C B N

ORGAN FOR COMAL BRUGER GRUPPE DANMARK

R

1. ARGANG - NR. 6.

NYE MASKINER

[1984]

I sidste nummer af CNB omtalte vi på denne plads det spændende i den udvikling, der foregår vedrørende COMAL.

Denne gang vil vi fokusere på en anden spændende udvikling:

Den tekniske.

Næsten daglig (mildt overdrevet) dukker der nye mikrodatamater op.

Mange af dem er som snydt ud af næsen på "de store", og det er i sig selv ikke særlig spændende; men prisen kan være det. Efterlignere sparer jo en god del af udviklingsomkostningerne.

Mest spændende er de banebrydende nye maskiner. F. eks. 16 bit maskinerne.

Nu har vi i en masse år (skal vi sige 4-5 stykker) vænnet os til 8 bit datamater og set en fortsat fuldkommengørelse af disse.

Nu kan de da ikke blive bedre!

Jo, det kan de såmænd nok; men så kommer presset fra den nye teknologi (= indsigt i teknikken) og skaber os den nye teknik (maskinerne og deres funktioner), der åbner helt nye muligheder: større hastighed, større lager, større antal kommandoer o. s. v.

Og det glæder vi os til at se og til at få andel i.

Også i Danmark er man med i denne udvikling.

Uden at træde andre for nær tør man vel anføre, at mange forventninger knytter sig til Regnecentralens Partner og Piccoline, som vi håber at kunne sige mere om, når de inden længe er tilgængelige.

Det samme gælder naturligvis andre fabrikater, der vil blive afprøvet og beskrevet, så snart vi er i besiddelse af dem.

INDHOLDSFORTEGNELSE

HÆDERSPRIS TIL BØRGE CHRISTENSEN

Regnecentralen har indstiftet en hædersprис, der hvert år skal uddeles til en person, som har gjort en særlig indsats i forbindelse med anvendelsen af datamater til undervisningsbrug.

Hædersprisen navn er Niels Ivar Bech-prisen efter Regnecentralens første direktør, der var særlig interesseret i en dansk EDB-udvikling inden for undervisningsområdet.

I år blev prisen tildelt seminarielektor Børge R. Christensen, Tønder.

Børge Christensen vil være kendt for dette blads læsere som faderen til COMAL og som den ihærdige forkæmper for udbredelsen og standardiseringen af dette sprog.

Vi kan med glæde konstatere, at flere og flere datamater efterhånden kan fås med COMAL, og at også stadig flere lande får øjnene op for COMAL's fordele - ikke mindst til undervisningsformål.

Til hædersprisen føjer vi vor hyldest og anerkendelse af Børge Christensens store og uegennyttige indsats.

Til lykke Børge!

Søren

REDAKTIONELT

Med COMAL BRUGER NYT nr. 6, som De nu sidder med i hånden, er 1. samling (1. årgang = 1983/84) komplet.

Til abonnenterne er der indlagt et girokort lydende på 90 kr. som betaling for 2. samling (2. årgang = 1984/85), der også bliver på 6 numre.

Et kig ind i fremtiden:

Vi tager nu hul på en artikelrække om hashteknik.

Det har som bekendt ikke noget at gøre med narkotika, men er en fremgangsmåde, der gør det muligt at søge meget hurtigt i store databaser.

I den seneste tid har vi fået adskillige henvendelser fra brugergrupper i andre lande. De fleste udgiver som CBG selv COMAL blade, og der kommer muligvis en udveksling af såvel programmer som tidsskrifter igang.

Hvis der blandt læserne er interesse for det, kan der blive tale om at fremstille disketter med de mest efterspurgte programmer fra CBN. Læserne vil så ved at rekvirere en sådan kunne slippe for ulejligheden med indtastning m. v.

Nye abonnenter kan ved henvendelse til redaktionen erhverve de hidtil udkomne numre af COMAL BRUGER NYT.

REGNECENTRALEN
Niels Bach:
Rc Comal 80
Brugervejledning.

ANMELDELSE

Regnecentralens brugervejledning til COMAL har hidtil foreligget i løsblade til indsættelse i ringbind.

Den nyeste er i modsætning hertil udformet som en indbundet bog i et tiltalende struktureret omslag og i et vældig handy format: 21 x 17 cm.

Bogen er på 263 sider, men forekommer absolut ikke klodset.

På disse sider findes snart sagt enhver oplysning, brugeren kan begære.

De første ca. 80 sider beskriver, hvordan man starter sin RC 700 (Piccolo), indlæser styresystemet CP/M og sproget Rc Comal 80, hvordan de to forskellige tastaturtyper er indrettet, og hvordan man indtaster programmer, formatterer disketter og kommanderer maskinen til at udføre programmer.

Herunder gennemgås alle vigtige oplysninger om fremgangsmåder ved programmering, f. eks. anvendelse af strenge, kontrolstrukturer, diskordrer, sekventielle og random acces filer, arrays, procedurer og funktioner samt errorhandler. Denne del afsluttes med en gennemgang af COMAL kommandoerne.

I det følgende afsnit gennemgås alle Rc Comal 80 nøgleord.

De er opført alfabetisk, og det er således let at finde rundt blandt dem.

Gennemgangen er konsekvent og systematisk. Alt er beskrevet i et letforståeligt sprog, og hvor det ikke er uundgåeligt, har man anvendt danske ord frem for engelske.

For hvert nøgleord anføres det, om det drejer sig om en COMAL sætning eller kommando, og der angives format, anvendelse, evt. virkemåde samt bemærkninger og kommentarer. Desuden er der talrige programeksempler, der illustrerer, hvordan det pågældende nøgleord bruges, og hvad det udfører.

