C B N

ORGAN FOR COMAL BRUGER GRUPPE DANMARK



1983, nr.4

0108 - 4925

# Transportabilitet

Det fagsprog, der anvendes ved omtale af computerudstyr og af arbejde med computere, er spækket med fremmedord - meget ofte direkte hentet fra engelsk og ofte med oprindelse i latin.

Transportabilitet er et af de sjældnere. Det kan snarest oversættes ved ordet overflytningsegenskab, og det dækker over det begreb, at et computerprogram skrevet på en maskine kan overføres til en anden.

I andre sammenhænge er transportabilitet en selvfølge. Eksempel: Et lydbånd indspillet på en kassettebåndoptager, kan umiddelbart afspilles på en vilkårlig anden (når der ses bort fra de mere sofistikerede tilfælde).

Men således er det ikke med diskettеindspilninger af computerprogrammer - end ikke programmer skrevet i samme computersprog, og her tænkes helt kontant på COMAL 80.

Det er en skam - og særlig nu, hvor mikrocomputere og endnu mindre maskiner dukker op forsynet med COMAL 80.

På denne plads har vi allerede tidligere prist det initiativ, der er taget for at standardisere sproget, og et og andet tyder på, at anstrengelserne så småt er ved at bære frugt. Men stadig er der forskellen i de maskinbestemte ordrer og i de forskellige operativsystemer.

Kan COMAL BRUGER NYT anvise nogen løsning af problemet?

Ja, vi kan da yde et bidrag:

Hvis operativsystemerne for to maskiner er forskellige, kan disketter fra den ene ikke afspilles i den anden; men man kan overføre programmer fra en maskine til en anden pr. direkte ledningsforbindelse, hvis maskinerne står ved siden af hinanden, eller pr. telefonlinie og anvendelse af modem ved hver maskine, hvis der er stor afstand mellem dem.

Men derved er ganske vist ikke alt klaret, for der skal et overspilningsprogram til.

Dog - er disse ting i orden, kan programmer overføres; men de kan ikke køre på den modtagende maskine. De skal først tilrettes: maskinbestemte ordrer skal ændres, og (de endnu eksisterende) forskelle i COMAL 80 skal også ændres til den nye maskines standard.

Det er vist forbeholdt den kommercielle del af computerbrugerne at anvende denne fremgangsmåde.

Andre med mere beskedne forhold må formodentlig gribe til "the hard way" og indtaste programmerne på deres egen maskine.

Kan vi så lette arbejdet for denne sidste kategori?

Ja, det kan vi - navnlig hvis vi er enige om at gøre det!

I dette nummer af CBN har vi en artikel om systembestemte variable, og i den foreslårs det, at man i begyndelsen af sit program deklarerer de maskinbestemte variable, man anvender undervejs.

Derved opnås, at man straks får øje på dem - og altså slipper for at skulle lede efter dem gennem hele programmet - og at det vil være overkommeligt at udskifte dem med de variable, der gælder for ens egen maskine.

Indrømmet: Fremgangsmåden løser ikke alle problemer ved manglende transportabilitet; men nogle af dem kan klares på denne måde.

Det ville være skønt, hvis indsendere af programmer til CBN, ville tage ideen op.

CBN

## Indholdsfortegnelse

Leder: Transportabilitet ..... 1

Processtyring med COMAL 80 ..... 2

Skærm-vindue med Commodore 8032 ..... 4

Array ..... 4

Maskinkode-underprogrammer ..... 6

Flash ..... 7

Systembestemte variable ..... 8

Anmeldelse ..... 10

Alternativ sikker input-rutine ..... 11

Regneprogram ..... 13

Ugedag ..... 15

## Processtyring med Comal 80

Da microcomputeren holdt sit indtog, blev den hurtigt taget i anvendelse i industrien til visse former for processtyring. I starten blev den nok mest brugt til vejesystemer, overvågning, temperaturstyringer og lignende opgaver, hvor kravene til hastighed kunne indfries af microcomputerens højniveausprog, ofte BASIC.

Erfaringerne viste de store muligheder i den videre anvendelse af microcomputere til industriele formål, og udviklingen af forskellige styresystemer så dagens lys. Microcomputeren blev brugt mere og mere til hastighedskrævende opgaver, maskinkode-underprogrammer blev blandet med BASIC, og resultatet blev et uoverskueligt og svært læseligt program.

Erfaringerne viste, at fremtidens krav måtte være et mere moderne, struktureret og modulært sprog, der kunne interruptdrives og suppleres med maskinkodeprogrammer på en fornuftig måde.

Det dansk udviklede sprog COMAL-80 havde samtidig gennemgået en udvikling, der gjorde dette sprog særdeles attraktivt til industrielle løsninger.

Ud over COMAL-80's fortræffelige strukturer mm. vil jeg her beskrive to ting ved CBM-COMAL-80, som gør dette sprog stærkt på det industrielle område.

INTERRUPT-procedurer.

Interrupt anvendes, når man ønsker, at hovedprogrammet skal afbrydes, uanset hvad det foretager sig, og omgående udføre et underprogram, for derefter at vende tilbage til hovedprogrammet og fortsætte afviklingen af dette, som om intet var hændt.

