernes »Privamat«

ABC 80

- Fælles indsats fra 3 nordiske industriforetagender, støttet af enestående undervisningsressourcer – gennem samarbejde med læreanstalter, forlag og lærebogsforfattere – har givet ABC 80 en start som peger mod verdenssucces.
- Hurtig Basic m. 94 instruktioner. nøjagtighed op til 29 cifre samt grafik.
- Professionelt tastatur – V 24 interface – 16 K RAM – Kassetteenhed – programpakke.
- ABC 80 tilbyder et utal af interfacemuligheder, bl.a. IEEE, parallel, serial, reedrelæ m.v.
- ABC 80 kan tilsluttes et stort udvalg af periferiudstyr: 40 el. 80 pos. printere, floppydiske m.v.

DATAUDSTYR FRA SC METRIC A/S

DATAAFDELINGEN, SKODSBORGVEJ 305, 2850 NÆRUM, TLF. (02) 80 42 00

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

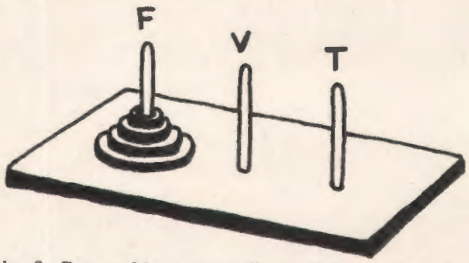


Fig. 3. Det gælder om at flytte skiverne fra pinden F (for FRA) til pinden T (for TIL). Man må kun flytte 1 skive ad gangen, og man må aldrig placere en større skive oven på en mindre. Hjælpestangen V (for VIA) må gerne bruges undervejs. Opgaven lyder altså: flyt tårnet fra F til T via V med kun 1 skive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

Dansk Data Elektronik præsenterer hermed

SPC/1 – *ET LILLE PROFESSIONELT DATASYSTEM*

– *DET FØRSTE DANSKUDVIKLEDE OG DANSK-PRODUCEREDE LAVPRIS DATAMATSYSTEM*

KOMPLET DRIFTSKLART DATAMATSYSTEM MED ALT NØDVENDIGT UDSTYR OG ID-COMAL TIL ÉN BRUGER.

19.500 KRONER

SPC/1 systemet, der er udviklet af og produceres af Dansk Data Elektronik, er et modulopbygget system bestående af professionelle komponenter, der også anvendes ved opbygningen af større og langt dyrere systemer. Dette giver driftssikkerhed og udbygningssikkerhed.

SPC/1 basiskonfigurationen består af følgende enheder:

Datamat med 32kb arbejdslager:
asynkron terminaltilslutning
asynkron printertilslutning

Baggrundslager i form af:
minidisketteenhed til lagring af 90 kb –
væsentlig hurtigere end kassettebånd

Skærmterminal med 24 linjer a 80 tegn:
små og store bogstaver
tastatur som en skrivemaskine med
god plads til fingrene.

Programmellet i SPC/1 er baseret på det af Dansk Data Elektronik udviklede multiprogrammeringssystem MIKADOS.
Fra fødslen er SPC/1 endvidere udstyret med ID-COMAL (struktureret BASIC) af hvis faciliteter følgende skal nævnes:

- *Løkkestruktur der overflødiggør brug af GO TO:*

  CASE - ENDCASE
  WHILE - ENDWHILE
  REPEAT - UNTIL

  dvs. mulighed for struktureret programmering

- *Kald af procedurer og funktioner ved navn med parameteroverførsel:*

  PROC - ENDPROC
  EXEC

- *Hurtig tilgang til filer på baggrundslageret for permanent opbevaring af såvel programmer som data. Data kan struktureres i både sekventielle og direkte filer og tilgangen sker ved anvendelsen af:*

  OPEN - GET - PUT - CLOSE

- *EDIT kommando til brug ved rettelser i en eksisterende programlinje – uden at denne skal indtastes påny! Ved syntaksfejl placeres cursor på det sted i programlinjen, hvor syntaksfejlen er opdaget. Dette er godt pædagogisk værktøj.*

- *Variabelnavne med op til 16 tegn*

- *Styring af skærmens cursor*

- *IN/OUT – sætninger til styring af ydre enheder som A/D og D/A omsættere*

- *Kald af programmer skrevet i assembler.*

SPC/1 sammen med ID-COMAL er således et datamatsystem i professionel udførelse, der kan anvendes til en lang række opgaver. Og prisen for SPC/1, der blot skal tilsluttes 220 volt for at kunne udføre Deres opgaver, er:

19.500 KRONER

Sammenlign denne pris med prisen på en terminal eller med prisen for vedligeholdelsen af en minidatamat!

SPC/1, der er modulært opbygget, kan udbygges betydeligt. Udbygningen kan ske, når De ønsker det, med:

- *større arbejdslager*
- *større baggrundslager (op til 80 Mb)*
- *flere terminaltilslutninger og terminaler*
- *andre programmeringssprog*

Det er således muligt at udvide SPC/1 så flere brugere (op til 8) samtidig kan afvikle programmer, hvorved ydre enheder som baggrundslager og printer kan deles mellem brugerne.
Skift mellem brugerne (under afvikling af flere programmer samtidigt) går meget hurtigt, da alle brugernes arbejdsområder konstant er placeret i arbejdslageret. Brugerområderne skal således ikke swappes, som det ofte er tilfældet i minidatamater. Dvs. maskinens tid udnyttes til afvikling af brugerens programmer og ikke til administration af maskinen selv!

Med hensyn til programmel kan SPC/1 udvides til foruden ID-COMAL at afvikle:

- *Editor (editering af programmer eller tekstbehandling)*
- *Assembler*
- *Debugger*
- *Pascal (fuld standard Pascal)*

Med henblik på tilkobling til større centraldatamat eller datamatnetværk kan SPC/1 udbygges med en synkron port eller HDLC port, således at SPC/1 kan udgøre en del af et distribueret system.

SPC/1 KONFIGURATIONER:

SPC/1 basiskonfiguration til afvikling af ID-COMAL for
én bruger
32 kb arbejdslager
Minidiskette til lagring af 90 kb
Skærmterminal — *19.500 kroner*

Udvidet én-bruger konfiguration af SPC/1 til afvikling af
ID-COMAL
Pascal
Editor/Assembler/Debugger

Systemet består af
Datamat med 64 kb arbejdslager
Dobbelt minidiskettedrev til lagring af 2 x 90 kb
Skærmterminal — *33.850 kroner*

Fire bruger konfiguration af SPC/1 til afvikling af ID-COMAL
96 kb arbejdslager – 16 kb arbejdslager pr. bruger.
Minidiskette incl. DMA kanal til lagring af 180 kb.
4 skærmterminaler. — *63.450 kroner*

Otte bruger konfiguration af SPC/1 til afvikling af ID-COMAL
160 kb arbejdslager – 16 kb arbejdslager pr. bruger.
Minidiskette incl. DMA kanal til lagring af 180 kb.
8 skærmterminaler — *110.100 kroner*

eller samme konfiguration blot med yderligere 20 Mb pladelager (5 Mb udskiftelig) — *199.100 kroner*

Priser på forskellige udvidelser:

32 kb arbejdslager	*6.450 kroner*
Printer fra	*8.000 kroner*
Strimmellæser	*7.500 kroner*
Strimmelhuller	*11.000 kroner*
Disketter (0,25 Mb)	*19.900 kroner*
Pladelager (20 Mb)	*89.000 kroner*

Alle opgivne priser er excl. moms!

dansk data elektronik aps
Herlev Hovedgade 207
2730 Herlev 02 84 5011

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

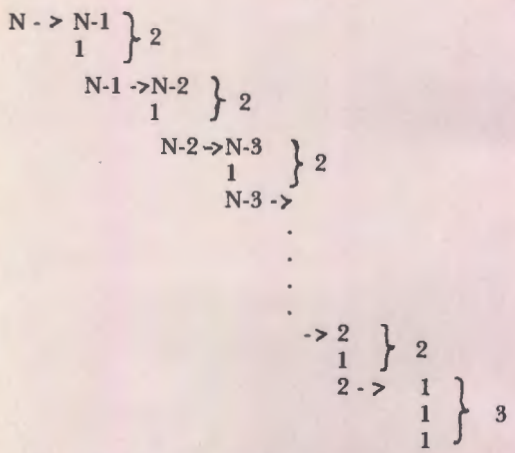


Fig. 5. Stakkens udvikling. Kun første parameter (antal skiver der skal flyttes) er vist. Tegnet - > betyder "afstakkes og erstattes af". Man ser altså f. eks. at det første N afstakkes og erstattes af de tre tal N-1, 1 *og* N-1, *hvoraf det sidste igen afstakkes, osv. indtil alle tal er reduceret til* 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 tre1-taller til sidst. Staklængden bliver altså:

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

```
0010 DIM T$(21)
0020 INPUT "ANTAL SKIVER= ",N0
0030 DIM A(2*N0-1,4)
0040 PRINT "TARNET I HANOI MED";N0;"SKIVER"
0050 PRINT
0060 LET S=0; F0=1; V0=2; T0=3
0070 LET T$="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

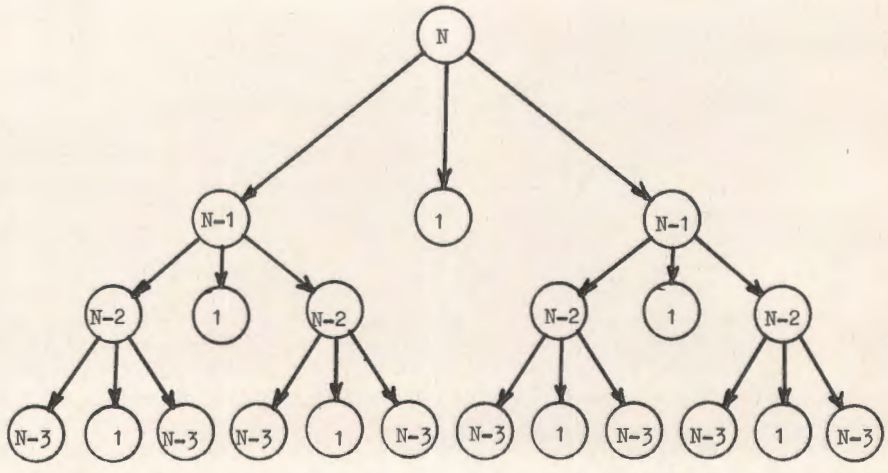


Fig. 8. Hvert knudepunkt (cirkel) symboliseret en værdi der stakkes. Når den afstakkes igen giver den anledning til de nye stakninger, som knudepunktet peger på. Kun kvistene i det fremkomne træ (cirkler med 1-taller) giver ikke anledning til nye stakninger. På figuren er kun vist fire niveauer af ialt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

Med skrivemaskinen som tegner

af Frank Jensen

Det er en kendt sag, at mange forhold anskueliggøres bedre med en tegning end med tal. Således vil en kurve som regel anskueliggøre et forløb bedre end en tabel. Jeg vil derfor gerne slå et slag for, at man tegner mere med sin datamaskine.

Tegninger kan laves på en såkaldt plotter; men det er jo ikke alle der har adgang til en sådan, og har man endelig én indenfor rækkevidde er den ikke altid lige nem at benytte. Det kan derfor være en god ide at benytte sin skrivemaskine (eller linieskriver) som plotter. Det er naturligvis ikke muligt at tegne jævnt afrundede kurver på denne måde; men der er alligevel plads til op til 9000 tegn på en enkelt side, hvilket svarer til antallet af raster i et avisbillede på 3 cm gange 4 cm. Så hvad et avisbillede af denne størrelse kan vise - og det er jo en del - det kan vi også afbilde på en enkelt side.

I det følgende vil jeg give et par eksempler på, hvad en skrivemaskine kan tegne. Programmerne er givet i Comal, fordi det tillader bedre variabelnavne end Standard Basic.

Funktioner af én variabel

Det simpleste - og også det letteste at illustrere - er en funktion af én variabel: y = f(x), som kan afbildes som en kurve i xy-planen. Og hvorfor nøjes med en enkelt funktion; vi kan lige så godt

```
0100 DIM LINIE$(132,1),CIFFER$(9,1),F(9)
0110 FOR I=1 TO 9
0120   READ CIFFER$(I)
0130   DATA "1","2","3","4","5","6","7","8","9"
0140 NEXT I
0150 READ XMIN,XMAX,YMIN,YMAX,LINIER,ANSLAG,ANTALF
0160 LET XSKALA=(XMAX-XMIN)/LINIER
0170 LET YSKALA=(YMAX-YMIN)/ANSLAG
0180 FOR A=1 TO ANSLAG+1
0190   LET LINIE$(A)=" "
0200 NEXT A
0210 REM  UDTEGNING AF FUNKTIONERNE
0220 FOR L=0 TO LINIER
0230   LET LINIE$(1)="."; LINIE$(ANSLAG+1)="."
0240   LET X=XMIN+XSKALA*L
0250   REM  BEREGN FUNKTIONSVAERDIERNE
0260   GOSUB 1000
0270   FOR I=1 TO ANTALF
0280     LET F(I)=INT((F(I)-YMIN)/YSKALA+1.5)
0290     IF F(I)<1 THEN LET F(I)=1
0300     IF F(I)>ANSLAG+1 THEN LET F(I)=ANSLAG+1
0310     LET LINIE$(F(I))=CIFFER$(I)
0320   NEXT I
0330   FOR A=1 TO ANSLAG+1
0340     PRINT LINIE$(A);
0350   NEXT A
0360   PRINT
0370   FOR I=1 TO ANTALF
0380     LET LINIE$(F(I))=" "
0390   NEXT I
0400 NEXT L
0410 STOP
0999 REM  FUNKTIONSVAERDIERNE BEREGNES HER
1000 LET F(1)=X
1010 LET F(2)=F(1)-X↑3/6
1020 LET F(3)=F(2)+X↑5/120
1030 LET F(4)=F(3)-X↑7/5040
1040 LET F(5)=F(4)+X↑9/362880
1050 LET F(6)=SIN(X)
1060 RETURN
2000 DATA 0,6,3,-2,2
2010 DATA 63,40,6
2020 END
```

Program KURVER

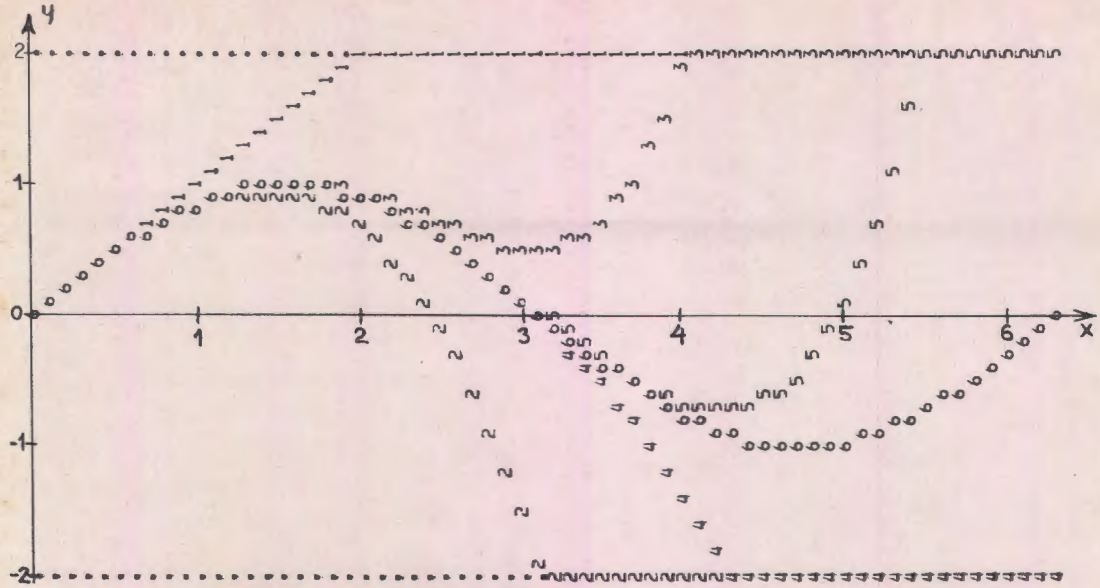


Fig. 1

give mulighed for at tegne mange funktioner på én gang: $y = f_1(x)$, $y = f_2(x)$, . . .

Det nemmeste er at vende y-aksen mod højre og x-aksen nedad på papiret, når kurverne trykkes. Hver trykt linie svarer da til en bestemt x-værdi, for hvilken vi bestemmer de tilhørende y-værdier, som så trykkes det rigtige sted på linien, idet hver funktion trykkes (tegnes) med sit symbol, f. eks. nummeret på funktionen.

Et program der kan tegne op til 9 kurver er vist i program KURVER. For hver x-værdi beregnes alle funktionsværdierne, og disse placeres i en tabel F. For let at kunne udskifte funktionsberegningerne foretages disse i et underprogram (fra linie 1000). Alle kurverne tegnes i området XMIN ≦ x ≦ XMAX og funktionsværdierne afbildes indenfor intervallet YMIN ≦ y ≦ YMAX. Funktionsværdierne der falder udenfor intervallet afbildes på kanten af intervallet. Det areal på papiret der trykkes på, fylder et antal LINIER i x-aksens retning og et antal ANSLAG i y-aksens retning. Der tegnes ANTALF funktioner samtidigt, idet ANTALF højst må være 9. Alle disse parametre indlæses i linie 150.

Parametrene LINIER og ANSLAG er defineret således, at der trykkes LINIER + 1 linier med hver ANSLAG + 1 anslag. Denne noget specielle definition skyldes, at har man f. eks. valgt XMIN=0 og XMAX=50 og gerne vil have trykt en linie for x=0, 5, 10, . . ., 50 så er det praktisk at opgive LINIER = 50/5 = 10 selv om man vitterligt ønsker at få trykt 11 linier (x=0 og x=50 incl.)

Selve programmet er ret enkelt. De tegn, der skal trykkes på en enkelt linie, placeres i tekst-tabellen LINIE$, idet funktionsværdierne F omsættes til et anslagsnummer, og der på denne plads i LINIE$ anbringes det ciffer (CIFFER$), der angiver funktionens nummer.

Den viste udgave af programmet illustrerer en rækkeudvikling af sin x:

$$\sin x = x - x^3/3! + x^5/5! - x^7/7! + x^9/9! - \ldots$$

idet $f_1(x)$, $f_2(x)$, . . . , $f_5(x)$ viser rækken når der tages henholdsvis 1, 2, . . . , 5 led med, og $f_6(x)$ viser sin x . Parametrene LINIER og ANSLAG er valgt således at der er 0.1 enhed såvel mellem to linier som mellem to anslag. Da der er anvendt en linieskriver som skriver 8 linier med 10 anslag pr. tomme, så bliver enhederne langs x- og y-aksen ikke lige lange. Resultatet er vist i figur 1 (med håndtegnede akser). Udskriften er drejet 90 grader for at figuren vender normalt; figuren er altså trykt fra venstre mod højre.

Funktion af to variable

En funktion af to variable, F = f(x,y), er vanskeligere at afbilde - det gælder for enhver slags tegneudstyr. Det mest almindelige er at tegne niveaukurver, dvs. kurver der følger punkter med samme funktionsværdier. Eksempler på niveaukurver er højdekurver i et atlas og isobarer på et vejrkort. Og her viser det sig - måske lidt overraskende - at skrivemaskinen er særdeles velegnet som tegneudstyr, og at programmet ikke bliver mere kompliceret, end det vi lige har set.

Lad os først se lidt nærmere på opgaven. Vi kan eksemplificere problemet ved at forestille os, at vi vil afbilde højden af en flade (funktionen) over xy-planen. Den del af xy-planen vi vil betragte, skal svare til det stykke af papiret, som vi

