CODEN: LUTFD2/(TFRT-5210)/1-044/(1978)

GENERELLA BERÄKNINGAR PÅ MÄTDATA

LARS-GÖRAN HALLSTRÖM

Department of Automatic Control
Lund Institute of Technology
Juni 1978

SAMMANFATTNING.

Denna rapport beskriver ett interaktivt programpaket med vars hjälp man kan göra generella beräkningar på mätdata,lagrade på massminne.Idén är i korthet att man genom att skriva satser (ungefär som i SIMNON) i en fil talar om hur beräkningarna ska utföras.Satserna kompileras sedan varvid en exekverbar pseudokod bildas.Genom att styra programmet via kommandon från en terminal får man beräkningarna utförda för de datafiler som anges i kommandots argument.

ABSTRACT.

This report describes an interactive program which performs computations on data files.The basic idea is that the user descrbes the computations by writing statements (similar to SIMNON statements) in a file.These statements are compiled and a pseudocode is created.By interpreting this code the program can perform the computations.The program is run by entering commands from a teletype.The arguments of the command consists of the names of the current data files.

INNEHÅLL:

1. INLEDNING.

Syftet med detta examensarbete har varit att skapa ett interaktivt program med vars hjälp man på ett enkelt sätt kan utföra generella beräkningar på mätdata lagrade på massminne. Genom att i en fil (när kallad beräkningsfilen) skriva deklarationer,tilldelningssatser,villkorssatser etc beskriver man de beräkningar man vill ha utförda på datafilen. Beräkningsfilen översättes till pseudokod av en kompilator,som i grunden bygger på SIMNON-kompilatorn.Jag har utgått ifrån denna kompilator där jag sedan tagit bort de delar som behandlar olika typer av system i SIMNON och tillfogat nya rutiner som tar hand om villkors- och repetitionssatser.Den är ochså modifierad så att indicerade variabler kan förekomma.De satser man skriver har alltså i princip samma struktur som satserna i SIMNON.Genom att styra exekveringen av programmet med kommandon från en terminal kan beräkningarna utföras för de datafiler som angetts i kommandot.

Arbetet har utförts vid instituttionen för reglerteknik, Lunds Tekniska Högskola,där Johan Wieslander varit handledare.

Programmen är skrivna i FORTRAN och körda på en PDP-15 dator. I de följande kapitlen visas först ett enkelt exempel följt av en beskrivning av beräkningsfilen och regler för hur den får se ut.Därpå beskrives de program som sköter kompileringen.Sedan följer en genomgång av tillgängliga kommandon och en kortfattad beskrivning av motsvarande program.I slutet finns några fullständiga exempel beskrivna.

2. ENKELT INLEDANDE EXEMPEL.

I en fil,i fortsättningen kallad beräkningsfil,beskriver man de operationer man vill göra på varje element i datafilerna. Antag att man vill begränsa storleken av sina data så att värden större än STOR ersättes med STOR och värden mindre än LITEN ersättes med LITEN.Man kan då skriva beräkningsfilen, som i fortsättningen antas ha namnet BERFI,så här:

```
OUTPUT UT              1
INPUT IN               2
STOR:300               3
LITEN:-300             4
IF IN>STOR THEN        5
IN=STOR                6
ENDIF
IF IN<LITEN THEN
IN=LITEN
ENDIF
UT=IN                  7
END
```

Satserna 1 och 2 är deklarationer av variabler som motsvarar elementen i datafilerna.Värdet av en variabel kan ändras i en tilldelningssats,t.ex. sats 6 och 7.Satserna 3 och 4 är parameterdeklarationer.Värdet av en parameter kan inte ändras. (Undantag:parameterblock,se kap.3.3)De beräkningar man vill göra beskriver man i satserna efter deklarationerna.Beräkningsfilen ska avslutas med END.

För att kompilera beräkningsfilen,varvid en exekverbar pseudokod och tabeller med variabel- och parametervärden bildas, används kommandot GENKOD.Man skriver (Namn skrivna med små bokstäver får man välja själv,max 5 bokstäver.Understrukna ord skrivs ut av programmet på skärmen.):

```
ENTER COMMAND:
GENKOD psfil←berfi
```

Härmed hamnar pseudokoden och tabellerna i filen med namnet PSFIL.

Antag nu att indata finns i filen FDAT1 och utdata ska skrivas i filen FDAT3.För att förbereda exekveringen av pseudo-

ENTER COMMAND:

GENOP fdat3←psfil fdat1

Härmed har man knutit ihop datafilerna med variabelnamnen UT och IN,avläst kod och tabellvärden i PSFIL och kontrollerat att antalet filer i kommandot stämmer överens med deklarationerna i beräkningsfilen.För att starta exekveringen skriver man därefter:

≥X

Resultatet av en körning visas nedan.

