RCSL Nr.: 42-i1758

Udgave: December 1981

Forfatter: Jørgen Hansen

Titel:

COMAL80
Programmeringsvejledning

Nøgleord:

Danish language, RC700, COMAL80, programmeringsvejledning.

Resumé:

Denne manual beskriver sproget COMAL80, som det er implementeret på RC700. Programmeringsvejledningen indeholder beskrivelser af de enkelte kommandoer, sætninger og funktioner i sproget og er beregnet som et opslagsværk for brugere af RC700, der ønsker at programmere i COMAL80.

Udgivet af A/S Regnecentralen af 1979, København

RCSL 42-i1591

INDHOLDSFORTEGNELSE SIDE

INTRODUKTION 1

ABS 4
AT 6
ATN 8
AUTO 10
CASE 12
CHR$ 14
CLOSE 16
CON 18
COS 20
CREATE 22
DATA 24
DEL 26
DELETE 28
DIM 30
DIR 34
DISMOUNT 36
END 38
ENTER 40
EOD 42
EOF 44
Etikette 46
EXEC 48
EXP 50
FOR 52
GET$ 54
GLOBAL 56
GOTO 58
IF 60
INPUT 64
INT 66
LEN 68
LIST 70
LOAD 72

INDHOLDSFORTEGNELSE (fortsat) SIDE

LOG 74
MARGIN 76
MOUNT 78
NEW 80
OPEN 82
ORD 84
PREFIX 86
PRINT 88
PRINT FILE 94
PROC 96
RANDOMIZE 104
READ 106
READ FILE 110
RENUMBER 112
REPEAT 114
RESTORE 116
RND 118
RUN 120
SAVE 122
SELECT OUTPUT 124
SGN 126
SIN 128
SIZE 130
SQR 132
STOP 134
TAB 136
TAN 138
Tildeling 140
WHILE 144
WRITE FILE 146
ZONE 148

INTRODUKTION

I det følgende er beskrevet de enkelte dele, som COMAL80 består af. På manualens venstresider står en udførlig beskrivelse, og på højresiderne er angivet det nøjagtige format og eventuelt et eksempel for de enkelte sætninger, kommandoer og funktioner.

Øverst på højresiderne står, om det drejer sig om

1) en COMAL80 sætning, der kan udføres under afvikling af et COMAL80 program,

2) en kommando, der kan udføres, idet den indtastes fra tastaturet, eller

3) en funktion, der kan indgå som del af et udtryk i en COMAL80 sætning.

Derefter er formatet beskrevet.
Den tekst, som er skrevet med store bogstaver, skal indtastes som den står. Det samme gælder tegnene kolon, semikolon, komma, venstreparentes, højreparentes, dollartegn og lighedstegn.

Når en tekst er skrevet med små bogstaver og omsluttet af tegnene < og >, skal den ikke skrives, som den står, men erstattes af en række tegn, som beskrevet herunder eller umiddelbart under formatet.

Er en del af formatet omsluttet af kantede parenteser, betyder det, at denne del kan udelades.

Er en del af formatet omsluttet af klammer ("tuborg'er"), kan den udelades eller skrives én eller flere gange.

Til at beskrive formaternes variable dele bruges som nævnt tekster omsluttet af < og >. Disse dele beskrives ved hjælp af tegnene ::= og |, der kan læses som henholdsvis "består af" og "eller".

For eksempel kunne man definere et heltal på følgende måde:

```
<heltal>  ::= [<fortegn>]<ciffer> {<ciffer>}
<fortegn> ::= +|-
<ciffer>  ::= 0|1|2|3|4|5|6|7|8|9
```

En række begreber går igen i mange forskellige formater, og de er beskrevet herunder.

En <sætning> er en COMAL80 sætning, der indtastes indledt af et linienummer. Et <navn> er en tekststreng, der består af mellem ét og seksten tal og bogstaver. Det første tegn i tekststrengen skal dog altid være et bogstav. En <talkonstant> er et tal, der er skrevet som heltal, decimaltal med decimalpunktum eller på eksponentiel form (f.eks. 15E4 eller .125E-2).

En <simpel sætning> er en sætning, som ikke indeholder nogen af følgende konstruktioner: CASE, DATA, END, FOR, GLOBAL, IF, PROC, REPEAT eller WHILE.

En <strengkonstant> er en tekststreng, der er omsluttet af anførselstegn (f.eks. "Dette er en tekststreng"). En formatdel, der kan være enten en <talkonstant> eller en <strengkonstant> kaldes blot en <konstant>. Et <navn> kan repræsentere en plads, der kan tildeles en værdi. Et sådant navn kaldes en <variabel>. Hvis den tildelte værdi er et tal, kaldes det en <talvariabel>, og hvis den tildelte værdi er en tekststreng, kaldes det en <strengvariabel>. Hvis en del af et format kan være en <konstant> eller en <variabel> eller være sammensat af flere af disse samt af funktioner og operatorer, kaldes denne del for et <udtryk>. Hvis et <udtryk> kan tillægges en talværdi, kaldes det et <taludtryk>, og hvis dets værdi er en streng, kaldes det et <strengudtryk>.

Udtryk kan sammensættes af flere udtryk, funktioner samt operatorerne +, -, *, /, DIV, MOD, ↑ og parenteser. Som eksempler på udtryk kan nævnes:

```
1
"Dette er en tekststreng"
i+1
a*SQR(b/10)
```

Et <logisk udtryk> er et <taludtryk>, der siges at være falsk, hvis det har værdien 0, og sandt, hvis værdien er forskellig fra 0.

Ved sammensatte logiske udtryk kan man bruge operatorerne NOT, AND, OR, IN, =, <, >, <=, >= og <>.

ABS

ANVENDELSE

COMAL80 funktionen "ABS" beregner den absolutte værdi (numeriske værdi) af et taludtryk. Den absolutte værdi er et positivt tal.

COMAL80 funktion skrives som: ABS(<udtryk>)
Matematisk symbol skrives som: |<udtryk>|

ABS

COMAL80 funktion

FORMAT

ABS (<taludtryk>)

- ABS -

Anvendes til at beregne den absolutte værdi (den numeriske værdi) af et <taludtryk>.

Eksempel

```
0080 INPUT i,j
0090 IF i <> j THEN
0100   PRINT "De to tal er ikke ens."
0110   PRINT "Forskellen er "; ABS(i-j)
0120 ENDIF
```

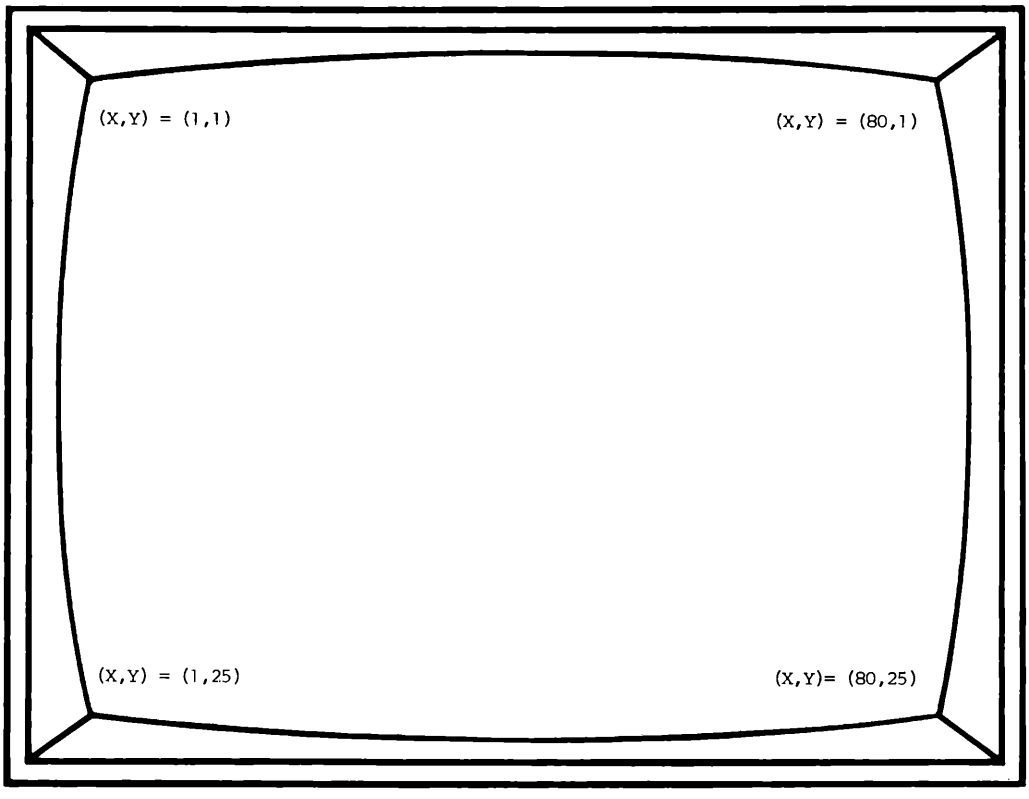
AT

ANVENDELSE

COMAL80 funktionen "AT" anvendes i forbindelse med sætningerne "PRINT" og "PRINT USING", og styrer dataskærmens lysende pegepind (cursoren) hen til det sted, hvorfra udskrivning af efterfølgende data skal begynde. <Taludtrykkene> er de skærmpositioner (X,Y koordinater) hvorfra udskrivningen starter.

X-koordinaten er et tal mellem 1 og 80.

Y-koordinaten er et tal mellem 1 og 25.



AT

COMAL80 funktion

FORMAT

AT(<taludtryk>,<taludtryk>)

- AT -

Anvendes i en "PRINT" sætning til at starte udskrivningen et bestemt sted på dataskærmen.

Eksempel

```
0010 PRINT CHR$(12)
0020 FOR i:= 1 TO 80 DO
0030   PRINT AT(1,2), i MOD 10
0040   IF i MOD 10 = 0 THEN PRINT AT(i,1), i DIV 10
0050 NEXT i
```

ATN

ANVENDELSE

COMAL80 funktionen "ATN" (arcus tangens) udregner den vinkel, hvis tangens er givet ved <taludtryk>. Vinklen bliver udtrykt i radianer.

$$1 \text{ radian} = \frac{360 \text{ grader}}{2 \text{ x pi}} = 57{,}29577951309 \text{ grader}$$

$$\text{pi} = 3{,}141592653589$$

COMAL80 funktionen skrives som: ATN(<taludtryk>)
Matematisk symbol skrives som : atn <taludtryk>

Se i øvrigt funktionerne "SIN", "COS" samt "TAN".

ATN

COMAL80 funktion

FORMAT

ATN(<taludtryk>)

- ATN -
Udregner den vinkel, hvis tangens er lig med <taludtryk>. Vinklen er udtrykt i radianer.

AUTO

ANVENDELSE

Ved indtastning af programlinier kan brugeren benytte kommandoen "AUTO", som automatisk sætter nyt linienummer før indtastning af ny programlinie.

Man kan som vist nedenfor kombinere anvendelsen af <startlinienummer> og <linienummerspring> i <linienummerliste>.

AUTO	Linienummereringen starter ved linie 0010, med spring på 0010
AUTO <startlinienummer>	Linienummereringen starter ved <startlinienummer>, med spring lig med <startlinienummer>.
AUTO <startlinienummer>,	Linienummereringen starter ved <startlinienummer>, med spring på 0010.
AUTO <startlinienummer>,<linienummerspring>	Linienummereringen starter ved <startlinienummer>, med spring lig med <linienummerspring>.
AUTO ,<linienummerspring>	Linienummereringen starter ved linie 0010, med spring lig med <linienummerspring>.

Se i øvrigt kommandoen "RENUMBER".

AUTO

COMAL80 kommando

FORMAT

```
AUTO [<linienummerliste>]
  <linienummerliste> ::=  <startlinienummer>|<startlinienummer>,|
                          <startlinienummer>,<linienummerspring>|
                          ,<linienummerspring>
  <startlinienummer> ::=  <taludtryk>
  <linienummerspring> ::= <taludtryk>
```

- AUTO -
Anvendes til automatisk linienummerering ved indtastning af programlinier.

Eksempel

Indtastet AUTO
Medfører følgende udskrivning af programlinienumre:

```
auto
0010
0020
0030
 ..
```

Indtastet AUTO 50,
Medfører følgende udskrivning af programlinienumre:

```
auto
0050
0060
0070
```

CASE

ANVENDELSE

"CASE" er en COMAL80 sætning, som udfører en eller flere programlinier afhængigt af et udtryks værdi. At forskellige blokke af programlinier udføres alt efter, hvilken værdi udtrykket har, kaldes en FORGRENING.

I en "CASE" forgrening benyttes sætningerne "WHEN" og "OTHERWISE" som udgange i forgreningen, hvor hver "WHEN"-udgang svarer til én eller flere bestemte værdier, hvorimod programmet vil anvende "OTHERWISE"-udgangen, hvis ikke der i "CASE" forgreningen har kunnet findes en "WHEN"-udgang, som passer til udtrykkets værdi. Der kan således være flere "WHEN"-udgange; men kun én "OTHERWISE"-udgang i en "CASE" forgrening. Det er ikke et krav, at der i en "CASE" forgrening findes en "OTHERWISE"-udgang, men hvis ingen af "WHEN"-udgangene er blevet benyttet, vil programudførelsen blive afbrudt med en fejludskrift, hvis "OTHERWISE" er blevet udeladt.

I "CASE" sætningen kan stå enten et taludtryk eller et strengudtryk. Alle udtryk i "WHEN" og "OTHERWISE" sætningerne skal være af samme type som udtrykket i "CASE" sætningen.

