COMAL80
Brugervejledning

RCSL Nr. 42-i 2179

RC INFORMATION	class	repl.		ident EJ821230	
RC700	RC 4000	RC 6000	RC 8000	RC 3600	page

subj. COMAL80 rev. 1.07

Comal80 rev. 1.06 er i forhold til rev. 1.05 forbedret på følgende punkter:

1. fejlrettelser
2. nye faciliteter
3. større brugerareal (18 k)

I det følgende er de væsentligste ændringer kort beskrevet. En udførlig beskrivelse findes i "Comal80 Brugervejledning" (RCSL Nr. 42-i2179).

- Brugerarealet er 18432 bytes
- Indtil 5 strømme kan være åbne samtidig.
- CHAIN-sætning:

  Ved udførelse af sætningen CHAIN <filnavn> slettes program og data, hvorefter programmet <filnavn> indlæses og startes.
- Automatisk opstart af program:

  Et Comal80-program indlæses og startes automatisk ved systemopstart, hvis det er gemt i en fil ved navn "logon".
- Arbejdsfil:

  I statuslinien angives navnet på den fil, der er LOAD'et ned i programlageret, eller navnet på den fil, det aktuelle program er SAVE'et i. Filen kan opdateres ved blot at taste SAVE uden filnavn.
- Styring af ydre enheder:

  Tilgang til HW-porte sker fra Comal80 ved at åbne til den "ydre enhed" PORT. Syntaksen for navnet er "/<portnr>/PORT", hvor <portnr> er portnummeret (0 til 255).
- DIR-kommandoen udskriver antal frie bytes tilbage på disketten.
- SYS-funktionen:

  SYS(2): nummeret på den linie, hvori fejl opstod.
  SYS(3): tiden siden opstart i enheder á 20 msek.
  SYS(4): antal frie bytes i lageret
  SYS(5): den aktuelle værdi af ZONE
  SYS(6): den aktuelle værdi af MARGIN
- RND-funktionen:

  Kaldes RND-funktionen med 2 argumenter:
  RND (MIN, MAX), returneres et tilfældigt heltal i intervallet:
  MIN<=X<=MAX.

COMAL80

Brugervejledning

A/S REGNECENTRALEN af 1979
Informationsafdelingen

December 1982
RCSL Nr. 42-i2179

Forfatter: Niels Bach

Nøgleord: Danish language, RC700, COMAL80, User's Guide

Resumé: Denne manual beskriver programmeringssproget COMAL80, samt hvordan, det er implementeret på RC700.

(314 trykte sider)

Copyright © 1982, A/S Regnecentralen af 1979
RC Computer A/S

Udgivet af A/S Regnecentralen af 1979, København

Brugere af denne manual gøres opmærksom på, at specifikationerne heri uden forudgående varsel kan ændres af RC. RC er ikke ansvarlig for typografiske fejl eller regnefejl, som kan forekomme i denne manual, og er ikke ansvarlig for skader forårsaget af benyttelsen af dette dokument.

INDHOLDSFORTEGNELSE SIDE

1. INTRODUKTION 1

2. OPSTART OG INDTASTNING AF PROGRAMMER 3
2.1 Tastaturet 4
2.2 Indtastning af programmer 6
2.2.1 Kommandoer til programindtastning 8
2.2.1.1 NEW 8
2.2.1.2 AUTO 9
2.2.1.3 RENUMBER 9
2.2.1.4 DEL 10
2.2.2 Kommandoer til programudførelse (RUN og CON) 11

3. SYMBOLER OG RESERVEREDE ORD 12
3.1 Identifikatorer 12
3.2 Reserverede ord 13

4. TALBEHANDLING 14
4.1 Reelle tal 14
4.2 Intern repræsentation af tal 15
4.3 Numeriske variable 15
4.4 Taltabeller 16
4.4.1 Taltabelkomponenter 16
4.4.2 Erklæring af taltabeller 16
4.5 Numeriske udtryk 17
4.5.1 Aritmetiske-, sammenlignings- og logiske operatorer 21
4.6 Tildeling 22
4.6.1 READ,DATA 22

5. STRENGBEHANDLING 23
5.1 Strengkonstanter 23
5.2 Strengvariable 23
5.2.1 Erklæring af strengvariable 23
5.2.2 Delstrenge 24
5.3 Strengudtryk 25

INDHOLDSFORTEGNELSE SIDE

5.3.1 Strengudtryk, der returnerer en strengværdi. 25
5.3.2 Strengudtryk, der returnerer en talværdi ... 26

6. KONTROLSTRUKTURER 28
6.1 Betingede sætninger 28
6.1.1 IF-sætninger 28
6.1.2 Multiforgreninger (CASE) 30
6.2 Løkkestrukturer 34
6.2.1 Tællesløjfer (FOR-NEXT) 34
6.2.2 REPEAT-konstruktionen 36
6.2.3 WHILE-konstruktionen 38

7. INDLÆSNING OG UDSKRIVNING 39
7.1 INPUT-sætningen 39
7.2 PRINT-sætningen 40
7.3 Datastrømme 41
7.3.1 Filer og ydre enheder 41
7.3.2 Skrivning og læsning af programfiler 43
7.3.3 Skrivning og læsning af datafiler 45
7.3.3.1 Sekventielle filer 46
7.3.3.2 Filer med direkte tilgang 48
7.3.3.3 Ydre enheder 51
7.3.4 Fjernelse af filer 52
7.3.5 Omdøbning af filer 53
7.4 Oversigt over I/O sætninger 53

8. PROCEDURER OG FUNKTIONER 54
8.1 Simple procedurer 56
8.2 Parameteroverførsel 57
8.3 REF angivelse 58
8.4 Lukkede procedurer 61
8.5 Externe procedurer 64
8.6 Handler 66
8.7 Funktioner 68

INDHOLDSFORTEGNELSE SIDE

9. KOMMANDOER 70
9.1 Kommandoer til programindtastning 71
9.2 Sætninger udført som kommandoer 72
9.2.1 RC700 som bordkalkulator 73
9.2.2 Fejlfinding i programmer 73
9.2.3 Datastrømsindlæsning/udskrivning 74
9.3 Diskettekommandoer (MOUNT og DIR) 74
9.4 Datastrømskommandoer 75

10. COMAL80 NØGLEORD 78
10.1 ABS 80
10.2 AND 81
10.3 AT 82
10.4 ATN 84
10.5 AUTO 86
10.6 CASE-WHEN-ENDCASE 88
10.7 CHAIN 91
10.8 CHR$ 92
10.9 CLOSE 94
10.10 CON 95
10.11 CONTINUE 96
10.12 COPY 97
10.13 COS 98
10.14 CREATE 99
10.15 DATA 101
10.16 DEL 103
10.17 DELETE 105
10.18 DIM 107
10.19 DIR 109
10.20 DISABLE 111
10.21 DIV 112
10.22 EDIT 113
10.23 ENABLE 115
10.24 END 116
10.25 ENTER 117

INDHOLDSFORTEGNELSE SIDE

10.26 EOD 119
10.27 EOF 120
10.28 ERR 122
10.29 Etikette 123
10.30 EXEC 124
10.31 EXP 125
10.32 FALSE 126
10.33 FOR 127
10.34 FOR-NEXT 129
10.35 FUNC-ENDFUNC 131
10.36 FUNC-EXTERNAL 133
10.37 GET$ 135
10.38 GLOBAL 137
10.39 GOTO 138
10.40 IF-THEN 139
10.41 IF-THEN-ENDIF 140
10.42 IF-THEN-ELSE-ENDIF 141
10.43 IN 143
10.44 INPUT 145
10.45 INPUT FILE 147
10.46 INT 149
10.47 KEY$ 151
10.48 LEN 152
10.49 LIST 153
10.50 LOAD 155
10.51 LOG 156
10.52 MARGIN 157
10.53 MOD 158
10.54 MOUNT 160
10.55 NEW 161
10.56 NOT 162
10.57 OPEN 163
10.58 OR 165
10.59 ORD 166
10.60 OUT 167

INDHOLDSFORTEGNELSE SIDE

10.61 PREFIX 168
10.62 PRINT 169
10.63 PRINT FILE 171
10.64 PRINT FILE USING 172
10.65 PRINT USING 173
10.66 PROC-ENDPROC 176
10.67 PROC-EXTERNAL 179
10.68 PROC-HANDLER 181
10.69 RANDOMIZE 183
10.70 READ 185
10.71 READ FILE 186
10.72 RENAME 188
10.73 RENUMBER 189
10.74 REPEAT-UNTIL 191
10.75 RESTORE 197
10.76 RETRY 199
10.77 RETURN 200
10.78 RND 202
10.79 RUN 204
10.80 SAVE 205
10.81 SELECT OUTPUT 206
10.82 SGN 207
10.83 SIN 208
10.84 SIZE 210
10.85 SQR 211
10.86 STOP 212
10.87 STR$ 213
10.88 SYS 216
10.89 TAB 217
10.90 TAN 218
10.91 Tildeling 219
10.92 TRUE 220
10.93 VAL 221
10.94 WHILE 222
10.95 WHILE-ENDWHILE 223

INDHOLDSFORTEGNELSE SIDE

10.96 WRITE FILE 225
10.97 ZONE 227

A. REFERENCER 228

B. FEJLMEDDELELSER 229
B.1 Fejl fundet under programindtastning 230
B.2 Kommando- eller udførelsesfejl 234
B.3 I/O-Fejlmeddelelser 238

C. YDRE ENHEDER 240
C.1 Skærmstyring 240
C.1.1 Blink og invers skrift 240
C.1.2 Semigrafisk tegnsæt 241
C.1.3 Skærm - tegngenerering og konvertering 242
C.2 Tastatur - tegngenerering og konvertering 245
C.2.1 Konverteringstabel for tastatur 248
C.2.2 Ændring af konverteringstabel 255
C.3 Printerstyring 256
C.3.1 RC861-printeren 256
C.3.2 RC862,RC867 printeren 257
C.4 HW-porte 258

D. DISTRIBUTIONSDISKETTEN 260
D.1 Oprettelse af systemdiskette 261
D.1.1 Oprettelse af systemdiskette på RC700 med 1 disketteenhed 261
D.1.2 Oprettelse af systemdiskette på RC700 med 2 disketteenheder 262
D.2 Gennemgang af distributionsprogrammets faciliteter. 263
D.2.1 Generering af systemdiskette 264
D.2.2 Generering af datadiskette 264
D.2.3 Kopiering af diskette 264
D.2.4 Test af diskette 265
D.2.5 Ændring af printerhastighed 265

INDHOLDSFORTEGNELSE SIDE

D.3 Fejlmeddelelser 266

E. ASCII tegnsættet 267

F. STORE PROGRAMEKSEMPLER 268
F.1 Enarmet tyveknægt 268
F.2 Tænker du på et dyr ? 276
F.3 Tænker du på et dyr ? (filudgave) 279
F.4 Data-indtastning (PROC-EXTERNAL) 282
F.5 Sortering (PROC-EXTERNAL) 285
F.6 Semigrafik 288
F.7 Liv 291
F.8 Tænk på et tal ! 294
F.9 Eksempel fra procedureafsnittet 296

G. AUTOMATISK OPSTART AF PROGRAMMER 300

1. INTRODUKTION 1.

COMAL (COMmon Algorithmic Language) er et programmeringssprog, der siden 1975 har været det mest anvendte sprog i den danske undervisningssektor. Sproget blev først implementeret af A/S Regnecentralen på datamatserierne RC3600, RC7000 og RC8000.

COMAL blev udarbejdet for at tilgodese de brugere, der ønskede bedre sprogstruktur og flere faciliteter end dem, man møder i BASIC. Dog ønskede man at fastholde det nære interaktive miljø, der er kendetegnet for en række BASIC-fortolkere, og som har gjort dette sprog velegnet til undervisningsbrug.

Det oprindelige forslag til COMAL blev udarbejdet af studielektor Børge Christensen, Tønder Statsseminarium i 1974.

I 1979 nedsatte man en arbejdsgruppe bestående af Børge Christensen, repræsentanter for undervisningssektoren samt repræsentanter for en række maskinleverandører.

Målet for arbejdsgruppen var, på grundlag af de erfaringer man bl.a. havde høstet fra de tidlige COMAL implementeringer, at udarbejde et forslag til forbedringer af COMAL.

Resultatet af dette arbejde blev COMAL80.

Denne manual indeholder en komplet beskrivelse af sproget COMAL80, samt en beskrivelse af, hvordan sproget er implementeret på RC700.

Manualen skulle kunne læses af såvel den uerfarne som den trænede programmør. Den er delt op i tre hoveddele:

Første del er afsnittene 2-9, der indeholder en beskrivelse af, hvordan man indtaster programmer, hvilke variabeltyper der eksisterer, samt en koncentreret gennemgang af en række COMAL80 faciliteter.

Anden del er afsnit 10, der er et referenceafsnit, hvori alle COMAL80-nøgleord er beskrevet præcist.

Tredje del er appendix A-F, der indeholder referencer, mulige fejlmeddelelser, en detaljeret gennemgang af skærm og tastatur, vejledning i brugen af distributionsdisketten, ASCII-tabel og en række COMAL80-programeksempler, der kan indtastes og udføres på RC700.

Den uerfarne programmør bør starte med at læse afsnit 2-9, hvorimod programmøren med COMAL-erfaring kan gå direkte til afsnit 10.

Manualen indeholder en række programmer som eksempler. Disse er alle indtastet og afprøvet på RC700. Programmerne skal illustrere sætningstyper, sprogstrukturer osv., og må ikke altid opfattes som værende det bedste program til løsning af det givne problem. Der er primært lagt vægt på at eksemplerne er letforståelige og lettilgængelige. Ud fra denne målsætning skal det derfor bemærkes, at programmerne forudsætter, at man indtaster tekst med små bogstaver. Dette er gjort ud fra ønsket om at gøre programmerne letlæselige og for ikke at gøre problemet omkring store/små bogstaver til det centrale.

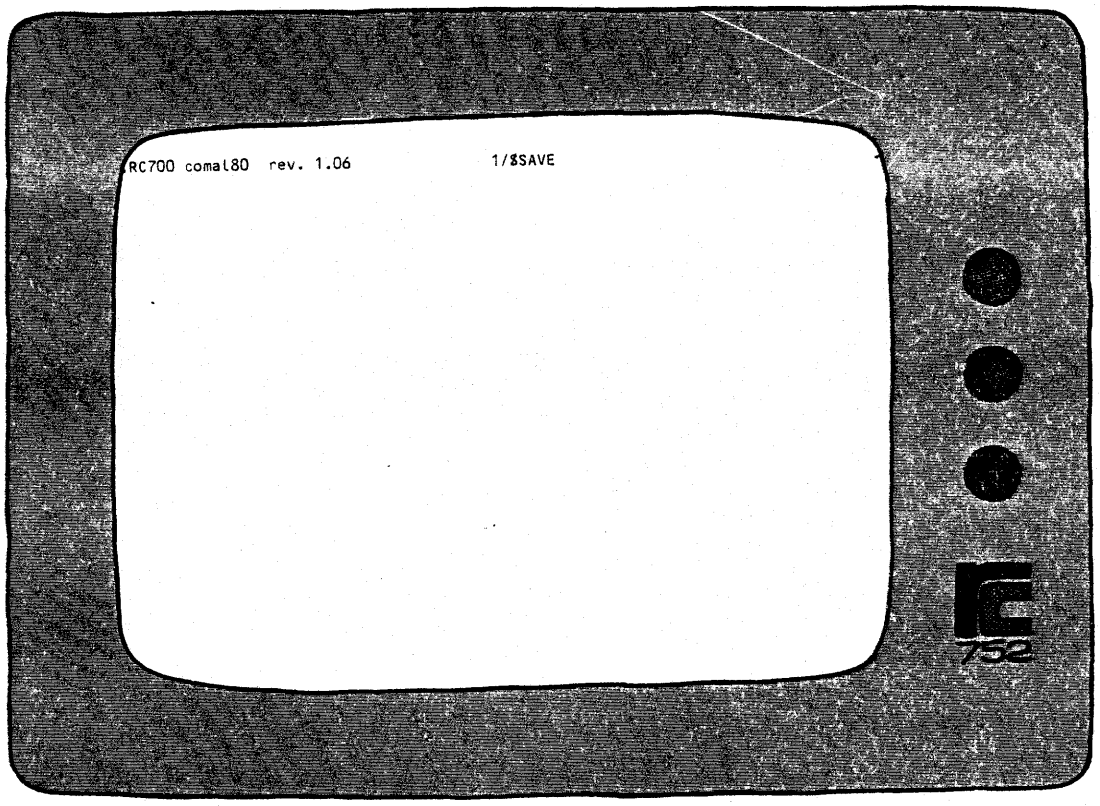
2. OPSTART OG INDTASTNING AF PROGRAMMER 2.

Før man kan komme igang med at bruge COMAL80 skal man selv oprette en systemdiskette. Dette gøres som beskrevet i appendix D. Vi går i det følgende ud fra, at man har oprettet en sådan.

Opstartsproceduren er herefter følgende :

1. Tænd for strømmen på RC700.
2. Indsæt systemdisketten i diskettestation nr 1
3. Tryk på RESET-knappen

COMAL80 startes nu op, og følgende skærmbillede fremkommer :

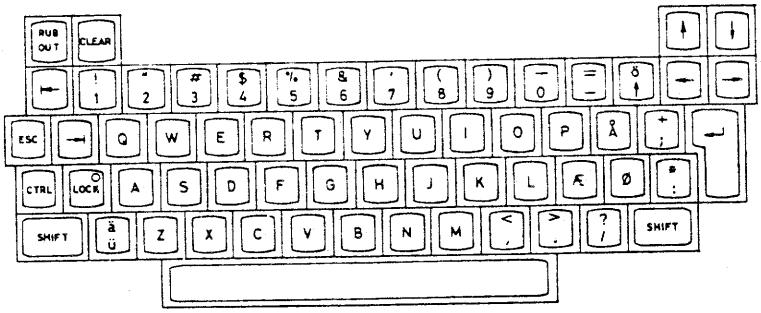


Cursoren (den blinkende firkant) fremkommer på skærmen, som tegn på at systemet er klar til indtastning.

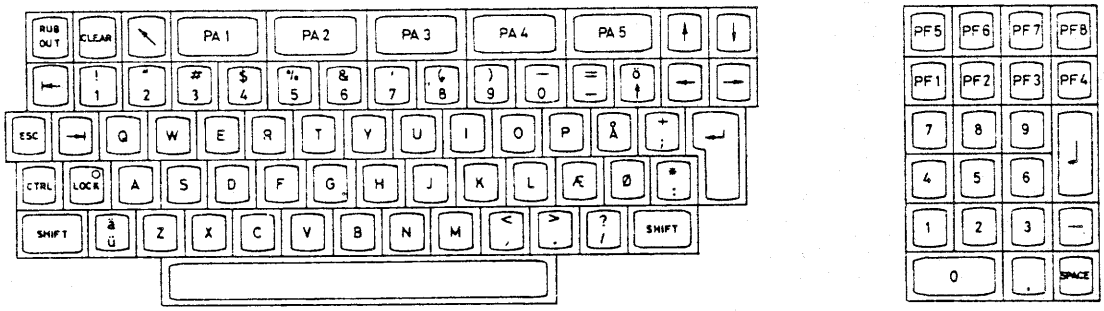
Der kan herefter indtastes program-sætninger eller kommandoer.

2.1 Tastaturet 2.1

På RC700 eksisterer to typer tastaturer, det almindelige (RC721):



og det udvidede (RC722):



Tastaturerne indeholder de almindelige taster, man kender fra en skrivemaskine, plus følgende:

RUB OUT : Slet sidst indtastede tegn på linien

CLEAR : Slet skærmen og stil cursoren i øverste venstre hjørne.

↑ : Flyt cursor en linie op

↓ : Flyt cursor en linie ned

→ : Flyt cursor en position til højre

← : Flyt cursor en position til venstre

↵ : Vognretur, ny linie. Tasten bruges altid til at afslutte sætninger eller kommandoer til systemet.

CTRL : Control-tasten. Trykkes ned sammen med et bogstav for at sende en specialkode til systemet.

ESC : Afbryder programudførelse

→| : Indsæt tegn i linie

|← : Slet tegn i linie

Følgende taster eksisterer kun på RC722 :

↖ : Flyt cursor til øverste venstre hjørne

PA1-PA5,

PF1-PF8 : Brugerdefinerbare specialtaster. Sender specialkoder (se appendix C).

SPACE : Mellemrumstast

COMAL80 på RC700 er skærmorienteret, dvs. at man kan benytte de fire cursorpile til at "hoppe" rundt i skærmbilledet, hvorved man kan "genbruge" linier, der står på skærmen. Sålænge en linie står et sted på skærmen, kan man indtaste den til systemet ved hjælp af pilene og vognretur-tasten.

Udover de beskrevne specialtaster, er følgende control-koder nyttige:

CTRL+↑ : Slet resten af linien

CTRL+RUB OUT : Slet resten af skærmen

CTRL+... betyder at CTRL tasten skal holdes nedtrykket mens den anden tast aktiveres.

Holdes en tast nedtrykket i mere end 0.7 sek., repeteres tegnet med 0.1 sek. intervaller, indtil tasten slippes igen.

2.2 Indtastning af programmer 2.2

Et program består af en række programlinier. Hver programlinie indledes med et linienummer, der bestemmer liniens placering i programmet.

Et linienummer er et heltal mellem 1 og 9999. Programlinierne kan indtastes i vilkårlig orden, idet COMAL80-systemet selv ordner dem efter voksende linienumre.

Efter linienummeret følger den egentlige sætning. Linien kan eventuelt afsluttes med en kommentar. En kommentar består af 2 skråstreger (//) efterfulgt af selve kommentaren.

Eksempel

```
100 PRINT CHR$(12) // SLETTER SKÆRMEN
|       |                    |
linienr sætning          kommentar
```

Linien skal afsluttes med en vognretur.

Når linien er afsluttet, vil COMAL80-systemet undersøge, om det er en korrekt COMAL-sætning (man siger at systemet syntaksanalyserer linien). Er linien korrekt, lagres den i program-lageret.

Hvis linien ikke er korrekt, udskrives en fejlmeddelelse i skærmens øverste højre hjørne, og cursoren placeres der, hvor fejlen blev fundet. Det er herefter muligt at rette i linien ved hjælp af de taster, der er beskrevet i afsn 2.1, og når rettelserne er foretaget, skal linien atter afsluttes med en vognretur.

Indtastes en sætning med samme linienummer, som en allerede eksisterende, vil den nye sætning erstatte den gamle.

Eksempler på lovlige sætninger

```
80 next j
70 print // tom linie
10 print "    "; // fire mellemrum
30 print // tom linie
50 print using "####":j; // fire pladser til j
40 for j:=1 to 10 do
20 for i:=1 to 10 do print using "####":i;
60 for i:=1 to 10 do print using "####":i*j;
```

Hvad de enkelte linier betyder, er ikke væsentlig i denne sammenhæng. Man kan få en udskrift af de indtastede linier ved at skrive

```
LIST
```

efterfulgt af tryk på vognretur.

Eksempel

Tastes linierne fra ovenstående eksempel ind og skrives LIST, fås følgende udskrift:

```
0010 PRINT "    "; // fire mellemrum
0020 FOR i:=1 TO 10 DO PRINT USING "####":i;
0030 PRINT // tom linie
0040 FOR j:=1 TO 10 DO
0050   PRINT USING "####":j; // fire pladser til j
0060   FOR i:=1 TO 10 DO PRINT USING "####":i*j;
0070   PRINT // tom linie
0080 NEXT j
```

Og når man er færdig med indtastningen, kan man udføre programmet ved at skrive

```
RUN
```

efterfulgt af tryk på vognretur.

Eksempel

Skrives RUN til programmet fra før, fås følgende udskrift:

	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

Vi har således fået udskrevet den "lille" tabel.

2.2.1 Kommandoer til programindtastning 2.2.1

Der findes en række kommandoer, der er nyttige at kende i forbindelse med programindtastning.

2.2.1.1 NEW 2.2.1.1

Før man starter indtastningen af et nyt program, skal man slette indholdet af programlageret og datalageret. Dette gøres med kommandoen :

```
NEW
```

2.2.1.2 AUTO

2.2.1.2

Hvis flere linier skal indtastes, er det praktisk at lade systemet sætte linienumrene under indtastningen. Dette gøres med kommandoen :

```
AUTO
```

hvilket bevirker, at linienummeret 0010 skrives på skærmen. Når linien er indtastet, tastes vognretur. Hvis linien er korrekt udskrives 0020, derefter 0030,0040 osv.

Med AUTO-kommandoen kan man både bestemme startlinienummeret og springet mellem linienumrene. Indtastes f.eks.:

```
AUTO 100,5
```

Udskrives først 0100. Derefter 0105,0110,... .

Man standser den automatiske linienummerering igen ved at trykke på ESC-tasten.

2.2.1.3 RENUMBER

2.2.1.3

En gang imellem er det nødvendigt at ændre linienumrene i et program f.eks. for at kunne indsætte yderligere programlinier. Dette gøres med kommandoen:

```
RENUMBER
```

der bevirker, at linienumrene i programlageret blive lavet om, således at første linie får linienummeret 0010, anden linie 0020, derefter 0030, 0040 osv.

I kommandoen kan man tillige angive startlinienummeret og springet mellem linienumrene efter omnummereringen. Hvis man angiver:

RENUMBER 100,20

vil linienumrene i programlageret blive lavet om til linienumrene 0100, 0120, 0140 osv.

2.2.1.4 DEL

2.2.1.4

Hvis man vil fjerne enkelte linienumre kan det gøres med kommandoen DEL.

Ønsker man f.eks. at fjerne linie 0070 i programlageret, er kommandoen

DEL 70

Ønsker man at fjerne linierne fra linie 0110 til linie 0270, er kommandoen

DEL 110,270

Ønsker man at fjerne linie 1000 og resten af programmet, er kommandoen

DEL 1000,

Hvis man endelig ønsker at fjerne alle linierne fra begyndelsen og til og med linie 0250, er kommandoen

DEL ,250

2.2.2 Kommandoer til programudførelse (RUN og CON) 2.2.2

Når et program skal udføres, gøres det med kommandoen

```
RUN
```

Hvis man ønsker, at udskriften fra programmet, der normalt kommer på skærmen, skal udskrives på printeren, gøres det med følgende to kommandoer:

```
SELECT OUTPUT "printer"
RUN
```

Ledetekst i INPUT-sætninger og fejlmeddelelser udskrives dog stadig på skærmen.

Man kan stoppe programudførelsen ved at trykke på ESC-tasten.

Herefter kan man ændre variables værdier, udskrive værdier og genstarte programmet med kommandoen

```
CON
```

Man må ikke ændre programlinier før en CON-kommando, men på samme måde som før, kan man ændre udskriftsenheden med følgende to kommandoer:

```
SELECT OUTPUT "printer"
CON
```

3. SYMBOLER OG RESERVEREDE ORD 3.

3.1 Identifikatorer 3.1

Identifikatorer benyttes til at navngive forskellige størrelser i et COMAL80-program. En identifikator består af et bogstav efterfulgt af op til 15 bogstaver og cifre eller tegnet understregning (_), altså et navn på maximalt 16 tegn.

Bemærk der skelnes ikke mellem store og små bogstaver i identifikatornavne. COMAL80-systemet udskriver altid navnene med små bogstaver.

Identifikatorer kan betegne følgende størrelser i et program:

- simple numeriske variable
- taltabeller (vektorer og matricer)
- simple strengvariable
- indicerede strengvariable (teksttabeller)
- brugerdefinerede funktioner
- procedurer
- formelle parametre
- etiketter

Hvad disse begreber dækker over, vil blive gennemgået i det følgende.

Samme identifikator må normalt ikke betegne forskellige størrelser i det samme program, og identifikatorerne må ikke være de samme som systemets reserverede ord, de såkaldte nøgleord.

3.2 Reserverede ord 3.2

Følgende ord er reserverede i COMAL80. De har en bestemt betydning overfor systemet, og må ikke benyttes i nogen anden betydning.

ABS	ENDCASE	LOAD	RETURN
AND	ENDFUNC	LOG	RND
APPEND	ENDIF		RUN
AT	ENDPROC	MARGIN	
ATN	ENDWHILE	MOD	SAVE
AUTO	ENTER	MOUNT	SELECT
	EOD		SGN
CASE	EOF	NEW	SIN
CHAIN	ERR	NEXT	SIZE
CHR	EXEC	NOT	SQR
CLOSE	EXP		STEP
CLOSED	EXTERNAL	OF	STOP
CON		OPEN	STR
CONTINUE	FALSE	OR	SYS
COPY	FILE	ORD	
COS	FOR	OTHERWISE	TAB
CREATE	FUNC	OUT	TAN
		OUTPUT	THEN
DATA	GET		TO
DEL	GLOBAL	PREFIX	TRUE
DELETE	GOTO	PRINT	
DIM		PROC	UNTIL
DIR	HANDLER		USING
DISABLE		RANDOM	
DISMOUNT	IF	RANDOMIZE	VAL
DIV	IN	READ	
DO	INPUT	REF	WHEN
DUMP	INT	RENAME	WHILE
		RENUMBER	WRITE
EDIT	KEY	REPEAT	
ELSE		RESTORE	ZONE
ENABLE	LEN	RESUME	
END	LIST	RETRY	

4. TALBEHANDLING 4.

4.1 Reelle tal 4.1

COMAL80 kan regne i følgende positive talområde

$$1.000000000000E\text{-}128 \leq n \leq 9.999999999999E126$$

et tilsvarende negativt og endelig tallet 0.

Hvis et tal kan udskrives med maximalt 13 cifre, vil det blive udskrevet direkte. Et reelt tal, der kræver mere end 13 cifre, udskrives på følgende form

<fortegn>n.nnnnnnnnnnnnE<fortegn>XXX

hvor n.nnnnnnnnnnn er et fortegnsløst tal med 12 decimaler, dog fjernes eventuelt højrestillede nuller, E betyder "gange 10 opløftet til potensen" og XXX er en fortegnsløs eksponent.

Eksempel

Tal	Udskrivningsformat
300000000000000000	3E+017
2048	2048
-.000000000123456	-1.23456E-010
200E5	2E+007

4.2 Intern repræsentation af tal 4.2

Internt gemmes reelle tal i 8 bytes, der har følgende form:

```
        Bit  0      7
     Byte 1  FUUUxxxx
             xxxxxxxx
             xxxxxxxx
             xxxxxxxx
             xxxxxxxx
             xxxxxxxx
             xxxxxxxx
             xxxxxxxx
     Byte 8  eeeeeeee
```

hvor F angiver fortegnet på mantissen, UUU er 3 ubrugte bit, xxxx... angiver mantissen, hvor hvert ciffer er gemt hexadecimalt, dvs hvert ciffer optager 4 bit, og eeeeeeee angiver eksponenten + 128 fremstillet binært.

4.3 Numeriske variable 4.3

En numerisk variabel, forkortet med symbolet <nvar>, er et eksempel på en identifikator, og har derfor et navn bestående af et bogstav efterfulgt af indtil 15 bogstaver, tal eller tegnet understregning (_).

Eksempel

Lovlige navne	Ulovlige navne
i42	%k
længde	7t
tid	efterspørgselsindex
gennemsnitsalder	

4.4 Taltabeller 4.4

En taltabel (eller array) består af en række tal. Værdierne eller elementerne i tabellen kaldes tabellens komponenter, og de kan hver for sig opfattes som numeriske variable. En taltabel kan have en eller to dimensioner. Har den dimensionen en, kaldes den en vektor, - har den dimensionen to, kaldes den en matrix.

4.4.1 Taltabelkomponenter 4.4.1

Hvert element i en taltabel er kendetegnet med navnet på taltabellen, efterfulgt af et index i parentes, f.eks.

```
pris(1), pris(2) ,... pris(9), pris(10)
```

For en todimensional taltabel angiver det første index rækkenummeret, og det andet index søjlenummeret. Tabellen kan f.eks. have følgende udseende :

```
antal(1,1), antal(1,2), antal(1,3), antal(1,4)
antal(2,1), antal(2,2), antal(2,3), antal(2,4)
antal(3,1), antal(3,2), antal(3,3), antal(3,4)
```

4.4.2 Erklæring af taltabeller 4.4.2

Før man kan benytte en taltabel, skal den være erklæret i en DIM sætning (se afsn 10.18). I denne sætning angives tabellens navn, dens dimension samt den øvre grænse for index. Den nedre grænse for et index er altid 1.

Eksempel 1

```
0010 DIM pris(10),antal(3,4)
```

Eksempel 2

```
0010 INPUT "Antal elever ? ":maxelev
0020 DIM højde(maxelev)
0030 FOR i:=1 TO maxelev DO INPUT "Højde ? ":højde(i)
0040 // Så er højden indtastet og lagret
```

Bemærkning

Man kan i ovenstående eksempel se, at det øvre index for en taltabel kan indlæses, og det behøver således ikke være fastlagt på det tidspunkt, man siger RUN. Det skal blot vare fastlagt før erklæringen af tabellen i programmet.

4.5 Numeriske udtryk 4.5

Et numerisk udtryk (beskrevet i kapitel 10 med <nudtr>) må bestå af:

1. Parenteser
2. Aritmetiske-,sammenlignings- og logisk operatorer
3. Talkonstanter
4. Variable
5. Numeriske funktioner
6. strengudtryk, der returnerer en talværdi (se afsn 5.3.2)

Talkonstanter, variable, numeriske funktioner og strengudtryk, der returnerer en talværdi, må alle optræde som operander i et numerisk udtryk.

+,-,*,/,↑,MOD,DIV

+,-,*,/,↑,MOD og DIV operatorerne udfører de almindelige regneoperationer, dvs.

+ : udfører addition af to argumenter, f.eks. 6+2 (=8)
- : udfører subtraktion af to argumenter, f.eks. 6-2 (=4)
* : udfører multiplikation af to argumenter, f.eks. 6*2 (=12)

/ : udfører division af to argumenter, f.eks. 6/2 (=3)

↑ : udfører potensopløftning af to argumenter, f.eks. 6↑2 (=36)

MOD :udregner værdien af heltalsresten ved en heltalsdivision, hvor den numeriske værdi af argumenterne benyttes.

Eksempel

11 MOD 4 = 3

-11 MOD 4 = 3

DIV :udfører heltalsdivision af to argumenter.

Eksempel

11 DIV 4 = 2

-11 DIV 4 = -2

Regler for de almindelige regneoperationer

De almindelige udregningsudtryk bliver udregnet efter følgende regler:

1. Udtryk i parenteser udregnes først. Hvis der optræder flere niveauer af parenteser, udregnes den inderste først.
2. Dernæst udregnes funktioner.
3. De almindelige regneoperationer har følgende udregningsprioritet:
   1. Positivt og negativt fortegn
   2. Potensopløftning
   3. Multiplikation, division, modulus og heltalsdivision
   4. Addition og subtraktion
4. Hvis to operatorer har samme prioritet, udregnes udtrykket fra venstre mod højre.

Eksempel 1

```
   7 + 5 - 3 * 4 ↑ 2 DIV 5 + 1
1: 7 + 5 - 3 *  16   DIV 5 + 1
2: 7 + 5 -   48      DIV 5 + 1
3: 7 + 5 -       9         + 1
4:   12  -       9         + 1
5:           3             + 1
6:                     4
```

Eksempel 2

```
   (7 + (5 - 3) * 4 ↑ 2 ) DIV 5 + 1
1: (7 +    2    * 4 ↑ 2 ) DIV 5 + 1
2: (7 +    2    *  16   ) DIV 5 + 1
3: (7 +         32       ) DIV 5 + 1
4:       39                DIV 5 + 1
5:                  7            + 1
6:                                 8
```

AND,OR,NOT

Operatorerne AND,OR og NOT svarer til *,+ og -, men hvor *,+ og - har numeriske argumenter, har AND,OR og NOT logiske argumenter. Et logisk argument har i princippet to værdier : sand (dvs <> 0) og falsk (dvs =0).

NOT har kun et argument, og udfører logisk negering.

Eksempel

a	NOT a
sand	falsk
falsk	sand

AND har to argumenter, og udfører logisk OG:

Eksempel

a	b	a AND b
sand	sand	sand
sand	falsk	falsk
falsk	sand	falsk
falsk	falsk	falsk

OR har to argumenter, og udfører logisk ELLER:

Eksempel

a	b	a OR b
sand	sand	sand
sand	falsk	sand
falsk	sand	sand
falsk	falsk	falsk

<,<=,>,>=,=,<>

Sammenligningsoperatorerne er følgende:

< : mindre end
> : større end
<= : mindre end eller lig med
>= : større end eller lig med
= : lig med
<> : forskellig fra

Alle sammenligningsoperatorerne har to argumenter. Resultatet af en sammenligning er altid enten sand (<>0) eller falsk (=0).

Bemærk

Man må ikke skrive en sammenligning som f.eks.:

```
5 < 0 < 10
```

Da udtrykket vil blive udregnet således:

```
5 < 0 < 10
1:    0(falsk) < 10
2:             1 (sand)
```

Den korrekte notation er

```
(5 < 0) AND (0 < 10)
```

4.5.1 Aritmetiske-, sammenlignings- og logiske operatorer 4.5.1

For sammensatte udtryk gælder følgende prioritetsregler:

Prioritet	Symbol	Funktion
1	+	Positivt fortegn
1	-	Negativt fortegn
2	↑	Potensopløftning
3	*	Multiplikation
3	/	Division
3	DIV	Heltalsdivision
3	MOD	Rest ved heltalsdivision
4	+	Addition
4	-	Subtraktion
5	=	Lig med
5	<>	Forskellig fra
5	<	Mindre end
5	>	Større end
5	<=	Mindre end eller lig med
5	>=	Større end eller lig med
6	NOT	Logisk negering
7	AND	Logisk OG
8	OR	Logisk ELLER

4.6 Tildeling 4.6

Numeriske variable kan tildeles værdier med sætninger som f.eks.

```
areal := pi * radius ↑2
```

Betydningen af sætningen er:

- udregn højre side og tildel værdien til variablen på venstre side.

Det er altså ikke et matematisk lighedstegn, da f.eks. følgende sætning er lovlig:

```
dagnr := dagnr + 1
```

Her udregnes værdien af dagnr + 1 og værdien tildeles dagnr, dagnr tælles med andre ord op med 1.

4.6.1 READ,DATA 4.6.1

Hvis man skal tildele variable konstante værdier kan det både gøres med almindelige tildelingssætninger, men det kan også gøres ved hjælp af READ og DATA-sætningerne.

Eksempel

```
0010 READ alder,højde
0020 DATA 75,180
```

Bemærkning

Når alder skal tildeles en værdi, tages første værdi i DATA-sætningen (=75). Variable højde får den følgende værdi (=180).

Indlæsningen må gerne foretages over flere READ sætninger ligesom værdierne må spredes over flere DATA-sætninger. Systemet vil ved udførsel af programmet samle værdierne i DATA-sætningerne til én lang liste.

5. STRENGBEHANDLING 5.

5.1 Strengkonstanter 5.1

En streng er en tekst, dvs. en række tegn (bogstaver, tal, mellemrum og specialsymboler). En strengkonstant er en streng, der er anført i anførselstegn ("). Strengkonstanter bruges oftest i PRINT,INPUT og DATA sætninger (se afsn 10).

Eksempel

```
0010 PRINT "Dette er en strengkonstant"
0020 INPUT "Indtast et tal :": i
0030 DATA "Denne streng kan læses med en READ-sætning"
```

5.2 Strengvariable 5.2

COMAL80 tillader brugen af både strengvariable og strengkonstanter. En strengvariabel er et eksempel på en identifikator, dvs den har et navn som består af et bogstav efterfulgt af indtil 15 bogstaver, tal eller symbolet understregning (_), efterfulgt af et dollar-tegn ($).

Eksempel

Lovlige strengnavne	Ulovlige strengnavne
tekst$	streng ($ mangler)
efternavn1$	7$ (skal starte med bogstav)
postnrogområde$	key$ (reserveret ord)

5.2.1 Erklæring af strengvariable 5.2.1

Både simple strengvariable og teksttabeller skal erklæres i en DIM-sætning før de benyttes (se afsn 10.18). I denne sætning angives strengens navn, det maximale antal tegn i strengen, samt, for teksttabellers vedkommende, det maximale antal elementer i tabellen.

Eksempel 1

```
DIM linie$ of 50
```

Når ovenstående sætning udføres, vil der blive reserveret plads i lageret til en streng på maximalt 50 tegn.

Eksempel 2

```
DIM tabel$(10) of 50
```

Når ovenstående sætning udføres, vil der blive reserveret plads i lageret til 10 tekster på hver maximalt 50 tegn.

5.2.2 Delstrenge

Man kan nøjes med at benytte en del af en strengvariabel i udtryk, udskrifter og lignende.

For en simpel strengvariabel er formatet:

<svar> (<i>:<j>)

og for en teksttabel er formatet :

<svar> (<k>,<i>:<j>)

<svar> : navnet på en strengvariabel
<k> : et numerisk udtryk, der udpeger et element i teksttabellen
<i> : et numerisk udtryk, der angiver, at delstrengen skal starte med det i'te tegn i <svar>
<j> : et numerisk udtryk, der angiver, at delstrengen skal slutte med det j'te tegn i <svar>

eksempel

```
tekst$ := "COMAL80 brugermanual"
```

Delstreng	Indhold
tekst$(1:5)	COMAL
tekst$(2*(3+8/2)+1:2↑4+9/3+1)	manual
tekst$(6:7)	80

5.3 Strengudtryk

Der eksisterer to typer strengudtryk:

- strengudtryk, der har en streng som "værdi"
- strengudtryk, der har et tal som værdi. Et sådant udtryk kan optræde istedet for almindelige numeriske variable.

5.3.1 Strengudtryk, der returnerer en strengværdi

Af operationer på strenge, der har strenge som værdi, eksisterer der tre typer :

- strengfunktioner
- delstrengsoperationer
- sammensætning af strenge

Af strengfunktioner, der har strenge som værdi, findes følgende:

CHR$(x) :	returnerer ASCII-tegnet svarende til x
GET$(s,n) :	returnerer n tegn læst fra strøm nr s
KEY$:	returnerer med det sidst indtastede tegn på tastaturet
STR$(x) :	konverterer tallet x til en streng

Delstrengsoperationen er beskrevet i afsnit 5.2.2.

Sammensætning af strenge kan ske ved angivelse af + mellem de enkelte elementer

Eksempel

```
0010 DIM tekst$ OF 15, navn$ of 7
0020 tekst$:="700"
0030 READ navn$
0040 DATA "piccolo"
0050 tekst$:="rc"+tekst$+chr$(32)+navn$ // chr$(32): blanktegn
0060 PRINT tekst$
0070 END
```

Hvis ovenstående program udføres udskrives:

```
rc700 piccolo
```

5.3.2 Strengudtryk, der returnerer en talværdi 5.3.2

Af strengudtryk, der returnerer en talværdi, skal nævnes:

- Sammenligningsoperatorerne
- IN-operatoren

Desuden har en række funktioner strenge som argumenter:

LEN(S$) : Længden af strengen S$
ORD(S$) : Tegnværdien af første tegn i S$
VAL(S$) : Konvertering af S$ til talværdi

To strenge (konstanter, variable eller udtryk) kan sammenlignes. Resultatet af sammenligningen er enten sand (=1) eller falsk (=0). Strengene sammenlignes tegn for tegn fra venstre mod højre på grundlag af deres ASCII-værdi (se appendix E) indtil enten en forskel konstateres eller man når til slutningen af den ene eller begge strenge.

Reglerne er herefter følgende:

- Hvis en forskel er konstateret på samme position i de to strenge, er den streng størst, hvis tegns ASCII-værdi er størst.