FDAT1:

99.0000	549.0000
-196.0000	0.0000
291.0000	59.0000
-384.0000	-116.0000
475.0000	171.0000
-564.0000	-224.0000
651.0000	275.0000
-736.0000	-324.0000
819.0000	371.0000
0.0000	-416.0000
89.0000	459.0000
-176.0000	0.0000
261.0000	49.0000
-344.0000	-96.0000
425.0000	141.0000
-504.0000	-184.0000
581.0000	225.0000
-656.0000	-264.0000
729.0000	301.0000
0.0000	-336.0000
79.0000	369.0000
-156.0000	0.0000
231.0000	39.0000
-304.0000	-76.0000
375.0000	111.0000
-444.0000	-144.0000
511.0000	175.0000
-576.0000	-204.0000
639.0000	231.0000
0.0000	-256.0000
69.0000	279.0000
-136.0000	0.0000
201.0000	29.0000
-264.0000	-56.0000
325.0000	81.0000
-384.0000	-104.0000
441.0000	
-496.0000	

FDAT3:

99.0000	300.0000
-196.0000	0.0000
291.0000	59.0000
-300.0000	-116.0000
300.0000	171.0000
-300.0000	-224.0000
300.0000	275.0000
-300.0000	-300.0000
300.0000	300.0000
0.0000	-300.0000
89.0000	300.0000
-176.0000	0.0000
261.0000	49.0000
-300.0000	-96.0000
300.0000	141.0000
-300.0000	-184.0000
300.0000	225.0000
-300.0000	-264.0000
300.0000	300.0000
0.0000	-300.0000
79.0000	300.0000
-156.0000	0.0000
231.0000	39.0000
-300.0000	-76.0000
300.0000	111.0000
-300.0000	-144.0000
300.0000	175.0000
-300.0000	-204.0000
300.0000	231.0000
0.0000	-256.0000
69.0000	279.0000
-136.0000	0.0000
201.0000	29.0000
-264.0000	-56.0000
300.0000	81.0000
-300.0000	-104.0000
300.0000	
-300.0000	

3.BERÄKNINGSFILEN.

Beräkningsfilen har följande struktur:

```
Variabeldeklarationer
. . . . . .
Parameterdeklarationer
Parameterblock
. . . . . .
Tilldelningssatser
Villkorssatser
Repetitionssatser
. . . . . .
END
```

3.1.VARIABELDEKLARATIONER.

En variabel kan vara av 4 olika slag:INPUT,OUTPUT,INPUT/VECTOR eller OUTPUT/VECTOR.Om en variabel deklareras INPUT eller OUTPUT innebär det att beräkningsfilen genomlöpes en gång för varje element i den datafil som motsvarar variablen i fråga.Om en variabel dessutom är VECTOR deklarerad innebär det att man varje gång beräkningsfilen genomlöpes har tillgång till alla element i motsvarande datafil.Det är en väsentlig skillnad på innebörden av en vektordeklaration i INPUT-fallet resp. OUTPUT-fallet.Studera följande 2 exempel:

Ex 1. VECTOR IN[40]
INPUT IN
Man har i detta fallet tillgång till hela inputfilen, som kan omfatta fler än 40 **poster** (ca 500 är vanligt). Det är alltså tillåtet att indicera enligt:
I=450
B=IN[I]+....
[40] betyder i det här fallet att 40 värden i taget hämtas från filen och läggs i värdetabellen.När ett index påträffas,som motsvarar en variabel vars värde inte finns i värdetabellen,hämtas 40 nya värden in från den del av filen som innehåller index.(Index är ju inget annat än postnummer i filen) Det totala till-

gängliga tabellutrymmet för variabler,parametrar (ej parameterblock) och literaler är 100 celler, vilket man får ta hänsyn till när man avgör hur stor del av en INPUT VECTOR fil som ska lagras åt gången.
Anm:Det kan tyckas onaturligt att i deklarationen ange hur stor del av filen man vill ha i en arbetsarea i stället för att ange hur stor del man vill arbeta med åt gången. (Vid t.ex. interpollering där man vill använda 3 värden åt gången kunde man tänka sig att skriva deklarationen som VECTOR[3].) Skälen till att deklarationen ser ut som den gör är att det kan förekomma fall då man vill använda värden från en kolumn i datafilen sekventiellt (enkel variabel) och en annan kolumn motsvarar en vektor.Man måste då hålla reda på vilken post i den sekventiella kolumnen som f.n. behandlas. När man ska flytta värden från vektor-kolumnen till tabellen händer följande:Man gör REWIND till första posten,hoppar över poster tills man träffar på rätt index,utför flyttningen,gör REWIND och hoppar över poster tills man åter är framme vid den post man hade före flyttningen.Det är angeläget att allt detta sker så få gånger som möjligt för att öka effektiviteten.Därför anger man i deklarationen ett arbetsutrymme som är så stort som möjligt så att man gör så få flyttningar som möjligt.

Ex 2. VECTOR UT[50]
OUTPUT UT
I detta fallet anger [50] den verkliga storleken av motsvarande datafil.Utrymme i värdetabellen reserveras för hela filen.En OUTPUT VECTOR fil kan inte vara längre än 50 poster.Detta är dock inte så illa som det verkar,ty i de fall denna deklaration används (t.ex. vid klassindelning av data) kan man göra beräkningarna i flera steg och använda t.ex. 5 kolumner om 50 värden i stället för en kolumn om 250 värden.Med hjälp av kommandot MOVE kan man sedan stuva om dessa 5 kolumner så att man får en kolumn om 250 värden.Observera att man måste ange hela filens längd i deklarationen även om man bara vill åt en del av den.