En "CASE" forgrening kan indeholde andre "CASE" forgreninger.

Se i øvrigt sætningen "IF ... THEN".

CASE

COMAL80 sætning

FORMAT

```
CASE <udtryk> OF
  WHEN <udtryk> {,<udtryk>}
   {<sætning>}
  OTHERWISE
   {<sætning>}
ENDCASE
```

CHR$

ANVENDELSE

COMAL80 funktionen "CHR$" anvendes ofte i forbindelse med COMAL80 sætningen "PRINT". "CHR$" angiver det tegn, der svarer til den ASCII-værdi, som står anført i <taludtryk>. ASCII- værdier er entydige værdier datamaskinen bruger til repræsentation af tegn.

F.eks. svarer ASCII-værdi "65" til det store bogstav -A-, hvorimod ASCII-værdi "97" svarer til det lille bogstav -a-.

En ASCII-værdi kan tillige svare til et tegn, der ikke er direkte synligt, men har en bestemt virkning. F.eks. svarer ASCII-værdi "12" til, at datamaskinen sletter hele skærmbilledet, og placerer datamaskinens lysende pegepind i øverste hjørne.

ASCII-alfabetet har 128 forskellige værdier (0-127).

"CHR$" funktionen kan imidlertid angive værdier mellem 0 og 255, idet værdier større end 127 har specielle betydninger (de anvendes f.eks. til at få blinkende inverterede udskrifter på skærmen).

Se i øvrigt funktionen ORD samt afsnittet om datamaskinens eget "alfabet".

CHR$

COMAL80 funktion

FORMAT

CHR$(<taludtryk>)

- CHR$ -
Angiver det tegn, som svarer til ASCII-værdien i <taludtryk>.

Eksempel

```
0010 PRINT CHR$(12),CHR$(7)
```

CLOSE

ANVENDELSE

COMAL80 sætningen "CLOSE" lukker en datastrøm, således at strømmen kan knyttes til en anden fil eller ydre enhed.

Lukningen medfører, at de resterende tegn bliver skrevet ud, hvis strømmen har været anvendt til skrivning.

Anvendes "CLOSE" uden strømnummer, lukkes alle strømme.

Kommandoerne "NEW" og "LOAD" lukker alle åbne strømme.

Se i øvrigt sætningen "OPEN".

CLOSE

COMAL80 kommando

COMAL80 sætning

FORMAT

CLOSE [<strømnummer>]

<strømnummer> ::= <taludtryk>

- CLOSE -

Anvendes til at lukke en datastrøm. Anføres "CLOSE" alene uden strømnummer bevirker det en lukning af alle åbne strømme i programmet.

Eksempel

```
0010 OPEN 1, "datafil", READ
0020 sluttegn:= 999
0030 READ FILE 1: a
0040 WHILE NOT EOF(1) DO
0050   PRINT a
0060   IF a = sluttegn THEN GOTO udhop
0070   READ FILE 1: a
0080 ENDWHILE
0090 udhop:
0100 CLOSE 1
```

CON

ANVENDELSE

COMAL80 kommandoen "CON" fortsætter udførelsen af et program, der er blevet midlertidigt stoppet af sætningen "STOP", eller ved tryk på 'ESC'.

Kommandoen "CON" fortsætter programudførelsen fra sætningen efter "STOP" eller fra den sætning, der følger umiddelbart efter den sidste sætning, der blev udført, før 'ESC' blev anvendt.

Før udførelsen fortsættes kan variables værdier inspiceres eller ændres, men der kan ikke slettes eller indføjes linier i programmet. "CON" kan altså kun fortsætte et program, der ikke er ændret.

Se i øvrigt COMAL80 sætningen "STOP".

CON

COMAL80 kommando

FORMAT

CON

- CON -
Anvendes til at fortsætte en programudførelse som er midlertidigt stoppet af programsætningen "STOP" eller ved tryk på 'ESC'.

COS

ANVENDELSE

COMAL80 funktionen "COS" udregner den trigonometriske funktion -cosinus- til en vinkel, der er udtrykt i radianer.

$$1 \text{ radian} = \frac{360 \text{ grader}}{2 \times \text{pi}} = 57{,}29577951309 \text{ grader}$$

$$\text{pi} = 3{,}141592653589$$

COMAL80 funktionen skrives som: COS(<taludtryk>)
Matematisk symbol skrives som : cos <taludtryk>

Se i øvrigt funktionerne "SIN", "TAN" samt "ATN".

COS

COMAL80 funktion

FORMAT

COS(<taludtryk>)

- COS -

Funktionen udregner cosinus til en vinkel udtrykt i radianer.

CREATE

ANVENDELSE

COMAL80 sætningen "CREATE" indsætter et filnavn i en diskettes katalog (indholdsfortegnelse) og reserverer plads til filen. En fil er på disketten opdelt i enheder af 512 tegn - sider. I "CREATE" sætningen angives, hvor mange sider der skal afsættes til filen.

Hvis der under en skrivning ikke er mere plads i filen, vil den automatisk blive udvidet.

Navngivningen af filer er beskrevet under sætningen "PREFIX".

CREATE

COMAL80 kommando

COMAL80 sætning

FORMAT

CREATE <filnavn>,<størrelse>

<filnavn> ::= <strengudtryk>
<størrelse ::= <taludtryk>

- CREATE -

Anvendes til at oprette en tom fil på et baggrundslager (diskette). Filen vil indeholde så mange sider à 512 tegn som angivet i <størrelse>.

Eksempel

```
0010 CREATE "datafil", 1
0020 OPEN 1, "datafil", WRITE
0030 REPEAT
0040   INPUT a
0050   WRITE FILE 1: a
0060 UNTIL a = 999
0070 CLOSE 1
```

DATA

ANVENDELSE

Et program skal benytte data i sine udregninger. Disse data kan stå som konstanter i programmet, eller de kan indlæses fra gang til gang. I mange tilfælde kan det også være hensigtsmæssigt at anbringe data i selve programmet, samlet på et sted.

I COMAL80 sætningen "DATA" placeres de opbevarede data, som tal eller strengkonstanter adskilt med komma'er (,). Programsætningen "DATA" benyttes ALTID sammen med sætningen "READ", således at "READ" sætningen læser de enkelte dataelementer i "DATA" sætningen et for et og tildeler disse til de variable, der står anført i "READ" sætningen en for en. "READ" sætningens variable får tildelt værdier efter samme regler som ved tildeling af værdier til tal- og strengvariable (f.eks. kun reelle tal og fortegn i talvariable).

Et program kan indeholde flere "DATA" sætninger, som tilsammen bliver opfattet som én stor liste af data, i den rækkefølge de står angivet i "DATA" sætningerne.

Systemet vedligeholder en "pegepind", som altid peger på det næste dataelement, der skal læses. Fra starten peges på det første element i den første "DATA" sætning.

Hver gang programmet indlæser et dataelement fra "DATA" sætningen flyttes pegepinden til næste dataelement. Hvis det læste dataelement er det sidste dataelement i én "DATA" sætning, fortsættes med det første dataelement i den næste "DATA"-sætning.

Man kan gentage indlæsningen af bestemte dataelementer, uden at indlæse de resterende dataelementer. Sætningen "RESTORE" anvendes til at flytte pegepinden hen til det <u>første</u> dataelement i den <u>første</u> sætning, eller til det <u>første</u> dataelement i den bestemte "DATA" sætning, som kommer efter en bestemt "Etikette" sætning.

Se i øvrigt sætningerne "READ", "RESTORE" samt "Etikette", samt funktionen EOD.

DATA

COMAL80 sætning

FORMAT

DATA <konstant> {,<konstant>}

- DATA - Anvendes til at opbevare data i selve programmet.

Eksempel

```
0010 DIM t$ OF 10
0020 READ i,j,k,t$
0030 PRINT i;j;k;t$
0040 DATA 1,2,3,"Tekst"
```

DEL

ANVENDELSE

COMAL80 kommandoen "DEL" anvendes til at slette programlinier i et program i arbejdsarealet.

Man kan som vist nedenfor kombinere anvendelsen af <startlinienummer> og <slutlinienummer> i <grænsenumre>.

DEL <startlinienummer> sletter linien <startlinienummer>

DEL <startlinienummer> sletter fra og med <startlinienummer> og programmet ud.

DEL <startlinienummer>,<slutlinienummer>
sletter programlinier fra og med <startlinienummer>, til og med <slutlinienummer>.

DEL ,<slutlinienummer>
sletter programlinier fra start til og med <slutlinienummer>.

DEL

COMAL80 kommando

FORMAT

```
DEL <grænsenumre>
  <grænsenumre>       ::= <startlinienummer>|<startlinienummer>,|
                          <startlinienummer>,<slutlinienummer>|
                          ,<slutlinienummer>
  <startlinienummer> ::= <taludtryk>
  <slutlinienummer> ::=  <taludtryk>
```

- DEL -

Anvendes til at slette en eller flere programlinier i det program, der ligger i programarealet.

Eksempel

Indlæs programmet "test" og slet linie nummer 140

```
  load "test"
  del 140
  list
                ..
       medfører 0130
                0150
                ..
```

Slet programlinierne fra linie nummer 170 og frem til slut.

```
  del 170       0100
  list
       medfører 0110
                ..
                0160
```

DELETE

ANVENDELSE

COMAL80 sætningen "DELETE" sletter det angivne navn i en disketes katalog og frigiver den plads, der har været afsat til filen.

DELETE

COMAL80 kommando

COMAL80 sætning

FORMAT

DELETE <filnavn>

<filnavn> ::= <strengudtryk>

- DELETE -

Anvendes til at slette en fil i baggrundslagerets katalog.

Eksempel

```
SELECT OUTPUT "printer"
LIST
DELETE "programfil"
SAVE "programfil"
```

DIM

ANVENDELSE

For at programmet kan benytte en variabel, skal man afsætte plads i arbejdsarealet til variablen, dog vil COMAL80 systemet selv afsætte plads i arbejdsarealet hvis det drejer sig om en simpel talvariabel. En simpel talvariabel er en variabel der kun indeholder et enkelt tal (max. 13 cifre). I alle andre tilfælde end ved talvariable skal der afsættes plads i arbejdsarealet for variablen ved hjælp af en "DIM"-sætning.

COMAL80 skelner mellem variable der udelukkende indeholder reelle tal og variable der indeholder både tal, tegn og bogstaver.

Taltabeller

Talsæt eller taltabeller er samlet om et fælles symbol. F.eks. er talsættet -A(I)-, hvor I antager værdierne 1, 2, 3, 4, 5 samlet om det fælles variabelsymbol -A-, eller sagt på en anden måde: En én-dimensional taltabel med 5 tal. Man fortæller programmet hvor stor en given tabel skal være, ved at dimensionere variablen med et indeks. -A(5)- betyder at der afsættes plads i arbejdsarealet til en én-dimensional taltabel (indiceret variabel), der kan indeholde indtil 5 reelle tal.

En to-dimensional taltabel er ligeledes samlet om et fælles variabel-symbol. F.eks. er talvariablen -B(I,K)-, hvor I antager værdierne 1, 2, 3, 4 og K antager værdierne 1, 2, 3; en to-dimensional taltabel med 4 rækker og 3 søjler med ialt 12 elementer. Man fortæller programmet hvor stor en given to-dimensional taltabel skal være, ved at dimensionere talvariablen med to indices. -B(4,3)- betyder, at der afsættes plads i arbejdsarealet til en to-dimensional taltabel (indiceret variabel) der kan indeholde indtil 12 reelle tal fordelt på 4 rækker og 3 søjler.

DIM

Strengvariable og strengtabeller

Der skal altid afsættes plads i arbejdsarealet til strengvariable og teksttabeller, før programmet kan anvende disse.

"DIM VARE$ OF 15" betyder, at der afsættes plads i arbejdsarealet til en strengvariabel VARE$, der kan indeholde indtil 15 tegn. En tabel af strenge er ligeledes samlet om et fælles symbol for tabellen. -KUNDE$(I)-, hvor I antager værdierne 1, 2, 3, 4 er samlet om det fælles variabelsymbol -KUNDE$-; men det fortæller intet om hvor store disse strenge er. Man skal derfor altid dimensionere strengtabellen både i antal og længde. DIM KUNDE$(4) OF 30 betyder, at der afsættes plads i arbejdsarealet til en strengtabel, der indeholder 4 strenge, som hver især kan indeholde op til 30 tegn.

BEMÆRK

Taltabeller, teksttabeller og strengvariable må ikke have samme "navne" som andre variable eller procedurer i programmet. Maksimumstørrelsen for en indiceret variabel eller strengvariabel er kun begrænset af arbejdsarealets størrelse. En variabel må kun defineres én gang. Ved en dimensionering bliver taltabeller nulstillet, hvorimod strengvariabel og strengtabeller defineres som "nulstrenge" uden indhold og af længden nul.

Det er tilladt at blande dimensionering af taltabeller, strengvariable og strengtabeller i en DIM-sætning.

DIM

OPBYGNING

--DIM A(5) — Dimensionér en én-dimensional taltabel med 5 tal
A(1) A(2) A(3) A(4) A(5)

--DIM B(4,3) — Dimensionér en to-dimensional taltabel med 4 rækker og 3 søjler.

B(1,1) B(1,2) B(1,3)
B(2,1) B(2,2) B(2,3)
B(3,1) B(3,2) B(3,3)
B(4,1) B(4,2) B(4,3)

-- DIM VARE$ OF 15 Dimensionér en strengvariabel, der kan indeholde op til 15 tegn.

