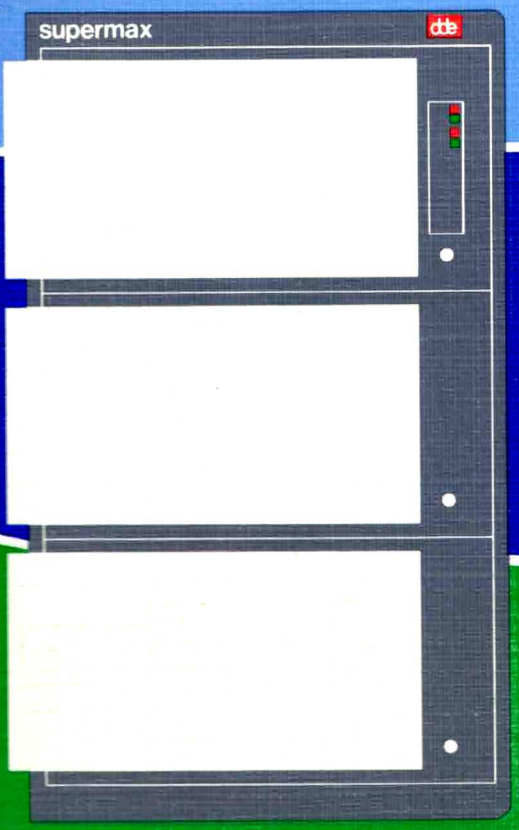
FINN CHRISTIANSEN
POUL KLAUSEN

COMAL 80
på Supermax



3. udgave

PROJEKTHÆFTE MED INSTRUKTIONER
OG OPGAVER

FORLAGET optima

Indholdsfortegnelse

3. udgave

telf. (97) 535580

ISBN 87 - 88662 - 62 - 4

Forord

Formålet med dette projekthæfte er at introducere brugen af COMAL-80 på Supermax-datamaten fra DDE på en let og overskuelig måde.

I fremstillingen er det tilstræbt at give en kortfattet og præcis beskrivelse af de mange muligheder, som dette programmeringssprog giver.

Indholdet dækker hele programmeringspensummet til højere handelseksamen, ligesom hæftet er velegnet som håndbog for merkonomer, enkeltfagselever og kursister på handelsskolens efteruddannelser.

I denne 3. udgave er indholdet ændret og skrevet om, så det præsenteres på en mere hensigtsmæssig måde. Samtidig er eksemplerne nu illustreret med moderne algoritme-strukturdiagrammer.

For at fremme indlæringen og overblikket er der i slutningen af hvert afsnit indsat repetitionsrammer og relevante øvelser.

Februar 1989

Forfatterne

1. Programmering på Supermax

Programmerne på en Supermax kan deles i 3 hovedgrupper:

Operativsystem (Unix)

Programmeringssprog

Applikationsprogrammer

Proggram-meringssprog

På maskinen findes der flere programmeringssprog, bl.a. COMAL-80, PASCAL, C, COBOL og ASSEMBLER.

Applikationer

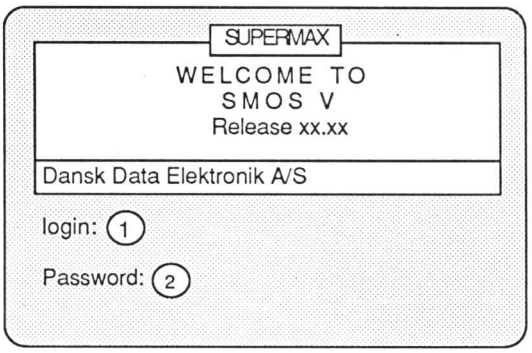
Applikationsprogrammer, der også kaldes brugerprogrammer, er f.eks. programmer til regnskab, tekstbehandling, statistik m.v.

For at kunne programmere på maskinen, skal der laves en **login** procedure.

Comal80 startes på Supermax således:

Login og Password

Tænd skærmen og tast **login** samt **Password** i startbilledet:



Forklaring til skærmbilledet:

1. Skriv dit brugernavn (login) efterfulgt af tryk på RETURN-tast (kaldes på nogle terminaltyper PAGE eller ENTER).

2. Skriv din adgangskode (password) efterfulgt af tryk på RETURN-tast. BEMÆRK, at indtastningen af sikkerhedsmæssige grunde er usynlig på skærmen.

Såfremt både login og password er indtastet korrekt, skifter systemet til det programmel, som det indtastede giver adgang til. I modsat fald skal indtastningen gentages.

Derefter skifter skærmbilledet til COMAL-80, som bl.a. giver en * i venstre side af skærmen. Hvert programmeringssprog har sit eget margintegn (prompt).

Så kan indtastningen af programmet starte.

Afslut

Når man ønsker at forlade COMAL-80 tastes STOP efterfulgt af tryk på RETURN-tasten

COMAL-80 programmer består af et antal *sætninger* (linier), der indledes med et *linienummer*. Desuden skal hver sætning nøje overholde en bestemt *syntaks*, d.v.s. bestemte måder at udtrykke sig på: bl.a. skal sætningerne være skrevet på engelsk. Desuden findes der en række *kommandoer* , der er instruktioner, man kan taste ind, for at få maskinen til at udføre bestemte handlinger.

Comal80

Syntaks

2. Struktureret programmering i COMAL-80

Ved hjælp af edb-programmer kan man løse mange beregnings- og udskriftsproblemer. Når et problem skal løses, må man først *strukturere* problemet og dets løsning, d.v.s. dele det op i mindre overskuelige dele, som systematisk kædes sammen.

Beskrivelsesmetoderne er mangfoldige og underkastet en vis form for mode. Vi har valgt at beskrive problemerne i dette hæfte ved hjælp af diagrammer, der ofte kaldes *algoritme-struktur-diagrammer*. Desuden forsynes alle diagrammer med tilsvarende COMAL-80 programlinier.

Numeriske variable

For at man kan regne med tal, skal der bruges *variable,* d.v.s. navngivne celler i edb-maskinen, hvor tegnene kan opbevares under kørslen.

Variabel

Variable skal have forskellige navne, som *skal* begynde med et bogstav, f.eks.

```
b7

sum
```

Generelt bør man vælge variabelnavne, der siger noget om, hvad de skal bruges til.

I det efterfølgende eksempel benyttes variabelnavnene: *antal, pris* og *sum.*

Da disse variable skal kunne rumme *tal* , kaldes de

Numerisk variabel

numerisk variable

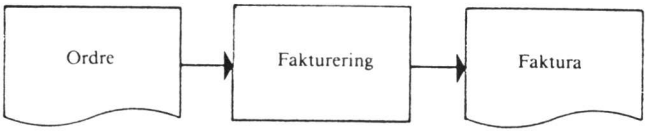
Før man begynder at indtaste et program, bør man altid skrive **NEW** efterfulgt af tryk på RETURN for at sikre, at der ikke ligger gamle programmer i sit brugerområde.

NEW

Eksempel 1

Eksempel 1

Lad os iagttage følgende lille system:



Strukturdiagram

```
LÆS ANTAL OG PRIS
SUM = ANTAL * PRIS
UDSKRIV SUM
```

Program

```
100 // Eksempel 1
110 INPUT antal,pris
120 sum:=antal*pris
130 PRINT sum
140 END
```

Forklaring:

//	linie 100	// betyder KOMMENTAR, d.v.s. at programmøren kan skrive, hvad han vil, uden at det betyder noget for programafviklingen. F.eks. kan man skrive programmets navn.
INPUT	linie 110	INPUT er en kommando, der betyder indlæsning til en eller flere variable. Variablerne hedder her *antal* og *pris*.
:=	linie 120	:= betyder *tildeling,* d.v.s. at variablen på venstre side af := tildeles værdien af udtrykket på højre side.
*		* betyder multiplikation. Tegnet kaldes for en regneoperator.
PRINT	linie 130	PRINT betyder *udskriv*
END	linie 140	END betyder, at programafviklingen skal stoppe.

Regneoperatorer

Regne-operator

I eksemplets linie 120 forekom tegnet *. Der findes andre regneoperatorer, som ofte benyttes i programmeringen. De almindelige matematiske regneregler gælder, d.v.s. at operatorerne har forskellig prioritet i den rækkefølge, de benyttes.

Følgende gælder:

Prioritet:	**Forklaring:**
1 ( )	Parentes
2 ↑	Potensopløftning
3 *	Multiplikation
3 /	Division
4 +	Addition
4 -	Subtraktion

Bemærk, at flere operatorer har samme prioritet. Hvis der forekommer operatorer med samme prioritet i en sætning, udføres de fra venstre mod højre.

Kørsel med det færdige program

Når programmet er indtastet, skal det kunne køre. Hvis der er syntaksfejl eller logiske fejl, meddeles det på skærmen, hvorefter fejlen rettes.

Typiske begynderfejl er, at nul forveksles med O (som i Ole) og at et-tal forveksles med l (som i Lise). Fejlene undgås bedst ved *altid* at benytte 10-tals tastaturet til højre, når det er tal, der skal tastes.

Programkørslen sættes igang ved at skrive

Start program

RUN

RUN

efterfulgt af tryk på RETURN-tasten.

Linie 110 vil resultere i et ? på skærmen, idet programmet jo beder om, at der indtastes antal og pris. Det kan være svært at huske, hvad der skal indtastes som svar. Derfor bør man benytte

Ledetekster i INPUT-sætninger.

Ledetekster

Linie 110 kan da skrives således:

110 INPUT "Indtast antal og stykpris ":antal,pris

Når programmet køres, fremkommer teksten der er skrevet mellem " " direkte på skærmen som ledetekst til operatøren. Programmet bliver derved mere *brugervenligt* .

BEMÆRK at der skal være et **:** mellem sidste anførselstegn og variabel-navnene.

Udskrivning af programmet

Udskriv

Ønskes programmet vist i sin helhed, skrives kommandoen

LIST
LI

LIST

Ønskes kun dele af programmet vist skrives følgende:

LIST 110	giver udskrift af linie 110
LIST 110,130	giver udskrift af linie 110 til 130
LIST 120,*	giver udskrift af programmet fra linie 120

Nye sætninger:	Forklaring:
//	Kommentarlinie
INPUT	Indlæsning til variable
:=	Tildeling
PRINT	Udskriv variabel og/eller tekst
END	Stop program

Nye kommandoer:	Forklaring:
NEW	Sletter gammelt program
RUN	Starter programkørsel
LIST	Udskrivning af programtekst
LIST X	Udskrivning af programtekstlinie nr. X
LIST X,Y	Udskrivning af do. fra linie X til linie Y
LIST Y, *	Udskrivning af do. fra linie Y til slut

Regneoperatorer:	Forklaring:
( )	Parentes - 1. prioritet
↑	Potensopløftning - 2. prioritet
* og /	Multiplikation og division - 3. prioritet
+ og -	Addition og subtraktion - 4. prioritet

Anvendelse af printer uden programsætninger

Kommandoerne RUN og LIST gav umiddelbart udskrift på skærmen.

Da der ofte ønskes udskrift på printer, introducerer vi allerede her en enkel måde at lave printerudskrift på af såvel programteksten som kørselsresultater.

Udskriv på printer

Der er 2 muligheder:

direkte udskrift

spool

I begge tilfælde skal der *reserveres en kanal til en printer*. Dette gøres ved hjælp af en såkaldt **OUTPUT** kommando.

Direkte udskrift betyder, at man venter, medens printeren udskriver medens *spool* betyder, at printet afleveres til en kø, der afvikles efterhånden som printeren bliver ledig.

Da Supermaxen er et flerbrugeranlæg, skal flere ofte deles om en eller få printere, hvorfor spool-print er velegnet, når ventetiden ønskes minimeret. Når brugerens print er afleveret til køen, kan der arbejdes videre med nye opgaver på skærmen. I denne bog har vi valgt udskrift på spool-printer.

Forbindelsen til printer nr. 0 (andre kan vælges) etableres ved at skrive efter stjernen i skærmkanten:

Spool

output spool -dprint0

eller

output spool -dprint0 -s

hvis man ikke ønsker udskriftskommentarer på skærmen.

Bemærk, at ***spool -dprint0*** *skal skrives med små bogstaver.*

På linien under skrives LIST eller RUN efter behov.

Nye kommandoer:	Forklaring:
output spool -dprint0	Udskrift på printer nr. 0 ved hjælp af spool-funktionen (printer-kø)
LIST	Programmet udskrives.
RUN	Kørselsresultaterne udskrives.

Øvelser

1. Skriv et program, der kan sammenlægge 3 tal samt udskrive gennemsnittet af dem. Indlæsning af de 3 tal skal ske ved en INPUT-sætning.

2. Beregn hvad følgende udtryk giver som resultat:

   (7*3+9)+(7*(3+9)) =

3. Skriv et COMAL80-program, der beregner og udskriver resultatet i øvelse 2.

4. Der ønskes konstrueret et system, der på grundlag af oplysninger om timetal og timelønsats kan beregne og udskrive løn ialt på skærmen.

   a. Beskriv problemet i et strukturdiagram.

   b. Lav et COMAL80-program, der kan løse problemet.

   c. Udskriv programteksten på printer (LIST).

   d. Udskriv løn ialt på printer.

5. Konstruer et system, der kan beregne totalbeløb på grundlag af antal enheder og pris pr. enhed. Til totalbeløbet skal der lægges 22% moms.

   Totalbeløbet excl. moms, momsbeløbet og totalbeløbet incl. moms skal udskrives.

   a. Beskriv problemet i et strukturdiagram.

   b. Lav et comal80 program, der løser problemet.

   c. Udskriv programmet på printer (LIST) og kør det bagefter med udskrift af resultaterne på printer (RUN).

6. Lav et system, der kan indlæse 4 eksamenskarakterer og finde gennemsnittet af dem. De 4 karakterer samt gennemsnittet skal udskrives.

   a. Beskriv problemet i et strukturdiagram.

   b. Lav et comal80 program, der løser problemet.

   c. Udskriv programmet på printer.

   d. Kør programmet. Resultatet skal udskrives på printer.

Alfameriske variable

Foruden numeriske variable, kan man også benytte variable, der skal kunne rumme *bogstaver og specialtegn*. Disse variable kaldes

alfameriske variable

Alfamerisk variabel

I modsætning til de numeriske variable, *bør* man fortælle systemet, hvor mange tegn, de alfameriske variable skal kunne rumme. Det gøres ved at *dimensionere* variablen i begyndelsen af programmet ved hjælp af **DIM**.

Det er dog ikke altid nødvendigt at dimensionere (erklære) en simpel strengvariabel i en DIM-sætning, idet en *ikke-erklæret* strengvariabel automatisk vil få tildelt længden 40 tegn.

DIM

Alfameriske variabelnavne skal altid begynde med et bogstav og *slutte med $*. Af samme grund kaldes de ofte for dollar-variable og kan f.eks. være:

$

a$

navn$

Eksempel 2

Eksempel 2

Vi opretter en alfamerisk variabel, der hedder navn$, som skal kunne rumme op til 30 tegn. I programmet vil det se således ud:

```
100 // Eksempel 2
110 DIM NAVN$ OF 30
120 INPUT "Indtast et navn   ": NAVN$
130 PRINT NAVN$
140 END
```

Forklaring:

linie 110	Variablen NAVN$ dimensioneres til max. 30 tegn.
linie 130	Indholdet i variablen NAVN$ udskrives.
BEMÆRK:	Man kan sikre sig, at der ikke kan skrives mere end de ønskede 30 tegn på følgende måde:

Eksempel 3

Eksempel 3

```
100 // Eksempel 3
110 DIM navn$ OF 30
120 EDIT "indtast et navn ": navn$
130 PRINT navn$
140 END
```

Forklaring:

EDIT

linie 120	Ved hjælp af EDIT låses feltet, så der ikke kan indtastes mere, end feltet er dimensioneret til.

I den videregående programmering kan denne facilitet benyttes i forbindelse med skærmlay-out, hvor felter udskrives på skærmen i forbindelse med EDIT-sætninger, der samtidig virker som INPUT-sætninger, hvor variabelværdien blot kan rettes istedet for at indtaste den påny. Især ved kartoteksvedligeholdelse er EDIT værdifuld.

En særlig facilitet ved de alfamerisk variable er, at man kan udtrække de enkelte tegn ved formen VAR$(x:y), hvor x er start- og y er slutposition for udtrækket (begge inkl.). Et sådant udtryk kaldes for en *delstreng*.

Eksempel 4

```
100 // Eksempel 4
110 DIM NAVN$ OF 12
120 NAVN$:= "Peter Hansen"
130 PRINT NAVN$(7:7)
140 PRINT NAVN$(1:5)
150 PRINT NAVN$(7:10)
160 END
```


Programmet giver følgende udskrift:

```
H
Peter
Hans
```

Nye sætninger:	Forklaring:
DIM	Dimensionering af alfameriske variable
EDIT	Benyttes i stedet for INPUT, hvis feltet skal låses, eller der skal kunne EDITERES (rettes) i en variabel, som systemet udskriver i et felt på skærmen.

Øvelser

7. Skriv et program der kan indlæse navn, adresse og by og bagefter udskriver oplysningerne.

8. Skriv et program der kan indlæse et varenavn, et antal og en stykpris, hvorefter antal og stykpris multipliceres. Varenavnet og resultatet af multiplikationen udskrives til sidst.

   Beskriv først forløbet i et strukturdiagram.

   Udskriv såvel programtekst som kørselsresultat på printer.

9. Lav et program der kan indlæse et personnavn på max. 25 tegn. Der skal i programmet indbygges sikkerhed for, at de 25 tegn ikke kan overskrides ved indtastningen.

3. Hjælpekommandoer ved programmering

Hjælpefunktioner

Under programmeringen er det af stor betydning, at man har forskellige hjælpefunktioner. De vigtigste er:

AUTO

REMOVE

EDIT

RENUMBER

AUTO
AU

AUTO er en hjælpekommando, der giver automatisk linienummerering. Kommandoen skrives efter stjernen i venstre skærmkant. Standardnummereringen starter med linie 100 og skifter med interval på 10. Såfremt andet linienummer ønskes som start skrives

AUTO xxx hvor xxx angiver første linienummer.

Ønskes intervaller, der afviger fra 10, skrives

AUTO x,y hvor x angiver startlinienummer og y intervallet mellem linierne.

Auto kan benyttes i forkortet form som

A U

ESC

AUTO afbrydes ved at trykke på ESC.

REMOVE
R

REMOVE benyttes til at fjerne overflødige linier i programmet. Kommandoen benyttes således:

REMOVE 100	betyder, at linie 100 fjernes.
REMOVE 100,120	betyder, at linie 100 til og med 120 fjernes.
REMOVE 120,*	betyder, at programmet slettes fra linie 120 og til slut. **(Benyt den med omtanke!)**

REMOVE kan benyttes i forkortet form som

R

Hvis

LREMOVE
LR

LREMOVE

benyttes, fjernes sætninger helt som ved brug af REMOVE. Den eneste forskel er, at de slettede linier listes (vises).

LREMOVE kan forkortes

LR

EDIT er en kommando, der benyttes, når programmøren ønsker at rette i programmet uden at skrive det hele om. Skrives

EDIT
E

EDIT	efter stjernen i skærmkanten kan man blade programmet igennem linie for linie og ved hjælp af *venstre og højrepilene* gå ind på den enkelte linie og rette eventuelle fejl. Efter hver linie tastes RETURN.
EDIT 120	betyder, at linie 120 kaldes frem til editering (rettelse).
EDIT 120,140	betyder, at linierne fra 120 til 140 kaldes frem en efter en til editering.
EDIT 120,*	betyder editering fra linie 120 til slut.

EDIT kan benyttes i forkortet form som

E

RENUMBER benyttes til at omnummerere et program med. Dette bruges ofte, hvis der er efterprogrammeret nogle linier med brudte numre. Så vil det se pænt ud, hvis springet i linienumrene er ens. Der skrives

RENUMBER
REN

RENUMBER	hvorefter hele programmet linienummereres om med 100 som første linie og med et spring på 10.
RENUMBER 10,5	programmet omnummereres til at starte med linie 10 med et spring mellem linienumrene på 5.