Eventuella VECTOR deklarationer ska alltid placeras först i beräkningsfilen för att ett tillräckligt antal celler ska kunna reserveras i värdetabellen.En variabel får inte vara enbart VECTOR deklarerad.Variabelnamn får högst omfatta 6 tecken(bokstäver eller siffror).

Ex. En input-variabel och en output-variabel:

```
INPUT IN
OUTPUT UT
```

Ex. I en input-fil vill man ha tillgång till samtliga element varje gång man ska beräkna ett värde i output-filen.50 värden åt gången från input-filen ska lagras i värdetabellen.

```
VECTOR IN[50]
INPUT IN
OUTPUT UT
```

Storleken av en vektor ska i en deklaration anges med ett heltal inom klammer,t.ex. [30].Med storlek menas här utrymme i värdetabellen.Varje deklaration kan innehålla flera variabelnamn.

Ex.

```
VECTOR VEK1[50] VEK2[40]
OUTPUT VEK1
INPUT VEK2 IN
```

Här finns två inputvariabler,IN och VEK2.Den senare är vektordeklarerad.Det finns en outputvariabel VEK1.

Antalet inputvariabler är begränsat till högst 3 st.Antalet outputvariabler ska alltid vara en.

3.2.PARAMETERDEKLARATIONER.

En parameterdeklaration har följande utseende:

```
parameternamn : värde
```

Värdet av en parameter,deklarerad på detta sätt,är fixt och kan inte ändras i någon tilldelningssats.Parametrar får inte utgöras av indicerade variabler.En sådan kan tilldelas värde i en vanlig tilldelningssats,t.ex. P[I]=3.1415.

Ex. PAR:2.5 ABC:1E-5 D:-77 C23:-9.76

3.3.PARAMETERBLOCK.

Deklarationen av ett parameterblock har följande utseende:

```
PARAMETERS grupp1     grupp2 .....
     p1:   värde(1,1)  värde(1,2) .....
     p2:   värde(2,1)  värde(2,2) .....
   ......
ENDPAR
```

I stället för PARAMETERS kan man använde det kortare PAR. Det set av parametrar,p1 p2 ...,som finns i deklarationen kan anta värden från någon av grupperna grupp1 grupp2 ... . Vilken grupp av värden som ska tilldelas parametrarna bestäms med kommandot GET PAR (se kap 5.2).

Ex. 3 parametrar A,B och C kan i två olika fall anta värdena 10.5,11.5,12.5 resp. 20.5,21.5,22.5 :

```
PAR FALL1 FALL2
 A: 10.5  20.5
 B: 11.5  21.5
 C: 12.5  22.5
ENDPAR
```

Innan man startar beräkningarna för det första fallet skriver man kommandot GET PAR FALL1.
Högst ett parameterblock får förekomma i beräkningsfilen.

Antalet grupper är maximerat till 10 st.,liksom antalet para-

3.4. TILLDELNINGSSATSER

En tilldelningssats har strukturen

variabel=uttryck

Variablen kan vara enkel eller indicerad.I det senare fallet skall det dock vara en OUTPUT-deklarerad variabel.Uttrycket kan vara ett enkelt uttryck eller ett villkorsuttryck,av samma slag som i SIMNON.Man har tillgång till följande funktioner:

SQRT(X)	EXP(X)	LN(X)
LOG(X)	SIN(X)	COS(X)
TAN(X)	ATAN(X)	ATAN2(X,Y)
ABS(X)	SIGN(X)	INT(X)
MOD(X,Y)	MIN(X,Y)	MAX(X,Y)

För en närmare beskrivning,i de fall där inte innebörden är självklar,hänvisas till SIMNON-manualen.
De operatorer man kan använda är + - / * ↑ >< OR AND NOT.

Ex. VAR=SIN(A)+IN-INT(C1,C2)+23

Om man använder indicerade variabler ska index utgöras av en enkel variabel omgiven av klammer.

Ex. INDEX:3
X1=4
OUT[X1]=IN[INDEX]+MAX(A,B)*25.05

Alla variabler och literaler representeras som reela storheter. Index omvandlas programmässigt till heltal enl.(jfr ex ovan): OUT[X1] → IX=X1+0.00005 (Fortran-avkortning)→ OUT[IX] beräknas.

Ex. VEK[J]=IF A>B AND C<D THEN MAX(A,B)/C ELSE D*IN[I]

En noggran genomgång av tilldelningssatser finns i SIMNON-manualen.

3.5. REPETITIONSSATSER.

En repetitionssats har följande struktur:

```
WHILE villkor DO
sats1
sats2
.....
ENDWHILE
```

Satserna sats1,sats2 etc kan vara av två slag,tilldelningssatser eller villkorssatser(se kap 3.6).