-- DIM KUNDE$(4) OF 30 Dimensionér en strengtabel, der kan indeholde op til 4 strenge, som hver især kan indeholde op til 30 tegn.

KUNDE$(1) KUNDE$(2) KUNDE$(3) KUNDE$(4).

COMAL80 kommando
COMAL80 sætning

FORMAT

DIM <erklæring> { ,<erklæring> }

<erklæring> ::= <talvariabel>(<taludtryk>)|
<taltabel>(taludtryk>, <taludtryk>)|
<streng> OF <taludtryk>|
<teksttabel>(<taludtryk>) OF <taludtryk>
<taltabel> ::= <navn>
<streng> ::= <navn>
<teksttabel> ::= <navn>

- DIM -
Anvendes til at afsætte plads i arbejdsarealet for:
taltabeller, strengvariable og teksttabeller.

Eksempel

```
0010 // Taltabel
0020 DIM a(5)
0030 FOR i:= 1 TO 5 DO a(i):= i
0040
0050 // Todimensional taltabel
0060 DIM b(5,5)
0070 FOR i:= 1 TO 5 DO
0080   FOR j:= 1 TO 5 DO
0090     b(i,j):= i+j
0100   NEXT j
0110 NEXT i
0120
0130 // Strengvariabel
0140 DIM s$ OF 10
0150 s$:= "Tekst"
0160
0170 // Teksttabel
0180 DIM t$(5) OF 10
0190 FOR i:= 1 TO 5 DO t$(i):= "Tekst" + CHR$(i+48)
```

DIR

ANVENDELSE

COMAL80 sætningen "DIR" udskriver navnene på de filer, der er anført i kataloget (indholdsfortegnelsen) på et bestemt baggrundslager (diskette). Normalt udskrives indholdsfortegnelsen på dataskærmen, hvis ikke der i det kørende program eller tidligere, har været anført COMAL80 sætningen "SELECT OUTPUT" til en anden ydre enhed end dataskærmen.

DIR

COMAL80 kommando
COMAL80 sætning

FORMAT

DIR <disketteenhed>
<disketteenhed> ::= <taludtryk>

- DIR -
Anvendes til at udskrive navnene på de filer der eksisterer i kataloget på et baggrundslager (diskette).

DISMOUNT

ANVENDELSE

COMAL80 sætningen "DISMOUNT" afbryder forbindelsen mellem filsystemet og den diskette der er placeret i den disketteenhed, hvis nummer er anført i "DISMOUNT" sætningen, ved at lukke katalogfilen på disketten, således at disketten kan udtages af disketteenheden.
Anfører brugeren en ny "DISMOUNT" sætning med samme disketteenhed, uden forinden at have anvendt en tilsvarende "MOUNT" filsætning, vil COMAL80 udskrive en fejlmeddelelse.

Se i øvrigt COMAL80 sætningen "MOUNT".

DISMOUNT

COMAL80 kommando
COMAL80 sætning

FORMAT

```
DISMOUNT <disketteenhed>
<disketteenhed> ::= <taludtryk>
```

- DISMOUNT -
Anvendes til at lukke katalogfilen på disketten, der sidder i den anførte disketteenhed.

END

ANVENDELSE

"END"-sætningen benyttes til at markere afslutningen af et program.

Når en "END"-sætning mødes standser programudførelsen med udskriften

```
END
at XXXX
```

hvor XXXX er linienummeret på "END"-sætningen. Programmet behøver ikke at indeholde nogen "END"-sætning, da programudførelsen standser, når programmets sidste sætning (med det højeste linienummer) er udført.

Se i øvrigt COMAL80 sætningen "STOP".

END

COMAL 80 sætning

FORMAT

END

- END -

Anvendes til logisk at afslutte et program.

ENTER

ANVENDELSE

Med kommandoen "ENTER" kan brugeren sammenflette flere programmer.

"ENTER" anvendes med fordel, når et program skal sammensættes af programdele eller procedurer fra andre programmer, eller et program skal sammensættes af underprogrammer og procedurer fra et procedurebibliotek.

Bemærk følgende regler for linienummerering ved sammenfletning. Det eksisterende program (programmet der ligger i programarealet) har lavere prioritet end de programlinier, som ønskes indflettet i det eksisterende program. Det betyder, at hvis det eksisterende program og de programlinier, som ønskes indlfettet, har samme linienumre, vil programlinier i det eksisterende program blive slettet, og de nye programlinier indsat istedet.

Indeholder det eksisterende program og det indflettede program forskellige programlinienumre, vil det sammenflettede program bestå af programlinier fra både det eksisterende og det indflettede program.

"ENTER" indlæser programmer på ASCII-form, dvs. programmer, som indlæses, skal være udskrevet med "LIST" kommandoen.

Se i øvrigt kommandoen "LIST".

ENTER

COMAL80 kommando

FORMAT

ENTER <strengudtryk>

- ENTER -
Anvendes til at indflette andre programdele i det program, der er placeret i programarealet.

EOD

ANVENDELSE

COMAL80 funktionen "EOD" benyttes i forbindelse med sætningerne "READ" og "DATA".

Funktionen "EOD" er en logisk funktion som antager funktionsværdien SAND, når det sidste element i datalisten er læst, i alle andre tilfælde vil funktionsværdien være FALSK.

Funktionen "EOD" kan således være en nyttig funktion i forbindelse med betingede sætninger som "IF - THEN - ELSE" og programløkker som "REPEAT - UNTIL" og "WHILE".

Se i øvrigt sætningerne "READ", "DATA" og "RESTORE".

EOD

COMAL80 funktion

FORMAT

EOD

- EOD -

Er en logisk funktion som har funktionsværdien sand når det sidste element i datalisten er læst.

Eksempel

```
0010 RESTORE
0020 WHILE NOT EOD DO
0030   READ a
0040   PRINT a
0050 ENDWHILE
0060 DATA 1,2
```

EOF

ANVENDELSE

COMAL80 funktionen "EOF" benyttes i forbindelse med sætningerne "READ FILE" og "INPUT FILE".

Funktionen "EOF" er en logisk funktion, som antager funktionsværdien SAND, når den sidste post (record) i den datafil, hvis nummer er angivet ved <taludtryk>, er blevet læst, i alle andre tilfælde vil funktionsværdien være FALSK.

Funktionen "EOF" kan således være en nyttig funktion i forbindelse med betingede sætninger som "IF - THEN - ELSE", eller i programløkker som "REPEAT - UNTIL" og "WHILE - ENDWHILE".

Se i øvrigt sætningerne "READ FILE" og "INPUT FILE".

EOF

COMAL80 funktion

FORMAT

EOF (<strømnummer>)
<strømnummer> ::= <taludtryk>

- EOF -

Er en logisk funktion som antager funktionsværdien SAND, når den sidste post (record) i datafilen er læst.

Eksempel

```
0010 OPEN 1, "datafil", READ
0020 sluttegn:= 999
0030 READ FILE 1: a
0040 WHILE NOT EOF(1) DO
0050   PRINT a
0060   IF a = sluttegn THEN GOTO udhop
0070   READ FILE 1: a
0080 ENDWHILE
0090 udhop:
0100 CLOSE 1
```

Etikette

ANVENDELSE

En "Etikette", er en programsætning uden nøgleord. Ved hjælp af en etikette (engelsk: label) kan man udpege et bestemt sted i programmet, hvortil der kan referes ved hjælp af sætningerne "GOTO" og "RESTORE".

Normalt udføres et program linie for linie startende med den første programlinie og sluttende med den sidste programlinie. Der kan være situationer, hvor brugeren ikke ønsker, at programmet skal følge reglen om den fortløbende programudførelse; men derimod fortsætte sin programudførelse fra et nærmere fastsat sted i programmet. Dette kan opnås ved, at man benytter sætningen GOTO efterfulgt af navnet på den etikette, der står umiddelbart før den næste sætning, der ønskes udført.

Se i øvrigt COMAL80 sætningerne "IF", "RESTORE" samt "GOTO".

Etikette

COMAL80 sætning

FORMAT

<etikette>:
<etikette> ::= <navn>

- Etikette -

Anvendes til at navngive et bestemt sted i programmet hvortil andre programsætninger kan referere.

Etiketter navngives efter samme regler som talvariable.

Eksempel

```
0010 OPEN 1, "datafil", READ
0020 sluttegn:= 999
0030 READ FILE 1: a
0040 WHILE NOT EOF(1) DO
0050   PRINT a
0060   IF a = sluttegn THEN GOTO udhop
0070   READ FILE 1: a
0080 ENDWHILE
0090 udhop:
0100 CLOSE 1
```

EXEC

ANVENDELSE

"EXEC" sætningen benyttes til kald af en COMAL80 procedure, som er erklæret i en "PROC" sætning med det <navn>, som står efter nøgleordet "EXEC".

Hvis proceduren er erklæret med parametre, må de aktuelle parametre i "EXEC"-sætningen svare nøje til de formelle parametre i "PROC"-sætningen både i antal og i type.

Værdiparametre kan være en variabel eller et udtryk, mens referenceparametre skal være et variablenavn. En-dimensionale talsæt og strengtabeller angiver ved hjælp af "()" efter variabelnavnet. To-dimensionale talsæt angives ved hjælp af "(,)" efter variabelnavnet.

Efter udførelsen af proceduren fortsættes med sætningen efter "EXEC"-sætningen.

Se i øvrigt COMAL80 sætningen "PROC".

EXEC

COMAL80 sætning

FORMAT

EXEC <navn> [(<parameterliste>)]

<parameterliste> ::= <parameter> {,<parameter>}

<parameter> ::= <udtryk>

- EXEC -

Anvendes til at aktivere en procedure erklæret med programsætningen "PROC".

Eksempel

```
0010 PROC find (a) CLOSED
0020   IF a <> 1 THEN
0030     faktor:= 2; rod:= SQR(a)
0040     WHILE a MOD faktor <> 0 AND faktor <= rod DO faktor:= faktor+1
0050     IF a MOD faktor <> 0 THEN faktor:= a
0060     PRINT faktor,
0070     EXEC find (a DIV faktor)
0080   ENDIF
0090 ENDPROC find
0100
0110 ZONE 4
0120 INPUT "Tal:       ": a
0130 PRINT "Faktorer:  ",
0140 EXEC find (a)
```

EXP

ANVENDELSE

COMAL80 funktionen "EXP" udregner tallet -e- (=2,718281828458) opløftet til en potens, givet ved den værdi, der står anført i (<taludtryk>).

COMAL80 funktion skrives som : EXP(<taludtryk>)

Matematisk symbol skrives som: $e^{\text{<taludtryk>}}$

Se i øvrigt funktionen "LOG".

EXP

COMAL80 funktion

FORMAT

EXP (<taludtryk>)

- EXP -

Anvendes til at udregne talstørrelsen -e- opløftet til den potens, der er givet ved (<taludtryk>).

FOR

ANVENDELSE

Brugeren kan få gentaget udførelsen af en eller flere programlinier et bestemt antal gange, ved at benytte COMAL80 sætningen "FOR" eller den udvidede sætning "FOR - NEXT". I den udvidede programsætning "FOR - NEXT" omsluttes de programlinier, der ønskes gentaget, af sætningerne "FOR" og "NEXT". En sådan gentagelse af programlinier kaldes en LØKKE.

Tællevariablen vil under udførelsen gennemløbe en række tal, bestemt af startværdi, slutværdi og trinværdi. Hvis der ikke er angivet nogen trinværdi antages værdien 1.

Når "FOR" sætningen udføres sættes tællevariablen lig startværdien. Herefter udføres løkkens sætninger, dvs. sætningen efter "THEN" i den simple "FOR" sætning eller sætningerne mellem "FOR" og "NEXT" i den udvidede "FOR-NEXT" løkke. Dette sker dog kun hvis følgende betingelser er opfyldt:

tællevariabel <= slutværdi (trinværdi er positiv)
eller
tællevariabel >= slutværdi (trinværdi er negativ)

Er betingelsen ikke opfyldt fra starten, vil løkken altså slet ikke blive udført. Efter hvert gennemløb af løkken bliver trinværdien adderet til tællevariablen, og løkken udføres igen, hvis ovenstående betingelse er opfyldt.

Løkken gentages indtil betingelsen ikke er opfyldt. Udførelsen fortsætter da med sætningen efter den simple "FOR" sætning eller sætningen efter "NEXT". Bemærk, at tællevariablen ikke behøver at antage slutværdien.

Det er tilladt at ændre tællevariablens værdi inde i løkken. Dette skal dog ske med omtanke. En "FOR - NEXT" løkke kan indeholde andre "FOR - NEXT" løkker, blot må de enkelte "FOR - NEXT" løkker ikke overlappe hinanden.

Se i øvrigt COMAL80 sætningerne "REPEAT" og "WHILE ... DO".

FOR

COMAL80 sætning

FORMAT

Simpel:

```
FOR <tællevariabel> := <startværdi> TO <slutværdi>
                       [step <trinværdi] DO <simpel sætning>
```

Udvidet:

```
FOR <tællevariabel> ::= <startværdi> TO <slutværdi>
                        [step <trinværdi>] DO
  {<sætning>}
NEXT <tællevariabel>
<tællevariabel> ::= <talvariabel>
<startværdi> ::=    <taludtryk>
<slutværdi> ::=     <taludtryk>
<trinværdi> ::=     <taludtryk>
```

Bemærk at den simple form altid skal skrives på én linie, og at dette også gælder udtrykket FOR ... TO ... STEP ... DO i den udvidede form.