RENUMBER kan benyttes forkortet som

REN

Nye kommandoer:	Forklaring:
AUTO	Start af automatisk linienummerering. Afbrydes med ESC.
REMOVE	Fjerner overflødige linier i programmet, f.eks. REMOVE 100.
EDIT	Benyttes, når der skal rettes i programmet, f.eks. EDIT 120
RENUMBER	Benyttes til om-nummerering af linierne i programmet.

4. De ydre enheder.

Det er muligt at åbne og lukke printeren i selve programmet, så nogle sætninger udskrives på printeren, mens andre udskrives på skærmen.

Spool-printer

Ved benyttelse af spool-print, dvs. at udskriften automatisk skrives ud i en kø, hvis printeren er optaget, kan der arbejdes videre på skærmen, mens udskriften ligger i køen. Dette er selvsagt meget tidsbesparende.

Hvis udskriften ønskes dirigeret til spool-printer nr. 0 indsættes følgende linie i programmet:

Select output

xxx SELECT OUTPUT "spool -dprint0 -s"

BEMÆRK, at printerdefinitionen er indhyldet i anførselstegn. Når der afsendes udskrift til en spool-printer, giver systemet en meddelelse på skærmen. Denne kan undgås ved at tilføje -s som vist ovenfor.

-s

Når udskriften ønskes dirigeret tilbage til skærmen, skrives følgende linie i programmet:

#o

xxx SELECT OUTPUT "#o"

BEMÆRK, at der efter nummertegnet # er tale om et lille o (som i og).

Eksempel 5

Eksempel 5

Nedenstående program læser navn og adresse for en række personer og udskriver for hver person en label på printeren:

```
100 // Eksempel 5
110 DIM NAVN$ OF 30, ADR$ OF 30, BY$ OF 20
120 CLEAR
130 PRINT "Programmet indlæser en persons navn og adresse"
140 PRINT "og udskriver det på printer."
150 PRINT "Indtast en persons navn og adresse: "
160 INPUT "Navn              ":NAVN$
170 INPUT "Adresse           ":ADR$
180 INPUT "Postnr. og by     ":BY$
190 SELECT OUTPUT "spool -dprint0 -s"
200 PRINT
210 PRINT "ADRESSE-LABEL: "
220 PRINT
230 PRINT NAVN$
240 PRINT ADR$
250 PRINT BY$
260 PRINT
270 SELECT OUTPUT "#o"
280 PRINT "Udskriften er nu sendt til printer"
290 END
```

Forklaring:

linie 120	Skærmen renses for tegn.	
linie 130 - 150	PRINT-sætninger, der udføres på skærmen.	
linie 160 - 180	INPUT-sætningerne giver mulighed for indtastning af navneoplysninger.	
linie 190	SELECT OUTPUT-sætning, der dirigerer udskriften til spool-printeren. Alle efterfølgende PRINT-sætninger bliver nu udført på printeren.	**Vælg printer**
linie 200 - 260	Disse PRINT-sætninger bliver udført på printeren.	
linie 270	SELECT OUTPUT-sætning, der dirigerer udskriften tilbage til skærmen ved "#o"	**Vælg skærm**
linie 280	Denne meddelelse udskrives på skærmen.	

Gem og hent programmet fra disken

Hvis man har skrevet et program, kan man lagre programmet på disken ved kommandoen:

PUT peter

PUT (Gem)

Programmet bliver da gemt på disken under navnet "peter" og kan nu senere læses ind og køres pæny. Når programmet lagres med kommandoen PUT, gemmes det i "tekstform". Det går en smule langsommere end med SAVE, men til gengæld kan programmet normalt direkte overføres til evt. nye versioner af COMAL80-sproget.

Indlæsning af programmet fra disken sker ved kommandoen:

GET peter

GET (Hent)

De programmer, der er lagret på disken, forsvinder ikke, når man stopper terminalen.

Hvis man indlæser et program fra disken og retter i det, kan man være interesseret i at gemme programmet under det samme navn igen. Det gøres med kommandoen:

PUTOLD peter

Den gamle version af programmet bliver da slettet.

PUTOLD forkortes PUTO.

Et program kan også lagres med kommandoen:

SAVE (Gem)

SAVE peter

Er programmet gemt med SAVE, hentes det frem igen med kommandoen:

LOAD (Hent)

LOAD peter

Ligesom ved PUTOLD kan man også her gemme oveni. Det sker ved kommandoen:

SAVEOLD

SAVEOLD peter

SAVEO

SAVEOLD forkortes SAVEO.

Forskellen mellem PUT og SAVE er den form, som programmerne er gemt på ude på disken. Det tilrådes at man gemmer programmer med PUT-kommandoen, sålænge de ikke er færdigtestede. Programmer, der er gemt med SAVE, kan køres umiddelbart uden først at skrive LOAD ved blot at skrive kommandoen:

RUN

RUN peter

Slet program

Har man lagret et program på disken, kan det slettes igen med kommandoen delete. Ønsker man at slette programmet med navnet "peter" tastes:

DELETE (Slet)

DELETE peter

BEMÆRK at man kun kan slette et program, hvis brugeren har fået tillagt denne rettighed.

Kataloguds krift

Udskriv katalog

Man kan altid få udskrevet en oversigt over hvilke programmer, der er lagret på disken, med kommandoen:

DIR

DIR

Ønskes kataloget udskrevet på printer, benyttes følgende kommando:

output spool -dprint0
DIR

hvorefter udskriften kommer på spool-printer nr. 0.

Programudskrift på printer

Hvis det skrevne program ønskes udskrevet på printer, kan følgende fremgangsmåde anvendes:

output spool -dprint0
LIST

Et øjeblik senere udskrives programteksten på spool-printer nr. 0.

Øvelser

10. Indtast eksempel 5 og kør programmet. Gem derefter programmet på disken ved hjælp af PUT-kommandoen.

    Tast NEW og hent programmet ind igen med GET-kommandoen. Kør programmet.

    Tilføj sætning:

    255 PRINT "Programmet har været gemt på disken"

    Gem programmet igen ved hjælp af PUTOLD-kommandoen. Tast NEW og hent programmet ind igen med GET. Kør programmet pånyt og kontroller, at linie 255 nu er med.

11. Skriv et program, der indlæser to linier i to variable linie1$ og linie2$. Programmet skal efterfølgende udskrive de to linier på skærmen, dernæst på printeren og endelig på skærmen igen.

Nye sætr inger:	Forklaring:
SELECT OUTPUT "spool -dprint0 -s"	Valg af udskrift på printer nr. 0, som i dette tilfælde er en spoolprinter. -s betyder, at der ikke under udskriften gives meddelelser på skærmen.
SELECT OUTPUT "#o"	Udskriften sættes tilbage til skærmen.

Nye kommandoer	Forklaring:
PUT lise	Lagring på disken af programmet "lise"
GET lise	Læs programmet "lise" fra disken Skal være lagret med PUT-kommandoen.
SAVE lise	Lagring på disken af programmet "lise"
LOAD lise	Læs programmet "lise" fra disken. Skal være lagret med SAVE.
PUTOLD (PUTO)	Lagring af program under samme navn igen, hvorved den gamle version slettes.
SAVEOLD (SAVEO)	Lagring af program under samme navn igen, hvorved den gamle version slettes.
DELETE lise	Sletning på disken af programmet "lise"
DIR	Udskrift af kataloget. Hvis der først tastes OUTPUT SPOOL -dprint0 vil udskriften komme på printer 0.
LIST	Tilsvarende udskrives programteksten.

5. Formattering af udskrifter

Formattering af udskrifter udføres ved anvendelse af PRINT USING. Det giver mulighed for at få udskrevet tal i kolonner samt afrundet til et bestemt antal decimaler.

Formatteret udskrift

Eksempel 6

Eksempel 6

Programmet herunder indlæser et varebeløb og en rabatprocent. Herefter udskriver programmet varekøb, rabat, beløb minus rabat, moms og fakturabeløb. Ved anvendelsen af PRINT USING kommer tallene til at stå pænt under hinanden.

```
100 // Eksempel 6
110 CLEAR
120 INPUT "Indtast varebeløb      ":BELØB
130 INPUT "Indtast rabatprocent  ":PCT
140 RABAT:=BELØB*PCT/100
150 NETTO:=BELØB-RABAT
160 MOMS:=NETTO*0.22
170 FAKBELØB:=NETTO+MOMS
180 PRINT
190 PRINT USING "Varebeløb                    #####.##   ":BELØB
200 PRINT USING "Rabat                        #####.##   ":RABAT
210 PRINT USING "Varebeløb - rabat            #####.##   ":NETTO
220 PRINT USING "Moms                         #####.##   ":MOMS
230 PRINT USING "Fakturabeløb                 #####.##   ":FAKBELØB
240 END
```

Når programmet køres vil det med tilfældigt valgte tal give følgende lay-out:

Kørsel af programmet

```
Varebeløb                2089,34
Rabat                     146,25
Varebeløb - rabat        1943,09
Moms                      427,48
Fakturabeløb             2370,57
```

Forklaring:

Linie 110	CLEAR sletter skærmbilledet.
Linie 120-130	Indlæsning af varebeløb og rabat.
Linie 140-170	Beregning
Linie 190-230	Udskrivning ved anvendelse af PRINT USING sætningen. Tekstkonstanten efter USING kaldes *formatstrengen*. I tekstkonstanten er der anført et *formatfelt* bestående af #. Et formatfelt må kun bestå af sådanne tegn og evt. et decimalpunkt.

CLEAR

Formatstreng

Formatfelt

#

Efter formatstrengen er variablen anført. Når sætningen udføres, bliver værdien af variablen skrevet i formatfeltet, sådan at tallet *højrestilles*, samtidig med at der afrundes efter normale afrundingsregler til det antal decimaler, der er afsat plads til.

En formatstreng må indeholde *flere formatfelter*. Sætningen skal da afsluttes med et tilsvarende antal variable, adskildt af , (komma).

Nye sætninger:	Forklaring:
CLEAR	Sætning, der rydder skærmen.
PRINT USING	Sætning, der anvendes i forbindelse med formatteret udskrift.

6. Betingede programsætninger

Betingede programsætninger benyttes, hvor programmøren kun ønsker en aktivitet udført, *hvis* bestemte betingelser er opfyldt.

6.1. IF - THEN

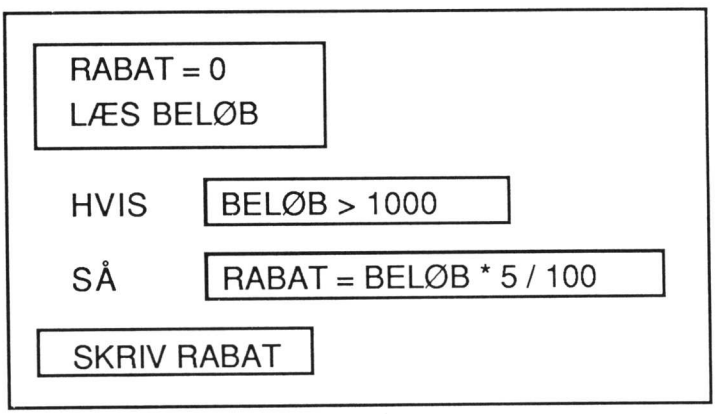
Eksempel 7
IF - THEN

Eksempel 7

Der ønskes beregnet 5% rabat af et varebeløb, hvis det er større end 1000 kr.

Problemet ser i blokdiagram og program således ud:

Strukturdiagram



Program

```
100 // Eksempel 7  IF/THEN
110 rabat:=0
120 INPUT "Indtast varebeløb ":beløb
130 IF beløb>1000 THEN rabat:=beløb*5/100
140 PRINT "Rabatten er ";rabat
150 END
```

Forklaring:

linie 130	Ved hjælp af IF, udføres sætningen kun, hvis udsagnet er sandt - altså hvis beløb er større end 1000. Ellers springes linien over.
linie 140	Indholdet af variablen 'rabat' udskrives med ledetekst.

I linie 130 anvendes tegnet >. Det er et eksempel på en *relations-operator*, der kan anvendes til at danne betingelser.

6.2. Relationsoperatorer

Der findes følgende relationsoperatorer:

Relations-operatorer

Tegn:	**Betydning:**
<	Mindre end
<=	Mindre end eller lig med
>	Større end
>=	Større end eller lig med
=	Lig med
<>	Forskellig fra

6.3. IF - THEN - ENDIF

IF
THEN
ENDIF

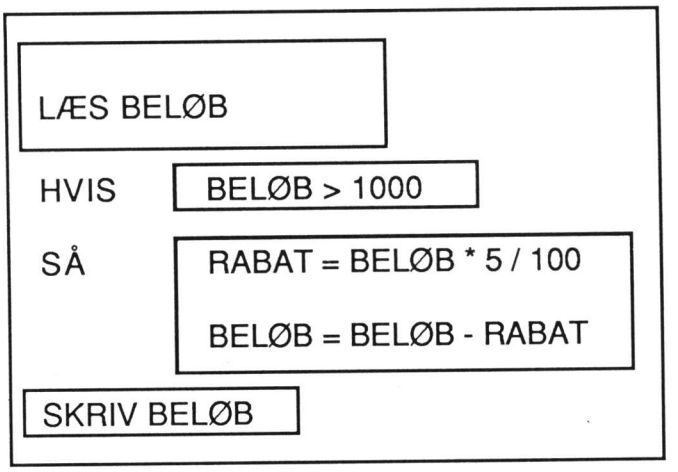
Det kan f.eks. være, at der på et varebeløb ydes rabat på 5%, hvis det overstiger kr. 1000,-. Varebeløbet udskrives med eller uden rabat.

Eksempel 8

Eksempel 8

Problemet kan beskrives og programmeres således:

Strukturdiagram



Program:

```
100 // Eksempel 8
110 // IF/THEN/ENDIF
120 INPUT "Indtast varebeløb  ":beløb
130 IF beløb > 1000 THEN
140     rabat:=beløb*5/100
150     beløb:=beløb-rabat
160 ENDIF
170 PRINT "Varebeløbet er  ";beløb
180 END
```

Forklaring:

linie 130	betyder, at kun hvis beløbet er større end 1000 udføres den efterfølgende del af programmet frem til ENDIF. Hvis betingelsen i linie 130 ikke er opfyldt, hopper systemet automatisk videre til linie 170 efter ENDIF.
linie 140	Variablen 'rabat' tildeles værdien af udtrykket på højresiden af :=
linie 150	Variablen 'beløb' tildeles værdien af udtrykket på højresiden, dvs. at beløb reduceres med rabat.

linie 160 ENDIF markerer, at IF-blokken slutter.

linie 170 Resultatet udskrives - her med ledetekst.

BEMÆRK: Hvis en IF-konstruktion ikke kan være på 1 linie, skal den starte med IF-sætningen og afsluttes med en ENDIF-sætning som i linie 160.

6.4. PRINT med ledetekst

I linie 170 blev anvendt PRINT med en ledetekst (også kaldet en tekstkonstant).

Såfremt ledetekst ønskes udskrevet, anføres teksten som vist mellem 2 anførselstegn således:

170 PRINT "Varebeløbet er ";beløb

PRINT med ledetekst

Det, der står mellem anførselstegnene, udskrives ordret igen.

BEMÆRK, at der efter sidste anførselstegn er anført et ; (semikolon). Derved udskrives den efterfølgende variabel lige efter ledeteksten.

6.5. PRINT med tabulering (TAB)

Såfremt der ønskes udskrift i en bestemt position på linien, der på skærmen har i alt 80 tegn, benyttes funktionen:

PRINT med TAB(x)

TAB(x)

hvor x angiver i hvilken position udskriften skal begynde. TAB(x) skal stå i en PRINT-sætning.

Linie 170 foran kan skrives således, hvis udskriften f.eks. skal begynde i position 20 på linien:

170 PRINT TAB(20),"Varebeløbet er ";beløb

6.6. IF - THEN - ELSE - ENDIF

IF THEN ELSE ENDIF

I et program kan der være behov for at undersøge flere betingelser. Lad os se på et lille eksempel:

Eksempel 9

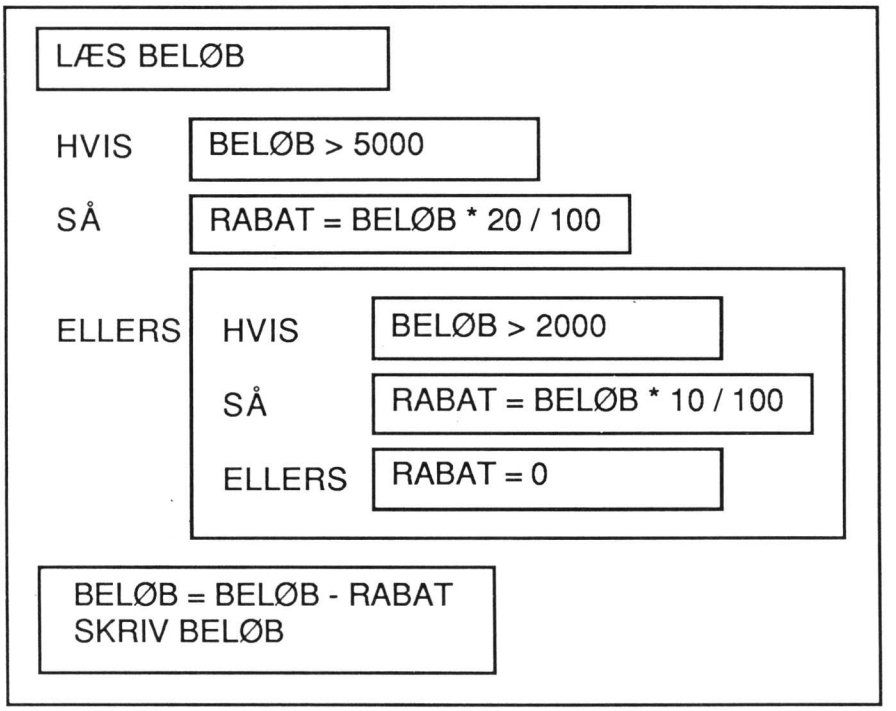
En virksomhed yder rabat på varebeløb efter følgende skala:

Ved køb over 5000 kr. ydes 20% rabat
Ved køb over 2000 kr. ydes 10% rabat
Ved mindre køb yder der ingen rabat.

Nettobeløbet ved hvert køb ønskes udskrevet.

Problemet kan løses således:

Strukturdiagram



Program

```
100 // Eksempel 9
110 // if/then/else/endif
120 INPUT "Indtast varebeløb ":beløb
130 IF beløb>5000 then
140     rabat:=beløb*20/100
150 ELSE
160     IF beløb>2000 THEN
170             rabat:=beløb*10/100
180     ELSE
190             rabat:=0
200     ENDIF
210 ENDIF
220 beløb:=beløb-rabat
230 PRINT "Nettobeløbet er ";beløb
240 END
```

Forklaring:

linie 130	Her undersøges, om beløbet er større end 5000. Hvis dette er tilfældet, udføres linie 140, hvorefter der hoppes til linie 220 efter sidste ENDIF.
linie 140	Udføres kun, hvis linie 130 er sand.
linie 150	ELSE betyder, at der hoppes til næste linie, hvis det foranstående udtryk ikke er sandt.
linie 160	Næste udtryk undersøges. Hvis det er sandt, udføres linie 170, hvorefter der hoppes til linie 220 efter sidste ENDIF.
linie 170	Udføres kun, hvis udtrykket i linie 160 er sandt.
linie 180	ELSE betyder, at der hoppes til næste linie, hvis det foranstående udtryk ikke er sandt.
linie 190	Udføres kun, hvis en af de foranstående betingelser ikke er sande.