Disse mindre programeksempler er i tillæg F suppleret med et antal programmer, der viser, hvorledes COMAL´s mange muligheder kan arbejdes sammen og løse større opgaver.

Alle disse programmer er udvalgt og skrevet med kyndig hånd og er en guldgrube for brugeren, der ikke alene kan hente nyttig viden deri, men tillige inspiration og direkte hjælp.

Fejlmeddelelser, ydre enheder og ASCII koden behandles i hvert sit tillægsafsnit.

Alt i alt er denne brugervejledning vellykket. Skulle man ønske sig en tilføjelse ved en evt. senere revision, så skulle det være et egentlig stikordsregister, der ville lette brugen af bogen yderligere.

Den er naturligvis først og fremmest beregnet på anvendelse af de brugere, der arbejder på Regnecentralens Piccolo; men dens værdi rækker langt derudover. Også andre COMAL brugere kan have glæde og udbytte af bogen, som herved anbefales på det bedste.

S. Chr. H.

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

Tryk:
tekst & tryk
Horsens

Abonnement:
Mindegade 42
8700 Horsens

REGISTER 2

I artiklen "Simpelt register " CBN nr. 5 side 4 blev det nævnt, at dette register ikke måtte opfattes som et færdigt arbejdsredskab, men mere som en ny ide om, hvordan man på en nem måde kan fremstille en såre enkel database.

Adskillige læsere har reageret på artiklen og fortalt, at de har lavet sig et register over det viste program, og de har i det store og hele været tilfreds med registret (med de begrænsninger, det ganske naturligt rummer). Men der er også fremkommet ønsker om ændringer og tilføjelser:

Det ville være rart, hvis registret ikke accepterede en tom record, der kunne fremkomme, hvis man under indskrivningen i stedet for en tekst kom til at taste SPACE eller RETURN.

Det er klaret ved korrektion af linie 630 og tilføjelse af linie 635 og 645 (se nedenfor).

Og så er der fremsat ønske om to nye funktioner: det burde være muligt at tilføje en record på en bestemt plads mellem de eksisterende, og man burde kunne udskrive registret eller dele deraf på printer.

De nævnte ønsker opfyldes med de tilføjelser til programmet, som vises nedenfor.

```
Tilføjelser:
0242      WHEN "p"
0244        printer'udskrift

0302      WHEN "t"
0303        indsæt

Korrektion:
0630      IF tekst$="" AND temp$<>"" THEN EXIT

Tilføjelser:
0635      IF tekst$=" " AND temp$<>"" THEN EXIT
0645      IF tekst$="" OR tekst$=" " THEN temp$:=""

1375    x:=1
1376    PRINT "nummer 1"

1424        x:+1

1435        PRINT "nummer ",x

1482    IF pil=LEN(reg$) THEN
1484      sidste
1486    ELSE


1625    PRINT CHR$(147)


Tilføjelsen med de nye procedurer:
1920
1930 PROC sidste
1940   PRINT "Ikke flere records    Ny kommando = h"
1950   REPEAT
1960     k$:=KEY$
1970   UNTIL k$="h"
1980 ENDPROC sidste
1990
```

```
2000 PROC indsæt
2010   INPUT "Hvilket nummer skal den stå som? ": ny
2020   temp$:=""
2030   LOOP
2040     INPUT "": tekst$
2050     IF tekst$="" AND temp$<>"" THEN EXIT
2060     IF tekst$=" " AND temp$<>"" THEN EXIT
2070     temp$:+tekst$+CHR$(13)
2075     IF tekst$="" OR tekst$=" " THEN temp$:=""
2080   ENDLOOP
2090   beg:=1; x:=1
2100   REPEAT
2110     delta:=CHR$(0) IN reg$(beg+1:LEN(reg$))
2120     x:+1
2130     beg:+(delta+1)
2140   UNTIL x=ny OR beg>LEN(reg$)
2150   IF ny=1 THEN
2160     reg$:=CHR$(0)+temp$+reg$
2170   ELSE
2180     reg$:=reg$(1:beg-1)+temp$+CHR$(0)+reg$(beg:LEN(reg$))
2190   ENDIF
2200 ENDPROC indsæt
2210
2220 PROC printer'udskrift
2230   PRINT "Ønskes selektiv udskrift j/n "
2240   REPEAT
2250     k$:=KEY$
2260   UNTIL k$ IN "jn"
2270   beg:=1
2280   ud:=0
2290   IF k$="j" THEN ud:=1
2300   REPEAT
2310     slut:=CHR$(0) IN reg$(beg+1:LEN(reg$))
2320     temp$:=reg$(beg+1:beg+slut-1)
2330     PRINT temp$
2340     IF ud=1 THEN
2350       vælg
2360     ELSE
2370       pr'udskriv
2380     ENDIF
2390     beg:+slut
2400   UNTIL beg=LEN(reg$)
2410 ENDPROC printer'udskrift
2420
2430 PROC vælg
2440   PRINT "Ønskes udskrift  j/n "
2450   REPEAT
2460     k$:=KEY$
2470   UNTIL k$ IN "jn"
2480   IF k$="j" THEN pr'udskriv
2490 ENDPROC vælg
2500
2510 PROC pr'udskriv
2520   SELECT OUTPUT "lp:" // COMAL 1.02: SELECT OUTPUT "lp"
2530   PRINT temp$
2540   SELECT OUTPUT "ds:" // COMAL 1.02: SELECT OUTPUT "ds"
2550 ENDPROC pr'udskriv
```

Bemærkninger til de enkelte ændringer:

Kommando "u":
Udskriften kommer nu med tilføjelse af "nummer " for hver record (se linie 1375,1376 og 1435) og slutter nu med:
Ikke flere records Ny kommando = h
(procedure "sidste" linie 1930 - 1980).