Eksempel

```
0010 INPUT a
0020 FOR i:= a TO 1 STEP -1 DO
0030   PRINT TAB(i);
0040   FOR j:= a TO 1 STEP -1 DO
0050     PRINT "**";
0060   NEXT j
0070   PRINT // linieskift
0080 NEXT i
```

GET$

ANVENDELSE

Funktionen "GET$" anvendes til at læse et bestemt antal tegn fra en fil på baggrundslager eller en ydre enhed, angivet ved hjælp af <strømnummer>. Sætningerne "READ FILE" og "INPUT FILE" kræver, at de læste tegn er ordnet på en bestemt måde. Hvor dette ikke er tilfældet kan "GET$" med fordel anvendes.

Det angivne <strømnummer> skal angive en strøm, som i forvejen er åbnet ved hjælp af en "OPEN" sætning.

"GET$" kan indgå i strengudtryk på linie med funktionen "CHR$" og kan tildeles til strengvariable eller udskrives i en "PRINT" sætning.

Kan det angivne antal tegn ikke læses, vil "EOF" funktionen blive SAND. Det faktiske antal læste tegn kan da findes ved hjælp af funktionen "LEN".

Se i øvrigt sætningen "OPEN".

GET$

COMAL80 funktion

FORMAT

GET$ (<strømnummer>,<antal>)

<strømnummer> ::= <taludtryk>
<antal> ::= <taludtryk>

- GET$ -
Læser et antal tegn fra en åben strøm.

GLOBAL

ANVENDELSE

"GLOBAL" sætningen anvendes til at gøre globale variable tilgængelige i en lukket procedure.

Normalt er alle variable i en lukket procedure lokale for proceduren, men man kan ved hjælp af "GLOBAL" sætningen angive, at visse variabelnavne skal betegne, ikke en lokal variabel, men den globale variabel med dette navn. Dette kan især være nyttigt, hvis man fra en lukket procedure ønsker at kalde en anden lukket procedure, idet procedurer ikke kan overføres som parametre.

De variable, der anføres i "GLOBAL" sætningen skal være kendte når proceduren kaldes, dvs. simple talvariable skal være tildelt en værdi, og simple strenge samt tal- og strengtabeller skal være dimensionerede.

Se i øvrigt sætningen "PROC".

GLOBAL

COMAL80 sætning

FORMAT

GLOBAL <variabelliste>

<variabelliste>::= <navn>{,<navn>}

- GLOBAL -
Ændrer status for variable eller procedurer, i en lukket procedure fra at være lokale, til at være globale.

GOTO

ANVENDELSE

COMAL80 sætningen "GOTO" benyttes til at dirigere programudførelsen til et nærmere angivet sted i programmet og derfra fortsætte programudførelsen.

Normalt udføres et program linie for linie startende med den første programlinie og sluttende med den sidste programlinie. Under særlige omstændigheder kan det være nødvendigt at fravige reglen om fortløbende udførelse af programmet.

Det er ikke tilladt at hoppe ud af procedurer!

Navnet i "GOTO" sætningen refererer til en "Etikette" med det samme <navn>.

Se i øvrigt sætningerne "Etikette", "RESTORE" samt IF ... THEN.

GOTO

COMAL80 sætning

FORMAT

GOTO <etikette>

<etikette> ::= <navn>

- GOTO -

Anvendes til at fortsætte programudførelsen fra den sætning, der følger efter den etikette, der er angivet.

Eksempel

```
0010 OPEN 1, "datafil", READ
0020 sluttegn:= 999
0030 READ FILE 1: a
0040 WHILE NOT EOF(1) DO
0050   PRINT a
0060   IF a = sluttegn THEN GOTO udhop
0070   READ FILE 1: a
0080 ENDWHILE
0090 udhop:
0100 CLOSE 1
```

IF

ANVENDELSE

HVIS et givet udsagn er opfyldt, SÅ skal programmet udføre en eller flere bestemte programlinier, ELLERS skal programmet udføre andre bestemte programlinier. Til en sådan "HVIS ... SÅ ... ELLERS" betinget konstruktion benyttes COMAL80 sætningerne:

- IF ... THEN
- ELSE
- ENDIF

En betinget konstruktion af denne type kan inddeles i tre typer:

- Den simple betingede "IF ... THEN" sætning
- Den udvidede betingede "IF ... THEN"
  "ENDIF" konstruktion
- Den forgrenede betingede "IF ... THEN"
  "ELSE"
  "ENDIF" konstruktion

Om en betinget sætning eller konstruktion kan man sige, at der findes en ja-udgang og en nej-udgang, svarende til at ja-udgangen benyttes når udsagnet er SANDT (udsagnet er opfyldt), og nej-udgangen benyttes når udsagnet er FALSK (udsagnet er ikke opfyldt).

Den simple "IF ... THEN" sætning:
Består kun af sætningen "IF ... THEN", og kan anvendes når udsagnet og de opgaver sætningen skal udføre, kan skrives i én sætning.

Er udsagnet SANDT, udføres sætningen.
Er udsagnet FALSK, udføres sætningens opgaver IKKE, og programudførelsen fortsætter med den programlinie, der kommer efter "IF ... THEN" sætningen.

IF

Den udvidede "IF ... THEN" - "ENDIF" konstruktion:
Består af to sætninger - "IF ... THEN" og "ENDIF", som omslutter de linier, der bliver udført, når udsagnet er SANDT.

Udsagnet undersøges:
Er udsagnet SANDT, udføres de programlinier, der er omsluttet af "IF ... THEN" sætningen og "ENDIF" sætningen.
Er udsagnet FALSK udføres ingen af de omsluttede programlinier, og programmet fortsætter med den programlinie der kommer efter "ENDIF" sætningen.

En "IF ... THEN" - "ENDIF" konstruktion kan indeholde andre "IF ... THEN" - "ENDIF" konstruktioner. Bemærk, at der til hver "IF ... THEN" sætning hører en "ENDIF" sætning.

Den forgrenede "IF ... THEN" - "ELSE" - "ENDIF" konstruktion:
Består af tre sætninger: "IF ... THEN", "ELSE" og "ENDIF", hvor sætningerne "IF ... THEN" og "ELSE" omslutter de programlinier, der bliver udført når udsagnet er SANDT, og sætningerne "ELSE" og "ENDIF" omslutter de programlinier, der bliver udført når udsagnet er FALSK.

Udsagnet undersøges:
Er udsagnet SANDT, udføres programlinierne, der er omsluttet af "IF ... THEN" og "ELSE" sætningerne, og programmet fortsætter efter den tilhørende "ENDIF" sætning.
Er udsagnet FALSK, udføres programlinierne, der er omsluttet af "ELSE" og "ENDIF" sætningerne, og programmet fortsætter efter den tilhørende "ENDIF" sætning.

Bemærk at der i begge grene, - imellem "IF ... THEN" og "ELSE" og imellem "ELSE" og "ENDIF", kan findes andre betingede "IF ... THEN" konstruktioner, både de udvidede og de forgrenede konstruktioner. Det kan i mange tilfælde være en hjælp at skrive kommentarer i de betingede sætninger "IF ... THEN", "ELSE" og "ENDIF". Bemærk at der til hver "IF ... THEN" sætning hører en "ENDIF" sætning.

IF

COMAL80 sætning

FORMAT

Simpel:

```
IF <logisk udtryk> THEN <simpel sætning>
```

Udvidet:

```
IF <logisk udtryk> THEN
  {<sætning>}
ENDIF
```

Forgrenet:

```
IF <logisk udtryk> THEN
  {<sætning>}
ELSE
  {<sætning>}
ENDIF
```

- IF -

Udfører en eller flere sætninger afhængigt af, om et logisk udtryk er SANDT eller FALSK.

Eksempel

```
0010 // Simpel
0020 INPUT i,j
0030 PRINT "De to tal er ";
0040 IF i <> j THEN PRINT "ikke ";
0050 PRINT "ens"
0060
0070 // Udvidet
0080 INPUT i,j
0090 IF i <> j THEN
0100   PRINT "De to tal er ikke ens."
0110   PRINT "Forskellen er "; ABS(i-j)
0120 ENDIF
0130
0140 // Forgrenet
0150 INPUT i
0160 PRINT "Tallet er ";
0170 IF SGN(i) >= 0 THEN
0180   PRINT "positivt"
0190 ELSE
0200   PRINT "negativt"
0210 ENDIF
```

INPUT

ANVENDELSE

Man kan indlæse værdier til et program ved at benytte sætningen "INPUT". COMAL80 skelner imellem indlæsning i talvariable og taltabeller og indlæsning i strengvariable og strengtabeller. Der kan kun indlæses tal, cifre, fortegn, decimalpunktum samt eksponent i talvariable eller taltabeller.

Hvis der i kaldet af "INPUT" er angivet en <ledetekst>, skrives denne ud. For hver <variabel>, der er angivet, stopper programudførelsen, og der udskrives et ventetegn (:), og programmet fortsætter først, når der er indtastet data, og man har trykket på RETURN-tasten. Hvis der som en <variabel> er angivet en <talvariabel>, skal de indtastede data være et tal. I modsat fald nægter programmet at godkende data og udskriver ventetegnet igen for at modtage nye data.

Tegn som skal indlæses i strengvariable og strengtabeller indlæses uden anførselstegn (").

Man kan indtaste flere reelle tal i samme "INPUT" sætning ved blot at taste et mellemrum mellem hvert af de reelle tal.

Et kald af "INPUT" bør ikke indeholde blandede tal- og strengvariable; men består af enten talvariable og taltabeller, eller strengvariable og strengtabeller. Afsluttes kaldet med et (;) vil uddata fra en efterfølgende "PRINT" sætning blive udskrevet i fortsættelse af det indtastede til "INPUT" sætningen.

INPUT

COMAL80 sætning

FORMAT

INPUT [<ledetekst>:] <variabel> {,<variabel>} [;]
<ledetekst> ::= <strengkonstant>

- INPUT -

Anvendes til at indlæse værdier indtastet fra tastaturet til variable.

Eksempel

```
0010 INPUT i
0020 PRINT i
0030
0040 DIM s$ OF 10
0050 INPUT "Tast en tekststreng: ": s$
0060 PRINT s$
0070
0080 PRINT "Kvadratroden af";
0090 INPUT "": i;
0100 PRINT " er lig med "; SQR(i)
```

INT

ANVENDELSE

COMAL80 funktionen "INT" beregner det største heltal som ikke er større end (<taludtryk>), hvilket betyder at "INT" ikke benytter normale op- og nedrundingsprincipper, men altid nedrunding til nærmeste heltal.

COMAL80 funktion skrives som: INT (<taludtryk>)
Matematisk symbol : $\lfloor$<taludtryk>$\rfloor$

INT

COMAL80 funktion

FORMAT

INT(<taludtryk>)

- INT -

Anvendes til at beregne det nærmeste heltal der ikke er større end det angivne <taludtryk>.

LEN

ANVENDELSE

COMAL80 funktionen "LEN" angiver det faktiske antal tegn i en strengvariabel, eller antallet af tegn mellem to (") anførselstegn (strengkonstant). Når en strengvariabel dimensioneres, afsættes der plads i arbejdsarealet til hele strengvariablen. Det er imidlertid ikke sikkert, at en strengvariabel altid er fyldt helt ud med tegn. Brugeren kan på et vilkårligt tidspunkt få opgivet det faktiske antal af tegn i en strengvariabel eller i en strengkonstant, ved at benytte funktionen "LEN".

Se i øvrigt COMAL80 sætningen "DIM".

LEN

COMAL-funktion

FORMAT

LEN(<strengudtryk>)

- LEN -

Anvendes til at undersøge antallet af tegn i et strengudtryk.

Eksempel

```
0010 DIM s$ OF 50
0020 INPUT "Tast tekststreng: ": s$
0030 tæller:= 0
0040 FOR i:= 1 TO LEN(s$) DO
0050   IF s$(i:1) IN "0123456789" THEN tæller:= tæller + 1
0060 NEXT i
0070 PRINT "Tekststrengen indeholder "; tæller; "cifre."
```

LIST

ANVENDELSE

COMAL80 kommandoen "LIST" udskriver fra det program, der ligger i programarealet, hele programmet eller dele heraf, og lister dette på den ydre enhed som er blevet anført i programsætningen "SELECT OUTPUT".

Har brugeren ikke anført en "SELECT OUTPUT" programsætning, vil programmet automatisk udskrive programdelen på dataskærmen.

Man kan som vist nedenfor kombinere anvendelsen af <startlinienummer> og <slutlinienummer>.

LIST	Udskriver hele programmet.
LIST <startlinienummer>	Udskriver programlinien <startlinienummer>.
LIST <startlinienummer>,	Udskriver fra og med programlinien <startlinienummer> til og med sidste programlinie.
LIST <startlinienummer>,<slutlinienummer>	Udskriver fra og med programlinie <startlinienummer> til og med programlinie <slutlinienummer>.
LIST ,<slutlinienummer>	Udskriver fra og med første programlinie til og med programlinie <slutlinienummer>.

LIST

COMAL80 kommando

FORMAT

```
LIST [<grænsenumre>]
<grænsenumre> ::= <startlinienummer>|
                  <startlinienummer>,|
                  <startlinienummer>,<slutlinienummer>|
                  ,<slutlinienummer>
<startlinienummer> ::= <taludtryk>
<slutlinienummer>  ::= <taludtryk>
```

- LIST -

Anvendes til at udskrive hele programmet eller dele heraf, ud på den valgte ydre enhed.