```
0100 DIM CIFFER$(10,1),TEGN$(1)
0110 FOR I=1 TO 10
0120   READ CIFFER$(I,1)
0130   DATA "+","1","2","3","4","5","6","7","8","9"
0140 NEXT I
0150 READ XMIN,XMAX,YMIN,YMAX,ANSLAG,LINIER
0160 LET XSKALA=(XMAX-XMIN)/ANSLAG
0170 LET YSKALA=(YMAX-YMIN)/LINIER
0180 REM  UDTEGNING AF NIVEAUKURVERNE
0190 FOR L=0 TO LINIER
0200   REM  BEREGN Y-VAERDIEN SOM SVARER TIL LINIE L
0210   LET Y=YMAX-YSKALA*L
0220   FOR A=0 TO ANSLAG
0230     REM  BEREGN X-VAERDIEN FOR ANSLAG NR. A
0240     LET X=XMIN+XSKALA*A
0250     REM  BEREGN FUNKTIONEN F=F(X,Y)
0260     GOSUB 1000
0270     LET FC=INT(F+.5)
0280     LET TEGN$=" "
0290     IF ABS(F-FC)>.3 THEN GOTO 0330
0300     LET FC=ABS(FC)
0310     LET TEGN$=CIFFER$(FC-INT(FC/10)*10+1)
0320     IF TEGN$="+" AND F<0 THEN LET TEGN$="-"
0330     PRINT TEGN$;
0340   NEXT A
0350   PRINT
0360 NEXT L
0370 STOP
0999 REM  FUNKTIONEN BEREGNES HER
1000 LET F=X*Y
1010 RETURN
2000 DATA -2,2,-2,2
2010 DATA 40,32
2020 END
```

Program NIVEAU

vil "tegne" på. På papiret er vi begrænset til at skrive på linierne, og på hver linie er der nogle faste skrivepositioner, som kan angives ved anslagets nummer talt fra liniens begyndelse. I modsætning til forrige eksempel kan vi nu lige så godt lade y-aksen pege opad og x-aksen mod højre. Hver linie vil altså svare til en fast y-værdi i xy-planen, og hvert anslag vil svare til en fast x-værdi.

Når man er ved en bestemt skriveposition kan man altså bestemme x og y, hvorefter funktionsværdien, f(x,y), kan findes. Funktionsværdien skal nu bruges til at bestemme hvad der skal skrives i den aktuelle skriveposition. Og der kan altså kun skrives ét tegn. Hvilket tegn det skal være, kan der filosoferes en del over. Det er hyppigt anvendt at vælge tegn med lille gråtone (som tegnene . og ,) til at angive små højder, og mørkere tegn (som W og M) til at angive større højder, akkurat som lysebrun og mørkebrun anvendes på et landkort; men det giver ikke så tydeligt et billede. Bedre er det - efter min mening - at skrive højden ud som en en-cifret talværdi. Som regel ved man så meget om sin funktion, at man kan gange (eller dividere) den med en passende faktor af 10, således at den stort set overalt er numerisk mindre end 10. Funktionsværdien kan så afrundes til et heltal, som vil være en-cifret og kan trykkes direkte på skrivepositionen. Erfaringen viser, at funktionsværdien godt må overstige 10, men så skal man blot kun trykke det højre ciffer (eneren) af heltalsværdien.

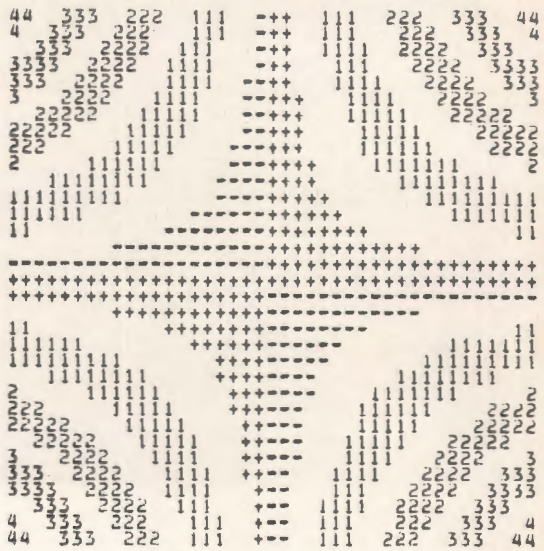


Fig. 2

Der er også et specielt problem, hvis funktionen skifter fortegn, idet der jo ikke er plads til også at trykke et fortegn. Dette problem kan løses på forskellig vis, og her har jeg valgt at trykke + eller – hvor der ellers skulle have stået 0, for på denne måde at angive om funktionen er positiv eller negativ i området.

Laver man et program efter disse retningslinier, får man papiret fyldt med "bælter" med samme trykte ciffer, hvor funktionen har værdier tæt på den angivne højde. Men da der er trykt et ciffer på hver eneste skriveposition er resultatet lidt vel gråt i gråt. Dette kan let forbedres ved kun at trykke cifre midt i bælterne, og trykke mellemrum i bælternes yderkanter, hvorved bælterne giver en tydelig fornemmelse af højdekurver. Dette er gjort på den måde, at når afrundingen af f(x,y) til et heltal er mindre end 0.3 trykkes f(x,y)'s højre ciffer, og når afrundingen er større trykkes et mellemrum.

Program NIVEAU er lavet efter disse principper. Programmet er baseret på, at funktionen beregnes i et underprogram, som brugeren skriver fra linie 1000. Desuden skal størrelsen af det stykke af xy-planen, der betragtes, angives ved at angive grænserne for x og y, XMIN ≤ x ≤ XMAX, YMIN ≤ y ≤ YMAX, og det tilsvarende stykke af papiret skal angives ved ANSLAG og LINIER på samme måde som i forrige afsnit. Et eksempel på en kørsel med funktionen f(x,y) = x·y er vist i figur 2. I modsætning til figur 1 er værdierne af ANSLAG og LINIER her valgt således at enhederne langs x- og y-aksen får samme længde, nemlig 25,4 mm (= 1 tomme).

Programmerne, der er angivet her, er meget skrabede; men de kan udvides efter behov og temperament med udskrifter af f. eks. x- og y-akser med enheder, og af funktionernes største og mindste fundne værdier. Programmerne er til gengæld meget generelle og kan faktisk benyttes til at tegne alt hvad fantasien strækker til, som eksemplet i figur 3 viser.