- Hvis strengene er ens position for position, er den streng størst, som er længst.

Eksempler

```
"AAA"   <   "AAB"
"RC"    <   "RC700"
"COMAL" =   "COMAL"
```

IN-operatoren returnerer positionen for en streng i en anden streng.

Eksempel

Udtryk	Værdi
"b" IN "abc"	2
"70" IN "RC700"	3
"opera" IN "operator"	1
"bg" IN "bog"	0
"" IN "tekst"	6

6. KONTROLSTRUKTURER 6.

6.1 Betingede sætninger 6.1

6.1.1 IF-sætninger 6.1.1

En betinget sætning betyder, at en sætning kun udføres såfremt en betingelse er opfyldt.

Den enkleste form for betingede sætninger er den simple IF-sætning:

Eksempel

```
0010 INPUT "Indtast et beløb :": beløb
0020 IF beløb>100 THEN PRINT "Beløbet er større end 100"
0030 END
```

Bemærkninger

linie 0010 : Her indtastes værdien af variablen beløb

linie 0020 : Den simple IF-sætning. Hvis værdien af beløb er større end 100 udskrives teksten:
Beløbet er større end 100
på skærmen.

Hvis man ønsker at udføre mere end en sætning, hvis betingelsen er opfyldt, kan man bruge den udvidede IF-sætning:

Eksempel

```
0010 INPUT "Indtast et beløb :": beløb
0020 IF beløb>100 THEN
0030   PRINT "Beløbet er større end 100"
0040   PRINT "Der gives 10 % rabat"
0050   beløb:=beløb*0.9
0060 ENDIF
```

Bemærkninger

linie 0010 : Her indtastes værdien af beløb

linie 0020-0060 : Den udvidede IF-sætning. Hvis betingelsen er sand (beløb>100) udskrives:
Beløbet er større end 100
Der gives 10 % rabat
Hvorefter beløbet sættes lig 90 % af det gamle beløb.

Bemærk, at man benytter ordet ENDIF til at markere, at her slutter rækken af sætninger, der skal udføres, hvis betingelsen er opfyldt.

Ønsker man at udføre en række sætninger, hvis en betingelse er opfyldt, og en anden række, hvis betingelsen ikke er opfyldt, kan man bruge IF-ELSE-ENDIF konstruktionen.

Eksempel

```
0010 INPUT "Indtast et beløb >": beløb
0020 IF beløb>100 THEN
0030   PRINT "Beløbet er større end 100"
0040   PRINT "Der gives 10 % rabat"
0050   beløb:= beløb*0.9
0060 ELSE
0070   PRINT "Beløbet er mindre end 100"
0080   PRINT "Ekspeditionsgebyret er 10 kr."
0090   beløb:= beløb+10
0100 ENDIF
```

Bemærkninger

linie 0010 : Her indtastes værdien af beløb.

linie 0030-0050 : Disse linier udføres, hvis betingelsen er opfyldt, dvs værdien af beløb er større end 100

linie 0070-0090 : Disse linier udføres, hvis betingelsen ikke er opfyldt.

6.1.2 Multiforgreninger (CASE) 6.1.2

Ved IF-ELSE-ENDIF kan man vælge mellem to alternative sætningslister, men ofte kommer man ud for at skulle kunne vælge mellem flere alternativer. Til dette formål benytte en multiforgrening (CASE). Den består af et nøgleudtryk og en række sætningslister, hvoraf kun en af sætningslisterne vil blive udført, afhængig af værdien af nøgleudtrykket.

Eksempel

```
0010 INPUT "Indtast et tal : ": tal
0020 CASE SGN(tal) OF
0030 WHEN -1
0040   PRINT tal;"er negativ"
0050 WHEN 0
0060   PRINT "0 er nul"
0070 WHEN 1
0080   PRINT tal;"er positiv"
0090 ENDCASE
0100 END
```

Bemærkninger

linie 0010 : Her tildeles tal en værdi fra tastaturet.
linie 0020 : SGN(tal) er her nøgleudtrykket, det er en funktion, der har værdien +1 hvis tal er positiv, 0 hvis tal er nul og -1 hvis tal er negativ.

Nøgleudtrykket kan enten være et numerisk udtryk eller et strengudtryk.

Eksempel

```
0010 DIM fkt$ OF 1
0020 INPUT "Indtast funktion: I(ndsæt,U(dskriv,S(lut ":fkt$
0030 CASE fkt$ OF
0040 WHEN "I","i"
0050   EXEC indsæt
0060 WHEN "U","u"
0070   EXEC udskriv
0080 WHEN "S","s"
0090   EXEC slut
0100 ENDCASE
0110 END
```

Bemærkning

Ovenstående program kan ikke umiddelbart udføres, da PROC indsæt, PROC udskriv og PROC slut ikke er erklæret.

CASE-konstruktionen udføres således, at den starter med den første WHEN-sætning og udregner, om et af udtrykkene efter WHEN er lig nøgleudtrykket. Hvis dette er tilfældet, udføres sætningerne efter WHEN, ellers undersøges udtrykkene efter det andet WHEN osv.

Hvis ingen WHEN-sætning indeholder et udtryk, der er lig nøgleudtrykket, udskrives en fejlmelding.

Dette kan dog undgås ved angivelse af en OTHERWISE-sætning:

Eksempel

```
0010 DIM fkt$ of 1
0020 INPUT "Indtast funktion: I(ndsæt,U(dskriv,S(lut ":fkt$
0030 CASE fkt$ OF
0040 WHEN "I","i"
0050   EXEC indsæt
0060 WHEN "U","u"
0070   EXEC udskriv
0080 WHEN "S","s"
0090   EXEC slut
0095 OTHERWISE
0096   PRINT "*** Funktionen eksisterer ikke"
0100 ENDCASE
0110 END
```

Bemærkning

Tastes der ikke I,i,U,u,S eller s, vil programmet nu udskrive:

*** Funktionen eksisterer ikke

Bemærk, at kun én sætningsliste udføres. Følgende program giver således ingen mening:

```
0010 CASE 2 OF
0020 WHEN 2
0030   PRINT "linie 0030"
0040 WHEN 2
0050   PRINT "linie 0050"
0060 ENDCASE
0070 END
```

WHEN-sætningen i linie 0020 vil "opfange" nøgleværdien fra linie 0010 og linie 0050 vil aldrig blive udført.

Dette kan bruges til store betingede strukturer som f.eks.:

```
0010 INPUT "Indtast brevets vægt > ": brevvægt
0020 IF brevvægt<=20 THEN
0030   porto:=200
0040 ELSE
0050   IF brevvægt<=100 THEN
0060     porto:=270
0070   ELSE
0080     IF brevvægt<=250 THEN
0090       porto:=400
0100     ELSE
0110       IF brevvægt<=500 THEN
0120         porto:=700
0130       ELSE
0140         porto:=1000
0150       ENDIF
0160     ENDIF
0170   ENDIF
0180 ENDIF
0190 PRINT "Portoen er ";porto;"øre."
```

Virkningen af denne konstruktion er den samme som virkningen af følgende CASE-konstruktion:

```
0010 INPUT "Indtast brevets vægt >": brevvægt
0020 CASE TRUE OF
0030 WHEN brevvægt<=20
0040   porto:=200
0050 WHEN brevvægt<=100
0060   porto:=270
0070 WHEN brevvægt<=250
0080   porto:=400
0090 WHEN brevvægt<=500
0100   porto:=700
0110 OTHERWISE
0120   porto:=1000
0130 ENDCASE
0140 PRINT "Portoen er ";porto;"øre."
```

Bemærkning

I linie 0020 sættes nøgleudtrykket til TRUE, dvs. konstanten SAND. Det bevirker, at WHEN-sætningerne gennemløbes, indtil der findes et udtryk, der er sandt. Er intet udtryk sandt, udføres sætningen efter OTHERWISE. Under alle omstændigheder vil kun en af sætningslisterne efter WHEN (eller OTHERWISE) blive udført.

6.2 Løkkestrukturer 6.2

I COMAL80 eksisterer 3 forskellige løkkestrukturer, de er alle beskrevet i det følgende.

6.2.1 Tællesløjfer (FOR-NEXT) 6.2.1

Begrundelsen for at have tællesløjfer ses bedst af et eksempel. Lad os forestille os, at vi har fået til opgave at udskrive en tabel over de 100 første positive tal, deres kvadrattal og deres kubiktal. Dette kunne gøres med følgende program :

```
0010 ZONE 20 // Sæt tabuleringen til 20 tegn
0020 PRINT "X","X*X","X*X*X"
0030 PRINT 1,1*1,1*1*1
0040 PRINT 2,2*2,2*2*2
...
1010 PRINT 99,99*99,99*99*99
1020 PRINT 100,100*100,100*100*100
1030 END
```

I stedet for dette lange program, kan man klare sig med følgende:

```
0010 ZONE 20 // Sæt tabuleringen til 20 tegn
0020 PRINT "X","X*X","X*X*X"
0030 FOR x:=1 TO 100 DO PRINT x,x*x,x*x*x
0040 END
```

Udskriften fra programmet bliver den samme.

Som man kan se, er virkningen af linie 0030 i det andet eksempel den samme som virkningen af linierne 0030,...,1020 i det første eksempel. Linie 0030 virker således:

1) Først sættes x lig 1
2) Dernæst testes, om x er blevet større end slutværdien (x>100). Hvis den er det, forsættes med linie 0040
3) Ellers udføres PRINT-sætningen, den udskriver x og x*x og x*x*x.
4) Derefter forøges x med 1 og man hopper til pkt 2)

Læg mærke til, at x forøges med 1 (trinværdien er 1). Ønskes en anden trinværdi (f.eks. 10) får sætning 30 følgende udseende :

```
0030 FOR x:=1 TO 100 STEP 10 DO PRINT x,x*x,x*x*x
```

Hermed får x værdierne 1,11,21,31,...,91. 91 er den sidste værdi, for hvis man lægger 10 til, er værdien større end 100 og slutbetingelsen opnået.

Trinværdien kan være negativ. Hvis den er det, vil løkken fortsætte, indtil x (også kaldet tællevariablen) er mindre end slutværdien.

Startværdien,slutværdien og trinværdien behøver ikke at være talkonstanter. Det er også tilladt at angive numeriske udtryk:

```
0030 FOR x:=i/10+1 TO SIN(y*2)*p DO ...
```

Man skal blot være opmærksom på, at udtrykket kun udregnes når man løber ind i løkken. Derefter huskes værdierne som konstanter.

Ønsker man at udføre mere end en sætning, kan man benytte den udvidede FOR-NEXT sætning. Den har følgende struktur :

```
0030 FOR x:=1 TO 100 DO
0031   PRINT x,x*x,x*x*x
0032 NEXT x
```

Alle sætninger mellem FOR og NEXT udføres for hvert gennemløb af løkken. Det er desuden tilladt at have FOR-NEXT løkker indeni hinanden. Hvis vi fortsætter med overstående program, kunne det have følgende udformning.

```
0030 FOR x:=1 TO 100 DO
0031   FOR potens := 1 to 3 do PRINT x↑potens,
0032   PRINT
0033 NEXT x
```

6.2.2 REPEAT-konstruktionen 6.2.2

Hvor FOR-NEXT sløjfen gentager en række sætninger på grundlag af en tællevariabel, kan man også gentage en række sætninger indtil et betingelse bliver opfyldt.

Eksempel

```
0010 RANDOMIZE
0020 ZONE 20
0030 PRINT "Terning 1","Terning 2"
0040 REPEAT
0050   terning1:=RND(1,6)
0060   terning2:=RND(1,6)
0070   PRINT terning1,terning2
0080 UNTIL terning1=terning2
0090 END
```

Bemærkninger

Programmet simulerer kast med to terninger. Programmet fortsætter med at køre, indtil det indtræffer, at de to terninger viser lige mange øjne.

linie 0010 : RANDOMIZE bevirker, at de "tilfældige tal" bliver forskellige fra gang til gang.

linie 0020 : Udskriftszonen sættes til 20 tegn.

linie 0030 : Overskrift udskrives

linie 0040-0080 : REPEAT-sløjfen. Sætningerne i linie 0090-0110 bliver gentaget indtil værdien af terning1 er lig værdien af terning2.

linie 0050-0060 : "Kast" af de to terninger. RND(1,6) er en funktion der returnerer med et tilfældigt heltal i intervallet 1,2,...,6

linie 0070 : Udskrift af antallet af terningernes øjne.

Man bør bemærke, at sætningen mellem REPEAT og UNTIL altid bliver udført mindst 1 gang . Dette gør, at konstruktionen ofte benyttes i forbindelse med INPUT og kontrol af det indtastede.

Eksempel

```
0010 DIM svar$ OF 3
0020 REPEAT
0030   INPUT "Ønsker du at fortsætte ? ": svar$
0040   IF svar$<>"ja" AND svar$<>"nej" THEN
0050     PRINT "Svar venligst ja eller nej"
0060   ENDIF
0070 UNTIL svar$="ja" OR svar$="nej"
```

Bemærkninger

linie 0010 : Alle strengvariable skal erklæres

linie 0030 : Her indlæses værdien af svar$ fra tastaturet

linie 0040-0060 : Hvis svar$ ikke er lig teksten "ja" eller "nej" udskrives:
Svar venligst ja eller nej

linie 0020-0070 : Linierne gentages, indtil svar$ er enten "ja" eller "nej"

6.2.3 WHILE-konstruktionen 6.2.3

I stil med REPEAT er WHILE en konstruktion, der gentager udførelsen af en eller flere sætninger på grundlag af en betingelse. Forskellen er, at man ved WHILE konstruktionen fortsætter sålænge en betingelse er opfyldt

Eksempel

```
0010 INPUT "Indtast det indsatte beløb : ": startbeløb
0020 INPUT "Indtast det ønskede beløb  : ": slutbeløb
0030 INPUT "Indtast rentesatsen        : ": rente
0040 PRINT "Termin           Saldo"
0050 saldo:=startbeløb; termin:= 0
0060 WHILE saldo<slutbeløb DO
0070   termin:= termin+1
0080   saldo:= saldo*(1+rente/100)
0090   PRINT USING "#####    ##########.##":termin,saldo
0100 ENDWHILE
```

Bemærkninger

Programmet udregner den tid der går før et indsat beløb i en bank får vokset sig op til et ønsket slutbeløb, forudsat at renten er konstant.

linie 0010-0030 : Her indtastes de ønskede værdier for startbeløbet, slutbeløbet og renten.

linie 0040 : Kolonneoverskrift

linie 0050 : Saldoen sættes lig det indsatte beløb, og terminnummeret sættes til nul.

linie 0060-0100 : Linierne repeteres, sålænge saldo er mindre end slutbeløb

linie 0070 : Terminen forøges med 1 periode

linie 0080 : Der lægges renter til saldoen

linie 0090 : Formatteret udskrift, der bevirker, at tallene kommer til at stå under hinanden (enere under enere, tiere under tiere osv.)

7. INDLÆSNING OG UDSKRIVNING 7.

Dette afsnit skal opfattes som en kort fremstilling af de muligheder man har for udskrift og indlæsning på skærm, printer, tastatur og diskettefiler.

En fuldstændig gennemgang af kommandoerne findes i kap 10 i følgende afsnit :

- AT (10.3)
- CLOSE (10.8)
- CREATE (10.13)
- DELETE (10.16)
- EOF (10.26)
- GET$ (10.36)
- INPUT (10.43)
- INPUT FILE (10.44)
- OPEN (10.56)
- PREFIX (10.59)
- PRINT (10.60)
- PRINT FILE (10.61)
- READ FILE (10.69)
- SELECT OUTPUT (10.79)
- TAB (10.87)
- WRITE FILE (10.94)

7.1 INPUT-sætningen 7.1

Når man skal indlæse tal eller tekst direkte fra tastaturet, kan det gøres med sætninger som f.eks.:

```
0010 INPUT alder,højde
```

Når denne sætning udføres, fremkommer cursoren på skærmen som tegn på, at systemet forventer indtastning fra tastaturet.

Brugeren kan derefter indtaste værdien for alder, efterfulgt af et komma (,) og værdien for højde, efterfulgt af vognretur, f.eks.

```
14,160
```

Da det kan være svært at huske rækkefølgen for inddata kan man få udskrevet en ledetekst med følgende sætning:

```
0010 INPUT "Indtast alder og højde > ":alder,højde
```

Når denne sætning udføres, udskrives

```
Indtast alder og højde >
```

Cursoren fremkommer tillige på skærmen, som et tegn på, at systemet forventer indtastning fra tastaturet. Herefter kan brugeren indtaste de ønskede værdier.

I anførselstegnene må der angives en vilkårlig tekst, som dog ikke selv må indeholde anførselstegn.

7.2 PRINT-sætningen 7.2

Når man skal udskrive resultater på skærmen benyttes PRINT-sætningen. Sætningen kan udskrive

- tekst
- værdien af et udtryk
- værdien af variable eller konstanter
- tomme linier

Efter PRINT kan man anføre en række elementer, adskilt med komma eller semikolon. Kommaet deler hver udskriftslinie i en række kolonner (zoner), hvis bredde kan angives med ZONE-sætningen.

Eksempel 1

```
0010 ZONE 10
0020 PRINT "x","tekst"
```

Når ovenstående program udføres, får følgende udskrift :

```
x          tekst
```

x udskrives i kolonne 1, tekst i kolonnerne 11 til 15. Udskriftskolonnerne er således 10 tegn i bredden. Hvis ZONE ikke angives, anvendes ZONE 0.

Angives semikolon mellem elementerne sættes 1 mellemrum, hvis foregående element er et tal, ellers sættes intet mellemrum.

Eksempel 2

```
0010 PRINT "RC";35*20;"Piccolo"
```

Når ovenstående sætning udføres, fås følgende udskrift:

```
RC700 Piccolo
```

7.3 Datastrømme 7.3

7.3.1 Filer og ydre enheder 7.3.1

På disketten kan man både lagre programmer og data. For at kunne gøre dette, skal programmet eller dataene have et navn. Et område, hvor program eller data gemmes, kaldes for en fil.

Et navn på en fil har følgende format :

```
<unit>/<navn>
```

<unit> er disketteenhedens nummer (altså 1 eller 2)
<navn> er et navn bestående af maximalt 11 tegn. Navnet må bestå af alle tegn bortset fra følgende tre: "?/ .

Lovlige filnavne:	Ulovlige filnavne:
1/HANOI	7/Mitprogram
2/Demoprogram	2/demoprog?

Ofte arbejder man kun på en bestemt disketteenhed ad gangen. I stedet for hele tiden at angive <unit>/ i alle navne kan man selv erklære et såkaldt præfix, som er en tekststreng, der vil blive sat foran filnavnene i sætningerne.

Eksempel

```
PREFIX "1/NBA"
LOAD "PROGRAM"
svarer til
PREFIX ""
LOAD "1/NBAPROGRAM"
```

Efter opstart af COMAL80 udfører systemet kommandoen

```
PREFIX "1/"
```

Hvis man alligevel ikke ønsker at benytte et eventuelt præfix, skal det første tegn i filnavnet være /.

Eksempel	Filnavn
PREFIX "1/"	
LOAD "/2/DEMO"	2/DEMO
SAVE "PROGRAM"	1/PROGRAM

De ydre enheder har tillige navne, så de kan bruges som filer. Filer og ydre enheder udgør tilsammen datastrømme.

Navnene på de ydre enheder er:

Navn	Enhed
1/console	Tastatur og skærm. Indlæses der fra tastaturet, vises tegnene på skærmen.
1/keyboard	Tastatur. Når der indlæses, vises tegnene ikke på skærmen.
1/printer	Skriver.
<portnr>/port	En af systemet HW-porte.

Omkring brugen af disse enheder henvises til afsnit 7.3.3.3.

7.3.2 Skrivning og læsning af programfiler 7.3.2

Hvis man har indtastet et program og ønsker at gemme det på disketten, kan dette gøres med kommandoen

SAVE <filnavn>

eller med kommandoen

LIST <filnavn>

hvor <filnavn> er navnet på en datastrøm angivet i anførselstegn (se afsn 7.3.1).

SAVE-kommandoen bevirker, at indholdet af programlageret bliver udskrevet direkte på disketten (binært), medens LIST-kommandoen bevirker, at programmet bliver udskrevet i tekstform på disketten (ASCII). LIST-kommandoen kan tillige bruges til at udskrive programmet på printeren.

Eksempel

LIST "printer"

Forudsat at PREFIX er sat til "1/" vil ovenstående kommando bevirke, at programmet i programlageret vil blive udskrevet på printeren.

Programmet kan hentes igen fra disketten med følgende kommandoer

LOAD <filnavn>

eller

ENTER <filnavn>

LOAD-kommandoen anvendes, hvis programmet er gemt med SAVE, og ENTER-kommandoen anvendes, hvis programmet er gemt med LIST.

LOAD-kommandoen sletter programlageret før den indlæser programmet, medens ENTER-kommandoen indlæser programmet oveni det program, der eventuelt ligger i programlageret.

Bemærk at ved LOAD af et program udskrives navnet på den LOADede fil i statuslinien øverst på skærmen. Hvis man herefter foretager nogle rettelser i programmet og ønsker at gemme det igen under samme navn, skrives blot

SAVE

hvorefter det rettede program vil erstatte det forrige.

SAVE/LOAD bør anvendes frem for LIST/ENTER, da

- Udskrivning og indlæsning går hurtigere
- Filerne normalt fylder mindre

7.3.3 Skrivning og læsning af datafiler 7.3.3

Der eksisterer to typer datafiler:

1. Sekventielle filer
2. Filer med direkte tilgang (RANDOM-filer)

En sekventiel fil er en datafil, hvor data kun kan læses eller skrives i en bestemt rækkefølge. Man kan sammenligne en sekventiel fil med et bånd på en båndoptager, hvor man kun kan spole helt tilbage, og så afspille forfra, hvis man har brug for noget inde på båndet.

En fil med direkte tilgang er en datafil, hvor man kan læse en vilkårlig del af filen, uden først at have læst det foregående. Man kan sammenligne en fil med direkte tilgang med en grammofonplade på en grammofon, hvor man kan flytte pickup-armen direkte ind til det, man har "brug" for.

Den generelle håndtering er den samme for begge filtyper, nemlig:

- Før brug skal filen oprettes med en CREATE-sætning (dette gøres dog automatisk ved sekventielle filer)
- I et program skal følgende trin udføres, hvis en fil bruges:
    1. "Åbning" med en OPEN-sætning
    2. Læsning/skrivning i filen
    3. "Lukning" med en CLOSE-sætning

7.3.3.1 Sekventielle filer

Når man skal bruge en sekventiel fil i programmet skal den åbnes (OPEN), dvs at filen skal knyttes til et såkaldt strømnummer i programmet. Formatet for en OPEN-sætning er :

OPEN <strømnr>,<filnavn>,<mode>

hvor
<strømnr> er et numerisk udtryk lig enten 1,2,3,4 eller 5
<filnavn> er navnet på filen
<mode> er enten WRITE,APPEND eller READ

Betydningen af <mode> er følgende:

- WRITE : Filen oprettes, og der skrives forfra i den
- APPEND : Der skrives i forlængelse af det, der tidligere er skrevet ud i filen.
- READ : Der læses forfra i filen.

Når man har åbnet filen, kan man enten læse eller skrive i den. Enhver reference til filen foregår via <strømnr>.

Man kan skrive i filen på forskellige måder
- WRITE FILE bevirker binært udskrift af data
- PRINT FILE bevirker tekstudskrift af data, dvs. i samme format som ved en simpel PRINT-sætning (ASCII-format).

WRITE FILE anvendes normalt i forbindelse med udskrift i diskettefiler, hvorimod PRINT FILE normalt benyttes i forbindelse med ydre enheder som f.eks. skærm og printer.

Eksempel

```
0010 OPEN FILE 1,"DATAFIL",WRITE
0020 REPEAT
0030   INPUT "Indtast et tal (0=slut) ": tal
0040   WRITE FILE 1:tal
0050 UNTIL tal=0
0060 CLOSE FILE 1
```

Bemærkninger

Programmet modtager tal fra tastaturet, og skriver dem i filen DATAFIL.

Linie 0010 : Filen oprettes og åbnes til skrivning

Linie 0020-0050 : Tal indtastes, gemmes i filen og hvis den indtastede værdi er nul hoppes ud af løkken. Nul er den sidste værdi, der gemmes i filen.

Linie 0060 : Filen lukkes efter brug

Herefter kan dataene indlæses igen. Det kan gøres med følgende program :

Eksempel

```
0010 OPEN FILE 1,"DATAFIL",READ
0020 WHILE NOT EOF(1) DO
0030   READ FILE 1:tal
0040   IF NOT EOF(1) THEN PRINT tal
0050 ENDWHILE
0060 CLOSE FILE 1
```

Bemærkninger

Linie 0010 : Filen åbnes til læsning

Linie 0020-0050 : Her benyttes funktionen EOF. Det er en logisk funktion, der bliver sand, når det sidste dataelement er læst. Linie 0030 og 0040 bliver således udført sålænge der er data i filen.

Linie 0030 : READ FILE er læsesætningen, der læser data skrevet med WRITE FILE.

Linie 0060 : Filen lukkes efter brug

READ FILE indlæser de data, der er udskrevet ved hjælp af WRITE FILE, og man må gerne angive flere argumenter i både READ FILE og WRITE FILE sætningerne. De enkelte argumenter skal da blot være adskilt med et komma (,).

Filer, der er skrevet med PRINT FILE kan indlæses med INPUT FILE. Her skal man blot erindre, at formatet i filen nøjagtig svarer til det format, man får med simple PRINT sætninger f.eks. på skærmen. Kommaer i parameterlisten vil derfor kun bevirke tabulering til næste PRINT-kolonne, hvilket kan give fejl i forbindelse med strengbehandling.

Hvis man ønsker at benytte PRINT FILE og INPUT FILE er det sikreste således kun at angive et PRINT- eller INPUT-argument pr. linie, da man derved er sikker på, at de enkelte argumenter bliver adskilt med vognretur i filen.

Når man er færdig med at læse eller skrive i filen, skal filen lukkes. Det gøres med sætningen

CLOSE FILE <strømnr>

der lukker én bestemt strøm, eller

CLOSE

der lukker alle åbne strømme.

7.3.3.2 Filer med direkte tilgang

7.3.3.2

En fil med direkte tilgang består af en række nummererede poster, der kan hentes hver for sig. På forhånd skal brugeren fastlægge, hvad en post skal indeholde og dermed udregne dens størrelse.

Ved udregning af poststørrelsen skal man benytte, at

- en numerisk variabel fylder 8 tegn
- en streng fylder "længden af strengen + 2" tegn

Eksempel

Hvis vi tænker os følgende dimensionering

```
DIM navn$(30),klasse$(5)
```

og lader nr,hægte betegne numeriske variable, da gælder:

Post	Poststørrelse
nr,navn$,klasse$	8+(30+2)+(5+2)=47
nr,hægte	8+8=16

Desuden skal brugeren vurdere det maximale antal poster i filen. Når dette er gjort, kan filen oprettes. Det gøres med sætningen

```
CREATE <filnavn>,<længde>
```

<filnavn> er en strengvariabel eller en strengkonstant der indeholder navnet på filen,
<længde> er et numerisk udtryk, der angiver filens længde i blokke a 1024 tegn.

Eksempel

```
CREATE "MEDLEMSDATA",20
```

Et udtryk for filens længde får man lettest ved at benytte formlen:

```
fillængde:=(poststørrelse*antalposter)/1024 + 1
```

Når man skal bruge filen i programmet skal den åbnes (OPEN), dvs at filen skal knyttes til et såkaldt strømnummer i programmet. I OPEN-sætning skal man ikke angive, om man ønsker at læse eller at skrive i filen, da begge dele er tilladt, når filen er åbnet.

Formatet for en OPEN-sætning er :

OPEN FILE <strømnr>,<filnavn>,RANDOM <recl>

hvor
<strømnr> er et numerisk udtryk, der enten er lig 1,2,3,4 eller 5.
<filnavn> er en strengvariabel eller en strengkonstant, der indeholder navnet på filen,
<recl> er et numerisk udtryk, der angiver poststørrelsen i tegn (bytes).

Når man har åbnet filen, kan man både læse og skrive i den.

Formatet for skrivning er:

WRITE FILE <strømnr>,<postnr>:<postbeskrivelse>

og formatet for læsning er:

READ FILE <strømnr>,<postnr>:<postbeskrivelse>

hvor
<strømnr> er det nummer, der anvendtes ved OPEN
<postnr> er den post, man ønsker at læse/skrive. Første post har nummeret 1.
<postbeskrivelse> er de argumenter, der skal læses/skrives.

Eksempel

```
0010 READ FILE 1,17:nr,navn$,klasse$
0020 WRITE FILE 2,92:nr,hægte
```

Efter brug skal filen lukkes igen ved hjælp af en CLOSE-sætning. Formatet er enten

CLOSE FILE <strømnr>

hvis man ønsker at lukke én bestemt strøm, eller

```
CLOSE
```

hvis man ønsker at lukke alle åbne strømme.

7.3.3.3 Ydre enheder

7.3.3.3

De ydre enheder skal behandles som normale sekventielle tekstfiler, dvs de åbnes med enten

```
OPEN <strømnr>,<navn>,READ
```

eller

```
OPEN <strømnr>,<navn>,WRITE
```

Man kan normalt kun skrive med PRINT FILE-sætningen samt læse med enten INPUT FILE-sætningen eller GET$-funktionen.

GET$-funktionen læser et angivet antal tegn fra en strøm.

Eksempel

```
0010 DIM t$ OF 1
0020 OPEN 1,"keyboard",READ
0030 REPEAT
0040   t$:=GET$(1,1)
0050   IF t$>="0" AND t$<="9" THEN PRINT t$;
0060 UNTIL t$<"0" OR t$>"9"
```

Bemærkninger

linie 0010 : Her dimensioneres en streng til at kunne indeholde eet tegn.

linie 0020 : Den ydre enhed "keyboard" åbnes. Når der indlæses tegn fra tastaturet, vises de ikke på skærmen.

linie 0040 : Der indlæses et tegn fra enheden "keyboard".

linie 0050 : Hvis tegnet er et tal, udskrives det på skærmen.

linie 0060 : Linierne 0030-0060 gentages indtil det indtastede tegn ikke er et tal.

På HW-portene er det muligt både at læse og skrive. For at spare på antallet af strømme er det lovligt både at læse og skrive på den samme strøm, selvom den er åbnet i enten READ- eller WRITE-mode.

For ikke at få automatiske linieskift i forbindelse med HW-port udskrivning, skal man huske at sætte sidebredden til 0 med sætningen

```
MARGIN 0
```

7.3.4 Fjernelse af filer

7.3.4

En fil kan fjernes fra disketten igen med sætningen

```
DELETE <filnavn>
```

Eksempel

```
0010 PREFIX "1/"
0020 DELETE "/2/Demoprogram"
0030 DELETE "HANOI"
```

7.3.5 Omdøbning af filer 7.3.5

Ønsker man at give filen et andet navn, kan den omdøbes med sætningen

```
RENAME <gl filnavn>,<nyt filnavn>
```

hvor
<gl filnavn> er filens nuværende navn
<nyt filnavn> er det ønskede filnavn uden angivelse af unit i navnet

Eksempel 2

```
RENAME "/2/Demoprogram","LOGON"
0010 RENAME "datafil","gldatafil"
```

Ulovlige omdøbninger:	Bemærkninger
RENAME "/2/Demo","/2/prog"	Man må ikke angive unitnr i <nyt filnavn>
RENAME "F","fildemonstration"	Det nye navn skal være lovligt

7.4 Oversigt over I/O sætninger 7.4

Skemaet i dette afsnit giver et overblik over den række indlæsnings- og udskrivnings- (I/O-) sætninger og kommandoer, der eksisterer i COMAL80.

Medium/Datatype	Indlæsning	Udskrivning	Format
Tastatur og skærm	INPUT	PRINT	ASCII
	KEY$		Direkte fra tastatur
Datastrømme	INPUT FILE	PRINT FILE	ASCII
	READ FILE	WRITE FILE	Binært
	GET$		Transparent
Programmer	ENTER	LIST	ASCII
	LOAD	SAVE	Binært

8. PROCEDURER OG FUNKTIONER 8.

Når man skal løse større problemer, er det oftest hensigtsmæssigt at splitte det store problem op i delproblemer, og løse disse hver for sig.

Til dette formål er procedurebegrebet udviklet.

For at illustrere anvendelsen af procedurer, betragter vi følgende programmeringsopgave:

> På en skole skal oprettes et register over eleverne og disses adresser. Til dette formål skal vi udarbejde et program, der kan:
>
> - indsætte nye elevers data
> - slette elevers data
> - rette i oplysningerne for en elev

Programmet har da følgende hovedstruktur:

```
opstart
GENTAG følgende:
     INDLÆS kommando
     HVIS kommando=ind SÅ
          få elevoplysninger
          find ledigt elevnr
          gem elevoplysninger
     HVIS kommando=ret SÅ
          få elevnr
          hent elevoplysninger på diskette
          ret i oplysninger
          gem elevoplysninger
     HVIS kommando=slet SÅ
          få elevnr
          hent elevoplysn
          slet elevoplysninger
INDTIL kommando=færdig
afslut
```

Programmet kunne se således ud i COMAL80:

```
0010 EXEC opstart
0020 REPEAT
0030   PRINT
0040   INPUT "Kommando: I(nd,R(et,S(let,F(ærdig ":k$
0050   PRINT
0060   CASE k$ OF
0070   WHEN "I","i"
0080     EXEC fåelevoplysn
0090     EXEC findledigtnr(elevnr)
0100     EXEC gemelev(elevnr)
0110   WHEN "R","r"
0120     EXEC fåelevnr
0130     EXEC hentelev(elevnr)
0140     EXEC retelev
0150     EXEC gemelev(elevnr)
0160   WHEN "S","s"
0170     EXEC fåelevnr
0180     EXEC hentelev(elevnr)
0190     EXEC sletelev(elevnr)
0200   WHEN "F","f"
0210     // Slut på program
0220   OTHERWISE
0230     PRINT "*** Ulovlig kommando"
0240   ENDCASE
0250 UNTIL k$="F" OR k$="f"
0260 EXEC afslut
```

Bemærkninger

EXEC betyder "udfør" og navnet bagefter refererer til et procedurenavn, der beskrives i det følgende. Læg mærke til, at flere procedurer (fåelevnr,hentelev,gemelev) kaldes flere gange. Procedurerne befinder sig andre steder i programmet, eller de er gemt på disketten som externe procedurer.

8.1 Simple procedurer 8.1

En af de procedurer som anvendes i programmet er proceduren fåelevoplysn. Den kunne f.eks. se således ud:

Eksempel

```
0400 PROC fåelevoplysn
0410   REPEAT
0420     INPUT "Klasse    :": klasse$
0430     INPUT "Navn      :": navn$
0440     INPUT "Adresse   :": adresse$
0450     INPUT "Postnr by :": postnrby$
0460     INPUT "Er oplysningerne korrekte ? (J/N) ": svar$
0470   UNTIL svar$="J" OR svar$="j"
0480 ENDPROC fåelevoplysn
```

Bemærkninger

linie 0400-0480 : Her defineres proceduren fåelevoplysn. Når der i hovedprogrammet står EXEC fåelevoplysn, vil sætningerne mellem PROC og ENDPROC blive udført. Proceduren indlæser de enkelte oplysninger for en elev, og giver brugeren mulighed for at ændre alt det indtastede.

En procedure må gerne kalde sig selv. I dette tilfælde kaldes proceduren rekursiv.

Til vores program har vi desuden brug for en procedure, der kan indlæse et nummer på en elev, og checke, om eleven eksisterer i registeret.

Eksempel

```
0290 PROC fåelevnr
0300   REPEAT
0310     INPUT "Elevnr : ":elevnr
0320     ok:=FALSE
0330     IF elevnr>=1 AND elevnr<=max THEN
0340       ok:=(status$(elevnr:elevnr)="O") // optaget
0350     ENDIF
0360     IF NOT ok THEN PRINT "*** Eleven eksisterer ikke"
0370   UNTIL ok
0380 ENDPROC fåelevnr
```

Bemærkninger

linie 0290-0380 : Her defineres proceduren fåelevnr. Når der i hovedprogrammet står EXEC fåelevnr, vil sætningerne mellem PROC og ENDPROC blive udført.

linie 0340 : status er en streng med længden max, der har følgende betydning.
status$(n:n)="O" : elevnr n optaget
status$(n:n)="S" : elevnr n slettet
status$(n:n)="L" : elevnr n ledig.

linie 0300-0370 : Man løber i sløjfen indtil brugeren angiver et elevnr, der er optaget (dvs. i brug).

8.2 Parameteroverførsel 8.2

Ofte er det nødvendigt at overføre data fra hovedprogrammet til proceduren. Dette kan f.eks. ske som i proceduren hentelev:

Eksempel

```
0560 PROC hentelev(nr)
0570   READ FILE 1,nr: klasse$,navn$,adresse$,postnrby$
0580 ENDPROC hentelev
```

Bemærkninger

Denne procedure har en forskel i forhold til de tidligere viste procedurer, idet der er angivet en variabel i en parentes efter procedurens navn. Denne variabel kaldes den formelle parameter til proceduren. Hvis man f.eks. kalder proceduren med sætningen EXEC hentelev(8) vil nr i linie 0560 blive erstattet med tallet 8. Kaldes proceduren med sætningen EXEC hentelev(x) vil nr blive erstattet med værdien af x.

Som almindelige parametre, kan man overføre numeriske variable og strengvariable.

8.3 REF angivelse 8.3

Hvis proceduren producerer en værdi, som skal tilbage til hovedprogrammet, kan dette gøres ved at angive REF foran parameteren.

Eksempel

```
0500 PROC findledigtnr(REF nr)
0510   nr:=1
0520   WHILE nr<=max AND status$(nr:nr)="0" DO nr:=nr+1
0530   PRINT "Eleven har fået nummer ";nr
0540 ENDPROC findledigtnr
```

Bemærkninger

linie 0500 : Før parameteren nr står ordet REF. Det angiver, at ikke blot værdien af parameteren i en tilsvarende EXEC sætning skal overføres, men at værdien af nr skal returneres til den kaldende parameter. Dette betyder, at hvis proceduren kaldes med sætningen EXEC findledigtnr(elevnr), vil variablen elevnr have samme værdi som nr, når der returneres fra proceduren.

linie 0520 : Her fremfindes et nr der ikke er optaget.

Da man returnerer værdier gennem REF-parametre, må man ikke kalde proceduren med andet end variable som parametre. Følgende kald er derfor ulovligt :

```
EXEC findledigtnr(8)
```

Et forslag til programmet, der er blevet skitseret i begyndelsen af dette kapitel, er vist i appendix F.

Hvis en parameter i en procedure er en vektor, en matrix eller en teksttabel, skal ordet REF altid angives foran parameteren.

Eksempel

```
0010 PROC skrivtabel(REF a(),dima)
0020   FOR i:=1 TO dima DO PRINT a(i),
0030   PRINT
0040 ENDPROC skrivtabel
0050 INPUT "Antal elever ":n
0060 DIM alder(n),højde(n)
0070 MARGIN 80
0080 ZONE 10
0090 FOR nr:=1 TO n DO
0100   INPUT "Alder ? ":alder(nr)
0110   INPUT " Højde ? ":højde(nr)
0120 NEXT nr
0130 PRINT "Alder:"
0140 EXEC skrivtabel(alder,n)
0150 PRINT "Højde:"
0160 EXEC skrivtabel(højde,n)
```

Bemærkninger

linie 0010-0040: Her erklæres en procedure med navnet skrivtabel. Proceduren har to parametre. Den første er en endimensional taltabel (en vektor), den anden er et tal. Parentesen efter a-et markerer, at parameteren er en vektor. Proceduren udskriver elementerne i a ud, fra element nr 1 til element nr dima.

linie 0050: Indlæsning af antallet af elever

linie 0060: Dimensionering af to vektorer.

linie 0070: Sidebredden sættes til 80 tegn.

linie 0080: Kolonnebredden sættes til 10 tegn.

linie 0090-0120: Indlæsning af alder og højde.

linie 0130: Udskrift af teksten Alder:

linie 0140: Kald af proceduren skrivtabel. Første parameter er vektoren alder, anden parameter er antallet af elementer i alder.

linie 0150: Udskrift af teksten Højde:

linie 0160: Kald af proceduren skrivtabel. Første parameter er nu vektoren højde, antan parameter er antallet af elementer i højde.

Hvis parameteren til en procedure skal være en matrix skrives f.eks. REF a(,) i procedurehovedet.

Parametertype	Eksempel på procedurehoved	Eksempel på procedurekald
Tal	PROC p(i)	EXEC p(7) eller
		EXEC p(j)
	PROC p(REF i)	EXEC p(j)
Vektor	PROC p(REF v())	EXEC p(v)
Matrix	PROC p(REF m(,))	EXEC p(m)
Streng	PROC p(s$)	EXEC p("tekst") eller
		EXEC p(s$)
	PROC p(REF s$)	EXEC p(s$)
Teksttabel	PROC p(REF t$())	EXEC p(t$)

8.4 Lukkede procedurer 8.4

Ofte bliver procedurer små selvstændige programmer med egne variabelnavne. For at undgå konflikt med hovedprogrammets variable, kan procedurene gøres lukkede.

Vi forestiller os, at vi skal lave et program, der skriver 10 linier med 10 stjerner på hver linie. Man kunne da lave følgende program:

Eksempel

```
0010 PROC stjerner
0020   FOR x:=1 TO 10 DO PRINT "*";
0030   PRINT
0040 ENDPROC stjerner
0050 FOR x:=1 TO 10 DO
0060   EXEC stjerner
0070 NEXT x
0080 END
```

Udføres ovenstående program, fås følgende udskrift:

```
**********
```

Altså kun én linie med 10 stjerner. Det som går galt i programmet er, at variablen med navnet x forsøges anvendt som tællevariabel i 2 sløjfer indeni hinanden. Når stjerner er blevet udført én gang, vil x have værdien 10, og slutbetingelsen for sløjfen i linierne 0050-0070 er opnået. Problemet kan løses, ved man angiver, at variablene i proceduren stjerner alle skal være lokale og ikke have noget med hovedprogrammets variable at gøre. Dette gøres ved angivelse af ordet CLOSED efter PROC stjerner. Proceduren er dermed en lukket procedure.

Eksempel

```
0010 PROC stjerner CLOSED
0020   FOR x:=1 TO 10 DO PRINT "*";
0030   PRINT
0040 ENDPROC stjerner
0050 FOR x:=1 TO 10 DO
0060   EXEC stjerner
0070 NEXT x
0080 END
```

Udføres ovenstående program, fås følgende udskrift:

```
**********
**********
**********
**********
**********
**********
**********
**********
**********
**********
```

Og dermed er problemet løst.

Man kan komme ud for, at man alligevel gerne vil have adgang til visse af hovedprogrammets variable eller procedurer fra en lukket procedure. Dette gøres med GLOBAL sætningen indeni den lukkede procedure.

Eksempel

```
0010 PROC p CLOSED
0020   GLOBAL j
0030   i:=10
0040   j:=10
0050 ENDPROC p
0060 i:=1; j:=1
0070 EXEC p
0080 PRINT "i=";i;" j=";j
```

Bemærkninger

linie 0010-0050: Her erklæres en lukket procedure med navnet p.

linie 0020 : Sætningen angiver, at den variabel j, der benyttes i proceduren er det samme som hovedprogrammets j.

linie 0030 : i sættes lig 10. Da i ikke optræder i GLOBAL sætningen, er det en lokal variabel, der er uafhængig af hovedprogrammets variabel i.

linie 0040 : j sættes lig 10. Da j er angivet i GLOBAL sætningen, er det hovedprogrammets j, der sættes lig 10.

linie 0060 : i og j i hovedprogrammet sættes lig 1

linie 0070 : proceduren p kaldes

linie 0080 : værdien af i og j udskrives. Hovedprogrammets i er ikke ændret i proceduren p, det er j derimod.