Eksempel

Indlæs programmet "test", og list hele programmet:

```
  load "test"
  list
                0100 .....
      medfører  0110 ......
                 ..
                0240 ......
```

List programlinierne mellem line nummer 100 og linie 200

```
  list 100,200
                0100 ....
      medfører  0110 .....
                 ..
                0200 ....
```

List programlinierne fra start frem til linie nummer 170

```
  list ,170
                0100 ...
      medfører  0110 .....
                 ..
                0170 ...
```

LOAD

ANVENDELSE

COMAL80 kommandoen "LOAD" overfører fra baggrundslageret (diskette) til programarealet, den programfil, hvis navn står anført i "LOAD" kommandoen. Den ønskede programfil skal tidligere været blevet gemt ("SAVE") under dette navn.

Programfilens navn må maksimalt bestå af 8 tegn.

"LOAD" udfører automatisk en NEW-kommando, som lukker evt. åbne filer.

LOAD

COMAL80 kommando

FORMAT

```
LOAD <strengkonstant>
```

- LOAD -

Anvendes til at overføre et program fra baggrundslageret til programarealet.

Eksempel

```
MOUNT 2
PREFIX "2/"
LOAD "programfil"
RUN
```

LOG

ANVENDELSE

COMAL80 funktionen "LOG" beregner den naturlige logaritme af <taludtryk>. Den naturlige logaritme har tallet -e- (= 2,718281828458) som grundtal, til forskel fra ti-tals logaritmen, der har tallet 10 som grundtal.

COMAL80 funktionen skrives som: LOG(<taludtryk>)
Matamatisk symbol skrives som : ln<taludtryk>

Se i øvrigt funktionen "EXP".

LOG

COMAL80 funktion

FORMAT

LOG(<taludtryk>)

- LOG -

Anvendes til at beregne den naturlige logaritme til et <taludtryk>.

MARGIN

ANVENDELSE

Brugeren kan selv bestemme længden af den udskrevne linie. I "MARGIN" sætningen anføres det antal tegn (positioner), der ønskes i den udskrevne linie.

Benytter brugeren ingen "MARGIN" sætninger i programmet vil programmet anvende en linielængde på 80 positioner (defaultværdi = "MARGIN 80"). Der er ingen højeste værdi for "MARGIN". Når antallet af udskrevne tegn overskrider "MARGIN"-værdien, fortsættes automatisk på næste udskriftslinie. "MARGIN" sætningen anvendes med fordel ved udskrivning på dataskærm og linieskriver.

Ved udskrivning af data i en datafil kan "MARGIN" sætningen ligeledes anvendes og i den forbindelse bør det nævnes at "MARGIN 0" betyder INTET LINIESKIFT, og at dataene derfor lægges efter hinanden i en endeløs række.

MARGIN

COMAL80 kommando

FORMAT

```
MARGIN <taludtryk>
```

- MARGIN -

Definerer antallet af tegn i udskriftslinien, inden der skiftes til ny linie.

MOUNT

ANVENDELSE

COMAL80 sætningen "MOUNT" sammenknytter filsystemet med den diskette der er placeret i den disketteenhed, hvis nummer er anført i "MOUNT" sætningen.

"MOUNT" bevirker en åbning af katalogfilen på den diskette der sidder i den disketteenhed hvis nummer fremgår af "MOUNT" filsætningen.

Udføres en ny "MOUNT" sætning på samme disketteenhed, uden forinden at have udført en tilsvarende "DISMOUNT" sætning, vil COMAL80 udskrive en fejlmeddelelse.

Ved opstarten udføres automatisk "MOUNT 1".

MOUNT

COMAL80 kommando

COMAL80 sætning

FORMAT

```
MOUNT <disketteenhed>
<disketteenhed> ::= <taludtryk>
```

- MOUNT -

Anvendes til at åbne katalogfilen på den diskette der sidder i den disketteenhed, hvis nummer er anført i "MOUNT" sætningen.

Eksempel

```
MOUNT 2
PREFIX "2/"
LOAD "programfil"
RUN
```

NEW

ANVENDELSE

Kommandoen "NEW" anvendes når brugeren ønsker at indtaste et nyt program. "NEW" fjerner det tidligere program fra programarealet, samt sletter dets dataarel. Tillige lukkes eventuelle åbne filer.

Linielængde og zonebredde (sat ved hjælp af "MARGIN" og "ZONE") ændres ikke.

"LOAD" kommandoen udfører automatisk "NEW".

NEW

COMAL80 kommando

FORMAT

NEW

- NEW -

Anvendes til at slette programareal og dataareal, samt lukke eventuelle åbne filer.

OPEN

ANVENDELSE

COMAL80 sætningen "OPEN" anvendes til at knytte en fil på baggrundslager eller en ydre enhed til en datastrøm, som benyttes ved skrivning og læsning. Det tilknyttede strømnummer identificerer filen eller den ydre enhed i sætninger til skrivning og læsning.

"OPEN" sætningen har også en parameter, der angiver hvorledes strømmen skal benyttes:

WRITE: skrivning
READ: læsning

En fil skal være oprettet, før den åbnes.

De ydre enheder, der kan angives i "OPEN" er

"console": skærm og tastatur
"printer": linieskriver

Se i øvrigt sætningen "PREFIX", hvor navngivningen af filer er beskrevet.

OPEN

COMAL80 kommando
COMAL80 sætning

FORMAT

```
OPEN <strømnummer>,<filnavn>,<mode>

<strømnummer> ::= <taludtryk>
<filnavn>     ::= <strengudtryk>
<mode>        ::= READ|WRITE
```

- OPEN -
Anvendes til at knytte et filnavn til en datastrøm. "OPEN" angiver tillige måden hvorpå strømmen benyttes.

ORD

ANVENDELSE

COMAL80 funktionen "ORD" angiver ASCII-værdien af det første tegn i en strengvariabel eller det første tegn mellem to anførselstegn (").

Funktionen "ORD" benyttes ofte sammen med den "omvendte" funktion "CHR$", som angiver hvilket tegn der svarer til en bestemt ASCII-værdi.

Se i øvrigt funktionen "CHR$" samt afsnittet om datamaskinens eget "alfabet".

ORD

COMAL80 funktion

FORMAT

```
ORD(<strengudtryk>)
```

- ORD -

Angiver ASCII-værdien for det første tegn i et <strengudtryk>.

PREFIX

ANVENDELSE

I COMAL80 består et fuldstændigt filnavn af et enhedsnummer efterfulgt af en "/" og det egentlige filnavn.
Ved hjælp af sætningen "PREFIX" kan man definere en tegnfølge, som sættes foran de filnavne, der angives i sætninger og kommandoer (et navnepræfix). Dette præfix tilføjes altid, med mindre det angivne filnavn indledes med "/".
Præfixet kan sættes til en vilkårlig tegnfølge, men oftest vil man sætte det til "1/" eller "2/", hvorved man ikke behøver at anføre enhedsnummeret i filnavne. Et program kan altså gøres uafhængigt af, hvilken enhed, der benyttes, hvis det relevante præfix sættes før programmet udføres.

Hvis præfix er "1/", må man for at få tilgang til filer på disketteenhed nummer 2 indlede filnavnet med "/2/".

Ved opstarten sættes præfix til "1/". Præfix forbliver uændret indtil en ny "PREFIX" sætning udføres. Det er tilladt at definere et tomt præfix ("").

PREFIX

COMAL80 kommando
COMAL80 sætning

FORMAT

```
PREFIX <navnepræfix>
<navnepræfix> ::= <strengudtryk>
```

- PREFIX -

Anvendes til at definere et præfix, der tilføjes filnavne angivet i sætninger og kommandoer.

Eksempel

```
MOUNT 2
PREFIX "2/"
LOAD "programfil"
RUN
```

PRINT

ANVENDELSE

COMAL80 sætningen "PRINT" anvendes til at danne udskrifter fra programmet. Parametrene til "PRINT" sætningen kan inddeles i 4 arter:

a: intet
b: tekst
c: variabel
d: taludtryk

De enkelte arter kan i vilkårligt antal, adskilt med skilletegn, anføres i en "PRINT" sætning.

"PRINT" sætningens <udtryk> udskrives altid venstrestillet, således at positive talværdier angives alene ved deres talværdi, hvorimod negative talværdier angives med fortegn og talværdi. Strengvariable venstrestilles ligeledes direkte. Rækkefølgen for udtrykkene i "PRINT" sætningen er afgørende for udskrivningen af udtrykkene. Det først angivne udtryk udskrives først, nummer to derefter osv., indtil alle parametre er udskrevet.

Anvendelse af skilletegnet komma (,)

En billedlinie på dataskærmen indeholder 80 positioner. Anvendes skilletegnet komma (,), inddeler COMAL80 billedliniens 80 positioner i afsnit, hvis bredde (kolonnebredde) er blevet defineret af COMAL80 sætningen "ZONE". Er "ZONE" værdien ikke blevet fastsat på noget tidspunkt i det kørende program eller siden sidste opstart, antager COMAL80 "ZONE" værdien -0-, der betyder, at kolonnebredden sættes til -0- positioner; herved sammentrykkes printlistens <værdier> uden mellemrum. Den første <værdi> i printlisten udskrives i første afsnit, den anden i andet afsnit osv. i henhold til <printlisten>. Fylder <værdi> mere end ét afsnit, lægges der automatisk beslag på næste afsnit.

PRINT

Anvendelse af skilletegnet semikolon (;)

Semikolon (;) mellem de enkelte udtryk undertrykker inddelingen i afsnit, og sammentrykker udtrykkene med et blankt tegn imellem tal, mens strenge udskrives uden mellemrum.

Sammenkædning med næste "PRINT" sætning

Når det sidste udtryk er udskrevet vil programmet skifte til første position på ny billedlinie, medmindre sætningen sluttes med et komma (,) eller et semikolon (;), hvorved billedlinieskiftet bliver undertrykt og udskrivningen følger reglerne for anvendelse af komma (,) og semikolon (;).

Udskrivning af blank billedlinie

En "PRINT" sætning, som ikke indeholder andet end ordet PRINT, medfører, at programmet foretager et billedlinieskift. Denne slags "PRINT" sætninger benyttes bl.a. til at udskrive blanke billedlinier som mellemrum i en udskrift, og til at foretage billedlinieskift efter "PRINT" sætninger som afsluttes med komma (,) eller semikolon (;).

Hvor kommer udskriften hen?

Normalt udskrives værdier og beregninger fra et program på dataskærmen, men brugeren kan anvende COMAL80 sætningen "SELECT OUTPUT" og med denne sætning få flyttet udskrivningen til en anden ydre enhed, f.eks. linieskriveren.

Udskrivning af en "TEKST"

"PRINT" sætningen opfatter en tekst som en konstant (en tegnstreng med anførselstegn omkring ("")). Teksten mellem de to anførselstegn udskrives i sin helhed, i overensstemmelse med parametrene.

Hvor på billedlinien starter udskrivningen (brug af TAB(X))

Ved en almindelig "PRINT" sætning starter udskrivningen af udtrykkene fra første position på billedlinien. Ønsker brugeren, at udskrivningen skal starte et bestemt sted på billedlinien, kan man anvende funktionen TAB(X) og "tabulere" sig hen til det sted

på billedlinien, hvorfra udskrivningen skal begynde. Det samme gør sig gældende, hvis "TAB(X)" står inde i sætningen og ikke forrest; så vil den næst følgende variabel eller tekst blive rykket hen til det positionsnummer, der står angivet i "TAB(X)", og først derefter bliver udskrivningen foretaget.

Udskrivning af variable og taludtryk

Indholdet af variable venstrestilles altid ved udskrivningen. Bemærk dog, at fortegn ved negative talværdier sættes foran variablens udskrevne værdi.

Et taludtryk opfattes af "PRINT" sætningen som en talvariabel, og udskrivningen følger derfor samme regler som for talvariable.

Brug af COMAL80 funktioner i "PRINT" sætninger

Foruden funktionen "TAB(X)", benyttes de to funktioner "CHR$(X)" og "ORD(A$)" ofte i "PRINT" sætninger. Begge funktioner har relation til et tegns ASCII-værdi (modulus 256 repræsentation), hvor "CHR$(X)" omsætter fra ASCII-værdi til tegnværdi (f.eks. 1, +, a, A, ... osv.) og "ORD(A$)" omsætter det første tegn i strengvariablen til ASCII-værdi.

Speciel udskrivning af billedlinie (brug af "MARGIN" sætning)

Brugeren kan selv definere længden på billedlinien ved i "MARGIN" sætningen at anføre det antal tegn, der skal være i billedlinien, inden billedlinieskift og en ny linie automatisk genereres. "MARGIN" kan være en nyttig sætning ved udskrivning på linieskriveren.

PRINT

SAMMENFATNING

COMAL80 sætningen "PRINT" anvendes til at danne udskrifter fra programmet.

Udtrykkene i "PRINT" sætningen kan være af følgende arter:

a: intet
b: tekst
c: variable
d: taludtryk

De enkelte arter kan i vilkårligt antal forekomme i samme "PRINT" sætning.

Skilletegnet komma (,) bevirker, at udtrykkene fordeles til billedliniens afsnit, bestemt af sætningen "ZONE".

Skilletegnet semikolon (;) bevirker, at der udskrives et blanktegn mellem tal, mens strenge udskrives uden mellemrum.Er der ikke plads til et udtryk på den resterende del af linien skiftes automatisk line. Sammenkædning af flere "PRINT" sætninger sker ved at benytte komma (,) eller semikolon (;) som afslutningstegn.