6.7. IF - THEN - ELIF - ENDIF

Ovenstående program kan skrives i en lidt kortere form ved at benytte ELIF på én linie i stedet for ELSE/IF på to linier.

Eksempel 10

Programmet ser da således ud:

```
100 // Eksempel 10
110 // if/then/elif/endif
120 INPUT "Indtast varebeløb ":beløb
130 IF beløb > 5000 THEN
140 rabat:=beløb*20/100
150 ELIF beløb > 2000 THEN
160
170     rabat:=beløb*10/100
180 ELSE
190     rabat:=0
200
210 ENDIF
220 beløb:=beløb-rabat
230 PRINT "Nettobeløbet er ";beløb
240 END
```

IF
THEN
ELIF
ENDIF

For sammenlignelighedens skyld er de tomme linier medtaget. Disse kan fjernes med REMOVE.

Forklaring:

linie 150	Er nu en sammenskrivning af ELSE og IF, der samtidig betyder, at det ene ENDIF udelades.
linie 200	ENDIF er udeladt, da ELIF ikke skal afsluttes med ENDIF.

6.8. Logiske operatorer

Logiske operatorer

Foruden de tidligere omtalte relationsoperatorer, findes såkaldte *logiske operatorer,* nemlig

AND OR NOT

De kan med fordel benyttes i betingede sætninger, hvor der skal testes på flere udsagn i samme sætning.

AND

AND - logisk "og" - der betyder, at begge udsagn i en sætning skal være sande for at hele sætningsudsagnet er sandt.

OR

OR - logisk "eller" - der betyder, at hvis blot ét af udsagnene i sætningen er sand, vil hele sætningsudsagnet blive opfattet som værende sand.

NOT

NOT - logisk "ikke" (negation) - der betyder, at hvis udsagnet i sætningen er sandt, opfattes hele sætningsudsagnet som værende falsk - og omvendt.

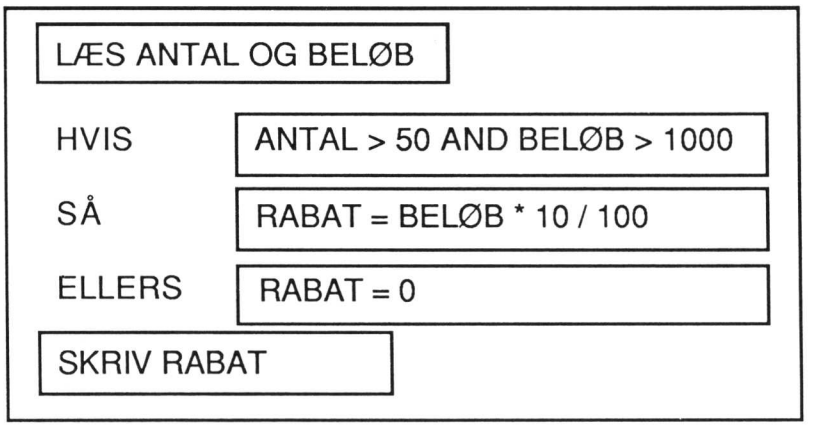
Eksempel 11

Eksempel med AND

En virksomhed yder 10% rabat til sine kunder, hvis de køber over 50 stk. pr. pr. gang *samt* køber for i alt over 1000 kr. Begge betingelser skal altså være opfyldt, for at der ydes rabat. Rabatbeløbet ønskes udskrevet.

Problemet kan løses således:

Strukturdiagram



Program

```
100 // Eksempel 11 - AND
110 INPUT "Indtast antal og beløb ":antal,beløb
120 IF antal>50 AND beløb>1000 THEN
130     rabat:=beløb*10/100
140 ELSE
150     rabat:=0
160 ENDIF
170 PRINT "Rabatten er kr. ";rabat
180 END
```

Forklaring:

linie 120	Kun hvis *både* antal>50 *og* beløb>1000 udføres linie 130. Ellers udføres linie 150.

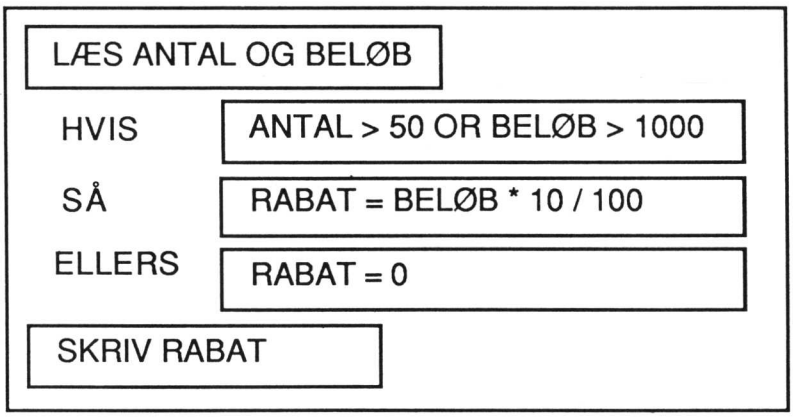
Eksempel 12

Eksempel med OR

Virksomheden yder nu i stedet for 10% rabat til sine kunder, hvis de køber *enten* over 50 stk. pr. gang *eller* køber for i alt over 1000 kr. Blot den ene af forudsætningerne er opfyldt, ydes der altså rabat.

Problemet kan beskrives således:

Strukturdiagram



Program

```
100 // Eksempel 12 - OR
110 INPUT "Indtast antal og beløb ":antal,beløb
120 IF antal>50 OR beløb>1000 THEN
130     rabat:=beløb*10/100
140 ELSE
150     rabat:=0
160 ENDIF
170 PRINT "Rabatten er kr. ";rabat
180 END
```

Forklaring:

linie 120 — Hvis *enten* antal>50 *eller* beløb>1000, udføres linie 130. Ellers udføres linie 150.

Eksempel 13

Eksempel med NOT

```
100 // Eksempel 13 - NOT
110 a:=11
120 IF NOT a=12 THEN
130     PRINT "Tallet a er <> 12"
140 ENDIF
150 END
```

I det foranstående eksempel, hvor der skulle købes mere end 50 stk. pr. gang og samtidig for over 1000 kr. kunne udsagnet skrives med NOT således:

120 IF NOT (antal<=50 OR beløb<=1000) THEN

BEMÆRK: NOT vender både relations- og logiske operatorer !

Sætningen svarer helt til

120 IF antal>50 AND beløb>1000 THEN

Nye operatorer:	Forklaring:
<	Mindre end
>	Større end
<=	Mindre end eller lig med
>=	Større end eller lig med
=	Lig med
<>	Forskellig fra
AND	Logisk operator - og
OR	Logisk operator - eller
NOT	Logisk operator - negation

Nye funktioner:	Forklaring:
TAB(x)	Funktion, der anvendes til at tabulere til en bestemt position på linien, inden udskrift

Nye sætninger:	Forklaring:
IF/THEN	Betinget sætning på én linie
IF/THEN/ENDIF	Betinget sætning
IF/THEN/ELSE/ENDIF	Betingede sætninger
IF/THEN/ELIF/ENDIF	Betingede sætninger
ENDIF	Slutlinie i en IF-konstruktion

Øvelser

12. I en virksomhed aflønnes sælgere, der har solgt for over 100.000 kr. pr. måned med et løntillæg oven i grundlønnen på 5% af omsætningen. Totallønnen udskrives.

    Beskriv problemet i et strukturdiagram, og skriv derefter et program, der på grundlag af indlæst salgsbeløb pr. måned og sælgerens grundløn selv finder ud af, om sælgeren skal have tillægget på de 5%.

13. I en anden virksomhed aflønnes sælgerne mere gradueret, idet sælgere, der har solgt for over 200.000 kr. pr. måned, får et tillæg på 15% af månedens salg, sælgere, der har solgt for over 100.000 kr. pr. måned, får 10% i tillæg, sælgere, der har solgt for over 50.000 kr. pr. måned, får 5% tillæg, medens resten ikke får noget tillæg til grundlønnen.

    Beskriv problemet i et strukturdiagram og skriv derefter et program, som på grundlag af indlæst salgsbeløb og grundløn beregner sælgerens samlede månedsløn. Grundløn, tillæg samt den samlede månedsløn skal udskrives.

14. På en skole gives point efter flid. Hvis man opnår over 50 point er man "særdeles flittig", over 25 point "flittig" - ellers er man "doven".

    Beskriv problemet i et strukturdiagram, og skriv derefter et program, der på grundlag af et indlæst pointtal kan udskrive en af de 3 flidsmeddelelser.

15. Posttaksterne for pakkeforsendelser i Danmark er følgende:

indtil 5 kg	24 kr.
over 5 kg indtil 10 kg	32 -
over 10 kg indtil 20 kg	55 -

    Tungere pakker forsendes på anden måde.

    Beskriv problemet i et strukturdiagram og skriv derefter et program, der på grundlag af indtastet vægt beregner posttaksten. Pakker over 20 kg skal afvises med en fejludskrift.

16. Til en eksamen skal der løses 5 opgaver. Der gives maksimalt følgende point for hver af de 5 opgaver:

opgave 1	10 point
opgave 2	20 point
opgave 3	15 point
opgave 4	15 point
opgave 5	40 point

Man har bestået prøven hvis det samlede pointtal er mindst 50, og der samtidig er opnået mindst 10 point i opgave 5.

Skriv et program, der indlæser de 5 point, beregner om eleven har bestået og udskriver teksten "Bestået" eller "Ikke bestået".

17. En virksomhed giver rabat til sine kunder afhængig af hvilken kundegruppe, de tilhører. Virksomheden opererer med følgende rabatgrupper:

Gruppe 1	2%	rabat
Gruppe 2	3%	rabat
Gruppe 3	5%	rabat
Gruppe 4	10%	rabat
Gruppe 5	15%	rabat
Øvrige:	0%	rabat

Beskriv problemet i et strukturdiagram og skriv derefter et program, der på grundlag af kundegruppe og salgsbeløb kan beregne rabat og fakturabeløb. Programmet skal opbygges med IF-THEN-ELSE-ENDIF konstruktioner.

6.9. CASE-konstruktion

CASE
OF
ENDCASE

Hvis der i et program optræder flere IF-sætninger inden i hinanden, kan man vælge en anden konstruktion, som ofte vil gøre programmerne lidt mere elegante. Konstruktionen kaldes en CASE-konstruktion og kan beskrives med følgende eksempel:

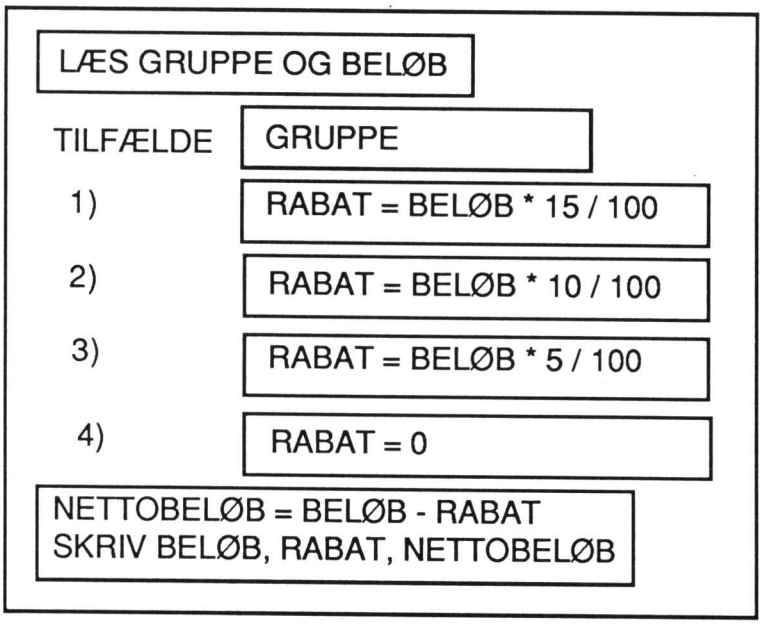
Eksempel 14

En virksomhed giver sine kunder forskellige rabatsatser ved køb afhængig af i hvilken kundegruppe, de hører hjemme. Der er følgende kundegrupper:

Gruppe 1	Der ydes 15% rabat
Gruppe 2	Der ydes 10% rabat
Gruppe 3	Der ydes 5% rabat
Gruppe 4	Der ydes ingen rabat

Problemet kan beskrives i et strukturdiagram og løses som vist i programmet nedenfor. Såvel beløb, rabat som beløb-rabat ønskes udskrevet.

Strukturdiagram



Program

```
100 // Eksempel 14 CASE
110 INPUT "Indtast kundegruppe og beløb ":gruppe,beløb
120 CASE gruppe OF
130 WHEN 1
140     rabat:=beløb*15/100
150 WHEN 2
160     rabat:=beløb*10/100
170 WHEN 3
180     rabat:=beløb*5/100
190 WHEN 4
200     rabat:=0
210 ENDCASE
220 nettobeløb:=beløb-rabat
230 PRINT "Varebeløb  ";beløb
240 PRINT "Rabat      ";rabat
250 PRINT "Nettobeløb ";nettobeløb
260 END
```

CASE
OF
WHEN
ENDCASE

Forklaring:

linie 120 Variablen *gruppe* undersøges for værdien ved hjælp af CASE *gruppe* OF

linie 130 WHEN 1 - Hvis *gruppe* er 1, udføres linie 140

linie 150 WHEN 2 - Hvis *gruppe* er 2, udføres linie 160

linie 170 WHEN 3 - Hvis *gruppe* er 3, udføres linie 180

linie 190 WHEN 4 - Hvis *gruppe* er 4, udføres linie 200

linie 210 CASE-konstruktionen afsluttes med ENDCASE, svarende til, at alle forgreninger i diagrammet samles igen.

linie 220 Variablen *nettobeløb* tildeles værdien beløb-rabat.

Hvis man vil sikre sig mod fejlindtastninger, kan der i linie 190 skrives

CASE
OF
WHEN
OTHERWISE
ENDCASE

190 OTHERWISE

Det får den betydning, at ikke blot kunder i gruppe 4, men også alle andre end fra 1 til 3 får rabat på 0%.

OTHERWISE kan med fordel benyttes til at sikre mod fejlindtastninger, idet man f.eks. kan lade programmet skrive "Fejl i inddata", hvis der ikke indtastes en godkendt værdi.

Nye sætninger:	Forklaring:
CASE	Multiforgrening med betingede sætninger i form af WHEN
WHEN	Betingelse i en CASE-konstruktion
OTHERWISE	Udføres, hvis der ikke i CASE-konstruktionen er fundet et sandt udsagn i en WHEN-sætning
ENDCASE	Slutmarkering i en CASE-konstruktion

Øvelser

18. I et forsikringsselskab betales der på autoforsikring en grundpræmie, der reguleres efter, hvilken præmiegruppe, bilejeren befinder sig i.

Grundpræmien er følgende:

Gruppe 1	kr. 1200
Gruppe 2	kr. 1500
Gruppe 3	kr. 2000
Gruppe 4	kr. 2600
Øvrige	kr. 3300

Beskriv problemet i et strukturdiagram og skriv derefter et program med en CASE- konstruktion, der på grundlag af indtastet gruppenummer kan udskrive det rigtige præmiebeløb.

19. En virksomhed giver rabat til sine kunder afhængig af hvilken kundegruppe, de tilhører. Virksomheden opererer med følgende rabatgrupper:

Gruppe 1	2%	rabat
Gruppe 2	3%	rabat
Gruppe 3	5%	rabat
Gruppe 4	10%	rabat
Gruppe 5	15%	rabat
Øvrige:	0%	rabat

Beskriv problemet i et strukturdiagram og skriv derefter et program, der på grundlag af kundegruppe og salgsbeløb kan beregne rabat og fakturabeløb. Programmet skal opbygges over en CASE-konstruktion.

20. Et konkurrerende forsikringsselskab benytter et mere differentieret præmiesystem, idet helårspræmien danner grundlag for den endelige præmieberegning, der forhøjes med 10%, hvis forsikringstager ikke har selvrisiko. Desuden forhøjes præmien, hvis der betales i flere rater pr. år.

Grundpræmien pr. år udgør følgende:

Gruppe 1	kr. 3000
Gruppe 2	kr. 2600
Gruppe 3	kr. 2200
Gruppe 4	kr. 1800
Gruppe 5	kr. 1500
Øvrige	kr. 1200

Selvrisikokoden sættes til 0, hvis forsikringstageren ikke har selvrisiko. Ellers sættes koden til 1.

Betalingskoden skal være følgende:

1	Ved helårlig betaling
2	Ved halvårlig betaling
3	Ved kvartårlig betaling

Hvis der betales halvårsvis, forhøjes den fundne årspræmie med kr. 50,- medens årspræmien forhøjes med kr. 100,-, hvis der betales kvartalsvis.

Der skal til sidst udskrives en opkrævning med oplysning om den samlede årspræmie, samt oplysning om der betales helårlig, halvårlig eller kvartårlig.

Udarbejd et strukturdiagram over problemet og skriv derefter et program, der kan udskrive de ønskede oplysninger på grundlag af indtastet gruppenummer, risiko- kode og betalingskode. Programmet skal indeholde CASE-konstruktioner i størst mulig omfang.

7. Løkkestrukturer

Løkke-strukturer

Programmering med løkkestrukturer betyder, at man laver programsætninger, der kan *gentages flere eller færre gange.* Vi skal i dette kapitel gennemgå følgende strukturer:

FOR - NEXT

WHILE - ENDWHILE

REPEAT - UNTIL

7.1. FOR - NEXT

Hvis antallet af gentagelser på forhånd *er kendt,* er FOR-NEXT-konstruktionen velegnet.

Lad os tage et lille eksempel.

Eksempel 15

FOR NEXT

Vi ønsker at lave en lille tabel fra 1 til 10, hvor tallet som resultat udskrives i 2. potens.

Program

```
100 // Eksempel 15 med FOR-NEXT
110 FOR x:=1 TO 10 DO
120     tal:=x*x
130     PRINT USING  "###":tal
140 NEXT x
150 END
```

Forklaring:

linie 110 I linien holdes regnskab med, hvor mange gange gentagelsen er sket. I den valgte variabel *x*, optælles antallet startende med 1 og stigende med 1 for hvert gennemløb, indtil det angiven sluttal - her 10 - er passeret. Når sluttallet passeres hoppes ud af løkken og videre med linien efter NEXT i linie 140.

STEP-værdien er fra starten altid 1, dvs. at der tælles opad med 1, hvergang FOR-sætningen passeres, men man kan angive en anden STEP-værdi, f.eks. ved at skrive

```
110 FOR x:= 10 TO 20 STEP 0.5 DO
```

STEP

I dette tilfælde vil tælleren starte med 10 og i hop på 0.5 pr. gennemløb fortsætte indtil 20 er passeret.

linie 120 Variablen *tal* tildeles værdien af x*x

linie 130 Variablen *tal* udskrives i formatet ###.

linie 140 NEXT betyder, at der vendes tilbage til FOR-sætningen, samt at tælleren - her *x* - forøges med STEP-værdien, som i eksemplet er lig med standardværdien 1.

Der kan være flere løkker inden i hinanden, også selv om det er af forskellig type.

Vigtig regel:

Hvis der optræder flere løkker inden i hinanden, skal de være opbygget som "kinesiske æsker", dvs. at den ene løkke skal være afsluttet, før man bevæger sig over i den næste.

Følgende eksempel vil belyse dette problem:

Eksempel 16

Vi vil nu lave et program, der kan lave en 10-tabel. En sådan tabel har som bekendt 2 koordinater, en x- og en y-koordinat.

Lad x være den variabel, der angiver søjlenummer, og y den, der angiver rækkenummer. Vi kan da skrive følgende program:

```
100 // Eksempel 16 - 10 tabel
110 FOR y:=1 TO 10 DO
120    FOR x:=1 TO 10 DO
130          PRINT USING "####":x*y;
140    NEXT x
150 PRINT
160 NEXT y
170 END
```

Hvis vi pynter lidt på denne model kan vi let udskrive en komplet tabel med såvel x- som y-angivelse og en overskrift.