Udføres ovenstående program fås således følgende udskrift:

```
i=1  j=10
```

8.5 Externe procedurer 8.5

I et større program vil der ofte være procedurer, der ikke benyttes særlig ofte. For at spare programlagerplads kan disse defineres som externe procedurer og placeres på en diskette. Proceduren optager da kun plads i programlageret, når den kaldes, og pladsen kan senere overtages af andre externe procedurer.

Eksempel

Under navnet PROC1 gemmes følgende procedure:

```
0010 PROC a CLOSED
0020   PRINT "a"
0030 ENDPROC a
```

Under navnet stjerne gemmes følgende procedure:

```
0010 PROC stjerne(n) CLOSED
0020   FOR i:=1 TO n DO PRINT "*";
0030   PRINT
0040 ENDPROC stjerne
```

programlageret:

```
0010 PROC a EXTERNAL "PROC1"
0020 PROC stjerne(n) EXTERNAL "stjerne"
0030 EXEC a
0040 EXEC stjerne(10)

RUN
a
**********

END
at 0040
```

Bemærkninger

PROC1 og stjerne er begge navne på diskettefiler, der indeholder de tilhørende procedurer. Procedurene er skrevet på disketten med kommandoen SAVE.

fil PROC1:

linie 0010-0030 : En extern procedure skal altid erklæres CLOSED.

linie 0020 : Proceduren udskriver et a før den returnerer.

fil stjerne:

linie 0010 : Proceduren har én parameter, n.

linie 0020 : Udskrift af n stjerner.

programlageret:

linie 0010 : Erklæring af, i hvilken fil proceduren befinder sig.

linie 0020 : Erklæring af, i hvilken fil proceduren befinder sig, samt en angivelse af, hvilke parametre proceduren har.

linie 0030 : Kald af PROC a

linie 0040 : Kald af PROC b

Flere externe procedure må ikke gemmes i samme programfil på disketten

Externe procedurer er meget nyttige, idet brugeren hermed har mulighed for at lave sit eget procedurebibliotek, hvor de enkelte procedurer i biblioteket kan benyttes af en række forskellige programmer.

8.6 Handler I/o fejl. 8.6

En HANDLER er en COMAL80 struktur, der har form som en procedure. Forskellen er blot, at man ikke kan "kalde" en HANDLER med EXEC, men at COMAL80 systemet "kalder" strukturen, når der opstår en fejl.

Eksempel

```
0010 PROC fejl HANDLER
0020   PRINT AT(60,1);CHR$(144);" *** TAL forventet ";CHR$(128)
0030   RETRY
0040 ENDPROC fejl
0050 PRINT CHR$(12)
0060 ENABLE fejl
0070 INPUT AT(10,10),"Indtast et tal :": tal
0080 PRINT AT(60,1);CHR$(31) // Slet eventuel fejlmed.
0090 DISABLE
0100 ...
```

Proc fejl Handler if fejl. – skal være tal.

Bemærkninger

linie 0010-0040 : Her defineres en PROC-HANDLER. Den udskriver teksten *** TAL forventet i øverste højre hjørne på skærmen.

linie 0040 : RETRY betyder, at programudførslen fortsætter med selve sætningen hvori fejlen opstod.

linie 0050 : Skærmen slettes

linie 0060 : HANDLERen fejl aktiveres.

linie 0070 : INPUT-sætning hvori der kan opstå fejl, hvis der ikke indtastes et tal, men f.eks. en tekst

linie 0080 : En eventuel fejlmeddelelse slettes

linie 0090 : HANDLERen deaktiveres, og systemet overtager fejlmeddelelserne igen.

Man må gerne erklære flere HANDLERe i samme program, men der kan selvfølgelig kun være en aktiv ad gangen. ENABLEr man en HANDLER medens en anden HANDLER er ENABLEd, DISABLEs den første HANDLER automatisk.

Man kan returnere til hovedprogrammet på to måder:

- RETRY bevirker, at man gentager udførelsen af den fejlbehæftede linie. Dette benyttes oftest i forbindelse med indlæsning/udskrivning.
- CONTINUE bevirker, at programudførelsen forsætter med linien efter den fejlbehæftede linie.

Eksempel

```
0010 PROC fejl HANDLER
0020   IF ERR=213 THEN CONTINUE
0030 ENDPROC fejl
0040 ENABLE fejl
0050 CREATE "randomfil",10
0060 DISABLE
0070 ...
```

Bemærkninger

linie 0020 : ERR er en systemfunktion, hvis værdi er lig nummeret på den fejl, der kaldte HANDLERen. Hvis fejl 0213 (FIL EKSISTERER ALLEREDE) opstår, skal programudførslen fortsætte med linie efter den linie hvori fejlen opstod

linie 0040 : HANDLERen fejl aktiveres

linie 0050 : Hvis filen med navnet randomfil eksisterer, vil HANDLERen blive kaldt.

linie 0060 : HANDLERen deaktiveres.

Hvis man ikke ønsker at fortsætte programudførslen i forbindelse med visse fejl, kan man enten have en STOP-sætning i HANDLERen eller man kan lade programudførelsen nå frem til ENDPROC sætningen. Nås ENDPROC-sætningen, udskriver systemet normal fejlmeddelelse. Følgende HANDLER vil således skive en tekst og derefter give normal fejlmeddelelse, som hvis ingen HANDLER havde været aktiveret:

Eksempel

```
0010 PROC fejlbegået HANDLER
0020   PRINT "Du har begået en fejl"
0030 ENDPROC fejlbegået // ENDPROC nås altid
0040 ENABLE fejlbegået
0050 i:=1/0
RUN
Du har begået en fejl
AT 0050
error: 0104
```

8.7 Funktioner 8.7

Udover de indbyggede funktioner i COMAL80 kan brugeren definere sine egne funktioner.

Eksempel

```
0010 FUNC max(a,b)
0020   IF a>b THEN
0030     RETURN a
0040   ELSE
0050     RETURN b
0060   ENDIF
0070 ENDFUNC max
0080
0090 PRINT max(7,9)
0100 j:=32
0110 størst:=max(max(1,j),11)
0120 PRINT størst
0130 END

RUN
9
32
END
at 0130
```

Bemærkninger

linie 0010-0070 : Her defineres en funktion med navnet max. Funktionen har to parametre, der skal være to tal, og funktionen returnerer med værdien af det største af tallene.

linie 0030,0050 : RETURN benyttes til at tildele funktionen sin værdi, samt til at returnere fra funktionen.

linie 0090 : Funktionen kan udskrives som enhver anden numerisk variabel eller funktion

linie 0110 : FUNC kan optræde alle steder, hvor der forventes et numerisk udtryk.

De regler, der er gennemgået i det foregående for procedurer, kan direkte anvendes på funktioner, dvs. man kan have almindelige parametre, REF parametre, lukkede funktioner samt externe funktioner.

9. KOMMANDOER 9.

En række af de sætninger og kommandoer, der benyttes til at skrive programmer i COMAL80 på RC700, gennemgås i dette kapitel. Det skal dog bemærkes, at de oftest benyttede kommandoer (NEW, AUTO, RENUMBER, DEL, RUN og CON) er gennemgået i afsnit 2.2.

Samtlige systemkommandoer er:

-	AUTO	(10.5)	Automatisk linienummerering
-	CON	(10.9)	Fortsæt programudførelse
-	COPY	(10.11)	Kopier fil
-	DEL	(10.15)	Slet programlinie(r)
-	DELETE	(10.16)	Slet fil
-	DIR	(10.18)	Udskriv fortegnelse over filer
-	EDIT	(10.21)	Ret programlinie(r)
-	ENTER	(10.24)	Indlæs program
-	LIST	(10.48)	Udskriv program
-	LOAD	(10.49)	Hent program
-	MOUNT	(10.53)	Monter diskette
-	NEW	(10.54)	Slet program- og data-lager
-	RENAME	(10.70)	Omdøb fil
-	RENUMBER	(10.71)	Omnummerer linienumre i program
-	RUN	(10.77)	Udfør program
-	SAVE	(10.78)	Gem program
-	SIZE	(10.82)	Udskriv forbrugt lager

En systematisk gennemgang af kommandoerne fås i afsnit 10, og dette kapitel skal udelukkende opfattes som en kort introduktion.

9.1 Kommandoer til programindtastning 9.1

Programindtastningskommandoerne er for næsten alles vedkommende beskrevet i afsnit 2.2 (og underafsnit).

De eneste "nye" kommandoer, er kommandoer til rettelse af et program, samt en kommando til at udskrive hvor meget lager, der er brugt.

Hvis man skal rette mange linier i et program, er det ikke altid nok at LISTe programmet på skærmen, og derefter foretage rettelserne ved hjælp af cursor-pilene og vognretur-tasten. I stedet kan man skrive :

```
EDIT
```

hvilket bevirker, at den første linie vil blive udskrevet på skærmen, samt at cursoren vil være placeret umiddelbart efter linienummeret. Man er derefter i stand til at rette i linien ved hjælp af cursorpilene, RUB OUT, indsæt-tegn og slet-tegn tasterne. Når rettelserne er foretaget tastes vognretur, og den næste linie udskrives.

Hvis man kun skal rette en del af programlinierne, kan man angive startlinienummeret og slutlinienummeret for de linier, der skal rettes. Hvis man skriver:

```
EDIT 50,170
```

vil alle linierne fra 50 til 170 blive udskrevet enkeltvis, som ovenfor beskrevet.

Hvis man ikke ønsker at rette yderligere linier indenfor det angivne interval, trykkes på ESC-tasten.

Der findes tillige en kommando, der udskriver, hvor meget lager der er benyttet i maskinen. Hvis man skriver:

```
SIZE
```

kan systemet f.eks. udskrive:

```
program  data    free
00310    00000   18122
```

hvilket betyder, at det program, man har indtastet fylder 310 tegn i lageret og at der på nuværende tidspunkt er 18122 ledige tegn i lageret. Man skal her blot huske, at alle programmer skal bruge yderligere lager til data, når de udføres.

9.2 Sætninger udført som kommandoer 9.2

En række COMAL80 sætninger kan udføres samtidig med at de indtastes. Dette gøres ved udeladelse af linienummeret i programlinien.

Denne facilitet er speciel nyttig i forbindelse med

- brug af RC700 som bordkalkulator
- fejlfinding i programmer
- datastrøms indlæsning/udskrivning

Der er dog en række COMAL80 sætninger, der ikke har mening som kommandoer. Disse sætninger er:

CASE-WHEN-ENDCASE	IF(-ELSE)-ENDIF
DATA	INPUT
DISABLE	PROC-ENDPROC
ENABLE	PROC-EXTERNAL
END	REPEAT-UNTIL
FOR(-NEXT)	RETRY
FUNC-ENDFUNC	RETURN
FUNC-EXTERNAL	STOP
GLOBAL	WHILE(-ENDWHILE)
GOTO	

9.2.1 RC700 som bordkalkulator 9.2.1

Udregninger kan foretages ved hjælp af tildelings- og PRINT-kommandoerne.

Eksempel	Kommantar
NEW	Så er program- og data-lager tomt
radius:=20	variablen radius oprettes og tildeles en værdi
pi:=3.14159265359	variablen pi oprettes og tildeles en værdi
PRINT 2*pi*radius	Resultatet udskrives på skærmen.

9.2.2 Fejlfinding i programmer 9.2.2

Det, at adskillige COMAL80-sætninger kan udføres som kommandoer, gør fejlfinding i programmer lettere.

Hvis man ikke retter i et program efter en RUN, er det således muligt at kalde procedurer enkeltvis med kommandoen

EXEC <navn>

Efter hvert enkelt kald af procedurerne er det muligt at udskrive samt ændre variables værdier, hvorved fejlfindingsproceduren er gjort simpel.

9.2.3 Datastrømsindlæsning/udskrivning 9.2.3

Alle I/O-sætninger kan tillige bruges som kommandoer, dvs man kan oprette, åbne, skrive, læse og lukker filer direkte fra tastaturet.

Hvis man f.eks. får fejl 0217: IKKE ÅBEN/ALLEREDE ÅBEN fordi en fil allerede er åben, kan man ofte klare sig med kommandoen

```
CLOSE
```

samt udføre programmet på ny.

Desuden er det f.eks. muligt at få udskrevet navnene på filerne på disketteunit nr 1 med kommandoerne

```
SELECT OUTPUT "printer"
DIR 1
```

9.3 Diskettekommandoer (MOUNT og DIR) 9.3

Hvis man skifter diskette eller ønsker at benytte disketteenhed nummer 2, skal man markere dette overfor systemet. Hvis man har skiftet disketten i enhed nummer 1, gøres dette med kommandoen:

```
MOUNT 1
```

Hvis disketten i enhed nummer 2 ønskes benyttet, eller hvis disketten er skiftet, skrives

```
MOUNT 2
```

Ønsker man at udskrive en oversigt over filerne på disketteenhed nummer 1 er kommandoen

```
DIR 1
```

Hvis man ønsker udskriften for enhed nummer 2, udskiftes ettallet naturligvis blot med et 2-tal.

Begge diskettekommandoer kan desuden benyttes som programsætninger.

Eksempel

Hvis man vil udskrive indholdet på disketteenhed nummer 1 og 2 på printeren, kan det gøres med følgende program :

```
0010 SELECT OUTPUT "printer"
0020 FOR i:=1 TO 2 DO DIR i
```

9.4 Datastrømskommandoer 9.4

Datastrømskommandoerne er for næsten alles vedkommende gennemgået i afsnit 7.3 (og underafsnit).

Den eneste "nye" kommando er kommandoen til kopiering af indholdet af en fil.

Eksempel

```
PREFIX ""
COPY "1/HANOI","2/HANOI"
```

Bemærkning

Præfixet sættes til den tomme streng. Det betyder, at de angivne filnavne ikke får sat anden tekst foran. Kommandoen kopierer filen HANOI fra disketteenhed nummer 1 over i en ny fil med navnet HANOI på disketteenhed nummer 2.

De to filnavne behøver naturligvis ikke være ens.

Man kan desuden kopiere filer på samme enhed.

Eksempel

```
PREFIX "1/"
DELETE "GLDATAFIL"
COPY "NYDATAFIL","GLDATAFIL"
```

Bemærkning

Med disse kommandoer har man fået kopieret en fil, så man har en gammel version af filen, hvis den nye bliver behandlet forkert i ens program. Dermed er alle data ikke tabt.

Resten af datastrømskommandoerne er som sagt beskrevet i afsnit 7.3, så vi bringer blot et resumé af dem her:

LIST <filnavn>

udskriver et program i tekstformat på enten en ydre enhed eller en diskettefil.
Programmet kan læses ind igen med kommandoen

ENTER <filnavn>

idet man dog skal bemærke, at program- og data-lageret ikke slettes i forbindelse med ENTER (ønskes det slettet, angives NEW-kommandoen før ENTER-kommandoen).

Eksempel

```
LIST "listfil"
programmet kan senere indlæses med kommandoerne
NEW
ENTER "listfil"
```

SAVE <filnavn>

gemmer programlageret på en diskettefil. Formatet i filen svarer til maskinens interne format. Det indlæses igen med kommandoen

LOAD <filnavn>

Eksempel

```
SAVE "savefil"
programmet kan senere indlæses med kommandoen
LOAD "savefil"
```

Man bør tilstræbe at bruge SAVE/LOAD ved normal arkivering af programmer på disketten, da kommandoerne fungerer betydeligt hurtigere og filerne normalt fylder mindre end ved LIST/ENTER.

En fil fjernes med kommandoen

DELETE <filnavn>

og ønsker man at omdøbe en fil, dvs. ændre navnet men ikke indholdet, er kommandoen

RENAME <gl filnavn>,<nyt filnavn>

Eksempel

RENAME "/1/oversigt","logon"

Bemærk, at man ikke skal angive disketteenhedens nummer i det nye filnavn.

10. COMAL80 NØGLEORD 10.

Dette afsnit skal betragtes som et referenceafsnit, som beskriver alle nøgleordene i COMAL80, dvs sætninger, kommandoer, funktioner, strukturer og operatorer. Dog er de almindelige regneoperatorer (+,-,*,/,↑) og sammenligningsoperatorerne (=,<>, <,>,<=,>=) ikke beskrevet.

Beskrivelsen af nøgleordene har følgende format:

10.x COMAL80 nøgleord

Type
Format
Operator prioritet
Anvendelse
Virkemåde
Bemærkninger
Eksempler

hvor

x : afsnitsnummeret

COMAL80 nøgleord : et eller flere reserverede COMAL80 ord

Type : om nøgleordet er en sætning, kommando, funktion eller operator

Format : den formaliserede syntaks for nøgleordet. Den følger følgende regler:

- Store bogstaver skal indtastes direkte
- Bløde parenteser () skal indtastes direkte
- Krøllede parenteser {} giver valgfrihed mellem de opskrevne muligheder

- Kantede parenteser [] angiver, at det indklammede ikke skal, men kan indtastes
- Tre punktummer ... angiver, at det foregående argument kan gentages
- Symboler i < > beskrives senere i afsnittet

Operator prioritet : et tal, der angiver i hvilken rækkefølge operatorerne udføres

Anvendelse : en kort beskrivelse af nøgleordet

Virkemåde : en beskrivelse af virkemåden for mere indviklede COMAL80 strukturer eller sætninger

Bemærkninger : Bemærkninger angående brugen af nøgleordet, advarsler, fejlmeldinger o.lign.

Eksempler : Illustrering af nøgleordets virkefelt med kortere eller længere eksempler. Yderligere eksempler findes i de foregående kapitler samt i appendix F.

BEMÆRK: Ikke alle afsnit anvendes ved alle nøgleord.

10.1 ABS 10.1

COMAL80 funktion

Format

ABS (<nudtr>)

<nudtr>: et vilkårligt numerisk udtryk.

Anvendelse

Funktionen anvendes til at beregne den absolutte værdi (den numeriske værdi) af et <nudtr>. Den absolutte værdi er et positivt tal.

Eksempel

```
0010 INPUT "Indtast 2 tal  ": tal1, tal2
0020 PRINT "Forskellen mellem ";tal1;"og ";tal2;
0030 PRINT "er ";ABS(tal1-tal2)
0040 END

RUN
Indtast 2 tal :-7 16
Forskellen mellem -7 og 16 er 23
END
at 0040
```

10.2 AND 10.2

COMAL80 operator

Format

<nudtr1> AND <nudtr2>

<nudtr1>: Et vilkårligt numerisk udtryk opfattet som logisk udtryk (0: falsk, <>0: sand).

<nudtr2>: Et vilkårligt numerisk udtryk opfattet som logisk udtryk (0: falsk, <>0: sand).

operatorprioritet = 7 (se afsn 4.5.1)

Anvendelse

Den logiske operator AND er sand (<>0), hvis både <nudtr1> og <nudtr2> er sande (<>0). Hvis enten <nudtr1> eller <nudtr2> er falsk (=0), bliver resultatet også falsk (=0).

Eksempel

```
0010 DIM logisk$(2) OF 5
0020 logisk$(1):= "FALSK"; logisk$(2):= "SAND"
0030 ZONE 10
0040 PRINT "AND","FALSK","SAND"
0050 FOR i:=1 TO 30 DO PRINT "-";
0060 PRINT
0070 FOR udsagn:=FALSE TO TRUE DO
0080   PRINT logisk$(udsagn+1),
0090   PRINT logisk$((udsagn AND FALSE)+1),
0100   PRINT logisk$((udsagn AND TRUE)+1)
0110 NEXT udsagn

RUN
AND        FALSK      SAND
------------------------------
FALSK      FALSK      FALSK
SAND       FALSK      SAND
```

10.3 AT 10.3

COMAL80 funktion

Format

AT (<nudtr1>,<nudtr2>)

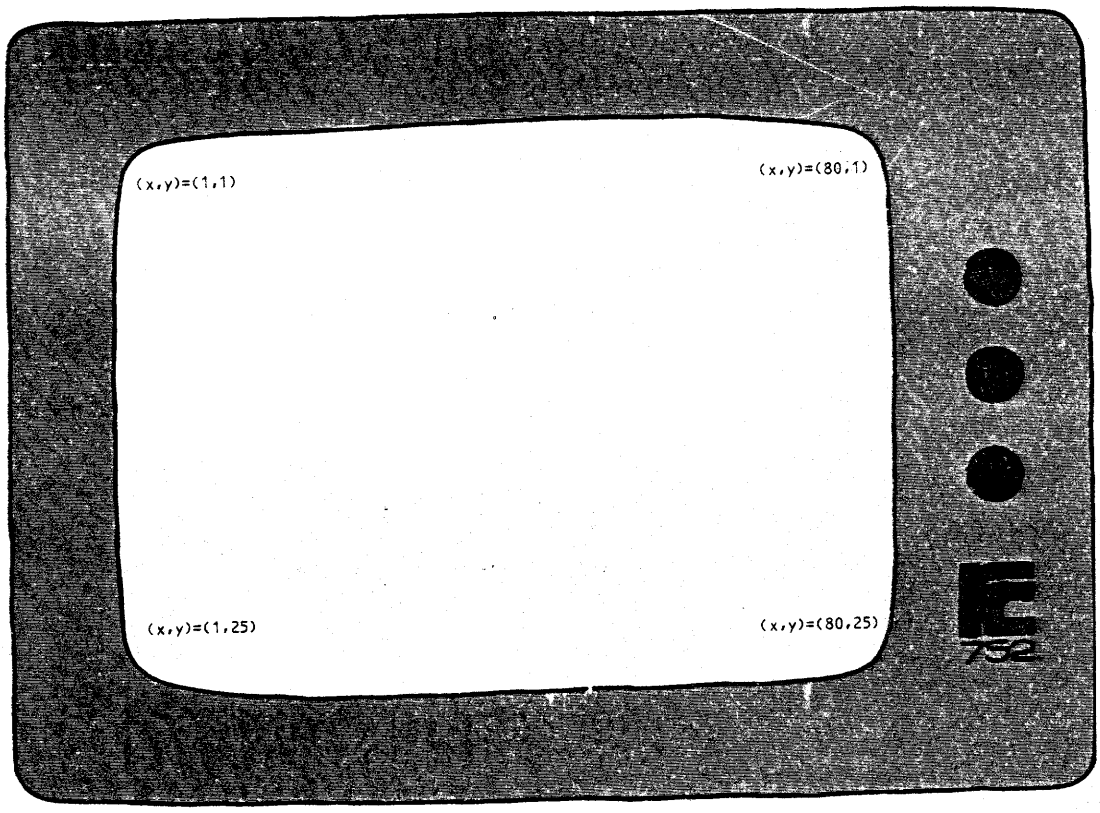
<nudtr1>: et numerisk udtryk med værdi i intervallet
1≤<nudtr1>≤80
<nudtr2>: et numerisk udtryk med værdi i intervallet
1≤<nudtr2>≤25

Anvendelse

Funktionen benyttes i PRINT- eller INPUT-sætninger til at flytte udskriften til en bestemt position på skærmen.

Bemærkninger

1. Skærmen adresseres på følgende måde med AT(x,y):



2. AT kan kun benyttes i forbindelse med udskrift på skærmen.

Eksempel 1

```
0010 PRINT CHR$(12) // slet skærmen
0020 INPUT AT(30,12),"Kassestørrelse :": kasse
0030 IF kasse>10 THEN kasse:=10
0040 IF kasse<1 THEN kasse:=1
0050 MARGIN 0
0060 FOR xkoord:=1 TO 80 DO
0070   PRINT AT(xkoord,12-kasse);"*";
0080   PRINT AT(xkoord,12+kasse);"*";
0090 NEXT xkoord
0100 FOR ykoord:=12-kasse TO 12+kasse DO
0110   PRINT AT(1,ykoord);"*";AT(80,ykoord);"*";
0120 NEXT ykoord
```

Eksempel 2

```
0010 PRINT CHR$(12);CHR$(132)// slet skærm og gå i grafisk mode
0020 PRINT AT(5,2);"COSINUS OG"// bemærk store bogstaver
0030 PRINT AT(5,3);"SINUS FUNKTIONEN"
0040 FOR j:=2 TO 80 DO PRINT AT(j,11);CHR$(200);// vandret streg
0050 FOR i:=1 TO 21 DO PRINT AT(40,i);CHR$(201);// lodret streg
0060 FOR rd:=-3.141 TO 3.141 STEP 0.1 DO
0070   PRINT AT(rd*10+40,-SIN(rd)*10+11);"S"
0080   PRINT AT(rd*10+40,-COS(rd)*10+11);"C"
0090 NEXT rd
0100 PRINT AT(1,22);CHR$(128);// tilbage til alm. mode
0110 END
```

COMAL80 funktion

Format

ATN(<nudtr>)

<nudtr>: Et vilkårligt numerisk udtryk

Anvendelse

Funktionen udregner den vinkel (udtrykt i radianer) hvis tangens er lig <nudtr>, dvs. at funktionen er den omvendte funktion til tangens (TAN).

Eksempel

```
0010 // Ud fra ATN kan de øvrige arcus funktioner udregnes
0020 FUNC asin(x)// Arcus sinus
0030   IF ABS(x)>=1 THEN
0040     IF ABS(x)=1 THEN RETURN SGN(x)*ATN(1)*2// ATN(1): pi/4
0050   ELSE
0060     RETURN ATN(x/SQR(1-x*x))
0070   ENDIF
0080 ENDFUNC asin
0090
0100 FUNC acos(x)// Arcus cosinus
0110   IF x=0 THEN
0120     RETURN ATN(1)*2
0130   ELSE
0140     IF ABS(x)<=1 THEN RETURN ATN(SQR(1-x*x)/x)+4*ATN(1)*(x<0)
0150   ENDIF
0160 ENDFUNC acos
0170
0180 FUNC acot(x)// Arcus catangens
0190   IF x<>0 THEN
0200     RETURN ATN(1/x)+4*ATN(1)*(x<0)
0210   ENDIF
0220 ENDFUNC acot
```

```
0230
0240 ZONE 20
0250 PRINT "i","asin(i)","acos(i)"
0260 FOR i:=0 TO 1 STEP 0.1 DO PRINT i,asin(i),acos(i)
0270 PRINT
0280 PRINT "i","acot(i)","atan(i)"
0290 FOR i:=0.1 TO 0.9 STEP 0.1 DO PRINT i,acot(i),ATN(i)
0300 END
```

i	asin(i)	acos(i)
0	0	1.570796326794
0.1	0.1001674211615	1.470628905634
0.2	0.2013579207902	1.369438406005
0.3	0.3046926540152	1.26610367278
0.4	0.4115168460677	1.159279480726
0.5	0.5235987755979	1.047197551196
0.6	0.6435011087933	0.9272952180014
0.7	0.7753974966105	0.7953988301838
0.8	0.9272952180014	0.6435011087927
0.9	1.119769514998	0.4510268117957
1	1.570796326794	0

i	acot(i)	atan(i)
0.1	1.471127674304	9.966865249115E-002
0.2	1.373400766946	0.1973955598499
0.3	1.279339532318	0.2914567944779
0.4	1.190289949683	0.380506377112
0.5	1.107148717794	0.4636476090008
0.6	1.030376826524	0.5404195002706
0.7	0.9600703624053	0.6107259643893
0.8	0.8960553845712	0.6747409422236
0.9	0.8379812250082	0.7328151017866

10.5 AUTO 10.5

COMAL80 kommando

Format

```
                <lnr> [,]
AUTO [ { , <lnr spring>      } ]
       <lnr>,<lnr spring>
```

<lnr>: første linienummer der skal indtastes
<lnr spring>: forskellen mellem linienumrene under indtastningen

Anvendelse

Kommandoen anvendes til automatisk linienummerering af et program, der er ved at blive indtastet.

Bemærkninger

1. Alle programlinier skal indledes med et linienummer. Hvis flere linier skal indtastes, simplificerer AUTO opgaven.

2. Man kommer ud af AUTO-tilstanden ved at trykke ESC på tastaturet

3. Hvis der optræder fejl under indtastningen, angives fejlen i skærmens øverste højre hjørne, og cursoren forbliver på linien.

4. Kombinationerne af parametrene til kommandoen AUTO, har følgende betydning:

AUTO	Linienummereringen starter ved linie 0010 og fortsætter med spring på 0010
AUTO <lnr>	Linienummereringen starter ved linie <lnr> og fortsætter med spring på <lnr>

AUTO <lnr>,	Linienummereringen starter ved linie <lnr> og fortsætter med spring på 0010
AUTO ,<lnr spring>	Linienummereringen starter ved linie 0010 og fortsætter med spring på <lnr spring>
AUTO <lnr>,<lnr spring>	Linienummereringen starter ved linie <lnr> og fortsætter med spring på <lnr spring>

Eksempel	Kommentar
AUTO	0010 udskrives på skærmen. Når linien er indtastet udskrives 0020, derefter 0030, 0040, osv.
AUTO 50	Linienummereringen bliver 0050, 0100, 0150, 0200 osv.
AUTO 50,	Linienummereringen bliver 0050, 0060, 0070, 0080 osv.
AUTO ,100	Linienummereringen bliver 0010, 0110, 0210, 0310 osv.
AUTO 50,100	Linienummereringen bliver 0050, 0150, 0250, 0350 osv.

10.6 CASE-WHEN-ENDCASE 10.6

COMAL80 struktur

Format

```
CASE <udtryk0> OF
WHEN <udtryk1a> [,<udtryk1b>]...
  <sætningsliste -1>

[ WHEN <udtrykna> [,<udtryknb>]... ]
[   <sætningsliste -n>             ]
...

[ OTHERWISE                        ]
[   <sætningsliste -0>             ]

ENDCASE
```

<udtryk0>: nøgleudtrykket,enten et numerisk udtryk eller et strengudtryk
De øvrige <udtryk..> skal være af samme type som <udtryk0>
<sætningsliste>: COMAL80 sætninger

Anvendelse

Konstruktionen benyttes til at få udført en ud af flere sætningsblokke afhængig af værdien af et udtryk.

Virkemåde

1. Værdien af <udtryk> i CASE-OF sætningen udregnes.

2. Værdien af udtrykkene i WHEN <udtryk> [,<udtryk>] sætningerne udregnes en for en, indtil der findes en værdi lig værdien i trin 1. Hvis sammenfaldet optræder i den i'te WHEN-sætning, vil <sætningsliste -i> blive udført.

3. Når <sætningsliste -i> er udført, og hvis <sætningsliste -i> ikke foretager et hop ud af konstruktionen, hoppes der til første sætning efter ENDCASE.

4. Hvis der ikke udregnes et WHEN <udtryk> der svarer til udtrykket i trin 1 udføres <sætningsliste -0> (sætningslisten efter OTHERWISE). Hvis OTHERWISE ikke er angivet, udskrives fejl 0115: ULOVLIG CASE VÆRDI.

Bemærkning

1. Konstruktionen kan ikke udføres som kommando.

Eksempel 1

```
0010 // eksempel på brug af case
0020 FOR i:= 1 TO 5 DO
0030   CASE i OF
0040   WHEN 1,3
0050     PRINT i;" (1 eller 3)"
0060   WHEN 5
0070     PRINT i;" (5)"
0080   OTHERWISE
0090     PRINT i
0100   ENDCASE
0110 NEXT i
0120 END

RUN
1  (1 eller 3)
2
3  (1 eller 3)
4
5  (5)
END
at 0120
```

Eksempel 2

```
0010 DIM måned$ OF 3
0020 WHILE TRUE DO
0030   INPUT "Månednavn (3 tegn er nok) : ": måned$
0040   CASE måned$ OF
0050   WHEN "jan", "mar", "maj", "jul", "aug", "okt", "dec"
0060     PRINT måned$;" har 31 dage."
0070   WHEN "apr", "jun", "sep", "nov"
0080     PRINT måned$;" har 30 dage."
0090   WHEN "feb"
0100     PRINT måned$;" har for det meste 28 dage."
0110   OTHERWISE
0120     PRINT "Denne måned eksisterer ikke"
0130   ENDCASE
0140 ENDWHILE
0150 END

RUN
Månednavn (3 tegn er nok) : april
apr har 30 dage.
Månednavn (3 tegn er nok) : nøv
Denne måned eksisterer ikke
Månednavn (3 tegn er nok) : <ESC>
STOP
at 0030
```

10.7 CHAIN 10.7

COMAL80 sætning

Format

CHAIN <filnavn>

<filnavn>: strengudtryk, der angiver et filnavn.

Anvendelse

Sætningen anvendes til at afslutte et program samt automatisk LOADe og udføre et nyt program, der er SAVE'd i en diskettefil.

Bemærkninger

1. Datalageret slettes når CHAIN-sætningen udføres

2. Programudførelsen i det angivne program (<filnavn>) starter i programlinien med det laveste linienummer

3. Programnavnet angivet med <filnavn> skal være gemt med SAVE.

4. Når man har CHAIN'et til et nyt program, vil det nye <filnavn> blive angivet i statuslinien, når programmet stopper.

Eksempel	Kommentar
0010 // Program a	
0020 PRINT "Program a"	
0030 END	
SAVE "programa"	Nu er programmet gemt under navnet programa.
NEW	Slet program- og datalageret.
0010 // Program b	
0020 PRINT "Program b"	
0030 CHAIN "programa"	
SAVE "programb"	Nyt program, der kalder programa.
RUN	
Program b	
Program a	
END	Bemærk, at filnavnet i status-
at 0030	linien er ændret til programa.

10.8 CHR$ 10.8

COMAL80 funktion

Format

CHR$(<nudtr>)

<nudtr>: vilkårligt numerisk udtryk mellem 0 og 255 (incl).

Anvendelse

Funktionen returnerer med det tegn, som svarer til ASCII-værdien af <nudtr>.

Bemærkninger

1. Sammenhængen mellem <nudtr> og tegnene er vist i appendix E.

2. Funktionen må anvendes i et vilkårligt strengudtryk.

3. Hvordan skærmen reagerer på forskellige CHR-koder fremgår af appendix C.

Eksempel

```
0010 // Programmet udskriver det normale tegnsæt på skærmen
0020
0030 MARGIN 64
0040 ZONE 8
0050 PRINT CHR$(12)// Blanker skærmen
0060 FOR ch:= 32 TO 127 DO PRINT ch;CHR$(ch),
0070 PRINT
0080 PRINT CHR$(7)// Bib
0090 END

RUN
```

```
32      33 !    34 "    35 #    36 $    37 %    38 &    39 '
40 (    41 )    42 *    43 +    44 ,    45 -    46 .    47 /
48 0    49 1    50 2    51 3    52 4    53 5    54 6    55 7
56 8    57 9    58 :    59 ;    60 <    61 =    62 >    63 ?
64 ü    65 A    66 B    67 C    68 D    59 E    70 F    71 G
72 H    74 I    74 J    75 K    76 L    77 M    78 N    79 O
80 P    81 Q    82 R    83 S    84 T    85 U    86 V    87 W
88 X    89 Y    90 Z    91 Æ    92 Ø    93 Å    94 @    95 _
96 ä    97 a    98 b    99 c    100 d   101 e   102 f   103 g
104 h   105 i   106 j   107 k   108 l   109 m   119 n   111 o
112 p   113 q   114 r   115 s   116 t   117 u   118 v   119 w
120 x   121 y   122 z   123 æ   124 ø   125 å   126 ö   127

END
at 0090
```

10.9 CLOSE 10.9

COMAL80 sætning/kommando

Format

CLOSE [[FILE] <strømnr>]

<strømnr>: et numerisk udtryk, hvis værdi angiver nummeret på en datastrøm.

Anvendelse

Sætningen/kommandoen bruges til at lukke en eller alle datastrømme.

Bemærkninger

1. Anføres CLOSE uden strømnummer, lukkes alle åbne strømme i programmet.

2. CLOSE benyttes til at lukke en datastrøm, så den eventuelt kan åbnes igen i en anden mode.

Eksempel

```
0010 OPEN FILE 1,"datafil",WRITE
0020 RANDOMIZE
0030 FOR i:=1 TO 10 DO WRITE FILE 1: RND
0040 CLOSE FILE 1 // Luk efter brug
0050 END
```

COMAL80 kommando

Format

CON

Anvendelse

CONtinue-kommandoen benyttes til at genstarte udførelsen af et program, der er stoppet af programsætningerne STOP, END, ved ESCape eller ved fejl.

Bemærkninger

1. Programudførelsen genoptages med sætningen umiddelbart efter den sætning, hvor programmet standsede.

2. Der må ikke slettes, indføjes eller rettes linier før CON.

3. Hvis man retter i et program og derefter skriver CON, fås fejludskriften 0095: CON IKKE TILLADT.

10.11 CONTINUE 10.11

COMAL80 sætning

Format

CONTINUE

Anvendelse

Bruges kun i en PROC-HANDLER. Sætningen bevirker, at programudførelsen fortsætter med sætningen efter den sætning, der forårsagede kaldet af PROC-HANDLER.

Bemærkning

1. Sætningen kan ikke udføres som kommando.

Eksempel

```
0010 PROC nuldiv HANDLER
0020   IF ERR=104 THEN
0030     PRINT "*** Division med nul"
0040     CONTINUE // Fortsæt hovedprogram
0050   ENDIF
0060 ENDPROC nuldiv
0070
0080 ENABLE nuldiv
0090 i := 1/0 // Fremprovoker fejl
0100 PRINT "Slut"
0110 END

RUN

*** Division med nul
Slut
END
at 0110
```

10.12 COPY 10.12

COMAL80 kommando

Format

COPY <filnavn1>, <filnavn2>

<filnavn1>: navnet på en datastrøm, der skal kopieres fra, navnet angives i anførselstegn

<filnavn2>: navnet på en datastrøm, der skal kopieres over i, navnet angives i anførselstegn.

Anvendelse

Kommandoen kopierer en datastrøm over i en anden.

Bemærkninger

1. Kommandoen kan bruges til at flytte filer mellem 2 disketteenheder, eller fra en diskettefil til en ydre enhed (skærm, printer).

2. Hvis <filnavn2> angiver en diskettefil må den ikke eksistere på forhånd, ellers udskrives fejl 0213: FIL EKSISTERER ALLEREDE.

Eksempel

Med denne kommando, kan man udskrive en fil, der er gemt med kommandoen LIST eller en datafil, der er skrevet med PRINT FILE, på printeren:

```
COPY "eksempela", "printer"
```

10.13 COS 10.13

COMAL80 funktion

Format

COS (<nudtr>)

<nudtr>: et numerisk udtryk, der angiver et radianantal

Anvendelse

Funktionen udregner cosinus til en vinkel udtrykt i radianer.

Eksempel

```
0010 PRINT "Programmet omformer grader til radianer ";
0020 PRINT "og udregner cosinus"
0030 PRINT
0040 pi:= ATN(1)*4
0050 INPUT "Indtast en vinkel (i grader) :": vinkel
0060 radianer:= vinkel*pi/180
0070 PRINT vinkel;"grader svarer til ";radianer;" Radianer."
0080 PRINT "COS( ";vinkel;") = ";COS(radianer)
0090 END

RUN
Programmet omformer grader til radianer og udregner cosinus

Indtast en vinkel (i grader) :60
60 grader svarer til 1.047197551196 Radianer.
COS( 60 ) = 0.5000000000007

END
at 0090
```

10.14 CREATE 10.14

COMAL80 sætning/kommando

Format

CREATE <filnavn>, <længde>

<filnavn>: navnet på en diskette fil, der skal dannes.
<længde>: vilkårligt positivt numerisk udtryk, begrænset af diskettens kapacitet.

Anvendelse

Sætningen/kommandoen benyttes til at oprette en fil med direkte tilgang på en diskette. <længde> angiver størrelsen af filen i 1024 tegns blokke.

Bemærkninger

1. Sætningen benyttes kun til at reservere plads til filer med direkte tilgang, da sekventielle filer automatisk oprettes og udvides

2. Forsøger man at oprette en fil, der allerede eksisterer, fås fejl 0213: FIL EKSISTERER ALLEREDE.

3. Er disketten skrivebeskyttet fås fejl 0209: DISKETTE SKRIVEBESKYTTET.

4. Har man skiftet diskette uden at udføre MOUNT-sætningen fås fejl 208: DISKETTE ER BLEVET SKIFTET.

5. Er der ikke plads til flere filer på disketten fås fejl 215: DISKETTE FULD.

Eksempel

```
0010 PRINT "Filopretter"
0020 PRINT
0030 DIM navn$ OF 20
0040 INPUT "Filnavn       :": navn$
0050 INPUT "Poststørrelse :": poststørrelse
0060 INPUT "Antal poster  :": antposter
0070 længde:=poststørrelse*antposter/1024 + 1
0080 CREATE navn$,længde
0090 PRINT "Filen er oprettet og fylder ";længde;"k bytes"
0100 END

RUN
Filopretter

Filnavn       :register
Poststørrelse :40
Antal poster  :100
Filen er oprettet og fylder 4 k bytes
END
at 0100
```

10.15 DATA 10.15

COMAL80 sætning

Format

DATA { <n konst> | <s konst> } [, { <n konst> | <s konst> }]...

<n konst>: numerisk konstant
<s konst>: streng konstant

Anvendelse

Sætningen bruges til at opbevare værdier i selve programmet som så kan læses ved hjælp af READ-sætningen.

Bemærkninger

1. Sætningen kan ikke udføres som kommando.

2. Ved starten af et programs udførsel (RUN) gennemgås programmet sekventielt, alle DATA-sætningerne kædes sammen til en dataliste og en datapegepind sættes til at pege på det første element i den første DATA-sætning.

3. Både tal- og streng-konstanter må optræde i samme DATA-sætning.

Eksempel

```
0010 DIM spørg$ OF 80, svar$ OF 80, korr$ OF 80
0020 EXEC overskrift
0030 WHILE NOT EOD DO
0040   READ spørg$,korr$
0050   antalsvar:=0
0060   REPEAT
0070     PRINT spørg$
0080     INPUT " ? ":svar$;
0090     IF svar$=korr$ THEN
```

```
0100        PRINT TAB(40);"** Korrekt **"
0110      ELSE
0120        PRINT TAB(40);">>>> Forkert !"
0130      ENDIF
0140      antalsvar:=antalsvar+1
0150    UNTIL antalsvar>=3 OR svar$=korr$
0160    IF svar$<>korr$ THEN
0170      PRINT "Det korrekte svar er ";korr$
0180    ENDIF
0190  ENDWHILE
0200  PROC overskrift CLOSED
0210    DIM txt$ OF 80
0220    // En lukket procedure har en lokal DATA-liste
0230    WHILE NOT EOD DO
0240      READ txt$
0250      PRINT txt$
0260    ENDWHILE
0270    DATA "Oversættelsesøvelse",""
0280    DATA "Oversæt følgende ord til dansk"
0290  ENDPROC overskrift
0300  DATA "table","bord","dog","hund","duck","and"

RUN
Oversættelsesøvelse

Oversæt følgende ord til dansk
table ? bord                          ** Korrekt **
dog ? and                             >>>> Forkert !
dog ? høne                            >>>> Forkert !
dog ? ælling                          >>>> Forkert !
Det korrekte svar er hund
duck ? and                            ** Korrekt **
END
at 0300
```

10.16 DEL 10.16

COMAL80 kommando

Format

```
      <lnr1> [,]
DEL { , <lnr2>       }
      <lnr1>, <lnr2>
```

<lnr1>: linienummeret på første sætning der skal slettes
<lnr2>: linienummeret på sidste sætning der skal slettes.