Kommando "s":
Slutter nu, når der ikke er flere records, der indeholder søgeordet, som ved kommando "u" med:
Ikke flere redords Ny kommando = h

Kommando "t":
Det er den nye kommando, ved hjælp af hvilken man kan tilføje en record på en bestemt plads mellem de eksisterende, idet programmet spørger:
Hvilket nummer skal den stå som?
(procedure "indsæt" linie 2000 - 2200).
For at man kan svare korrekt, må man først have fundet pladsen ved hjælp af "u" (udskriv).
Når nummeret er indtastet, foregår tilføjelsen ganske som en normal indskrivning.
Bruger man et recordnummer, der er højere end det for tiden højeste i registret, så får record'en simpelthen det nummer, der følger efter det hidtil højeste, uanset hvor højt et nummer man valgte.

Kommando "p":
Ønskes udskrift på printer, vælges "p" (procedure printer'udskrift linie 220 - 2410), og programmet spørger:
Ønskes selektiv udskrift j/n. (Linie 2230).
Med selektiv udskrift forstås udskrift af en eller flere udvalgte records.
Svarer man "j", viser programmet samtlige records - en efter en - og spørger hver gang:
Ønskes udskrift j/n. (Procedure "vælg" linie 2430 - 2490).
Svares "j", skriver printeren straks den pågældende record ud på papiret præcis som den står på skærmen.
Svares "n", kommer næste record frem på skærmen, og spørgsmålet om udskrift stilles igen.

Sådan fortsættes, til alle records har været fremme på skærmen.
Svarer man - på spørgsmålet om selektiv udskrift - i stedet "n", skriver printeren øjeblikkelig samtlige records ud.
Udskrivningen kan til enhver tid standses ved et tryk på STOP; men har man rettet eller tilføjet i registret, går disse ændringer tabt, med mindre man forinden har benyttet "*", der får programmet til at lægge det korrigerede register over på disketten.

NB. Nogle printere skal indstilles, så de selv sørger for ny linie (LF = linefeed), ellers vil flere linier i en og samme record blive skrevet oven i hinanden.

SCREEN-DUMP

De fleste skærmkopiprogrammer er opbygget på følgende måde:
1) Hvor starter skærmen?
2) PEEK den første skærmadresse
3) Omsæt den til ASCII kode
4) Udskriv tegnet på printeren
5) Fortsæt således, til alle tegn er overført

Denne metode er lidt besværlig. Men ligesåvel som man kan åbne tastaturet som en fil, sådan kan man også åbne skærmen som en fil, hvorved det hele blliver meget enklere (og hurtigere), idet punkt 2) og 3) slås sammen til eet. Ligeledes udskrives der nu en hel linie ad gangen.

Programmet benyttes på følgende måde:

Ligger det allerede i computerens arbejdslager, flytter man cursoren hen på en tom linie - IKKE DEN SIDSTE! - og skriver RUN.

Findes programmet ikke i arbejdslageret, men på en diskette, gøres som ovenfor, idet man skriver RUN "SKÆRMKOPI".

Lidt om programmet, der er skrevet på en CBM 4032:

I linie 130 aflæses, hvor på skærmen der er skrevet RUN (for COMMODORE 64: PEEK (214)). Denne linie bliver derefter slettet ved hjælp af proceduren slet´linie.

I linie 150 kaldes cursoren "hjem", og skærmens startadresse aflæses. PEEK(197) indeholder den "høje" adresse for, hvor cursoren befinder sig (for COMMODORE 64: PEEK(210)), og denne adresse er nok, da skærmen altid skal starte fra begyndelsen af en "side".

Skærmen åbnes i linie 170, og da cursoren er kaldt "hjem", vil maskinen automatisk starte med at læse den første skærmadresse.

I linie 180 aflæses, om computeren er indstillet til små/store bogstaver (hvis den ikke er det, vil her normalt stå 12). Det er nødvendigt at man ved dette, for at man kan give printeren (her en CBM 4022) den rigtige sekundæradresse: S7 betyder "skriv med små/store bogstaver", og s8 betyder "skriv med store bogstaver/ grafik". For COMMODORE 64 ændres PEEK (59468) til PEEK(53272) og 14 ændres til 23.

I linie 290 aflæses, om det aktuelle tegn er skrevet med revers skrift.

Linie 320 fanger en evtuel RETURN. Man kan ellers risikere, at printeren springer en ekstra linie over.

Ønsker man at have rutinen som en del af et program, skal linierne 130 - 140, 480 - 490 samt proceduren slet´linie fjernes.

Da proceduren er lukket, kan den kaldes flere gange, uden at der forekommer redimensionering af b$.

Er man den lykkelige ejer af en CBM 8032, kan programmet let ændres til at passe til denne.