Ex.
```
WHILE A>B AND IN[I]<17 DO
    IN[I]=A/B
    A=IF IN[I]>10 THEN A ELSE A+10
    ENDWHILE
```

3.6. VILLKORSSATSER.

En enkel villkorssats har följande struktur:

I:
```
IF villkor THEN
sats1
sats2
.....
ELSE
satsa
satsb
.....
ENDIF
```

eller

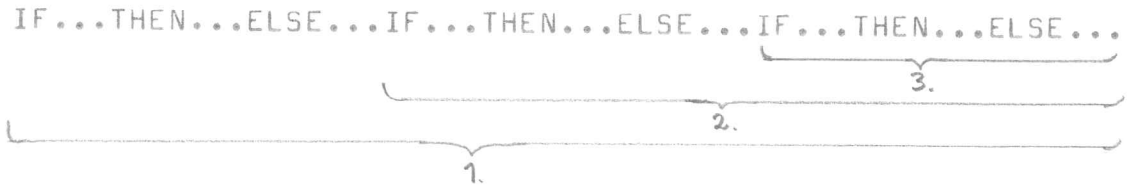
II:
```
IF villkor THEN
sats1
sats2
.....
ENDIF
```

Satserna 1,2,a,b etc kan vara tilldelningssatser,repetitionssatser eller villkorssatser.

Ex.
```
IF A>B THEN
    I=1
    WHILE I<11 DO
    IN[I]=A
    I=I+1
    ENDWHILE
```

```
ELSE
    X=11
    IN[X]=B
ENDIF
```

Flera villkorssatser kan kombineras enligt följande modell:



Maximalt kan 5 st satser nystas på detta sätt.Varje villkors-sats avslutas med ENDIF,vilket i fallet ovan skulle betyda att satsen efter sista ELSE ska följas av 3 st ENDIF.Det är inte tillåtet att nysta samman de båda strukturerna I och II. Om man vill skriva en sådan sats,så kan den alltid omformas till två enkla villkorssatser.

4. KOMPILERING AV BERÄKNINGSFILEN.

För att ta fram en kompilator för beräkningsfilen har jag utgått ifrån SIMNON-kompilatorn,som är ett paket omfattande ca 50 subrutiner.Ifrån denna kompilator har jag strukit de delar som behandlar olika typer av system i SIMNON .Jag har vidare infört nya subrutiner som sköter kompileringen av loopar,villkorssatser och parameterblock.Som en följd av detta har jag också ändrat resp. tillfogat nya feltester och felutskrifter. I fig. 1 visas strukturen hos kompilatorn.För åskådlighetens skull har inte stränghanteringsrutiner,felutskriftsrutiner samt en del smårutiner som ofta anropas tagits med.I det följande beskrives kort vad som händer i de olika rutinerna:

COMP
räddar undan tabellpekare(se nedan) och anropar sedan i tur och ordning underliggande rutiner som identifierar och avläser den aktuella raden i bufferten.Om raden visar sig vara riktig uppdateras tabellpekarna efter kompileringen.

RESW
Avläser första ordet i raden och känner igen vissa reserverade ord (INPUT,OUTPUT,VECTOR,PARAMETERS,PAR,ENDPAR,WHILE, ENDWHILE,IF,ELSE,ENDIF,END).Resultatet blir en kodsiffra som är =0 om inget reserverat ord förekom,2,3...eller 13 om ett reserverat ord inledde raden.

I fortsättningen förekommer följande tabeller (COMMON-block):

VARTB1 innehåller variabelnamn

VARTB2 innehåller pekare som pekar på resp. variabels värde i värdetabellen.

VARTB3 innehåller variablens typ (se nedan)

VALUES värdetabell,innehåller variablernas och literalernas värden

CMPVAR innehåller felflagga och pekare till aktuellt element i VARTB1-3.