Anvendelse

Kommandoen anvendes til at slette en eller flere programlinier i det program, der ligger i programlageret.

Bemærkninger

1. Kombinationerne af parametrene til DEL-kommanoden har følgende betydning:

DEL <lnr1>	sletter linien med linienummer <lnr1>
DEL <lnr1>,	sletter alle linierne hvor linienummeret >= <lnr1>.
DEL ,<lnr2>	sletter alle linierne hvor linienummeret <= <lnr2>
DEL <lnr1>,<lnr2>	sletter alle linierne hvor <lnr1> <= linienummeret <= <lnr2>.

Eksempel

Hvis vi i det følgende forudsætter, at programlageret indeholder følgende program:

```
0010 PRINT
0020 PRINT
0030 PRINT
0040 PRINT
0050 PRINT
0060 PRINT
```

Da vil:

```
del 20     slette linie 0020
del 40,    slette linierne 0040, 0050, 0060
del ,20    slette linierne 0010, 0020
del 30,40  slette linierne 0030, 0040
```

10.17 DELETE 10.17

COMAL80 sætning/kommando

Format

DELETE <filnavn>

<filnavn>: Navnet på en diskettefil, der ønskes slettet.

Anvendelse

Sætningen/kommandoen sletter filen <filnavn> fra diskettens katalog.

Bemærkninger

1. Selv om filen ikke eksisterer, udskrives der ikke fejlmeddelelse.

2. Er disketten skrivebeskyttet udskrives fejlen 0209: DISKETTE SKRIVEBESKYTTET.

Eksempel

```
0010 // Programmet sletter de angivne filnavne
0020 PROC fejl HANDLER
0030   PRINT "*** ";filnavn$;" er ikke slettet"
0040   CASE SYS(0) OF
0050   WHEN 209
0060     PRINT "*** Diskette skrivebeskyttet"
0070     CONTINUE
0080   WHEN 214
0090     PRINT "*** Fil eksisterer ikke"
0100     CONTINUE
0110   WHEN 212
0120     PRINT "*** Fejl i filnavn"
0130     CONTINUE
0140   WHEN 222
0150     PRINT "*** Fil skrivebeskyttet"
0160     CONTINUE
0170   OTHERWISE
```

```
0180      // ingen speciel aktion
0190    ENDCASE
0200 ENDPROC fejl
0210
0220 DIM unit$ OF 1,navn$ OF 11,filnavn$ OF 14
0230 ENABLE fejl
0240 PRINT "Programmet sletter filer på disketter"
0250 REPEAT
0260    PRINT
0270    INPUT "Unitnr : ": unit$
0280    INPUT "Filnavn :": navn$
0290    filnavn$:= "/"+unit$+"/"+navn$
0300    DELETE filnavn$
0310 UNTIL FALSE
```

10.18 DIM

10.18

COMAL80 sætning/kommando

Format

```
     { <svar> OF <nudtr>                 }   [    { <svar> OF <nudtr>                 } ]
DIM  { <svar>$(<nudtr1>) OF <nudtr2>     }   [ ,  { <svar>$(<nudtr1>) OF <nudtr2>     } ] ...
     { <vektor>(<nudtr>)                 }   [    { <vektor>(<nudtr>)                 } ]
     { <matrix>(<nudtr1>,<nudtr2>)       }   [    { <matrix>(<nudtr1>,<nudtr2>)       } ]
```

<nudtr>,<nudtr1>,<nudtr2>: positive numeriske udtryk
<svar>: navnet på en strengvariabel eller en teksttabel
<vektor>: navnet på et éndimensionalt array
<matrix>: navnet på et todimensionalt array

Anvendelse

Sætningen/kommandoen anvendes til at erklære reelle talsæt (vektorer og matricer), strenge og teksttabeller.

Bemærkninger

1. Det er ikke tilladt at redimensionere en allerede dimensioneret variable.

2. En DIM-sætning må gerne optræde lokalt i en lukket (CLOSED) procedure eller funktion. Hver gang proceduren eller funktionen kaldes, oprettes variablen pàny, og når man returnerer fra proceduren eller funktionen fjernes den igen.

3. Efter at DIM er udført, har de reelle elementer værdien 0 og strenge længden 0.

4. Den nedre grænse for index i et array er altid 1.

5. Opbygning:

DIM V(5) Dimensionerer et én-dimensionalt array (vektor) med 5 tal:
V(1) V(2) V(3) V(4) V(5)

DIM M(4,3) Dimensionerer et to-dimensionalt array (matrix) med 4 rækker og 3 søjler:
M(1,1) M(1,2) M(1,3)
M(2,1) M(2,2) M(2,3)
M(3,1) M(3,2) M(3,3)
M(4,1) M(4,2) M(4,3)

DIM S$ OF 15 Dimensionerer en strengvariabel, der kan indeholde op til 15 tegn.

DIM T$(4) OF 30 Dimensionerer en teksttabel, der kan indeholde op til 4 strenge, hver på ind til 30 tegn:
T$(1) T$(2) T$(3) T$(4)

COMAL80 sætning/kommando

Format

DIR [<nudtr>]

<nudtr>: numerisk udtryk med værdi 1 eller 2

Anvendelse

Sætningen/kommandoen udskriver en liste over filerne på diskette nummer <nudtr>

Bemærkning

1. Normalt udskrives listen på skærmen. Dette kan dog ændres med SELECT OUTPUT sætningen.

2. Angives <nudtr> ikke, forventes at PREFIX er sat til enten 1/ eller 2/, og DIR vil da udskrive filerne på prefix-enheden

3. Oversigten over filnavnene afsluttes med en angivelse af den resterende plads på disketten.

Eksempel 1	Kommentar
SELECT OUTPUT "printer"	Udskrift af navnene på diskette nr
DIR 1	1's filer på printeren

Eksempel 2

```
0010 // programmet udskriver filerne på en diskette
0020 // i alfabetisk orden
0030
0040 PROC sort(venstre, højre) CLOSED
0050   GLOBAL x$,k$,a$
0060   // proceduren sorterer elementer i rækken k$(venstre)...k$(højre)
0070   i:= venstre; j:= højre
0080   x$:= k$((i+j) DIV 2)
0090   REPEAT
0100     WHILE k$(i)<x$ DO i:=i+1
0110     WHILE x$<k$(j) DO j:=j-1
```

```
0120      IF i<=j THEN
0130        a$:=k$(i); k$(i):=k$(j); k$(j):=a$
0140        i:=i+1; j:=j-1
0150      ENDIF
0160    UNTIL i>j
0170    IF venstre<j THEN EXEC sort(venstre,j)
0180    IF i<højre THEN EXEC sort(i,højre)
0190 ENDPROC sort
0200
0210 INPUT "Listning af hvilken unit ?": unit
0220 MARGIN 16
0230 ZONE 16
0240 SELECT OUTPUT "kat$$$"
0250 DIR unit // Udskriv kataloget i filen kat$$$
0260 SELECT OUTPUT "console"
0270
0280 DIM k$(100) OF 11,x$ OF 11,a$ OF 11
0290 OPEN 1,"kat$$$", READ
0300 n:= 1
0310 WHILE NOT EOF(1) DO
0320   INPUT FILE 1: k$(n)
0330   IF k$(n)<>"kat$$$      " AND NOT EOF(1) THEN n:=n+1
0340 ENDWHILE
0350 n:=n-3
0360 PRINT "Unit nr ";unit
0370 CLOSE 1
0380 DELETE "kat$$$"// fjernes efter brug
0390
0400 // nu ligger filnavne i k$(1)...k$(n) (usorteret)
0410
0420 EXEC sort(1,n)// sorter elementerne
0430 MARGIN 80
0440 FOR i:= 1 TO n DO PRINT k$(i),
0450 PRINT
0460 END
```

10.20 DISABLE 10.20

COMAL80 sætning

Format

DISABLE

Anvendelse

Sætningen gør en eventuel ENABLE'd PROC-HANDLER inaktiv og lader systemet selv håndtere kommende fejl.

Bemærkning

1. Systemet udfører automatisk en DISABLE, hvis en ny PROC-HANDLER ENABLE's.

Eksempel

```
0010 PROC skrivnr HANDLER
0020   PRINT "*** Fejl nr :";ERR
0030   CONTINUE
0040 ENDPROC skrivnr
0050
0060 ENABLE skrivnr
0070 i:= SQR(-7) // Fremprovoker fejl
0080 DISABLE // Normal fejlbehandling
0090 i:= LOG(0)
0100 END

RUN

*** Fejl nr :103
AT 0090
err : 0106
```

10.21 DIV 10.21

COMAL80 operator

Format

<nudtr1> DIV <nudtr2>

<nudtr1>: vilkårligt numerisk udtryk

<nudtr2>: vilkårligt numerisk udtryk forskellig fra nul.

Operator prioritet = 3

Anvendelse

Operatoren foretager en heltalsdivision

Bemærkning

1. Definitionen på a DIV b er:
   SGN(a/b)*INT(INT(ABS(a+0.5))/INT(ABS(b+0.5)))

2. <nudtr1> og <nudtr2> må ikke være større end 32767 eller mindre end -32768

Eksempel

```
0010 FUNC divi(a,b)
0020   RETURN SGN(a/b)*INT(INT(ABS(a+0.5))/INT(ABS(b+0.5)))
0030 ENDFUNC divi
0040
0050 RANDOMIZE
0060 ZONE 20
0070 PRINT "A","B","A DIV B","divi(A,B)"
0080 FOR i:=1 TO 10 DO
0090   REPEAT
0100     tal1:=RND*1000-500; tal2:=RND*1000-500
0110   UNTIL INT(tal2)<>0
0120   PRINT tal1,tal2,tal1 DIV tal2,divi(tal1,tal2)
0130 NEXT i
0140 END
```

10.22 EDIT 10.22

COMAL80 kommando

Format

```
        <lnr1> [,]
EDIT { , <lnr2>       }
        <lnr1>, <lnr2>
```

<lnr1>: linienummeret på den første sætning, hvori der skal rettes.
<lnr2>: linienummeret på den sidste sætning, hvori der skal rettes.

Anvendelse

Kommandoen anvendes til at rette i hele eller dele af det program, der ligger i programlageret.

Bemærkninger

1. Systemet udskriver den første sætning i området og cursoren fremkommer lige efter linienummeret. Brugeren kan nu rette den pågældende sætning ved hjælp af systemets editeringsfaciliteter (se afsnit 2). Når de ønskede rettelser er foretaget tastes vognretur, sætningen syntaks analyseres og lagres. Derefter udskrives den næste sætninger, der så kan rettes osv.

2. Kombinationerne af parametrene til EDIT-kommandoen har følgende betydning:

EDIT	Giver brugeren mulighed for at rette i hele programmet i programlageret.
EDIT <lnr1>	Giver brugeren mulighed for at rette linien med linienummeret = <lnr1>

EDIT <lnr1>, Giver brugeren mulighed for at rette de linier, hvor linienummeret >= <lnr1>

EDIT ,<lnr2> Giver brugeren mulighed for at rette de linier, hvor linienummeret <= <lnr2>

EDIT <lnr1>, <lnr2> Giver brugeren mulighed for at rette de linier, hvor <lnr1> <= linienummeret <= <lnr2>

3. En alternativ måde at rette i et program er først at udskrive dele af det med kommandoen LIST og derefter benytte tastaturets 4 pile samt øvrige editeringsfaciliteter til at rette i programmet med. Men husk at hver rettet linie først bliver lagret når man har trykket vognretur.

10.23 ENABLE 10.23

COMAL80 sætning

Format

ENABLE <navn>

<navn>: Navnet på en PROC-HANDLER.

Anvendelse

Sætningen aktiverer en PROC-HANDLER.

Bemærkninger

1. Sætningen kan ikke udføres som kommando.

2. En PROC-HANDLER gøres inaktiv ved hjælp af DISABLE-sætningen.

Eksempel

```
0010 PROC fejlproc HANDLER
0020   IF ERR>=200 THEN
0030     PRINT "*** I/O-Fejl NR ";SYS(0)
0040     PRINT "*** Strømnr ";SYS(1)
0050   ELSE
0060     PRINT "*** Fejl nr ";SYS(0)
0070   ENDIF
0080   REPEAT
0090     INPUT "*** Funktion: F(ortsæt, G(entag, S(lut :": O$
0100     IF O$="F" THEN CONTINUE
0110     IF O$="G" THEN RETRY
0120   UNTIL O$="S"
0130 ENDPROC fejlproc
0140
0150 DIM O$ OF 1
0160 ENABLE fejlprcc
...
```

10.24 END 10.24

COMAL80 sætning

Format

END

Anvendelse

Benyttes til at markere afslutningen af et program.

Bemærkninger

1. Sætningen kan ikke udføres som kommando.

2. COMAL80 kræver ikke, at programmet indeholder en END sætning, da programudførslen standser, når programmets sidste sætning (med det højeste linienummer) er udført.

3. Programmet må gerne indeholde flere END-sætninger. Når en af disse mødes udskrives
   END
   at XXXX
   hvor XXXX er linienummeret på END-sætningen.

Eksempel

```
...
0090 print "Slut på program"
0100 end
run
Slut på program
END
at 0100
```

10.25 ENTER 10.25

COMAL80 kommando

Format

ENTER <filnavn>

<filnavn>: navnet på en diskettefil angivet i anførselstegn.

Anvendelse

Kommandoen bruges til at sammenflette programlinierne i en diskettefil med programlinierne i programlageret.

Bemærkninger

1. Hvis programlagerets program og diskettefilens program har fælles linienumre, vil programlagerets linie blive slettet og diskettefilens linie blive indsat.

2. De linienumre, der eksisterer i programlageret, men ikke i diskettefilen, vil ikke blive slettet ved ENTER.

3. Hvis programmet i maskinens lager ikke ønskes benyttet, bør kommandoen NEW gives før ENTER kommandoen.

4. For at et program kan læses ind ved hjælp af ENTER, skal det være gemt ved hjælp af LIST.

5. Systemet udskriver linierne efterhånden som de indlæses.

Eksempel	Kommentar
20 // program a	
30 a:=7	
list "eksempela"	Nu er programmet gemt under navnet "eksempela"
new	Sletter programlageret
10 // program b	Nye programlinier indtastes
30 B:=3	
list	
0010 // program b	og listes på skærmen
0030 b:=3	
enter "eksempela"	"eksempela" indlæses oveni program
0020 // program a	b, og systemet lister de linier
0030 a:=7	man indlæser
list	
0010 // program b	Slutresultatet: en blanding af
0020 // program a	program a og program b, hvor
0030 a:=7	linie 0030 er overskrevet.

COMAL80 funktion

Format

EOD

Anvendelse

EOD er en logisk funktion uden argumenter, som har værdien SAND (<>0) når det sidste element i datalisten er læst, ellers har den værdien FALSK (=0)

Eksempel

```
0010 ZONE 10
0020 PRINT "EOD = ";EOD
0030 WHILE NOT EOD DO
0040   READ tal
0050   PRINT "EOD = ";EOD, "TAL = ";tal
0060 ENDWHILE
0070 DATA 7, 9, 13
0080 END

RUN
EOD = 0
EOD = 0  TAL = 7
EOD = 0  TAL = 9
EOD = 1  TAL = 13
END
at 0080
```

10.27 EOF 10.27

COMAL80 funktion

Format

EOF(<strømnr>)

<strømnr>: et udtryk, hvis værdi er lig nummeret på en datastrøm.

Anvendelse

EOF er en logisk funktion, som antager værdien SAND (<>0) når man forsøger at læse udover det sidste dataelement i en datastrøm, ellers har den værdien FALSK (=0).

Bemærkning

1. funktionen har kun mening i forbindelse med READ FILE, INPUT FILE og GET$.

Eksempel

```
0010 // åben filen, og skriv tallene fra 1 til 10 ud i den
0020 OPEN 1, "datafil", WRITE
0030 FOR i:=1 TO 10 DO WRITE FILE 1: i
0040 CLOSE 1// luk filen igen
0050
0060
0070 // åben filen igen, og læs tallene nok engang
0080 OPEN 1, "datafil", READ
0090 ZONE 20
0100 PRINT "EOF", "tal"
0110 REPEAT
0120   READ FILE 1: tal
0130   PRINT EOF(1), tal
0140 UNTIL EOF(1)
0150 CLOSE 1
0160
0170 DELETE "datafil"// slet datafilen efter brug

RUN
EOF       tal
0         1
0         2
0         3
0         4
0         5
0         6
0         7
0         8
0         9
0         10
1         10
END
at 0170
```

10.28 ERR 10.28

COMAL80 funktion

Format

ERR

Anvendelse

Bruges kun i PROC-HANDLER. Funktionen returnerer med nummeret for sidste RUN-time fejl.

Bemærkning

1. Funktionen svarer præcis til SYS(0)

Eksempel

```
0010 PROC esctest HANDLER
0020   IF ERR=100 THEN
0030     escset:=TRUE
0040     CONTINUE
0050   ENDIF
0060 ENDPROC esctest
0070
0080 ENABLE esctest
0090 escset:=FALSE
0100 antal:=1; max:=50
0110 DIM alder(max),højde(max)
0120 REPEAT
0130   INPUT "Alder ? (ESC=slut) ":alder(antal);
0140   IF NOT escset THEN
0150     INPUT " Højde ? ":højde(antal)
0160     antal:=antal+1
0170   ENDIF
0180 UNTIL escset
0190 antal:=antal-1
0200 DISABLE
0210 // Nu er alder og højde indtastet
```

10.29 Etikette 10.29

COMAL80 sætning

Format

<navn>:

<navn>: Et navn lavet efter samme regler som variabelnavne.

Anvendelse

Sætningen anvendes til at navngive et bestemt sted i programmet, hvortil andre programsætninger kan referere.

Bemærkninger

1. Man kan referere til etiketter i sætningerne GOTO, RESTORE.

2. En variabel og en etikette må ikke have det samme navn.

Eksempel

```
0010 max:=50
0020 DIM alder(max),højde(max)
0030 FOR nr:=1 TO 50 DO
0040   INPUT "Alder ? (-1=slut) ":alder(nr);
0050   IF alder(nr)=-1 THEN GOTO behandling
0060   INPUT " Højde ? ":højde(nr)
0070 NEXT nr
0080 nr:=51 // Vi trækker 1 fra om et øjeblik
0090 behandling:
0100 nr:=nr-1
0110 // Nu er alder og højde indtastet
```

10.30 EXEC 10.30

COMAL80 sætning/kommando

Format

EXEC <navn> [(<par> [,<par>]...)]

<navn>: navnet på en procedure

<par> : navnet på en simpel variabel, en vektor, en matrix, en strengvariabel, en teksttabel, et strengudtryk eller et numerisk udtryk.

Anvendelse

Sætningen anvendes til at kalde en procedure.

Virkemåde

1. Proceduren med navnet <navn> fremfindes. De evt. aktuelle parametre i EXEC bliver overført til de eventuelle formelle parametre i PROC-sætningen. Hvis antallet eller typen af parametrene ikke passer, udskrives fejl 0112: PARAMETER FEJL eller fejl 0109: TYPE KONFLIKT.

2. Programudførelsen fortsætter i underprogrammet indtil RETURN eller ENDPROC mødes.

3. Derefter fortsætter programudførelsen med næste sætning efter EXEC.

Bemærkning

1. Sætningen kan ikke udføres som kommando, hvis man har rettet i programmet siden sidste RUN.

2. Se desuden afsnit 8: Procedurer og funktioner

COMAL80 funktion

Format

EXP(<nudtr>)

<nudtr>: vilkårligt numerisk udtryk

Anvendelse

Funktionen udregner værdien af e (=2,71828182846) opløftet til potensen <nudtr>.

Eksempel

```
0010 FUNC cosh(x)//cosinus hyperbolsk
0020   RETURN (exp(x)+1/exp(x))/2
0030 ENDFUNC cosh
0040 FUNC sinh(x)//sinus hyperbolsk
0050   RETURN (exp(x)-1/exp(x))/2
0060 ENDFUNC sinh
```

Kommentar

Ved hjælp af EXP kan man udregne cosinus hyperbolsk og sinus hyperbolsk.

10.32 FALSE

10.32

COMAL80 konstant

Format

FALSE

Anvendelse

FALSE er en konstant med værdien 0.

Eksempel

```
0010 DIM måned$ OF 3, svar$ OF 3
0020 REPEAT
0030   INPUT "Skriv en måneds navn >": svar$
0040   fundet:= FALSE
0050   RESTORE
0060   WHILE NOT fundet AND NOT EOD DO
0070     READ måned$, dage
0080     IF måned$=svar$ THEN
0090       fundet:= TRUE
0100       PRINT måned$;" har ";dage;" dage."
0110     ENDIF
0120   ENDWHILE
0130   IF NOT fundet THEN PRINT "Den måned kender jeg ikke"
0140 UNTIL FALSE// uendelig løkke
0150 DATA "jan",31,"feb",28,"mar",31,"apr",30,"maj",31,"jun",30
0160 DATA "jul",31,"aug",31,"sep",30,"okt",31,"nov",30,"dec",31
```

```
RUN
Skriv en måneds navn >jul
jul har 31 dage.
Skriv en måneds navn >jam
Den måned kender jeg ikke
Skriv en måneds navn >jan
jan har 31 dage.
```

COMAL80 sætning

Format

FOR <tvar>:= <stværdi> TO <slværdi> [STEP <trværdi>] DO <simpel sætning>

<tvar>: tællevariabel, en simpel numerisk variable
<stværdi>: numerisk udtryk, startværdi
<slværdi>: numerisk udtryk, slutværdi
<trværdi>: numerisk udtryk, trinværdi
<simpel sætning>: en simpel COMAL80 sætning, normalt CLOSE, CREATE, DELETE, EXEC, INPUT, INPUT FILE, MOUNT, OPEN FILE, PRINT, PRINT USING, PRINT FILE, READ, READ FILE, RESTORE, SELECT OUTPUT, tildeling, WRITE FILE.

Anvendelse

Sætningen benyttes til at udføre en <simpel sætning> et bestemt antal gange.

Virkemåde

1. <stværdi>, <slværdi> og <trværdi> udregnes. Hvis <trværdi> ikke er angivet, bliver den sat til +1.

2. <tvar> sættes lig <stværdi>

3. Hvis <trværdi> er positiv (negativ) og <tvar> er større (mindre) end <slværdi> er slutbetingelsen opfyldt og programmet fortsætter med næste sætning.

4. <simpel sætning> udføres.

5. <tvar> sættes lig <tvar> plus <trværdi> og trin 3 udføres igen.

Bemærkninger

1. Sætningen kan ikke udføres som kommando.

2. Ønskes mere end en <simpel sætning> udført, skal konstruktionen FOR-NEXT benyttes.

Eksempel

```
0010 ZONE 10
0020 FOR i: = 1 TO 10 STEP 2 DO PRINT i,
0030 END

RUN
1          3          5          7          9
END
at 0030
```

10.34 FOR-NEXT 10.34

COMAL80 struktur

Format

FOR <tvar>:= <stværdi> TO <slværdi> [STEP <trværdi>] DO
  <sætningsliste>
NEXT <tvar>

<tvar>: tællevariabel, en simpel numerisk variable
<stværdi>: numerisk udtryk, startværdi
<slværdi>: numerisk udtryk, slutværdi
<trværdi>: numerisk udtryk, trinværdi
<sætningsliste>: COMAL80 sætninger

Anvendelse

Konstruktion benyttes til at repetere udførslen af en <sætningsliste> et bestemt antal gange.

Virkemåde

1. <stværdi>, <slværdi> og <trværdi> udregnes. Hvis <trværdi> ikke er angivet, sættes den lig +1.

2. <tvar> sættes lig <stværdi>

3. Hvis <tværdi> er positiv (negativ), og <tvar> er større (mindre) end <slværdi>, er slutbetingelsen opfyldt, og programmet fortsætter med første sætning efter NEXT-sætningen.

4. <sætningsliste> udføres.

5. <tvar> sættes lig <tvar> plus <trværdi>, og trin 3 udføres igen.

Bemærkninger

1. Strukturen kan ikke udføres som kommando.

2. FOR-NEXT kan indeholde andre FOR-NEXT sætninger.

3. Hvis NEXT mangler fås fejl 0096: FEJL I PROGRAMSTRUKTUR.

4. Hvis man hopper ind i en FOR-NEXT løkke udenom FOR-sætningen fås fejl 0116: ULOVLIGT HOP.

10.35 FUNC-ENDFUNC 10.35

COMAL80 struktur

Format

```
FUNC <navn> [( <par>  [,<par>]... )] [CLOSED]
  [GLOBAL <symbol> [,<symbol>]...]
  <sætningsliste>
ENDFUNC <navn>
```

<par>: { [REF] <nvar> | [REF] <svar> | REF <ntabl> ([,]) | REF <stabl> () }

<navn>:	Navn på funktionen (max. 16 tegn)
<nvar>:	Vilkårligt numerisk variabel navn
<svar>:	Vilkårligt strengvariabel navn
<ntabl>:	Vilkårligt vektor- eller matrix-navn
<stabl>:	Vilkårligt teksttabel navn
<symbol>:	Navn på en global procedure, funktion, variabel, strengvariabel, vektor, matrix eller teksttabel
<sætningsliste>:	COMAL80 sætninger.

Anvendelse

Konstruktionen benyttes til at definere en funktion.

Bemærkninger

1. <navn> skal være forskellig fra alle andre navne på procedurer, funktioner, etiketter, numeriske variable og strengvariable.

2. En funktion kan gøres lukket (CLOSED), så variable, der optræder i funktionen, er lokale og ikke berører variablerne i resten af programmet (de globale variable). Ønsker man at benytte en variabel, der optræder i resten af programmet, benyttes GLOBAL-sætningen. De lokale variable, der optræder i funktionen, slettes når man forlader funktionen.

3. Anføres REF foran variabelnavnet i parameterlisten (den formelle parameter) vil variablen i kaldet af funktionen (den reelle parameter) blive ændret samtidig med, at den formelle parameter ændres i funktionen. På denne måde kan man returnere resultater via parametrene.

4. Værdien af funktionen tildeles med RETURN-sætningen, hvorved man tillige returnerer fra funktionen.

Eksempel

```
0010 FUNC fahrenheit(celcius)
0020   // Funktionen omformer celcius til fahrenheit
0030   RETURN celcius*9/5+32
0040 ENDFUNC fahrenheit
0050
0060 MARGIN 80
0070 ZONE 10
0080 PRINT "Celcius","Fahrenheit"
0090 FOR c:=0 TO 100 STEP 20 DO PRINT c,fahrenheit(c)

RUN
Celcius    Fahrenheit
0          32
20         68
40         104
60         140
80         176
100        212
END
AT 0090
```

10.36 FUNC-EXTERNAL 10.36

COMAL80 sætning

Format

FUNC <navn> [(<par> [,<par>]...)] EXTERNAL <filnavn>

<par>: ⎧ [REF] <nvar> ⎫
⎪ [REF] <svar> ⎪
⎪ REF <ntabl> ([,]) ⎪
⎩ REF <stabl> () ⎭

<navn>: Navn på funktionen (max 16 tegn)
<nvar>: Vilkårligt numerisk variabelnavn
<svar>: Vilkårlig strengvariabelnavn
<ntabl>: Vilkårligt vektor- eller matrix-navn
<stabl>: Vilkårlig tekstabelnavn
<filnavn>: Navn på den diskettefil, hvori den externe funktion er gemt med kommandoen SAVE.

Anvendelse

Sætningen benyttes til at erklære en extern funktion.

Bemærkninger

1. Den externe funktion skal være SAVE't i <filnavn> og første linie i dette program skal være et funktionshoved for en lukket (CLOSED) funktion. Den lukkede funktion må ikke indeholde GLOBAL-sætninger.

2. Parameterbeskrivelsen i FUNC-EXTERNAL skal svare nøjagtig til parameterbeskrivelsen i funktionshovedet i den SAVE'de funktion.

3. Senere i programmet kan man referere til den externe funktion som man refererer til en almindelig FUNC-ENDFUNC. Ved hvert funktionskald af den externe funktion bliver den indlæst fra disketten.

4. Den externe funktion kan indeholde andre procedurer og funktioner, både åbne, lukkede samt externe.

5. Hvis en extern funktion kalder sig selv, indlæses den ikke igen.

6. Stopper programudførelsen under udførelsen af en extern funktion, er det kun denne, man kan se med LIST-kommandoen. SAVE-kommandoen gemmer da også kun den externe funktion hvorimod RUN genstarter hovedprogrammet. NEW sletter både hovedprogram og den externe funktion.

7. Externe funktioner gør det muligt at oprette biblioteker af funktioner, der kan benyttes af flere forskellige programmer.

10.37 GET$ 10.37

COMAL80 funktion

Format

GET$ (<strømnr>, <nudtr>)

<strømnr>: et numerisk udtryk, hvis værdi angiver nummeret på en åben datastrøm.

<nudtr>: et vilkårligt positivt numerisk udtryk.

Anvendelse

Funktionen anvendes til at læse et bestemt antal tegn (<nudtr>) fra en diskettefil eller en ydre enhed.

Bemærkning

1. Funktionen benyttes til at læse "formatløst" i en datastrøm.

Eksempel 1

Dette lille program udskriver indholdet af en diskettefil. Brugeren har mulighed for at vælge en ren hexadecimal udskrift, eller vælge mellem tekst- og hexadecimal-udskrift.

```
0010 DIM unit$ OF 1, navn$ OF 11, hextal$ OF 16
0020 DIM buf$ OF 512, svar$ OF 3
0030 hextal$:= "0123456789ABCDEF"
0040 INPUT "Unitnr : ": unit$
0050 INPUT "Filnavn: ": navn$
0060 REPEAT
0070   INPUT "Tekst og Hex-udskrift : ": svar$
0080 UNTIL svar$="ja" OR SVAR$="nej"
0090 tekst:= (svar$="ja")
0100 OPEN 1, "/"+unit$+"/"+navn$, READ
0110 recnc:= 1
0120 WHILE NOT EOF(1) DO
```

```
0130    buf$:= GET$(1,512)
0140    PRINT "Fil : ";navn$;TAB(40);"recordnr : ";recno
0150    FOR i:= 1 TO 512 DO
0160      IF (i-1) MOD 30=0 THEN
0170        PRINT
0180        PRINT USING "#### ":i;
0190      ENDIF
0200      IF i MOD 2=1 THEN PRINT " ";
0210      IF tekst AND (buf$(i:i)>=" " AND buf$(i:i) <="å") THEN
0220        PRINT ".";buf$(i:i);
0230      ELSE
0240        i1:=ord(buf$(i:i)) DIV 16 + 1
0250        i2:=ord(buf$(i:i)) MOD 16 + 1
0260        PRINT hextal$(i1:i1);hextal$(i2:i2);
0270      ENDIF
0280    NEXT i
0290    PRINT
0300    PRINT
0310    recno:= recno+1
0320  ENDWHILE
0330  CLOSE
0340  END
```

Eksempel 2

Kopieringsprogram

```
0010 DIM ifil$ OF 20, ufil$ OF 20, buf$ OF 512
0020 INPUT "INPUTFIL: ": ifil$
0030 INPUT "OUTPUTFIL: ": ufil$
0040 CREATE ufil$, 0
0050 OPEN 1, ifil$, READ
0060 OPEN 2, ufil$, WRITE
0070 WHILE NOT EOF(1) DO
0080   PRINT FILE 2: GET$(1,512)
0090 ENDWHILE
0100 CLOSE
0110 END
```

10.38 GLOBAL 10.38

COMAL80 sætning

Format

GLOBAL <navn> [,<navn>]...

<navn>: navn på en variabel, vektor, matrix, streng, strengtabel, etikette, procedure eller funktion.

Anvendelse

Sætningen anvendes kun i FUNC-ENDFUNC og PROC-ENDPROC, der er CLOSED. Sætningen benyttes til at angive hvilke globale variable der skal kunne refereres til i en lukket procedure eller funktion.

10.39 GOTO

COMAL80 sætning

Format

GOTO <navn>

<navn>: Navnet på en etikette.

Anvendelse

Sætningen anvendes til at foretage et hop i programudførslen.

Bemærkninger

1. Når GOTO udføres, bliver den næste sætning, der skal udføres, sætningen efter etiketten <navn>.

2. Det er ikke tilladt at hoppe ind i strukturerne IF-THEN-ELSE-ENDIF, CASE-ENDCASE, PROC-ENDPROC, REPEAT-UNTIL, WHILE-ENDWHILE og FOR-NEXT.

3. Man må ikke hoppe ud af PROC-ENDPROC.

Eksempel

```
0010 DIM svar$ OF 3
0020 WHILE TRUE DO
0030   INPUT "Ønsker du at fortsætte (ja/nej) : ": svar$
0040   IF svar$="nej" THEN GOTO slut
0050 ENDWHILE
0060 slut:
0070 END

RUN

Ønsker du at fortsætte (ja/nej) : ja
Ønsker du at fortsætte (ja/nej) : nej
END
at 0070
```

10.40 IF-THEN 10.40

COMAL80 sætning

Format

IF <logisk udtryk> THEN <simpel sætning>

<logisk udtryk>: et udtryk der, når det udregnes, har værdien sand (<>0) eller falsk (=0).

<simpel sætning>: en simpel COMAL80 sætning, dvs. CLOSE, CREATE, DELETE, END, EXEC, GOTO, INPUT, INPUT FILE, MOUNT, OPEN FILE, PRINT, PRINT FILE, PRINT USING, READ, READ FILE, RESTORE, RETURN, SELECT OUTPUT, STOP, tildeling, WRITE FILE.

Anvendelse

Bruges til at betinge udførslen af en sætning.

Virkemåde

1. Hvis <logisk udtryk> er sand (<>0), bliver <sætning> udført. Hvis <sætning> ikke bevirker hop i programmet, vil programudførslen fortsætte med første programlinie efter IF-THEN sætningen.

2. Hvis værdien af <logisk udtryk> er falsk (=0), vil <sætning> ikke blive udført og programudførelsen fortsætter med første linie efter IF-THEN sætningen.

Eksempel

```
0010 // Simpel forgrening
0020 INPUT i, j
0030 PRINT "De to tal er ";
0040 IF i<>j THEN PRINT "ikke ";
0050 PRINT "ens"
0060 END
```

10.41 IF-THEN-ENDIF 10.41

COMAL80 struktur

Format

```
IF <logisk udtryk> THEN
   <sætningsliste>
ENDIF
```

<logisk udtryk>: et udtryk, som kan have værdien sand (<>0) eller falsk (=0)

<sætningsliste>: COMAL80 sætninger.

Anvendelse

Konstruktionen bruges til at gøre udførelsen af en blok sætninger betinget af, om et udtryk er sandt eller falsk.

Virkemåde

1. Hvis værdien af <logisk udtryk> er sand (<>0), vil <sætningsliste> blive udført een gang.

2. Programudførelsen vil fortsætte med den første sætning efter ENDIF sætningen hvis man ikke foretager hop ud af strukturen.

Bemærkning

1. Sætningen kan ikke udføres som kommando.

Eksempel

```
0010 // IF-THEN-ENDIF
0020 INPUT i,j
0030 IF i<>j THEN
0040   PRINT "De to tal er ikke ens."
0050   PRINT "Forskellen er ";ABS(i-j)
0060 ENDIF
0070 END
```

10.42 IF-THEN-ELSE-ENDIF 10.42

COMAL80 struktur

Format

```
IF <logisk udtryk> THEN
   <sætningsliste 1>
ELSE
   <sætningsliste 2>
ENDIF
```

<logisk udtryk>: et udtryk, som kan have værdien sand (<>0) eller falsk (=0).

<sætningsliste 1>: en række COMAL80 sætninger der vil blive udført, hvis værdien af <logisk udtryk> er sand (<>0).

<sætningsliste 2>: en række COMAL80 sætninger der vil blive udført, hvis værdien af <logisk udtryk> er falsk (=0).

Anvendelse

Konstruktionen benyttes til at udføre en af to blokke COMAL80 sætninger, afhængig af værdien af et logisk udtryk.

Virkemåde

1. <logisk udtryk> udregnes.

2. Hvis værdien af <logisk udtryk> er sand (<>0) udføres <sætningsliste 1>

3. Hvis værdien af <logisk udtryk> er falsk (=0) udføres <sætningsliste 2>

4. Når <sætningsliste 1> eller <sætningsliste 2> er blevet udført, og der ikke er foretaget hop ud af listen, fortsættes programudførelsen med første sætning efter ENDIF.

Bemærkning

1. Hvis man hopper ind i <sætningsliste 1> eller <sætningsliste 2> udenom IF-THEN-ELSE, udskrives fejlmeddelelsen 116: ULOVLIGT HOP.

Eksempel

```
0010 REPEAT
0020   READ tal1,tal2
0030   PRINT tal1;"+ ";tal2;
0040   INPUT "= ?": svar
0050   IF svar=tal1+tal2 THEN
0060     PRINT "Det er korrekt"
0070   ELSE
0080     PRINT "Det er forkert"
0090     PRINT "Det rigtige svar er: ";tal1+tal2
0100   ENDIF
0110 UNTIL EOD
0120 DATA 2,0,3,1,5,3
RUN
2 + 0 = ? 2
Det er korrekt
3 + 1 = ? 5
Det er forkert
Det rigtige svar er: 4
5 + 3 = ? 8
Det er korrekt
END
at 0120
```

COMAL80 operator

Format

<sudtr1> IN <sudtr2>

(<sudtr1> IN <sudtr2>) returnerer et ikke negativt heltal

<sudtr1>: strengudtryk

<sudtr2>: strengudtryk

Operatorprioritet = 5

Anvendelse

Operatoren finder positionen for <sudtr1> i <sudtr2>.

Bemærkninger

1. Operatoren returnerer positionen for <sudtr1>'s første tegn i <sudtr2>.

2. Findes <sudtr1> ikke i <sudtr2> returneres værdien 0.

3. Hvis <sudtr1> er tom returneres med længden af <sudtr2> plus 1.

Eksempel

```
0010 DIM ifil$ OF 20,ufil$ OF 20
0020 DIM fra$ OF 132,til$ OF 132,l$ OF 132
0030 PRINT "Programmet retter den samme tekst"
0040 PRINT "alle steder i en tekstfil."
0050 PRINT "Dog kun en rettelse pr. linie"
0080 INPUT "Inputfil :": ifil$
0090 INPUT "Outputfil :": ufil$
0100 INPUT "Hvilken tekst skal ændres ?": fra$
0110 INPUT "Ændres til hvilken tekst ?": til$
0120 OPEN 1,ifil$, READ
0130 OPEN 2,ufil$, WRITE
0140 forsk:=LEN(til$)-LEN(fra$)
0150 WHILE NOT EOF(1) DO
0160   INPUT FILE 1:l$
0170   IF NOT EOF(1) THEN
0180     p:=(fra$ IN l$)
0190     IF p<>0 THEN
0200       l$(p:len(l$)+forsk):=til$+l$(p+LEN(fra$):LEN(l$))
0210     ENDIF
0220   ENDIF
0230   PRINT FILE 2: l$
0240 ENDWHILE
0250 CLOSE
0260 END
```

10.44 INPUT 10.44

COMAL80 sætning

Format

INPUT [<skonst>:] { <nvar> / <svar> } [, { <nvar> / <svar> }] ...

INPUT AT(<nudtr1>,<nudtr2>) [, <skonst>]: { <nvar> / <svar> } [, { <nvar> / <svar> }] ...

<skonst>: en strengkonstant
<nvar> : en numerisk variabel
<svar> : en strengvariabel
<nudtr1>: et numerisk udtryk med værdi i intervallet 1≤ <nudtr1> ≤ 80
<nudtr2>: et numerisk udtryk med værdi i intervallet 1≤ <nudtr2> ≤ 25

Anvendelse

Sætningen bruges til at tildele værdier til variable fra tastaturet, når et program kører.

Virkemåde

1. Hvis AT(<nudtr1>,<nudtr 2>) er angivet, flyttes udskriften til den tilhørende position på skærmen (se AT afsnit 10.3).

2. Hvis <skonst> er angivet udskrives denne.

3. Cursoren fremkommer på skærmen, som tegn på at systemet forventer input.

4. Herefter kan brugeren indtaste værdien for første INPUT-element.

5. Skal en numerisk variabel indtastes, kan indtastningen af den afsluttes med vognretur, eller, hvis der er yderligere INPUT-elementer, med et komma eller et mellemrum.

6. Skal en strengvariabel indtastes skal indtastningen afsluttes med vognretur.

7. Hvis der er flere INPUT-elementer, fortsættes med det næste element (hop til trin 5).

8. Hvis INPUT-sætningen afsluttes med et semikolon (;) vil efterfølgende udskrift fortsættes umiddelbart efter det indtastede, ellers udføres en vognretur.

10.45 INPUT FILE 10.45

COMAL80 sætning/kommando

Format

INPUT FILE <strømnr>: { <nvar> / <svar> } [, { <nvar> / <svar> }] ...

<strømnr>: et taludtryk, hvis værdi angiver nummeret på en åben datastrøm.

<nvar> : en numerisk variabel

<svar> : en strengvariabel

Anvendelse

Sætningen indlæser data i ASCII-format fra en datastrøm, der er åbnet (OPENed) i READ-mode.

Bemærkninger

1. Hvert argument i INPUT FILE-sætningen skal have samme type (numeriske eller strenge) som dataene i datastrømmen, ellers fås fejl 0109: TYPEKONFLIKT.

2. Hvis datastrømmen er en diskette-fil, skal den være skrevet ved hjælp af PRINT FILE, SELECT OUTPUT (og DIR, LIST, PRINT etc.).

3. Input datastrømmen skal være opbygget således at tegnet CR (se appendix E) adskiller de enkelte elementer.

4. Hvis længden af en streng i datastrømmen er større end den dimensionerede længde af den tilsvarende streng-variabel, vil de overskydende tegn blive ignoreret.

5. I øvrigt henvises til afsn. 7: Indlæsning og udskrivning.

Eksempel

```
0010 DIM ifil$ OF 17,ufil$ OF 17,l$ OF 132
0020 INPUT "Listning af hvilket program ? ": ifil$
0030 INPUT "Listning hvorhen ? ":ufil$
0040 INPUT "Sidehøjde ? ": sidehøjde
0050 OPEN FILE 1,ifil$,READ
0060 SELECT OUTPUT ufil$
0070 lnr:=0; side:=0
0080 EXEC sideskift
0090 WHILE NOT EOF(1) DO
0100   INPUT FILE 1: l$
0110   IF NOT EOF(1) THEN
0120     PRINT l$
0130     lnr:=lnr+1
0140     if lnr=sidehøjde THEN EXEC sideskift
0150   ENDIF
0160 ENDWHILE
0170 CLOSE
0180 SELECT OUTPUT "console"
0190 END
0200 PROC sideskift
0210   side:=side+1; lnr:=1
0220   PRINT CHR$(12);TAB(70);"SIDE ";side
0230 ENDPROC sideskift
```

10.46 INT 10.46

COMAL80 funktion

Format

INT(<nudtr>)

<nudtr> : et vilkårligt numerisk udtryk

Anvendelse

Funktionen udregner det nærmeste heltal, der ikke er større end <nudtr>.

Eksempel

```
0010 FUNC afrund(x)// afrunder efter normale regler
0020   return INT(x+0.5)
0030 ENDFUNC afrund
0040
0050 MARGIN 80
0060 ZONE 20
0120 PRINT "Tal", "Heltalsværdi", "Afrundet"
0180 RANDOMIZE
0190 FOR i:= 1 TO 10 DO
0200   tal:= RND*100-50
0210   PRINT tal, INT(tal), afrund(tal)
0220 NEXT i
0230 END
```

```
RUN
Tal                 Heltalsværdi    Afrundet
38.8569663724       38              39
2.98225784899       2               3
-1.55790098222      -2              -2
-21.79328204523     -22             -22
13.90425481596      13              14
-34.64199176265     -35             -35
-27.57420626506     -28             -28
-12.49762589527     -13             -12
43.28272182272      43              43
-25.63532655509     -26             -26
END
at 0230
```

10.47 KEY$ 10.47

COMAL80 funktion

Format

KEY$

Anvendelse

Funktionen returnerer med værdien af den sidste tast, der er trykket ned på tastaturet.