Eksempel 17

Følgende program udskriver den lille tabel:

```
100 // Eksempel 17 - lille tabel
110 CLEAR
120 PRINT " D E N  L I L L E  T A B E L."
130 PRINT
140 FOR N:=1 TO 58 DO PRINT "-";
150 PRINT
160 PRINT TAB(12);"1   2   3   4   5   6   7   8   9   10"
170 FOR N:=1 TO 58 DO PRINT "-";
180 PRINT
190 FOR Y:= 1 TO 10 DO
200    PRINT USING "### !":Y;
210    FOR X:=1 TO 10 DO PRINT USING "####":Y*X;
220    PRINT
230 NEXT Y
240 FOR N:=1 TO 58 DO PRINT "-";
250 PRINT
260 END
```

KØRSEL

```
D E N  L I L L E  T A B E L.
```

		1	2	3	4	5	6	7	8	9	10
1	!	1	2	3	4	5	6	7	8	9	10
2	!	2	4	6	8	10	12	14	16	18	20
3	!	3	6	9	12	15	18	21	24	27	30
4	!	4	8	12	16	20	24	28	32	36	40
5	!	5	10	15	20	25	30	35	40	45	50
6	!	6	12	18	24	30	36	42	48	54	60
7	!	7	14	21	28	35	42	49	56	63	70
8	!	8	16	24	32	40	48	56	64	72	80
9	!	9	18	27	36	45	54	63	72	81	90
10	!	10	20	30	40	50	60	70	80	90	100

Forklaring:

linie 140 — Udskriver en hel linie med -. Når der i forbindelse med FOR-NEXT kun skal udføres en sætning, må konstruktionen skrives på samme linie. Der skal da *ikke* være nogen NEXT-sætning. Bemærk, at linien udskrives tegnvis.

linie 160 — TAB(12) betyder, at udskriften starter 12 tegn inde på linien.

linie 200 — PRINT USING-sætning, der udskriver venstre margin i tabellen. Der afsluttes med semikolon for at annullere linieskift.

linie 210 PRINT USING-sætning, der udskriver de enkelte linier i tabellen. Der afsluttes med semikolon for at annullere linieskift efter det enkelte tal. Linieskift skal først finde sted, når en hel linie er skrevet.

Bemærk, at *semikolon (;)* efter numerisk variable medfører, at der automatisk indsættes ét mellemrum mellem tallene. Anvendes i stedet *komma (,)* udskrives tallene i forlængelse af hinanden uden mellemrum.

linie 220 Tom PRINT-sætning, der giver linieskift.

7.2. WHILE-ENDWHILE

WHILE
ENDWHILE

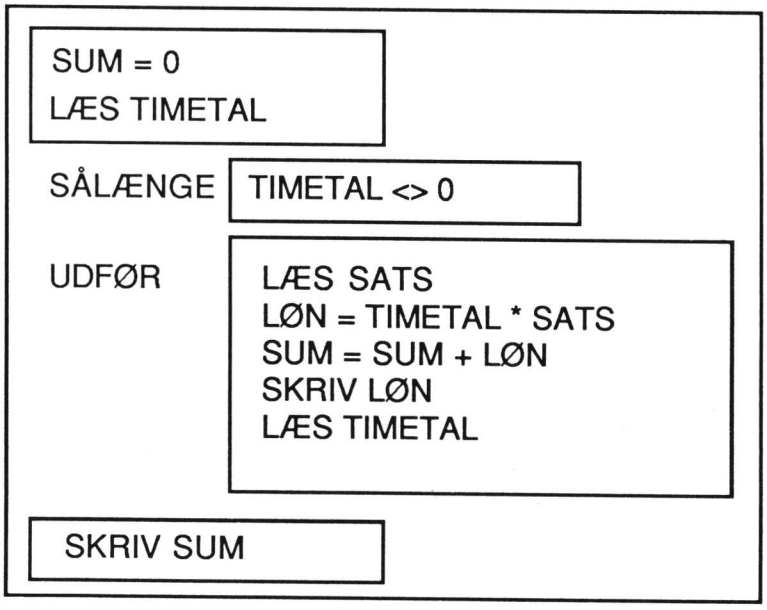
Hvis antallet af gennemløb *ikke* på forhånd er givet, bør man vælge en anden konstruktion, hvor der kan indbygges en *stopværdi*, så operatøren selv er herre over, hvornår programkørslen skal afbrydes.

Eksempel 18

I en virksomhed ønskes lavet et edb-program, der kan udskrive en række lønbeløb, beregnet på grundlag af et *antal timer* og en *timesats*. Beregningen skal kunne gentages, indtil der som *antal timer* indtastes en slutværdi, som her sættes til 0. Når stopværdien indtastes, skal systemet udskrive den totale lønsum.

Problemet kan beskrives og løses således:

Strukturdiagram



Program

```
100 // Eksempel 18 WHILE
110 sum:=0
120 INPUT "Indtast timetal ":tal
130 WHILE tal <>0 DO
140     INPUT "Indtast lønsats ":sats
150     løn:=tal*sats
160     sum:=sum+løn
170     PRINT USING "Løn i alt kr. ######.##":løn
180     INPUT "Indtast timetal ":tal
190 ENDWHILE
200 PRINT USING "Lønsummen er ######.## ":sum
210 END
```

Forklaring:

linie 130	WHILE-konstruktionen starter med at undersøge, om udsagnet er sandt - her om *tal* <> 0. Hvis det er tilfældet, gennemløbes løkken ned til ENDWHILE. Hvis *tal* er = 0, hoppes løkken over hvorefter systemet fortsætter med linien lige efter ENDWHILE.
linie 160	Her opsamles summen af de beregnede lønninger. Summen udskrives først i linie 200.
linie 180	INPUT-linie, der er nødvendig for af give variablen *tal* en ny værdi, før der returneres til WHILE fra linie 190.
linie 190	ENDWHILE betyder, at der returneres til WHILE-sætningen i linie 130.
linie 200	Her udskrives den opsamlede lønsum, efter at WHILE-løkken er forladt.

7.3. REPEAT - UNTIL

REPEAT UNTIL

En tredie type løkkekonstruktion er REPEAT - UNTIL. Den virker helt som den netop omtalte WHILE - ENDWHILE - konstruktion, men dog således, at *betingelsen først testes til sidst i konstruktionen.* Det betyder, at - modsat WHILE-konstruktionen - vil sætningerne i en REPEAT - UNTIL *altid* blive udført mindst én gang.

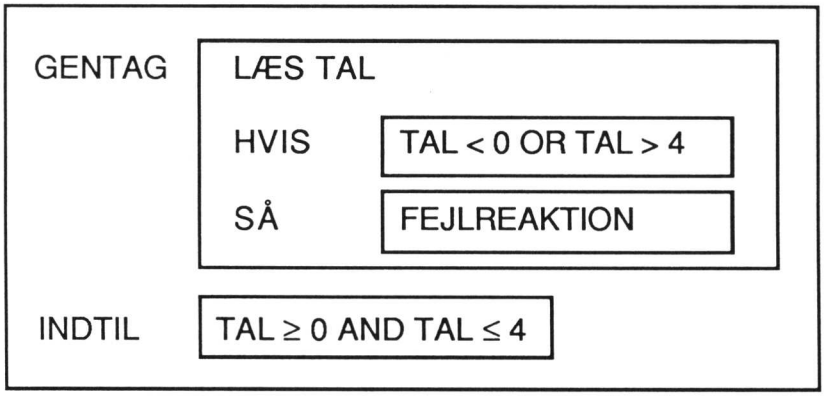
Princippet kan belyses med følgende lille eksempel:

Eksempel 19

Et program skal læse et tal, der skal være enten 0, 1, 2, 3 eller 4. Hvis der indtastes et tal forskellig herfra, skal indlæsningen gøres om.

Problemet kan illustreres og programmeres således:

Strukturdiagram



Program

```
100 // Eksempel 19
110 REPEAT
120     INPUT "Indtast et tal (0, 1, 2, 3, 4)  ":tal
130     IF tal <0 OR tal>4 THEN PRINT "Ulovlig værdi !"
140 UNTIL tal>=0 AND tal <=4
150 END
```

Forklaring:

linie 110 REPEAT-sætningen starter REPEAT - UNTIL konstruktionen.

linie 140 UNTIL-sætningen afslutter REPEAT - UNTIL konstruktionen. Efter UNTIL står en betingelse. Sætningerne mellem REPEAT og UNTIL udføres *indtil den opstillede betingelse bliver sand.*

Nye sætninger:	Forklaring:
FOR	Startordre i løkke med kendt antal gennemløb
NEXT	Slutordre i FOR-løkke
STEP	Angiver interval for optælling i FOR-sætning, hvis standardværdien på 1 ønskes fraveget.
WHILE	Startordre i en løkke med ukendt antal gennemløb
ENDWHILE	Slutordre i WHILE-løkke
REPEAT	Startordre i en løkke med mindst 1 gennemløb
UNTIL	Slutordre i REPEAT-løkke
STOP	Ubetinget stopsætning.

Øvelser

21. Skriv et program, der kan udskrive en momstabel til varebeløb fra kr. 10,- til kr. 100,- med interval på kr. 5,-, hvor momsen beregnes som 22% af varebeløbet.

    På hver linie skal beløbet uden moms, momsbeløbet samt beløbet incl. moms udskrives.

22. Skriv et program, der kan udskrive den store tabel fra 10 til 100 med interval på 10.

23. Til en benzinstation skal der skrives et program, der kan lave statistik over, hvordan dagens kunder fordeler sig på mænd og kvinder. Hvis der kommer en mand, indtastes koden 1. Er det en kvinde, tastes koden 2. Stopværdien sættes til 0.

    Beskriv problemet i et strukturdiagram og skriv derefter et program, der indeholder en WHILE - ENDWHILE konstruktion.

    Afprøv programmet på mindst 10 kodeværdier, som du på forhånd har valgt ud. Kontroller kørselsresultatet med dit forhåndsvalg.

24. En virksomhed ønsker at få fremstillet et edb-program til beregning af medarbejder- nes lønninger. Der skal som grundlag indtastes medarbejdernummer, antal timer og lønsats. Lønnen beregnes som antal timer * lønsats, og der skal efter hver beregning udskrives medarbejdernummer, antal timer, lønsats og løn i alt.

    Proceduren ønskes gentaget indtil medarbejdernummer indtastes med et 0 som stopværdi.

    Programmet skal slutte af med at udskrive den summen af de beregnede lønbeløb.

    Beskriv problemet i et blokdiagram, og skriv derefter programmet. Prøvekør evt. først programmet med skærmudskrift.

25. Skriv et program, der indeholder en REPEAT - UNTIL konstruktion, og som på grundlag af et indtastet tal udskriver tallet samt kvadratroden af tallet.

Rutinen skal kunne gentages, indtil der indlæses et negativt tal.

26. Skriv et program med en REPEAT - UNTIL konstruktion, der kan løse det i øvelse 23 beskrevne statistikproblem. Rutinen skal kunne gentages indtil der indtastes en stopværdi på 0.

27. Skriv et program, der kan udskrive en rentetabel for $(1 + r)^n$, hvor renten skal variere fra 1 til 8 procent med et interval på 1/4% og antallet af terminer fra 1 til 20.

28. HUSHØJ's administrative personale er månedslønnet, medens sælgerne er provisionslønnet samt får en mindre fast månedsløn.

Der ønskes konstrueret et program, der kan beregne og udskrive lønnen pr. medarbejder på en lønningsliste med følgende lay-out:

LØNNINGSLISTE

MED.NR.	NAVN	GRUNDLØN	PROVISION	MÅNEDSLØN
999999	xxxxxxxxxxxxxxx	9999.99	9999.99	99999.99
999999	xxxxxxxxxxxxxxx	9999.99	9999.99	99999.99
-	-	-	-	-
-	-	-	-	-
TOTAL		9999999.99	9999999.99	99999999.99

Inddata for hver medarbejder er følgende:

medarbejdernummer (stopværdi = -1)
kode (1 betyder ren månedsløn, 2 betyder sælger med provision og månedsløn.
navn (max. 20 tegn)
månedlig grundløn
salg (kr.) i måneden (kun hvis det er en sælger)
provisionsprocent (kun hvis det er en sælger)

Kørslen afsluttes som vist med udskrift af en totallinie.

Beskriv problemets løsning i et blokdiagram.

Skriv programmet.

8. Indicerede variable

Index

Indicerede variable anvendes til at definere et antal variable med samme navn. De enkelte variable skelnes fra hinanden ved hjælp af et *index*.

8.1. En-dimensionale talsæt

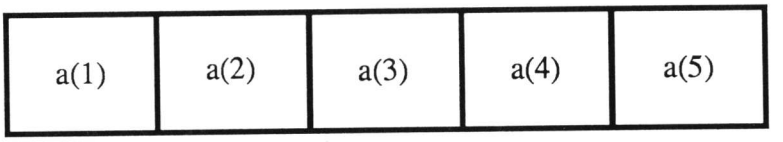
Vi vil først se på den variant, hvor variablen kun varierer i en enkelt dimension.

DIM a(5)

F.eks. skal vi til variablen *a* indlæse 5 tal. Variablen *a* skal da først dimensioneres, så den kan rumme de 5 tal. Dette gøres i en programsætning således

```
xxx DIM a(5)
```

Dette medfører, at der afsættes plads til 5 variable med navnene:



Indekset i parentesen er i sig selv en variabel, som kan benyttes i forbindelse med indlæsning af tal til variablen således:

Eksempel 20

```
100 DIM a(5)
110 FOR n:= 1 TO 5 DO
120     INPUT "Indtast et tal  ": a(n)
130 NEXT n
```

Ved hvert gennemløb skifter værdien i parentesen fra 1 op til 5, hvorfor de indlæste tal ved indlæsningen placeres i henholdsvis a(1), a(2), ... a(5).

Ønskes de indlæste tal senere udskrevet, kan de kaldes på tilsvarende måde:

```
140 FOR n:= 1 TO 5 DO
150     PRINT a(n)
160 NEXT n
170 END
```

Linie 100 til 170 kan nu køres som ét program.

Sådanne variable kaldes også *en-dimensionale*.

Eksempel 21

Følgende program indlæser 10 tal i en *dimensioneret variabel.* Derefter læses og udskrives tallene igen - her i formatteret form.

```
100 // Eksempel 21
110 DIM TABEL(10)
120 CLEAR
130 FOR n:= 1 TO 10 DO
140    PRINT USING "Indtast tal nr. ##: ":n;
150    INPUT "":TABEL(n)
160 NEXT n
170 PRINT
180 PRINT "Følgende tal er indtastet: "
190 PRINT
200 FOR n:= 1 TO 10 DO PRINT USING "########":TABEL(n)
210 END
```

Forklaring:

linie 100	Her oprettes en indiceret variabel med navnet TABEL. Til variablen knyttes 10 tal (celler), nemlig TABEL(1), TABEL(2), TABEL(3), ..., TABEL(10).
linie 120 - 150	FOR - NEXT løkke, der indlæser de 10 tal. Hvert tal læses i linie 140.
linie 190	FOR-sætning, hvor tallene udskrives igen, først indholdet af variablen TABEL(1), TABEL(2), ... , TABEL(10).

Eksempel 22

I Comal-80 findes forskellige *funktioner*, bl.a. RND, der som resultat giver et *tilfældigt tal.* I nedenstående program er denne funktion anvendt til at simulere kast med en terning.

RND

```
100 // Eksempel 22  - RND
110 DIM TERNING(6)
120 RANDOMIZE
130 FOR n:=1 TO 1000 DO
140    KAST:=RND(1,6)
150    TERNING(KAST):=TERNING(KAST) + 1
160 NEXT n
170 CLEAR
180 PRINT "Simulering af 1000 kast med terning: "
190 PRINT
200 PRINT "øjne        antal gange"
210 PRINT "_____________________"
220 FOR n:=1 TO 6 DO PRINT USING "#    ####":n,TERNING(n)
230 END
```

Forklaring:

linie 110	Der oprettes en indiceret variabel med plads til 6 tal.
RANDOMIZE linie 120	RANDOMIZE er en sætning, der betyder, at rækken af tilfældige tal bliver forskellig hver gang, programmet køres.
linie 140	RND-funktionen beregner et tilfældigt tal mellem 1 og 6, begge medregnet, hvorefter resultatet her tildeles variablen KAST.
linie 150	Bestemt af værdien af KAST forøges en af variablerne med 1. Hvis f.eks. KAST er 2, er det den indicerede variabel TERNING(2), der forøges med 1.

Eksempel 23

Indicerede alfameriske variable

Det er også muligt at abejde med *indicerede alfameriske variable*. Programmet herunder indlæser en dato i form af et helt tal, deler tallet i 3 dele og udskriver resultatet igen med månedens navn i stedet for.

```
100 // Eksempel 23
110 DIM MD$(12) OF 10
120 CLEAR
130 INPUT "Indtast en dato (ddmmåå)  ":DATO
140 DAG:=DATO DIV 10000
150 MDR:=(DATO DIV 100) - DAG*100
160 ÅR:=DATO MOD 100
170 FOR n:=1 TO 12 DO READ MD$(n)
180 PRINT
190 PRINT USING "d.##'  ":DAG;
200 PRINT MD$(MDR);
210 IF ÅR>=10 THEN
220   PRINT USING " 19## ":ÅR
230 ELSE
240   PRINT USING " 190# ":ÅR
250 ENDIF
260 // Månedsnavne til indlæsning i READ-sætning:
270 DATA "januar","februar","marts","april","maj","juni","juli"
280 DATA "august","september","oktober","november"
290 DATA "december"
300 END
```

Hvis der f.eks. indtastes

231089

bliver kørselsresultatet

d. 23' oktober 1989

Forklaring:

linie 110 — Her oprettes en indiceret alfamerisk variabel med plads til 12 alfameriske variable, nemlig MD$(1), MD$(2), ..., MD$(12), hver dimensioneret til et maksimalt indhold på 10 tegn.

linie 140 — DIV er en funktion, der udfører *heltalsdivision*. F.eks. er 75 DIV 9 = 8. **DIV**

linie 160 — MOD er en funktion, der angiver *rest ved en heltalsdivision*. F.eks. er 75 MOD 9 = 3. **MOD**

linie 170 — FOR-sætning, der ved anvendelse af READ tildeler den indicerede variabel MD$(n) i alt 12 månedsnavne som værdi. Disse værdier, der er *tekstkonstanter*, hentes fra DATA-sætninger i programmet (her linie 270 - 290). Første gang en READ-sætning udføres, hentes *første konstant* i første DATA-sætning. Næste gang en READ-sætning udføres, hentes næste konstant osv. **READ** **DATA**

linie 270 - 290 — Sætningerne indeholder de konstanter, der skal læses af READ-sætningen. Konstanterne kan stå i en eller flere DATA-sætninger. De vil altid blive læst i rækkefælge uanset hvor i programmet de er placeret.

Eksempel 24

Indicerede variable er velegnede ved *sortering af en talrække*. Herunder er vist et program, der sorterer tal i *stigende orden*.

```
100 // Eksempel 24, sortering af tal
110 CLEAR
120 // indlæsning
130 INPUT "Indtast antal tal ":n
140 DIM V(n)
150 PRINT
160 FOR x:=1 TO n DO
170     PRINT USING "Indtast tal nr. ### ":x;
180     INPUT "":V(x)
190 NEXT x
200 // sortering
210 FOR x:=1 TO n-1 DO
220     FOR y:=x+1 TO n DO
230         IF V(x) > V(y) THEN z:=V(x);V(x):=V(y);V(y):=z
240     NEXT y
250 NEXT x
260 // udskrivning
270 ZONE 20
280 PRINT
290 FOR x:= 1 TO n DO PRINT V(x),
300 PRINT
310 END
```

Forklaring:

linie 130	Her indlæses *antallet* af tal, der skal sorteres.
linie 140	V er en indiceret variabel, der oprettes til at kunne indeholde et antal tal, svarende til antallet der er indlæst i linie 130.
linie 160 - 190	Indlæsning af de tal, der skal sorteres. Under indlæsningen placeres tallene automatisk i en ny variabel V(n), hvor n angiver variablens index.
linie 210 - 250	Sorteringen udføres. Talrækken gennemløbes i et antal dobbeltløkker. I linie 230 testes størrelsesforholdet mellem to tal. Evt. ombytning af to tal finder sted i linie 230.
ZONE linie 270	ZONE-sætningen anvendes til at inddele udskriftslinien i kolonner. Her sættes kolonnebredden til 20. Hvis man vil anvende denne kolonnebredde, skal der anvendes *komma* som skilletegn i PRINT-sætninger.
linie 290 - 300	Den sorterede talrække udskrives.