Birger Dam Sørensen

PS. Lad mig nævne, at det er en god ide at have en diskette, hvor rutiner, man ofte bruger, er LIST´et ind. Så kan man, efterhånden som man skal bruge dem, MERGE dem ind i de programmer, man laver.

```
0010 // list"§:1.skærmkopi"
0020
0030 // for cbm 4032 - printer 4022
0040
0050 // birger dam sørensen 1984
0060
0070 skærmkopi
0080
0090 PROC skærmkopi CLOSED
0100   DIM b$ OF 3*40
0110   rvs$:=CHR$(18); off$:=CHR$(146); hjem$:=CHR$(19)
0120   SELECT OUTPUT "ds:"
0130   start:=PEEK(216)
0140   slet´linie
0150   PRINT hjem$,
0160   skærmstart:=PEEK(197)*256
0170   OPEN FILE 3,"ds:",READ
0180   IF PEEK(59468) BITAND 14=14 THEN
0190     SELECT OUTPUT "lp:/s7"
0200   ELSE
0210     SELECT OUTPUT "lp:/s8"
0220   ENDIF
0230   PRINT CHR$(13)
0240   SELECT OUTPUT "lp:/s0"
0250   FOR linie:=0 TO 24 DO
0260     b$:=""
0270     FOR position:=0 TO 39 DO
0280       a$:=GET$(3,1)
0290       IF PEEK(skærmstart+((linie*40)+position))>127 THEN
```

```
0300          b$:+rvs$+a$+off$
0310        ELSE
0320          IF a$=CHR$(13) THEN a$:=" "
0330          b$:+a$
0340        ENDIF
0350      ENDFOR position
0360      PRINT b$
0370    ENDFOR linie
0380    CLOSE FILE 3
0390    SELECT OUTPUT "ds:"
0400
0410    PROC slet'linie
0420      CURSOR start,1
0430      PRINT SPC$(40)
0440    ENDPROC slet'linie
0450
0460 ENDPROC skærmkopi
0470
0480 PRINT CHR$(19),
0490 END ""
```

NB: I linie 0010 skal tegnet § erstattes med commercial "at" (snabel-a).

CBN

LÆSERBREV

Jeg er netop blevet medlem af CBG og har derfor modtaget de hidtil udkomne numre af CBN.

Det undrer mig, at der slet ikke foregår nogen debat om anvendeligheden af COMAL. Man hører i det hele taget sjældent kritiske røster angående COMAL.

Hvorfor er der ingen kritik? Skyldes det, at COMAL virkelig er uovertruffet, eller skyldes det, at COMAL er blevet så udbredt på danske skolemaskiner, at det nærmest er standard? Al standardisering må selvfølgelig glæde enhver.

Som matematiker tiltales jeg af COMAL's gennemførte strukturering; men jeg er kommet til at tvivle på, om det egentlig er det bedste sprog for f. eks. folkeskoleelever, der kun skal lære den mest elementære programmering i forbindelse med et kursus i datalære.

Jeg tror det ikke.

Jeg tror, at man i folkeskolen skal holde sig til den kerne som findes i BASIC.

Lad os ikke glemme, at COMAL er en krydsning mellem PASCAL og BASIC, som jo begge er gode til hvert sit formål: PASCAL til at lave store, omfattende programmer uden for mange fejl, og BASIC til at lave små programmer, uden at der fordres megen viden om programmering.

Fordelen ved COMAL er, at sproget i en vis udstrækning kan bruges til begge formål; men man bør være sig bevidst, hvilket formål man vil tilgodese, når man laver programmer.

Den kraftige strukturering med mange procedurer er jo en god hjælp til at undgå fejl i programmeringen. En dygtig programmør vil let kunne gennemgå de enkelte alternativer og se, hvad der vil ske i forskellige tilfælde.

COMAL programmer er lette at overskue for rimeligt gode programmører; men det betyder ikke, at COMAL programmer er let overskuelige for nybegyndere. Man kan som regel ikke med et blik overskue strukturen i et program. Det indeholder ofte en logisk struktur med mange niveauer, der ikke fremgår umiddelbart. Meget ofte opererer man med procedurer, selv om hver enkelt af disse kun anvendes et enkelt sted i programmet. Det ser nemlig så ordentligt og struktureret ud. Specielt procedurebegrebet kan gøre det svært umiddelbart at overskue programmerne. Procedurerne står tit langt fra det sted, hvor de anvendes, og de indeholder ofte andre procedurer på flere niveauer.

Ved simple programmer burde man måske helt undgå procedurer, eller man skulle måske kun anvende dem til noget, der skal bruges flere steder i et program.

Ofte ser man et program opbygget således:

```
0010 ind
0020 beregn
0030 skriv
```

Herefter følger så de tre anvendte procedurer. Noget sådant giver kun overblik, dersom man er meget inde i, hvad der ligger gemt i de tre procedurer. Disse skal altså læses først, før man kan læse og forstå hovedprogrammet. Det er også svært at gennemskue, hvilket niveau en given procedure er på, da der ofte er tale om flere forskellige.

Jeg har også erfaret, at strukturen:
if - then - elif - elif - else - endif
ikke er helt så let at anvende, som man skulle tro. Det skyldes muligvis problemet med at erkende forskellen på elif og else.

Jeg tror også, at den ret simple konstruktion:
if - then - else - endif
virker anderledes end den logiske struktur, som vi normalt bruger i vort daglige sprog.

Det er ret sjældent, at vi specificerer både en handling, hvis et eller andet indtræffer, og en handling, hvis det ikke indtræffer.

Ofte vil vi blot undlade at handle, hvis ikke en bestemt hændelse indtræffer.

Netop derfor oplever de fleste nybegyndere BASIC strukturen:
if - then
som værende mere i overensstemmelse med deres sædvanlige logiske tænkemåde.

Det vil ofte for nybegyndere være lettere at bruge en række betingede sætninger efter hinanden.

Det er kun i få tilfælde, man får problemet med at skullle hoppe på grund af nogle betingede sætninger, som ikke har en tom "fællesmængde".