```
0999 REM  FUNKTIONEN BEREGNES HER
1000 LET F=.5
1010 LET RAD2=X*X+Y*Y
1020 IF RAD2>64 THEN RETURN
1030 IF RAD2<46 AND (X-Y-.84>0 OR X-Y+.84<0) THEN GOTO 1060
1040 LET F=-1
1050 RETURN
1060 IF Y<-1.5 OR Y>5.5 THEN RETURN
1070 IF Y>=.5 THEN GOTO 1120
1080 IF X>5 OR X-Y-1.1<0 AND X-Y+1.1>0 OR X<=5 THEN RETURN
1090 IF X<-3.6 THEN LET F=1
1100 IF X>-3.3 THEN LET F=5
1110 RETURN
1120 IF Y<0 THEN RETURN
1130 LET XX=3.7*Y/5.5-3.7+.5*SIN(1.9*Y)
1140 IF X<XX AND X>XX-1.1 THEN LET F=8
1150 RETURN
2000 DATA -7.9,7.9,-7.9,7.9
2010 DATA 16,6
2020 END
```

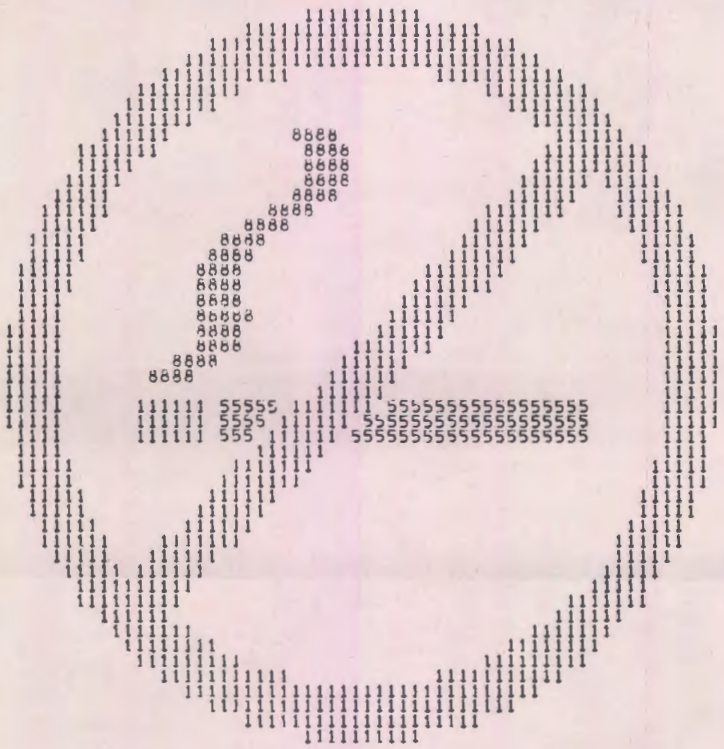


Fig. 3

PÅ VEJ TIL DEM?

Systemet består af:
1 Alpha LSI computer med 24 K ord lager
3 BEEHIVE dataskærme
1 MANNESMANN matrix printer
1 TRUE DATA stregmarkeringskortlæser
1 Dual flexible disk system."Floppy disk"
1 GNT papirstrimmellæser

Kontakt venligst:
L. Graff-Nielsen, Tlf. 02/80 42 00 lok. 32

DATAUDSTYR FRA SC METRIC A/S

DATAAFDELINGEN, SKODSBORGVEJ 305, 2850 NÆRUM, TLF. (02) 80 42 00

QUICKSORT

Af Børge Christensen

I de foregående artikler har vi set på forskellige metoder til sortering. Vi har set, at man kan sortere ved at indsætte elementer i rækken, idet en del af denne samtidig rykkes til side, hver gang et element indsættes. Man kan også gå rækken igennem og efterhånden bytte om på nogle af elementerne. Når man har gjort det tilstrækkeligt ofte, er rækken sorteret.

Det er almindeligt, at man taler om sortering ved udvælgelse (selection), indsættelse (insertion) og ombytning (exchange). I 1962 opstillede den verdensberømte datalog C. A. R. Hoare en ny metode til sortering. Denne metode beror på en kombination af ombytning og opdeling af rækken i delrækker. Den af Hoare opstillede algoritme viste sig at overgå forventningerne med hensyn til effektivitet, og Hoare gav den selv navnet quicksort. Metoden regnes stadig for den hurtigste til sortering af en forelagt række, i hvert fald når denne ikke på forhånd opfylder specielle betingelser – fx. at være delvist sorteret.

I denne artikel vil jeg gennemgå Hoares algoritme og opstille et program for den. Jeg må bede læseren prøve at følge alle detaljerne i gennemgangen. Som det så ofte er tilfældet, kommer man ikke ganske gratis til de bedste ting. Quicksort er en meget fin algoritme og er forlængst blevet en klassiker i sin genre, men som man måske kunne vente det, er den ikke helt simpel i sin struktur. Lad os begynde med en lille øvelse:

Betragt følgende række af tal:

31 30 50 46 (32) 46 21 11 51

Tallet 32 i midten er indkredset, og det skal bruges på følgende måde: Begynd yderst til venstre i rækken og ryk frem i denne sålænge de tal, du møder, er mindre end tallet 32. Stands op, når du møder et tal, som er større end eller lig med 32. Sæt en streg under dette. Begynd derpå yderst til højre i rækken og ryk tilbage i denne, sålænge de tal, du møder, er større end 32. Stands op, når du møder et tal, som er mindre end eller lig med 32 og sæt en streg under dette. Skriv nu det understregede tal til venstre lige under det understregede til højre og omvendt:

31 30 $\frac{50}{11}$ 46 (32) 46 21 $\frac{11}{50}$ 51

Ryk atter frem i rækken til venstre for 32 og stands som før, hvis du møder et tal, der er større end eller lig med 32. Sæt også en streg under det. Ryk derpå videre tilbage i rækken til højre og stands, når du møder et tal, der er mindre end eller lig med 32. Sæt streg under det og byt det om med det sidst fundne tal til venstre på samme måde som før:

31 30 $\frac{50}{11}$ $\frac{46}{21}$ (32) 46 $\frac{21}{46}$ $\frac{11}{50}$ 51

Fortsæt på samme måde, indtil søgningen fra venstre møder søgningen fra højre. Skriv de tal, der ikke er blevet ombyttet, op i rækken nedenunder på deres nuværende pladser. Resultatet skulle gerne være følgende:

31 30 11 21 (32) 46 46 50 51

Tappenstreg

Som man vil se, har vi fået opdelt den oprindelige række i to delrækker, om hvilke det gælder, at *ethvert element i rækken til venstre er mindre end eller lig med ethvert element i rækken til højre.* Tallet 32 siges at være brugt som pivot. Ordet er fransk og udtales: *pi'vo,* og ifølge fremmedordbogen betyder det: *en tap, hvorom noget drejer sig.* Udtrykket er altså særdeles velvalgt.

I visse tilfælde bliver pivot'en selv flyttet under ombytningerne, og kan selvfølgelig give anledning til overvejelser over selve opdelingen. Lad os se på følgende række:

15 3 25 10 21

Som pivot bruges elementet 25, og ved at bruge fremgangsmåden, som vi beskrev ovenfor, når vi først frem til følgende situation:

15 3 25 10 21

idet det første element fra venstre, der er mindre end eller lig med pivot'en er pivot'en selv. Ved ombytning får vi følgende række:

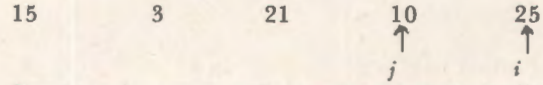
15	3	21	10	25
		↑		↑
		i		*j*

hvor de lodrette pile viser de pladser, hvor vi foreløbig er standset (*i* bevæger sig mod højre, *j* mod venstre). Vi skal nu igen rykke frem fra venstre, og vi møder ikke noget element, der er større end eller lig med pivot'en, før vi igen møder denne i følgende situation:

15	3	21	10	25
				↑
				i,j

Efter reglerne skulle vi nu rykke *j*-pilen til venstre og *i*-pilen til højre, men derved kommer de to pile til at optræde i helt forkerte roller, og det kan

bevirke, at vi foretager nogle ombytninger, som helt ødelæger det, vi indtil nu har opbygget. Vi vedtager derfor, at *søgningen skal standse, når j-pilen står til venstre for i-pilen.* Slutsituationen bliver i dette tilfælde :



Da pivot-elementet er flyttet, bruger vi de to pile til at afgrænse delrækkerne, idet vi vedtager, at den venstre delrække skal indbefatte elementerne *fra og med det, der står længst til venstre, og til og med det, som j-pilen peger på,* og at den højre delrække skal indbefatte elementerne *fra og med det, som i-pilen peger på, og til og med det, der står yderst til højre.* I eksemplet ovenfor består delrækken til højre altså kun at et element, som iøvrigt netop er pivot'en.

Øvelse.
Betragt den talrække, som er anført i den første øvelse, og udfør opdelingen af denne, idet der nu bruges en *i*-pil og en *j*-pil istedet for understregninger. Hvor står de to pile, når opdelingen er slut?

GROVSORTERING

Man kan gøre sig endnu flere overvejelser over denne opdeling, men det viser sig til syvende og sidst, at følgende algoritme altid virker efter hensigten:

Sætningen: *pivot: = element nær midten* kan realiseres ved fx. følgende:

pivot: = a((i + j) d-iv 2)

og sætningen: *ombyt a(i) og a(j)* kan fx. realiseres således:

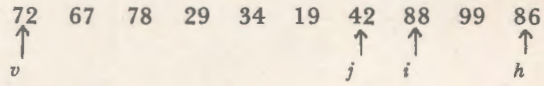
w: = a(i); a(i) : = a(j); a(j) : = w

Ved den således opstillede algoritme opnår man en art "grovsortering" med pivot'en som "maske", og i princippet er det let at gøre sorteringen helt færdig. Man fortsætter blot med at foretage den samme proces med de to delrækker, som derved opdeles i nye delrækker, der igen opdeles, og således videre, indtil der fremkommer *delrækker med kun ét element.* En række bestående af ét element må nødvendigvis være færdigsorteret! I princippet altså simpelt – det er der så meget her i verden, der er. Lad os hellere gennemgå et eksempel i detaljer.

Eksempel.
Der er forelagt følgende talrække:

72 67 78 29 86 19 88 42 99 34

Tallet 86 udvælges som pivot, og opdelingen giver følgende resultat (kontroller!):