Bemærkninger

1. Hvis ingen tast har været rørt siden sidste INPUT, GET$ eller KEY$ returneres med værdien CHR$(0).

2. Proceduren venter således ikke på, at der trykkes på en tast. Hvis brugeren ønsker dette, skal han åbne en datastrøm til tastaturet (keyboard) og foretage kald af GET$ (se appendix F.4).

Eksempel

Denne funktion venter en bestemt tid på et tegn fra tastaturet og returnerer, hvis tiden er gået, eller der er modtaget et tegn.

```
0010 FUNC tchar(tid) CLOSED
0020   nu:=SYS(3)
0030   REPEAT
0040     i:=ORD(KEY$)
0050   UNTIL SYS(3)>nu+tid OR i<>0
0060   RETURN i
0070 ENDFUNC tchar
```

10.48 LEN 10.48

COMAL80 funktion

Format

LEN(<sudtr>)

<sudtr>: et vilkårligt strengudtryk

Anvendelse

Funktionen returnerer den aktuelle længde (antal tegn) af strengen <sudtr>.

Bemærkninger

1. LEN-funktionen må optræde i et vilkårligt numerisk udtryk.

2. Hvis <sudtr> er den tomme streng, returneres med værdien 0.

3. Bemærk, at funktionen returnerer med det aktuelle antal tegn i strengen og ikke med det antal, en strengvariabel er dimensioneret til.

Eksempel

```
0010 DIM tekst$ OF 80, konsonant$ OF 20, vokal$ OF 8
0020 konsonant$:= "bcdfghjklmnpqrstvwxz"
0030 vokal$:= "aeiouyæøå"
0040 INPUT "Indtast tekst (små bogstaver):": tekst$
0050 antkons:= 0; antvokal:= 0
0060 FOR i:= 1 TO LEN(tekst$) DO
0070   IF tekst$(i:i) IN konsonant$ THEN antkons:= antkons+1
0080   IF tekst$(i:i) IN vokal$ THEN antvokal:= antvokal+1
0090 NEXT i
0100 PRINT tekst$
0110 PRINT "indeholder ";antkons;"konsonanter"
0120 PRINT "og ";antvokal;"vokaler."
0130 END
```

10.49 LIST 10.49

COMAL80 kommando

Format

```
                <lnr1> [,]
LIST [{                 , <lnr2>}]      [<filnavn>]
                <lnr1>,<lnr2>
```

<lnr1>: linienummeret på den første sætning der skal listes.
<lnr2>: linienummeret på den sidste sætning der skal listes.
<filnavn>: navnet på en datastrøm angivet i anførselstegn.

Anvendelse

Kommandoen anvendes til at udskrive hele eller dele af det program, der ligger i programlageret.

Bemærkninger

1. Angives <filnavn>, udskrives programmet på <filnavn> istedet for på skærmen.

2. Kombinationerne af parametrene til LIST-komamndoen har følgende betydning:

LIST	Udskriver hele programmet i programlageret.
LIST <lnr1>	Udskriver linien med linienummeret <lnr1>
LIST <lnr1>,	Udskriver alle de linier, hvor linienummeret >= <lnr1>
LIST , <lnr2>	Udskriver alle de linier, hvor linienummeret <= <lnr2>
LIST <lnr1>,<lnr2>	Udskriver alle de linier, hvor <lnr1> <=linienummeret <= <lnr2>

3. Ønskes udskriften standset midlertidigt, trykkes på mellemrumstangenten. Ved næste tryk genstartes udskriften.

4. Udskriften afbrydes ved tryk på ESC-tasten.

Eksempel 1

Hvis vi i det følgende forudsætter, at programlageret indeholder følgende program:

```
0010 print
0020 print
0030 print
0040 print
0050 print
0060 print
```

Da vil:

1.	LIST	udskrive linierne 0010, 0020, 0030, 0040, 0050, 0060
2.	LIST 20	udskrive linien 0020
3.	LIST 40,	udskrive linierne 0040, 0050, 0060
4.	LIST ,20	udskrive linierne 0010, 0020
5.	LIST 30,40	udskrive linierne 0030, 0040

Eksempel 2

Ønskes programmet i programlageret udskrevet på printeren udføres kommandoerne

```
LIST "printer"
```

10.50 LOAD 10.50

COMAL80 kommando

Format

LOAD [<filnavn>]

<filnavn>: navnet på en diskettefil angivet i anførselstegn.

Anvendelse

Kommandoen bruges til at hente et program fra en diskettefil. Programmet skal være gemt med SAVE-kommandoen.

Bemærkning

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

2. Udelades <filnavn> vil kommandoen hente indholdet af den fil, der er angivet i statuslinien.

3. Efter LOAD vil statuslinien indeholde navnet på den fil, der er LOADed.

Eksempel

LOAD "/2/MITPROGRAM"

COMAL80 funktion

Format

LOG(<nudtr>)

<nudtr>: et positivt numerisk udtryk.

Anvendelse

Funktionen udregner den naturlige logaritme af <nudtr>.

Eksempel

```
0010 FUNC log10(x)// fkt. udregner ti-tals logaritmen til x
0020   RETURN LOG(x)/LOG(10)
0030 ENDFUNC log10
0040
0050 ZONE 20
0060 PRINT "x","nat.log","titalslog."
0070 FOR i:= 1 TO 10 DO
0080   tal:= RND*200
0090   PRINT tal,LOG(tal),log10(tal)
0100 NEXT i
0110 END
```

```
RUN
x                   nat.log           titalslog.
22.32006388358      3.105485999676    1.348695433287
167.8980906691      5.123357192184    2.225045757385
164.0659968311      5.100268766578    2.215018581549
7.41540988152       2.003560251407    0.8701351613471
129.5512502923      4.864076558003    2.112441608696
177.4245282954      5.178545325793    2.249013659278
8.59768468306       2.15149294411     0.9343815134811
161.549995607       5.084814665078    2.208306950545
174.0521613794      5.159355032333    2.240679420723
134.8887992722      4.904450729909    2.129975888766
END
at 0110
```

10.52 MARGIN

COMAL80 sætning/kommando

Format

MARGIN <nudtr>

<nudtr>: Vilkårligt ikke negativt numerisk udtryk.

Anvendelse

Sætningen/kommandoen sætter højre-margen på udskriftslinien.

Bemærkninger

1. Standardværdien ved opstart er MARGIN 80.

2. Angives MARGIN 0 vil systemet på intet tidspunkt tilføje et linieskift i udskriften. Dette bør angives i forbindelse med skærmadressering (AT-funktionen).

Eksempel

```
0010 MARGIN 40
0020 ZONE 5
0030 FOR i:= 1 TO 10 DO PRINT i,
0040 PRINT
0050 MARGIN 30
0060 FOR i:= 1 TO 10 DO PRINT i,
0070 PRINT
0080 END

RUN
1     2     3     4     5     6     7     8
9     10
1     2     3     4     5     6
7     8     9     10
END
at 0080
```

COMAL80 operator

Format

<nudtr1> MOD <nudtr2>

<nudtr1>: et vilkårligt numerisk udtryk opfattet som heltalsvariabel

<nudtr2>: et vilkårligt numerisk udtryk forskellig fra nul opfattet som heltalsvariabel.

Operatorprioritet = 3

Anvendelse

Operatoren udregner resten ved en heltalsdivision

Bemærkninger

1. Definitionen på a MOD b er lig
   ABS(INT(a+0.5)-(a DIV b)*int(b+0.5))

2. <nudtr1> og <nudtr2> må ikke have værdier større end 32767 eller mindre end -32768.

Eksempel

```
0010 FUNC sfd(m,n)
0020   IF n=0 THEN
0030     RETURN m
0040   ELSE
0050     RETURN sfd(n,m MOD n)
0060   ENDIF
0070 ENDFUNC sfd
0080
0090 ZONE 20
0100 WHILE NOT EOD DO
0110   READ tal1,tal2
0120   PRINT tal1,tal2,sfd(tal1,tal2)
0130 ENDWHILE
0140 DATA 7,14,60,24,13,19
RUN
7                   14                  7
60                  24                  12
13                  19                  1
END
at 00140
```

10.54 MOUNT 10.54

COMAL80 sætning/kommando

Format

MOUNT <nudtr>

<nudtr>: Et numerisk udtryk, hvis værdi angiver et unitnr (dvs. 1 eller 2).

Anvendelse

Sætningen/kommandoen benyttes til at angive overfor systemet at der er monteret en ny diskette i drive nr. <nudtr>, hvorpå der må læses og skrives.

Bemærkning

1. Udskiftes en diskette i et drive, skal man udføre en MOUNT, førend man kan skrive eller læse på disketten. Er maskinen udstyret med 8" disketter fås ellers fejlmeldingen 0208: DISKETTE ER BLEVET SKIFTET.

2. Man må ikke skifte diskette og udføre MOUNT før alle filer på disketten er lukket (CLOSEd), ellers fås fejl 0219: ÅBNE FILER PÅ ENHED.

3. MOUNT udføres automatisk for disketten i disketteenhed nummer 1 i forbindelse med opstart af COMAL80.

10.55 NEW 10.55

COMAL80 kommando

Format

NEW [filnavn]

<filnavn> : navnet på en diskettefil angivet i anførselstegn.

Anvendelse

Kommandoen bruges til at slette programmet og data i maskinens lager, samt lukke eventuelle åbne filer.

Kommentarer

1. NEW anvendes, når brugeren ønsker at indtaste et nyt program.

2. MARGIN- og ZONE-værdierne ændres ikke ved NEW.

3. Alle åbne datastrømme lukkes ved NEW.

4. Angives filnavn vil filnavnet blive udskrevet i statuslinien, ellers vil den indeholde navnet 1/$SAVE.

10.56 NOT 10.56

COMAL80 operator

Format

NOT <nudtr>

<nudtr>: vilkårligt numerisk udtryk opfattet som logisk udtryk (0:falsk, <>0: sand).

Operatorprioritet = 6

Anvendelse

Operatoren anvendes til at negere det logiske udtryk <nudtr>. Hvis <nudtr> er falsk (=0) er NOT <nudtr> sand (=1), hvis <nudtr> er sand (<>0) er NOT <nudtr> falsk (=0).

Eksempel

```
0010 OPEN 1,"klassedata",READ
0020 sumalder:=0; sumhøjde:=0; antal:=0
0030 READ FILE 1: alder,højde
0040 WHILE NOT EOF(1) DO
0050   sumalder:=sumalder+alder; sumhøjde:=sumhøjde+højde
0060   antal:=antal+1
0070   READ FILE 1: alder,højde
0080 ENDWHILE
0090 PRINT "Antal indlæste data : ";antal
0100 PRINT "Gennemsnitsalder    : ";sumalder/antal
0110 PRINT "Gennemsnitshøjde    : ";sumhøjde/antal
0120 END
```

COMAL80 sætning/kommando

Format

OPEN [FILE] <strømnr>,<filnavn> , { READ | WRITE | APPEND | RANDOM <recl> }

<strømnr>: Vilkårligt numerisk heltal i intervallet 1 ≤<strømnr> ≤ 5

<filnavn>: Navnet på en ydre enhed eller en diskettefil.

<recl>: Numerisk heltal, der angiver det maximale antal tegn i hver post

Anvendelse

Sætningen/kommandoen benyttes til at knytte et logisk nummer (<strømnr>) til en datastrøm

Bemærkninger

1. OPEN-sætningen har én parameter, der angiver, hvorledes strømmen skal benyttes:
   READ: læsning af fil (READ FILE- og INPUT FILE-sætninger)
   WRITE: skrivning af fil (WRITE FILE- og PRINT FILE-sætninger)
   APPEND: skrivning af fil, når uddata skal hægtes bag på allerede udskrevet data, (WRITE FILE- og PRINT FILE-sætninger).
   RANDOM: binær udskrivning og indlæsning af direkte tilgang (random access) fil (READ FILE-, WRITE FILE-sætninger).

2. En diskettefil oprettes automatisk, hvis WRITE angives.

3. Hvis READ, WRITE angives, vil systemet positionere til begyndelsen af filen. Hvis APPEND angives, vil systemet positionere til lige efter de sidst udskrevne data.

4. Når OPEN er udført tilknyttes <strømnr> til strømmen.
   Yderligere referencer (READ, WRITE, CLOSE etc) til strømmen skal herefter foregå via <strømnr>.

5. <recl> angiver poststørrelsen i tegn (8 bit bytes).
   Følgende udregningsregler skal anvendes:
   - en numerisk variabel fylder 8 tegn
   - en streng fylder "længden af strengen + 2 " tegn

6. Iøvrigt henvises til afsnit 7: Indlæsning og udskrivning.

Eksempel

```
0010 OPEN FILE 1,"datafil",READ
0020 sum:=0; antal:=0
0030 READ FILE 1: tal
0040 WHILE NOT EOF(1) DO
0050   sum:=sum+tal; antal:=antal+1
0060   READ FILE 1:tal
0070 ENDWHILE
0080 CLOSE FILE 1
0090 PRINT antal;" tal indlæst. Summen af tallene er :";sum
0100 END
```

COMAL80 operator

Format

<nudtr1> OR <nudtr2>

<nudtr1>: vilkårligt numerisk udtryk opfattet som logisk udtryk (0= falsk, <>0= sand).

<nudtr2>: vilkårligt numerisk udtryk opfattet som logisk udtryk (0=falsk, <>0= sand).

Operatorprioritet = 8

Anvendelse

Den logiske operator OR er sand (<>0), hvis enten <nudtr1> er sand (<>0), eller <nudtr2> er sand (<>0), eller begge er sande. Udtrykket er altså kun falsk (=0), hvis både <nudtr1> og nudtr2> er falske.

Eksempel

```
0010 DIM logisk$(2) OF 5
0020 logisk$(1):="SAND"; logisk$(2):="FALSK"
0030 ZONE 10
0040 PRINT "a","b","a OR b"
0050 FOR i:=1 TO 30 DO PRINT "-";
0060 PRINT
0070 FOR a:=FALSE TO TRUE DO
0080   FOR b:=FALSE TO TRUE DO
0090     PRINT logisk$(a+1),logisk$(b+1),logisk$((a OR b)+1)
0100   NEXT b
0110 NEXT a
0120 END
RUN
a          b          a OR b
------------------------------
SAND       SAND       SAND
SAND       FALSK      SAND
FALSK      SAND       SAND
FALSK      FALSK      FALSK
```

COMAL80 funktion

Format

ORD(<sudtr>)

Anvendelse

Funktionen returnerer ASCII-værdien for det første tegn i <sudtr>.

Bemærkninger

1. Værdien, der returneres, er lig den interne værdi. Sammenhængen mellem et tegn og dets interne værdi fremgår af appendix E.

2. ORD-funktionen må indgå i et vilkårligt numerisk udtryk.

10.60 OUT 10.60

COMAL80 sætning/kommando

Format

OUT <filnavn>

<filnavn>: navnet på en datastrøm.

Anvendelse

Sætningen/kommandoen anvendes til at vælge udskriftsdatastrøm.

Bemærkning

1. Sætningen/kommandoen er identisk med sætningen/kommandoen SELECT OUTPUT(10.81).

10.61 PREFIX 10.61

COMAL80 sætning/kommando

Format

PREFIX <sudtr>

<sudtr>: et vilkårligt strengudtryk.

Anvendelse

Sætningen/kommandoen bruges til at definere en tegnfølge, som automatisk sættes foran de filnavne, der angives i sætninger og kommandoer.

Bemærkninger

1. <sudtr> kan være en vilkårlig tegnfølge, men normalt angives "1/" eller "2/" da man derved ikke behøver at angive unitnummeret i filnavne.

2. Ved opstart sættes præfixet til "1/".

3. Ønsker man ikke at benytte præfixet i et filnavn, skal filnavnet blot indledes med "/".

COMAL80 sætning/kommando

Format

```
                  <nudtr>                    <nudtr>
PRINT  { <sudtr>  }[ {,} { <sudtr>  } ]... { , }
       { <printfkt> } {;} { <printfkt> }     { ; }
```

<printfkt>: TAB- eller AT-funktionen
<nudtr>: et numerisk eller logisk udtryk
<sudtr>: en strengkonstant eller strengvariabel.

Anvendelse

Sætningen/kommandoen anvendes til udskrivning af resultater, tekster m.v.

Bemærkninger

1. Udskrift af <nudtr>
   Numerisk udtryk (heltal, decimale eller E-notation) udskrives med følgende format.
   <fortegn><tal>
   Fortegnet er enten minus (-) eller tomt (altså intet mellemrum)

2. Udskrift af <sudtr>
   Strengudtryk udskrives uden indledende og efterfølgende mellemrum.

3. Udskrift af <printfkt>
   <printfkt> udføres. Hvis TAB-argumentet er mindre end nuværende print position ignoreres funktionen (se desuden AT (10.3), TAB(10.89)).

4. Anvendelse af ,

Udskriftslinien er inddelt i udskriftssøjler. Den første udskriftssøjle starter i position 0, den næste i positionen, der er angivet i en ZONE-sætning. Før hvert PRINT-argument udskrives sammenlignes dets længde med den resterende plads på linien. Er der ikke nok plads, udføres en vognretur, og udskriften fortsætter i første udskriftssøjle.

5. Anvendelse af ;

Angives semikolon (;) efter <nudtr> udskrives et mellemrum (et blankt tegn). Angives semikolon efter <sudtr> eller <printfkt>, udskrives intet.

10.63 PRINT FILE 10.63

COMAL80 sætning/kommando

Format

PRINT FILE <strømnr>: { <nudtr> / <sudtr> / <printfkt> } [{ , / ; } { <nudtr> / <sudtr> / <printfkt> }] ... { , / ; }

<strømnr>: et numerisk udtryk, hvis værdi angiver nummeret på en åben datastrøm

<printfkt>: TAB-funktion

<nudtr>: et numerisk eller logisk udtryk

<sudtr>: en strengkonstant eller strengvariabel.

Anvendelse

Sætningen/kommandoen benyttes til at skrive data i ASCII format i en datastrøm.

Bemærkninger

1. PRINT FILE-sætningen anvendes til at udskrive data til en ASCII enhed, (console, printer) eller til en diskettefil.

2. Datastrømmen skal være åbnet (OPEN) i WRITE-mode.

3. Udskriftsreglerne er i øvrigt de samme som for PRINT (10.62).

10.64 PRINT FILE USING 10.64

COMAL80 sætning/kommando

Format

PRINT FILE <strømnr>: USING <sudtr>: <nudtr> [,<nudtr>]... [;]

<strømnr>: et numerisk udtryk, hvis værdi angiver nummeret på en åben datastrøm.
<sudtr>: streng udtryk, format streng
<nudtr>: numerisk udtryk

Anvendelse

Sætningen/kommandoen anvendes til at udskrive værdier i en datastrøm i et bestemt format.

Bemærkninger

1. Se bemærkningerne til PRINT FILE (10.63) og PRINT USING (10.65).

10.65 PRINT USING 10.65

COMAL80 sætning/kommando

Format

PRINT USING <sudtr>: <nudtr> [,<nudtr>]... [;]

<sudtr>: streng udtryk, format streng.
<nudtr>: numerisk udtryk

Anvendelse

Sætningen/kommandoen anvendes til at udskrive værdier i et bestemt format.

Bemærkninger

1. <sudtr> må være et vilkårligt strengudtryk, der udskrives direkte, idet # og . ændres efter følgende regler:
   NB: Δ angiver mellemrum i udskriftsformatet i det følgende.

   a Heltalsudskrift

   Hvert # i <sudtr> afsætter plads til et ciffer (0...9) eller et negativt fortegn (-):

<sudtr>	<nudtr>	Udskrift	Kommentarer
####	50	ΔΔ50	Tal bliver højrestillet indenfor formattet, med indledende blanke tegn.
####	-37	Δ-37	Fortegnet udskrives umiddelbart før cifrene.
####	1.52	ΔΔΔ2	Decimaltal afrundes.
####	2750	2750	Positivt fortegn optager ikke plads.
####	-4096	****	Hvis <nudtr> kræver flere positioner end angivet udskrives stjerner.

b Decimaltalsudskrift

Decimalpunktummet (.) udskriver et punktum på en fast plads i udskriftsformatet. Der skal altid angives # på begge sider af punktummet. Hvis <nudtr> indeholder flere decimaler end angivet, afrundes. Hvis <nudtr> indeholder færre decimaler end angivet, fyldes op med nuller. For heltalsdelen følges reglerne i pkt. a.

<sudtr>	<nudtr>	Udskrift	Kommentarer
###.##	50	Δ50.00	Decimalerne udskrives altid.
###.##	3.985	ΔΔ3.99	Decimaler afrundes.
###.##	4096	******	Hvis <nudtr> har for mange cifre til venstre for decimal-punktummet, erstattes hele formatstrengen af stjerner.

2. Antallet af felter i formatstrengen skal svare til antallet af argumenter i argumentlisten.

Eksempel

```
0010 DIM format$ OF 20
0020 format$:="##    #."
0030 pi:= ATN(1)*4
0040 PRINT "DEC. PI"
0050 FOR decimaler:= 1 TO 11 DO
0060    format$:= format$+"#"
0070    PRINT USING format$:decimaler,pi
0080 NEXT decimaler
0090 END

RUN
DEC. PI
 1    3.1
 2    3.14
 3    3.142
 4    3.1416
 5    3.14159
 6    3.141593
 7    3.1415927
 8    3.14159265
 9    3.141592654
10    3.1415926536
11    3.14159265359
END
at 0090
```

10.66 PROC-ENDPROC 10.66

COMAL80 struktur

Format

```
PROC <navn> [( <par>  [,<par>]... )]  [CLOSED]
  [GLOBAL <symbol> [,<symbol>]...]
  <sætningsliste>
ENDPROC <navn>
```

<par>: { [REF] <nvar> | [REF] <svar> | REF <ntabl>([,]) | REF <stabl>() }

<navn>: Navn på proceduren (max. 16 tegn)
<nvar>: Vilkårligt numerisk variabel navn
<svar>: Vilkårligt strengvariabelnavn
<ntabl>: Vilkårligt vektor- eller matrix-navn
<stabl>: Vilkårligt teksttabelnavn
<symbol>: Navn på en global procedure, variabel, strengvariabel, vektor, matrix eller teksttabel.
<sætningsliste>: COMAL80 sætninger.

Anvendelse

Konstruktionen benyttes til at definere en procedure.

Bemærkninger

1. <navn> skal være forskellig fra elle andre navne på funktioner, etiketter, reelle og strengvariable.

2. En procedure kan gøres lukket (CLOSED) så variable, der optræder i proceduren, er lokale og ikke berører variablerne i resten af programmet (de globale variable). Ønsker man at benytte en variabel, der optræder i resten af programmet, benyttes GLOBAL-sætningen. De lokale variable, der optræder i proceduren, slettes, når man forlader proceduren.

3. Anføres REF foran variabelnavnet i parameterlisten (de formelle parametre), vil den tilsvarende parameter i EXEC-sætningen (den reelle parameter) blive ændret, samtidig med at den formelle parameter ændres nede i proceduren. På denne måde kan man returnere resultater via parametrene.

Eksempel

```
0010 PROC flyt(n,tårn1,tårn2,tårn3)
0020   IF n>1 THEN EXEC flyt(n-1,tårn1,tårn3,tårn2)
0030   EXEC vis(n,tårn1,tårn2)
0040   IF n>1 THEN EXEC flyt(n-1,tårn3,tårn2,tårn1)
0050 ENDPROC flyt
0060
0070 PROC init
0080   MARGIN 0
0090   PRINT CHR$(12);AT(1,2);CHR$(132)
0100   DIM højde(3),pos(3)
0110   højde(1):=antal; højde(2):=0; højde(3):=0
0120   pos(1):=5; pos(2):=30; pos(3):=55
0130   DIM skive$(antal) OF 22,tom$ OF 22
0140   tom$:="                      "
0150   skive$(1):="          "+CHR$(127)+CHR$(127)+"          "
0160   FOR i:=2 TO antal DO
0170     skive$(i):=skive$(i-1,2:11)+CHR$(127)
0180     skive$(i,12:22):=CHR$(127)+skive$(i-1,12:21)
0190   NEXT i
0200   FOR y:=20 TO 21-antal STEP -1 DO
0210     PRINT AT(pos(1),y);skive$(antal+y-20);
0220   NEXT y
0230 ENDPROC init
0240
0250 PROC vis(n,fra,til)
0260   x:=pos(fra)
0270   FOR y:=20-højde(fra) TO 20-antal STEP -1 DO
0280     PRINT AT(x,y);skive$(n);AT(x,y+1);tom$;
0290   NEXT y
```

```
0300    retning:=SGN(til-fra)
0310    WHILE x<>pos(til) DO
0320      PRINT AT(x,20-antal);skive$(n);
0330      x:=x+retning
0340    ENDWHILE
0350    FOR y:=20-antal TO 20-højde(til) DO
0360      PRINT AT(x,y);skive$(n);AT(x,y-1);tom$;
0370    NEXT y
0380    højde(fra):=højde(fra)-1; højde(til):=højde(til)+1
0390  ENDPROC vis
0400
0410  PRINT "Tårnene i HANOI"
0420  PRINT
0430  INPUT "Antal skiver (ml. 1 og 10) ":antal
0440  EXEC init
0450  EXEC flyt(antal,1,2,3)
0460  PRINT AT(1,21);
```

10.67 PROC-EXTERNAL 10.67

COMAL80 sætning

Format

PROC <navn> [(<par> [,<par>]...)] EXTERNAL <filnavn>

<par>: { [REF] <nvar> | [REF] <svar> | REF <ntabl>([,]) | REF <stabl>() }

<navn>: Navn på proceduren (max. 16 tegn)
<nvar>: Vilkårligt numerisk variabelnavn
<svar>: Vilkårligt strengvariabel navn
<ntabl>: Vilkårligt vektor- eller matrix-navn
<stabl>: Vilkårligt teksttabelnavn
<filnavn>: Navn på diskettefil, hvori den externe procedure er gemt med kommandoen SAVE.

Anvendelse

Sætningen benyttes til at erklære en extern procedure.

Bemærkninger

1. Den externe procedure skal være SAVE't i <filnavn>, og første linie i dette program skal være et procedurehoved for en lukket (CLOSED) procedure. Den lukkede procedure må ikke indeholde GLOBAL sætninger.

2. Parameterbeskrivelsen i PROC-EXTERNAL skal svare nøjagtig til parameter-beskrivelsen i procedurehovedet i den SAVE'ede procedure.

3. Senere i programmet kan man referere til den externe procedure, som man refererer til en almindelig PROC-ENDPROC. Ved hver EXEC af den externe procedure bliver den indlæst fra disketten.

4. Den externe procedure kan i sig selv indeholde andre procedurer og funktioner, både åbne, lukkede og externe.

5. Hvis en extern procedure kalder sig selv, indlæses den ikke igen.

6. Stopper programudførslen under udførelsen af en extern procedure, er det kun denne man kan se med LIST-kommandoen. SAVE-kommandoen gemmer også kun den externe procedure hvorimod RUN genstarter hovedprogrammet. NEW sletter både hovedprogram og den externe procedure.

7. Externe procedurer gør det muligt at oprette biblioteker af procedurer, der kan benyttes af flere forskellige programmer.

10.68 PROC-HANDLER 10.68

COMAL80 struktur

Format

```
PROC <navn> HANDLER
  <sætninger>
ENDPROC <navn>
```

<navn>: navnet på handleren (max. 16 tegn).
<sætninger>: COMAL80 sætninger.

Anvendelse

En PROC-HANDLER er en fejlhåndteringsprocedure, der kaldes automatisk i tilfælde af fejl.

Bemærkninger

1. En PROC-HANDLER kan ikke kaldes med EXEC eller som funktion.

2. En PROC-HANDLER aktiveres med sætningen ENABLE (se afsnit 10.23).

3. Man kan returnere på 3 forskellige måder fra en PROC-HANDLER:

   a. ENDPROC sætningen nås, hvilket bevirker, at programudførslen standses med en sædvanlig fejludskrift.

   b. Hvis en CONTINUE-sætning udføres, vil programudførslen fortsætte med sætningen efter den sætning, der forårsagede fejlen. (Se afsnit 10.11).

   c. Hvis en RETRY-sætning udføres, vil programudførslen fortsætte med den sætning, der forårsagede fejlen (se afsnit 10.76).

4. Nummeret på den fejl, der bevirkede, at PROC-HANDLER blev kaldt, er værdien af SYS(0) eller ERR.

5. Et tryk på ESC-tasten bevirker kald af PROC-HANDLER, og stopper således ikke programudførslen.

10.69 RANDOMIZE 10.69

COMAL80 sætning

Format

RANDOMIZE

Anvendelse

Sætningen får "tilfældigtal-generatoren" til at starte på et nyt sted i følgen af tilfældige tal.

Bemærkninger

1. RND funktionen genererer normalt den samme følge af tilfældige tal, startende med den samme værdi. Dette er praktisk under indkøringen af programmer. Når programmet er testet er det derimod praktisk at starte et nyt sted i følgen. For at opnå dette, skal brugeren angive RANDOMIZE før første kald af RND.

2. RANDOMIZE udregner det nye RND-tal på grundlag af en intern tæller i forbindelse med skærmstyringen.

Eksempel

```
0010 RANDOMIZE
0020 WHILE TRUE DO
0030    FOR i:= 1 TO 3 DO PRINT RND
0040    STOP
0050 ENDWHILE

RUN                                  Både RUN og CON starter et
0.9726401133793                      nyt sted i følgen.
0.7816163391192
0.3161324487011

STOP
at 0040
RUN
0.9416661551832
0.6788533738851
0.377864580389
```

```
STOP
at 0040

CON
0.814314058549
0.7384990342428
0.1156652263127

STOP
at 0040
```

COMAL80 sætning

Format

READ { <nvar> } [, { <nvar> }] ...
     { <svar> }   { <svar> }

<nvar>: en numerisk variabel

<svar>: en streng variabel

Anvendelse

Sætningen læser værdier fra DATA-sætninger og tildeler værdierne til variablene angivet i READ sætningerne.

Bemærkninger

1. READ bruges altid i forbindelse med DATA-sætninger

2. Rækkefølgen af de forskellige typer af parametre i READ-sætningen skal svare til rækkefølgen af værdierne i DATA-sætningerne

3. For hver gang READ læser et dataelement, flyttes datapegepinden til det næste element i listen af DATA-sætninger.

4. Er typen af READ-variablen (numerisk eller streng) ikke den samme som typen af det tilhørende DATA-element udskrives fejlmeddelelsen 0109: TYPE KONFLIKT.

5. Hvis man med READ sætningen forsøger at læse flere data end antallet af elementer i DATA sætningerne udskrives fejlmeddelelsen 0117: IKKE FLERE DATA.

6. Man kan flytte "rundt på" datapegepinden ved hjælp af RESTORE.

7. EOD-funktionen bliver sand samtidig med, at det sidste element i DATA-sætningerne bliver læst.

10.71 READ FILE 10.71

COMAL80 sætning/kommando

Format

READ FILE <strømnr> [,<recnr>]: { <nvar> / <svar> } [, { <nvar> / <svar> }] ...

<strømnr>: Et taludtryk, hvis værdi angiver nummeret på en åben datastrøm.

<recnr>: Et taludtryk, der angiver et postnummer (kun strømme åbnet i RANDOM mode).

<nvar>: En numerisk variabel

<svar>: En strengvariabel.

Anvendelse

Sætningen/kommandoen indlæser data i binært format fra en datastrøm, der er åbnet (OPENed) i READ- eller RANDOM-mode.

Bemærkninger

1. Hver variabel i argumentlisten, skal have samme type (numeriske eller strenge) som dataene i datastrømmene.

2. Da sætningen indlæser binært format, anvendes sætningen til at læse data, der er skrevet med WRITE FILE-sætningen.

3. <recnr> kan angives hvis filen er åbnet i RANDOM-mode.

4. EOF-funktionen kan anvendes til at bestemme hvornår det sidste dataelement er læst.

Eksempel

```
0010 PROC matreadfile(strøm,REF mat(,),dim1,dim2) CLOSED
0020   FOR i:=1 TO dim1 DO
0030     FOR j:=1 TO dim2 DO READ FILE strøm: mat(i,j)
0040   NEXT i
0050 ENDPROC matreadfile
0060
0070 PROC matreadrndfile(strøm,post,REF mat(,),dim1,dim2) CLOSED
0080   READ FILE strøm,post: mat(1,1)
0090   FOR j:=2 TO dim2 DO READ FILE strøm:mat(1,j)
0100   FOR i:=2 TO dim1 DO
0110     FOR j:=1 TO dim2 DO READ FILE strøm: mat(i,j)
0120   NEXT i
0130 ENDPROC matreadrndfile
```

COMAL80 sætning/kommando

Format

RENAME <filnavn1>, <filnavn2>

<filnavn1>: Navnet på en diskettefil, der skal omdøbes.
<filnavn2>: Det nye navn til <filnavn1>.

Anvendelse

Sætningen/kommandoen anvendes til at omdøbe en diskettefil.

Bemærkning

1. Både <filnavn1> og <filnavn2> er strengudtryk.

2. <filnavn2> må ikke indeholde unitnr.

10.73 RENUMBER 10.73

COMAL80 kommando

Format

```
                <lnr> [,]
RENUMBER[{  , <lnrspring>   }]
         {<lnr>,<lnrspring> }
```

<lnr>: Første linienummer efter omnummereringen.

<lnrspring>: Forskellen mellem linienumrene efter omnummereringen.

Anvendelse

Kommandoen anvendes til at omnummerere linienumrene i et program.

Bemærkning

1. Kombinationerne af parametrene til RENUMBER har følgende betydning:

RENUMBER	Programlinierne bliver nummereret så de starter med 0010 og har spring på 0010 mellem linierne.
RENUMBER <lnr>	Programlinierne bliver nummereret så de starter med <lnr> og har spring på <lnr> mellem linierne.
RENUMBER <lnr>,	Programlinierne bliver nummereret så de starter med <lnr> og har spring på 0010 mellem linierne.
RENUMBER ,<lnrspring>	Programlinierne bliver nummereret så de starter med 0010 og har spring på <lnrspring> mellem linierne.

RENUMBER <lnr>,<lnrspring>	Programlinierne bliver nummereret så de starter med <lnr> og har spring på <lnrspring> mellem linierne.

Eksempel

Vi forudsætter i det følgende, at programlageret indeholder følgende program:

```
0001 print
0007 print
0012 print
0100 print
```

Da vil

RENUMBER	omnummerere linierne til 0010, 0020, 0030, 0040
RENUMBER 20	omnummerere linierne til 0020, 0040, 0060, 0080
RENUMBER 20,	omnummerere linierne til 0020, 0030, 0040, 0050
RENUMBER ,50	omnummerere linierne til 0010, 0060, 0110, 0160
RENUMBER 100,20	omnummerere linierne til 0100, 0120, 0140, 0160

10.74 REPEAT - UNTIL 10.74

COMAL80 struktur

Format

REPEAT
   <sætningsliste>
UNTIL <logisk udtryk>

<sætningsliste>: COMAL80 sætninger
<logisk udtryk>: et udtryk, der kan have værdien sand (<>0) eller falsk (=0)

Anvendelse

Konstruktionen benyttes til at gentage udførelsen af en blok sætninger indtil et udsagn er sandt.

Virkemåde

1. <sætningsliste> udføres

2. <logisk udtryk> udregnes

3. Hvis værdien af <logisk udtryk> er falsk, hoppes tilbage til trin 1.

4. Hvis værdien af <logisk udtryk> er sand, fortsætter programmet med den sætning, der følger efter UNTIL-sætningen.

Bemærkninger

1. Strukturen kan ikke udføres som kommando.

2. En REPEAT-UNTIL løkke må gerne indeholde andre REPEAT-UNTIL løkker.

3. Hvis man hopper ind i en REPEAT-UNTIL løkke udenom REPEAT sætningen fås fejludskriften 0116: ULOVLIGT HOP.

4. NB! <sætningsliste> udføres altid mindst en gang.

Eksempel 1

```
0010 MARGIN 80
0020 ZONE 4
0030 i:= 1
0040 REPEAT
0050   PRINT i,
0060   i:= i+i
0070 UNTIL i>10
0080 PRINT
0090 PRINT "Efter UNTIL er i=";i
0100 END

RUN
1   2   3   4   5   6   7   8   9   10
Efter UNTIL er i=11
END
at 0100
```

Eksempel 2

```
0010 MARGIN 80
0020 ZONE 4
0030 i:= 20
0040 REPEAT
0050   PRINT "Bliver ALTID udført mindst 1 gang"
0060   i:= i-1
0070   PRINT i,
0080 UNTIL i>10
0090 PRINT
0100 PRINT "Efter UNTIL er i=";i
0110 END

RUN
Bliver ALTID udført mindst 1 gang
19
Efter UNTIL er i=19
END
at 0110
```

Kommentar

Sætningerne mellem REPEAT og UNTIL udføres altid mindst en gang

Eksempel 3

```
0010 PRINT "Dette program udskriver et blokdiagram på skærmen."
0020 REPEAT
0030   INPUT "Antal søjler (max 12):": antal
0040 UNTIL antal>0 AND antal <=12 AND INT(antal)=antal
0050 DIM søjler(antal),blok$ OF 7,grafik$ OF 1,normal$ OF 1,box$ OF 1
0051 grafik$:= CHR$(132)
0052 normal$:= CHR$(128)
0053 box$:= CHR$(127)
0060 blok$:= grafik$+box$+box$+box$+box$+normal$
0070 FOR i:= 1 TO antal DO INPUT "Værdi (ml. 0 og 20):": søjler(i)
0080
0090 PRINT CHR$(12);AT(1,22);normal$
0100 FOR i:= 1 TO 79 DO PRINT "-";// vandret streg
0110 FOR i:= 1 TO antal DO PRINT AT(i*6+2,23);søjler(i);
0120 værdi:= 1
0130 REPEAT
0140   skrevetsøjle:= FALSE
0150   FOR i:= 1 TO antal DO
0160     IF søjler(i)>=værdi THEN
0170       skrevetsøjle:= TRUE
0180       PRINT AT(i*6,22-værdi);blok$
0190     ENDIF
0200   NEXT i
0210   værdi:= værdi+1
0220 UNTIL NOT skrevetsøjle OR værdi>20
0230 PRINT AT(1,22)
0240 END
```

Eksempel 4

```
0010 antbyer:= 8
0020 DIM afstand(antbyer,antbyer)
0030 DIM by$ OF 10,bynavn$ OF 10,svar$ OF 3
0040 PRINT "Programmet finder afstande mellem ";
0050 PRINT "de 8 største byer på Fyn"
0060 PRINT
0070 RESTORE afstandstabel
0080 FOR i:= 1 TO antbyer DO
0090   FOR j:= 1 TO i DO
0100     READ afstand(i,j)
0110     afstand(j,i):= afstand(i,j)
0120   NEXT j
0130 NEXT i
0140
0150 REPEAT
0160   REPEAT
0170     INPUT "Hvorfra ": bynavn$
0180     EXEC findby
0190   UNTIL byfundet
0200   fraby:= bynr
0210
0220   REPEAT
0230     INPUT "Hvortil ": bynavn$
0240     EXEC findby
0250   UNTIL byfundet
0260   tilby:= bynr
0270
0280   PRINT "Afstanden er ";afstand(fraby,tilby);" km."
0290   PRINT
0300   REPEAT
0310     INPUT "Vil du prøve igen ?": svar$
0320     IF svar$<>"ja" and svar$<>"nej" THEN
0330       PRINT "Svar venligst ja eller nej"
0340     ENDIF
0350   UNTIL svar$="ja" OR svar$="nej"
0360
0370 UNTIL svar$="nej"// hovedløkken gennemløbes til svar er nej
0380 END
```

```
0390
0400 PROC findby
0410   RESTORE byer
0420   bynr:= 0
0430   REPEAT
0440     bynr:= bynr+1
0450     READ by$
0460   UNTIL bynr=antbyer OR by$=bynavn$
0470   byfundet:= (bynavn$=by$)// logisk variabel
0480   IF NOT byfundet THEN
0490     PRINT "Den by kender jeg ikke, jeg kender kun:"
0500     RESTORE byer
0510     FOR bynr:= 1 TO antbyer DO
0520       READ by$
0530       PRINT by$
0540     NEXT bynr
0550   ENDIF
0560 ENDPROC findby
0570
0580 byer:
0590 DATA "assens", "bogense","fåborg","kerteminde"
0600 DATA "middelfart","nyborg","odense","svendborg"
0610
0620 afstandstabel:
0630 DATA 0
0640 DATA 42,0
0650 DATA 36,63,0
0660 DATA 59,51,63,0
0670 DATA 34,30,68,67,0
0680 DATA 67,59,47,19,75,0
0690 DATA 39,30,38,22,45,29,0
0700 DATA 62,72,26,51,86,36,42,0
```

```
RUN
Programmet finder afstande mellem de 8 største byer på Fyn

Hvorfra odense
Hvortil assens
Afstanden er 39 km.

Vil du prøve igen ?ja
Hvorfra rynkeby
Den by kender jeg ikke, jeg kender kun:
assens
bogense
fåborg
kerteminde
middelfart
nyborg
odense
svendborg
Hvorfra kerteminde
Hvortil middelfart
Afstanden er 67 km.

Vil du prøve igen ?nej
END
at 0380
```

10.75 RESTORE 10.75

COMAL80 sætning

Format

RESTORE [<navn>]

<navn>: navnet på en etikette i programmet.

Anvendelse

Sætningen bruges til at flytte data pegepinden til enten den første DATA sætning eller til den første DATA sætning, der står efter den angivne etikette.

Bemærkninger

1. Hvis RESTORE sætningen bruges uden angivelse af <navn>, flyttes data pegepinden til det første dataelement i den første DATA-sætning.

2. Hvis RESTORE sætningen indeholder angivelse af <navn>, vil datapegepinden blive flyttet til det første dataelement i den DATA-sætning, der står lige efter etiketten <navn>.

Eksempel

```
0010 READ held
0020
0030 RESTORE navn
0040 READ antalnavne
0050 DIM navne$(antalnavne) OF 30
0060 FOR i:= 1 TO antalnavne DO READ navne$(i)
0070
0080 RESTORE postnumre
0090 READ antalpostnr
0100 DIM postnr(antalpostnr)
0110 FOR i:= 1 TO antalpostnr DO READ postnr(i)
0120
0130 DATA 7,9,13
```

```
0140
0150 postnumre:
0160 DATA 4,2750,2770,2830,2000// 4 er antallet der læses i 0090
0170
0180 navn:
0190 DATA 3, "peter","jens","søren"// 3 er antallet der læses i 0040
0200
0210 FOR i:= 1 TO antalnavne DO PRINT navne$(i)
0220 PRINT
0230 FOR i:= 1 TO antalpostnr DO PRINT postnr(i)
0240 END

RUN
peter
jens
søren

2750
2770
2830
2000
```

10.76 RETRY 10.76

COMAL80 sætning

Format

RETRY

Anvendelse

Sætningen anvendes kun i en HANDLER. Sætningen bevirker, at programudførslen fortsætter med den sætning, der forårsagede kaldet af PROC-HANDLER.