Løkkestrukturerne:
while - endwhile og repeat - until
har også voldt store problemer, fordi man skal have et meget stort overblik over det program, man er ved at lave. Ellers kan man ikke afgøre, om det er smartest at teste i begyndelsen eller i slutningen af en løkke.

Her tror jeg, at vi vil få stor gavn af loop - endloop strukturen, som findes i nogle COMAL versioner. Den, tror jeg, ligger meget tæt på normal tankegang.

Jeg håber, at andre kan bidrage med erfaringer omkring de problemer, jeg her har rejst, og jeg imødeser med glæde en udveksling af alt erfaringsmateriale omkring brugen af COMAL sproget, så vi kan udnytte det bedst muligt.

Med venlig hilsen

Erik Dam
Skjoldsvej 27
3650 Ølstykke

TERNINGKAST

Et program uden krumspring.

Ved at vælge et voksende antal kast kan man iagttage, hvorledes fordelingen mellem forekomsten af de enkelte "øjne" bliver mere og mere ligelig.

Bemærk den lette optælling i linie 130 - 150 og anvendelsen af KEY$ i linie 180.

I linie 230 slettes de tidligere resultater af hukommelsen.

S.J.

```
0010 // list "1.terningkast"
0020 DIM a(6), k$ OF 1
0030 REPEAT
0040   PRINT CHR$(147);
0050   mærke:=0
0060   FOR n:=1 TO 5 DO PRINT
0070   INPUT "antal kast ": n
0080   FOR y:=1 TO n DO
0090     x:=RND(1,6)
0100     a(x):+1
0110   ENDFOR y
0120   FOR n:=1 TO 2 DO PRINT
0130   FOR y:=1 TO 6 DO
0140     PRINT "antal  ",y,"'ere ",a(y)
0150   ENDFOR y
0160   PRINT
0170   PRINT "flere kast   j/n "
0175   REPEAT
0180     k$:=KEY$
0185   UNTIL k$ IN "jn"
0190   IF k$="n" THEN
0200     mærke:=0
0210   ELSE
0220     mærke:=1
0230     FOR y:=1 TO 6 DO a(y):=0
0240   ENDIF
0250 UNTIL mærke=0
```

C-B-N

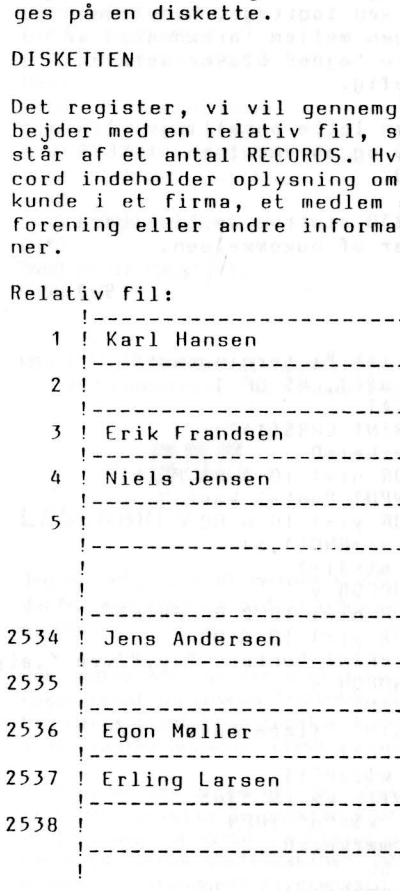
HASH REGISTER

I sidste nummer af "CBN" beskrev vi et simpelt register, der blot skulle vise et pricip. I denne artikel starter vi på opbygningen af et HASH-register; men før vi begynder, vil vi lige gennemgå lidt om, hvordan et register kan opbygges på en diskette.

DISKETTEN

Det register, vi vil gennemgå, arbejder med en relativ fil, som består af et antal RECORDS. Hver record indeholder oplysning om en kunde i et firma, et medlem af en forening eller andre informationer.

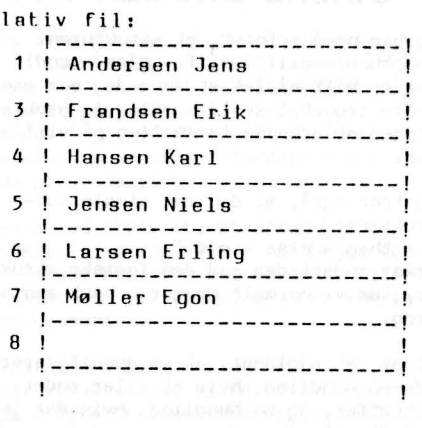
Relativ fil:



Fidusen ved en relativ fil er nu, at vi kan læse indholdet af record nr. 3, og omgående vil "Erik Frandsen" være indlæst. Man skal ikke søge i filen, hvis man kender nummeret på den record, hvori den ønskede oplysning står. Problemet kan være at huske alle disse numre.

I eksemplet står navnene ikke i nogen bestemt rækkefølge. Man kunne forestille sig at de stod i alfabetisk rækkefølge.

Relativ fil:



Hvis navnene er sorteret i alfabetisk rækkefølge, er det muligt at lave binær søgning og dermed rimelig hurtigt finde det ønskede navn uden at kende record-nummeret.

Ulempen ved at forvente, at alle records altid står i alfabetisk orden, er, at det medfører, at et nyt navn, der indskrives, rent fysisk skal sorteres på plads i alfabetisk rækkefølge. En sådan indskrivning kan således medføre en større række fysiske overflytninger af indholdet af en større række records, hvilket er meget tidskrævende.