Det signal, der forårsager afbrydelsen kommer for det meste fra en ydre enhed, der kræver omgående service.

Lad os forestille os en situation, hvor dette at kunne afbryde hovedprogrammet udefra vil være en stor fordel:

Hovedprogrammet i computeren administrerer kundesystem, ordresystem, fakturering mm., og er samtidig koblet til ti vægte. Hovedprogrammet arbejder normalt og kan anvendes af brugeren på normal vis, men når en af vægtene har et resultat klar, afsender vægten et interrupt. Hovedprogrammet afbrydes et kort øjeblik for at indlæse resultatet fra den aktuelle vægt, og få millisekunder efter, arbejder hovedprogrammet videre. Det indlæste vejeresultat er nu tilgængeligt for hovedprogrammet og vægten blev serviceret uden, da den havde et resultat klart.

Der kunne også forekomme en mere kritisk situation i en styring af en maskine, hvor hovedprogrammet er i gang med en større beregning eller sortering, og der bliver trykket på et nødstop. Hvis hovedprogrammet ikke kunne afbrydes, ville nødstopsproceduren ikke blive udført, før beregningen var udført, og hovedprogrammet kom til det sted i programforløbet, hvor det testede for nødstop. Med interrupt-faciliteten vil nødstopproceduren altid blive udført omgående, uden at programmøren skal tænke på, hvornår situationen opstår.

I CBM-COMAL-80 er det muligt i hovedprogrammet at definere, hvilket af flere interrupt-underprogrammer der skal være aktive i den aktuelle del af hovedprogrammet.

Alle microprossorer kan udføre interrupt på maskinkodeniveau, men det nye er, at alle nu på simpel vis kan anvende dette princip.

```
0010 INTERRUPT nødstop
0020
0030 LOOP
0040   PRINT "Hovedprogrammet
       kører"
0050 ENDLOOP
0060
0070 PROC nødstop
0080   PRINT "Nødstop udføres"
0090 ENDPROC nødstop
```

Eksemplet består i virkeligheden af to programmer, og underprogrammet "nødstop" kaldes ikke af hovedprogrammet. Men da "nødstop" er defineret som interrupt, vil det blive udført hvis et udvendigt signal bliver aktivt.

C B N
COMAL BRUGER NYT
ISSN 0108-4925

Udgiver:
C B G
Comal Bruger Gruppe
Mindegade 42
8700 Horsens

Tlf. 05 - 62 15 67
Giro 1 75 23 75

Ansvarshavende redaktør:
S. Chr. Hansen

Layout:
Frank Hejndorf

Tryk:
tekst & tryk
Horsens

Abonnement:
Mindegade 42
8700 Horsens

## Skærm-vindue med Commodore 8032

Den form for vindue, der hentydes til her, er en metode til at afgrænse et rektangulært område af den ordinære skærm, som på denne model er 25 linier med hver 80 tegn.

På den ordinære skærm er det muligt at udføre print, input, slet skærm, scroll op og ned, mm.

Når man "sætter" et vindue, der er mindre end den ordinære skærm, kan alle disse funktioner stadig benyttes, men kun inden for det afsatte vindue.

Det er således muligt at opsætte et pænt skærmbillede og derefter udvælge et område til input/output kommunikation med brugeren.

Fordelen er selvfølgelig, at man med een ordre kan slette dette afgrænsede område. Eller programmet kan udføre print- og inputsætninger og lade skærmen rulle på normal vis, alt sammen uden at den omkringstående tekst ødelægges.

Det er muligt at definere mange forskellige vinduer og skifte mellem dem. Denne funktion er derfor en stor lettelse, når de ellers meget omfattende programmer for styring af skærmens kosmetik skal skrives og udføres.

Proceduren til at sætte skærmen med er meget enkel:

```
0010 PROC vindue(linie'1,
     kolonne'1,linie'2,kolonne'2)
0020   PRINT AT linie'1,kolonne'1:
       CHR$(15)
0030   PRINT AT linie'2,kolonne'2:
       CHR$(143)
0040 ENDPROC vindue
```

CHR$(15) sætter øverste venstre hjørne.

CHR$(143) sætter nederste højre hjørne.

CHR$(19)+CHR$(19) sletter vinduet, og man vender tilbage til normal skærmstørrelse.

Man kan nu opbygge en række procedurer med relevante navne, MENUOMRÅDE, ARBEJDSOMRÅDE, eller hvad man nu kan finde på, og disse procedurer skal blot kalde proceduren VINDUE med de ønskede parametre.

Her er et eksempel:

```
0050
0060 menu'område(0)
0070 print H - Høj
0080 print L - Lav
0090 while not key$ in hl do null
0100 menu'område(1)
0110 arbejds'område(1)
0120 input hvor høj er du: : højde
0130
0140 proc menu'område(slet)
0150   vindue(1,70,25,80)
0160   if slet then print
       chr$(147)
0170 endproc menu'område
0180
0190 proc arbejds'område(slet)
0200   vindue(10,10,20,60)
0210   if slet then print
       chr$(147)
0220 endproc arbejds'område
```

## Array

Undertiden har man brug for at lade computeren arbejde med og huske en række tal.