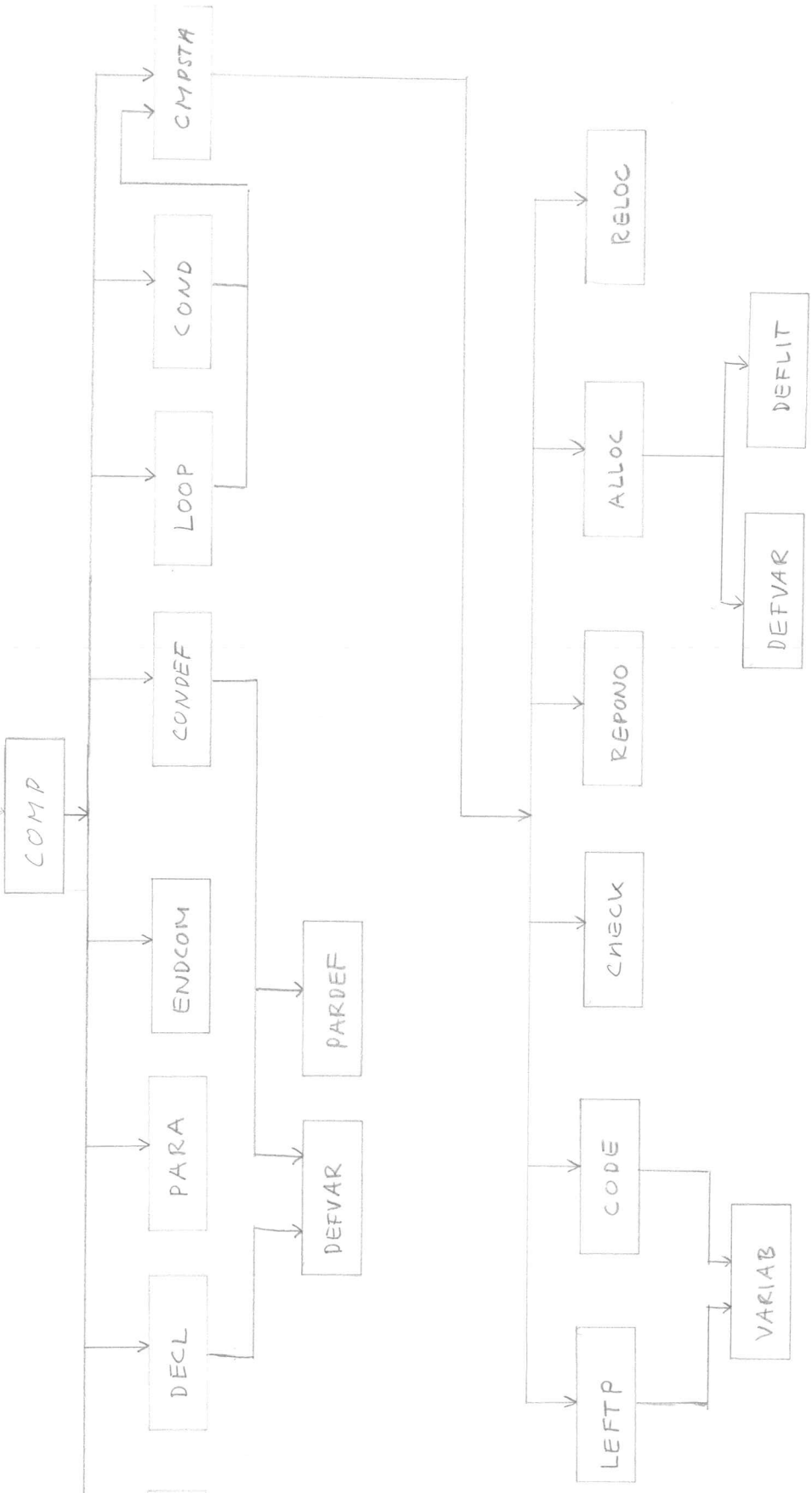


FIG 1. Strukturen hos kompilatorn.Rutiner för stränghantering,felutskrifter och en

VARTB1,VARTB2 och VARTB3 ligger parallellt,dvs. mot variabeln i t.ex. VARTB1(3) svarar pekaren i VARTB2(3) och typen i VARTB3(3).

DECL
avläser variabelnamnet i en deklaration.Typen av variabel bestäms.Om variabeln har varit tidigare deklarerad (dvs vektor-deklarerad) ändras den gamla typen.I annat fall anropas

DEFVAR
som lägger in variablerna i en deklaration i VARTB1 och typerna i VARTB3.Plats reserveras i VALUES och motsvarande värden nollställes.Om det är en vektor som har deklarerats,reserveras det antal celler i VALUES som anges av talet inom klammarna.Pekaren i VARTB2 pekar på det första elementet i VALUES som hör till resp. variabel.

PARA
läser av första raden i en deklaration av ett parameterblock. Gruppnamnen läggs i en COMMON-area PARTAB.I denna area finns också plats för de parameternamn och -värden som hör till blocket.För att indikera att de parameterdeklarationer som följer tillhör ett parameterblock sättes en indikator,MODE=4.

ENDCOM
läser av sista raden,lägger in koden för END i arean för pseudokod,COMMON /PSCODE/,samt kontrollerar att antalet IF/ENDIF resp. WHILE/ENDWHILE är lika.

CONDEF
läser av en parameterdeklaration.Om det är en enkel parameterdeklaration,t.ex. P1:2.5,läggs variabelnamn och -typ in i VARTB1 och VARTB3 via subrutinen DEFVAR.Värdet läggs in i VALUES via rutinen DEPOS.Om deklarationen emellertid skulle tillhöra ett parameterblock,dvs. om MODE=4,anropas

PARDEF
som lägger in parameternamn och -värden i arean PARTAB.

LOOP
organiserar kompileringen av en repetitionssats.I fallet 'WHILE' anropas CMPSTA med en switch satt,som talar om att det som följer på WHILE är ett uttryck,dvs. en sats utan vänsterled ska behandlas av CMPSTA.Vidare registreras det aktuella läget,LW,i pseudokoden och en kontrollräknare ökas ett steg.I fallet 'ENDWHILE' läggs koden för ENDWHILE in i PSCODE och därefter stoppas LW in.LW anger alltså adressen i pseudokoden till WHILE-satsens början.Kontrollräknaren minskas därefter ett steg.

COND
organiserar kompileringen av en villkorssats.I fallet 'IF' anropas CMPSTA med en switch satt,på samma sätt som i LOOP. I fallet 'ELSE' läggs koden för ELSE in i PSCODE.Vidare sker en registrering av en del adresser i pseudokoden som behövs för att klara nystade villkorssatser.I fallet 'ENDIF' sker viss registrering av samma skäl som ovan.Kontrollräkning av antalet IF/ENDIF sker analogt med räkningen i LOOP.

CMPSTA
organiserar kompileringen av en tilldelningssats genom att i tur och ordning anropa de underliggande rutiner som beskrivs nedan(se även fig 1).Under själva kompileringen av en sats läggs alla variabler temporärt i en area VARIBL.Först sedan det konstaterats att satsen är riktig flyttas variablerna till VARTB1.