8.2. To-dimensionale talsæt

Især ved opbygning og arbejde med tabeller kan det være fordelagtigt at arbejde med talsæt i flere dimensioner.

Forestil dig en tabel af denne type:

a(1,1)	a(1,2)	a(1,3)
a(2,1)	a(2,2)	a(2,3)

Variablen *a* skal da dimensioneres således:

```
xxx DIM a(2,3)
```

som giver i alt 2 * 3 = 6 variabler med de viste dobbeltindex.

Eksempel 25

Ved hjælp af dette eksempel vil vi lave en lille markedsundersøgelse om TV-sportsudsendelserne i henh. TV1 og TV2. TV1 resultaterne skal lagres i tabellens linie 1 og TV2's i linie 2. Der kan i analysen gives følgende svar: God (1), middel (2) eller dårlig (3).

Tabellen kan herefter illustreres således:

	God	Middel	Dårlig
TV1	a(1,1)	a(1,2)	a(1,3)
TV2	a(2,1)	a(2,2)	a(2,3)

Hver gang der afgives et svar fra en TV-seer, forøges den relevante variabel med 1.

Der er f.eks. indhentet følgende 11 udsagn fra forskellige TV-seere:

	TV-kanal	Bedømmelse
	1	1
	1	2
	2	1
	2	1
	2	3
	1	3
	1	1
	1	1
	1	2
	2	2
	2	1
Slutværdi:	0	0

Tallene kan indtastes og behandles i følgende program:

```
100 // Eksempel 25
110 DIM a(2,3)
120 INPUT "Indtast kanalnr. (1 - 2) og bedømmelse (1 - 3) ":n,m
130 WHILE n>0 THEN
140     a(n,m):=a(n,m)+1
150     INPUT "Indtast kanalnr. (1 - 2) og bedømmelse (1 - 3) ":n,m
160 ENDWHILE
```

I takt med indtastningen, opdateres tælleren i variablen a(n,m), hvor n,m styrer, i hvilken variabel, opdateringen finder sted, f.eks. i a(1,2), hvis seeren har givet TV1 karakteren "middel".

Resultatet udskrives i resten af programmet, der ser således ud:

```
160 // udskrivning af markedsanalysens resultat
170 CLEAR
180 PRINT "M A R K E D S U N D E R S Ø G E L S E"
190 PRINT
200 PRINT"   GOD          MIDDEL        DÅRLIG"
210 PRINT
220 FOR n:=1 TO 2 DO
230     FOR m:=1 TO 3 DO
240         PRINT a(n,m);
250     NEXT m
260     PRINT
270 NEXT n
280 END
```

Forklaring:

linie 110	Den to-dimensionale variabel *a* sættes til 2*3=6 celler.
linie 140	Den enkelte variabel forøges med 1. Variabelværdien bestemmes af *n* og *m*.
linie 240	Variablen a(n,m) udskrives i to FOR-NEXT løkker med række 1 (TV1) først.
linie 260	Giver linieskift efter udskriften af hele linie 1 (3 tal i alt).

Eksempel 26

På en skole har der været afholdt eksamen i fagene *dansk, tysk, engelsk, regnskab og matematik*. Skolen ønsker nu udskrevet en statistik, der viser, hvordan karaktererne fordeler sig på de enkelte fag. Karakterfordelingen kan i et program præsenteres ved hjælp af en to-dimensional indiceret variabel, dvs. en tabel. Her skal man burge et dobbelt index - et til rækkerne og et til kolonnerne.

```
100 // Eksempel 26 - karakterstatistik
110 DIM KAR(5), T(5,10), FAG$(5) OF 10
120 CLEAR
130 PRINT "dansk, tysk, engelsk, regnskab, matematik (STOP=-1) ";
140 INPUT "":KAR(1),KAR(2),KAR(3),KAR(4),KAR(5)
150 WHILE KAR(1) <> -1 DO
160     FOR n:= 1 TO 5 DO
170         IF KAR(n) = 0 THEN
180             T(n,1):=T(n,1) + 1
190         ELIF KAR(n) = 2 THEN
200             T(n,2):=T(n,2) + 1
210         ELIF KAR(n) = 13 THEN
220             T(n,10):=T(n,10) + 1
230         ELSE
240             T(n,KAR(n)-2):=T(n,KAR(n)-2) + 1
250         ENDIF
260     NEXT n
```

```
270     PRINT "dansk, tysk, engelsk, regnskab, matematik (STOP=-1) ";
280     INPUT "":KAR(1),KAR(2),KAR(3),KAR(4),KAR(5)
290 ENDWHILE
300 // udskrivning
310 SELECT OUTPUT "spool -dprint0 -s"
320 FOR n:= 1 TO 5 DO READ FAG$(n)
330 PRINT
340 PRINT "K A R A K T E R F O R D E L I N G"
350 PRINT
360 PRINT
370 PRINT "FAG  00  03  5  6  7  8  9  10  11  13"
380 FOR n:= 1 TO 60 DO PRINT "-";
390 PRINT
400 FOR n:= 1 TO 5 DO
410     PRINT FAG$(n);TAB(10);
420     FOR m:= 1 TO 10 DO PRINT USING "####"  ":T(n,m);
430     PRINT
440 NEXT n
450 SELECT OUTPUT "#o"
460 // fagdataoplysninger
470 DATA "Dansk","Tysk","Engelsk","Regnskab","Matematik"
480 END
```

Forklaring:

linie 110	Her dimensioneres i alt 3 forskellige indicerede variable. KAR er en almindelig en-dimensioneret numerisk variabel med plads til 5 tal. T er en to-dimensional numerisk variabel med plads til 5 * 10 tal = 50 tal. FAG$ er en alfamerisk indiceret variabel med plads til 5 tekstkonstanter hver af max. 10 tegns længde.
linie 140 - 290	Indlæsning af karakterer og opsummering af resultatet i den dobbelt indicerede variabel T. I linie 140 og 280 indlæses 5 karakterer i rigtig fagrække. FOR-NEXT løkken i linie 160 - 260 gennemløber de 5 indlæste karakterer og afgør, hvor i tabellen T der skal ske ajourføring af tælleren.
linie 300 - 440	Resultatet udskrives som en tabel på spool-printer .

Nye sætninger:	Forklaring:
DIM	Anvendes til at oprette en alfamerisk eller indiceret variabel.
DIM V(20)	Opretter en indiceret numerisk variabel med 20 tal
DIM T(10,8)	Opretter en 2-dimensional indiceret numerisk variabel med 10 x 8 = 80 tal
DIM A$(10) OF 20	Opretter en alfamerisk indiceret variabel til 10 tekster, hver med max. 20 tegn.
READ	Sætning, der anvendes til at tildele en variabel en værdi. Værdien hentes fra en DATA-sætning i programmet.
DATA	Sætning, der indeholder en række konstanter, der indlæses ved anvendelse af READ-sætningen.
RANDOMIZE	Sætning, der anvendes til at få tildelingen af tilfældige tal til at starte et vilkårligt sted, hver gang programmet udføres.
ZONE	Bruges til at ændre den ZONE, der benyttes ved print. ZONE er fra starten 0, men kan ændres f.eks. til 20 ved at skrive ZONE 20.

Nye funktioner:	Forklaring:
RND	Funktion, der bestemmer et tilfældigt tal. X := RND. X får en værdi mellem 0 og 1. X := RND(n). X får en hel værdi mellem 0 og n, begge medregnet. X := RND(m,n). X for en hel værdi mellem m og n, begge medregnet.
DIV	Heltalsdivision. A DIV B bestemmer det antal gange B går op i A. 17 DIV 3 = 5
MOD	Rest ved heltalsdivision. A MOD B bestemmer rest ved division af B op i A. 17 MOD 3 = 2

Øvelser

29. Skriv et program, der indlæser 20 tal i en indiceret variabel.
Programmet skal derefter udskrive tallene i omvendt rækkefølge.

30. Et program skal indeholde følgende DATA-sætninger:

DATA 6,8,12,67,8,3,78,9,12,23,56,67,2

DATA 9,11,13,93,4,7,5,53,15,78,4,19

Disse tal - 25 i alt - skal nu indlæses i en indiceret variabel TAL(25). Brug READ-sætning i programmet til indlæsningen.

Derefter skal programmet udskrive

det mindste tal
det største tal og
gennemsnittet af tallene.

Beskriv først problemet i et strukturdiagram. Skriv derefter programmet.

31. I forbindelse med en eksamen skal der udarbejdes et program, der tester om en elev har bestået. Programmet skal indlæses et antal karakterer. Eleven har bestået eksamen, hvis *gennemsnittet af karaktererne er mindst 5,5* og hvis samtidig *summen af de 2 laveste karakterer plus gennemsnittet af resten er mindst 13*.

Hvis eleven består udskrives:

BESTÅET, GENNEMSNIT xx.x

Ellers udskrives:

IKKE BESTÅET !

Beskriv opgaven i et strukturdiagram. Skriv derefter programmet.

Foretag derefter de nødvendige ændringer i programmet, så også elevens navn og fagene indlæses. Programmet skal udskrive et selvkonstrueret eksamensbevis, der indeholder navn, fag, karakterer og "BESTÅET, GENNEMSNIT xx.x" eller
"IKKE BESTÅET !"

Foretag endelig de nødvendige ændringer, så programmet ikke blot behandler resultatet for én enkelt elev, men for et vilkårligt antal elever.

32. For et antal kunder indlæses kundenummer og salgsbeløb. Den samme kunde kan forekomme flere gange.

Kundenummeret er 3-cifret, hvor første ciffer angiver distrikt:

1: Hovedstaden
2: Øerne øst for Storebælt
3: Øvrige Danmark
4: Udland

De to følgende cifre er et løbenummer.

Kundenumre med en forkert distriktskode skal afvises af systemet som fejl.

Der ønskes udskrevet en liste for hvert distrikt, der for hver kunde viser salgsbeløb i alt.

Til sidst udskrives en liste, der for hvert distrikt viser det totale salgsbeløb, samt en total for alle distrikter i alt.

Beskriv problemet i et diagram. Skriv derefter et program, der kan klare opgaven.

33. Et ølbryggerlaug ønsker gennemført en vurdering af årets produkter ved hjælp af edb. Medlemmerne har i årets løb færdigbrygget 4 nye ølsorter:

Volmers Grums
Grøftens Trøst
Mørk Slam
Troldvand

Man har nu udvalgt et panel af erfarne ølsmagere, der skal vurdere produkterne. Hver ølsmager giver for et produkt en af følgende karakterer:

fantastisk
god
middel
dårlig
udrikkelig

Efter bedømmelsen af et produkt afleverer en ølsmager en bedømmelsesseddel, der indeholder produktets navn og karakteren. Ølbryggerlauget ønsker nu til sin micro-computer skrevet et program, der efter indtastning af bedømmelsessedlerne kan udskrive en statistik som vist nedenfor.

En sådan statistik er en tabel med 6 rækker og 5 kolonner, hvorfor den bør indarbejdes i programmet som en 2-dimensional tabel. Resultaterne skal udskrives som procent-tal. Beskriv først problemet i et diagram. Skriv derefter programmet.

Statistikken skal præsenteres med følgende lay-out:

BEDØMMELSESSTATISTIK - ØLBRYGGERLAUGET

KARAKTER	ØLTYPE				
	Volmers Grums	Grøftens Trøst	Mørk Slam	Troldvand	I alt
fantastisk					
god					
middel					
dårlig					
udrikkelig					
I alt					

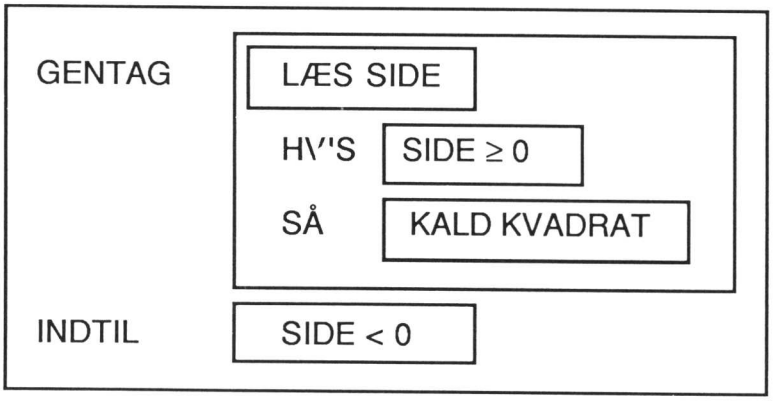
9. Procedurer

Når et program bliver stort, er det svært at overskue. Man kan dele programmet op i *underprogrammer*. Et underprogram kaldes en *procedure*. En procedure kan kaldes fra et andet sted i programmet. Når en procedure kaldes, udføres de sætninger, som proceduren består af. Når procedurens sætninger er udført, fortsættes med udførelsen af programmet umiddelbart efter den sætning, hvorfra proceduren blev kaldt.

Procedure

Eksempel 27

Nedenstående program læser siden i et kvadrat og beregner kvadratets areal. Selve arealet beregnes og udskrives i en procedure KVADRAT.



```
100 // program, der beregner arealet af en kvadrat
110
120 PROC KVADRAT
130    AREAL:= SIDE*SIDE
140    PRINT
150    PRINT "Kvadratets areal er:        ";AREAL
160    PRINT
170    ENDPROC KVADRAT
180
190 REPEAT
200    INPUT "Indtast kantlængden (STOP= -1):    ":SIDE
210       IF SIDE>=0 THEN EXEC KVADRAT
220 UNTIL SIDE<0
230 END
```

Forklaring:

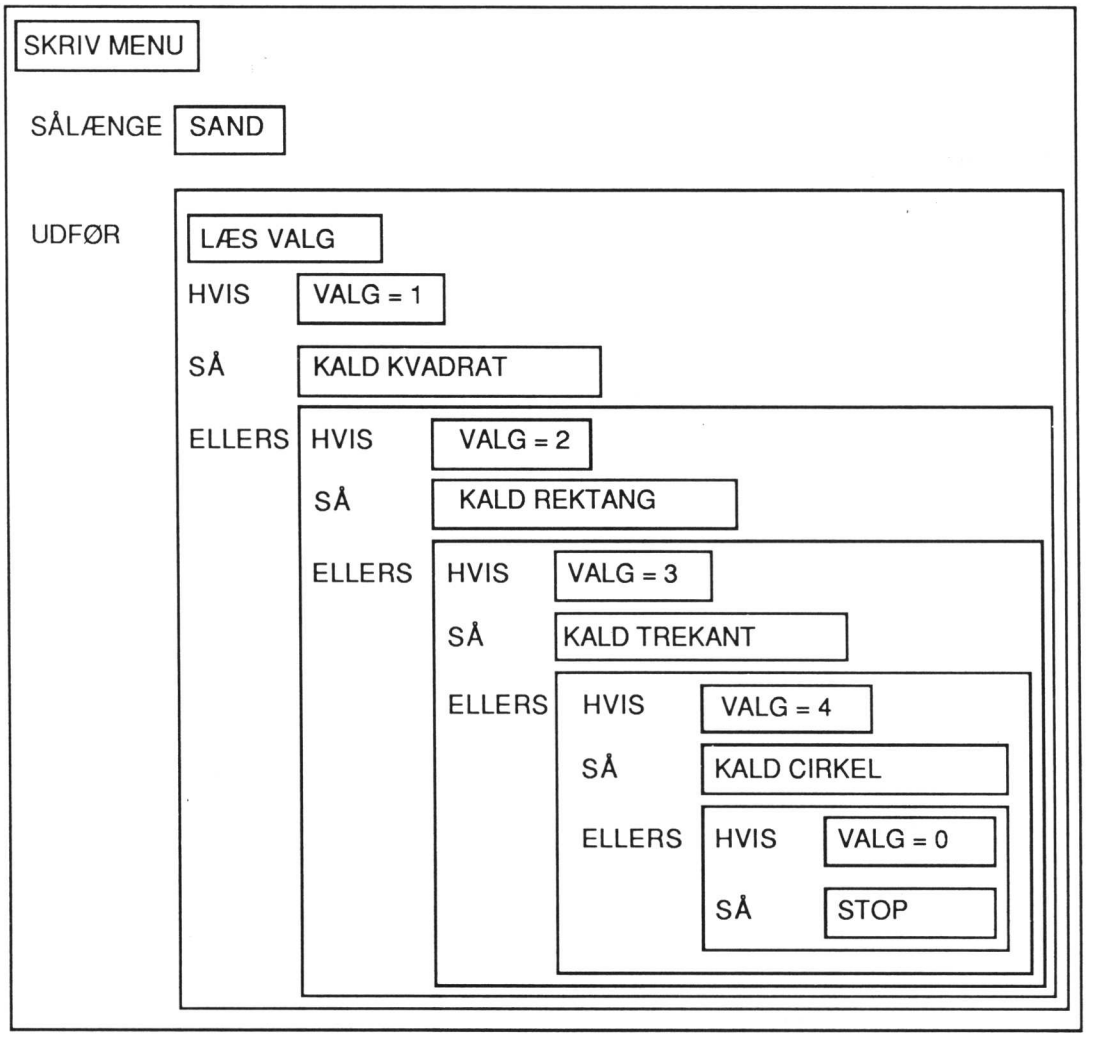
PROC	linie 120	PROC-sætningen, der fortæller hvor proceduren starter. I PROC-sætningen skal procedurens navn stå!
ENDPROC	linie 170	ENDPROC-sætningen afslutter proceduren. Her skal procedurens navn også stå.
EXEC	linie 210	Hvis sidelængden ikke er negativ kaldes proceduren. EXEC-sætningen kalder den procedure, hvis navn står efter EXEC.

En procedure består således af en eller flere sætninger, der er afgrænset af specielle sætninger PROC og ENDPROC. En procedure har et navn. Der gælder samme regler for navnet på en procedure som for navnet på en numerisk variabel.