HASH-REGISTER

Med et HASH-register tilnærmer man sig et register, hvori der ikke skal søges efter oplysningerne, - når man har navnet, kan man slå direkte op i den record, hvori de ønskede oplysninger befinder sig -, og samtidig har man et register, hvori det er hurtigt at oprette nye navne og slette eksisterende navne.

Hvis man ønsker at skrive et hash-register, starter man med først at skaffe sig en pibe skåret i bornholmsk åndenød. Når denne er bragt til veje, skal man implementeres i et af de mere skumle miljøer, hvor stoffet handles. Nu er det så bare at vente på fuldmånens komme. Piben stoppes først med babyhånd, derefter med damehånd og sidst med mandehånd, hvorefter den fyres af.

Når virkningen indtræffer (det mærkes tydeligt), sætter man sig til tastaturet og basker løs en halv times tid, - og man har sig nu et hash-register.

Der findes dog også en anden metode at lave et hash-register efter.

Hovedprincippet i et hash-register er, at man nedbryder søgenøglen (navnet på en person, et firmanavn eller lignende) til et tal, som er det record-nummer, hvori man placerer oplysningerne med den pågældende søgenøgle. SØGENØGLEN LAVES OM TIL ET TAL, SOM ER RECORD-NUMMERET.

```
0010 INPUT "Navn: ":søgenøgle$
0020 recordnr=hash(søgenøgle$)
```

Til funktionen "hash" overføres en tekst, og der returneres et tal. - Hvad er kravene til denne funktion?

Vi skal forestille os registeret som et stort skab (relativ fil) med måske 4000 skuffer (records). Hver skuffe indeholder oplysninger om et individ, og skufferne er nummereret fra 1 til 4000 svarende til record nr. 1 til record nr. 4000.

Hvis vi har en række oplysninger, som skal gemmes under navnet "Egon Møller", sendes navnet til hash-funktionen, som f.eks. returnerer tallet 2536. Vi gemmer nu oplysningerne i skuffe (record) nr. 2536. Kravet til hash-funktionen er nu, at den for et hvilket som helst andet navn altid returnerer et nummer, der er forskelligt fra 2536 og samtidig ligger i intervallet 1-4000, og hvis navnet "Egon Møller" anvendes, skal hash-funktionen altid returnere tallet 2536 - ellers kan vi jo ikke finde oplysningerne igen.

Lykken ville være, hvis en sådan funktion kunne laves! Det ville nemlig betyde, at man ikke skulle bekymre sig om numrene, men alene kunne bruge navne, når man opretted og hentede oplysninger fra et register. Man skulle aldrig søge i registeret, men ville altid kunne slå direkte op på de ønskede oplysninger, hvilket i praksis ville betyde, at uanset hvor mange records der var i et register, ville oplysningerne, ved indtastning af navnet alene, kunne læses ind i computeren i løbet af et sekund.

Hash-registeret løser faktisk denne opgave, omend der er nogle specialtilfælde at tage hensyn til; men i store træk virker det som ovenstående. Og hvordan ser denne vidunder-funktion da ud? Ja - den kan faktisk laves på mange måder, der alle vil virke mere eller mindre godt.

Kravene var jo, at der skulle genereres et tal ud fra en tekst. Dette tal skulle ligge inden for et bestemt interval (f. eks. 1-4000) og to forskellige tekster måtte ikke returnere samme tal, - eller sagt på en anden måde -, vi ønsker en STOR SPREDNING AF TALLENE INDEN FOR DET GIVNE INTERVAL.

```
0010 FUNC hash(tekst$)
0020   sum:=0
0030   FOR i:=1 TO len(tekst$) DO
0040     sum:+ord(tekst$(i))
0050   ENDFOR i
0060   RETURN sum
0070 ENDFUNC hash
```

Ovenstående funktion vil returnere et tal, der er udledt af teksten. Men der er et krav, der ikke er opfyldt. Det er ikke sikkert, at tallet vil ligge indenfor et på forhånd givet interval. Dette kan dog nemt klares.

```
0060 RETURN sum MOD 4000
```

MOD returnerer resten efter heltalsdivisionen. Dvs. at uanset hvor stort et tal "sum" indeholder, vil der nu altid returneres et tal, der er større end eller lig med nul og mindre end 4000. Da mange relative filsystemer starter med record nummer 1 og ikke 0, kan vi addere 1.

```
0060 RETURN sum MOD 4000+1
```

Vi ved endnu ikke, om denne funktion vil leve op til kravet om stor spredning. Det bedste, man kan gøre i forbindelse med hash-algoritmen, er at afprøve den i praksis ved at lade funktionen tygge eks. 100 navne igennem og undersøge, om spredningen er god nok. Ovenstående funktion vil vise sig at være særdeles dårlig.

En effektiv måde at øge spredningen på er at anvende primtal som basetal mm.

```
0010 FUNC hash(tekst$)
0020   sum:=0
0030   FOR i£:=1 TO LEN(tekst$)-1
       DO
0040     sum:+ORD(tekst$(i£))*107
0050     sum:+ORD(tekst$(i£)+1)
         *431
0060   ENDFOR i£
0070   RETURN sum MOD 4001+1
0080 ENDFUNC hash
```

Ovenstående funktion arbejder med en rimelig spredning; men det er tilrådeligt at afprøve sin hash-funktion med eksempler på nogle typiske tekster i det aktuelle register. Pas på at "sum" ikke antager så stor en værdi, at computeren går over i eksponentnotering.