```
a(1..n): vektor af tal
i,j: pile
pivot, w: tal
i:=1; j:=n //pile på yderelementerne//
pivot:=element nær midten
gentag
  sålænge a(i)<pivot udfør
    i:=i+1 //ryk pilen til højre//
  slut sålænge
  sålænge a(j)>pivot udfør
    j:=j-1 //ryk pilen til venstre//
  slut sålænge
  hvis i<=j så
    ombyt a(i) og a(j);
    i:=i+1; j:=j-1 //ryk pile til næste position//
  slut hvis
indtil j<i //j kommer før i//
```

Jeg har udstyret rækken med to randpile til udpegning af hhv. første og sidste element. Vi vil nu fortsætte med delrækken til venstre *(v..j)*, men inden vi gør det, må vi notere os grænserne for rækken til højre, så vi siden kan finde tilbage til den. Disse grænser er (8,10). Derpå flytter vi randpilene hen på rækken til venstre og får efter at have anvendt opdelings-algoritmen på denne:

19	29	78	67	34	72	42
↑	↑	↑				↑
v	*j*	*i*				*h*

Vi noterer os grænsepilene for delrækken til højre: (3,7) og fortsætter med delrækken til venstre med følgende resultat:

29	19
↑	↑
v,j	*i,h*

Her indtræffer en vaskeægte sensation! Såvel venstre- som højre-delrækken består af netop ét element (singleton-rækker), og vi er altså færdige med at sortere dem. Vi kan se at venstre-delrækken er en singleton derved, at *v*-pilen og *j*-pilen peger på det samme element, og for højre-delrækken gælder det tilsvarende for *i*-pilen og *h*-pilen.

Vi ser efter i vore notater og konstaterer, at den sidste højre-delrække, vi efterlod, indeholder elementer med indeks fra 3 til 7. Vi flytter randpilene hen til den, opdeler og får:

34	67	78	72	42
↑	↑			↑
j,v	*i*			*h*

Vi noterer grænsepilene for delrækken til højre: (4,7), konstaterer at delrækken til venstre er færdigsorteret og tager derfor straks fat på delrækken fra nr. 4 til nr. 7 med følgende resultat:

67	42	72	78
↑	↑	↑	↑
v	*j*	*i*	*h*

Vi har hermed en ny venstre- og højre-delrække, hvor vi må notere grænserne for delrækken til højre og fortsætte med delrækken til venstre. Jeg vil imidlertid overlade det til læseren som en øvelse at gøre sorteringen færdig og istedet se lidt nærmere på de notater vedrørende højre-delrækkernes grænseindeks, vi gør undervejs i algoritmen.

PROTOKOLLER

Først noterede vi grænserne (8,10) og derpå grænserne (3,7). De sidstnævnte blev brugt senere i sorteringen, og derpå noteredes grænserne (4,7), der dog straks efter blev taget i brug ved den sidste opdeling, vi har set på. Denne giver til gengæld anledning til, at grænserne (6,7) noteres. Lad os tænke os, at vi har gjort vore notater i en protokol med nummererede linjer. Først ser protokollen således ud:

1 (8,10)
2 (3,7)

Så tages rækken (3..7) i brug, og protokollen ser sådan ud:

1 (8,10)
2 ~~(3,7)~~

Det sæt randindekser, der er blevet læst, er samtidig streget ud. Derpå får vi følgende protokol:

1 (8,10)
2 ~~(3,7)~~ (4,7)

Når rækken (4..7) er taget under behandling, ser protokollen sådan ud:

1 (8,10)
2 ~~(3,7)~~ ~~(4,7)~~

Efter at indeksparret (6,7) er føjet til, har vi:

1 (8,10)
2 ~~(3,7)~~ ~~(4,7)~~ (6,7)

Herefter sker der blot det, at rækken (6..7) sorteres færdig, uden at der kommer nye højre-delrækker til. Det næste par, der læses, er altså grænsepilene for den højre-delrække, vi lagde til side ved første opdeling, rækken (8..10), og det viser sig, at den ikke giver anledning til nye højre-rækker. Der sker således ikke andet med den sidste udgave af protokollen ovenfor, end at først (6,7) og siden (8,10) overstreges. Når der ikke er noteret flere højre-rækker i protokollen, og den sidste venstre-delrække er sorteret, er hele processen afsluttet.

STAK-BEGREBET

Under sorteringen er det altså nødvendigt, at man kan "føre bog" over de højre-delrækker, der venter på at blive sorteret. Hertil kan man med fordel benytte en særlig datastruktur, som kaldes en stak. Det er muligt, at læseren ikke tidligere har anvendt denne struktur, så jeg vil give en kort gennemgang af den. Hvis man er vant til at skrive programmer i COMAL eller Basic, kender man uden tvivl til datastrukturerne kø og tabel.
Lad os se på følgende sætning:

DATA 45,67,34,23,89

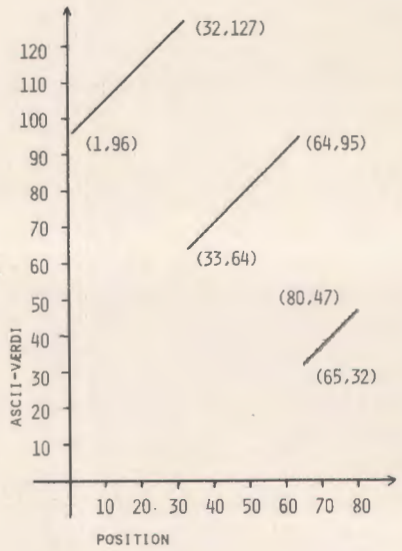
Fortsættes side 28

RC 7000 - ÅREN
RC-NYHEDSORGANET FOR RC 7000-BRUGERE

STYRING AF CURSOR

på RC 822/823/824 skærme

Det er selvfølgelig muligt at styre cursor'en på en RC skærm, der er tilsluttet en RC 7000. Og styring af udskrifterne på skærmen, og bl. a. udnyttelse af andre specielle funktioner, er med til at hæve den brugsmæssige kvalitet af systemet.
Der kan "tegnes" på skærmen med en styrbar cursor, man kan f.x. slette forkerte svar og lade skærmen stå klar til det næste svar i undervisningsprogrammer og lignende, samt lave indsætningsopgaver, f.x. cloze test.
Ja, faktisk kan man lave en hel lille "skildpadde-simulator" a la den berømte "turtle" fra USA; det er i alle fald gjort på det rejsende anlæg.
RC's dataskærme reagerer på ASCII koden 6 (for konsollen 134) ved at opfatte de næste to tegn, som modtages, som henholdsvis position på linjen og linjenummer. Afhængig af hvilke tegn der sendes, vil cursor'en placeres på den valgte plads på skærmen, og næste aktivitet, udskrift eller input, vil så foregå der.
Desværre er de nødvendige koder ikke liniære funktioner af positionerne, som det fremgår af figuren. Men ved simple midler kan man lave små procedurer, som klarer positioneringen af cursor'en.
Ud fra de eksempler, som er medtaget her, og ved en lille smule eksperimentering, skulle der ingen problemer være med at benytte cursor positionering.



EKSEMPEL PÅ ANVENDELSE AF SELVDEFINERET FUNKTION TIL CURSOR ADRESSERING:

```
0010 DEF FNC(X)=95+X-((X-1) DIV 32)*64
0020 PRINT "JEG VIL NU SÆTTE EN STJERNE PÅ SKÆRMEN."
0030 INPUT "I HVILKEN POSITION (1 - 80) :",X
0040 INPUT "OG I HVILKEN LINIE (1 - 24) :",Y
0050 PRINT "<140>"
0060 PRINT "<134>";CHR(FNC(X));CHR(FNC(Y));"*"
```

MAN KUNNE OGSÅ HAVE BRUGT DENNE FUNKTION:

```
0010 DEF FNC(X)=95+X-(X>32)*64-(X>64)*64
```

NY VERSION AF RC COMAL

Siden den første RC COMAL version 00.00 kom for efterhånden år tilbage, er der løbende indført forbedringer, og fundne fejl er blevet rettet.
Den sidst frigivne version har nu nummer 01.18, og her er der indført en mulighed for at registrere tilslutninger og afbrydelser af forbindelse til systemet.
Men dette er ikke de eneste muligheder, som man får med de nye LOGON og LOGOF funktioner.
Funktionerne er iøvrigt så overordentlig simple, at man kan tale om et slags columbus æg.
Når en terminal tilslutter sig systemet ved at skrive COMAL til ATT, leder systemet efter et program, der hedder LOGON. Hvis dette program findes, vil det blive loaded og startet. Og ved afbrydelse af forbindelsen, når man skriver BYE, vil systemet på samme måde lede efter et program med navnet LOGOF og starte det, hvis det findes.
Det er på denne måde op til brugeren selv at lægge sådanne funktioner ind i opstarts- og afbrydelsesproceduren, som man har behov for. Disse funktioner kan være alt muligt fra en slags Password-kontrol, registrering af tidsforbrug til specielle og generelle meddelelser og meget andet. Man laver nemlig simpelthen et COMAL program, der udfører det ønskede.
Ved at vælge denne måde at få et LOG-system på, får man en stor fleksibilitet; den enkelte bruger kan medtage præcis det han har behov for, og selvfølgelig udelade alt det, der ingen interesse har for hans situation.
Man kan selvfølgelig hævde, at denne metode ikke er 100% sikker, idet en dygtig "indbryder" nok skal finde et eller andet hul, men med fornuftig brug af ON ESC sætninger kan systemet faktisk gøres særdeles sikkert, hvis man ellers holder nogle få detaljer hemmelige. (Disse detaljer kan selvfølgelig så ikke beskrives her).
Som eksempler på de nye muligheder man får med den nye LOG funktion kan nævnes:

Password

Når man tilslutter sig systemet, bliver man afkrævet et password. Hvis dette er lovligt, bliver man automatisk tilsluttet til det område (subcatalog), som man har lov at arbejde i.

Registerring

Systemet kan automatisk registrere, hvem der har været tilsluttet og hvor længe.

Avis/meddelelser

Der kan bringes såvel generelle meddelelser til alle, der tilslutter sig, men der kan også bringes specielle meddelelser til bestemte personer i forbindelse med både tilslutning og afbrydelse af forbindelsen.

Kik

Man kan arrangere sig, så man kan "se", hvormange og evt. hvem der på et givet tidspunkt er tilsluttet.
Faktisk er det vel næsten kun ens fantasi, der sætter grænse for anvendelsesmulighederne.

Den sidste version af dette LOG-system og de få praktiske detaljer, som skulle holdes hemmelige, vil man kunne få gennem Regnecentralen samtidigt med overgang til rev. 01.18.

Herunder vises et eksempel på en praktisk udformning af et sign-on og et sign-off program, samt et "kik" program, der kan vise hvilke brugere, der er aktive på systemet. Programmerne er en del af et programkompleks, der er lavet ef en gymnasieelev i Aalborg. I programkomplekset indgår flere hjælpeprogrammer, som af pladshensyn ikke medtages her.

"KIK" PROGRAM:

```
0010 CLOSE
0020 DIM PW$(10)
0030 ON ERR THEN GOTO 0030
0040 OPEN FILE(1,0)"LOGUS"
0050 FOR N=1 TO 9
0060   READ FILE(1,N)PW$,S
0070   IF PW$<>"FEJL" THEN PRINT PW$,S
0080 NEXT N
0090 CLOSE
```

SIGN-OFF PROGRAM:

```
0010 CLOSE
0020 OPEN FILE(1,0)"LOGUS"
0030 LET X=SYS(4)+2-(SYS(4)=32)*33
0040 READ FILE(1,X)PW$,S
0050 WRITE FILE(1,X)"FEJL",-9
0060 CLOSE
0070 OPEN FILE(1,2)"LOGAF"
0080 IF PW$<>"FEJL" THEN WRITE FILE(1)PW$,-1,SYS(1),SYS(2),SYS(3),➔
0090 CLOSE
0100 BYE                    ➔ SYS(4),SYS(5),SYS(11),SYS(12),SYS(13)
```

SIGN-ON PROGRAM:

```
0010 ON ESC THEN BYE
0020 ON ERR THEN BYE
0030 DIM TEXT$(80),ACID$(10),SUBCA$(5),PW$(10)
0040 EXEC PASSWORD
0050 EXEC LOGIN
0060 EXEC USER
0070 EXEC AVIS
0080 CLOSE
0090 CONNECT SUBCA$,KEY
0100 NEW
0110 PROC PASSWORD
0120   INPUT "<10><10>PASSWORD: ",ACID$
0130   LET PW$=ACID$,"A"
0140   ON ERR THEN GOTO 0140
0150   OPEN FILE(1,1)"LOGPW"
0160   ON ERR THEN BYE
0170   WHILE NOT EOF(1) AND PW$<>ACID$ DO
0180     READ FILE(1)PW$,SUBCA$,KEY
0190   ENDWHILE
0200   CLOSE
0210   IF PW$<>ACID$ THEN
0220     LET ACID$="FEJL"
0230     EXEC USER
0240     BYE
0250   ENDIF
0260   CLOSE
0270 ENDPROC
0280 PROC LOGIN
0290   ON ERR THEN GOTO 0290
0300   OPEN FILE(1,2)"LOGAF"
0310   ON ERR THEN BYE
0320   WRITE FILE(1)PW$,1,SYS(1),SYS(2),SYS(3),SYS(4),SYS(5),➔
0330   CLOSE
0340 ENDPROC                    ➔SYS(11),SYS(12),SYS(13)
0350 PROC USER
0360   OPEN FILE(1,0)"LOGUS"
0370   WRITE FILE(1,SYS(4)+2-(SYS(4)=32)*33)ACID$,SYS(4)
0380   CLOSE
0390 ENDPROC
0400 PROC AVIS
0410   ON ERR THEN GOTO 0410
0420   OPEN FILE(1,1)"LOGAV"
0430   ON ERR THEN BYE
0440   PRINT
0450   PRINT
0460   READ FILE(1)TEXT$
0470   LET BREAK=1
0480   IF TEXT$="NOSTOP" THEN
0490     ON ESC THEN EXEC PROTECT
0500     LET TEXT$="DENNE MEDDELELSE KAN IKKE STANDSES!!"
0510   ELSE
0520     ON ESC THEN LET BREAK=0
0530   ENDIF
0540   WHILE NOT EOF(1) AND BREAK DO
0550     PRINT TEXT$
0560     READ FILE(1)TEXT$
0570   ENDWHILE
0580   CLOSE
0590   PROC PROTECT
0600     ON ESC THEN EXEC PROTECT
0610   ENDPROC
0620 ENDPROC
```

RC 8000 ÅREN

Måske bliver det inden længe nødvendigt, at RC 7000 ÅREN tager navneforandring til RC 7000/RC 8000 ÅRERNE.
Efterhånden er og bliver der solgt flere og flere RC 8000 datamater til undervisningssektoren, og det skal nævnes, at RC har foretaget en ret så drastisk prisnedsættelse på RC 8000, så den nu så absolut er blevet også af praktisk interesse for undervisningssektoren.
Regnecentralen er nemlig gået over til halvlederlagre på RC 8000, og dermed har prisen kunnet sættes ned.
Yderligere har man lavet en ny "lille": RC 8000 model 10, der koster væsentlig mindre, end en RC 7000 gjorde for bare få år siden...
Det kan nok forventes, at mange nye brugere starter med en lille RC 8000, men det er lige så sikkert, at mange af de gamle RC 7000 brugere vil kikke alvorligt på at få en RC 8000 model 35 til at opgradere deres anlæg med.
Prisen er nemlig ikke spor afskrækkende mere...
Ring til Thorkild Maaetoft og hør nærmere...

RC 7000 - ÅREN

UDGIVER: A/S REGNECENTRALEN
Falkoner Alle 1
2000 København F.
Tlf. (01) 10 53 66.

SUCCESFULD KONKURRENCE

I forrige nummer af RC 7000 Åren blev der udskrevet en lille programmeringskonkurrence, og sandelig om der ikke var flere gode løsninger fra læserne.
Vinderen blev Erik Nielsen, Helsingør, der klarede opgaven med et program på kun én linie. Løsningen var den samme, som oprindelig blev lavet på et kursus for gymnasielærere i Sverige.
Erik Nielsen vil modtage sin flydende præmie med posten.
Der var ikke udsat nogen yderligere præmier, men dommerkommiteen fandt dog en anden indsendt løsning så elegant, specielt syntes man godt om digtet, at man besluttede at uddele en ekstrapræmie til Grete Illum, Virum.
Der bliver kun ét gæt, hvad denne præmie kommer til at bestå af. . .

Det vindende program