Hvis et program har flere procedurer, skal alle *procedurenavnene være forskellige,* og de skal *også være forskellige fra programmets variabelnavne !*

Eksempel 28

Programmet herunder har fire procedurer. Hver procedure beregner arealet af en bestemt geometrisk figur. Hovedprogrammet udskriver en menu. Herfra vælges en af de 4 figurer, og afhængigt af dette valg kaldes en procedure.




```
100 DIM V$ OF 1
110 // Program, der beregner arealet af en kvadrat
120
130 PROC KVADRAT
140     CURSOR 10,18
150     INPUT "Indtast kantlængde:      ":SIDE
160     AREAL:=SIDE*SIDE
170     CURSOR 10,22
180     PRINT "Kvadratets areal er:     ";AREAL
190 ENDPROC KVADRAT
200
210 PROC REKTANG
220     CURSOR 10,18
230     INPUT "Indtast længde:          ":SIDE
240     CURSOR 10,19
250     INPUT "Indtast bredde:          ":BREDDE
260     AREAL:=SIDE*BREDDE
270     CURSOR 10,22
280     PRINT "Rektanglets areal er:    ";AREAL
290 ENDPROC REKTANG
300
```

```
310 PROC TREKANT
320     CURSOR 10,18
330     INPUT "Indtast grundlinie:          ":GRUND
340     CURSOR 10,19
350     INPUT "Indtast højde:               ":HØJDE
360     AREAL:=0.5*HØJDE*GRUND
370     CURSOR 10,22
380     PRINT "Trekantens areal er:         ";AREAL
390     PRINT
400 ENDPROC TREKANT
410
420 PROC CIRKEL
430     CURSOR 10,18
440     INPUT "Indtast radius:              ":RAD
450     AREAL:=2*RAD*3.1415926
460     CURSOR 10,22
470     PRINT "Cirklens areal er:           ";AREAL
480 ENDPROC CIRKEL
490
500 CLEAR
510 CURSOR 20,1
520 PRINT "A R E A L B E R E G N I N G"
530 CURSOR 25,3
540 PRINT "1. KVADRAT"
550 CURSOR 25,5
560 PRINT "2. REKTANGEL"
570 CURSOR 25,7
580 PRINT "3. TREKANT"
590 CURSOR 25,9
600 PRINT "4. CIRKEL"
610 CURSOR 25,11
620 PRINT "0. STOP"
630 CURSOR 30,13
640 PRINT "Vælg figur:"
650 CURSOR 25,15
660 PRINT "**************************"
670 WHILE TRUE DO
680     V$:=""
690     CURSOR 45,13
700     PRINT "      "
710     REPEAT
720         CURSOR 45,13
730         EDIT "":V$
740         IF (NOT V$ IN "12340") OR LEN(V$)=0 THEN PRINT CHR$(7);
750     UNTIL V$ IN "12340" AND LEN(V$)>0
760     CURSOR 1,18
770     PRINT CHR$(27)+CHR$(89)
```

```
780    CASE V$ OF
790    WHEN "1"
800       EXEC KVADRAT
810    WHEN "2"
820       EXEC REKTANG
830    WHEN "3"
840       EXEC TREKANT
850    WHEN "4"
860       EXEC CIRKEL
870    WHEN "0"
880       STOP
890    ENDCASE
900 ENDWHILE
```

Forklaring:

linie 500 - 600 — Her udskrives en funktionsmenu på skærmen, hvor man indtaster et valg for den figur, der ønskes behandlet.

linie 670 — Det er en WHILE-sætning, hvor betingelsen altid er sand. Kørslen afbrydes inde i løkken (sætning 880). **WHILE**

linie 740 — Sætningen tester om det indtastede funktionsvalg har en lovlig værdi.

IN

IN er en operator, der tester, om en tekst findes som en del-tekst i en anden tekst. Hvis det ikke er tilfældet, returnerer IN-operatoren værdien 0; ellers returneres positionen for det første tegn.

F.eks.

"per" IN "peter hansen" = 0

"hans" IN "peter hansen" = 7

LEN

LEN er en funktion, der bestemmer antallet af tegn i en tekst.

CHR$(7)

CHR$(7) giver alarm !

Nye funktioner	Forklaring
IN	Operator, der tester om en tekst forekommer i en anden tekst. F.eks. A$ IN B$ tester om indholdet af A$ forekommer som en delstreng i B$. Er det tilfældet, returneres positionen i B$ for det første tegn i A$; ellers returneres værdien 0.
LEN	Funktion, der bestemmer antal tegn i en tekst.
CHR$(7)	Giver alarm

Nye sætninger	Forklaring
PROC	PROC-sætningen indleder en procedure. Efter PROC skal prodecurens navn stå.
ENDPROC	Sætning, der afslutter en procedure. Efter ENDPROC skal procedurens navn stå.
EXEC	Sætning, der kalder en procedure. Efter EXEC skal procedurens navn stå. Når proceduren er udført, fortsætter programmet med udførelsen af første sætning efter EXEC.

Eksempel 29

Parameteroverførsel

Globale variable

Dette eksempel er identisk med eksempel 28. Der overføres kun kvadratets sidelængde som en parameter. Det er ofte uheldigt, at en procedure arbejder på *globale variable*. Hvis et program har mange procedurer, kan det være svært at gennemskue, hvornår en variabel ændres. Dette problem kan man løse ved at føre alle variable over som parametre.

```
100 // Program, der beregner arealet af en kvadrat
110
120 PROC KVADRAT(KANT)
130    AREAL:=KANT*KANT
140    PRINT
150    PRINT "Kvadratets areal er:       ";AREAL
160    PRINT
170 ENDPROC KVADRAT
180
190 REPEAT
200    INPUT "Indtast kantlængden (STOP = -1):          ":SIDE
210       IF SIDE>=0 THEN EXEC KVADRAT(SIDE)
220 UNTIL SIDE <0
230 END
```

Forklaring:

linie 120 Efter procedurenavnet er der en parameter KANT. Alle parametre skal stå i parentes efter procedurenavnet.
Disse parametre kaldes *formelle parametre*, og de får først en værdi, når proceduren kaldes.

Formelle parametre

linie 130 Arealet beregnes på grundlag af parameterens værdi.

linie 210 Proceduren kaldes. Efter procedurenavnet står en variabel. Værdien af denne variabel overføres ved kaldet til procedurens formelle parameter. De variable (parametre), der anvendes ved kaldet af en procedure i EXEC-sætningen, kaldes de *aktuelle parametre*.

Aktuelle parametre

Eksempel 30

Eksemplet her udfører det samme som ovenstående, men indlæsningen af sidelængden udføres af en procedure LÆSSIDE. Den har en parameter til sidelængden. Den er foranstillet ordet REF. Det betyder, at *værdien af den formelle parameter føres tilbage til den aktuelle parameter i EXEC-sætningen, når proceduren forlades*. Proceduren KVADRAT er helt identisk med ovenstående.

REF

```
100 // Program, der beregner arealet af en kvadrat
110
120 SIDE:=0
130
140 PROC KVADRAT(KANT)
150    AREAL:=KANT*KANT
160    PRINT
170    PRINT "Kvadratets areal er:       ";AREAL
180    PRINT
190 ENDPROC KVADRAT
200
210  PROC  LÆSSIDE(REF  SIDE)
220    INPUT "Indtast kantlængden (STOP = -1):          ":SIDE
230 ENDPROC LÆSSIDE
240
250 REPEAT
260    EXEC LÆSSIDE(SIDE)
270    IF SIDE>=0 THEN EXEC KVADRAT(SIDE)
280 UNTIL SIDE <0
290 END
```

Forklaring:

linie 210 Der er en formel parameter SIDE. Det er en REF-parameter, der betyder, at den værdi SIDE har, når proceduren afsluttes, tildeles til den aktuelle parameter i EXEC-sætningen (linie 260).

BEMÆRK:

Der er ingen forbindelse mellem navnet på den formelle parameter og den aktuelle parameter, dvs. at det er uden betydning om de hedder det samme eller har forskellige navne.

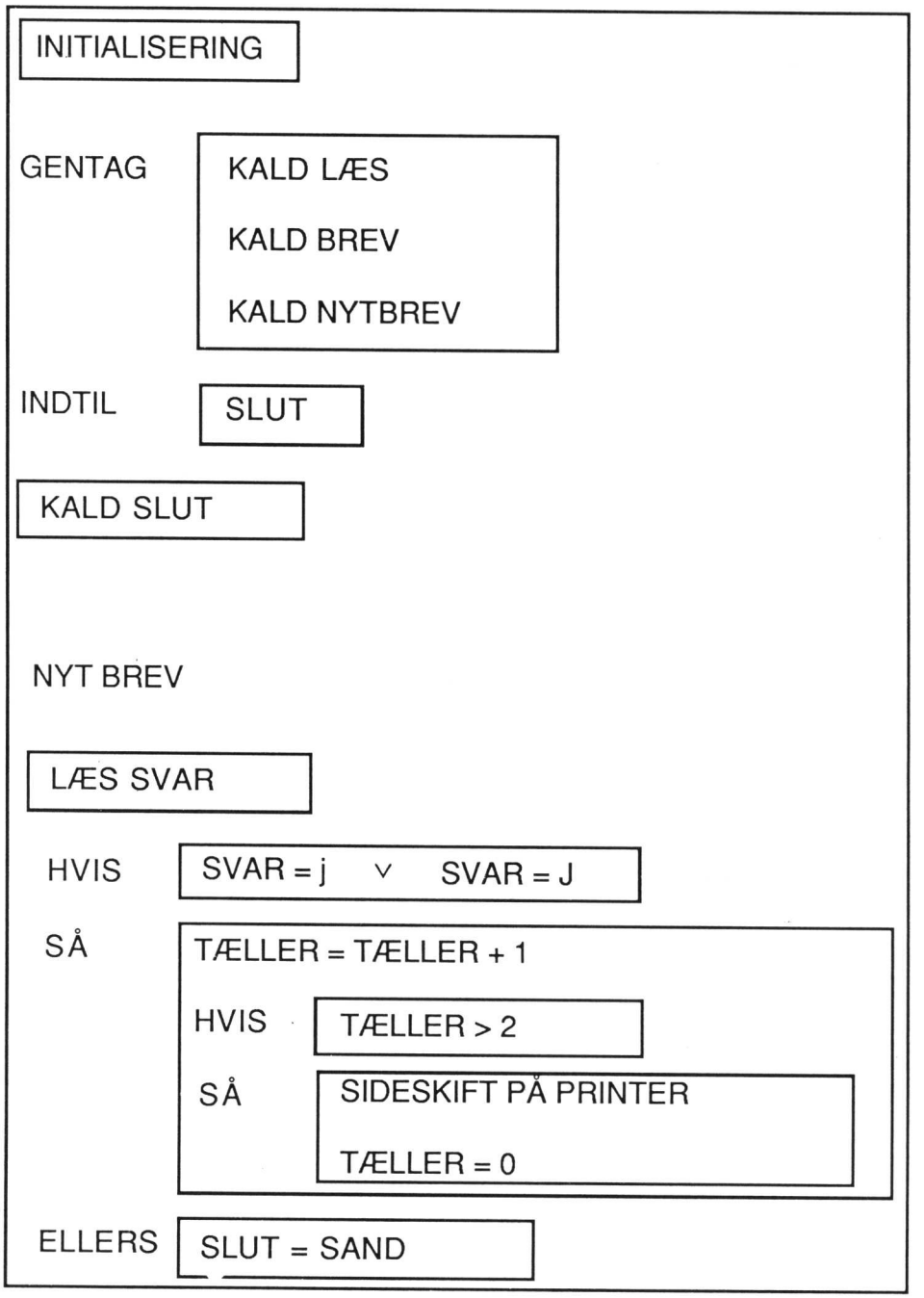
Lokal parameter

Den formelle parameter er *lokal* for proceduren og er derfor *ikke kendt uden for denne*.

Eksempel 31

Det følgende program udskriver nytårshilsener på printeren.

Eksempel


INITIALISERING
GENTAG
KALD LÆS
KALD BREV
KALD NYTBREV
INDTIL
SLUT
KALD SLUT
NYT BREV
LÆS SVAR
HVIS
SVAR = j ∨ SVAR = J
SÅ
TÆLLER = TÆLLER + 1
HVIS
TÆLLER > 2
SÅ
SIDESKIFT PÅ PRINTER
TÆLLER = 0
ELLERS
SLUT = SAND

```
100 // Eksempel
110
120 // Initialisering
130 DIM NAVN$ OF 30,ADR$ OF 30,POSTNR_BY$ OF 30,SVAR$ OF 1
140 SLUT:=FALSE
150 TÆLLER:=0
160 SELECT OUTPUT "spool -dprint0 -s"
170
180 PROC SKIFT // sideskift på printer
190     PRINT CHR$(12)
200 ENDPROC SKIFT
210
220 PROC NYTBREV(SV$,REF T,REF SLUT) // Spørger om der er flere
230     INPUT "er der flere breve (j/n) ?       ":SV$
240     IF SV$="j" OR SV$="J" THEN
250         T:=T+1
260         IF T>2 THEN
270             EXEC SKIFT
280             T:=0
290         ENDIF
300     ELSE
310         SLUT:=TRUE
320     ENDIF
330 ENDPROC NYTBREV
340
350 PROC BREV(NAVN$,ADR1$,ADR2$) // udskriver brev på printer
360     PRINT "Kære"
370     PRINT
380     PRINT NAVN$
390     PRINT ADR1$
400     PRINT ADR2$
410     PRINT
430     PRINT"*************************************************"
440     PRINT"*                                               *"
450     PRINT"*    Du ønskes hermed et godt nytår. Jeg hå-    *"
460     PRINT"*    ber på et fortsat godt samarbejde i det    *"
470     PRINT"*    nye år.                                    *"
480     PRINT"*                                               *"
490     PRINT"*                          Med venlig hilsen    *"
500     PRINT"*                               The Computer    *"
510     PRINT"*                                               *"
520     PRINT"*************************************************"
530     FOR N:=1 TO 5 DO PRINT
540 ENDPROC BREV
550
```

```
560 PROC LÆS(REF NAVN$,REF ADRES$,REF BY$) // indlæs navn mv
570     INPUT "Indtast navn:                ":NAVN$
580     INPUT "Indtast adresse:             ":ADRES$
590     INPUT "Indtast postnr. og by:       ":BY$
600 ENDPROC LÆS
610
620 // Hovedprogram
630 REPEAT
640     EXEC LÆS(NAVN$,ADR$,POSTNR_BY$)
650     EXEC BREV(NAVN$,ADR$,POSTNR_BY$)
660     EXEC NYTBREV(SVAR$,TÆLLER,SLUT)
670 UNTIL SLUT
680 EXEC SKIFT
```

Forklaring:

linie 180 - 200 Procedure, der anvendes til *sideskift* på printeren. CHR$(12) sender en styrekode til printeren, der giver sideskift.

CHR$(12)

linie 220 - 330 Procedure, der anvendes til at spørge, om der skal skrives flere breve. Er dette tilfældet, tælles tælleren T op. Den anvendes til at styre, om der skal skiftes side på printeren. Der kan være flere breve på samme side. Hvis der ikke skal skrives flere breve, sættes et "flag" i linie 310, der viser, at kørslen skal stoppe. Der testes på dette "flag" i linie 670.

linie 350 - 540 Procedure, der udskriver brevet på printeren.

linie 560 - 600 Procedure, der indlæser navn og adresse på modtageren. Bemærk, at alle parametre er REF-parametre, da værdierne jo skal føres tilbage til hovedprogrammet.

linie 640 Her kaldes procedure LÆS.

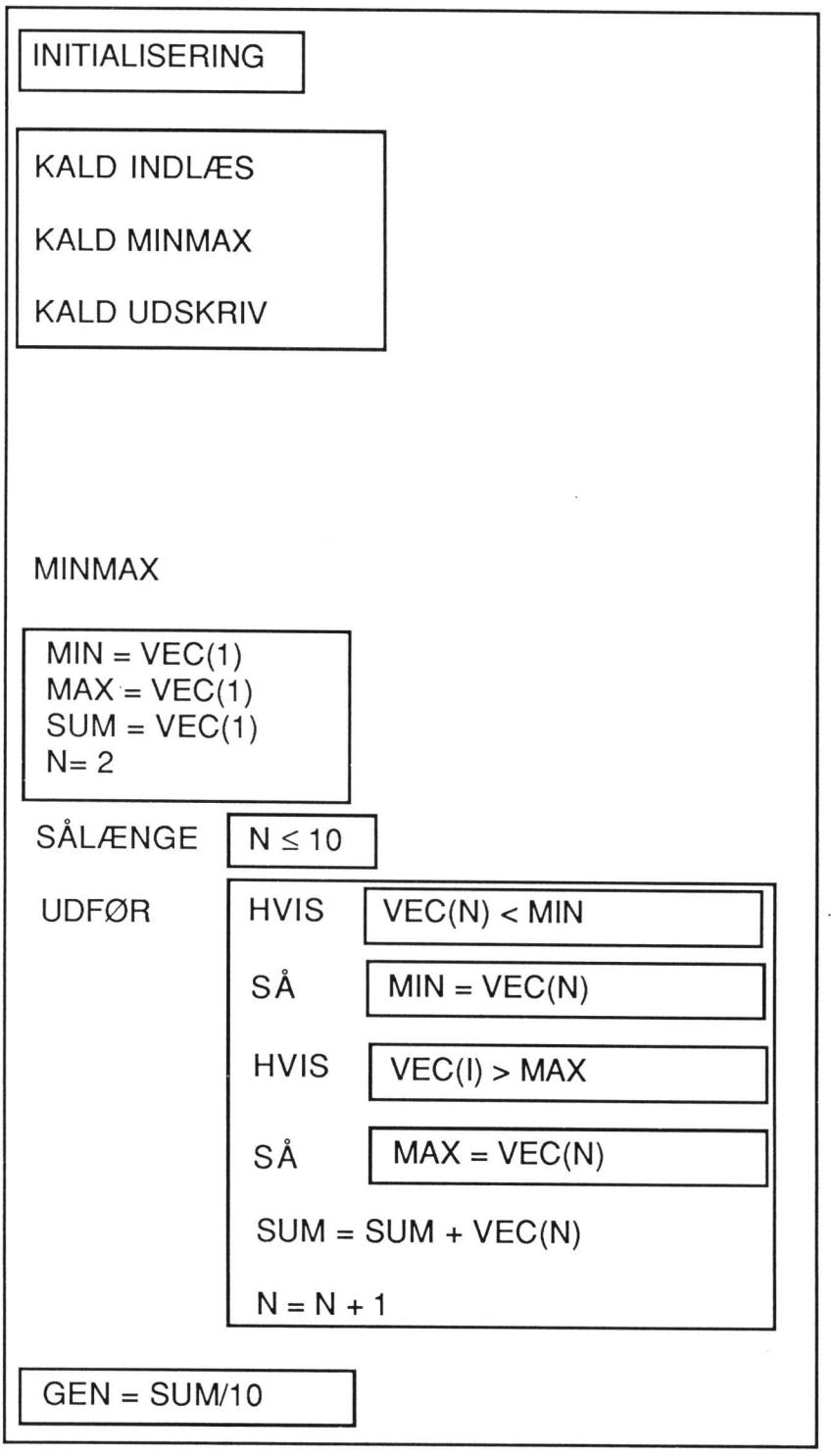
linie 650 Her kaldes procedure BREV.

linie 660 Her kaldes procedure NYTBREV.

Eksempel 32

Overførsel af indexeret variabel

Det sidste eksempel skal vise overførslen af en indexeret variabel som parameter. Programmet indlæser 10 tal i en indexeret variabel, hvorefter programmet finder det mindste tal, det største tal og gennemsnittet.



```
100 // Eksempel
110
120 DIM VEC(10)
130 MIN:=0;MAX:=0;GEN:=0
140
150 PROC INDLÆS(REF VEC()) // indlæsning af 10 tal
160     FOR N:=1 TO 10 DO
170         PRINT USING "Indtast tal nr. ##:        ":N;
180         INPUT "":VEC(N)
190     NEXT N
200 ENDPROC INDLÆS
210
220 // finder minimum, maksimum og gennemsnittet
230 PROC MINMAX(REF V(),REF MIN,REF MAX,REF MY)
240     MIN:=V(1);MAX:=V(1);SUM:=V(1)
250     FOR N:=2 TO 10 DO
260         IF V(N)<MIN THEN MIN:=V(N)
270         IF V(N)>MAX THEN MAX:=V(N)
280         SUM:=SUM+V(N)
290     NEXT N
300     MY:=SUM/10
310 ENDPROC MINMAX
320
330 PROC UDSKRIV(MIN,MAX,GEN) // udskrivning af resultat
340     PRINT
350     PRINT USING "Det mindste tal er:          #########":MIN
360     PRINT USING "Det største tal er:          #########":MAX
370     PRINT USING "Gennemsnittet er:       #########.##":GEN
380 ENDPROC UDSKRIV
390
400 // hovedprogram
410 EXEC INDLÆS(VEC)
420 EXEC MINMAX(VEC,MIN,MAX,GEN)
430 EXEC UDSKRIV(MIN,MAX,GEN)
440 END
```

Forklaring:

linie 120 — Her erklæres en indexeret variabel med plads til 10 tal

Erklæring af indexeret variabel

linie 150 - 200 — Procedure til indlæsning af de 10 tal. Proceduren har en formel parameter VEC, der er en indexeret variabel. Det vises ved at tilføje en tom parentes til navnet på parameteren.

linie 230 - 310 — Procedure, der finder det mindste tal, det største tal og gennemsnittet. Bemærk, at den indexerede parameter V er vist som en REF-parameter selv om dens værdi ikke ændres og derfor ikke skal føres tilbage til hovedprogrammet.