Udskriften kan dirigeres til andre ydre enheder end dataskærmen, ved at man benytter sætningen "SELECT OUTPUT".

For andre sætninger i tilknytning til "PRINT" sætningen se i øvrigt sætningerne:

- "PRINT USING"
- "MARGIN"
- "ZONE"

PRINT

COMAL80 kommando

COMAL80 sætning

FORMAT

```
PRINT
```

eller

```
PRINT <udtryk> {<skilletegn><udtryk>} [<skilletegn>]
      <skilletegn> ::= ,|;
```

- PRINT -

Anvendes til udskrivning af resultater.

Eksempel

```
0010 DIM s$ OF 50
0020 INPUT "Tast tekststreng: ": s$
0030 PRINT "Tekststrengen indeholder "; LEN(s$); "tegn."
```

PRINT FILE

ANVENDELSE

COMAL80 sætningen "PRINT FILE" skriver data på ASCII form til en sekventiel strøm. Strømmen skal i forvejen være åbnet i mode "WRITE".

I "PRINT FILE" anvendes skilletegnene komma (,) og semikolon (;), på samme måde som i sætningen "PRINT". I øvrigt gælder præcis de samme regler som for "PRINT".

Se i øvrigt COMAL80 sætningerne "WRITE FILE" og "OPEN" samt "PRINT".

PRINT FILE

COMAL80 kommando
COMAL80 sætning

FORMAT

```
PRINT FILE <strømnummer>: <udtryk> {<skilletegn><udtryk>}
           [<skilletegn>]
<strømnummer> ::= <taludtryk>
<skilletegn>  ::= ,|;
```

- PRINT FILE -
Anvendes til at skrive data på ASCII form til en datastrøm.

Bemærk, at kommandoen eller sætningen altid skal stå på én linie.

PROC

ANVENDELSE

"PROC" og "ENDPROC"-sætningerne benyttes til at erklære en procedure hvortil der kan refereres fra andre steder i programmet ved hjælp af <navn>.

En procedure består af et procedurehoved, som er "PROC"-sætningen, og en procedurekrop, som er sætningslisten mellem "PROC" og "ENDPROC".

Navnet efter "ENDPROC" skal være det samme som navnet efter "PROC". Dette kontrolleres, når programmet udføres.

Når "PROC"-sætningen mødes under programudførelsen, fortsættes med sætningen efter "ENDPROC"-sætningen. Sætningslistens sætninger kan kun udføres ved at proceduren kaldes, enten ved hjælp af en "EXEC"-sætning eller som en funktion, indgående i et aritmetisk udtryk.

Procedureerklæringer kan placeres, hvor som helst i programmet, typisk dog samlet først eller sidst.

BRUG AF PROCEDURER

Ofte har man brug for at udføre den samme funktion flere steder i et program, f.eks. kontrol af indtastede talværdier. I stedet for at skrive de nødvendige sætninger, hver gang, de skal bruges, kan man samle sætningerne i en procedure (et underprogram), som så kan kaldes flere forskellige steder i programmet. Herved sparer man ikke alene plads, men programmet bliver også mere overskueligt.

I det hele taget bliver programmet mere læseligt, hvis sætninger, der udfører en bestemt funktion, samles i en procedure.

Et program kommer således til at bestå af et hovedprogram, der angiver programmets hovedstruktur, samt et antal procedurer, der udfører programmets funktioner i detaljer.

KALD AF PROCEDURER

En procedure kaldes - dvs. sætningerne i procedurekroppen udføres - ved hjælp af sætningen "EXEC" (en procedure kan også kaldes på anden måde - herom senere).

Når EXEC-sætningen mødes gemmes oplysninger om, hvor fra proceduren er kaldt, således at man kan vende tilbage - returnere - , når proceduren er udført. Herefter fortsætter programudførelsen med sætningen efter "PROC"-sætningen.

Den sidste sætning i en procedure er altid en tilhørende "ENDPROC"-sætning. Når denne mødes fortsætter programudførelsen med den sætning, der følger efter den "EXEC"-sætning, hvormed proceduren blev kaldt.

OM PARAMETRE

En procedure må på en eller anden måde "vide", hvilke variable den skal arbejde på og hvor, den skal gemme resultaterne. Dette kan ske ved at procedurens inddata før kaldet placeres i bestemte variable, og at proceduren afleverer resultater i ligeledes aftalte variable.

Lad os som eksempel tage en procedure, der beregner, hvilket af to tal, der er det største. En sådan procedure kan for eksempel se således ud:

```
PROC MAXIMUM
  MAX:=A
  IF B>MAX THEN MAX:=B
ENDPROC MAXIMUM
```

Proceduren MAXIMUM vil altså gemme værdien af den største af de to variable A og B i variablen MAX.

Hvis man vil beregne maximum af to andre variable må man altså sætte A og B lig med de to aktuelle variable og derefter kalde proceduren MAXIMUM.

Dette kaldes at overføre parametre til proceduren.

I COMAL80 kan man imidlertid få systemet til at overføre parametrene for sig. Procedure MAXIMUM erklæres da på følgende måde:

```
PROC MAXIMUM (A, B)
   MAX:=A
   IF B > MAX THEN MAX:=B
ENDPROC MAXIMUM
```

Hvis man vil beregne maximum af de to variable I og J kaldes proceduren ved hjælp af sætningen

```
EXEC MAXIMUM (I, J)
```

I dette tilfælde vil A få tildelt I's værdi og B vil få tildelt J's værdi, hvorefter procedurens sætninger udføres. A og B kaldes de formelle parametre, mens I og J kaldes de aktuelle parametre. Hver gang proceduren kaldes vil de formelle parametre, blive tildelt de aktuelle værdier, der anføres i procedurekaldet.

Proceduren MAXIMUM kunne også have været kaldt på f.eks. følgende måde:

```
EXEC MAXIMUM (A, 8)
eller
EXEC MAXIMUM (A+3, B-2)
```

Da det er værdien af de aktuelle parametre, der overføres, kaldes A og B for værdiparametre.

De formelle parametre eksisterer kun inde i proceduren - de er lokale for proceduren. Sådanne lokale variable oprettes, når proceduren kaldes, og nedlægges, når der returneres fra proceduren. Dette betyder, at der udmærket i hovedprogrammet kan eksistere andre variable (globale) med navnene A og B, de vil ikke blive forvekslet med de formelle parametre A og B.

Når der inde i proceduren refereres til A, er det den lokale varabel A, der benyttes og ikke den globale.

REFERENCEPARAMETRE

I eksemplet med procduren MAXIMUM var det underforstået, at resultatet blev afleveret i variablen MAX. Man kunne ønske sig, at resultatvariablen kunne overføres som parameter, f.eks. på følgende måde:

```
PROC MAXIMUM (A, B, MAX)
```

Dette går imidlertid ikke, da MAX jo er en lokal variabel, og derfor vil blive nedlagt, når man returnerer fra proceduren. Grunden til at det ikke går er, at der er en fundamental forskel på A og B og resultatet MAX. A og B er værdier, der føres ind i proceduren, mens MAX er en værdi, som skal føres ud af proceduren.

For at løse dette problem benytter man i COMAL80 referenceparametre, der er kendetegnet ved, at der før variabelnavnet i parameterlisten står nøgleordet "REF".

Procedurehovedet ser da således ud:

```
PROC MAXIMUM (A, B, REF MAX)
```

og et kald kunne se sådan ud

```
EXEC MAXIMUM (I, J/2, STØRST)
```

"REF" fortæller da, at det ikke er værdien af STØRST, som skal overføres til MAX, men en reference (henvisning). Dette betyder, at variablen inde i proceduren skifter navn, men det er stadig den samme variable, der refereres til.

Som man kan se, kan man ved hjælp af referenceparametre føre værdier både ind i og ud af procedurer.

Hvornår skal man så bruge referenceparametre? Selvfølgelig hvis variablen skal modificeres af proceduren, men også i andre tilfælde. Referenceparametre er således særlig nyttige ved strengvariable, talsæt og strengtabeller, idet man i tilfælde af parameteroverførsel ved hjælp af referenceparametre ikke skal kopiere hele variablens dataområde og således sparer både tid og plads.

Er der tale om talsæt og strengtabeller må man angive dette ved efter variabelnavne at anføre "()" for endimensionale talsæt og strengtabeller og "(,)" for todimensionale talsæt.

I COMAL80 kan talsæt og strengtabeller kun overføres som referenceparametre.

PROCEDURER SOM FUNKTIONER

Ofte har en procedure en enkelt returparameter, som efter proceduredekaldet benyttes i et taludtryk. Proceduren har da karakter af en funktion på linie med f.eks. LOG og SIN.

I COMAL80 er det muligt at lade en procedure indgå i et taludtryk som funktion, hvorved proceduren kaldes automatisk, når udtrykket beregnes. Proceduren skal da indeholde mindst en tildelingssætning, hvor procedurenavnet står på venstre side af ':=' og funktionsværdien på højre side.

Proceduren MAXIMUM fra forrige eksempel kan, hvis den skal benyttes som funktion, erklæres på følgende måde:

```
PROC MAXIMUM (A, B)
   MAXIMUM:=A
   IF B > A THEN MAXIMUM:= B
ENDPROC MAXIMUM
```

og kaldes f.eks. sådan

```
MAX:= MAXIMUM (I, J)
```

eller

```
IF MAXIMUM (I, J) = J THEN N:= N+1
```

Når der returneres fra proceduren, erstattes funktionskaldet med funktionsværdien, og der fortsættes med beregningen af udtrykket.

Proceduren kan også kaldes ved hjælp af "EXEC". Funktionsresultatet vil da ikke blive benyttet.

LUKKEDE PROCEDURER

De hidtil omtalte procedurer har været åbne procedurer, forstået på den måde, at alle variable, bortset fra de formelle parametre er globale.

Det vil sige, at alle variable i programmet kan benyttes inde i proceduren, ligesom variable, der erklæres i proceduren er defineret uden for proceduren efter procedurekaldet.

I COMAL80 er der mulighed for at erklære lukkede procedurer. Dette gøres ved hjælp af nøgleordet "CLOSED" i procedurehovedet, således:

```
PROC MAXIMUM (A, B) CLOSED
```

Alle variable, der erklæres inde i proceduren, vil da være lokale, ligesom de formelle parametre, hvilket vil sige, at de kun er kendt inde i proceduren, og vil blive nedlagt, når der returneres fra proceduren. Det betyder også, at variable, der er erklæret udenfor proceduren ikke kan benyttes inde i proceduren (se dog COMAL80 sætningen "GLOBAL").

Fordelen ved lukkede procedurer er, at variable, der kun benyttes inde i proceduren, ikke optager plads efter procedurekaldet, og at der ikke opstår navnekonflikter på grund af tilfældige navnesammenfald.

Variable i en lukket procedure kan altså navngives uafhængigt af det øvrige programs variable. Dette er især af betydning, hvis der benyttes færdige procedurebiblioteker.

REKURSIVE PROCEDURER

Det er tilladt at lade procedurer kalde sig selv (rekursivt). Dette er særlig nyttigt i forbindelse med procedurer med parametre og lukkede procedurer, idet der bliver oprettet et nyt sæt af lokale parametre for hvert kald af proceduren.

I øvrigt vil brugen af rekursive procedurer ikke blive nærmere omtalt her. Der henvises til lærebøger i programmering.

PROC

COMAL80 sætning

FORMAT

PROC <navn> [(<parameter> {,<parameter>})] [CLOSED]
{<sætning>}
ENDPROC <navn>

<parameter> ::= <variabel>| REF <variabel>|REF <taltabel>()|
REF <taltabel>(,)|REF <strengtabel>()

- PROC - Anvendes til at erklære en procedure.

Eksempel

```
0010 PROC find (a) CLOSED
0020   IF a <> 1 THEN
0030     faktor:= 2; rod:= SQR(a)
0040     WHILE a MOD faktor <> 0 AND faktor <= rod DO faktor:= faktor+1
0050     IF a MOD faktor <> 0 THEN faktor:= a
0060     PRINT faktor,
0070     EXEC find (a DIV faktor)
0080   ENDIF
0090 ENDPROC find
0100
0110 ZONE 4
0120 INPUT "Tal:       ": a
0130 PRINT "Faktorer:  ",
0140 EXEC find (a)
```

RANDOMIZE

ANVENDELSE

COMAL80 sætningen "RANDOMIZE" bevirker, at funktionen "RND" starter generingen af tilfældige tal et tilfældigt sted i rækkefølgen af tilfældige tal.

Normalt vil funktionen "RND" generere den samme rækkefølge (sekvens) af tilfældige tal, hvergang kommandoen "RUN" aktiveres, dvs. hver gang et program udføres.

Det kan være hensigtsmæssigt at benytte programsætningen "RANDOMIZE" og programfunktionen "RND" sammen, således at generingen af tilfældige tal starter et tilfældigt sted i rækkefølgen af tilfældige tal.

Se i øvrigt funktionen "RND".

RANDOMIZE

COMAL80 sætning

FORMAT

RANDOMIZE

- RANDOMIZE -

Anvendes til at starte generingen af tilfældige tal mellem 0 og 1 et tilfældigt sted i rækkefølgen af tilfældige tal. Funktionen "RND" genererer de tilfældige tal.