Eksempel

```
0010 PROC checkdiskette HANDLER
0020   IF SYS(0)=214 THEN
0030     PRINT "*** Fil eksisterer ikke"
0040     PRINT "*** Indsæt korrekt diskette i "
0050     INPUT "*** disketteenhed nummer 1 og tast vognretur": S$
0060     MOUNT 1
0070     RETRY
0080   ENDIF
0090 ENDPROC checkdiskette
0100 DIM S$ OF 1
0110 ENABLE checkdiskette
0120 OPEN FILE 1,"datafil",READ
0130 DISABLE
0140 CLOSE
0150 END
```

10.77 RETURN 10.77

COMAL80 sætning

Format

RETURN [<nudtr>]

<nudtr>: et vilkårligt numerisk udtryk.

Anvendelse

Sætningen anvendes til at returnere fra en procedure eller en funktion. Hvis RETURN optræder i en FUNC-ENDFUNC vil <nudtr> blive funktionens værdi.

Bemærkninger

1. PROC-ENDPROC konstruktionen behøver ikke at indeholde en RETURN-sætning, da proceduren altid vil returnere ved ENDPROC.

2. Hvis systemet under programudførelsen af FUNC-ENDFUNC når frem til ENDFUNC udenom en RETURN-sætning fås fejl 0113: FUNKTIONSVÆRDI UDEFINERET.

Eksempel

```
0010 FUNC lige(n)
0020   RETURN (n+1) MOD 2 // 1:sand  0:falsk
0030 ENDFUNC lige
0040
0050 REPEAT
0060   INPUT "Indtast et tal mellem -32768 og 32767 (0 stopper):":tal
0070   PRINT tal;" er ";
0080   IF NOT lige(tal) then print "u";
0090   PRINT "lige"
0100 UNTIL tal=0
0110 END
```

```
RUN
Indtast et tal (0 stopper): 7
7 er ulige
Indtast et tal (0 stopper): -4
-4 er lige
Indtast et tal (0 stopper): 0
0 er lige
END
at 0110
```

10.78 RND 10.78

COMAL80 funktion

Format

RND [(<nudtr1>,<nudtr2>)]

<nudtr1> : et vilkårligt numerisk udtryk
<nudtr2> : et vilkårligt numerisk udtryk

Anvendelse

RND er en numerisk funktion, der genererer et tilfældigt tal.

Bemærkninger

1. De genererede tal er pseudo-tilfældige, idet det næste tal beregnes ud fra det forrige.

2. Angives <nudtr1> og <nudtr2> vil RND generere et heltal i intervallet fra <nudtr1> til <nudtr2>. Udelades <nudtr1> og <nudtr2> genereres et decimaltal mellem 0 og 1.

3. RND genererer altid den samme følge af tilfældige tal. Ved NEW, RUN og systemopstart gentages følgen med de samme værdier.

4. RANDOMIZE sætningen bevirker, at der startes på et vilkårligt sted i følgen af tilfældige tal.

Eksempel 1

```
0010 WHILE TRUE DO
0020   FOR i:= 1 TO 3 DO PRINT RND
0030   STOP
0040 ENDWHILE
```

Kommentar

```
RUN
0.1116003194179
0.8394904533458
0.8203299841557
STOP
at 0030
RUN
0.1116003194179
0.8394904533458
0.8203299841557
STOP
at 0030
CON
0.0370770494076
0.6477562514615
0.8871226414774
STOP
at 0030
```

RUN kommandoen repeterer de samme tal påny, CON fortsætter følgen.

Eksempel 2

```
0010 RANDOMIZE
0020 tal:= RND(0,99)  // tilfældigt tal mellem 0 og 99 (incl)."
0030 forsøg:= 0// antal forsøg
0040 PRINT "Gæt et tal mellem 0 og 99 (incl).
0050 REPEAT
0060   INPUT "Indtast dit gæt >": gæt;
0070   forsøg:= forsøg+1
0080   CASE TRUE OF
0090   WHEN gæt<tal
0100     PRINT " For lavt !"
0110   WHEN gæt>tal
0120     PRINT " For højt !"
0130   WHEN gæt=tal
0140     PRINT " TILLYKKE! du fandt tallet på ";forsøg;". forsøg."
0150   ENDCASE
0160 UNTIL gæt=tal
0170 END
```

10.79 RUN 10.79

COMAL80 kommando

Format

RUN [<filnavn>]

<filnavn>: navnet på en diskettefil angivet i anførselstegn.

Anvendelse

Kommandoen starter udførslen af det program, der ligger i programlageret, eller den LOAD'er og udfører et program, der er SAVE'd på en diskettefil.

Bemærkninger

1. Datalageret slettes når RUN kommandoen afgives.

2. Programudførslen starter i programlinien med det laveste linienummer.

3. Programnavnet angivet med <filnavn> skal være gemt med kommandoen SAVE.

4. Ønskes udskriften standset midlertidigt, trykkes på mellemrumstasten. Ved næsten tryk startes den atter.

10.80 SAVE 10.80

COMAL80 kommando

Format

SAVE [<filnavn>]

<filnavn>: navnet på en diskettefil angivet i anførselstegn.

Anvendelse

Kommandoen bruges til at gemme det program, der ligger i programlageret på en diskette. Programmet kan senere indlæses med LOAD.

Bemærkninger

1. Programmet gemmes i binært format.

2. Hvis <filnavn> eksisterer på disketten udskrives fejl 0213: FIL EKSISTERER ALLEREDE.

3. Udelades <filnavn> vil programmet blive gemt i filen, hvis navn er angivet i statuslinie. Hvis filen allerede eksisterer, slettes den først, hvorefter det nye program SAVEs.

4. Efter SAVE vil statuslinien indeholde navnet på den fil, der er SAVEd.

Eksempel

```
SAVE "/1/MITPROGRAM"
```

10.81 SELECT OUTPUT 10.81

COMAL80 sætning/kommando

Format

SELECT OUTPUT <filnavn>

<filnavn>: navnet på en datastrøm.

Anvendelse

Sætningen/kommandoen anvendes til at vælge udskriftsdatastrøm.

Bemærkning

1. Al udskrift, der normalt fremkommer på skærmen (fra PRINT, DIR etc) vil blive udskrevet på <filnavn>. Dog udskrives INPUT-tekst og fejlmeddelser stadig på skærmen.

2. Hvis <filnavn> er navnet på en diskettefil, vil den automatisk blive oprettet.

3. Når RUN afbrydes eller afsluttes, sættes SELECT OUTPUT tilbage til datastrømmen "/1/console".

Eksempel

```
0010 DIM svar$ OF 1
0020 REPEAT
0030   INPUT "Ønskes udskrift på printer eller skærm (p/s) ":svar$
0040 UNTIL svar$ IN "pPsS"
0050 IF svar$ IN "pP" THEN SELECT OUTPUT "printer"
0060 ...
1000 SELECT OUTPUT "console" // Kan ikke gøres for mange gange
1010 END
```

COMAL80 funktion

Format

SGN (<nudtr>)

<nudtr>: et numerisk udtryk

Anvendelse

Funktionen har værdien +1 hvis <nudtr> er større end nul, 0 hvis <nudtr> er lig nul og -1 hvis <nudtr> er mindre end nul.

Eksempel

```
0010 WHILE NOT EOD DO
0020   READ tal
0030   PRINT TAL;" er ";
0040   CASE SGN(tal) OF
0050   WHEN -1
0060     PRINT "negativ"
0070   WHEN 0
0080     PRINT "nul"
0090   WHEN 1
0100     PRINT "positiv"
0110   ENDCASE
0120 ENDWHILE
0130
0140 DATA 6,-6,0
```

```
6 er positiv
-6 er negativ
0 er nul
END
at 0140
```

10.83 SIN 10.83

COMAL80 funktion

Format

SIN (<nudtr>)

<nudtr>: et numerisk udtryk, der angiver et radianantal.

Anvendelse

Funktionen udregner sinus til en vinkel udtrykt i radianer.

Eksempel

```
0010 FUNC rad(vinkel)// procedure omformer vinkel -> radianer
0020   return vinkel*ATN(1)/45//    atn(1)=pi/4
0030 ENDFUNC rad
0040
0050 // lav en tabel over sinus
0060
0070 MARGIN 80
0080 ZONE 20
0090 PRINT "vinkel","radianer","sinus"
0100 FOR v:= 0 TO 90 STEP 5 DO PRINT v,rad(v),SIN(rad(v))
0110 END
```

```
RUN
vinkel     radianer                 sinus
0          0                        0
5          8.726646259971E-002      8.715574274761E-002
10         0.1745329251994          0.1736481776669
15         0.2617993877991          0.2588190451025
20         0.3490658503986          0.3420201433254
25         0.4363323129984          0.4226182617404
30         0.5235987755982          0.5
35         0.6108652381977          0.5735764363506
40         0.6981317007975          0.6427876096864
45         0.7853981633973          0.7071067811865
50         0.8726646259971          0.7660444431189
55         0.9599310885966          0.8191520442895
60         1.047197551196           0.8660254037844
65         1.134464013796           0.9063077870373
70         1.221730476395           0.9396926207864
75         1.308996938995           0.9659258262897
80         1.396263401595           0.9848077530132
85         1.483529864194           0.9961946980927
90         1.570796326794           1
END
at 0110
```

10.84 SIZE 10.84

COMAL80 kommando

Format

SIZE

Anvendelse

Angiver fordelingen af frit lager og lager benyttet til henholdsvis program og data.

Bemærkning

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

Eksempel

```
SIZE
program  data     free
00178    00012    18242
```

10.85 SQR 10.85

COMAL80 funktion

Format

SQR (<nudtr>)

<nudtr>: et ikke-negativt udtryk.

Anvendelse

Funktionen uddrager kvadratroden af <nudtr>.

Bemærkning

1. Forsøges kvadratroden uddraget af et negativt tal, fås fejludskriften 103: KVADRATROD AF NEGATIVT TAL.

Eksempel

```
0010 ZONE 16
0020 PRINT SQR(25),SQR(25.1)
0030 END
0040
RUN
5                    5.00999001995
END
at 0030
```

COMAL80 sætning

Format

STOP

Anvendelse

Sætningen standser udførslen af et program

Bemærkninger

1. Sætningen kan ikke udføres som kommando.

2. STOP-sætninger må placeres et vilkårligt sted i programmet. Når en af disse mødes udskrives:

   STOP
   at XXXX

   hvor XXXX er linienummeret på STOP-sætningen.

3. Når STOP-sætningen er udført kan brugeren fortsætte programudførslen med CON-kommandoen.

10.87 STR$ 10.87

COMAL80 funktion

Format

STR$ (<nudtr>)

<nudtr>: Vilkårligt numerisk udtryk

Anvendelse

Funktionen konverterer et numerisk udtryk til en streng.

Bemærkninger

1. Funktionen omformer f.eks. tallet 1047 til strengen "1047".

2. Funktionen udfører det omvendte af VAL.

Eksempel 1

```
0010 FUNC cprnummerok(cprnr) CLOSED
0020   DIM tal$ OF 10
0030   tal$:=STR$(cprnr)
0040   WHILE len(tal$)<10 DO tal$:="0"+tal$
0050   sum:=0
0060   for i:=1 TO 7 DO sum:=sum+VAL(tal$(11-i:11-i))*i
0070   for i:=2 TO 4 DO sum:=sum+VAL(tal$(5-i:5-i))*i
0080   RETURN (sum MOD 11 = 0)
0090 ENDFUNC
0100 INPUT "Indtast et cprnummer :": nr
0110 IF cprnummerok(nr) THEN
0120   PRINT "CPR nummer OK"
0130 ELSE
0140   PRINT "CPR nummer forkert"
0150 ENDIF
0160 END
```

Eksempel 2

```
0010 PROC enotation(tal,REF s$) CLOSED
0020   // Proceduren omformer tal til en streng, der har
0030   // formatet <fortegn>x.xxxxxxxxxxxxE<fortegn>nnn
0040   DIM fortegn$ OF 1,mantisse$ OF 14
0050   DIM ex$ OF 4,t$ OF 19
0060   IF tal>=0 THEN
0070     fortegn$:="+"
0080   ELSE
0090     fortegn$:="-"
0100   ENDIF
0110   t$:=STR$(ABS(tal))
0120   eposi:=("E" IN t$)
0130   IF eposi<>0 THEN
0140     ex$:=t$(eposi+1:LEN(t$))
0150     mantisse$:=t$(1:eposi-1)
0160     IF LEN(mantisse$)=1 THEN mantisse$:=mantisse$+"."
0170   ELSE
0180     pktpos:=("." IN t$)
0190     IF pktpos<>0 THEN
0200       t$:=t$(1:pktpos-1)+t$(pktpos+1:LEN(t$)) // . fjernes
0210     ELSE
0220       pktpos:=LEN(t$)+1
0230     ENDIF
0240     potens:=0
0250     IF t$<>"0" THEN
0260       potens:=pktpos-2; i:=1
0270       WHILE t$(i:i)="0" DO i:=i+1; potens:=potens-1
0280       t$:=t$(i:LEN(t$))
0290     ENDIF
0300     IF potens>=0 THEN
0310       ex$:="+0"+STR$(potens DIV 10)+STR$(potens MOD 10)
0320     ELSE
0330       ex$:="-0"+STR$(-potens DIV 10)+STR$(-potens MOD 10)
0340     ENDIF
0350     mantisse$:=t$(1:1)+"."
0360     IF LEN(t$)>1 THEN mantisse$:=mantisse$+t$(2:LEN(t$))
0370   ENDIF
```

```
0380    mantisse$:=mantisse$+"000000000000"
0390    s$:=fortegn$+mantisse$+"E"+ex$
0400 ENDPROC enotation
0410
0420 DIM talstr$ OF 20
0430 ZONE 30
0440 WHILE NOT EOD DO
0450    READ tal
0460    EXEC enotation(tal,talstr$)
0470    PRINT tal,talstr$
0480 ENDWHILE
0490 DATA -1,3.5E+070,-4E-032,0.000000001
0500 DATA 999999.99,0.1,0.01,0.001
0510 DATA 1,10,100,1000,10000,100000,1000000

RUN
-1                              -1.000000000000E+000
3.5E+070                        +3.500000000000E+070
-4E-032                         -4.000000000000E-032
0.000000001                     +1.000000000000E-010
999999.99                       +9.999999900000E+005
0.1                             +1.000000000000E-001
0.01                            +1.000000000000E-002
0.001                           +1.000000000000E-003
1                               +1.000000000000E+000
10                              +1.000000000000E+001
100                             +1.000000000000E+002
1000                            +1.000000000000E+003
10000                           +1.000000000000E+004
100000                          +1.000000000000E+005
1000000                         +1.000000000000E+006
```

10.88 SYS 10.88

COMAL80 funktion

Format

SYS(<nudtr>)

<nudtr>: et numerisk udtryk.

Anvendelse

Funktionen returnerer systeminformation afhængig af værdien af <nudtr>:

Funktion	Information
SYS(0)	Fejlnummeret for sidste RUN-time fejl (har kun en værdi forskellig fra nul i en PROC-HANDLER).
SYS(1)	Strømnummeret for sidste strøm man har benyttet (har kun en værdi forskellig fra nul i en PROC-HANDLER).
SYS(2)	Linienummeret på den linie, hvori der opstod fejl (har kun en værdi forskellig fram nul i en PROC-HANDLER).
SYS(3)	Antal 20 msek siden opstart.
SYS(4)	Antal frie bytes i lageret.
SYS(5)	Den aktuelle ZONE-værdi.
SYS(6)	Den aktuelle MARGIN-værdi.

10.89 TAB 10.89

COMAL80 funktion

Format

TAB (<nudtr>)

<nudtr>: et numerisk udtryk i området 1<=<nudtr> <=
sidebredden angivet med MARGIN-sætningen.

Anvendelse

Funktionen anvendes kun i PRINT-sætninger og bevirker tabulering til den kolonne, der er angivet ved <nudtr>.

Kommentarer

1. Udskrivningskolonnerne nummereres fra 1.

Eksempel

```
0010 MARGIN 80
0020 ZONE 0
0030 FOR i:=1 TO 60 DO PRINT USING "#":i MOD 10;
0040 PRINT
0050 WHILE NOT EOD DO
0060   READ pos
0070   PRINT TAB(pos);"* ";pos
0080 ENDWHILE
0090 DATA 1,7,40,38

RUN
12345678901234567890123456789012345678901234567789012345678 90
* 1
      * 7
                                              * 40
                                            * 38
```

10.90 TAN 10.90

COMAL80 funktion

Format

TAN (<nudtr>)

<nudtr>: et numerisk udtryk, der angiver et radianantal.

Anvendelse

Funktionen udregner tangens til en vinkel udtrykt i radianer.

10.91 Tildeling 10.91

COMAL80 sætning/kommando

Format

```
{ <nvar>:= <nudtr> }  [ ; { <nvar>:= <nudtr> } ]...
{ <svar>:= <sudtr> }      { <svar>:= <sudtr> }
```

<nvar>: en numerisk variabel
<nudtr>: et numerisk udtryk
<svar>: en strengvariabel
<sudtr>: et strengudtryk

Anvendelse

Sætningen/kommandoen anvendes, når en variabel skal tildeles en værdi.

Bemærkninger

1. <nvar>, <svar> skal være indicerede, hvis de er erklæret således.

2. Angående formatet for numeriske udtryk henvises til afsnit 4.

3. Angående formatet for strengudtryk henvises til afsnit 5.

Eksempel

```
0010 DIM navn$ OF 20
0020 navn$:="COMAL80"
0030 ejok:=(navn$<"COMAL")
0040 pi:=ATN(1)*4; slut:=FALSE
```

COMAL80 konstant

Format

TRUE

Anvendelse

TRUE er en konstant med værdien 1

Eksempel

```
0010 // Eksempel på en uendelig løkke
0020 WHILE TRUE DO
...
0100 ENDWHILE
```

10.93 VAL 10.93

COMAL80 funktion

Format

VAL(<sudtr>)

<sudtr>: et vilkårligt strengudtryk

Anvendelse

Funktionen udregner værdien af et tal, der optræder i en streng.

Bemærkninger

1. Funktionen gør det muligt at bruge strengvariable i INPUT-sætningen, og derefter omforme strengen til en numerisk værdi.

2. VAL omformer alle former for talrepresentation (incl. decimal- og exponentiel-notation).

3. VAL udfører det modsatte af STR$.

10.94 WHILE 10.94

COMAL80 sætning

Format

WHILE <logisk udtryk> DO <simpel sætning>

<logisk udtryk>: et udtryk, der, når det udregnes, har værdien sand (<>0) eller falsk (=0).

<simpel sætning>: en simpel COMAL80 sætning, f.eks. EXEC, INPUT, INPUT FILE, PRINT, PRINT FILE, READ, READ FILE, tildeling, WRITE FILE.

Anvendelse

Benyttes til at gentage udførelsen af en sætning så længe et udtryk er sandt.

Virkemåde

1. Hvis værdien af <logisk udtryk> er falsk (=0) hoppes til næste sætning efter WHILE-sætningen.

2. <simpel sætning> udføres.

3. Hop til trin 1.

Bemærkninger

1. Sætningen kan ikke udføres som kommando.

2. Da <simpel sætning> kun udføres, så længe <logisk udtryk> er sand (<>0), er det ikke sikkert, at <simpel sætning> bliver udført overhovedet.

3. Ønskes mere end en <simpel sætning> udført, skal konstruktionen WHILE-ENDWHILE benyttes.

10.95 WHILE-ENDWHILE 10.95

COMAL80 struktur

Format

```
WHILE <logisk udtryk> DO
  <sætningsliste>
ENDWHILE
```

<logisk udtryk>: et udtryk, der når det udregnes, har værdien sand (<>0) eller falsk (=0).

<sætningsliste>: COMAL80 sætninger.

Anvendelse

Konstruktionen benyttes til at gentage udførelsen af <sætnings liste> sålænge <logisk udtryk> er sand.

Virkemåde

1. Hvis værdien af <logisk udtryk> er falsk (=0) hoppes til sætningen efter ENDWHILE sætningen.

2. <sætningsliste> udføres.

3. Hop til trin 1.

Bemærkninger

1. Sætningen kan ikke udføres som kommando.

2. WHILE-ENDWHILE må gerne indeholde andre WHILE- og WHILE-ENDWHILE sætninger.

3. Hvis ENDWHILE mangler fås fejl 0096: FEJL I PROGRAMSTRUKTUR.

4. Hvis man hopper ind i en WHILE-ENDWHILE løkke udenom WHILE-sætningen fås fejl 0116: ULOVLIGT HOP.

5. Da <sætningsliste> kun udføres, sålænge <logisk udtryk> er sand (<>0), er det ikke sikkert, at <sætningsliste> udføres overhovedet.

Eksempel

```
0010 INPUT "Indtast et beløb (kroner.ører) :": beløb
0020 beløb:= INT(beløb*20+0.5)/20
0030 PRINT "Afrundet til nærmeste hele femøre :";beløb
0040 DIM navn$ OF 50
0050 WHILE NOT EOD DO
0060   READ kroner,navn$
0070   antal:= 0
0080   WHILE beløb>=kroner DO
0090     beløb:= beløb-kroner
0100     antal:= antal+1
0110   ENDWHILE
0120   PRINT antal;navn$
0130 ENDWHILE
0140 DATA 1000,"Tusindkronesedler"
0150 DATA 500,"Femhundredkronesedler"
0160 DATA 100,"Hundredkronesedler"
0170 DATA 50,"Halvtredskronesedler"
0180 DATA 20,"Tyvekronesedler"
0190 DATA 10,"Tikroner"
0200 DATA 5,"Femkroner"
0210 DATA 1,"Enkroner"
0220 DATA 0.25,"Femogtyveøre"
0230 DATA 0.1,"Tiøre"
0240 DATA 0.05,"Femøre"
```

10.96 WRITE FILE 10.96

COMAL80 sætning/kommando

Format

WRITE FILE <strømnr> [,<recnr>] :{ <nudtr> / <sudtr> }[, { <nudtr> / <sudtr> }] ...

<strømnr>: et numerisk udtryk, hvis værdi angiver nummeret på en åben datastrøm.

<recnr>: et numerisk udtryk, hvis værdi angiver nummeret på en post (angives kun ved RANDOM-filer).

<nudtr>: et numerisk udtryk

<sudtr>: et strengudtryk

Anvendelse

Sætningen/kommandoen udskriver data i binært format i en datastrøm. Filen skal være åbnet (OPEN) i WRITE-, APPEND- eller RANDOM-mode.

Bemærkninger

1. Dataene kan læses igen med READ FILE-sætningen.

2. I binært format fylder en numerisk variabel 8 tegn (bytes) og en streng 2 tegn plus den aktuelle længde.

3. Hvis man forsøger at udskrive en record (i en RANDOM-fil), der er længere end den record-længde, der er specificeret i OPEN-sætningen, fås fejl 0221: END-OF-RECORD.

Eksempel

```
0010 PROC matwritefile(strøm,REF mat(,),dim1,dim2) CLOSED
0020   FOR i:=1 TO dim1 DO
0030     FOR j:=1 TO dim2 DO WRITE FILE strøm: mat(i,j)
0040   NEXT i
0050 ENDPROC matwritefile
0060
0070 PROC matwriterndfile(strøm,post,REF mat(,),dim1,dim2) CLOSED
0080   WRITE FILE strøm,post: mat(1,1)
0090   FOR j:=2 TO dim2 DO WRITE FILE strøm: mat(1,j)
0100   FOR i:=2 TO dim1 DO
0110     FOR j:=1 TO dim2 DO WRITE FILE strøm: mat(i,j)
0120   NEXT i
0130 ENDPROC matwriterndfile
```

10.97 ZONE 10.97

COMAL80 sætning/kommando

Format

ZONE <nudtr>

<nudtr>: et numerisk udtryk med værdi i intervallet 0≤ <nudtr> ≤ sidebredden angivet med MARGIN sætningen.

Anvendelse

Sætningen/kommandoen benyttes til at angive kolonnebredden mellem elementer i en PRINT sætning adskilt med komma (,).

Bemærkninger

1. Standardværdien ved opstart er ZONE 0. Dette bevirker at PRINT-elementerne udskrives uden mellemrum.

2. Da den maximale ZONE-værdi er afhængig af MARGIN værdien, er det praktisk først at angive MARGIN og derefter ZONE.

3. Hvis MARGIN-værdien er nul, kan ZONE-værdien være vilkårlig.

Eksempel

```
0010 MARGIN 30
0020 ZONE 10
0030 FOR i:= 1 TO 10 DO PRINT i,
0040 PRINT
0050 ZONE 5
0060 FOR i:= 1 TO 10 DO PRINT i,
0070 PRINT
0080 END
```

```
1         2         3
4         5         6
7         8         9
10
1    2    3    4    5    6
7    8    9    10
```

 REFERENCER

[1] RCSL Nr. 42-i1797:
RC700 Brugervejledning

[2] RCSL NR. 42-i2063:
COMAL80 Referencekort

B. FEJLMEDDELELSER B.

De fejl, der kan opstå i COMAL80 systemet, kan deles i 3 grupper:

1. Fejl fundet under programindtastning.

   Hvis en fejl konstateres under programindtastning, udskrives en fejlmeddelelse i skærmens øverste højre hjørne, og cursoren placeres umiddelbart efter det sted, hvor fejlen blev fundet.
   En liste over disse fejlmeddelelser findes i afsnit B.1.

2. Fejl fundet under program-, eller kommando-udførsel.

   Fejlmeddelelserne til disse udskrives på følgende form:
   at nnnn
   error: eeee

   nnnn : linienummeret på sætningen hvor fejlen blev fundet.
   eeee : et tal under 200, hvis betydning er beskrevet i afsnit B.2.

   Opstår fejlen under kommandoudførsel, udskrives der intet linienummer.

3. Fejl fundet under disketteoperationer.

   Fejlmeddelelserne til disse har samme format som fejlmeddelelserne under pkt 2, men fejlnumrene er større end 200 og er beskrevet i afsnit B.3.

B.1 Fejl fundet under programindtastning B.1

linie for lang

Fejlmeddelelsen betyder, at den linie man har indtastet ikke kan repræsenteres internt, hvorfor den må brydes op i flere linier

ulovligt tegn

Systemet har fundet et tegn, det ikke accepterer andre steder, end i strengkonstanter og kommentarer, f.eks. %. Cursoren er placeret efter det ulovlige tegn.

ulovligt linienummer

Et linienummer må ikke være mindre end 1 eller større end 9999

stakoverløb

Der er så lidt ledigt lager tilbage, at systemet ikke kan omforme sætningen til det interne format.

variabel forventet

Systemet forventede en variabel på det sted, hvor cursoren er placeret, f.eks

```
0010 READ "niels"
```

')' forventet

Systemet forventede en parentesafslutning ')' på det sted, hvor cursoren er placeret, f.eks.

```
0010 a(1,1,1) := 10
```

fejl i indicering

operand forventet

Systemet forventede en operand på det sted, hvor cursoren er placeret.

```
0010 a:=10*
```

ulovlig type

Man har f.eks. angivet en streng, hvor en numerisk udtryk var forventet (eller omvendt).

```
0010 L$ := 7
```

syntaks fejl

COMAL80 har genkendt den første del af linien som en sætning, men de sidste tegn giver ingen mening, eller der mangler et tegn, eller et nøgleord bruges som variabelnavn.

```
0010 a:=5  b:=6
```

fejl i konstant

Opskrivningen af en konstant følger ikke reglerne.

```
0010 a:=3.1E-
```

navn for langt

Et navn må maximalt være på 16 tegn.

" forventet

Systemet mangler et anførselstegn (") til afslutningen af en strengkonstant.

```
0010 1$ := "COMAL80
```

'(' forventet

Systemet forventede at møde en begyndelsesparentes på det sted, hvcor cursoren er placeret.

```
0010 DIM a,b(5)
```

',' forventet

Systemet forventede at møde et komma på det sted, hvor cursoren er placeret.

```
0010 OPEN FILE 1,"DATAFIL" READ
```

':=' forventet

systemet forventede at finde et tildelingstegn på det sted, hvor cursoren er placeret.

```
0010 a=5
```

':' forventet

systemet forventede at finde et kolon på det sted, hvor cursoren er placeret.

```
0010 l$(1,2,3):="TEKST"
```

'OF' forventet

OF skal skrives som afslutning af en CASE-sætning.

```
0010 CASE a
```

'TO' forventet

Systemet kan ikke finde TO i en FOR-sætning.

```
0010 FOR i:=1 STEP 2 TO 10 DO
```

'DO' forventet

DO skal skrives i en WHILE- eller i en FOR-sætning.

```
0010 WHILE i>10
```

'THEN' forventet

THEN skal skrives i en IF-sætning.

```
0010 IF svar$="SLUT" PRINT "Farvel"
```

'REF' forventet

Taltabeller og teksttabeller skal REF specificeres som parametre.

```
0010 PROC P(a(,))
```

navn forventet

Systemet forventede et navn (f.eks. på en label) der, hvor cursoren er placeret.

```
0010 GOTO 1000
```

ulovlig kommando

Sætningen kan ikke udføres som kommando (se afsn 2.3).

```
WHILE i<=10 DO PRINT i
```

ikke implementeret

Faciliteten er endnu ikke implementeret i systemet.

0095 : CON IKKE TILLADT
Man forsøger at bruge CON i forbindelse med et program, man har rettet i.

0096 : FEJL I PROGRAMSTRUKTUR (FOR-NEXT, IF-ENDIF osv)
Systemet kan ikke finde den tilhørende NEXT til FOR, ENDIF til IF osv.

0097 : IKKE SAVE FIL
LOAD-kommandoen er anvendt på en fil, der ikke indeholder et SAVEd program.

0098 : IKKE IMPLEMENTERET
Faciliteten er endnu ikke implementeret i systemet.

0099 : SYSTEMFEJL
I dette tilfælde bør man udfylde en RC FEJLMEDDELELSE, der beskriver fejlen.

0100 : ESCAPE
Opstår kun i forbindelse med en HANDLER. Hvis ESC trykkes ned, mens en brugerdefineret HANDLER er ENABLE'd, vil HANDLER'en blive kaldt, og SYS(0) have værdien 100.

0101 : ULOVLIG HELTALSVÆRDI
Et sted, hvor systemet forventer et heltal, har en variabel en værdi større end 32768.

0102 : LOG TIL NEGATIVT TAL
Forsøg på at udregne logaritmen af et ikke-positivt tal.

0103 : KVADRATROD AF NEGATIVT TAL
Forsøg på at uddrage kvadratroden af et negativt tal.

0104 : DIVISION MED NUL
Forsøg på at dividere med nul, eller med et tal, der er så lille, at det opfattes som nul af systemet.

0106 : ARITMETISK OVERLØB

En udregning giver et resultat, der er udenfor COMAL80's talområde, dvs større end 9.99...E+126 eller mindre end -9.99...E-126.

0108 : STAKOVERLØB

Programmet er for stort. Split det f.eks op i externe procedurer.

0109 : TYPEKONFLIKT

Fejl i f.eks. parametrene til en procedure.
Eksempel:

```
0010 PROC p(i)
0020 ENDPROC p
0030 EXEC p("tekst")
```

0110 : VARIABEL IKKE ERKLÆRET

Alle tekstvariable, vektorer og matricer skal erklæres.

0111 : VARIABEL ALLEREDE ERKLÆRET

En etikette, en tekstvariabel og en numerisk variabel må ikke have det samme navn.

0112 : PARAMETER FEJL

Parametrene til en procedure passer ikke til de specificerede, f.eks.

```
0010 PROC tom
0020 ENDPROC tom
0030 EXEC tom(3)
```

0113 : FUNKTIONSVÆRDI UDEFINERET

Funktionens værdi er udefineret for det givne argument, f.eks.

```
0010 FUNC funk(i)
0020   IF i>10 then RETURN i
0030 ENDFUNC funk
0040 j:=funk(2)
```

0114 : ULOVLIG FILIDENTIFIKATOR

Stømnummeret er ikke 1,2,3,4 eller 5, f.eks.

```
0010 OPEN 10,"DATAFIL",READ
```

0115 : ULOVLIG CASE VÆRDI

Argumentet efter CASE er ikke specificeret i nogen WHEN-sætning, og OTHERWISE er ikke opgivet,f.eks.

```
0010 i:=6
0020 CASE i OF
0030 WHEN 1
0040   PRINT "1 er fundet"
0050 ENDCASE
```

0116 : ULOVLIGT HOP

Man må ikke hoppe ind i en konstruktion af typen PROC-ENDPROC,IF-ELSE-ENDIF, REPEAT-UNTIL, WHILE-ENDWHILE, osv. f.eks.

```
0010 GOTO e
0020 IF FALSE THEN
0030 e:
0040   PRINT "FALSE"
0050 ENDIF
```

0117 : IKKE FLERE DATA

For mange variable i READ-sætningerne i forhold til antallet af DATA-elementer, f.eks.

```
0010 READ a,b
0020 DATA 2
```

0118 : FEJL I INPUT

Hvis man f.eks. indtaster tekst til en INPUT-sætning, der forventer et tal, og brugeren har ENABLE'd sin egen HANDLER, vil HANDLER'en blive kaldt, og SYS(0) have værdien 118.

0119 : IKKE CLOSED PROC

En EXTERN procedure skal være lukket (CLOSED)

0120 : INDEX FEJL

Et array indiceres udenfor sine grænser. BEMÆRK at nederste indexgrænse altid er 1.

0121 : FEJL I PRINT USING

Formatstrengen er f.eks. forkert, f.eks.

0010 PRINT USING "##.##.##" : 10

B.3 I/O-Fejlmeddelelser B.3

I/O-fejl er fejl, der er opstået i forbindelse med indlæsnings- eller udskrivnings-operationer.

0201 : END-OF-FILE
Forsøg på at læse en fil ud over end-of-file markeringen.

0203 : PRINTER TIMEOUT
Printeren er ikke klar til at modtage tegn, selv efter at systemet har forsøgt flere gange indenfor ca. 25 sekunder.

0204 : ULOVLIG OPERATION

0205 : DISKETTEFEJL
En fysisk fejl er opdaget på disketten, f.eks. paritetsfejl.

0206 : TIMEOUT
Systemet kan ikke "få kontakt" med disketten, selv ikke efter, at det har forsøgt flere gange indenfor ca. 2 sekunder.

0207 : DISKETTE OFFLINE
Disketteenheden er f.eks. slukket.

0208 : DISKETTE ER BLEVET SKIFTET
Hvis man ønsker at skrive på en diskette, skal man først udføre en MOUNT kommando.

0209 : DISKETTE SKRIVEBESKYTTET
Disketten er blevet skrivebeskyttet, og det er derfor umuligt at bruge kommandoerne WRITE FILE, INPUT FILE, SAVE og LIST.

0211 : PRINTER OFFLINE
Printeren er f.eks. slukket eller forbundet forkert.

0212 : FEJL I FILNAVN
Filnavnet indeholder f.eks. ulovlige tegn.

0213 : FIL EKSISTERER ALLEREDE
Der eksisterer allerede en fil med det angivne navn på disketten.

0214 : FIL EKSISTERER IKKE
Den fil, der refereres til, eksisterer ikke på den angivne diskette.

0215 : DISKETTE FULD
Der er ikke mere plads på disketten.

0217 : IKKE ÅBEN/ALLEREDE ÅBEN
Forsøg på at åbne en strøm med et strømnr, der allerede er åben, eller forsøg på at referere til et strømnr, der ikke er åben.

0218 : RESERVERET
Forsøg på at skrive på printeren via flere stømme samtidigt.

0219 : ÅBNE FILER PÅ ENHED
Man kan ikke udføre DISMOUNT, hvis ikke alle filer er CLOSEd.

0220 : ULOVLIGT POST NUMMER
Postnummeret er enten negativt,0 eller for stort.

0221 : END-OF-RECORD
Forsøg på at læse eller skrive udover record-størrelsen.

0222 : FIL SKRIVEBESKYTTET
Man kan ikke skrive i systemfiler.

0223 : LINIE FOR LANG
Forsøg på at inlæse en linie, der er for lang. (INPUT FILE).

C. YDRE ENHEDER

C.1 Skærmstyring

Det er ved hjælp af kontroltegn muligt at udføre forskellige former for skærmstyring. Typiske eksempler kan være sletning af skærm, linieskift samt direkte styring af cursoren.

Udførelsen af den ønskede kontrolfunktion sker ved anvendelse af CHR$(X) i PRINT sætninger.

Følgende kontrolfunktioner er til rådighed:

X	CHR$(X)
5	Slet et tegn i linie
7	'BELL', dvs. hørbart signal (kun RC702)
8	Cursor en position til venstre (back space)
9	Indsæt et tegn i linie
10	Linieskift (LF): cursor en position ned og hen på første position på linien
12	Slet skærm (FF): cursor til position 1,1
13	Vognretur (CR): cursor til første position på linien
24	Cursor en position til højre
26	Cursor en position op
29	Cursor til position (1,1) (home up)
30	Slet fra aktuel position til slutningen af linien (EOL)
31	Slet fra aktuel position til slutningen på skærmbilledet (EOF)

C.1.1 Blink og invers skrift

Dele af skærmbilledet kan bringes til at blinke og/eller fremtræde som invers skrift (invers video). Dette gøres ved at sende bestemte kontroltegn større end 128, hvorefter uddata fra de efterfølgende printsætninger vil se ud som specificeret ved de anvendte kontroltegn.

CHR$(128) sætter skærmstyringen tilbage til normal skrift

X	CHR$(X)
128+2 =130	Blink
128+4 =132	Aktivering af semigrafisk tegnsæt
128+16=144	Invers skrift
128+32=160	Understregning
128	Normal skrift

Eksempel

```
0030 PRINT CHR$(130);"RC700";CHR$(128)
```

Bemærkning

Teksten RC700 vil blinke

De forskellige faciliteter kan anvendes sammen, således vil f.eks. X=128+2+16 bevirke blinkende, invers skrift.

Bemærk, at kontroltegnet i sig selv fylder en position (en blank) på skærmen. Kontroltegnet vil kun have virkning, så længe det befinder sig på skærmen, det vil sige, at virkningen f.eks. ophører, hvis billedet "ruller", og tegnet dermed forsvinder.

Hvis man benytter variable til at angive de forskellige værdier, vil det øge programmets overskuelighed, f.eks.

Eksempel

```
0010 blink:=2; invers:=16
0020 PRINT CHR$(128+blink+invers);"RC700";CHR$(128)
```

C.1.2 Semigrafisk tegnsæt

C.1.2

Skærmen indeholder mulighed for udskrivning af et semigrafisk tegnsæt. Denne mulighed kan f.eks. anvendes til kurvetegning, stolpediagrammer og simplere grafisk afbildning.

Det semigrafiske tegnsæt aktiveres ved brug af kontroltegnet 132. Returnering til normalt tegnsæt sker ved brug af kontroltegnet 128.

Eksempel

```
0010 PRINT CHR$(132);"RC700";CHR$(128)
```

Figur C.2 er en komplet fortegnelse over skærmens semigrafiske tegnsæt.

C.1.3 Skærm - tegngenerering og konvertering

Ved udskrift til skærmen sendes et 8-bit tegn til skærm-programmet. Tegnet konverteres her via skærmkonverteringstabellen, der i den danske version er en én til én konvertering.

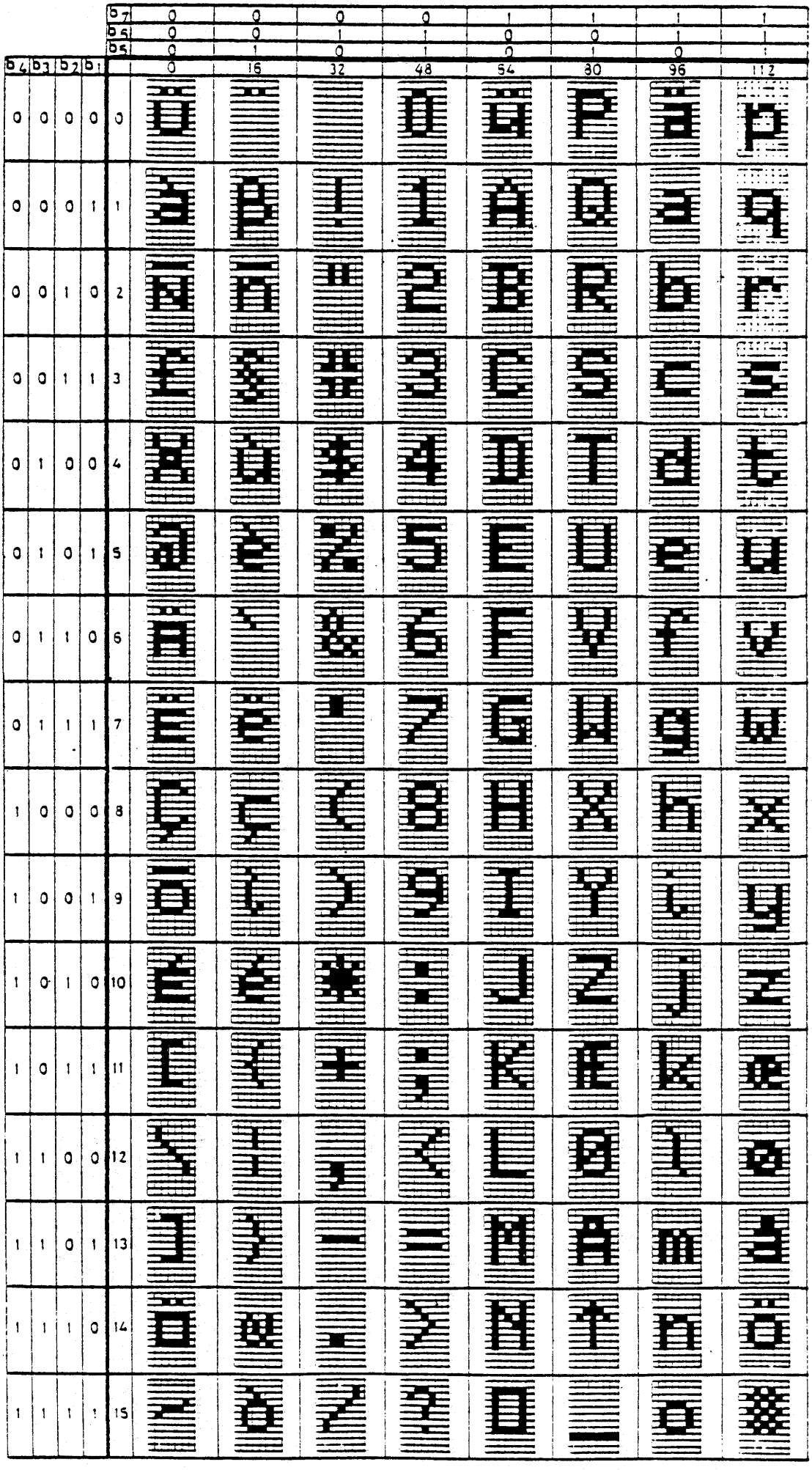
Under udskrivning af et tegn kan man være enten i normal eller semi-grafisk tilstand. I den normale tilstand konverteres alene tegn mindre end 128. I semigrafisk tilstand sker der ingen konvertering. De to tegnsæt fremgår af fig. 1 og 2. Det bemærkes, at tegnværdier i intervallet 0-31 er forbeholdt styretegn til skærmen, hvorfor det tilsvarende værdiområde i tegnsættene må genereres på anden måde, som det fremgår af følgende tegninddeling:

G1: kontroltegn (0-31)
G2: alfanumeriske tegn - udvidet ((32-127) + (192-223))
G3: specielle skærmfunktioner (128-191)
G4: semigrafiske tegn - udvidet ((32-127) + (192-223))

Grupperne G1 og G3 er omtalt i afsnit C.1, C.1.1 og C.1.2

Grupperne G2 og G4 repræsenterer tegnsættet i henholdsvis normal og semi-grafisk tilstand som angivet i fig. C.1 og C.2. Det bemærkes, at man skal angive værdiområdet 192-223 for at få vist de enkelte tegnværdier mindre end 32. Hertil anvendes PRINT-sætningen og CHR$-funktionen.