Man kan da "gemme" dem i et array.

Array betyder såmænd kun række, men ordet har i computersproget fået en noget udvidet funktion, som vi ikke skal gå nærmere ind på

i denne artikel, der udelukkende er beregnet på at vise den allersimpleste anvendelse af begrebet array.

Her er fem tal: 7, 5, 2, 8 og 4.

Her er de opstillet på række i et meget enkelt system:

```
Tal 1 = 7
Tal 2 = 5
Tal 3 = 2
Tal 4 = 8
Tal 5 = 4
```

De kunne være fremkommet som udskrift af et gennemløb af dette program:

```
0010 // save "array 1"
0020 DIM tal(5)
0030 FOR nr:=1 TO 5 DO
0040   tal(nr):=RND(1,10)
0050   PRINT "tal";nr;"=";tal(nr)
0060 ENDFOR nr
0070
0100 FOR n:=1 TO 5 DO PRINT tal(n)
```

Maskinen kan huske et sådant array - d. v. s. den kan huske tallene, og den kan huske rækkefølgen, de kommer i, så vi kan kalde vort array frem igen ved f. eks. at tilføje en linie:

```
0100 FOR n:=1 TO 5 DO PRINT tal(n)
```

Computerens evne til at huske tallene kan vi udnytte på mange måder.

I det følgende eksempel har vi brug for tallene 1 til 10 i en vilkårlig rækkefølge, men uden gentagelser.

Vi laver et program, der skal frembringe tallene, prøve dem og sluttelig udskrive dem.

Programmet ser sådan ud:

```
0010 // save "array 2"
0020
0030 //select output lp
0040
0050 DIM tal(10)
0060
0070 FOR nr:=1 TO 10 DO
0080   frembring
0090   prøv
0100   skriv
0110 ENDFOR nr
0120
0130 PROC frembring
0140   tal(nr):=RND(1,10)
0150 ENDPROC frembring
0160
0170 PROC prøv
0180   FOR a:=1 TO nr-1 DO
0190     // PRINT "tal(",nr,") = ",
0200     // PRINT tal(nr),"; tal(",
0210     // PRINT a,") = ",tal(a)
0220     IF tal(nr)=tal(a) THEN
0230       a:=0
0240       frembring
0250     ENDIF
0260   ENDFOR a
0270 ENDPROC prøv
0280
0290 PROC skriv
0300   // PRINT "tal(",nr,") blev";
0310   PRINT tal(nr)
0320   // PRINT
0330 ENDPROC skriv
```

Alle linier med // kan overspringes.

Proceduren FREMBRING laver et tilfældigt tal (et random-tal) mellem 1 og 10.

Proceduren PRØV undersøger, om det frembragte tal er nyt eller er et, der allerede er accepteret.

Lad os se på undersøgelsen for tal nr. 6's vedkommende. - Det er måske et 3-tal.

Proceduren undersøger nu tallene nr. 1 til 5 (fra 1 til NR -1).

Proceduren henter de først frembragte fem tal - de kaldes nu TAL(A) - et ad gangen - og sammenligner dem et for et med det aktuelle tal - TAL(NR).

Hvis et af de første fem tal er lig TAL(NR), så går programmet ikke videre - tallet bliver altså ikke udskrevet. I stedet går programmet tilbage til proceduren FREMBRING, der producerer et nyt tal; men forinden er tallet A blevet nulstillet i linie 230, så den næste sammenligning kommer til at omfatte alle tallene fra A = 1 til A = NR -1.

Men er TAL(NR) ikke lig noget af de første fem tal, så træder proceduren SKRIV i funktion, og dermed er TAL(NR) accepteret som et nyt og ikke tidligere præsenteret tal, der udskrives.

Efter en tids forløb står alle tal fra 1 til 10 på skærmen i vilkårlig rækkefølge, men kun en gang hver.

Vil man have tallene udskrevet på printer, slettes // i linie 30.

Også linierne 190 - 210 samt linie 300 og 320 kan sættes i funktion ved at slette //.

De frembringer en udskrift af, hvad maskinen laver, når den fremstiller et tal, prøver det og udskriver det.

Denne del af programmet giver brugeren et indblik i, hvor mange gange maskinen må forkaste et af de tal, som den foreslår i linie 140.

CBN

## Maskinkode-underprogrammer

I CBM-COMAL-80 er det muligt at anvende maskinkodeprogrammer på en fornuftig og overskuelig måde. Maskinkoden implementeres som procedurer og funktioner på maskinkodeniveau, men ligner ellers almindelige COMAL-procedurer og -funktioner. Maskinkodeprocedurer defineres ved et procedurenavn, et antal parametre, en endproc og så selve proceduren skrevet i maskinkode.

Det kræver en assembler og et rimeligt kendskab til maskinsprogsprogrammering at skrive disse procedurer og funktioner, og det vil blive for omfattende at komme ind på dette område her, hvorimod en beskrivelse af nogle få anvendelsesområder måske kan vække interesse for denne mulighed.

De fleste microcomputere er udstyret med en 8-bit parallelport, og det er derfor muligt at lave programmer til styring af mindre processer. For at gøre anvendelsen af en 8-bit port overskuelig, kunne man skrive nogle procedurer og funktioner i COMAL.