```
LIST
0010 PRINT "    *<13><10>   * *<13><10>  * * *<13><10> * * * *<13><10>
* COMAL *<13><10> * * * *<13><10>  * * *<13><10>   * *<13><10>    *<13><10>"
```

„Kunstværket"

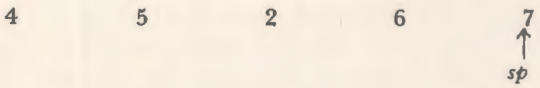
```
   LIST
   0010 DIM L$(45)
   0020 LET L$="    *       * *     * * *   * * * * * COMAL *"
   0030 FOR R=-4 TO 4
   0040   PRINT L$(37-9*ABS(R),45-9*ABS(R))
   0050 NEXT R

* RUN
    *
   * *
  * * *
 * * * *
* COMAL *
 * * * *
  * * *
   * *
    *

END
AT 0050
*
```

KUNST I COMAL GJORT MED KRYDS
PÅ EN TTY ' ER
DATA-TIDENS JULELYS
I DE DANSKE BYER.
DETTE KUNSTVÆRK SKÆNKER JEG
RC SOM FORÅRING,
VENTER AT DE SENDER MIG
RIGTIG SHERRY HEERING.

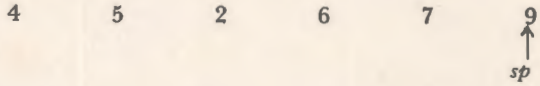
GRETE ILLUM

Fortsat fra side 23

Rækken af data, som er anført efter nøgleordet DATA, udgør netop en datakø. Det vil sige, at man kun har adgang til de enkelte elementer én for én og fra en ende af. Det datum, der er anført først, bliver også først læst. Man taler undertiden om *first in - first out (fifo)* princippet. I en tabel derimod har man adgang til ethvert element, blot man angiver dets nummer eller indeks. En sådan tabel benytter vi netop ved quicksort til at indeholde de data, der skal sorteres. En stak er en art "uretfærdig kø", hvor det element, der er kommet sidst ind, kommer først ud. Vi har altså at gøre med et *last in - first out (lifo)* princip. Lad os tænke os, at vi har en stak, der ser sådan ud:



Pilen, der peger på det hidtil sidste element, kaldes stak-pilen (the stack pointer). Man tænker sig ofte, at stakken står lodret med det sidst ankomne element øverst, og man taler da om, at det pågældende element befinder sig i stakkens top (the stack top). I en stak kan man skrive og læse. Et element, som skrives i stakken, indsættes altid i dennes top, og samtidig rykkes stakpilen frem til dette element. Lad os tænke os, at vi skriver tallet 9 i stakken ovenfor. Derefter ser den sådan ud:



Man siger også, at elementet 9 er blevet stakket (has been pushed). Når man læser i en stak, får man altid fat i det element, der befinder sig i toppen – altså det, som er ankommet sidst – og samtidig rykkes stakpilen tilbage (ned) til det foregående element, og man kan ikke mere få fat på det element, der netop er læst, medmindre man er i stand til udefra at håndtere stakpilen. Et element, der er læst, må altså betragtes som slettet, og det vil i alle tilfælde ske, hvis der skrives et nyt element i stakken. Når et element er blevet læst i en stak, siger man også, at det er blevet afstakket (has been popped). Man sammenligner ofte en datastak med en stabel tallerkener eller bakker i et cafeteria. Den tallerken eller bakke, der sidst er kommet tilbage fra opvask og lagt i stablen, er også den, der først bliver fjernet af den næste kunde.

I nogle sprog kan man arbejde med stakke direkte, idet PUSH og POP findes som nøgleord. Det gælder fx. assemblersprogene til mikroprocessorerne, der på dette punkt er mange minidatamater overlegne. I assemblerkoden for ZILOG-80 mikroprocessoren betyder sætningerne: PUSH HL og POP HL således, at der skrives hhv. læses i en stak, der på forhånd er reserveret i lageret, og som processoren iøvrigt selv holder styr på med en indbygget stakpil (SP-registreret).

SIMULERING AF STAK

I COMAL eller Basic findes ikke sproglige faciliteter til direkte oprettelse og vedligeholdelse af en stak, men ved hjælp af en tabel og lidt håndkraft kan man lave en effektiv simulering af en stak. Tabellen kan fx. oprettes og stak-pilen initialiseres ved:

```
DIM STAK (20)
LET SP = 0
```

Lad os tænke os, at vi ønsker at stakke det tal, der er tildelt den variable J som værdi. Det kan vi gøre ved tildelingerne:

```
LET SP = SP + 1; STAK (SP) = J
```

Denne sætning er altså ækvivalent med PUSH J. Ønsker vi omvendt at afstakke et tal, der skal tildeles J som værdi, kan vi benytte:

```
LET J = STAK (SP); SP = SP – 1
```

Den sætning er ækvivalent med POP J.

Nu kan vi omsider skrive en algoritme til udførelse af quicksort. Kernen i den tidligere skrevne opdelingsalgoritme – dvs. den, der begynder med det første gentag og ender med det sidste indtil – lader jeg indtil videre være repræsenteret ved sætningen:

udfør opdelingen af rækken

Vi får dermed følgende algoritme:

```
proc quicksort
  a(1..n), stak(1..20,1..2): vektorer af tal;
  sp, v, h, i, j: indices;
  pivot, w: tal
  //stak indices til første og sidste element//
  gentag
    //læs toppen af stakken//
    v:=stak(sp,1); h:=stak(sp,2); sp:=sp-1
    gentag
      i:=v; j:=h; pivot:=a((i+j) div 2)
      udfør opdeling af rækken
      hvis i<h så //hvis der er en højre-delrække//
        //så stak rand-indices for den//
        sp:=sp+1; stak(sp,1):=i; stak(sp,2):=h
      slut hvis
      h:=j //flyt h-pilen til venstre delrække//
    indtil v>=j //venstre delrækken består af ét element//
  indtil sp=0 //stakken er tom, ikke flere højre-delræk.//
endproc quicksort
```

Et program, som udfører processen, er vist på side 30.

EFFEKTIVITETEN

Som nævnt i indledningen, er quicksort i næsten alle tilfælde den hurtigste af alle sorteringsalgoritmer. Det skal dog nævnes, at det ikke kan betale sig at bruge den, hvis de rækker, der skal sorteres, er meget korte (færre end 10 - 15 elementer). Det vil sige, at de rækker, der er brugt i eksemplerne, i grunden er for korte til at retfærdiggøre quicksort, og de har kun deres berettigelse som det, de er: eksempler. Ved sortering af store datamængder kan man derimod opnå næsten drastiske forbedringer ved at bruge quicksort i stedet for fx. straight exchange (se DATALÆRE, sept. 78). Ved kørslen af SEMREG (se DATALÆRE, sept. 77) blev 2-3000 ansøgere sorteret ad gangen. Det viste sig at tage flere timer med Bubblesort, mens Quicksort kunne klare det på 20-30 min!

Ved sortering af rækker, som i forvejen er næsten sorterede, er quicksort endog meget uegnet. N. Wirth har bemærket, at fremgangsmåden da snarere må betegnes "slowsort".

På grundlag af quicksort har man udviklet andre sorteringsmetoder, som i specialtilfælde er endnu mere effektive end ophavet selv. Et ofte benyttet trick består i, at delrækker, der er kommet ned under en vis længde, bliver sorteret ved fx. straight insertion, der på den måde kommer til at virke som en slags "assistent" for quicksort.

EKSTERNE SORTERINGER

De sorteringsmetoder, vi hidtil har set på i denne artikelserie, har alle virket på datamængder, som er lagret i datamatens arbejdslager. Man kalder også sådanne metoder interne sorteringer. Hvis man skal sortere store filer på fx. diske eller disketter, kan disse i almindelighed ikke kopieres ned i arbejdslageret på én gang. Ved sortering af sådanne kan man altså ikke bruge interne metoder. Man må i stedet anvende såkaldt eksterne sorteringer, og de næste artikler vil handle om algoritmer for disse.

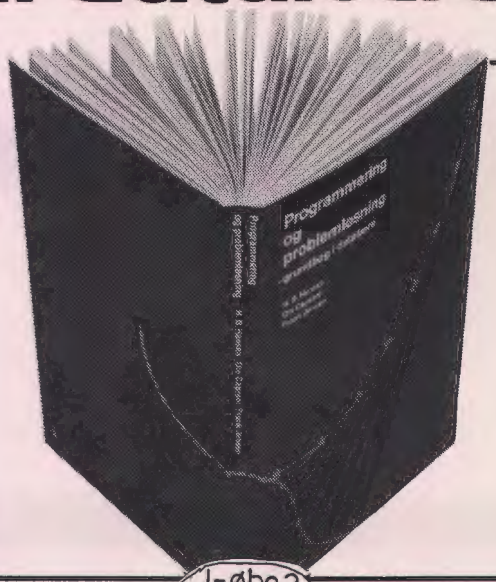
Øvelse.
Rækkerne på næste side er udskrevet af et Quicksortprogram, som angiver en række snap-shots. Pilene er rand-pile for den delrække, som står for at skulle sorteres. Rekonstruer stakken (tag hellere en lang blyant!).

```
0010 PROC QUICSORT
0020   DIM STAK(20,2)
0030   LET SP=1; STAK(SP,1)=1; STAK(SP,2)=N
0040   REPEAT ** BEGYND I TOPPEN AF STAKKEN **
0050     LET V=STAK(SP,1); H=STAK(SP,2); SP=SP-1
0060     REPEAT ** OPDEL VEKTOREN A **
0070       LET I=V; J=H; X=A((V+H) DIV 2)
0080       REPEAT
0090         WHILE A(I)<X DO
0100           LET I=I+1
0110         ENDWHILE
0120         WHILE X<A(J) DO
0130           LET J=J-1
0140         ENDWHILE
0150         IF I<=J THEN
0160           LET W=A(I); A(I)=A(J); A(J)=W
0170           LET I=I+1; J=J-1
0180         ENDIF
0190       UNTIL I>J
0200       IF I<H THEN
0210         REM ** STAK VISERE TIL HØJRE DELRAEKKE **
0220         LET SP=SP+1; STAK(SP,1)=I; STAK(SP,2)=H
0230       ENDIF
0240       LET H=J
0250     UNTIL V>=H
0260   UNTIL SP=0
0270 ENDPROC QUICSORT
0280 REM //---------------//
0290 REM ** MAIN **
0300 INPUT "HVOR MANGE TAL: ",N
0310 DIM A(N)
0320 FOR I=1 TO N
0330   LET A(I)=INT(RND(0)*90+10)
0340   PRINT A(I);
0350 NEXT I
0360 EXEC QUICSORT
0370 PRINT
0380 FOR I=1 TO N
0390   PRINT A(I);
0400 NEXT I
0410 END
```

```
77  36  83  82  57  43  31  95  60  81  42  43  75  49  13

77  36  83  82  57  43  31  95  60  81  42  43  75  49  13
↑                                                       ↑

77  36  83  82  57  43  31  13  60  81  42  43  75  49  95
↑                                                   ↑

13  31  83  82  57  43  36  77  60  81  42  43  75  49  95
↑   ↑

13  31  83  82  57  43  36  77  60  81  42  43  75  49  95
        ↑                                               ↑

13  31  49  75  57  43  36  43  60  42  81  77  82  83  95
        ↑                       ↑

13  31  42  43  36  43  57  75  60  49  81  77  82  83  95
        ↑       ↑

13  31  42  36  43  43  57  75  60  49  81  77  82  83  95
        ↑   ↑

13  31  36  42  43  43  57  75  60  49  81  77  82  83  95
                        ↑               ↑

13  31  36  42  43  43  57  49  60  75  81  77  82  83  95
                        ↑       ↑

13  31  36  42  43  43  49  57  60  75  81  77  82  83  95
                    ↑   ↑                       ↑

13  31  36  42  43  43  49  57  60  75  81  77  82  83  95
                                    ↑               ↑

13  31  36  42  43  43  49  57  60  75  77  81  82  83  95
                                        ↑           ↑

13  31  36  42  43  43  49  57  60  75  77  81  82  83  95
```

Nye bøger til datalære

Programmering og problemløsning



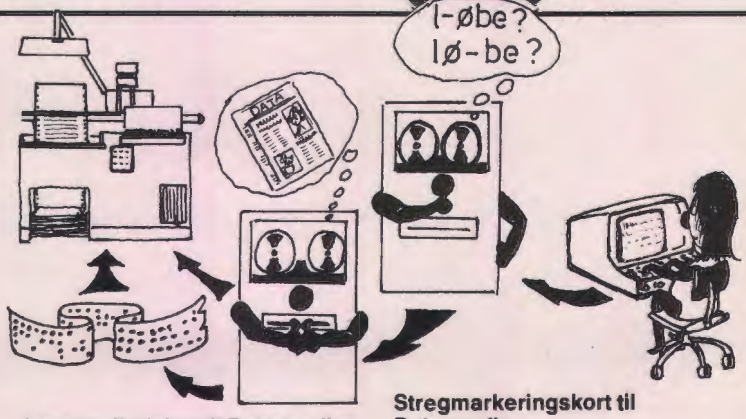
- en grundbog i datalære.
Af H. B. Hansen, Ole Caprani og Frank Jensen.
224 sider, illustreret. Kr. 75,40.
Bogen dækker det stof, der er angivet for den grundlæggende del i Betænkning om edb-undervisning i det offentlige uddannelsessystem (Johnsen-betænkningen).
»... vil kunne læses også af forudsætningsløse, der gerne vil have boglig assistance til et selvstudium ... For elever med edb-adgang vil det være nemt at efterligne den fyldige eksempel-samling ...«
- Politiken

Data og dig

Af Flemming Lund og
Ib Mogens Johansen.
2. udgave 1978.
136 sider, illustreret. Kr. 50,15.

Data og dig er et nyt materiale til undervisning i datalære i folkeskolens 8.-10. skoleår. Materialet giver dels en indføring i, hvordan datamaskinen arbejder, dels en orientering om datamaskinens anvendelse i samfundet.

Til elevbogen knytter sig:



Lærervejledning til Data og dig
52 sider. Kr. 31,25.

Stregmarkeringskort til Data og dig
Pakke med 2000 kort, kr. 156,85.

Udvalgte elementer fra datamaskinens fysik

Af Per Holm.
I serien Fysik- og Kemilærerforeningens skrifter.
64 sider, illustreret. Kr. 59,00.

Datamaskinens opbygning gennemgås, herunder muligheder og funktioner som f. eks.: Simulering af udsagn, Addition af binære tal, Oversættere, Elektronisk teknik, Registre og Multiplikationer af binære tal. Emnerne er belyst med diagrammer, eksempler og opgaver. Beregnet til datalæreundervisning i gymnasiet og på HF.

GYLDENDAL

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.

P. N. TRYK