LEFTP
känner igen vänsterledet i en sats.En för de följande rutinerna viktig variabel är NLVAR,som har följande innebörd:

NLVAR=0,inget vänsterled (repetitions- eller villkorssats)
NLVAR=1,enkel variabel i vänsterledet
NLVAR=2,indicerad variabel i vänsterledet
LEFTP anropar sedan

VARIAB
som stoppar in variabelnamnet i VARIBL.Om variabeln är enkel anger IPNT(1) variabelns adress i VARIBL och IPNT(2)=0. Om den däremot är indicerad pekar IPNT(1) på variabeln själv och IPNT(2) på dess index.

CODE
kodar högerledet av en sats varvid resultatet blir en vektor ICODE innehållande kodsiffror som representerar operatorer, funktioner etc.Variabler fylls på i VARIBL via rutinen VARIBS efterhand som de dyker upp i uttrycket.En variabel anges i ICODE med kodsiffran för variabel följd av en eller två pekare till VARIBL beroende på om variabeln är enkel eller indicerad.På motsvarande sätt som variabler behandlas numeriska värden i uttrycket (literaltabell,literalpekare).

CHECK
går igenom ICODE och letar efter formella fel i uttrycket.

ALLOC
anropas sedan,när uttrycket kontrollerats av CHECK.ALLOC ger som resultat 3 vektorer med pekare,ILPNTS för vänstervariabler,IRPNTS för högervariabler och LITPNT för literaler.Om en variabel i VARIBL ej finns i VARTB1 sätts den dit på en ledig plats.Läget registreras i IRPNTS eller ILPNTS.Om variablen tidigare fanns i VARTB1,dvs. den har förekommit i en sats tidigare i filen,sätts pekaren i IRPNTS eller ILPNTS.Literaler behandlas analogt med variabler,frånsett att de lagras bakifrån i VALUES.I ALLOC sker vidare några kontroller av variablernas typ:
-Vänsterledet får inte utgöras av en input-vektor,ty man riskerar då att förlora ett beräknat värde om den aktuella delen av filen skulle kastas ut ur värdetabellen och ersättas av en annan del.

-Alla **output**-variabler och parametrar ,utom de som tillhör ett parameterblock,i högerledet måste ha tilldelats värde tidigare.

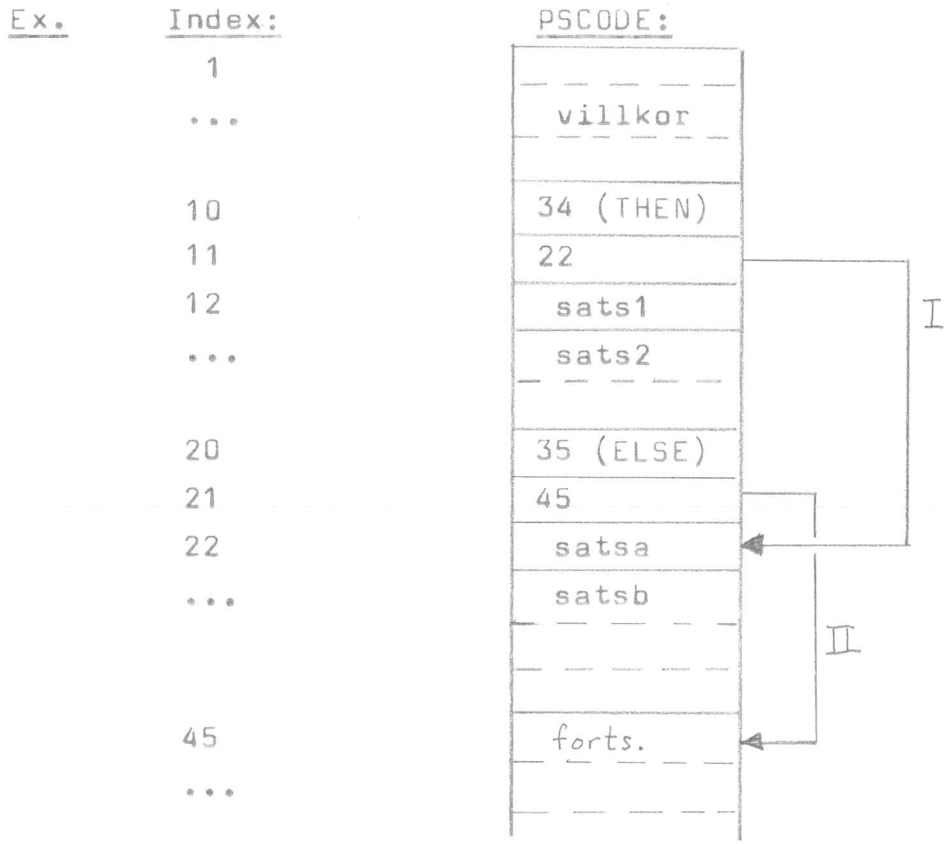
REPONO
omvandlar ICODE,som ju var en direkt översättning av ett uttryck till kodsiffror,till omvänd polsk notation,som läggs i vektorn IRPN.
Ex. VL=A+B/C+10 blir A B C / + 10 + = VL

RELOC
är slutstationen i kompileringen.Här bildas pseudokoden,PSCODE.I princip är det frågan om en översättning av IRPN.Alla pekare i PSCODE är negativa (undantag:i samband med DO eller THEN,se nedan).Variabelpekare anger adresser i VARTB1-3 medan literalpekare anger adresser i VALUES.Om en repetitions- eller villkorssats kompileras,vilket anges av den switch som sattes i samband med anropet av CMPSTA,bestämmes en del adresser i PSCODE som behövs för att kunna göra hopp i pseudokoden under exekveringen.Betydelsen av de olika kodsiffrorna i PSCODE anges i tabell 1 nedan.Det är inte alla siffror i intervallet 1-32 som används till kod.