F.eks:
```
0010 PROC set(3,0)
```

Denne procedure skal sætte bit nr. 3 lav og vil derfor indeholde ordrer som PEEK og POKE mm., og vil absolut være anvendelig et stykke hen af vejen.

Problemerne melder sig, hvis de 8 bit decodes til et større antal digitale input/output, hvoraf nogle anvendes til BCD-dataopsamling eller analoge convertere. Procedurerne vil stadig kunne skrives i COMAL; men de bliver uforholdsmæssigt store og langsomme, hvilket ikke kan accepteres i industrien.

Vi kan derfor vælge at skrive det hele i maskinkode, evt. brænde det ind i en PROM og montere denne i computeren. For brugeren er COMAL -80 nu udvidet med et antal nye muligheder. Da disse underprogrammer ikke kan LIST'es, bliver selve styreprogrammet mere overskueligt, og det bliver adskilligt hurtigere.

I maskinkode har vi lavet en procedure, der kan sætte et bit:
SET(bitnummer,tilstand)
og en funktion, der kan læse et bit og returnere tilstanden:
BIT(bitnummer)
Selve maskinsprogspakken kalder vi interface.

Vi prøver nu at løse en simpel opgave: En DC-motor skal køre tilbage til sit endestop. Endestoppet er forbundet med bit nummet 124. Bit 78 starter motoren. Når bit 45 er lav, kører motoren baglæns, når den er høj forlæns.

```
0010 USE interface
0020
0030 endestop=124
0040 motor=78
0050 omdrejning=45
0060 baglæns=0
0070 forlæns=1
0080
0090 set(omdrejning,baglæns)
0100 set(motor,on)
0110 WHILE NOT bit(endestop) do
     NULL
0120 set(motor,off)
```

I linie 10 fortæller ordren USE, at vi skal bruge en maskinaprogspakke med navnet "interface". Fra linie 30 til 70 giver vi de aktuelle bit-numre navne.
I linie 90 sættes motoren til baglæns. I linie 100 startes motoren. I linie 110 gør vi ingenting, så længe endestoppet ikke er aktiveret. Når endestoppet aktiveres, fortsætter programmet. I linie 120 stopper vi motoren.

Det, der er interessant for en bruger af en computer, der er tilkoblet et styresystem, er ikke, hvordan det teknisk lader sig gøre at manipulere elektronikken til at gøre det rigtige, men derimod, hvordan han via et program kan udføre de ønskede funktioner.

Ole Holmgren

## Flash

Når man er i gang med at taste en lang række inddata, ser man som regel ikke så meget på selve skærmen. Derfor kan der smutte nogle fejl med, som kunne være undgået, hvis man i tide var blevet gjort opmærksom på dem.

I sådanne tilfælde kan man programmere en biblyd (CHR$(7)) ind; men man kan også fremkalde et kraftigt lysglimt på skærmen, som man let kan opfatte uden at se direkte på den. Programmet hertil ser sådan ud:

```
0010 LOOP
0020   INPUT " nummer  > ": nummer
0030   IF nummer=5 THEN flash
0040 ENDLOOP
0050
0060
0070 PROC flash
0080   POKE 59520,12
0090   FOR q:=1 TO 20 DO
0100     POKE 59521,0
0110     POKE 59521,16
0120   ENDFOR q
0130 ENDPROC flash
```

Proceduren <flash> er en såkaldt underrutine uden parameteroverføring. Det vil sige, at man ikke overfører værdier til underrutinen.

Linierne 0010 - 0060 er til afprøvning af proceduren. De skal slettes, før selve proceduren indgår i et større program. Bemærk linie 0030, hvor proceduren kaldes i tilfælde af, at man indtaster 5.

Frank Hejndorf

CBN

CBN hører gerne din mening.

# Systembestemte variable

Nedenstående program er ikke særlig spændende; men det illustrerer efter min opfattelse en meget praktisk form at disponere sine programmer.

Ideen er, at man i initieringsfasen sætter alle sine maskinafhængige variable. Dels letter det oversigten, og dels forenkler det muligheden for at ændre programmet, så det kan køre på en anden maskintype, end det oprindelig er skrevet for.

Et par kommentarer:

Programmet er skrevet til en CBM maskine med 40 karakterer på skærmen.

0030 buf: bufferlængde, inputbuf: inputbuffer, cur: cursor on/off skram: sk(ærm)-ram.

1090 chr(218) er grafiktegnet for ruder.

9050 tænd'cursor og sluk'cursor bruges i indtastningsprocedurer for at undgå, at cursoren efterlader hvide klatter rundt omkring på skærmen.

9180 svarer til poke 59468,12.

9220 svarer til poke 59468,14.

9260 sekundantallet laves om til jiffies.

9410 anvendes ved filer efter åbning og lukning. Således:
IF disk'status = 0 THEN // alt i orden.