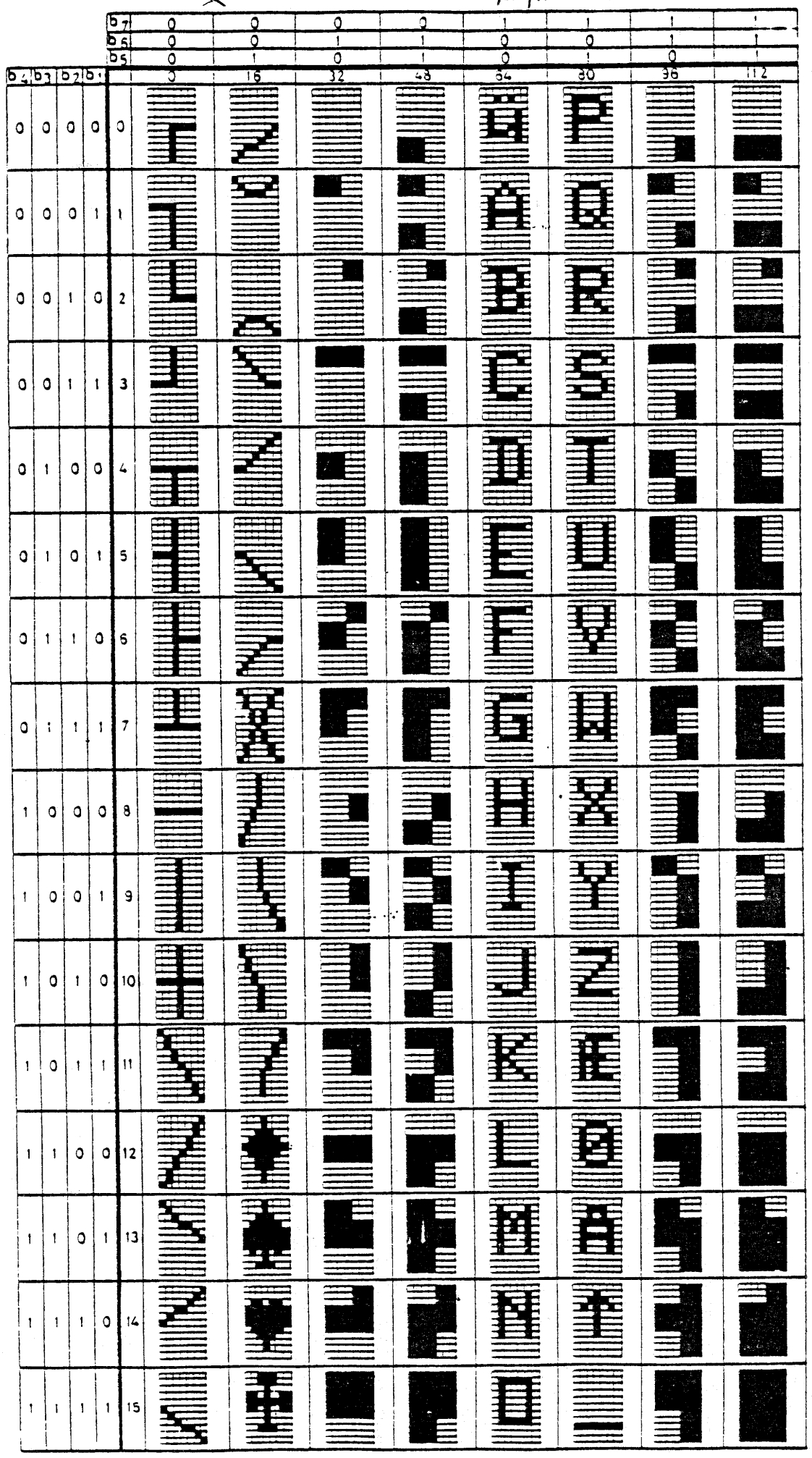
normal mode 192 - 223
Ikke printer



Figur C.1: Dansk ASCII alfanumerisk tegntabel.

CHR# (132) 192-223.

244 Bogstaver kan ikke komme på printer i samme område



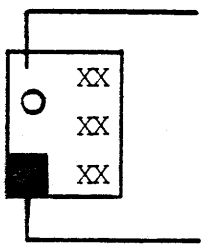
Figur C.2: Dansk ASCII semigrafisk tegnsæt.

C.2 Tastatur - tegngenerering og konvertering C.2

Ved anslag leverer tastaturet et 8-bit tegn (256 forskellige muligheder) som vist i fig. C.3 og C.4. Dette tegn konverteres herefter til et nyt 8-bit tegn ved hjælp af inddata konverteringstabellen. Tabellen er vist i afsnit C.2.1, hvor kolonne 1 viser den konverterede værdi. Kolonne 2 viser den originale tegnværdi, som genereres i tastaturet ifølge fig. C.3 og C.4. Tegnet, dvs. værdien i kolonne 1, afleveres til COMAL80, hvor yderligere tegnbehandling kan finde sted før en eventuel udskrivning af tegnet på skærmen.

Det konverterede tegn kan henføres til 4 grupper således:

G1: kontroltegn (0-31)
G2: inddata redigeringstegn i COMAL80 (0-31)
G3: grafiske tegn (32-127)
G4: PF og PA tegn (128-143) + (144-153)



tastatur gravering.

- værdi fra tastatur , når 'SHIFT' anvendes
- værdi fra tastatur , når 'SHIFT' ikke anvendes
- værdi fra tastatur , når 'CTRL' anvendes (hvis defineret)

ændret.

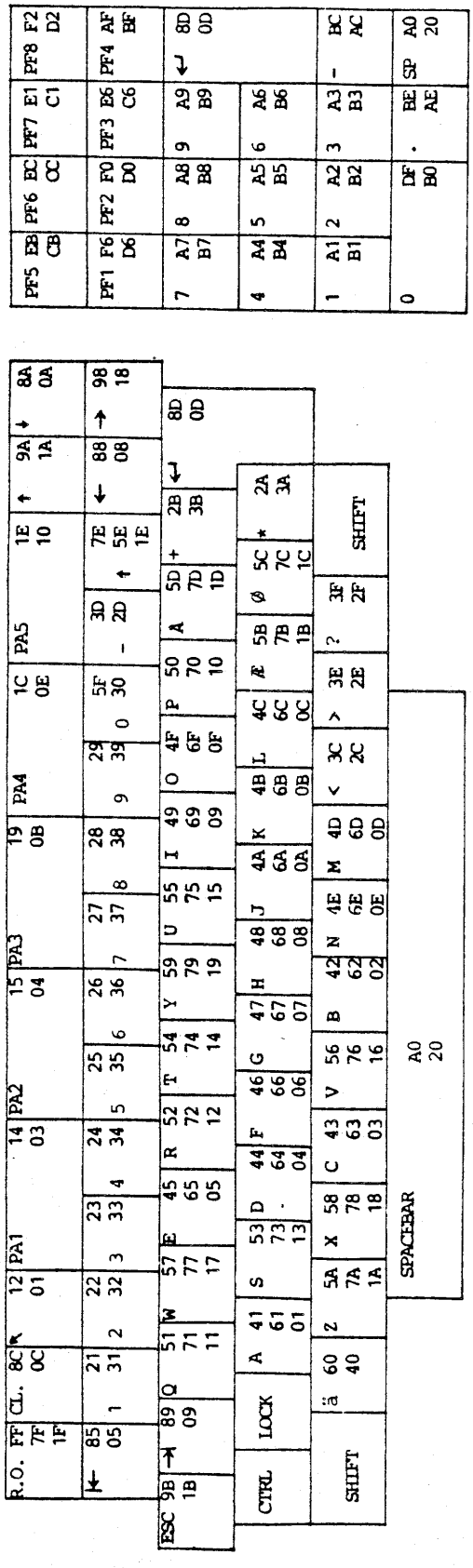


FIG.C.3: RC721/22 tastatur i dansk udgave (serienummer 0-50/383). Hexadecimal notation.

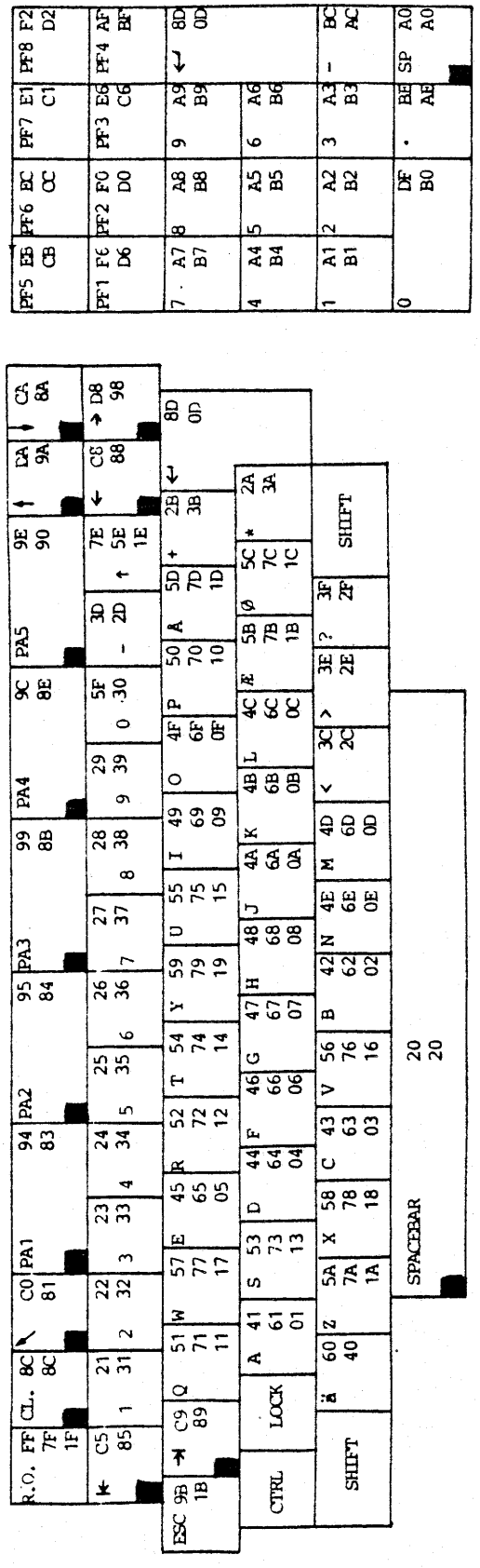


FIG. C.4: RC721/22 tastatur i dansk udgave (serienummer 51/384-). Hexadecimal notation.

Gruppen G1 er beskrevet i afsnit C.1. Tegnene benyttes til skærmstyring. Ved indtastning vil tegn fra denne gruppe normalt ikke blive registreret. Dog vil visse tegn blive anvendt som redigeringstegn, se nedenfor. Dette gælder ved indtastning af sætninger og kommandoer, samt i INPUT-sætninger i et program. Det bemærkes dog, at funktionen KEY$, samt GET$ fra datastrømmen 1/keyboard eller 1/console som funktionsresultat uden undtagelse giver den indtastede værdi konverteret ifølge konverteringstabellen i afsnit C.2.1.

8: cursor til venstre
24: cursor til højre
13: vognretur
4, 5: slet tegn i tegnfølge
9: afsæt plads til tegn i tegnfølge
11: slet resten af linien
127: slet tegn
27: afbryd indtastning

Gruppen G3 repræsenterer den danske udgave af alfanumeriske ASCII tegnsæt og understøttes generelt.

Gruppen G4 består af tegn med værdier større end 127. Tegnene understøttes kun af funktionerne KEY$ og GET$. Da de angivne tegnværdier falder uden for det normale 7-bit tegn-interval (ASCII-tegn), kan værdierne tillægges specialbetydning efter ønske i de enkelte COMAL80-programmer.

PA- og PF-tasterne har fået tillagt følgende værdier i rækkefølge (angivet i parentes):

PF1(128), ..., PF8(135); shift PF1(136), ..., shift PF8(143);

PA1(144), ..., PA5(148); shift PA1(149), ..., shift PA5(153);

C.2.1 Konverteringstabel for tastatur

Inddata til system	Inddata (hexadecimal) fra tastatur	tastatur ident	funktion	special funktion
0	0(00)			
29	1(01)	ctrl+a		
2	2(02)	ctrl+b		
3	3(03)	ctrl+c		
4	4(04)	ctrl+d		slet tegn i tegnfølge
5	5(05)	ctrl+e		slet tegn i tegnfølge
6	6(06)	ctrl+f	xy-addressering	ingen funktion
7	7(07)	ctrl+g	hørbart signal	ingen funktion
8	8(08)	ctrl+h	cursor til venstre	
9	9(09)	ctrl+i	tabulering (4*Δ)	ingen funktion
10	10(0A)	ctrl+j	cursor ned	ingen funktion
11	11(0B)	ctrl+k		slet resten af linien
12	12(0C)	ctrl+l	slet skærm	ingen funktion
13	13(0D)	ctrl+m,		
			vognretur	plus cursor ned
14	14(0E)	ctrl+n		
15	15(0F)	ctrl+o		
16	16(10)	ctrl+p		
17	17(11)	ctrl+q		
18	18(12)	ctrl+r		
19	19(13)	ctrl+s		
20	20(14)	ctrl+t		
21	21(15)	ctrl+u		
22	22(16)	ctrl+v		
23	23(17)	ctrl+w		ingen funktion
24	24(18)	ctrl+x	cursor til højre	
25	25(19)	ctrl+y		
26	26(1A)	ctrl+z	cursor op	ingen funktion
27	27(1B)	ctrl+æ,		
		esc		afbrydelse
28	28(1C)	ctrl+ø		
29	29(1D)	ctrl+å	nulstil cursor position	ingen funktion
30	30(1E)	ctrl+	slet resten af linien	ingen funktion
31	31(1F)	ctrl+rubout		
			slet resten af skærmen	ingen funktion
32	32(20)	'space bar',		
		shift+'space bar'		
			mellemrum (Δ)	
33	33(21)	!		

Inddata til system	Inddata (hexadecimal) fra tastatur	tastatur ident	funktion	special funktion
34	34(22)	"		
35	35(23)	#		
36	36(24)	$		
37	37(25)	%		
38	38(26)	&		
39	39(27)	` (pling)		
40	40(28)	(		
41	41(29)	)		
42	42(2A)	*		
43	43(2B)	+		
44	44(2C)	, (komma)		
45	45(2D)	- (minus)		
46	46(2E)	.		
47	47(2F)	/		
48	48(30)	0 (nul)		
49	49(31)	1		
50	50(32)	2		
51	51(33)	3		
52	52(34)	4		
53	53(35)	5		
54	54(36)	6		
55	55(37)	7		
56	56(38)	8		
57	57(39)	9		
58	58(3A)	:		
59	59(3B)	;		
60	60(3C)	<		
61	61(3D)	=		
62	62(3E)	>		
63	63(3F)	?		
64	64(40)	ü (tysk y)		
65	65(41)	A		
66	66(42)	B		
67	67(43)	C		
68	68(44)	D		
69	69(45)	E		
70	70(46)	F		
71	71(47)	G		
72	72(48)	H		
73	73(49)	I		
74	74(4A)	J		

Inddata til system	Inddata (hexadecimal) fra tastatur	tastatur ident	funktion	special funktion
75	75(4B)	K		
76	76(4C)	L		
77	77(4D)	M		
78	78(4E)	N		
79	79(4F)	O		
80	80(50)	P		
81	81(51)	Q		
82	82(52)	R		
83	83(53)	S		
84	84(54)	T		
85	85(55)	U		
86	86(56)	V		
87	87(57)	W		
88	88(58)	X		
89	89(59)	Y		
90	90(5A)	Z		
91	91(5B)	Æ		
92	92(5C)	Ø		
93	93(5D)	Å		
94	94(5E)			
95	95(5F)	_	(understregning)	
96	96(60)	ä	(tysk æ)	
97	97(61)	a		
98	98(62)	b		
99	99(63)	c		
100	100(64)	d		
101	101(65)	e		
102	102(66)	f		
103	103(67)	g		
104	104(68)	h		
105	105(69)	i		
106	106(6A)	j		
107	107(6B)	k		
108	108(6C)	l		
109	109(6D)	m		
110	110(6E)	n		
111	111(6F)	o		
112	112(70)	p		
113	113(71)	q		
114	114(72)	r		
115	115(73)	s		

Inddata til system	Inddata (hexadecimal) fra tastatur	tastatur ident	funktion	special funktion
116	116(74)	t		
117	117(75)	u		
118	118(76)	v		
119	119(77)	w		
120	120(78)	x		
121	121(79)	y		
122	122(7A)	z		
123	123(7B)	æ		
124	124(7C)	ø		
125	125(7D)	å		
126	126(7E)	ö (tysk ø)		
127	127(7F)	rubout		slet tegn
128	128(80)			
29	129(81)		nulstil cursor position	ingen funktion
130	130(82)			
144	131(83)	PA1		
145	132(84)	PA2		
5	133(85)			slet tegn i tegnfølge
134	134(86)			
135	135(87)			
8	136(88)		cursor til venstre	
9	137(89)		tabulering (4*Δ)	afsæt plads i tegnfølge
10	138(8A)		cursor ned	ingen funktion
146	139(8B)	PA3		
12	140(8C)	clear, shift+clear	slet skærm	
13	141(8D)	shift	vognretur	
147	142(8E)	PA4		
143	143(8F)			
148	144(90)	PA5		
145	145(91)			
146	146(92)			
147	147(93)			
149	148(94)	shift+PA1		
150	149(95)	shift+PA2		
150	150(96)			
151	151(97)			
24	152(98)		cursor til højre	
151	153(99)	shift+PA3		
26	154(9A)		cursor op	ingen funktion

Inddata til system	Inddata (hexadecimal) fra tastatur	tastatur ident	funktion	special funktion
155	155(9B)	shift+esc		
152	156(9C)	shift+PA4		
157	157(9D)			
153	158(9E)	shift+PA5		
159	159(9F)			
32	160(A0)	space, shift+space		
49	161(A1)	1		
50	162(A2)	2		
51	163(A3)	3		
52	164(A4)	4		
53	165(A5)	5		
54	166(A6)	6		
55	167(A7)	7		
56	168(A8)	8		
57	169(A9)	9		
170	170(AA)			
48	171(AB)			
45	172(AC)	- (minus)		
173	173(AD)			
46	174(AE)	.		
139	175(AF)	shift+PF4		
48	176(B0)	0 (nul)		
49	177(B1)	1		
50	178(B2)	2		
51	179(B3)	3		
52	180(B4)	4		
53	181(B5)	5		
54	182(B6)	6		
55	183(B7)	7		
56	184(B8)	8		
57	185(B9)	9		
186	186(BA)			
48	187(BB)			
45	188(BC)	- (minus)		
189	189(BD)			
46	190(BE)	.		
131	191(BF)	PF4		
29	192(C0)		nulstil cursor position	
134	193(C1)	PF7		
194	194(C2)			

Inddata til system	Inddata (hexadecimal) fra tastatur	tastatur ident	funktion	special funktion
195	195(C3)			
196	196(C4)			
5	197(C5)	shift		slet tegn i tegnfølge
130	198(C6)	PF3		
199	199(C7)			
8	200(C8)	shift+	cursor til venstre	
9	201(C9)	shift+	tabulering (4*Δ)	ingen funktion
10	202(CA)	shift+	cursor ned	ingen funktion
132	203(CB)	PF5		
133	204(CC)	PF6		
205	205(CD)			
206	206(CE)			
207	207(CF)			
129	208(D0)	PF2		
209	209(D1)			
135	210(D2)	PF8		
211	211(D3)			
212	212(D4)			
213	213(D5)			
128	214(D6)	PF1		
215	215(D7)			
24	216(D8)	shift+	cursor til højre	
217	217(D9)			
26	218(DA)	shift+	cursor op	ingen funktion
219	219(DB)			
220	220(DC)			
221	221(DD)			
222	222(DE)			
48	223(DF)	0 (nul)		
224	224(E0)			
142	225(E1)	shift+PF7		
226	226(E2)			
227	227(E3)			
228	228(E4)			
229	229(E5)			
138	230(E6)	shift+PF3		
231	231(E7)			
232	232(E8)			
233	233(E9)			
234	234(EA)			
140	235(EB)	shift+PF5		

Inddata til system	Inddata (hexadecimal) fra tastatur	tastatur ident	funktion	special funktion
141	236(EC)	shift+PF6		
237	237(ED)			
238	238(EE)			
239	239(EF)			
137	240(F0)	shift+PF2		
241	241(F1)			
143	242(F2)	shift+PF8		
243	243(F3)			
244	244(F4)			
245	245(F5)			
136	246(F6)	shift+PF1		
247	247(F7)			
248	248(F8)			
249	249(F9)			
250	250(FA)			
251	251(FB)			
252	252(FC)			
253	253(FD)			
254	254(FE)			
127	255(FF)	shift+rubout		slet tegn

C.2.2 Ændring af konverteringstabel

C.2.2

Konverteringstabellen til tastaturet ligger i en diskettefil med navnet convtab, og den indlæses, hver gang COMAL80 startes op.

I ganske få tilfælde er det praktisk at ændre denne, f.eks. kan man ændre de værdier PA- og PF-tasterne på det udvidede tastatur afgiver. En anden mulighed er at indrette konverteringstabellen så brugeren f.eks. kun kan indtaste små bogstaver (også selv om SHIFT-tasten nedtrykkes).

Eksempel

Følgende lille program ændrer konverteringstabellen, så alle bogstaver opfattes som små bogstaver ved indtastning.

```
0010 DIM tabel$ OF 256
0020 MARGIN 0
0030 OPEN FILE 1,"convtab", READ
0040 tabel$:=GET$(1,256)
0050 CLOSE FILE 1
0060 // Nu er tabellen indlæst. Husk at første element i
0070 // tabel$ svarer til 0te element i konverteringstabellen
0080 FOR i:=65 TO 93 DO
0090   tabel$(i+1:i+1):=CHR$(32+i) // store -->> små
0100 NEXT i
0110 DELETE "convtab"
0120 OPEN FILE 1,"convtab", WRITE
0130 PRINT FILE 1:tabel$;
0140 CLOSE FILE 1
0150 END
```

C.3 Printerstyring C.3

Dette afsnit indeholder en introduktion til en række printeres faciliteter. For en fuldstændig beskrivelse henvises til de enkelte printermanualer.

C.3.1 RC861-printeren C.3.1

RC861-printeren har 3 forskellige skrifttyper:

- komprimeret (132 tegn pr. linie)
- normal (80 tegn pr. linie)
- elongeret (40 tegn pr. libie)

Yderligere indeholder printeren mulighed for udskrift med et semigrafisk tegnsæt.

De forskellige faciliteter aktiveres ved hjælp af CHR$-funktionen anvendt i PRINT-sætninger:

X	CHR$(X)
14	Aktivering af semigrafisk tegnsæt
15	Aktivering af almindeligt alfanumerisk tegnsæt
29	Udskrift med komprimeret skrift
30	Udskrift med normal skrift
31	Udskrift med elongeret skrift

Når man har specificeret et tegnsæt og/eller en skriftstype, vil printeren skrive som angivet, indtil brugeren skrifter tegnsæt og/eller skriftstype ved at udskrive nye kontroltegn. Derfor er det en god ide at afslutte et program med sætningen

```
PRINT CHR$(15);CHR$(30)
```

således, at printeren "starter" rigtigt, når den næste gang benyttes.

C.3.2 RC862,RC867 printeren C.3.2

RC862 og RC867 printeren har en række faciliteter, der kan styres fra et program ved hjælp af printkommandoer.

Skriftstyper

CHR$(29) : Komprimeret skrift
CHR$(30) : Normal skrift
CHR$(29)+CHR$(31) : Lidt elongeret skrift
CHR$(30)+CHR$(31) : Elongeret skrift

Liniemellemrum

CHR$(27)+CHR$(54) : 6 linier pr tomme
CHR$(27)+CHR$(56) : 8 linier pr tomme

Tegnsæt

CHR$(14) : Semigrafisk tegnsæt
CHR$(15) : Almindeligt alfanumerisk tegnsæt

Formularkontrol

CHR$(12) : Fører papiret frem til første linie på ny side (TOF : Top Of Form)
CHR$(27)+CHR$(53) : Sætter TOF til nuværende position
CHR$(27)+CHR$(70)+"nn" : Sætter sidehøjden til nn linier
CHR$(27)+CHR$(70)+"nn" : Laver nn linieskift
(11) +"07" : - 7

Diverse

CHR$(19) : Afmelder printeren (slukker SEL lampen)
CHR$(24) : Sletter printerbufferen

C.4 HW-porte

Nedenfor er angivet en oversigt over der fysiste portnumre for RC702. Det skal understreges, at disse porte skal anvendes med den allerstørste forsigtighed, da COMAL80-systemet ikke yder nogen form for beskyttelse.

Om brugen af portene henvises til leverandørernes tekniske manualer.

Navn	Komponenttype	Port	Kommentar
Skærm	I8275 (Intel)	0	Kontrol
		1	Data
Diskette	uPD765 (NEC)	4	Kontrol
	eller I8272(Intel)	5	Data
Seriel	Z80SIO2 (Zilog)	8	Data, Terminalport
I/O		9	Data, Printerport
		10	Kontrol, Terminalport
		11	Kontrol, Printerport
CTC	Z80-CTC	12	Kanal 0, til SIO term
		13	Kanal 1, til SIO printer
		14	Kanal 2, int displ
		15	Kanal 3, int flop
Parallel	Z80-PIO	16	Data, Tastatur
I/O		17	Data, Parallel port
		18	Kontrol, Tastatur
		19	Kontrol, Parallel port
Intern switch		20	8 bit fra switchen
		22	Motor kontrol floppy
DMA	AM9517A-4	240-	
	I3237-2	255	

Eksempel

Følgende to procedurer udskriver eller indlæser et tegn på/fra en port.

```
0010 PROC pout(port,d)
0020   OPEN FILE 5,"/"+STR$(port)+"/PORT", WRITE
0030   marg:=SYS(6) // Gem MARGINs værdi
0040   MARGIN 0
0050   PRINT FILE 5:CHR$(d);
0060   CLOSE FILE 5
0070   MARGIN marg // Reetabler MARGINs værdi
0080 ENDPROC pout
0090
0100 PROC pin(port,REF d)
0110   OPEN FILE 5,"/"+STR$(port)+"/PORT", READ
0120   d:=ORD(GET$(5,1))
0130   CLOSE FILE 5
0140 ENDPROC pin
```

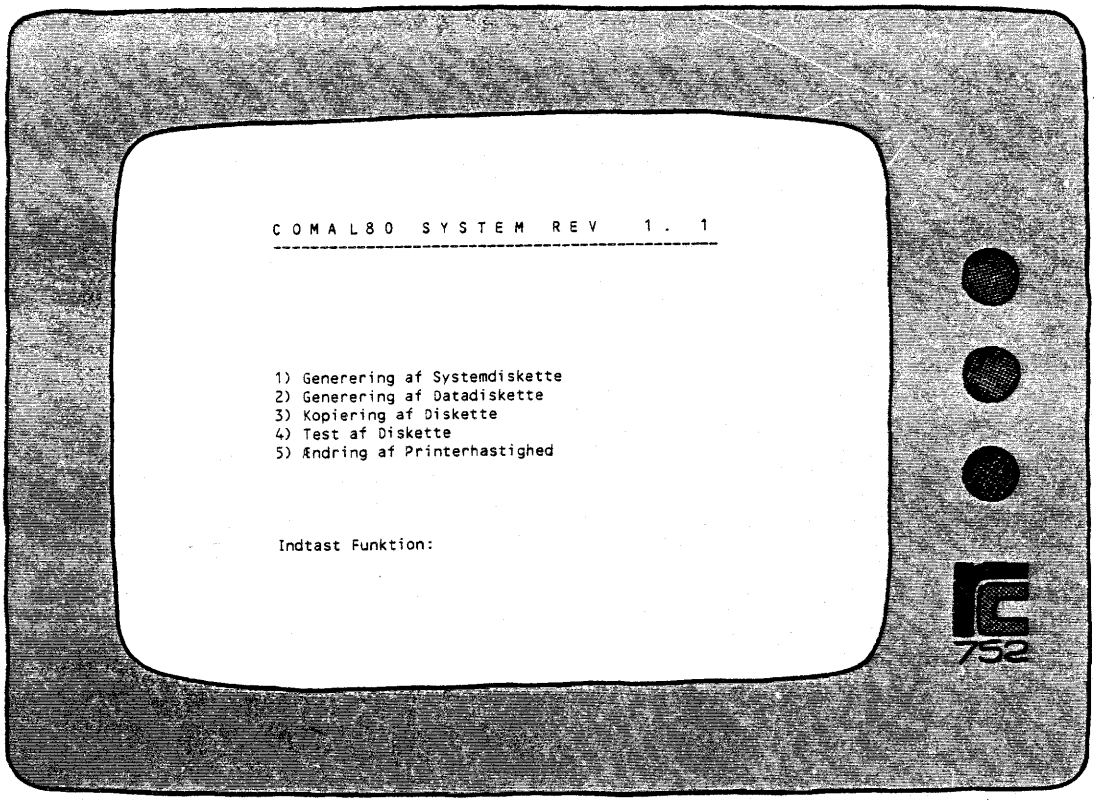
D. DISTRIBUTIONSDISKETTEN D.

COMAL80-systemet distribueres på en enkelt 8" eller 5 1/4" diskette. På denne diskette ligger et program til oprettelse af COMAL80 systemdisketter samt til kopiering og generering af disketter.

Opstartsproceduren er følgende :

1. Tænd for strømmen på RC700.
2. Indsæt distributionsdisketten i diskettestation 1
3. Tryk på RESET-knappen.

Distributionsprogrammet startes nu op, og følgende tekst bliver udskrevet på skærmen :



Den ønskede funktion udvælges ved indtastning af det tilhørende nummer efterfulgt af tryk på vognretur-tasten.

D.1 Oprettelse af systemdiskette D.1

For at kunne danne en systemdiskette, skal man have en tom diskette, eller en diskette, der gerne må slettes.

D.1.1 Oprettelse af systemdiskette på RC700 med 1 disketteenhed D.1.1

Før man kan lægge et system på en tom diskette, skal den altid først gøres til datadiskette.

Dette gøres ved at følge følgende opskrift:

1. TAST: 2 efterfulgt af vognretur
2. Ny menu: Generering af datadiskette

```
Angiv diskettestation (1/2)                    1
```

3. TAST: vognretur
4. TAG distributionsdiskette ud af disketteenhed
5. INDSÆT tom diskette i disketteenhed
6. TAST: vognretur

Herefter slettes disketten i disketteenheden, og når dette er sket, udskriver programmet atter hovedmenuen på skærmen. Disketten i disketteenheden vil nu kunne benyttes som datadiskette.

Nu skal der lægges system på disketten. Det gøres ved at følge denne opskrift:

1. TAST: 1 efterfulgt af vognretur
2. Ny menu: Generering af systemdiskette

Generering på diskettestation (1/2) 2

3. TAST: 1 efterfulgt af vognretur
4. Linie: Distributionsdisketten i diskettestation (1/2) 1
5. TAST: vognretur
6. Linie: Indsæt distributionsdiskette, tast vognretur
7. GØR som der bliver bedt om
8. Linie: Indsæt kopidiskette, tast vognretur
9. GØR som der bliver bedt om
10. Systemet vil nu hoppe tilbage til trin 6, sålænge det er nødvendigt. Når den nye systemdiskette er genereret, vender systemet tilbage til hovedmenuen.

D.1.2 Oprettelse af systemdiskette på RC700 med 2 disketteenheder D.1.2

Før man kan lægge et system på en tom diskette, skal den altid først gøres til datadiskette.

Dette gøres ved at følge følgende opskrift:

1. TAST: 2 efterfulgt af vognretur
2. Ny menu: Generering af datadiskette

Angiv diskettestation (1/2) 1

3. TAST: 2 efterfulgt af vognretur
4. INDSÆT tom diskette i disketteenhed nr 2
5. TAST: vognretur

Herefter slettes (formatteres) disketten i disketteenhed nr 2, og når dette er sket, udskriver programmet atter hovedmenuen igen. Disketten i disketteenhed nr 2 kan nu benyttes som datadiskette.

Nu skal der lægges system på disketten. Det gøres ved at følge følgende opskrift:

1. TAST: 1 efterfulgt af vognretur
2. Ny menu: Generering af systemdiskette

Generering på diskettestation (1/2) 2

3. TAST: vognretur
4. Linie: Distributionsdisketten i diskettestation (1/2) 1
5. TAST: vognretur
6. Linie: Indsæt distributionsdiskette, tast vognretur
7. KONSTATER: at distributionsdisketten stadig sidder i enhed nr 1
8. TAST: vognretur
9. Linie: Indsæt kopidiskette, tast vognretur
10. KONSTATER: at kopidisketten stadig sidder i enhed nr 2
11. TAST: vognretur

Herefter lægges der system over på disketten i disketteenhed nr 2. Når disketten er genereret, vender systemet tilbage til hovedmenuen.

D.2 Gennemgang af distributionsprogrammets faciliteter D.2

Ud fra hovedmenuen kan man vælge 5 funktioner, der benyttes i forbindelse med vedligeholdelse af systemet. Fælles for de 5 funktioner er, at brugeren først angiver en række parametre. Derefter beder programmet om de nødvendige disketter og det udfører den ønskede funktion.

Generelt kan siges, at under indtastningen af parametre udskriver programmet nogle standardværdier udfor de ønskede størrelser. Hvis brugeren ønsker standardparametrene, skal der blot tastes vognretur, ellers tastes det ønskede svar efterfulgt af vognretur.

D.2.1 Generering af systemdiskette

Funktionen benyttes til at lægge COMAL80 systemfilerne ned på en enten nyformatteret datadiskette eller på en diskette hvorpå der ligger et gammelt system og/eller brugerfiler.

BEMÆRK!!! Disketten skal være formatteret før funktionen benyttes.

Parametre	Std værdier
Generering på diskettestation (1/2)	2
Distributionsdisketten i diskettestation (1/2)	1

D.2.2 Generering af datadiskette

Funktionen benyttes til at formattere disketter. Når en diskette er formatteret, kan den benyttes som COMAL80 datadiskette.

5 1/4 " disketter skal altid formatteres før brug, hvorimod 8 " disketter leveres formatterede. Det anbefales dog ligeledes at formattere tomme 8 " disketter efter modtagelsen.

Parametre	Std værdier
Angiv diskettestation (1/2)	1

D.2.3 Kopiering af diskette

Funktionen benyttes til at kopiere hele disketter. Ønsker man kun at kopiere enkelte filer skal COMAL80 kommandoen COPY benyttes.

Parametre	Std værdier
Kopiering fra diskettestation (1/2)	1
Kopiering til diskettestation (1/2)	2

D.2.4 Test af diskette

Man kan vælge mellem to forskellige testformer:

- checklæsning af adressemærker
- udfør skriv/læse test

Checklæsning af adressemærker kan foretages på alle formatterede diskette uden data går tabt, da funktionen kun læser adressemærkerne på disketten.

Udfør skriv/læse test er en generel medie- og systemtest. Testen må kun udføres på nyformatterede disketter eller på formatterede disketter, hvis indhold gerne må slettes. Testen udskriver først et testdatamønster på disketten og derefter checklæses disse data.

Parametre	Std værdier
Angiv diskettestation (1/2)	1

D.2.5 Ændring af printerhastighed

Hvis man benytter en printer, der ikke kører 1200 baud er det nødvendigt at ændre hastigheden i systemet. Man kan vælge mellem at ændre hastigheden på distributionsdisketten eller på de enkelte systemdisketter.

Parametre	Std værdier
Systemdiskette i diskettestation	1

Når man har angivet diskettestationen udskrives:

Indsæt diskette, tast vognretur

Når dette er foretaget vil systemet udskrive en oversigt over de hastigheder man kan vælge. Så snart man har valgt hastighed udskrives informationen på disketten.

D.3 Fejlmeddelelser D.3

Fra distributionsprogrammet kan man få to typer fejlmeddelelser:

- fejl fundet under indtastning af parametre
- fejl fundet under udførelse af funktion

fejl fundet under indtastning af parametre har følgende format :

* ulovlig parameter * : <ulovlig parameter>

og dette betyder at brugeren ikke har indtastet en "lovlig" værdi. Programmet fortsætter med at skrive * ulovlig parameter * indtil brugeren indtaster en acceptabel parameter.

fejl fundet under udførelse af funktion vil blive ledsaget af en tekst, der beskriver hvor på disketten fejlen er fundet. Desuden beder programmet om reaktion på fejlen, idet det udskriver:

Indtast funktion: g(entag,f(ortsæt,s(lut:

g betyder gentagelse af den operation, der medførte fejlen
f betyder fortsættelse med den næste operation,
s betyder afbrydelse af funktionen.

E. ASCII tegnsættet E.

ASCII tegnene samt deres decimale og oktale værdier

DEC	OKT	TEGN	DEC	OKT	TEGN	DEC	OKT	TEGN	DEC	OKT	TEGN
0	000	NUL	32	040	SP	64	100	ü	96	140	ä
1	001	SOH	33	041	!	65	101	A	97	141	a
2	002	STX	34	042	"	66	102	B	98	142	b
3	003	ETX	35	043	#	67	103	C	99	143	c
4	004	EOT	36	044	$	68	104	D	100	144	d
5	005	ENQ	37	045	%	69	105	E	101	145	e
6	006	ACK	38	046	&	70	106	F	102	146	f
7	007	BEL	39	047	'	71	107	G	103	147	g
8	010	BS	40	050	(	72	110	H	104	150	h
9	011	HT	41	051	)	73	111	I	105	151	i
10	012	LF	42	052	*	74	112	J	106	152	j
11	013	VT	43	053	+	75	113	K	107	153	k
12	014	FF	44	054	,	76	114	L	108	154	l
13	015	CR	45	055	-	77	115	M	109	155	m
14	016	SO	46	056	.	78	116	N	110	156	n
15	017	SI	47	057	/	79	117	O	111	157	o
16	020	DLE	48	060	0	80	120	P	112	160	p
17	021	DC1	49	061	1	81	121	Q	113	161	q
18	022	DC2	50	062	2	82	122	R	114	162	r
19	023	DC3	51	063	3	83	123	S	115	163	s
20	024	DC4	52	064	4	84	124	T	116	164	t
21	025	NAK	53	065	5	85	125	U	117	165	u
22	026	SYN	54	066	6	86	126	V	118	166	v
23	027	ETB	55	067	7	87	127	W	119	167	w
24	030	CAN	56	070	8	88	130	X	120	170	x
25	031	EM	57	071	9	89	131	Y	121	171	y
26	032	SUB	58	072	:	90	132	Z	122	172	z
27	033	ESC	59	073	;	91	133	Æ	123	173	æ
28	034	FS	60	074	<	92	134	Ø	124	174	ø
29	035	GS	61	075	=	93	135	Å	125	175	å
30	036	RS	62	076	>	94	136	↑	126	176	ö
31	037	US	63	077	?	95	137	_	127	177	DEL

F. STORE PROGRAMEKSEMPLER F.

Dette afsnit indeholder en række store programeksempler, der skulle illustrere nogle af de mange muligheder man har for programmering i COMAL80.

F.1 Enarmet tyveknægt F.1

Det første store programeksempel er et spil. Programmet fungerer ligesom en enarmet tyveknægt, bortset fra, at det er lidt for rundhåndet med gevinster.

Programmet er ret stort, men det giver også spilleren mulighed for at holde de enkelte hjul for at forbedre sine gevinstchancer. Ved hjælp af skærmstyringen illuderer programmet de tre hjul, der drejer rundt.