Indexerede parametre skal altid være REF-parametre !

linie 330 - 380 — Procedure, der udskriver resultatet.

Øvelser

34. Et program skal kunne udskrive medlemskort på printeren:

```
------------------------------------------

                MEDLEMSKORT

          LYSTFISKERFORENINGEN SKALLEN

          Gyldigt fra: XXXXXX  til: XXXXXX

          Navn:          XXXXXXXXXXXXXXXXXXXX
          Adresse:       XXXXXXXXXXXXXXXXXXXX
          Postnr. og by: XXXXXXXXXXXXXXXXX

          Kontingent:    XXX

------------------------------------------
```

Skriv et program, der kan udskrive medlemskortet på printeren. Programmet skal indeholde en procedure til indlæsning og en procedure til udskrivning af medlemskortet.

35. Skriv et program, der indlæser en dato i form af et 8-cifret heltal på formen *ååååmmdd*. Programmet skal teste, om det er en lovlig dato. Programmet skal bestå af en procedure til indlæsning af datoen, en procedure til at teste datoens lovlighed og en procedure til at udskrive resultatet.

36. Til det internationale motionsløb "Fur Rundt" ønskes udformet et program, der løbende holder rede på, hvem der ligger på de 3 første pladser.

Når en løber kommer i mål indtastes:

løbernummer (heltal mellem 1 og 2000)

navn

tid

Hvis løberen har anvendt mindre tid end en af de 3 første, placeres løberen, og der udskrives en ajourført placeringsoversigt på skærmen. Ved opstart sættes løbernummer, navn og tid til

```
9999  XXXXXXXXXXXXXXXXXXXX          999.9
```

Kørslen stoppes ved indtastning af 0 for løbernummer.

Som eksempel på opbygning af skærmbilledet kan følgende foreslås:

```
PLACERING:

PLAC    LØBERNR    NAVN                    TID
-------------------------------------------------
1         75       Valborg Severinsen      255.7
2        108       Johannes Madsen         276.2
3         34       Agnes Mikkelsen         312.0
-------------------------------------------------

løbernummer (0 = stop):    9999
løbernavn:                 XXXXXXXXXXXXXXXXXXXXX
tid:                       999.9
-------------------------------------------------
```

Når løbet er afsluttet udskrives en liste over samtlige løbere i løbernummerorden med angivelse af navn og tid. Listen afsluttes med uskrivning af antal løbere i alt og den gennemsnitlige løbetid således:

```
-------------------------------------------------

FUR RUNDT

DELTAGERLISTE MED OPNÅEDE TIDER

NR      NAVN                                TID
-------------------------------------------------
9999    XXXXXXXXXXXXXXXXXXXXXXX             999.9
9999    XXXXXXXXXXXXXXXXXXXXXXX             999.9
.       .                                   .
.       .                                   .


-------------------------------------------------

ANTAL LØBERE I ALT:              9999

GENNEMSNITLIG LØBETID:           999.9

-------------------------------------------------
```

Skriv programmet.

37. Skriv et program, der læser en række talsæt bestående af "kundenr" og "salg år til dato":

Inddataeksempel:

10485, 78631.50, 10210, 5100.00, 10914, 91235.85, . . .

Programmet skal dernæst udskrive en statistik som vist herunder:

```
*****************************
*      SALGSSTATISTIK       *
*****************************
```

Kundenr.	Salg år til dato	Afvigelse
9999999	999999999.99	999999.99
9999999	999999999.99	999999.99
.	.	.
.	.	.

I kolonnen "Afvigelse" skal der stå, hvor meget "Salg år til dato" afviger fra gennemsnittet. Dette tal kan være positivt, 0 eller negativt.

Programmet skal inddeles i procedurer.

Beskriv først opgaven ved hjælp af et diagram. Skriv derefter programmet.

38. Der ønskes konstrueret et program, der kan udskrive en oversigt over den månedlige betaling for børn i en daginstitution. Et eksempel på en udskrift er vist nedenfor.

Inddata består af en række sammenhørende talpar for hvert barn bestående af henholdsvis barnets alder og familiens indkomst, f.eks.

7, 75800, 1, 195000, ...

Den månedlige betaling afhænger af barnets alder og familiens indkomst i henhold til følgende tabel:

Barnets alder	Familiens indkomst						
	0 - 50000	50001-180000			over 180000		
		mindst	% af beløb over 50000	højst	mindst	% af beløb over 180000	højst
0 - 2	50	50	0.9	1100	1100	1.0	2150
3 - 6	100	100	0.6	700	700	0.8	1300
7 - 14	150	150	0.3	350	350	0.5	675

Uddataeksempel:

BETALINGSOVERSIGT:

Antal familier, der betaler følgende pr. måned:

Aldersgruppe	50 kr.	51 - 400 kr.	401 - 1000 kr.	over 1000 kr.
0 - 2 år	1	3	7	2
3 - 4 år	0	1	4	5
5 - 6 år	0	5	0	2
7 - 14 år	0	1	0	0

10. Datafiler

Datafiler

Ligesom programmer kan lagres på disken, kan data også lagres i *filer* på disken.

Sekventielle filer

I COMAL80 har man to slags filer - *sekventielle filer* og *direkte filer*. Data lagres i en fil i poster. De data, der logisk hører sammen, kaldes en *post*.

Direkte filer

For en vare skal der f.eks. lagres varenummer, varenavn og lagerbeholdning. Disse tre data kaldes tilsammen for en post.

Post

Når man læser eller skriver i en fil, læser eller skriver man altid en post ad gangen.

Post

Varenummer (8)	Varenavn (40 + 1)	Lager-beholdning (8)

Ved behandling af *sekventielle filer* skal man altid læse eller skrive fra *start af filen*. Man kan ikke blot læse en enkelt post inde i filen. Dog har man mulighed for ved skrivning i filen at *tilføje nye poster sidst i filen.*

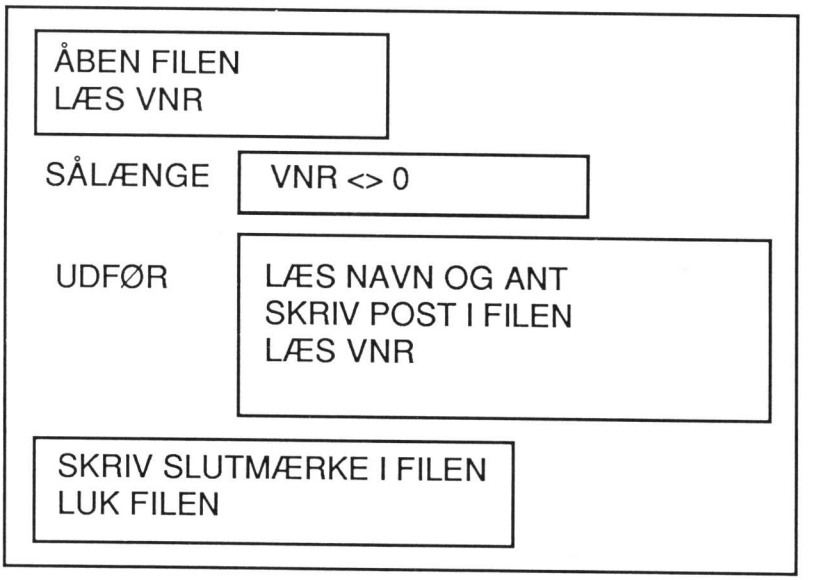
Ved *direkte filer* har *hver post et nummer*. Her kan man direkte læse eller skrive i en bestemt post ved at angive postens nummer.

Åbning af filen

Inden en fil kan anvendes, skal den altid først åbnes. Hvis man forsøger at åbne en fil, der ikke findes, bliver den automatisk oprettet.

Eksempel 33

Programmet herunder opretter en sekventiel fil og skriver et antal vareposter i filen:



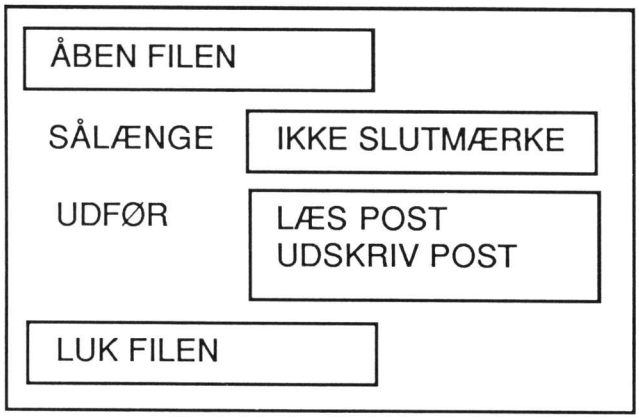
```
100 // udskrivning i en sekeventiel fil
110
120 OPEN FILE 1,"varefil",WRITE
130 DIM NAVN$ OF 40
140 CLEAR
150
160 INPUT "Indtast varenummer (STOP = 0):     ":VNR
170 WHILE VNR>0 DO
180     NAVN$:=""
190     EDIT "Indtast varenavn:                ":NAVN$
200     INPUT "Indtast antal enehder:          ":ANT
210     PRINT
220     WRITE FILE 1:VNR,NAVN$,ANT                          WRITE FILE
230     INPUT "Indtast varenummer (STOP = 0):     ":VNR
240 ENDWHILE
250
260 ENDFILE FILE 1
270 CLOSE FILE 1
280 END
```

Forklaring:

linie 120	Filen åbnes og bliver herved oprettet på disken under navnet "varefil". Filen åbnes i WRITE-mode. Det betyder, at der kan skrives i filen.	WRITE-mode
linie 220	Her skrives en post i filen. Efter FILE står et *kanalnummer*, der skal være det nummer, der er anvendt i OPEN-sætningen.	OPEN
linie 260	Når der ikke skal skrives flere poster i filen, skal der sættes et slutmærke. Det sker med ENDFILE-sætningen.	ENDFILE
linie 270	CLOSE-sætningen lukker filen igen.	CLOSE

Eksempel 34

Programmet herunder læser de vareposter, der blev skrevet i filen "varefil" i ovenstående eksempel.




```
100 // indlæsning af data fra en sekventiel fil
110
120 OPEN FILE 1,"varefil",READ
130 DIM NAVN$ OF 40
140 CLEAR
150
160 WHILE NOT EOF(1) DO
170     READ FILE 1:VNR,NAVN$,ANT
180     PRINT "VARENUMMER:                ";VNR
190     PRINT "varenavn:                  ";NAVN$
200     PRINT "antal enehder:             ";ANT
210     PRINT
220 ENDWHILE
230
240 CLOSE FILE 1
250 END
```

READ FILE

Forklaring:

READ-mode	linie 120	Filen åbnes i READ-mode. Det betyder, at man kan læse i filen.
	linie 170	Her læses én varepost i filen.
	linie 240	Filen lukkes.

Eksempel 35

Hvis man har en sekventiel fil, og man ønsker at *tilføje nye poster til filen*, kan de tilføjes sidst i filen ved at åbne den i APPEND-mode.

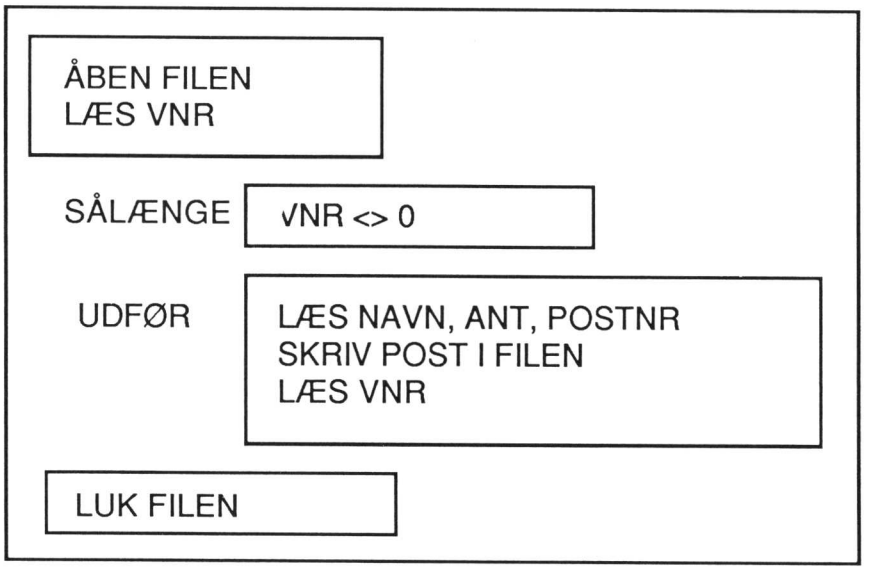
APPEND-mode

```
100 // udskrivning i en sekeventiel fil
110
120 OPEN FILE 1,"varefil",APPEND
130 DIM NAVN$ OF 40
140 CLEAR
150
160 INPUT "Indtast varenummer (STOP = 0):     ":VNR
170 WHILE VNR>0 DO
180     NAVN$:=""
190     EDIT "Indtast varenavn:                ":NAVN$
200     INPUT "Indtast antal enehder:          ":ANT
210     PRINT
220     WRITE FILE 1:VNR,NAVN$,ANT
230     INPUT "Indtast varenummer (STOP = 0):     ":VNR
240 ENDWHILE
250
260 ENDFILE FILE 1
270 CLOSE FILE 1
280 END
```

Eksempel 36

I nedenstående eksempel oprettes den samme varefil, som omtalt foran, igen - men nu som *direkte fil*.

Direkte fil



```
100 // udskrivning i en relativ fil
110
120 OPEN FILE 1,"varefil",RANDOM,58
130 DIM NAVN$ OF 40
140 CLEAR
150
160 INPUT "Indtast varenummer (STOP = 0):     ":VNR
170 WHILE VNR>0 DO
180     NAVN$:=""
190     EDIT "Indtast varenavn:                ":NAVN$
200     INPUT "Indtast antal enehder:          ":ANT
210     INPUT "Indtast postnummer:             ":REC
220     PRINT
230     WRITE FILE 1,REC:VNR,NAVN$,ANT
240     INPUT "Indtast varenummer (STOP = 0):      ":VNR
250 ENDWHILE
260
270 CLOSE FILE 1
280 END
```

Forklaring:

RANDOM-mode linie 120

Filen oprettes på disken under navnet "varefil". Da det er en direkte fil, åbnes den i RANDOM-mode. Endvidere skal man angive *postlængden*, der betyder, hvor mange tegn hver post fylder på disken.

Et tal fylder altid 8 tegn og en tekst fylder aktuel længde + 1.

Vareposten består af 2 tal og 1 tekst. Da NAVN$ er dimensioneret til 40 tegn, bliver postlængden 8 + 41 + 8 = 57.

Postnummer linie 210

I en direkte fil har hver post et *postnummer*. Det nummer skal man angive, når man læser eller skriver i filen. Postnummeret indlæses i linie 190.

linie 230

Her skrives en post i filen. Efter kanalnummeret står postnummeret, der er indlæst i linie 190.

Eksempel 37

Læs en post

Nedenstående program læser en eller flere poster i filen, der er oprettet i ovenstående eksempel. Den eneste nye sætning er linie 180, der læser en post.

```
100 // indlæsning af data fra en direkte fil
110
120 OPEN FILE 1,"varefil",RANDOM,58
130 DIM NAVN$ OF 40
140 CLEAR
150
```

```
160 INPUT "Indtast postnummer (STOP = 0) :     ":REC
170 WHILE REC>0 DO
180     READ FILE 1,REC:VNR,NAVN$,ANT
190     PRINT USING "varenummer:              ###### ":VNR
200     PRINT "varenavn:                        ";NAVN$
210     PRINT USING "antal enehder:           ######":ANT
220     PRINT
230     INPUT "Indtast postnummer (STOP = 0):       ":REC
240 ENDWHILE
250
260 CLOSE FILE 1
270 END
```

Eksempel 38

Ajourfør en post

Følgende program ajourfører lagerbeholdningen i filen "varefil". Bemærk, at hver gang man anvender en direkte fil, skal den åbnes i RANDOM-mode og med samme postlængde hver gang.

```
100 // ajourføring af lagerbeholdning
110
120 OPEN FILE 1,"varefil",RANDOM,58
130 DIM NAVN$ OF 40
140 CLEAR
150
160 INPUT "Indtast postnummer (STOP = 0) :     ":REC
170
180 WHILE REC>0 DO
190
200     READ FILE 1,REC:VNR,NAVN$,ANT
210
220     PRINT USING "varenummer:              ###### ":VNR
230     PRINT "varenavn:                        ";NAVN$
240     PRINT USING "antal enehder:           ######":ANT
250     PRINT
260     INPUT "Indtast lagertilgang:                ":TILG
270     ANT:=ANT+TILG
280
290     WRITE FILE 1,REC:VNR,NAVN$,ANT
300
310     PRINT
320     INPUT "Indtast postnummer (STOP = 0):       ":REC
330
340 ENDWHILE
350
360 CLOSE FILE 1
370 END
```

Eksempel 39

Et større eksempel

Vi afslutter behandlingen af datafiler med et lidt større eksempel. Vi vil bruge den samme varefil, som vi har anvendt i ovenstående eksempler.

Programmet skal kunne oprette en ny vare i filen, udskrive indholdet af en varepost, ajourføre lagerbeholdningen samt slette en vare i filen. Det vil vi gøre ved at skrive et *hovedprogram,* der kalder de enkelte operationer på filen som procedurer.

Varefilen skal være en direkte fil. I ovenstående eksempler skulle vi indtaste postnummeret, når vi skulle læse eller skrive i en varepost. Det kan man ikke leve med i praksis, da man ikke kan huske dette tal.

Index

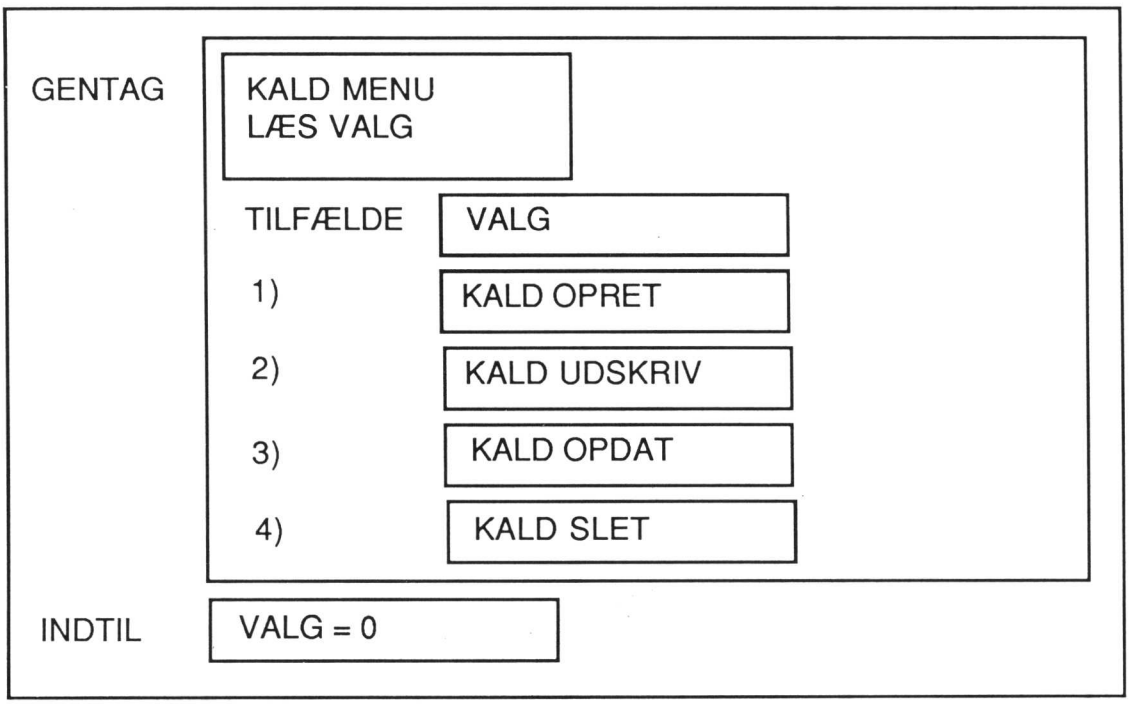
Dette problem har vi løst ved at lave et *index,* der indeholder varenumerene. Dette index er lagret i en *sekventiel fil.* Når man skal finde et bestemt varenummer, søger man varenummeret i indexfilen. Postnummeret er varenummerets relative nummer i indexfilen.