Mogens Møller Nielsen

```
0010 // list "1.systembest var"
0020
0030 buf:=158; inputbuf:=623; cur:=167; skram:=32768
0040 kar'valg:=59468; grafik'store:=12; små'store:=14
0050 tasttryk:=151; shift'key:=152
0060
0070 // ========================
0080
1000 ryd'skærm
1010 skift'til'små'og'store
1020 vent(2)
1030
1040 PRINT AT 10,13: "Indledning"
1050 afvent'tast
1055
1060 ryd'skærm
1070 skift'til'grafik'og'store
1080 vent(.5)
1085
1090 PRINT AT 10,13: CHR$(218),CHR$(218),CHR$(218);"slut";CHR$(218),
     CHR$(218),CHR$(218)
1100 END ""
1110
```

```
9000 // ========================
9010 PROC tøm'buffer  // sletter 'gamle' indtastninger
9020   POKE buf,0
9030 ENDPROC tøm'buffer
9040 // ========================
9050 PROC tænd'cursor
9060   POKE cur,0
9070 ENDPROC tænd'cursor
9080 // ========================
9090 PROC sluk'cursor
9100   POKE cur,1
9110 ENDPROC sluk'cursor
9120 // ========================
9130 PROC ryd'skærm
9140   PRINT CHR$(147)
9150 ENDPROC ryd'skærm
9160 // ========================
9170 PROC skift'til'grafik'og'store
9180   POKE kar'valg,grafik'store
9190 ENDPROC skift'til'grafik'og'store
9200 // ========================
9210 PROC skift'til'små'og'store
9220   POKE kar'valg,små'store
9230 ENDPROC skift'til'små'og'store
9240 // ========================
9250 PROC vent(tid) CLOSED  // kaldes med vent(2) for 2 sekunders pause
9260   vente:=tid*60
9270   tid:=TIME
9280   WHILE TIME-tid<vente DO NULL
9290 ENDPROC vent
9300 // ========================
9310 PROC afvent'tast
9320   vent(2)
9330   PRINT AT 25,3: "(Tryk på en tast, når du er klar.)",CHR$(19)
       // home
9340   tøm'buffer
9350   REPEAT
9360     tal:=ORD(KEY$)
9370   UNTIL tal<>0
9380   PRINT AT 25,1: SPC$(39),CHR$(19)
9390 ENDPROC afvent'tast
9400 // ========================
9410 FUNC disk'status CLOSED
9420   DIM st$ OF 2
9430   st$:=STATUS$
9440   RETURN VAL(st$)
9450 ENDFUNC disk'status
9460 // ========================
```

## Anmeldelse

Leif Pehrsson, Ester M. Christensen, Bjarne Aagaard
4 * COMAL
ISBN 87-7351-066-1
222 sider. Format A 5. Kr. 129,20
2 disketter: kr. 732,00
Priserne er incl. moms.
forlaget systime a/s, Herning

Forfatterne til "Datalære med mikrodatamat" (anmeldt i CBN 1983/1) har udgivet en ny lærebog - ganske anderledes end den første, der indeholdt meget samfundsrelateret stof til faget datalære. Noget tilsvarende findes så godt som ikke i den nye bog, der stort set kun beskæftiger sig med programmeringens svære kunst, og det skal med det samme siges, at indlæringen lettes betydeligt ved brug af denne bog.

Målet er i øvrigt ambitiøst: at gennemgå hele COMAL kernen, som den benyttes af regnecentralens Piccolo; men det lykkes på bedste måde.

I begyndelsen gås der småt til værks. Der kræves ingen som helst forudsætninger.

Bogen er tænkt anvendt i gymnasiet (fortrinsvis det matematiske), på HF og på seminarierne; men der vil intet være til hinder for, at andre kan have lige så stor udbytte af den.

Mange eksempler er hentet fra matematikken; men de af dem, der vil være for avancerede for en ikke-matematiker, kan roligt overspringes, der er nok at tage fat på endda.

Tempoet sættes naturligvis op - ganske jævnt gennem hele forløbet; men på intet tidspunkt føler man sig i den velkendte situation: "Det indses umiddelbart, - - - ".

Bogens titel: "4 * COMAL" lægger jo op til visse forventninger om gennemgang af og sammenligning mellem de forskelllige maskiners måde at arbejde med COMAL 80 på. Der er da også vist billeder af 4 maskiner, ligesom deres tastatur er aftrykt; men det er stort set også det eneste, man hører til andre maskiner end Piccoloen.

Adskillige steder henviser bogen til manual'en for den aktuelle maskine eller til det skema på side 216 - 217, der viser nogle af forskellene mellem Piccolo, Comet/ Ditamat, SPC/1 og New Brain.

Men det skall straks tilføjes, at man stort set ikke savner de oplysninger, som titlne måske forjætter, for i eksemplerne har man bestræbt sig på at udforme programmerne sådan, at de maskinbestemte ordrer undgås, hvor det er muligt. Det gør det selvsagt meget lettere at anvende programmerne på andre COMAL datamater.

Bogens opbygning og udformning er præget af erfarne pædagogers viden om betydningen af systematik og klar overskuelig opbygning af den enkelte side.

Det er let at orientere sig i bogen - dens værdi som håndbog fortjener en kraftig understregning - og dens valg af eksempler med tilhørende opgøver er forbilledlig.

Et meget stort plus rummer disketterne, der er fremstillet til bogen. De indeholder såvel eksempler som øvelser og færdige forslag til opgaveløsninger.