Eksempel

```
0010 snit:= 0; max:= 20
0020 RANDOMIZE
0030 FOR i:= 1 TO max DO
0040   j:= RND
0050   PRINT j
0060   snit:= snit+j
0070 NEXT i
0080 snit:= snit/max
0090 PRINT "Gennemsnit: "; snit
```

...data, som er opbevaret i program-
...som står anført i "READ"
...sætningens variable får tildelt
...tildeling af værdier til tal-

...med sætningen "DATA". De
...sætningen, hvorfra de enkelte
...program kan indeholde flere
...blive betragtet som én stor liste
...som vilkårligt i programmet;
...i programmet. Be-
...sætningen skal være mindre
...elementer i "DATA"

...regepind til næste
...n. Den første "READ" sæt-
...ved det første dataelement
...ang "READ" sætningen får
...indtil til det næste dataele-
..."READ" sætninger og for
...dataelement, flyttes data-
...den første variable tilhøre i
...dataelement i strengen af "DATA"
...tildelt.

READ

I nogle situationer ønsker brugeren måske at gentage indlæsningen af de første dataelementer, uden at indlæse de resterende dataelementer. Sætningen "RESTORE" anvendes til at flytte pegepinden hen til det første dataelement i den første "DATA" sætning. Eller til det første dataelement i den bestemte "DATA" sætning som kommer efter en bestemt "Etikette".

Se i øvrigt sætningerne "DATA" og "RESTORE".

READ

COMAL80 sætning

FORMAT

READ <variabel> {,<variabel>}

- READ -

Anvendes til at læse data der er opbevaret i en eller flere "DATA" sætninger i selve programmet.

Eksempel

```
0010 RESTORE
0020 WHILE NOT EOD DO
0030   READ a
0040   PRINT a
0050 ENDWHILE
0060 DATA 1,2
```

READ FILE

ANVENDELSE

COMAL80 sætningen "READ FILE" læser data på intern form skrevet ved hjælp af sætningen "WRITE FILE".

Strømmen skal i forvejen være åbnet i mode "READ".

Se i øvrigt sætningerne "WRITE FILE" og "OPEN".

READ FILE

COMAL80 kommando
COMAL80 sætning

FORMAT

READ FILE <strømnummer>: <variabel> {,<variabel>}
strømnummer> ::= <taludtryk>

- READ FILE -
Anvendes til at læse data på intern form fra en datastrøm.

RENUMBER

ANVENDELSE

COMAL80 kommandoen "RENUMBER" benyttes til at omnummerere programlinier i et eksisterende program (det program, der findes i programarealet).

Typisk anvendes "RENUMBER" i forbindelse med omredigering af et program under udarbejdelse, eller ved sammenfletning med et andet program gemt på baggrundslageret (diskette).

Man kan som vist nedenfor kombinere anvendelsen af <startlinie-nummer> og <linienummerspring>.

RENUMBER	Programlinierne nummereres fra 0010 og med spring på 0010.
RENUMBER <startlinienummer>	Programlinierne nummereres fra <startlinienummer> og med spring på <startlinienummer>.
RENUMBER <startlinienummer>,<linienummerspring>	Programlinierne nummereres fra <linienummerspring> og med spring på <linienummer-spring>.
RENUMBER ,<linienummerspring>	Programlinierne nummereres fra 0010 og med spring på <linienummerspring>.
RENUMBER <startlinienummer>,	Programlinierne nummereres fra <startlinienummer> og med spring på 0010.

RENUMBER

COMAL80 kommando

FORMAT

```
RENUMBER [<linienummerliste>]
<linienummerliste>::=   <startlinienummer>|
                        <startlinienummer>,|
                        <startlinienummer>, <linienummerspring>|
                        ,<linienummerspring>
<startlinienummer> ::=  <taludtryk>
<linienummerspring> ::= <taludtryk>
```

- RENUMBER -

Anvendes til at omnummerere programlinier i et eksisterende program (det program der er i programarealet).

Eksempel

Indlæs programmet "test" og omnummerer hele programmet:

```
  load "test"
  list
  renumber
  list
               0010 .....
               0020 ...
      medfører  ..
                ..
               0150
```

Omnummerer fra og med linienummer 150 med spring = 50

```
  renumber 150,50
  list
               0150 ....
               0200 ....
      medfører  ..
                ..
               0850
```

REPEAT

ANVENDELSE

Når brugeren ønsker at gentage et antal programlinier, indtil et bestemt udsagn er sandt, kan COMAL80 sætningen "REPEAT" og den tilhørende sætning "UNTIL" benyttes. De to sætninger omslutter en række programlinier, som bliver udført sålænge udsagnet er falsk. En sådan gentagelse af programlinier kaldes en LØKKE. Først når udsaget er sandt, fortsættes med den programlinie, der kommer efter "UNTIL" sætningen. De omsluttede programlinier bliver derfor altid udført mindst én gang.

En "REPEAT - UNTIL" løkke kan indeholde andre "REPEAT - UNTIL" løkker.

En typisk "REPEAT - UNTIL" opgave er gennemlæsning af et register fra en diskette. F.eks. når brugeren vil søge efter et bestemt kundenummer i kunderegistret, eller hente faste løndata fra et løndataregister. Se i øvrigt sætningen "READ FILE".

En anden "REPEAT - UNTIL" opgave er omdannelse af en talvariabel til en strengvariabel f.eks. ved ajourføringer af kartoteksoplysninger. Eksempel "REPEAT" viser en sådan omdannelse.

Se i øvrigt sætningen "WHILE", der i sin brug ofte kan erstatte sætningen "REPEAT".

REPEAT

COMAL 80 sætning

FORMAT

```
REPEAT
  {<sætning>}
UNTIL <logisk udtryk>
```

Et falsk udsagn bevirker, at programlinierne bliver gentaget fra første programlinie efter "REPEAT" sætningen.

Et sandt udsagn bevirker, at programudførelsen fortsættes fra den programlinie der kommer efter "UNTIL" sætningen. Da værdien af <udsagn> først beregnes, når "UNTIL" sætningen mødes, vil de omsluttede programlinier altid blive udført mindst én gang.

Eksempel

```
0010 CREATE "datafil", 1
0020 OPEN 1, "datafil", WRITE
0030 REPEAT
0040   INPUT a
0050   WRITE FILE 1: a
0060 UNTIL a = 999
0070 CLOSE 1
```

RESTORE

ANVENDELSE

COMAL80 sætningen "RESTORE" benyttes sammen med sætningerne "READ" og "DATA".

Sætningen "RESTORE" bevirker, at brugeren kan få programmet til at foretage indlæsningen af dataelementer der står anført i en bestemt "DATA" sætning. Normalt benyttes ordet "RESTORE" alene og bevirker da, at DATA-pegepinden flyttes hen til det første dataelement i den første "DATA" sætning, hvorfra sætningen "READ" indlæser værdierne i sine variable en for en.

Anføres der i "RESTORE" sætningen navnet på en "Etikette" sætning, bevirker dette, at datapegepinden flyttes hen til det første dataelement i den første "DATA" sætning, der kommer efter "Etikette" sætningen.

Se i øvrigt sætningerne "READ", "DATA" samt "Etikette".

RESTORE

COMAL80 sætning

FORMAT

```
RESTORE [<navn>]
```

- RESTORE -

Anvendes til at gentage læsning fra den første "DATA" sætning. Indeholder "RESTORE" sætningen et <navn> flyttes datapegepinden til den første "DATA" sætning, der står efter den tilsvarende "Etikette".

Eksempel

```
0010 RESTORE
0020 WHILE NOT EOD DO
0030   READ a
0040   PRINT a
0050 ENDWHILE
0060 DATA 1,2
```

RND

ANVENDELSE

COMAL80 funktionen "RND" benyttes til at generere et "tilfældigt" tal i intervallet $0 \leq tal < 1$.

De genererede tal er pseudo-tilfældige, idet det næste tal beregnes ud fra det foregående. En række af pseudo-tilfældige tal vil dog opfylde en række statistiske betingelser for tilfældige tal.

Tilfældige tal benyttes til simuleringer og spil.

Ofte har man brug for et helt tal i et bestemt interval, M til N. Et sådant kan genereres ved hjælp af udtrykket

```
X:= INT (RND*(N-M)) + M.
```

Normalt vil funktionen "RND" generere den samme rækkefølge af tilfældige tal, hver gang et program udføres.

Ønsker brugeren at anvende uens rækkefølger af tilfældige tal, skal man anføre COMAL80 sætningen "RANDOMIZE", der bevirker, at genereringen af tilfældige tal starter et tilfældigt sted i datamaskinens række af tilfældige tal.

Se i øvrigt COMAL80 sætningen "RANDOMIZE".

RND

COMAL80 funktion

FORMAT

RND

- RND -

Anvendes til at generere et tilfældigt tal i intervallet

$0 \leq tal < 1$.

Eksempel

```
0010 snit:= 0; max:= 20
0020 RANDOMIZE
0030 FOR i:= 1 TO max DO
0040   j:= RND
0050   PRINT j
0060   snit:= snit+j
0070 NEXT i
0080 snit:= snit/max
0090 PRINT "Gennemsnit: "; snit
```

RUN

ANVENDELSE

COMAL80 kommandoen "RUN" aktiverer udførelsen af det program som findes i programarealet, og udfører programmets instruktioner linie for linie begyndende med det laveste linienummer.

Når "RUN" kommandoen aktiveres, slettes dataarealet inden programmet udføres.

RUN

COMAL80 kommando

FORMAT

RUN

- RUN -
Anvendes til at igangsætte programudførelsen af programmets instruktioner linie for linie begyndende med det laveste linienummer.

Eksempel

```
MOUNT 2
PREFIX "2/"
LOAD "programfil"
RUN
```

SAVE

ANVENDELSE

COMAL80 kommandoen "SAVE" benyttes til at gemme et program på baggrundslageret (diskette). Programmet gemmes på binær form, under det filnavn som står anført efter "SAVE".

Bemærk at navnet maksimalt må være på 8 tegn og at navnet altid skal være skrevet som et navn mellem to anførselstegn (") eller som en strengvariabel.

Eksisterer der på baggrundslageret en programfil af samme filnavn, bliver "SAVE" kommandoen afvist. Brugeren skal derfor altid for at kunne gemme en rettet programfil, først slette den gamle programfil fra baggrundslageret ("DELETE"), førend "SAVE" kommandoen med samme navn benyttes.

Et program, som er gemt ved hjælp af "SAVE", kan indlæses ved hjælp af "LOAD".

Se i øvrigt COMAL80 kommandoerne "LOAD" og "DELETE".

SAVE

COMAL80 kommando

FORMAT

SAVE <strengkonstant>

- SAVE -

Anvendes til at gemme programmer i binær form på baggrundslageret (diskette).

Eksempel

```
SELECT OUTPUT "printer"
LIST
DELETE "programfil"
SAVE "programfil"
```

SELECT OUTPUT

ANVENDELSE

Uddata i forbindelse med sætningee "PRINT" vil normalt komme ud på dataskærmen ("console").

Sætningen "SELECT OUTPUT" sætter brugeren i stand til selv at dirigere udskriften hen til en bestemt ydre enhed, f.eks. linieskriveren ("printer") eller en fil på baggrundslager.

Bemærk at sætningen "SELECT OUTPUT" vil dirigere alt fra sætningen "PRINT" til den angivne ydre enhed indtil programmet møder en ny sætning "SELECT OUTPUT" med angivelse af en ny ydre enhed.

Indeholder programmet ikke programsætningen "SELECT OUTPUT" vil programmet automatisk udskrive data fra sætningerne "PRINT" og "PRINT USING" på dataskærmen.

Se i øvrigt sætningerne "PRINT" og "PRINT FILE".

SELECT OUTPUT

COMAL80 kommando

COMAL80 sætning

FORMAT

SELECT OUTPUT <enhed>

<enhed> ::= strengudtryk

- SELECT OUTPUT -

Anvendes til at dirigere uddata fra "PRINT" og "PRINT USING" sætningerne til en bestemt ydre enhed eller fil.

Eksempel

```
SELECT OUTPUT "printer"
LIST
DELETE "programfil"
SAVE "programfil"
```

SGN

ANVENDELSE

COMAL80 funktionen "SGN" beregner det algebraiske fortegn af et <taludtryk>. Funktionsværdien af "SGN" er:

```
 1 (en)       : hvis <taludtryk> er positivt
-1 (minus en): hvis <taludtryk> er negativt
 0 (nul)      : hvis <taludtryk> er nul
```

Se i øvrigt COMAL80 funktionerne "ABS" og "INT".

SGN

COMAL80 funktion

FORMAT

SGN (<taludtryk>)

- SGN -

Anvendes til at bestemme fortegnet for et <taludtryk>.

Eksempel

```
0150 INPUT i
0160 PRINT "Tallet er ";
0170 IF SGN(i) >= 0 THEN
0180   PRINT "positivt"
0190 ELSE
0200   PRINT "negativt"
0210 ENDIF
```

SIN

ANVENDELSE

COMAL80 funktionen "SIN" udregner den trigonometriske funktion -sinus- til en vinkel, der er udtrykt i radianer.

$$1 \text{ radian} = \frac{360 \text{ grader}}{2 \text{ x pi}} = 57,29577951309 \text{ grader}$$

$$\text{pi} = 3,141592653589$$

COMAL80 funktionen skrives som: SIN(<taludtryk>)
Matematisk symbol skrives som : sin <taludtryk>

Se i øvrigt funktionerne "COS", "TAN" samt "ATN".

SIN

COMAL80 funktion

FORMAT

SIN (<taludtryk>)

- SIN -

Udregner sinus til en vinkel udtrykt i radianer.