1 OR	15 variabel i VL,enkel variabel följs av pekare,indicerad variabel följs av '25' och 2 pekare
2 AND	
3 NOT	
4 <	16 funktion,följs av kod som specificerar slaget av funktion.
5 >	
6 +	17 hoppinstr.
7 -	18 avslutar pseudokoden
8 *	25 indicerad variabel i HL,följs av 2 pekare
9 /	
10 unitärt -	31 DO,följs av adress för ev. hopp
11 ↑	32 ENDWHILE,följs av adress för ev. hopp
12 hoppinstr.	
13 hoppinstr.	34 THEN,följs av adress för ev. hopp
14 enkel variabel i HL, följs av pekare	35 ELSE,följs av adress för ev. hopp

TABELL 1: Innebörden av kodsiffrorna i pseudokoden PSCODE.

I följande exempel visas hur hoppen åstadkommes i pseudokoden i en villkorssats, IF villkor THEN sats1, sats2..ELSE satsa satsb...ENDIF.



Om villkoret är sant skippas 11:e elementet i koden, annars görs hoppet I. Efter det att man nått fram till ELSE görs hoppet II. Satsen efter ENDIF börjar med 45:e elementet i PSCODE.

För att sköta felutskrifter finns rutinerna ERROR och WERROR. Den förra hissar en felflagga, skriver ut den felaktiga raden varpå WERROR skriver ett felmeddelande. Ett kompileringsfel orsakar alltid att exekveringen av programmet stoppas.

5. KOMMANDON.

Programmet skriver ut ENTER COMMAND: på skärmen och väntar sig då ett av följande kommandon:GENKOD,GENOP,MOVE eller END.

5.1. GENKOD.

För att kompilera beräkningsfilen används kommandot

```
GENKOD psfil←berfi
```

Filnamn med små bokstäver väljes fritt av användaren,max 5 tecken.← får inte omges av blanktecken.Resultatet av kommandot blir att beräkningsfilen,som här antages heta BERFI,kompileras.Därpå skrives pseudokod och tabeller ut i filen PSFIL.Nedan visas ett exempel på en beräkningsfil och motsvarande pseudofil.(Båda dessa filer kan listas av användaren genom lämpliga PIP-kommandon)

Ex. BERFI:

```
OUTPUT UT
INPUT IN
PAR GR1 GR2 GR3
A0:  3   3   3
A1:  3   3   3
A2:  2  -1  -1
A3: -1   0   1
ENDPAR
UT=A0+IN*(A1+IN*(A2+IN*A3))
END
```

(Forts. på exemplet.)

PSFIL:

```
    0      6      6  100     23
UT        IN        A0        A1        A2        A3
   1   2   3   4   5   6
   4   3  -7  -7  -7  -7
  14  -3  14  -2  14  -4  14  -2  14  -5  14  -2  14  -6   8   6   8   6   8   6
  15  -1  18
     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
     0.0000     0.0000     0.0000     0.0000
     3      4
GR1       GR2       GR3
A0        A1        A2        A3
     3.0000     3.0000     2.0000    -1.0000     3.0000     3.0000    -1.0000     0.0000
     3.0000     3.0000    -1.0000     1.0000
```

Rad 1 innehåller pekare till tabellerna.

Rad 2 innehåller VARTB1,dvs variabelnamnen och parameternamnen.

Rad 3 innehåller VARTB2,dvs pekare till VALUES.

Rad 4 innehåller VARTB3,dvs typerna för resp. variabler.

Rad 5-6 innehåller PSCODE.

Rad 7-19 innehåller VALUES,såsom den ser ut direkt efter kompileringen.

Rad 20-24 innehåller PARTAB och finns bara med om beräkningsfilen innehåller ett parameterblock.

5.2. GENOP OCH DESS SUBKOMMANDON.

För att förbereda exekveringen används kommandot

GENOP output←psfil input1 input2 input3

Ett filnamn kan skrivas enligt något av följande exempel:
DATFI - syftar på första kolonnen i filen DATFI.
FIL(3) - syftar på tredje kolonnen i filen FIL.
FIL(4 1 2) - syftar på fjärde,första och andra kolonnen i filen FIL.
Den kolumn som anges för output-filen ska vara mindre än eller lika med det befintliga antalet kolumner +1 i output-filen. En output-fil måste alltid finnas med liksom minst en pseudofil.Antalet inputfiler,eller rättare sagt:det sammanlagda antalet aktuella kolonner hos input-filerna,får vara 0-3 st. Om output-filen inte tidigare finns på skivan anges detta med en * omedelbart efter GENOP.En ny fil kommer då att skapas.