Disketterne fås til de nævnte 4 maskiner samt til COMMODORE (CBM) og til TEXAS 99/4a.

Bogen kan anbefales som en af de bedste på markedet, og for alle undervisere i COMAL 80 er den faktisk et "must".

S. Chr. H.

## Alternativ input-rutine

I CBN 1983/3 viste Lars Lauraen en sikker input-rutine skrevet i COMAL.

Jeg har også i mit arbejde med at lave undervisningsprogrammer til folkeskolen været utilfreds med den normale INPUT-sætning, da undervisningsprogrammer som bekendt skal kunne arbejde på trods af nogle ind imellem utraditionelle elev-INPUT's.

Derfor har jeg lavet den nedenfor aftrykte rutine. Programmet er skrevet til en CBM 4032 (skærmbredde: 40 karakterer).

Programmet reagerer kun på de tegn, som man med rimelighed vil kunne forvente i de pågældende programmer samt RETURN og DEL (maskinens viskelæder).

Programmet accepterer ordlængder hen til slutningen af den linje, hvorpå man er startet (ligegyldig hvor på linjen man starter).

Hvis man ønsker at fastlægge en maximal længde på strengen for hvert INPUT, kan man lave de ændringer, der er beskrevet nedenunder programmet. Procedurekaldet hedder så f. eks. ORDIND(8).

Denne mulighed er absolut anbefalelsesværdig i forbindelse med indsættelsesprogrammer, hvor man kan undgå, at brugeren skriver oveni (og sletter) en tekst.

LÆNGDEN SKAL AFPASSES, SÅ INPUT'TET IKKE GÅR UD OVER SKÆRMENS BREDDE.

Programmet kan ikke udvides, så INPUT'tet bliver på flere linjers længde.

KORT FORKLARING TIL PROGRAMMET

110-140 er blot lavet for at vise proceduren i brug i en (meget lille) sammenhæng.

190 Svar$ sættes lig med ingenting.
Svar$ bruges til at opsummere INPUT'tet.

200+210 Vi får fastlagt, i hvilken linje og på hvilken position på linjen cursoren befinder sig ved start (ordrerne er specielle for CBM).

220 POS angiver hele tiden cursorens aktuelle position på linjen.

240 Som cursor har jeg valgt at bruge en vandret understreg.

250 IND$ er det tegn, maskinen modtager, og som skal testes.

260-270 Hvis vi ikke overskrider linjens længde kan vi afprøve, om IND$ skal summeres op i SVAR$.

280-300 Hvis tegnet hører til blandt de acceptable tegn. bliver det summeret ind i SVAR$, udskrivet på cursorens position og POS bliver talt en op.

330-390 Hvis tegnet var DEL, sker der ingenting, hvis cursoren befinder sig i startpositionen, ellers bliver cursoren slettet, POS bliver talt en ned, og det sidste tegn i SVAR$ bliver slettet.

260-420 Maskinen fortsætter med at modtage tegn, indtil der trykkes på RETURN, og SVAR$ er forskelig fra ingenting.

430 Cursoren slettes.

```
0010 // inputprogram
0020 // Torben Baunsø
0030 // august 1983
0040 // programmet kan klare op til 39 karakterers input i en streng.
0050 // man kan ikke brække et input i højre del af skærmen.
0060
0070
0080 DIM svar$ OF 40, ind$ OF 2
0090
0100
0110 PRINT chr$(147)
0120 PRINT AT 10,1: "skriv her: ",
0130 ordind
0140 PRINT chr$(17),chr$(17),"ordet var: ",svar$  // 2 x cursor ned
0150
0160
0170
0180 PROC ordind
0190   svar$:=""
0200   linje:=PEEK(216)+1  // cursorlinjen
0210   startpos:=PEEK(198)+1  // cursorpositionen
0220   pos:=startpos
0230   REPEAT
0240     PRINT AT linje,pos:chr$(164)  // understreg
0250     ind$:=GET$(0,1)
0260     IF pos<40 THEN
0270       IF ind$ IN "abcdefghijklmnopqrstuvwxyzæøå,.?;:
         0123456789+-*/='()" THEN
0280         svar$:=svar$+ind$
0290         PRINT AT linje,pos: ind$
0300         pos:=pos+1
0310       ENDIF
0320     ENDIF
0330     IF ind$=CHR$(20) THEN  // delete
0340       IF pos-1<startpos THEN
0350         CURSOR linje,startpos
0360       ELSE
0370         PRINT AT linje,pos:
0380         pos:=pos-1
0390         svar$:=svar$(1:LEN(svar$)-1)
0400       ENDIF
0410     ENDIF
0420   UNTIL ind$=CHR$(13) AND svar$<>""
0430   PRINT AT linje,pos: " "
0440 ENDPROC ordind
```

Følgende ændringer skal foretages i proceduren, hvis man ønsker at kunne fastlægge en maximal strenglængde for hvert INPUT:

130 Efter procedurekaldet ordind skal der tilføjes en parentes, der indeholder den parameter, der angiver strengens længde. F. eks.:
130 ordind (12)

180 PROC ordind (strenglængde)

260 IF pos-startpos<strenglængde THEN

## Regneprogram

Her er et program, der blev flikket sammen på et begynderkursus i kendskab til mikrodatamater.