SIZE

ANVENDELSE

COMAL80 kommandoen "SIZE" fortæller brugeren den øjeblikkelige status for anvendelsen af den del af det interne lager, som er til rådighed for brugeren.

Når kommandoen "SIZE" benyttes, udskrives på dataskærmen 3 talstørrelser. -program- som angiver hvor mange bytes programmet fylder; -data- som angiver hvor mange bytes dataarealet lægger beslag på, og endelig; -free- som angiver hvor mange bytes programmet kan disponere over, under selve programudførelsen.

Dataarealet vil være tomt hvis programmet ikke har været udført. Efter en udførelse af programmet er dataarealets størrelse et udtryk for, hvor mange variable programmet benytter.

SIZE

COMAL80 kommando

FORMAT

SIZE

- SIZE -
Angiver fordelingen mellem frit lager samt lager benyttet til henholdsvis programareal og dataareal (areal beslaglagt af variable).

Eksempel

Indlæs programmet "test" og udskriv foruddisponeret og ledigt areal til programudførelsen.

```
mount 1
load "test"
size
              program data free
    medfører:    3720  300 7500
```

SQR

ANVENDELSE

COMAL80 funktionen "SQR" beregner kvadratroden af et positivt <taludtryk>.

COMAL80 funktion skrives som : SQR(<taludtryk>)
Matematisk symbol skrives som: $\sqrt{\text{<taludtryk>}}$

SQR

COMAL80 funktion

FORMAT

SQR(<taludtryk>)

- SQR -

Anvendes til at uddrage kvadratroden af et positivt <taludtryk>.

STOP

ANVENDELSE

Udførelsen af et igangværende program kan stoppes et nærmere fastsat sted i programmet, når man anvender COMAL80 sætningen "STOP". Dette kan være nyttigt i forbindelse med afprøvningen af et program.

Når programmet i sin udførelse møder programsætningen "STOP", standser programudførelsen og systemet udskriver linienummeret for "STOP" sætningen. Brugeren kan igen fortsætte programudførelsen med kommandoen "CON" (for continue).

Se i øvrigt COMAL80 sætningen "END" samt COMAL80 kommandoen "CON".

STOP

COMAL80 sætning

FORMAT

STOP

- STOP -
Anvendes til at stoppe et igangværende program et nærmere fastsat sted i programmet.

TAB

ANVENDELSE

COMAL80 funktionen "TAB" benyttes i forbindelse med sætningen "PRINT". Funktionen "TAB" bevirker, at der foretages en tabulering hen til den position på linien, hvorfra udskriften skal starte.

"TAB" funktionens <taludtryk> kan angive en vilkårlig talværdi, decimaltal såvel som heltal, idet programsystemet selv afrunder til et heltal. Normalt benyttes et positivt heltal som <taludtryk>.

F.eks. bevirker følgende programlinie:

```
100 PRINT TAB(40);"COMAL"
```

at udskrivningen af teksten -COMAL- starter i position 40 på den udskrevne linie. For uddybning af positionering på den udskrevne linie henvises til sætningen "PRINT". I de tilfælde, hvor den aktuelle position på den udskrevne linie har en større talværdi end "TAB" funktionens <taludtryk>, er "TAB" funktionen virkningsløs.

Se i øvrigt COMAL80 sætningerne "PRINT" og "ZONE".

TAB

COMAL80 funktion

FORMAT

TAB(<taludtryk>)

- TAB -

Anvendes til at bestemme fra hvilken position en given udskrivning skal starte på udskriftslinien.

"TAB" funktionen benyttes sammen med sætningen "PRINT".

Eksempel

```
0010 INPUT a
0020 FOR i:= a TO 1 STEP -1 DO
0030   PRINT TAB(i);
0040   FOR j:= a TO 1 STEP -1 DO
0050     PRINT "**";
0060   NEXT j
0070   PRINT // linieskift
0080 NEXT i
```

TAN

ANVENDES

COMAL80 funktionen "TAN" udregner den trigonometriske funktion -tangens- til en vinkel, der er udtrykt i radianer.

$$1 \text{ radian} = \frac{360 \text{ grader}}{2 \text{ x pi}} = 57,29577951309 \text{ grader}$$

$$\text{pi} = 3,141592653589$$

COMAL80 funktionen skrives som: TAN(<taludtryk>)
Matematisk symbol skrives som : tan <taludtryk>

Se i øvrigt funktionerne "SIN", "COS" samt "ATN".

TAN

COMAL80 funktion

FORMAT

TAN(<taludtryk>)

- TAN -

Udregner tangens til en vinkel udtrykt i radianer.

Tildeling

ANVENDELSE

En tildeling er en COMAL80 sætning uden nøgleord, der benyttes til at tildele en eller flere variable bestemte værdier.

At tildele en værdi til en variabel, er ensbetydende med at brugeren i sit program ønsker at angive, at en bestemt variabel skal indeholde en bestemt værdi. Eller sagt på en anden måde: 2 + 2 er 4, men hvis ikke programmet får at vide hvor resultatet skal gemmes, er programmet ude af stand til sidenhen at benytte resultatet til noget.

At tildele en variabel en værdi, er altså det samme som at fortælle programmet, hvor det skal gemme værdien for senere at kunne benytte værdien igen.

COMAL80 skelner mellem variable, der indeholder tal, og variable, der indeholder tegn. Variable der indeholder tegn kendes på, at variabelnavnet slutter med et -$- tegn.

Talvariable kan kun tildeles talværdier, som består af cifre (0-9), decimalpunktum (.), fortegnene (+ og -) og eksponentialtegnet - E - og i øvrigt hverken bogstaver eller specialtegn. En talvariabel kan tildeles en anden talvariabels værdi.

Bemærk, at en talværdi foruden et egentligt tal eller anden talvariabel, også kan skrives som et taludtryk, sammensat af talvariable, talkonstanter, funktionskald og aritmetiske operatorer (f.eks. +, -, x, /).

Strengvariable og teksttabeller kan tildeles alle tegn (cifre, bogstaver og specialtegn). En strengvariabel kan tildeles en anden strengvariabels værdi.

Tildeling

Bemærk, at tildeling af værdier til en strengvariabel altid skal anføres i anførselstegn ("), med mindre det er en anden strengvariabel eller strengtabel. En strengvariabel kan endvidere tildeles værdien af et strengudtryk, dvs. strengvariable, strengkonstanter og kald af "CHR$" funktionen sammensat af operatoren plus (+).

Bemærk yderligere, at det ikke kan lade sig gøre at sammenligne talvariable med strengvariable. Ønskes en sammenligning, SKAL talvariablen eller taltabellen omformes til en strengvariabel eller en strengtabel.

Tildeling

COMAL80 kommando
COMAL80 sætning

FORMAT

<variabel> := <udtryk> {;<variabel> := <udtryk>}

- Tildeling -

Anvendes, når brugeren ønsker at tildele en variabel en værdi.

WHILE

ANVENDELSE

Man kan få gentaget udførelsen af én eller flere programlinier, sålænge et bestemt udsagn er sandt, ved at benytte COMAL80 sætningen "WHILE", enten i det simple format, eller i det udvidede format.

Brugeren kan vælge den simple "WHILE", hvis udsagnet og opgaven kan skrives på én linie.

I det udvidede format omslutter sætningerne "WHILE" og "ENDWHILE" de programlinier, der ønskes gentaget. En sådan gentagelse af programlinier kaldes en LØKKE. De omsluttede programlinier bliver kun udført, sålænge udsagnet er SANDT. Først når udsagnet er FALSK, fortsættes med den programlinie, der kommer efter løkken, også selvom løkken kun består af den simple "WHILE" sætning.

Specielt er det vigtigt, at hvis udsagnet er FALSK fra start, udføres løkken aldrig.

Se i øvrigt sætningerne "FOR" og "REPEAT".

WHILE

COMAL80 sætning

FORMAT

Simpel:

WHILE <logisk udtryk> DO <simpel sætning>

Udvidet:

```
WHILE <logisk udtryk> DO
  {<sætning>}
ENDWHILE
```

Eksempel

```
0010 RESTORE
0020 WHILE NOT EOD DO
0030   READ a
0040   PRINT a
0050 ENDWHILE
0060 DATA 1,2
```

WRITE FILE

ANVENDELSE

COMAL80 sætningen "WRITE FILE" skriver data på intern form i en sekventiel fil.

Strømmen skal i forvejen være åbnet i mode "WRITE".

På intern form fylder et reelt tal 8 tegn. En streng fylder 2 tegn plus den aktuelle længde af strengen.

Se i øvrigt COMAL80 sætningerne "PRINT FILE" og "OPEN".

WRITE FILE

COMAL80 kommando

COMAL80 sætning

FORMAT

WRITE FILE <strømnummer>: <udtryk> {,<udtryk>}

<strømnummer> ::= <taludtryk>

- WRITE FILE -

Anvendes til at skrive data på intern form.

Eksempel

```
0010 CREATE "datafil", 1
0020 OPEN 1, "datafil", WRITE
0030 REPEAT
0040   INPUT a
0050   WRITE FILE 1: a
0060 UNTIL a = 999
0070 CLOSE 1
```

ZONE

ANVENDELSE

COMAL80 sætningen "ZONE" angiver bredden af de kolonner, som linien opdeles i, når der i programsætningen "PRINT" anvendes komma (,) som skilletegn mellem elementer i printlisten. Kolonnebredden vil vedblive at have samme bredde (zone), i alle programmer indtil en ny "ZONE" sætning eller "ZONE" kommando ændrer på kolonnebredden. Værdien i "ZONE" behøver ikke at kunne divideres op i udskrivningsliniens 80 positioner, som et helt tal, idet COMAL80 selv undersøger om der er plads på udskrivningslinien. Er kolonnebredden ("ZONE"-værdien) større end antallet af resterende positioner på udskrivningslinien, fortsættes udskrivningen automatisk på en ny udskriftslinie.

Bemærk:
Angiver brugeren ingen "ZONE" kommando eller ingen "ZONE" sætninger i sine programmer, antager computeren at kolonnebredden er 0 positioner (default: ZONE 0).

-ZONE 0- bevirker at print-elementerne udskrives uden blanktegn imellem printelementerne. F.eks. kan dette udnyttes ved datoangivelse, når årstallet er opdelt i tre variable - år, måned og dag.

"ZONE"-værdien kan ikke være større end "MARGIN"-værdien, hvis denne er større end nul. Hvis man forsøger at angive en større værdi, vil "ZONE"-værdien blive sat lig med "MARGIN"-værdien. Hvis "MARGIN"-værdien er lig med nul, kan "ZONE"-værdien være vilkårlig.

ZONE

COMAL80 sætning

FORMAT

ZONE <kolonnebredde>

<kolonnebredde>::= <taludtryk>

- ZONE -

Angiver kolonnebredden for elementer i en "PRINT" sætning adskilt med komma (,).

Eksempel

```
0010 PROC find (a) CLOSED
0020   IF a <> 1 THEN
0030     faktor:= 2; rod:= SQR(a)
0040     WHILE a MOD faktor <> 0 AND faktor <= rod DO faktor:= faktor+1
0050     IF a MOD faktor <> 0 THEN faktor:= a
0060     PRINT faktor,
0070     EXEC find (a DIV faktor)
0080   ENDIF
0090 ENDPROC find
0100
0110 ZONE 4
0120 INPUT "Tal:       ": a
0130 PRINT "Faktorer:  ",
0140 EXEC find (a)
```

LÆSERBEMÆRKNINGER

Titel: COMAL80
Programmeringsvejledning

RCSL Nr.: 42-i1758

A/S Regnecentralen af 1979 bestræber sig på at forbedre kvalitet og brugbarhed af sine publikationer. For at opnå dette ønskes læserens kritiske vurdering af denne publikation.

Kommenter venligst manualens fuldstændighed, nøjagtighed, disposition, anvendelighed og læsbarhed:

Angiv fundne fejl (reference til sidenummer):

Hvordan kan manualen forbedres:

Andre kommentarer:

Navn: _____ Stilling: _____

Firma: _____

Adresse: _____

Dato: _____

På forhånd tak!

42-i 1289

..... Fold her .....

..... Riv ikke - Fold her og hæft .....

Frankeres
som
brev

REGNECENTRALEN
af 1979
Informationsafdelingen
Lautrupbjerg 1
2750 Ballerup

FINLAND
RC SCANIPS OY
Espoo, 0 51 35 22

FRANCE
RC COMPUTER S.A.R.L.
Paris, 12 33 53 63

HOLLAND
REGNECENTRALEN (NEDERLAND) B.V.
Gouda, 1820-29455

KUWAIT
KUWAITI DANISH COMPUTER CO. S.A.K.
Safat, 83 01 60

NORWAY
A/S RC DATA
Jessheim, 29 70 220

PHILIPPINES
CARDINAL ELECTRONICS CORPORATION
Metro Manila, 882478

SWEDEN
SCANIPS DATA AB
Stockholm, 8 34 91 55

SWITZERLAND
RC COMPUTER AG
Basel, 61 22 90 71

UNITED KINGDOM
REGNECENTRALEN (UK) LTD.
London, 1 606 3252

UNITED STATES
LOCKHEED ELECTRONICS COMPANY, Inc.
New Jersey, 201 757 1600

WEST GERMANY
RC COMPUTER G.m.b.H.
Frankfurt, 611 66 4006

RC COMPUTER
A/S REGNECENTRALEN af 1979

HEADQUARTERS: LAUTRUPBJERG 1 - DK 2750 BALLERUP - DENMARK
Phone: + 45 2 65 80 00 - Cables: rcbalrc - Telex: 35 214 rcbaldk