Programmet benytter det semigrafiske tegnsæt til at danne spillesymbolerne. Disse hentes ind fra disketten. Før man kører programmet skal man derfor danne symbolerne. Det gøres med følgende program:

```
0010 OPEN FILE 1,"tyvsymbol", WRITE
0020 DIM text$ OF 18
0030 // Definering af grafikstrenge
0040 FOR i:=0 TO 5 DO
0050   FOR j:=1 TO 7 DO
0060     FOR k:=1 TO 18 DO
0070       READ tegnværdi
0080       text$(k:k):=CHR$(tegnværdi)
0090     NEXT k
0100     WRITE FILE 1:text$
0110   NEXT j
0120 NEXT i
0130 CLOSE
0140
0150 // grafik for 'BAR'
0160 DATA 32,124,124,124,124,124,124,124,124
```

```
0170 DATA 124,124,124,124,124,124,124,124,32
0180 DATA 32,127,35,35,35,43,127,39,35
0190 DATA 35,43,127,35,35,35,43,127,32
0200 DATA 32,127,32,127,53,32,127,32,127
0210 DATA 127,32,127,32,127,127,32,127,32
0220 DATA 32,127,32,115,49,40,127,32,115
0230 DATA 115,32,127,32,99,115,112,127,32
0240 DATA 32,127,32,127,53,32,127,32,127
0250 DATA 127,32,127,32,116,43,127,127,32
0260 DATA 32,127,112,112,112,120,127,112,127
0270 DATA 127,112,127,112,127,125,114,127,32
0280 DATA 32,47,47,47,47,47,47,47,47
0290 DATA 47,47,47,47,47,47,47,47,32
0300
0310 // grafik for 'KLØR'
0320 DATA 32,32,32,32,32,32,32,96,124
0330 DATA 124,48,32,32,32,32,32,32,32
0340 DATA 32,32,32,32,32,32,32,43,127
0350 DATA 127,39,32,32,32,32,32,32,32
0360 DATA 32,32,32,32,112,112,32,32,106
0370 DATA 53,32,32,112,112,32,32,32,32
0380 DATA 32,32,32,126,127,127,125,124,126
0390 DATA 125,124,126,127,127,125,32,32,32
0400 DATA 32,32,32,34,47,47,33,32,106
0410 DATA 53,32,34,47,47,33,32,32,32
0420 DATA 32,32,32,32,32,32,32,32,122
0430 DATA 117,32,32,32,32,32,32,32,32
0440 DATA 32,32,32,32,32,96,120,126,127
0450 DATA 127,125,116,48,32,32,32,32,32
0460
0470 // grafik for 'KLOKKE'
0480 DATA 32,32,32,32,32,120,126,127,127
0490 DATA 127,127,125,116,32,32,32,32,32
0500 DATA 32,32,32,32,106,127,127,127,127
0510 DATA 127,127,127,127,53,32,32,32,32
0520 DATA 32,32,32,32,106,127,127,127,127
0530 DATA 127,127,127,127,53,32,32,32,32
0540 DATA 32,32,32,32,106,127,127,127,127
0550 DATA 127,127,127,127,53,32,32,32,32
```

```
0560 DATA 32,32,32,32,106,127,127,127,127
0570 DATA 127,127,127,127,53,32,32,32,32
0580 DATA 32,32,112,124,127,127,127,127,127
0590 DATA 127,127,127,127,127,124,112,32,32
0600 DATA 32,32,32,32,32,32,32,43,127
0610 DATA 127,39,32,32,32,32,32,32,32
0610
0620 // grafik for 'BLOMME'
0630 DATA 32,32,32,32,32,32,32,32,32
0640 DATA 32,32,32,32,32,32,32,32,32
0650 DATA 32,32,32,32,112,124,124,127,127
0660 DATA 127,127,124,124,112,32,32,32,32
0670 DATA 32,32,120,127,127,127,127,127,127
0680 DATA 127,127,127,127,127,127,116,32,32
0690 DATA 32,32,127,127,127,127,127,127,127
0700 DATA 127,127,127,127,127,127,127,32,32
0710 DATA 32,32,43,127,127,127,127,127,127
0720 DATA 127,127,127,127,127,127,39,32,32
0730 DATA 32,32,32,32,35,47,47,127,127
0740 DATA 127,127,47,47,35,32,32,32,32
0750 DATA 32,32,32,32,32,32,32,32,32
0760 DATA 32,32,32,32,32,32,32,32,32
0770
0780 // grafik for 'ÆBLE'
0790 DATA 32,32,32,32,32,32,32,32,32
0800 DATA 32,96,38,32,32,32,32,32,32
0810 DATA 32,32,32,32,32,32,32,32,96
0820 DATA 56,33,32,32,32,32,32,32,32
0830 DATA 32,32,32,32,96,112,112,32,106
0840 DATA 96,112,112,112,32,32,32,32,32
0850 DATA 32,32,32,122,127,127,127,127,127
0860 DATA 127,127,127,127,127,48,32,32,32
0870 DATA 32,32,32,127,127,127,127,127,127
0880 DATA 127,127,127,127,127,53,32,32,32
0890 DATA 32,32,32,111,127,127,127,127,127
0900 DATA 127,127,127,127,127,33,32,32,32
0910 DATA 32,32,32,32,43,47,127,127,127
0920 DATA 127,127,47,39,33,32,32,32,32
0930
```

```
0940 // grafik for 'KIRSEBÆR'
0950 DATA 32,32,32,32,32,32,34,100,32
0960 DATA 96,52,32,32,32,32,32,32,32
0970 DATA 32,32,32,32,32,32,32,42,100
0980 DATA 37,32,32,32,32,32,32,32,32
0990 DATA 32,32,32,32,32,32,32,96,37
1000 DATA 41,112,112,112,32,32,32,32,32
1010 DATA 32,32,32,32,32,112,112,53,32
1020 DATA 104,127,127,127,125,32,32,32,32
1030 DATA 32,32,32,32,126,127,127,127,52
1040 DATA 34,111,127,127,39,32,32,32,32
1050 DATA 32,32,32,32,43,127,127,63,33
1060 DATA 32,32,32,32,32,32,32,32,32
1070 DATA 32,32,32,32,32,32,32,32,32
1080 DATA 32,32,32,32,32,32,32,32,32
```

Herefter følger det egentlige program:

```
0010 PROC esc HANDLER
0020   PRINT AT(1,22);
0030   IF ERR=100 THEN
0040     IF NOT esc_enabled THEN CONTINUE
0050     IF penge>=0 THEN
0060       PRINT CHR$(128);"    Det var hyggeligt, lad mig prøve at slå dig";
0070       PRINT " en anden gang !!";
0080     ELSE
0090       PRINT CHR$(128);"    Det var hyggeligt at vinde over dig !!";
0100     ENDIF
0110   ENDIF
0120 ENDPROC esc
0130 PROC opstart
0140   RANDOMIZE
0150   esc_enabled:=FALSE; maxi:=17
0160   ENABLE esc
0170   DIM text$(42) OF 18
0180   DIM svar$ OF 1,pos(3),snur(3),hjul(maxi,3)
0190   DIM sletlinie$ OF 1
0200   MARGIN 0
```

```
0210  textmarg:=53
0220  penge:=0; antal:=0; sletlinie$:=CHR$(30)
0230  i1:=RND(1,maxi); i2:=RND(1,maxi); i3:=RND(1,maxi)
0240  bar:=7*0+1; klør:=7*1+1; klokke:=7*2+1; blomme:=7*3+1
0250  æble:=7*4+1; kirsebær:=7*5+1
0260  PRINT CHR$(12)
0270  PRINT AT(1,1);"           G E V I N S T L I S T E"
0280  PRINT AT(1,2);"-----------------------------------------------"
0290  PRINT AT(1,3);"! BAR        BAR        BAR          200 MØNTER !"
0300  PRINT AT(1,4);"! KLØR       KLØR       KLØR         150 MØNTER !"
0310  PRINT AT(1,5);"! KLOKKE     KLOKKE     KLOKKE/BAR    18 MØNTER !"
0320  PRINT AT(1,6);"! BLOMME     BLOMME     BLOMME/BAR    14 MØNTER !"
0330  PRINT AT(1,7);"! ÆBLE       ÆBLE       ÆBLE/BAR      10 MØNTER !"
0340  PRINT AT(1,8);"! KIRSEBÆR   KIRSEBÆR   KIRSEBÆR/BAR  10 MØNTER !"
0350  PRINT AT(1,9);"! KIRSEBÆR   KIRSEBÆR   ----------     5 MØNTER !"
0360  PRINT AT(1,10);"! KIRSEBÆR   -------    ----------     2 MØNTER !"
0370  PRINT AT(51,2);"-----------------------------"
0380  FOR i:=3 TO 10 DO PRINT AT(80,i);"!";
0390  PRINT AT(1,11);
0400  FOR i:=1 TO 49 DO PRINT "-";
0410  PRINT " ";
0420  FOR i:=51 to 80 DO PRINT "-";
0430  PRINT CHR$(140)
0440  x1:=8; x2:=313; x3:=54
0450  PRINT AT(1,14);CHR$(132);AT(1,22);CHR$(128);
0460  PRINT AT(textmarg,6);" Vent lige lidt,tak"
0470  // definering af grafikstrenge
0480  OPEN 1,"tyvsymbol", READ
0490  FOR i:=1 TO 42 DO READ FILE 1:text$(i)
0500  CLOSE
0510  FOR i:=1 TO 3 DO
0520    FOR j:=1 TO 17 DO READ hjul(j,i)
0530  NEXT i
0540  DATA 36,29,22,36,15,29,8,36,1,22,15,29,8,36,22,29,1
0550  DATA 22,29,8,36,15,29,1,36,22,29,8,36,22,1,29,15,36
0560  DATA 1,29,15,22,36,8,29,22,36,1,29,15,36,8,22,29,36
0570  PRINT AT(x1-3,13);CHR$(192+0);AT(x3+20,13);CHR$(192+1);
0580  PRINT AT(x1-3,21);CHR$(192+1);AT(x3+20,21);CHR$(192+3);
0590  FOR i:=1 TO 68 DO
```

```
0600      PRINT AT(x1-3+i,13);chr$(200);AT(x1-3+i,21);chr$(200);
0610    NEXT i
0620    FOR i:=1 TO 7 DO
0630      FOR j:=0 TO 3 DO PRINT AT(j*23+5,13+i);CHR$(201);
0640    NEXT i
0650    PRINT AT(x1+20,13);CHR$(196);AT(x3-3,13);CHR$(196);
0660    PRINT AT(x1+20,21);CHR$(199);AT(x3-3,21);CHR$(199);
0670    holdt:= FALSE
0680    FOR i:= 1 TO 3 DO snur(i):= TRUE
0690    PRINT AT(textmarg,3);"Tast 1,2 eller 3 for HOLD",
0700    PRINT AT(textmarg,4);"kun når  H O L D ",CHR$(128),"lyser    ";
0710 ENDPROC opstart
0720
0730 PROC snurhjul
0740    antal:= antal+1; penge:= penge-1
0750    h1:=RND(3,8); h2:=RND(h1+1,h1+5); h3:=RND(h2+1,h2+5)
0760    h1:=h1*snur(1); h2:=h2*snur(2); h3:=h3*snur(3)
0770    REPEAT
0780      i1:=(i1+snur(1)) MOD maxi; h1:=h1-snur(1); snur(1):=h1>0
0790      i2:=(i2+snur(2)) MOD maxi; h2:=h2-snur(2); snur(2):=h2>0
0800      i3:=(i3+snur(3)) MOD maxi; h3:=h3-snur(3); snur(3):=h3>0
0810      t1:=hjul(i1+1,1); t2:=hjul(i2+1,2); t3:=hjul(i3+1,3)
0820      FOR linie:=14 TO 20 DO
0830        PRINT AT(x1,linie),text$(t1),
0840        PRINT AT(x2,linie),text$(t2),
0850        PRINT AT(x3,linie),text$(t3),
0860        t1:=t1+1; t2:=t2+1; t3:=t3+1
0870      NEXT linie
0880    UNTIL h1+h2+h3=0
0890    t1:=t1-7; t2:=t2-7; t3:=t3-7
0900 ENDPROC snurhjul
0910
0920 PROC gevinst
0930    vundet:= 0
0940    CASE t1 OF
0950    WHEN bar
0960      IF t2=bar AND t3=bar THEN vundet:=200
0970    WHEN klør
0980      IF t2=klør AND t3=klør THEN vundet:=150
```

```
0990    WHEN klokke
1000      IF t2=klokke AND (t3=klokke OR t3=bar) THEN vundet:=18
1010    WHEN blomme
1020      IF t2=blomme AND (t3=blomme OR t3=bar) THEN vundet:=14
1030    WHEN æble
1040      IF t2=æble AND (t3=æble OR t3=bar) THEN vundet:=10
1050    WHEN kirsebær
1060      vundet:=2
1070      IF t2=kirsebær THEN
1080        vundet:=5
1090        IF t3=kirsebær THEN vundet:=10
1100      ENDIF
1110    ENDCASE
1120    IF vundet>0 THEN
1130      PRINT AT(textmarg,6);"Du har vundet ";vundet;
1140      PRINT " mønter.";
1150      penge:= penge+vundet
1160    ELSE
1170      PRINT AT(textmarg,6);"Du fik ingen gevinst        "
1180    ENDIF
1190    IF penge>=0 THEN
1200      PRINT AT(textmarg,9);"Gevinst ialt: "
1210      PRINT AT(textmarg,10);penge;" mønter";
1220    ELSE
1230      PRINT AT(textmarg,9);"Underskud ialt: "
1240      PRINT AT(textmarg,10);-penge;" mønter";
1250    ENDIF
1260    PRINT " på ";antal;" spil. "
1270  ENDPROC gevinst
1280
1290  PROC holdhjul
1300    FOR i:=x1 TO x3 STEP x2-x1 DO
1310      PRINT AT(i+2,22);"               ";chr$(128);
1320    NEXT i
1330    FOR i:=1 TO 3 DO snur(i):=TRUE
1340    IF NOT holdt AND vundet=0 THEN
1350      holdt:=FALSE
1360      PRINT AT(textmarg+7,4),CHR$(144),
1370      REPEAT
1380        REPEAT
1390          REPEAT
```

```
1400          svar$:=KEY$
1410        UNTIL ORD(svar$)>0
1420      UNTIL svar$ IN "123" OR ORD(svar$)=13
1430      IF svar$ IN "123" THEN
1440        i:=ORD(svar$)-ORD("0")
1450        snur(i):=NOT snur(i)
1460        IF NOT snur(i) THEN
1470          PRINT AT(pos(i)+2,22);chr?(144),"  H O L D    ",chr$(128)
1480        ELSE
1490          PRINT AT(pos(i)+2,22);"              ";
1510        ENDIF
1520      ENDIF
1530    UNTIL svar$=CHR$(13)
1540    FOR i:=1 TO 3 DO
1550      holdt:=holdt OR snur(i)=0
1560    NEXT i
1570  ELSE
1580    holdt:= FALSE
1590    REPEAT
1600      svar$:=KEY$
1610    UNTIL ORD(svar$)=13
1620  ENDIF
1630 ENDPROC holdhjul
1640
1650 MARGIN 0
1660 ZONE 0
1670 EXEC opstart
1680 REPEAT
1690   PRINT AT(textmarg,6);" Ok nu kører vi        "
1700   esc_enabled:=FALSE
1710   EXEC snurhjul
1720   EXEC gevinst
1730   esc_enabled:=TRUE
1740   EXEC holdhjul
1750 UNTIL FALSE
```

F.2 Tænker du på et dyr ? F.2

Andet programeksempel er også et spil. RC700 stiller spilleren spørgsmål og forsøger at gætte hvilket dyr, spilleren tænker på. Hvis RC700 gætter forkert, vil det spørge om forskellen mellem det dyr, den tænkte på, og det dyr spilleren tænkte på. Dermed kender RC700 endnu et dyr.

Programmet her har den ulempe, at det ikke kan huske dyrene fra RUN til RUN.

```
0010 DIM SVAR$ OF 30, nytsvar$ OF 30, ind$ OF 30// strengvariabel
0020 DIM spørgsmål$(100) OF 30//                   teksttabel
0030 DIM træ(100,3)//                              matrix
0040 ophav:= 3; venst:= 1; højre:= 2; max:= 2; dommedag:= FALSE
0050 spørgsmål$(2):= "en elefant"
0060 træ(1,venst):= 2; træ(2,ophav):= 1
0070 REPEAT
0080   REPEAT
0090     INPUT "Tænker du på et dyr? ": svar$
0100     IF svar$="nej" THEN GOTO farvel
0110   UNTIL svar$="ja"
0120   knude:= træ(1,venst); slut:= FALSE
0130   REPEAT
0140     REPEAT
0150       PRINT spørgsmål$(knude);
0160       INPUT " ? ": svar$
0170     UNTIL svar$="ja" OR svar$="nej"
0180     IF svar$"ja" THEN
0190       slut:= (NOT træ(knude,venst))
0200       IF træ(knude,venst) THEN
0210         knude:= træ(knude,venst)
0220       ENDIF
0230     ELSE
0240       slut:= (NOT træ(knude,højre)
0250       IF træ(knude,højre) THEN
0260         knude:= træ(knude,højre)
0270       ELSE
```

```
0280          EXEC indsknude
0290        ENDIF
0300      ENDIF
0310    UNTIL slut
0320 UNTIL dommedag
0330 farvel:
0340 PRINT "Nå men så farvel for denne gang"
0350 END
0360 PROC indsknude
0370   max:= max+1
0380   parent:= træ(knude,ophav)
0390   IF træ(parent,venst)=knude THEN
0400     træ(parent,venst):=max
0410   ELSE
0420     træ(parent,højre):= max
0430   ENDIF
0440   træ(knude,ophav):= max
0450   INPUT "Hvad er det så ? ": nytsvar$
0460   PRINT "Hvad skal jeg spørge om for at kende forskel på"
0470   PRINT spørgsmål$(knude);" og ";nytsvar$
0480   INPUT " ? ": ind$
0490   REPEAT
0500     PRINT "og hvad er svaret for ";nytsvar$
0510     INPUT " ? ": svar$
0520   UNTIL svar$="ja" OR SVAR$="nej"
0530   spørgsmål$(max):= ind$
0540   træ(max,ophav):= parent
0550   træ(max,venst):= (max+1)*(svar$="ja")+knude*(svar$="nej")
0560   træ(max,højre):= knude*(svar$"ja")+(max+1)*(svar$="nej")
0570   max:= max+1
0580   spørgsmål$(max):= nytsvar$
0590   træ(max,ophav):= max-1
0600 ENDPROC indsknude
```

```
RUN
Tænker du på et dyr ? ja
en elefant ? nej
Hvad er det så ? en hund
Hvad skal jeg spørge om for at kende forskel på
en elefant og en hund ? har det en snabel
og hvad er svaret for en hund ? nej
Tænker du på et dyr ? ja
har det en snabel ? nej
en hund ? nej
Hvad er det så ? en kat
Hvad skal jeg spørge om for at kende forskel på
en hund og en kat ? får det killinger
og hvad er svaret for en kat ? ja
Tænker du på et dyr ? ja
har det en snabel ? nej
får det killinger ? ja
en kat ? ja
Tænker du på et dyr ? nej
Nå men så farvel for denne gang
END
at 0340
```

F.3 Tænker du på et dyr ? (filudgave) F.3

Tredje programeksempel svarer til det andet, bortset fra, at de tidligere dyr og spørgsmål gemmes i en fil. Før programmet kan udføres, skal datafilen oprettes. Det gøres med følgende program

```
0010 CREATE "anima",100 // tallet bestemmes helt af brugeren
0020 OPEN 1,"anima",RANDOM 64
0030 WRITE FILE 1,1:"",2,0,2
0040 WRITE FILE 1,2:"en elefant",0,0,1
0050 CLOSE
```

Herefter følger det egentlige program:

```
0010 DIM SVAR$ OF 38, nytsvar$ OF 38, ind$ OF 38// strengvariable
0020 DIM spørgsmål$ OF 38
0030
0040 dommedag:= FALSE
0050 OPEN FILE 1,"anima", RANDOM 64
0060 READ FILE 1,1: spørgsmål$,knude,højre,max
0070 REPEAT
0080   READ FILE 1,1: spørgsmål$,knude,højre,ophav
0090   REPEAT
0100     INPUT "Tænker du på et dyr? ": svar$
0110     IF svar$="nej" THEN GOTO farvel
0120   UNTIL svar$="ja"
0130   slut:= FALSE
0140   REPEAT
0150     READ FILE 1,knude: spørgsmål$,venst,højre,ophav
0160     REPEAT
0170       PRINT spørgsmål$;
0180       INPUT " ? ": svar$
0190     UNTIL svar$="ja" OR svar$="nej"
0200     IF svar$="ja" THEN
0210       slut:= (NOT venst)
0220       IF venst THEN
0230         knude:= venst
0240       ENDIF
0250     ELSE
```

```
0260        slut:= (NOT højre)
0270        IF højre THEN
0280          knude:= højre
0290        ELSE
0300          EXEC indsknude
0310        ENDIF
0320      ENDIF
0330    UNTIL slut
0340  UNTIL dommedag
0350  farvel:
0360  PRINT "Nå men så farvel for denne gang"
0370  READ FILE 1,1: spørgsmål$,venst,højre,ophav
0380  WRITE FILE 1,1: spørgsmål$,venst,højre,max
0390  CLOSE
0400  END
0410  PROC indsknude
0420    max:=max+1
0430    parent:=ophav
0440    READ FILE 1,parent: spørgsmål$,venst,højre,ophav
0450    IF venst=knude THEN
0460      venst:=max
0470    ELSE
0480      højre:=max
0490    ENDIF
0500    WRITE FILE 1,parent: spørgsmål$,venst,højre,ophav
0510    READ FILE 1,knude: spørgsmål$,venst,højre,ophav
0520    ophav:=max
0530    INPUT "Hvad er det så ? ": nytsvar$
0540    PRINT "Hvad skal jeg spørge om for at kende forskel på"
0550    PRINT spørgsmål$;" og ";nytsvar$;
0560    INPUT " ? ": ind$
0570    WRITE FILE 1,knude: spørgsmål$,venst,højre,ophav
0580    REPEAT
0590      PRINT "og hvad er svaret for ";nytsvar$
0600      INPUT " ? ": svar$
0610    UNTIL svar$="ja" OR SVAR$="nej"
0620    spørgsmål$:=ind$
0630    ophav:=parent
0640    venst:=(max+1)*(svar$="ja")+knude*(svar$="nej")
```

```
0650   højre:=knude*(svar$="ja")+(max+1)*(svar$="nej")
0660   WRITE FILE 1,max: spørgsmål$,venst,højre,ophav
0670   max:=max+1
0680   spørgsmål$:=nytsvar$
0690   ophav:=max-1
0700   WRITE FILE 1,max: nytsvar$,0,0,ophav
0710 ENDPROC indsknude
```

F.4 Data-indtastning F.4

Her er et eksempel på en extern procedure til et procedurebibliotek.

Proceduren læser data fra skærmen i visse faste felter, som brugeren angiver. Parametrene til proceduren er følgende:

```
PROC dataentry(n,REF t$(),REF v$(),REF b$())
```

n : antal inputfelter på skærmen
t$: teksttabel indeholdende ledetekst
v$: teksttabel indeholdende standardværdier ved kald
og aktuelle værdier ved returnering
b$: teksttabel til beskrivelse af de enkelte felter, så
b$(i,1:1) = CHR$(x-koordinat til feltets start)
b$(i,2:2) = CHR$(y-koordinat til feltets start)
b$(i,3:3) = CHR$(inputfeltets maximale længde)

I selve proceduren kan man benytte cursorpilene, vognreturtasten og de almindelige taster. Man vender tilbage til hovedprogrammet ved at trykke på |< -tasten.

Hvis det lyder lidt indviklet så kør dette lille program, der benytter proceduren:

```
0010 PROC dataentry(n,REF t$(),REF v$(),REF b$()) EXTERNAL "de"
0020 ant:=4
0030 DIM tekst$(ant) OF 40,stdvær$(ant) OF 40,besk$(ant) OF 3
0040 FOR i:=1 TO ant DO
0050   READ tekst$(i),stdvær$(i),y,x,lgd
0060   besk$(i):=CHR$(x)+CHR$(y)+CHR$(lgd)
0070 NEXT i
0080 EXEC dataentry(ant,tekst$,stdvær$,besk$)
0090 PRINT CHR$(12)
0100 FOR i:=1 TO ant do PRINT stdvær$(i)
0110 DATA "Navn","",10,10,40
0120 DATA "Addr","",11,10,40
0130 DATA "By  ","",12,10,20
```

```
0140 DATA "Telf","(02)658000",20,10,10

0010 PROC dataentry(n,ref t$(),REF v$(),REF b$()) CLOSED
0020   PRINT CHR$(12)
0030   MARGIN 0
0040   DIM CH$ OF 1
0050   pilv:= 8; piln:= 10; pilh:= 24; pilo:= 26
0060   OPEN 1,"keyboard", READ
0070   FOR i:= 1 TO n DO
0080     PRINT AT(ORD(b$(i,1:1)),ORD(b$(i,2:2)));t$(i);
0090     FOR j:= 1 TO ORD(b$(i,3:3)) DO PRINT ".";
0100     PRINT AT(ORD(b$(i,1:1))+LEN(t$(i)),ORD(b$(i,2:2)));v$(i)
0110   NEXT i
0120   felt:= 1; pos:= 1
0130   PRINT AT(ORD(b$(felt,1:1))+LEN(t$(felt)),ORD(b$(felt,2:2)));
0140   REPEAT
0150     ch$:= GET$(1,1)
0160     char:= ORD(ch$)
0170     CASE TRUE OF
0180     WHEN char=pilv
0190       IF pos>1 THEN
0200         pos:= pos-1
0210         PRINT ch$;
0220       ENDIF
0230     WHEN char=pilh
0240       IF pos<=LEN(v$(felt)) AND pos<ORD(b$(felt,3:3)) THEN
0250         pos:= pos+1
0260         PRINT ch$;
0270       ENDIF
0280     WHEN char=pilo
0290       felt:= felt-1
0300       IF felt=0 THEN felt:= n
0310       PRINT AT(ORD(b$(felt,1:1))+LEN(t$(felt)),ORD(b$(felt,2:2)));
0320       pos:= 1
0330     WHEN char=piln,char=13
0340       felt:= felt+1
0350       IF felt>n THEN felt:= 1
0360       PRINT AT(ORD(b$felt,1:1))+LEN(t$(felt)),ORD(b$(felt,2:2)));
0370       pos:=1
```

```
0380      WHEN char>=32 AND char<=127
0390        IF pos<=ORD(b$(felt,3:3)) THEN
0400          v$(felt,pos:pos):= ch$
0410          pos:= pos+1
0420          PRINT ch$;
0430        ENDIF
0440      OTHERWISE
0450        PRINT CHR$(7);
0460      ENDCASE
0470    UNTIL char=5
0480    CLOSE 1
0490  ENDPROC dataentry
```

F.5 Sortering (PROC-EXTERNAL) F.5

Her er et andet eksempel på en extern procedure, der "hører hjemme" i et procedure-bibliotek.

Proceduren sorterer en fil, der består af en række strenge udskrevet med WRITE FILE. Proceduren sorterer filen i alfabetisk rækkefølge.

Parametrene til proceduren er:

PROC sort(ifil$,ufil$,lgd)

ifil$: inputfil, skrevet i ovenstående format
ufil$: outputfil, må ikke eksistere i forvejen
lgd : den maximale længde af strengen i filen

```
0010 PROC sort(ifil$,ufil$,lgd) CLOSED
0020   PROC quicksort(fra,til)
0030     s:=1; stak(1,1):=fra; stak(1,2):=til
0040     REPEAT
0050       venstr:=stak(s,1); højre:=stak(s,2); s:=s-1
0060       REPEAT
0070         i:=venstr; j:=højre; x$:=navne$((i+j) DIV 2)
0080         REPEAT
0090           WHILE navne$(i)<x$ DO i:=i+1
0100           WHILE x$<navne$(j) DO j:=j-1
0110           IF i<=j THEN
0120             sk$:=navne$(i); navne$(i):=navne$(j); navne$(j):=sk$
0130             i:=i+1; j:=j-1
0140           ENDIF
0150         UNTIL i>j
0160         IF j-venstr<højre-i THEN
0170           IF i<højre THEN
0180             s:=s+1; stak(s,1):=i; stak(s,2):=højre
0190           ENDIF
```

```
0200          højre:=j
0210        ELSE
0220          IF venstr<j THEN
0230            s:=s+1; stak(s,1):=venstr; stak(s,2):=j
0240          ENDIF
0250          venstr:=i
0260        ENDIF
0270      UNTIL venstr>=højre
0280    UNTIL s=0
0290  ENDPROC quicksort
0300
0310  PROC flet(slut)
0320    RENAME ufil$,"sort0001"
0330    pil1:=1; pil2:=1
0340    OPEN FILE 1,"sort0001", READ
0350    OPEN FILE 2,ufil$, WRITE
0360    READ FILE 1: nl$
0370    WHILE pil2<slut AND NOT EOF(1) DO
0380      IF nl$<navne$(pil2) THEN
0390        WRITE FILE 2: nl$
0400        READ FILE 1:nl$
0410      ELSE
0420        WRITE FILE 2: navne$(pil2)
0430        pil2:=pil2+1
0440      ENDIF
0450    ENDWHILE
0460    IF NOT EOF(1) THEN
0470      REPEAT
0480        WRITE FILE 2: nl$
0490        READ FILE 1: nl$
0500      UNTIL EOF(1)
0510    ELSE
0520      FOR i:=pil2 TO slut DO WRITE FILE 2: navne$(i)
0530    ENDIF
0540    CLOSE FILE 1
0550    CLOSE FILE 2
0560    DELETE "sort0001"
0570  ENDPROC flet
```

```
0580
0590   OPEN FILE 1,ufil$, WRITE
0600   CLOSE FILE 1
0610
0620   max:=INT((SYS(4)-4*lgd-1088)/(4+lgd))
0630   DIM navne$(max) OF lgd,n1$ OF lgd,n2$ OF lgd
0640   DIM x$ OF lgd,sk$ OF lgd,stak(10,2)
0650
0660   OPEN FILE 3,ifil$, READ
0670   antal:=0
0680   REPEAT
0690     n:=1
0700     WHILE (NOT EOF(3)) AND n<=max DO
0710       READ FILE 3: navne$(n)
0720       n:=n+1
0730     ENDWHILE
0740     n:=n-1
0750     IF EOF(3) THEN n:=n-1; antal:=antal-1
0760     IF n<>0 THEN
0770       EXEC quicksort(1,n)
0780       EXEC flet(n)
0790     ENDIF
0800   UNTIL EOF(3)
0810   CLOSE FILE 3
0820 ENDPROC sort
```

F.6 Semigrafik

RC700 har ikke fuld grafik, men kun de semigrafiske tegn, der er beskrevet i appendix C. Ved hjælp af disse kan man dog lave ganske pæne kurver og søjlediagrammer. Procedurerne i dette afsnit opdeler skærmen i 158 tegn på tværs (x-værdien) og 75 tegn i højden (y-værdien).

Procedurerne har følgende virkerfelter:

PROC setdot(x,y) "tænder" punktet (x,y)

PROC drawline(x0,y0,x1,y1) tegner en "linie" fra (x0,y0) til (x1,y1)

PROC init sætter skærmen i semigrafisk mode, og initialiserer de nødvendige variable

PROC move,moveto,turnto,turn fungerer ligesom plotterkommandoer, idet man angiver hvorledes "pennen" skal bevæge sig i forhold til aktuel position.

PROC move(x) tegner en linie, x enheder lang i aktuel retning

PROC moveto(x0,y0) tegner en linie fra aktuel position til (x0,y0).

PROC turnto(d) ændrer aktuel retning til d grader

PROC turn(d) ændrer aktuel retning med d grader

```
0010 PROC setdot(x,y)
0020   y:=75-y
0030   jjj:=y DIV 3; iii:=x DIV 2
0040   jjj:=jjj + 1; iii:=iii + 1
0050   c$:=screen$(jjj,iii:iii)
0060   bit:=tabel((y MOD 3)*2+x MOD 2+1)
```

```
0070    c$:=CHR$(ORD(c$) MOD bit+bit+(ORD(c$) DIV (bit*2))*bit*2)
0080    PRINT AT(iii+1,jjj);c$;
0090    screen$(jjj,iii:iii):=c$
0100  ENDPROC setdot
0110
0120  PROC drawline(x0,y0,x1,y1)
0130    EXEC setdot(x0,y0)
0140    CASE TRUE OF
0150    WHEN ABS(x1-x0)<ABS(y1-y0)
0160      dy:=(x1-x0)/(y1-y0)
0170      x:=x0
0180      FOR y:=y0 TO y1 STEP SGN(y1-y0) DO
0190        x:=x+dy
0200        EXEC setdot(x,y)
0210      NEXT y
0220    WHEN ABS(x1-x0)>ABS(y1-y0)
0230      dx:=(y1-y0)/(x1-x0)
0240      y:=y0
0250      FOR x:= x0 TO x1 STEP SGN(x1-x0) DO
0260        y:=y+dx
0270        EXEC setdot(x,y)
0280      NEXT x
0290    OTHERWISE
0300      y:=y0
0300      FOR x:=x0 TO x1 STEP SGN(x1-x0) DO
0320        y:=y+SGN(y1-y0)
0330        EXEC setdot(x,y)
0340      NEXT x
0350    ENDCASE
0360  ENDPROC drawline
0370
0380  PROC moveto(xkor,ykor)
0390    IF NOT penup THEN
0400      EXEC drawline(oldx,oldy,xkor,ykor)
0410    ENDIF
0420    oldx:=xkor; oldy:=ykor
0430  ENDPROC moveto
0440
0450  PROC move(x)
```

```
0460   xkor:=oldx+x*xangle; ykor:=oldy+x*yangle
0470   IF NOT penup THEN
0480     EXEC drawline(oldx,oldy,xkor,ykor)
0490   ENDIF
0500   oldx:=xkor; oldy:=ykor
0510 ENDPROC move
0520
0530 PROC turnto(d)
0540   deg:=d
0550   xangle:=COS(deg*pi/180); yangle:=SIN(deg*pi/180)
0560 ENDPROC turnto
0570
0580 PROC turn(d)
0590   deg:=(deg+d) MOD 360
0600   xangle:=COS(deg*pi/180); yangle:=SIN(deg*pi/180)
0610 ENDPROC turn
0620
0630 PROC init
0640   PRINT CHR$(12)
0650   DIM screen$(25) OF 79,space$ OF 40,c$ OF 1,tabel(6)
0660   FOR i:=1 TO 6 DO READ tabel(i)
0670   DATA 1,2,4,8,16,64
0680   oldx:=1; oldy:=1; pi:=ATN(1)*4
0690   deg:=0; xangle:=1; yangle:=0; penup:=FALSE
0700   MARGIN 0
0710   space$:="                                        "
0720   FOR i:=1 TO 25 DO
0730     screen$(i):=space$+space$
0740     PRINT AT(1,i);CHR$(132);
0750   NEXT i
0760 ENDPROC init
0770
```

F.7 Liv

Her er et eksempel på simulering af amøbers vækst. Udgangspunktet er et kvadrat, der er befolket med en række amøber. Man ændrer fordelingen af amøber efter visse regler, hvorefter man udskriver denne næste tilstand (generation). Således fortsættes et antal gange, enten indtil tilstanden er stationær eller et maximalt antal generationer er nået.

Reglerne for en amøbes udvikling er:

- hvis en amøbe har mere end 3 naboer, dør den af sult
- hvis en amøbe har mindre end 2 naboer, dør den af kedsomhed.
- hvis en amøbe har 2 eller 3 naboer, lever den videre
- hvis et tomt felt har præcis 3 naboer, fødes en ny amøbe.

```
0010 INPUT "Maximalt antal generationer >": maxgen
0020 READ stør
0030 DIM mat1(stør+2,stør+2),mat2(stør+1,stør+1)
0040
0050 gen:= 0
0060 stabil:= FALSE
0070 EXEC læsmat2
0080
0090 PRINT CHR$(12)
0100 EXEC udskriv
0110 WHILE gen<maxgen AND NOT stabil DO
0120   EXEC mat1ligmat2
0130   EXEC dannygeneration
0140   EXEC udskriv
0150 ENDWHILE
0160
0170 PROC læsmat2
0180   FOR i:= 2 TO stør+1 DO
0190     FOR j:= 2 TO stør+1 DO READ mat2(i,j)
0200   NEXT i
0210 ENDPROC læsmat2
0220
```

```
0230 PROC matlligmat2
0240   For i:= 2 TO stør+1 DO
0250     FOR j:= 2 TO stør+1 DO matl(i,j):= (mat2(i,j)<>0)
0260   NEXT i
0270 ENDPROC matlligmat2
0280
0290 PROC udskriv
0300   PRINT AT(1,1);"Generation nr.";gen
0310   FOR j:= 1 TO stør+2 DO PRINT "+";
0320   PRINT
0330   FOR i:= 2 TO stør+1 DO
0340     PRINT "+";
0350     FOR j:= 2 TO stør+1 DO
0360       IF mat2(i,j)=0 THEN
0370         PRINT " ";
0380        ELSE
0390          PRINT "*";
0400        ENDIF
0410     NEXT j
0420     PRINT "+"
0430   NEXT i
0440   FOR j:= 1 TO stør+2 DO PRINT "+";
0450   PRINT
0460 ENDPROC udskriv
0470
0480 PROC dannygeneration
0490   stabil:= TRUE
0500   FOR i:= 2 TO stør+1 DO
0510     venstre:=0
0520     midt:= matl(i-1,2)+matl(i,2)+matl(i+1,2)
0530     FOR j:= 2 TO stør+1 DO
0540       højre:= matl(i-1,j+1)+matl(i,j+1)+matl(i+1,j+1)
0550       n:= venstre+midt+højre-matl(i,j)
0560       CASE n OF
0570       WHEN 2
0580         mat2(i,j):= matl(i,j) // ændret tilstand
0590       WHEN 3
0600         mat2(i,j):= 1 // ny bakterie fødes
0610       OTHERWISE
```

```
0620          mat2(i,j):= 0
0630        ENDCASE
0640        IF mat1(i,j)<>mat2(i,j) THEN stabil:= FALSE
0650        venstre:= midt; midt:= højre
0660      NEXT j
0670    NEXT i
0680    gen:= gen+1
0690 ENDPROC dannygeneration
0700 DATA 5
0710 DATA 0,0,0,0,0
0720 DATA 0,1,1,1,0
0730 DATA 1,0,1,0,1
0740 DATA 0,1,1,1,0
0750 DATA 0,0,0,0,0
```

F.8 Tænk på et tal ! F.8

Dette programeksempel er også et spil. Spilleren skal tænke på et tal, hvorefter maskinen vil udskrive en række skemaer med tal, og spørge, om det tænkte tal er blandt de udskrevne. Efter at have udskrevet skemaer et antal gange, fortæller maskinen hvilket tal spilleren tænkte på.

```
0010 PROC opstart
0020   toerpotens:= 6
0030   max:= afrund(2↑toerpotens)
0040   PRINT CHR$(12);"Tænk på et tal mellem 0 og ";max-1
0050   DIM tal(max/2), svar$ OF 3
0060   ZONE 10
0070   MARGIN 0
0080 ENDPROC opstart
0090
0100 FUNC afrund(x)
0110   RETURN INT(x+0.5)
0120 ENDFUNC afrund
0130
0140 PROC udvælgtal(potens)
0150   offer:= afrund(2↑potens)
0160   pil:= 1
0170   FOR i:= 1 TO max DO
0180     IF (i DIV offer) MOD 2=1 THEN tal(pil):= i; pil:= pil+1
0190   NEXT i
0200 ENDPROC udvælgtal
0210
0220 PROC udskriv
0230   PRINT AT(1,2);CHR$(31)
0240   FOR i:= 1 TO max/2 DO PRINT tal(i),
0250   PRINT
0260 ENDPROC udskriv
0270
0280 EXEC opstart
0290 svaret:= 0
0300 FOR gang:= 0 TO toerpotens-1 DO
```

```
0310   EXEC udvælgtal(gang)
0320   EXEC udskriv
0330   REPEAT
0340     PRINT AT(1,20);CHR$(31);
0350     INPUT "Var tallet blandt de udskrevne (ja/nej) ": svar$
0360   UNTIL svar$="ja" OR svar$="nej"
0370   IF svar$="ja" THEN svaret:= svaret+afrund(2↑gang)
0380 NEXT gang
0390 PRINT "Du tænkte på ";svaret
0400 END
```

F.9 Eksempel fra procedureafsnittet F.9

Som lovet i afsnit 8 bringer vi her en total udskrift af en mulig løsning på problemet omkring elevregistreringen.

Først skal datafilerne oprettes. Det gøres f.eks. med følgende program:

```
0010 PRINT CHR$(12);"Oprettelse af kartoteksfiler"
0020 READ klasselgd,navnlgd,adresselgd
0030 READ postnrlgd,maxelev
0040
0050 DATA 5,40,40,40,500
0060
0070 DIM status$ OF maxelev
0080 FOR i:=1 TO maxelev DO status$:=status$+"L" // Ledig
0090
0100 DELETE "elevoplysn"
0110 DELETE "elevdata"
0120
0130 postlængde:=klasselgd+2+navnlgd+2+adresselgd+2+postnrlgd+2
0140 CREATE "elevdata",postlængde*maxelev/1024
0150 OPEN FILE 1,"elevoplysn", WRITE
0160 WRITE FILE 1: klasselgd,navnlgd,adresselgd,postnrlgd,maxelev
0170 WRITE FILE 1: status$
0180 CLOSE
0190
0200 PRINT "Slut på oprettelse"
0210 END
```

Hovedprogram:

```
0010 EXEC opstart
0020 REPEAT
0030   PRINT
0040   INPUT "Kommando: I(nd,R(et,S(let,F(ærdig ":k$
0050   PRINT
0060   CASE k$ OF
0070   WHEN "I","i"
0080     EXEC fåelevoplysn
```

```
0090      EXEC findledigtnr(elevnr)
0100      EXEC gemelev(elevnr)
0110    WHEN "R","r"
0120      EXEC fåelevnr
0130      EXEC hentelev(elevnr)
0140      EXEC retelev
0150      EXEC gemelev(elevnr)
0160    WHEN "S","s"
0170      EXEC fåelevnr
0180      EXEC hentelev(elevnr)
0190      EXEC sletelev(elevnr)
0200    WHEN "F","f"
0210      // slut på program
0220    OTHERWISE
0230      PRINT "*** Ulovlig kommando"
0240    ENDCASE
0250  UNTIL k$="F" OR k$="f"
0260  EXEC afslut
0270  END
0280
0290  PROC fåelevnr
0300    REPEAT
0310      INPUT "Elevnr : ":elevnr
0320      ok:=FALSE
0330      IF elevnr>=1 AND elevnr<=max THEN
0340        ok:=status$(elevnr:elevnr)="0") // optaget
0350      ENDIF
0360      IF NOT ok THEN PRINT "*** Eleven eksisterer ikke"
0370    UNTIL ok
0380  ENDPROC fåelevnr
0390
0400  PROC fåelevoplysn
0410    REPEAT
0420      INPUT "Klasse    :": klasse$
0430      INPUT "Navn      :": navn$
0440      INPUT "Adresse   :": adresse$
0450      INPUT "Postnr by :": postnrby$
0460      INPUT "Er oplysningerne korrekte ? (J/N) ": svar$
0470    UNTIL svar$="J" OR svar$="j"
```

```
0480 ENDPROC fåelevoplysn
0490
0500 PROC findledigtnr(REF nr)
0510   nr:=1
0520   WHILE nr<=max AND status$(nr:nr)="O" DO nr:=nr+1
0530   PRINT "Eleven har fået nummer ";nr
0540 ENDPROC findledigtnr
0550
0560 PROC hentelev(nr)
0570   READ FILE 1,nr: klasse$,navn$,adresse$,postnrby$
0580 ENDPROC hentelev
0590
0600 PROC gemelev(nr)
0610   WRITE FILE 1,nr: klasse$,navn$,adresse$,postnrby$
0620   status$(nr:nr):="O"
0630 ENDPROC gemelev
0640
0650 PROC sletelev(nr)
0660   EXEC skrivelev
0670   INPUT "Skal eleven slettes ? (J/N) ": svar$
0680   IF svar$="J" OR svar$="j" THEN status$(nr:nr):="S"
0690 ENDPROC sletelev
0700
0710 PROC skrivelev
0720   PRINT "Klasse    :";klasse$
0720   PRINT "Navn      :";navn$
0730   PRINT "Adresse   :";adresse$
0740   PRINT "Postnr by :";postnrby$
0750 ENDPROC skrivelev
0760
0770 PROC retelev
0780   REPEAT
0790     EXEC skrivelev
0800     INPUT "Ændring: K(lasse,N(avn,A(dresse,P(ostnr by,F(ærdig ":f$
0810     CASE f$ OF
0820     WHEN "K","k"
0830       INPUT "Ny klasse     ": klasse$
0840     WHEN "N","n"
0850       INPUT "Nyt navn      ": navn$
```

```
0860      WHEN "A","a"
0870        INPUT "Ny adresse    :": adresse$
0880      WHEN "P","p"
0890        INPUT "Ny postnr by  :": postnrby$
0900      OTHERWISE
0910        // Ingen aktion
0920      ENDCASE
0930    UNTIL f$="F" OR f$="f"
0940 ENDPROC retelev
0950
0960 PROC opstart
0970    PRINT CHR$(12);"K a r t o t e k s p r o g r a m"
0980    PRINT "------------------------------"
0990    DIM k$ OF 1,svar$ OF 1,f$ OF 1
1000    OPEN FILE 1,"elevoplysn", READ
1010    READ FILE 1: klasselgd,navnlgd,adresselgd,postnrlgd,max
1020    DIM klasse$ OF klasselgd,navn$ OF navnlgd
1030    DIM adresse$ OF adresselgd,postnrby$ OF postnrlgd
1040    DIM status$ OF max
1050    READ FILE 1:status$
1060    CLOSE FILE 1
1070    postlængde:=klasselgd+2+navnlgd+2+adresselgd+2+postnrlgd+2
1080    OPEN FILE 1,"elevdatya", RANDOM postlængde
1090    elevnr:=0
1100 ENDPROC opstart
1110
1120 PROC afslut
1130    CLOSE FILE 1
1140    OPEN FILE 1,"nyelevoplys", WRITE
1150    WRITE FILE 1: klasselgd,navnlgd,adresselgd,postnrlgd,max
1160    WRITE FILE 1: status$
1170    CLOSE FILE 1
1180    DELETE "elevoplysn"
1190    RENAME "nyelevoplys","elevoplysn"
1200    PRINT "Slut på kartoteksprogram"
1210 ENDPROC afslut
```

G. AUTOMATISK OPSTART AF PROGRAMMER G.

Ved opstart er det muligt, at få et program indlæst og udført automatisk, hvis programmes gemmes (SAVE) under navnet "logon".

Når COMAL80 startes op ledes efter en fil med navn "logon" på diskettestation nummer 1. Findes filen ikke, udskrives "COMAL80 rev x.xx" og systemet er klar til indtastning.

Findes filen derimod, indlæses programmet og startes umiddelbart.

LÆSERBEMÆRKNINGER

Titel: COMAL80 Brugervejledning RCSL Nr.: 42-i2179

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