Det prætenderer på ingen måde at være et godt undervisningsprogram, men skal blot opfattes som et eksempel på, hvorledes et program kan sammensættes af procedurer.

Undervejs forekommer en del konstruktioner, der i al deres enkelhed formodentlig kan være til gavn for andre begyndere.

Det egentlige program står i linie 90 - 110. Det sørger for stadig gentagelse. Men der er dog en ting, der ikke må gentages, og det er dimensioneringen, hvorfor denne er holdt for sig selv i den indledende procedure: opstart, der kaldes i linie 30 og deklareres i linie 50 - 70.

Progrsmproceduren (linie 130 - 170) omfatter: 1. menuen, 2. valget (og gennemførelsen) samt 3. prøvning af svarets rigtighed.

I tilrettelægningen er der taget hensyn til, at programmet skal kunne køre på alle maskiner med COMAL-80.

Der er kun benyttet maskinbestemte ordrer i linie 300 og 1270, hvor de pågældende CHR$ værdier let kan udveksles med andre relevante.

I øvrigt er procedurerne af pædagogiske grunde anført i den rækkefølge, de tages i brug. Som bekendt er dette absolut ikke et krav, som COMAL-syntaxen stiller; men det letter i hvert fald gennemgangen af programmet for nybegyndere.

I linie 360 anvendes konstruktionen "in", der er overordentlig nyttig, idet den i en meget kort form forhindrer fejlvalg i at forkludre det videre forløb.

I CASE-konstruktionen indgår "OTHERWISE". Linie 510 og 1020 viser, at man let slipper ud af spekulationer om, hvad programmet skal gøre, når der ikke kan vælges andet end det, programmet tillader: det skal slet ikke gøre noget (DO NULL), og det er jo en rimelig anvisning, når det aldrig kan havne i denne situation.

Men for øvrigt kan "OTHERWISE" jo helt udelades.

Linie 490 viser, hvordan man kommer ud af loop'en (linie 80 - 110).

I samme linie viser konstruktionen: END "", hvorledes man slipper for at få den sædvanlige bemærkning: "END AT 1330" udskrevet. I stedet ses kun en blinkende cursor.

I linie 940 - 960 samt linie 980 - 1000 er der benyttet et fif for dels at give et positivt resultat ved subtraktion og dels give et heltal som facit ved division.

Fidusen ligger i linie 940 og 980, hvor man udfører en addition henholdsvis multiplikation for at få den rette minuend og dividend.

Tallene a, b og c skal derefter naturligvis placeres i de rigtige roller i regnestykkerne (linie 950 - 960 og 990 - 1000).

P.S. Som allerede nævnt er programmet alene ment som et lærestykke i, hvordan programmer kan opbygges overskueligt; men man kunne jo henvise til, at det ville have været nærliggende at indføje muligheden for valg af opgavernes sværhedsgrad, ligesom noget med gentagelse og hjælp ved løsningen også kunne komme på tale.