Når en varepost slettes, skal den blot slettes i indexfilen ved at der skrives 0 på den pågældende plads.

Opret index-fil

Inden programmet kan køre, skal vi først oprette indexfilen. Hertil behøver vi et lille hjælpeprogram. Det udskriver 100 nuller i indexfilen. Vareregistret har da plads til 100 poster.

```
100 // Eksempel
110
120 OPEN FILE 1,"vareindex",WRITE
130 NØGLE:=0
140 FOR n:=1 TO 100 DO WRITE FILE 1:NØGLE
150 ENDFILE FILE 1
160 CLOSE FILE 1
170
180 OPEN FILE 2,"varereg",RANDOM,58
190 CLOSE FILE 2
200
210 END
```




```
100 // Eksempel
110
120 DIM NØGLER(100),NAVN$ OF 40,SV$ OF 1,VG$ OF 1
130
140 PROC LÆSINDEX (REF NØGLER()) // indlæs index til lageret
150   OPEN FILE 2,"vareindex",READ
160   FOR n:=1 TO 100 DO READ FILE 2:NØGLER(n)
170   CLOSE FILE 2
180 ENDPROC LÆSINDEX
190
200 PROC SKRIVINDEX (REF NØGLER()) //skriv index tilbage på disken
210   OPEN FILE 2,"vareindex",WRITE
220   FOR n:=1 TO 100 DO WRITE FILE 2:NØGLER(n)
230   CLOSE FILE 2
240 ENDPROC SKRIVINDEX
250
```

```
260 PROC OPRET (REF NØGLER(),NAVN$)  // opret vare
270   OPEN FILE 1,"varereg",RANDOM 58
280   CLEAR
290   REPEAT
300     INPUT "Indtast varenummer (STOP = 0):        ":VNR
310     IF VNR>0 THEN
320       NAVN$:=" "
330       EDIT "Indtast varenavn:                     ":NAVN$
340       INPUT "Indtast antal enheder:               ":ANT
350       PRINT
360       REC:=0;n:=1
370       WHILE (REC=0) AND (n<=100) DO
380         IF NØGLER(n)=0 THEN
390           REC:=n;NØGLER(n):=VNR
400         ELSE
410           n:=n+1
420         ENDIF
430       ENDWHILE
440       WRITE FILE 1,REC:VNR,NAVN$,ANT
450     ENDIF
460   UNTIL VNR=0
470   CLOSE FILE 1
480 ENDPROC OPRET
490
500 PROC UDSKRIV (REF NØGLER(),NAVN$)  // forespørgsel på vare
510   OPEN FILE 1,"varereg",RANDOM,58
520   CLEAR
530   REPEAT
540     INPUT "Indtast varenummer (STOP = 0):        ":VNR
550     IF VNR>0 THEN
560       REC:=0;n:=1
570       WHILE (REC=0) AND (n<=100) DO
580         IF NØGLER(n)=VNR THEN
590           REC:=n
600         ELSE
610           n:=n+1
620         ENDIF
630       ENDWHILE
640       IF REC=0 THEN
650         PRINT CHR$(7);"VARENUMMER FINDES IKKE !"
660       ELSE
670         READ FILE 1,REC:VNR,NAVN$,ANT
680         PRINT "varenummer:                          ";VNR
690         PRINT"varenavn:                             ";NAVN$
700         PRINT"antal enheder:                        ";ANT
710         PRINT
720       ENDIF
730     ENDIF
740   UNTIL VNR=0
750   CLOSE FILE 1
760 ENDPROC UDSKRIV
770
```

```
780 PROC OPDAT(REF NØGLER(),NAVN$) // opdater lagerbeholdning
790   OPEN FILE 1,"varereg",RANDOM,58
800   CLEAR
810   REPEAT
820     INPUT "Indtast varenummer (STOP = 0):        ":VNR
830     IF VNR>0 THEN
840       REC:=0;n:=1
850       WHILE (REC=0) AND (n<=100) DO
860         IF VNR=NØGLER(n) THEN
870           REC:=n
880         ELSE
890           n:=n+1
900         ENDIF
910       ENDWHILE
920       IF REC=0 THEN
930         PRINT CHR$(7);"VARENUMMER FINDES IKKE !"
940       ELSE
950         PRINT REC
960         READ FILE 1,REC:VNR,NAVN$,ANT
970         PRINT "varenummer:                    ";VNR
980         PRINT"varenavn:                       ";NAVN$
990         PRINT"antal enheder:                  ";ANT
1000        PRINT
1010        INPUT "Indtast lagertilgang:          ":TILG
1020        ANT:=ANT+TILG
1030        WRITE FILE 1,REC:VNR,NAVN$,ANT
1040        PRINT
1050      ENDIF
1060    ENDIF
1070  UNTIL VNR=0
1080  CLOSE FILE 1
1090 ENDPROC OPDAT
1100
```

```
1110 PROC SLET(REF NØGLER(),NAVN$)  // slet vare
1120  CLEAR
1130  OPEN FILE 1,"varereg",RANDOM,58
1140  REPEAT
1150    INPUT "varenummer (STOP = 0):          ":VNR
1160    IF VNR>0 THEN
1170      REC:=0;n:=1
1180      WHILE (REC=0) AND (n<=100) DO
1190        IF NØGLER(n)=VNR THEN
1200          REC:=n
1210        ELSE
1220          n:=n+1
1230        ENDIF
1240      ENDWHILE
1250      IF REC=0 THEN
1260        PRINT CHR$(7);"VARENUMMER FINDES IKKE !"
1270      ELSE
1280        READ FILE 1,REC:VNR,NAVN$,ANT
1290        PRINT "varenavn:                              ";NAVN$
1300        PRINT
1310        SV$:="N"
1320        EDIT "Ønskes varen slettet (j/n) ?          ":SV$
1330        PRINT
1340        IF SV$="J" OR SV$="j" THEN NØGLER(REC):=0
1350      ENDIF
1360    ENDIF
1370  UNTIL VNR=0
1380  CLOSE FILE 1
1390 ENDPROC SLET
1400
1410 PROC MENU(REF VG$) // funktionsmenu og indtastning af valg
1420  CLEAR
1430  CURSOR 30,1
1440  PRINT "**************************"
1450  CURSOR 30,2
1460  PRINT "* L A G E R S Y S T E M  *"
1470  CURSOR 30,3
1480  PRINT "**************************"
1490  CURSOR 25,5
1500  PRINT "1.   OPRET VARE"
1510  CURSOR 25,7
1520  PRINT "2.   UDSKRIV VARE"
1530  CURSOR 25,9
1540  PRINT "3.  AJOURFØR VARE"
1550  CURSOR 25,11
1560  PRINT "4.  SLET VARE"
1570  CURSOR 25,14
1580  PRINT "0.  STOP"
1590  CURSOR 30,17
1600  PRINT "indtast funktionsvalg:"
```

```
1610  VG$:=""
1620  REPEAT
1630    CURSOR 55,17
1640    EDIT " ":VG$
1650    IF (NOT VG$ IN "01234") OR (LEN(VG$)=0) THEN PRINT CHR$(7);
1660  UNTIL VG$ IN "01234" AND LEN(VG$)>0
1670 ENDPROC MENU
1680
1690 // hovedprogram
1700  EXEC LÆSINDEX(NØGLER)
1710  REPEAT
1720    EXEC MENU(VG$)
1730    CASE VG$ OF
1740    WHEN "1"
1750      EXEC OPRET(NØGLER,NAVN$)
1760    WHEN "2"
1770      EXEC UDSKRIV(NØGLER,NAVN$)
1780    WHEN "3"
1790      EXEC OPDAT(NØGLER,NAVN$)
1800    WHEN "4"
1810      EXEC SLET(NØGLER,NAVN$)
1820    OTHERWISE
1830      // do nothing
1840    ENDCASE
1850  UNTIL VG$="0"
1860  EXEC SKRIVINDEX(NØGLER)
1870  END
```

Nye sætninger	Forklaring
OPEN	OPEN FILE sætningen anvendes til at åbne en datafil på disken. Filen kan åbnes som: OPEN FILE kanal,"filnavn",WRITE OPEN FILE kanal,"filnavn",READ OPEN FILE kanal,"filnavn",APPEND OPEN FILE kanal,"filnavn",RANDOM,længde
CLOSE FILE	CLOSE FILE sætningen lukker en fil
WRITE FILE	WRITE FILE sætningen udskriver en post i en fil.
READ FILE	READ FILE sætningen læser en post i en fil
ENDFILE FILE	ENDFILE FILE bruges til at markere slutmærke i en sekventiel fil.

Nye funktioner	Forklaring
EOF	Funktion, der tester, om slutmærket er læst i en sekventiel fil.
VAL	Konverterer taludtryk i en alfamerisk variabel til tal i en numerisk variabel.
STR$	Konverterer tal i en numerisk variabel til en talværdi i en alfamerisk variabel.

Øvelser

39. Skriv et program, der ved anvendelse af RND-funktionen skriver 20 tal i en sekventiel fil.

Skriv derefter et program, der læser de 20 tal i filen og udskriver dem på skærmen.

Skriv et program, der ved RND-funktionen tilføjer 10 nye tal til filen. Anvend dernæst ovenstående program til at udskrive filens indhold på skærmen.

40. Et kunderegister skal bestå af poster med følgende indhold:

kundenummer	(1 - 999)
kundenavn	(max. 30 tegn)
adresse	(max. 30 tegn)
postnr. og by	(max. 20 tegn)
telefonnr.	(xx-xxxxxx)
saldo	

Løs følgende opgaver:

Først beskrives opgaverne i form af strukturdiagrammer og dernæst skrives de tilsvarende programmer:

a) Skriv et program, der opretter posten som en direkte fil.

b) Skriv et program, der kan udskrive indholdet af en kundepost på printeren.

c) Skriv et program, der kan ajourføre alle data med undtagelse af kundenummer i en kundepost. Programmet skal først udskrive indholdet af posten på skærmen. Dernæst skal man kunne ændre postens data og endelig skal den ajourførte post skrives tilbage i filen.

d) Skriv et program, der kan slette en kundepost i filen. Posten må kun kunne slettes, hvis saldo = 0.

41. De 4 programmer fra foregående øvelse skal nu bygges sammen til ét program. Programmerne lægges ind som procedurer, der skal kunne kaldes fra et hoved-program.

Beskriv først opgaven i et eller flere strukturdiagrammer.

Skriv derefter programmet.

42. Der skal oprettes et adressekartotek, der indeholder følgende personoplysninger:

nøgle	heltal
efternavn	(20 karakterer)
fornavn(e)	(20 karakterer)
adresse	(30 karakterer)
postnr	4-cifret tal
by	(20 karakterer)
telefonnr	(10 karakterer)

Kartoteket oprettes som en direkte fil. Når en post oprettes i kartoteket, tildeles den et nummer (nøglen), der skal være det postnummer, som posten gemmes under i filen.

a) Skriv et program, der kan oprette kartoteket.

b) Skriv et program, der kan oprette poster i kartoteket. Ved tildeling af postnummer, skal programmet teste, at der ikke i forvejen findes en post i kartoteket med dette nummer. Er det tilfældet, skal programmet udskrive en fejlmeddelelse.

c) Skriv et program, der læser et postnummer. Hvis der findes en post i kartoteket med dette nummer, skal postens indhold udskrives på skærmen. Findes posten ikke, skrives en fejlmeddelelse på skærmen.

d) Skriv et program, der kan ændre indholdet af en post. Man skal indtaste postnummer. Hvis posten findes, udskrives indholdet på skærmen hvorefter der skal være mulighed for at ændre indholdet. Findes posten ikke, udskrives en fejlmeddelelse på skærmen.

e) Skriv et program, der kan slette en post i kartoteket. Man skal indtaste postnummer. Hvis posten findes, udskrives postens indhold på skærmen, og der skal spørges, om man ønsker posten slettet. Hvis posten ikke findes, skal der udskrives en fejlmeddelelse på skærmen.

f) Skriv et program, der kan udskrive en liste over alle poster på printeren.

g) Skriv et program, hvor man indtaster et efternavn. Programmet skal da på skærmen udskrive alle poster, der indeholder dette efternavn.

h) Alle programmerne (b - g) skal nu ændres til procedurer. Man skal dernæst samle alle procedurerne til ét program. Desuden skal der laves et menuprogram, hvorfra de enkelte procedurer kaldes.

i) Overvej mulighederne for at udbygge programmet, så man kan søge på alle felter og kombinationer af felter.

11. Styring af skærm og printer

Der findes en række faciliteter og styrekoder til udskrivning på skærm og printer, som gør det let at konstruere mere brugervenlige skærmbilleder og udskrifter.

Eksempel 40

```
100 // Eksempel
110 DIM NAVN1$ OF 20, NAVN2$ OF 20
120 CLEAR
130 CURSOR 20,5
140 INPUT "Indtast fornavn      ":NAVN1$
150 CURSOR 20,7
160 INPUT "Indtast efternavn    ":NAVN2$
170 CURSOR 20,10
180 PRINT NAVN2$;" , ";NAVN1$
190 END
```

Forklaring:

Cursor-styring

CURSOR er en sætning, der bevirker, at cursoren stiller sig et bestemt sted på skærmen. Efter CURSOR står to koordinater x,y, hvor x angiver hvor langt inde på linie, cursoren skal placeres, og y angiver på hvilken linie det er.

xxx CURSOR 20,5

betyder, at cursoren stiller sig i linie 5 fra oven og 20 positioner inde på linien.

Eksempel 41

Følgende program læser et navn og udskriver det tre steder på skærmen, men med forskellig præsentation.

CURSOR

```
100 // Eksempel
110 DIM NAVN$ OF 20
120 CLEAR
130 CURSOR 20,5
140 INPUT "Indtast et navn  ":NAVN$
150 CURSOR 10,9
160 PRINT CHR$(27)+CHR$(106);NAVN$;CHR$(27)+CHR$(107)
170 CURSOR 40,12
180 PRINT CHR$(27)+CHR$(108);NAVN$;CHR$(27)+CHR$(109)
190 CURSOR 30,15
200 PRINT CHR$(27)+CHR$(110);NAVN$;CHR$(27)+CHR$(111)
210 END
```

Forklaring:

Invers video	linie 160	Navnet udskrives i *invers video*. CHR$(27)+CHR$(106) sætter skærmen i invers mode. CHR$(27)+CHR$(107) sætter skærmen tilbage til normal skrift igen.
Understreg	linie 180	Navnet understreges. CHR$(27)+CHR$(108) starter understregning af udskriften. CHR$(27)+CHR$(109) sætter skærmen tilbage til normal skrift igen.
Blinkende skrift	linie 200	Navnet udskrives med blinkende skrift. CHR$(27)+CHR$(110) starter den blinkende skrift, medens CHR$(27)+CHR$(111) sætter skærmen tilbage til normal skrift igen.

Da hver skærmtype og model har sine egne tegnkodeværdier kan det i nogle tilfælde være nødvendigt at lave en RESET på skærmen (nulstilling). Det gøres i Comal80 f.eks. ved at tilføje linie 125 således:

```
125 PRINT CHR$(27)+CHR$(65)
```

Eksempel 42

Grafik

Nedenstående program tegner en ramme på skærmen. Rammen tegnes ved anvendelse af skærmens grafiske symboler. I hver PRINT-sætning i programmet genereres et grafisk symbol.

```
100 // Eksempel
110 CLEAR
120 RX:=30;RY:=5;RA:=20;RB:=4
130 // udskrivning af rammen
140 CURSOR RX,RY
150 PRINT CHR$(27)+CHR$(71)+CHR$(65)
160 FOR n:= 1 TO RA - 1 DO
170    CURSOR RX + n,RY
180    PRINT CHR$(27)+CHR$(71)+CHR$(73)
190 NEXT n
200 CURSOR RX + RA,RY
210 PRINT CHR$(27)+CHR$(71)+CHR$(66)
220 FOR n:= 1 TO RB - 1 DO
230    CURSOR RX,RY + n
240    PRINT CHR$(27)+CHR$(71)+CHR$(74)
250    CURSOR RX + RA,RY + n
260    PRINT CHR$(27)+CHR$(71)+CHR$(74)
270 NEXT n
280 CURSOR RX,RY+RB
290 PRINT CHR$(27)+CHR$(71)+CHR$(67)
300 FOR n:= 1 TO RA - 1 DO
310    CURSOR RX + n,RY + RB
320    PRINT CHR$(27)+CHR$(71)+CHR$(73)
330 NEXT n
```

```
340 CURSOR RX + RA,RY + RB
350 PRINT CHR$(27)+CHR$(71)+CHR$(68)
360 // En tekst skrives inde i rammen
370 CURSOR RX + 3,RY + 2
380 PRINT "Dette er en ramme"
390 END
```

Skrifttyper

På printeren findes forskellige skrifttyper:

elongeret (udvidet)
komprimeret
normal

Elongeret

CHR$(31) giver elongeret udskrift.

Komprimeret

CHR$(28) aktiverer komprimeret udskrift.

Normal

CHR$(30) aktiverer normal udskrift.

CHR$-værdierne skal forekomme i PRINT-sætninger.

Eksempel 43

```
100 // Eksempel    - PRINTTYPER
110 SELECT OUTPUT "spool -dprint0 -s"
120 PRINT CHR$(31);"Elongeret udskrift !";CHR$(30)
130 PRINT
140 PRINT "Normal udskrift !"
150 PRINT
160 PRINT CHR$(28);"Komprimeret udskrift !";CHR$(30)
170 END
```

Der kan dog forekomme afvigende printerkoder på forskellige modeller. Se da værdierne i den medfølgende printer-manual.

Nye funktioner:	Forklaring:
CURSOR x,y	Sætning, der placerer cursoren i en bestemt position, hvor x og y skal være heltal.

Nye funktioner:	Forklaring:
CHR$(x)	Funktion, der til et tal tilordner et tilsvarende tegn eller kontroltegn. Funktionen anvendes ofte til styring af udskrivning på skærm eller printer.

Nye funktioner:	Forklaring:
SQR(x)	Funktion, der beregner kvadratroden af x, hvor x er et tal eller et numerisk udtryk.

Index

OPTIMAS Supermax-serie omfatter følgende:

UNIPLEX II+	Introduktion til elektronisk regneark
UNIPLEX II+	Tekstbehandling
UNIPLEX II+	Introduktion til database
BUDGETTERING PÅ EDB	Budgetsimulering i handels- og produktionsvirksomheder
COMAL80 PÅ SUPERMAX	Projekthæfte og håndbog til HH, efg, merkonom, efteruddannelses-kurser m.v.

Til alle bøger i denne serie er der fremstillet en diskette med modeller, eksempler og øvelser.

Sats: FICO ApS
Tryk: FN-tryk, Skive

ISBN 87 - 88662 - 62 - 4