Selv om hash-funktionen nu returnerer recordnumre med stor spredning, vil nogle tekster uundgåeligt returnere recordnumre, der allerede er optaget og anvendt i registeret. Før vi går over til at tage hensyn til disse specialtilfælde, skal vi lige se på, om der er mere, vi kan gøre for at undgå dette. Og det er der faktisk.

Forholdet mellem, hvor mange records vi ønsker at udnytte, og vort basetal er af stor betydning. Hvis vi ønsker at lave et register, hvor der skal være plads til 4001 individer, og vort basetal er 4001, kan vi hurtigt se, at de sidste 1500 individer, vi indtaster, har alle mulige chancer for at ramme oveni et allerede anvendt recordnummer.

Hvis vi ønsker, at registeret skal kunne indeholde 2000 individer, og vort basetal er 4001, bliver dette problem reduceret betydeligt. Vi kan altså se, at der skal nogen albueplads til i registeret. Faktisk skal vi rent fysisk oprette et register, der er betydelig større end den plads, vi kommer til at udnytte, og det koster mange kilobytes ude på disketten til ingen verdens nytte. Derfor arbejder et hash-register normalt over en indexfil.

Denne indexfil er igen en relativ fil med f.eks. 4001 records, hvor hver record kun indeholder et nyt recordnummer; nemlig nummeret på en record i en anden relativ fil, hvor oplysningerne står, og som måske kun har 2000 records. På denne måde har vi reduceret muligheden for at ramme et recordnummer, der allerede er anvendt, uden at bruge oceaner af kostbar plads på disketten.

I næste nummer vil vi begynde at se på, hvordan de to relative filer kan oprettes og organiseres, samt hvordan programmerne til at arbejde med disse filer kan laves.

O.H.

NB: tegnet £ skal erstattes med nummertegnet ("havelågen").

C-B-N

SPØRGEBREVKASSEN

Spørgsmål:

Hvis to COMMODORE maskiner bruger IEEE-488 bussen samtidig, opstår der "koks".

Kan man lave en rutine, der checker bussen, umiddelbart før man tager den i brug, og stiller maskinen i venteposition, hvis bussen er optaget.

Vi kan bruge rutinen i programmer i folkeskolen, hvor der bruges printer og disc.

Svar:

COMMODORE maskinerne er rigtigt nok udstyret med en IEEE-488 bus, som er beregnet til, at en enkelt computer umiddelbart kan kobles til et større antal ydre enheder (printer, floppy disk, modem, plotter, digitizer mm.), som blot har forskellige adresser på denne bus. Hvis flere computere alle skal anvende disse enheder, kræver det et multibrugersystem. Dog accepterer bussen, at flere computere kobles sammen med disse enheder, og de enkelte computere kan hver for sig uden problemer anvende alle disse enheder, - den eneste begrænsning i denne opstilling vil være, at to eller flere computere ikke må anvende bussen samtidig.

Hvis man ikke kan leve med denne begrænsning, skal man enten anskaffe et multibrugersystem, eller selv fremstille en metode, der kan afhjælpe problemet.

Det er ikke muligt at checke bussen for aktivitet og derefter sikre sig at den er ledig, før man selv går på bussen. Men da alle COMMODORE maskiner er udstyret med en 8-bit userport, er det muligt at anvende denne til en simpel løsning på problemet.

Den ene metode vil være at trække en ledning med to ledere fra computer til computer (stel og bit 0) og anvende dette signal som et busy/not busy signal. Eller man kan anvende alle 8 bit og dermed også muligheden for at lave en kø med prioritet. Selve rutinen til test og administration af disse signaler bør laves i maskinkode.

Spørgsmål:

Længden af en streng, der indskrives i en datasætning, begrænses af 4 pladser til linienummer, 1 space og 4 pladser til ordet DATA.

Man taber altså pladser til 9 tegn.

Hvordan kan man undgå det, sådan at en "data streng" kan blive på 80 karakterer?

Svar:

Man vil altid tabe de første 9-11 tegn i en datasætning, men det skulle ikke være noget problem i et program.

Udprintning af strenge fra datasætninger kan gøres på følgende måde:

```
0010 DIM TEKST$ OF 80
0020
0030 WHILE NOT EOD DO
0040   READ TEKST$
0050   PRINT TEKST$,
0060 ENDWHILE
0070
0080 END "Programmet er slut"
0090
0100 DATA "Dette er en streng på
     80 tegn, som skal "
0110 DATA "printes på skærm og læ-
    ·ses fra datasætning"
```

Hvis en tekst skal læses fra datasætning og over i en streng, kan det gøres således:

```
0010 DIM TEKST$ OF 80, TEMP$ OF 70
0020
0030 WHILE NOT EOD DO
0040   READ TEMP$
0050   TEKST$:+TEMP$
0060 ENDWHILE
0070
0080 PRINT TEKST$
0090
0100 END "Programmet er slut"
0110
0120 DATA "Dette er en streng på
     80 tegn, som skal "
0130 DATA "printes på skærm og læ-
     ses fra datasætning"
```

Linie 30 i begge eksempler:
WHILE NOT End Of Data DO
Altså - så længe vi ikke er nået til enden af data (datasætninger) gør følgende:
1. eks. læs data, print data
2. eks. læs data, adder til tekst.

C-B-N

TIPSKUPON

Programmet udskriver på skærmen 9 tipsrækker.

Forinden spørges der for hver kamp om, hvor stor procentchancen er for hjemmesejr (0 - 100) og derefter for uafgjort (igen 0 - 100).

Programmet tager ikke hensyn til, om et af procenttallene eller deres sum overstiger 100.

I linie 220 tildeles den enkelte kamp et tal mellem 0 og 100.