I øvrigt åbner programmet rige muligheder for en forbedring af de benyttede skærmbilleder, som der slet ikke er lagt vægt på.

```
0010 // list "@1.regning"
0020
0030 opstart
0040
0050 PROC opstart
0060   DIM valg$ OF 1
0070 ENDPROC opstart
0080
0090 LOOP
0100   program
0110 ENDLOOP
0120
0130 PROC program
0140   tekst'menuvalg
0150   ønske
0160   test'svar
0170 ENDPROC program
0180
0190 PROC tekst'menuvalg
0200   clear
0210   PRINT AT 2,5: "a = addition"
0220   PRINT AT 4,5: "m = multipli-
       kation"
0230   PRINT AT 6,5: "s = subtrak-
       tion"
0240   PRINT AT 8,5: "d = division"
0250   PRINT AT 10,5: "p = program-
       slut"
0260   PRINT AT 14,5: "vælg - skriv
       bogstav - tryk return"
0270 ENDPROC tekst'menuvalg
0280
0290 PROC clear
0300   PRINT CHR$(147)
0310 ENDPROC clear
0320
0330 PROC ønske
0340   REPEAT
0350     valg$:=KEY$
0360   UNTIL valg$ IN "amsdp"
0370   CASE valg$ OF
0380   WHEN "a"
0390     addition
0400   WHEN "m"
0410     multiplikation
0420   WHEN "s"
0430     subtraktion
0440   WHEN "d"
0450     division
0460   WHEN "p"
0470     clear
0480     PRINT AT 12,15: "tak for
         denne gang!"
0490     END
0500   OTHERWISE
0510     NULL
0520   ENDCASE
0530 ENDPROC ønske
0540
0550 PROC addition
0560   PRINT AT 1,15: "addition"
0570   tal'gen("addition")
0580   PRINT AT 5,1: tal'1;"+";
       tal'2;
0590   INPUT "= ": svar
0600 ENDPROC addition
0610
0620 PROC multiplikation
0630   PRINT AT 1,15:
       "multiplikation"
0640   tal'gen("multiplikation")
0650   PRINT AT 5,1:
       tal'1;"*";tal'2;
0660   INPUT "= ": svar
0670 ENDPROC multiplikation
0680
0690 PROC subtraktion
0700   PRINT AT 1,15: "subtraktion"
0710   tal'gen("subtraktion")
0720   PRINT AT 5,1:
       tal'1;"-";tal'2;
0730   INPUT "= ": svar
0740 ENDPROC subtraktion
0750
0760 PROC division
0770   PRINT AT 1,15: "division"
0780   tal'gen("division")
0790   PRINT AT 5,1:
       tal'1;":";tal'2;
0800   INPUT "= ": svar
0810 ENDPROC division
0820
0830 PROC tal'gen(regneart$)
0840   clear
0850   a:=RND(1,9); b:=RND(1,9)
0860   CASE regneart$ OF
0870   WHEN "addition"
0880     tal'1:=a; tal'2:=b
0890     facit:=tal'1+tal'2
0900   WHEN "multiplikation"
0910     tal'1:=a; tal'2:=b
0920     facit:=tal'1*tal'2
0930   WHEN "subtraktion"
0940     c:=a+b
0950     tal'1:=c; tal'2:=a
0960     facit:=b
0970   WHEN "division"
0980     c:=a*b
0990     tal'1:=c; tal'2:=a
1000     facit:=b
1010   OTHERWISE
1020     NULL
1030   ENDCASE
1040 ENDPROC tal'gen
1050
```

```
1060 PROC test'svar
1070   IF svar=facit THEN
1080     rigtig
1090   ELSE
1100     forkert
1110   ENDIF
1120 ENDPROC test'svar
1130
1140 PROC rigtig
1150   PRINT AT 7,5: "rigtigt!"
1160   dyt(2)
1170   pause(2)
1180 ENDPROC rigtig
1190
1200 PROC forkert
1210   PRINT AT 7,5: "forkert!"
1220   dyt(5)
1230   pause(2)
1240 ENDPROC forkert
1250
1260 PROC dyt(antal)
1270   FOR i:=1 TO antal DO PRINT
       CHR$(7)
1280 ENDPROC dyt
1290
1300 PROC pause(sek)
1310   TIME 0
1320   WHILE TIME<60*sek DO NULL
1330 ENDPROC pause
```

## Ugedag

Ved aftaler om møder, fester o. lign. synes spørgsmålet om, hvornår på ugen begivenheden finder sted, at være et ustandselig tilbagevendende problem.

Dette lille program finder og udskriver ugedagen af en indlæst dato.

Når systemet venter for indtastning, indskrives dag, måned og år adskilt af komma (,) elier af et mellemrum (space).

Programmet arbejder med stor hastighed - selv ved fjerne datoer, idet ugedagen findes ved beregning (hentet fra "The Matematics Teacher").

I programmet er 1583 sat som mindste årstal. Det skyldes, at man på det tidspunkt indførte den gregorianske kalender, som vi stadig bruger.

I min udgave af dette program er der ikke taget højde for indtastninger som f. eks. den 30. februar eller den 31. april, da interessen for sådanne kørsler må anses for at være minimal. Det står naturligvis den enkelte programmør frit at "idiotsikre" sit program på sådanne punkter.

Esben Dalsgaard

```
0010 // list 1.1.ugedag
0020 DIM ugedag$(7) OF 7
0030 FOR i:=1 TO 7 DO READ ugedag$(i)
0040 DATA "lørdag","søndag","mandag","tirsdag"
0050 DATA "onsdag","torsdag","fredag"
0060
0070 REPEAT
0080   PRINT CHR$(147)
0090   INPUT "Opgiv dag, måned, år: ": dag,måned,år
0100   ok:=(år>=1583 AND måned<=12 AND dag<=31)
0110 UNTIL ok
0120
0130 IF måned<3 THEN
0140   år:-1
0150   måned:+12
0160 ENDIF
0170
0180 uge:=dag+2*måned+INT(3*(måned+1)/5)
0190 uge:=uge+år+år DIV 4-år DIV 100+år DIV 400+2
0200 ugedag'nr:=uge MOD 7+1
0210
0220 PRINT
0230 PRINT "ugedag: ",ugedag$(ugedag'nr)
0240
0250 END""
```

C B G
COMAL BRUGER GRUPPE
MINDEGADE 42
8700 HORSENS

tlf 05 - 62 15 67
giro 1 75 23 75

# I N D M E L D E L S E  I  C B G

Indmeldelse foretages ved udfyldelse af nedenstående blanket, der sendes til ovenstående adresse.

Samtidig indbetales kr. 90,00 - prisen for 1 års abonnement på "COMAL BRUGER NYT", der agtes udgivet med 6 numre pr. år.

Indbetaling kan ske pr. check eller ved overførsel til:

Giro nr. 1 75 23 75, COMAL BRUGER GRUPPE, Mindegade 42, 8700 Horsens.

Undertegnede

Navn:.....

Adresse:
Gade:.....

Postnummer:..... By:

Indmelder sig i COMAL BRUGER GRUPPE.

Kr. 90,00 vedlægges i check ....

indbetales pr. giro ....

Sæt venligst X

Venlig hilsen

COMAL BRUGER GRUPPE

S. Chr. Hansen