I linierne 230 og 250 sammenlignes dette tal med de indtastede sandsynligheder, og sammenligningen afgør, om kampens resultat skal være et ettal, et kryds eller et total.

Bemærk, hvorledes linierne 240, 260 og 280 successivt opbygger en streng af ni udfald for hver af de tretten kampe.

Proceduren "udskriv" i linie 330 - 370 klarer udskriften af de 13 strenge.

Ønskes udskrift på printer, indsættes en linie:

```
0335 SELECT OUTPUT "LP"
```

Og for god ordens skyld bør der også tilføjes en linie:

```
0365 SELECT OUTPUT "DS"
```

S.J.

```
0010 // list "1.tips"
0020 DIM tegn$(13) OF 40
0030 DIM hjem(13), uafg(13)
0040 PRINT CHR$(147); // slet skærm
0050 indtast'ods
0060 valg'tegn
0070 PRINT CHR$(147); // slet skærm
0080 FOR n:=1 TO 4 DO PRINT
0090 udskriv
0100 PROC indtast'ods
0110   FOR x:=1 TO 13 DO
0120     PRINT
0130     PRINT "Kamp nummer ",x
0140     PRINT
0150     INPUT "Hjemmesejr ": hjem(x)
0160     INPUT "Uafgjort   ": uafg(x)
0170   ENDFOR x
0180 ENDPROC indtast'ods
0190 PROC valg'tegn
0200   FOR x:=1 TO 9 DO
0210     FOR y:=1 TO 13 DO
0220       v:=RND(1,100)
0230       IF v<hjem(y) THEN
0240         tegn$(y):=tegn$(y)+ 1
0250       ELIF v<hjem(y)+uafg(y) THEN
0260         tegn$(y):=tegn$(y)+ x
0270       ELSE
0280         tegn$(y):=tegn$(y)+ 2
0290       ENDIF
0300     ENDFOR y
0310   ENDFOR x
0320 ENDPROC valg'tegn
0330 PROC udskriv
0340   FOR x:=1 TO 13 DO
0350     PRINT tegn$(x)
0360   ENDFOR x
0370 ENDPROC udskriv
```

C-B-N

FJERNE BLANKTEGN

Når man ønsker at fjerne blanktegn i slutningen af en streng, kan det gøres med proceduren vist nedenfor. Linierne 0010 - 0070 bruges til afprøvning af proceduren. De skal slettes, inden proceduren indgår i et program.

Der er tale om en såkaldt rekursiv procedure. Det vil sige en procedure, der kalder sig selv indefra. I tilfældet her sker det fra linie 0110.

Det, der sker, er følgende: Med dimensioneringerne i linie 0010 kan strengvariablen blive 30 pladser lang. Måden at skrive variabelnavnet på i linie 0020 medfører, at strengen altid vil blive disse 30 pladser lang, uanset hvad man skriver i inputsætningen.

Skriver man f. eks. "Hans", vil tekststrengen bestå af "Hans" efterfulgt af 26 blanktegn. Det er praktisk, at længden af strenge på forhånd er kendt, nåtr de skal udskrives på skærm eller papir. Herved er det nemt at opnå en pæn opstilling.

Det er imidlertid ikke særlig praktisk at skrive "Hans Petersen" i en adresse, hvor der indgår 26 tomme pladser mellem "Hans" og "Petersen". Det er ligeledes dårlig programmering, hvis man i stedet skriver "Petersen, Hans". Det ville uden videre kunne lade sig gøre, hvis længden til efternavnet netop var på 8 pladser.

Når proceduren kaldes fra linie 0040, spørges der i linie 0090, om det sidste felt er CHR$(32), svarende til et blanktegn. Hvis det ikke er tilfældet, er opgaven løst.

I modsat tilfælde afskæres det sidste felt i strengen, som jo altså var et blanktegn. Det sker i linie 0100. Der står:

tekst$ = tekst$ minus det sidste felt.

Herefter er "Hans" blevet til en streng på 29 pladser.

I linie 0110 kaldes proceduren igen. Denne gang inde fra sig selv. Der er altså ingen forskel på, om kaldet kommer udefra eller indefra selve proceduren.

Næste trin: Der sker nøjagtigt det samme som før. Der testes om, hvorvidt sidste felt er et blanktegn. I bekræftende fald fjernes det o.s.v. Når strengen er 4 pladser lang, forlades proceduren, og derefter står der "Hans" uden blanktegn efter.

Rekursive procedurer er en smule vanskeligere at håndtere end de andre procedurer. Det betaler sig imidlertid at lære at bruge dem, fordi de sparer både tid og plads. Samtidig er det en elegant måde at løse adskillige problemer på.

Frank Hejndorf

```
0010 DIM navn$ OF 30, tekst$ OF 30
0020 INPUT "navn  > ": navn$(1:30)
0030 PRINT "<",navn$,">"
0040 strip'blank'tegn(navn$)
0050 PRINT "<",navn$,">"
0060
0070
0080 PROC strip'blank'tegn(REF tekst$)
0090   IF tekst$(LEN(tekst$):LEN(tekst$))=CHR$(32) THEN
0100     tekst$:=tekst$(1:LEN(tekst$)-1)
0110     strip'blank'tegn(tekst$)
0120   ENDIF
0130 ENDPROC strip'blank'tegn
```

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.

INDMELDELSESBLANKET

Undertegnede

Navn:.....

Adresse:
Gade:.....

Postnummer:..... By:.....

Att.:.....

Indmelder sig i COMAL BRUGER GRUPPE.

Kr. 90,00 vedlægges i check ....

indbetales pr. giro ....

Sæt venligst X