Ex. GENOP FILUT←PS1 FILIN
GENOP FDAT1←KOMP FDAT3(2) FDAT2(4) FDAT3(3)
GENOP* NEWFI←PSFIL FDAT3(4 1) FDAT1
GENOP HEJ←SVEJS HEJ

Det är viktigt att input-filerna hamnar i rätt ordningsföljd i kommandot för att de ska knytas till rätt variabel i beräkningsfilen.Filerna ska stå i samma ordning som motsvarande variabler dyker upp i beräkningsfilens deklarationer.Om man är osäker kan man lista pseudofilen,där man direkt i tabellen VARTBL ser i vilken ordning variablerna kommer.

Ex. VECTOR UT[25] IN2[25]
INPUT IN1 IN2
OUTPUT UT
Här knyts filer och variabler ihop enligt:
FDAT3-UT,FDAT1-IN2 och FDAT2-IN1.

GENOP FDAT3←PSFIL FDAT1 FDAT2

Ex. VECTOR IN3[10] GENOP FUT←PSFIL DAT(1 2) F(4)
OUTPUT UT
INPUT IN1 IN2 IN3
ger: FUT-UT,DAT(1)-IN3,DAT(2)-IN1 och F(4)-IN2.

Antalet beräkningar som utföres bestäms av det minsta antalet poster i de input- och outputfiler som motsvarar enkla variabler.En input-vektor fil får inte vara kortare än den kortaste bland de input-filer som motsvarar enkla variabler. Skälet härför är att även en vektor-fil läses element för element,ty det kan ju vara så att en kolumn i filen motsvarar en enkel variabel medan en annan kolumn motsvarar en indicerad variabel.En kombination som inte kan förekomma är en ny output-fil motsvarande en enkel variabel och en input-fil som motsvarar en indicerad variabel.Ett sådant arrangemang saknar mening eftersom man inte vet hur många element outputfilen då ska innehålla.
Efter kommandot GENOP skrivs en > ut på skärmen.Man kan då skriva något av följande subkommandon:

a) >X
b) >DISPLAY PAR
c) >CHANGE PAR namn:värde
d) >SAVE PAR filnamn
e) >GET PAR grupp
f) >DELETE PAR
g) > (vagnretur)

a) >X,startar exekveringen av pseudokoden.
b) >DISPLAY PAR,visar aktuella parametrar och deras värden på skärmen.
c) >CHANGE PAR namn:värde,ändrar värdet av en eller flera parametrar som inte tillhör ett parameterblock.
Ex.>CHANGE PAR PI:3.15 >CHANGE PAR P1:10.6 P2:10.7 P3:13
d) >SAVE PAR filnamn,lägger ut aktuella parametrar och deras värden i den fil som anges.
Ex. >SAVE PAR PFIL

e) >GET PAR grupp,tilldelar värden till parametrar tillhörande ett parameterblock enligt den grupp som anges.

Ex. >GET PAR GRUPP1

f) >DELETE PAR,upphäver tilldelningen enligt e) och måste göras innan man ånyo kan använda kommandot GET PAR.

g) >följt av vagnretur ger utskrift av ENTER COMMAND: på skärmen varpå man åter kan använda något av kommandona GENKOD, GENOP, MOVE eller END.

Följande exempel visar hur man kan hantera de olika subkommandona:

Ex. Beräkningsfilen:

```
INPUT IN
OUTPUT UT
PAR NR1 NR2 NR3 NR4
P1: 1.5 2.5 3.5 1.1E2
P2:-9.1 9.2 9.3 9.4
ENDPAR
P3:56.56
P4:57.57
PI:3.14159
UT= IN + P1+ P2+P3+PI +P4
END
```

Dialogen på skärmen:

```
ENTER COMMAND:

GENKOD PSFP←PFILE
WELL DONE - COMPILATION WAS SUCCESFUL

ENTER COMMAND:

GENOP FDAT1←PSFP FDAT1(2)
>DISPLAY PAR
P3          =     56.5600
P4          =     57.5700
PI          =      3.1415

>GET PAR NR1
>DISPLAY PAR
P1          =      1.5000
P2          =     -9.1000
P3          =     56.5600
P4          =     57.5700
PI          =      3.1416
>CHANGE PAR P1:99 P2:99\\:99
   VARIABLES OR PARAMETERS IN A PARAMETER BLOCK
   CAN NOT BE CHANGED
```

```
>DISPLAY PAR
P1          =      1.5000
P2          =     -9.1000
P3          =     56.5600
P4          =     57.5700
PI          =      3.1500
>CHANGE PAR P3:999 P4:777
>DISPLAY PAR
P1          =      1.5000
P2          =     -9.1000
P3          =    999.0000
P4          =    777.0000
PI          =      3.1500

>DELETE PAR
>DISPLAY PAR
P3          =    999.0000
P4          =    777.0000
PI          =      3.1500
>GET PAR NR3
>DISPLAY PAR
P1          =      3.5000
P2          =      9.3000
P3          =    999.0000
P4          =    777.0000
PI          =      3.1500
>SAVE PAR FILIL
```

Listning av FILIL:

```
DOS-15 UV3A002
$PIP

DOSPIP V3A002

>T TT←RK FILIL SRC


P1          =      3.5000
P2          =      9.3000
P3          =    999.0000
P4          =    777.0000
PI          =      3.1500

DOSPIP V3A002
```

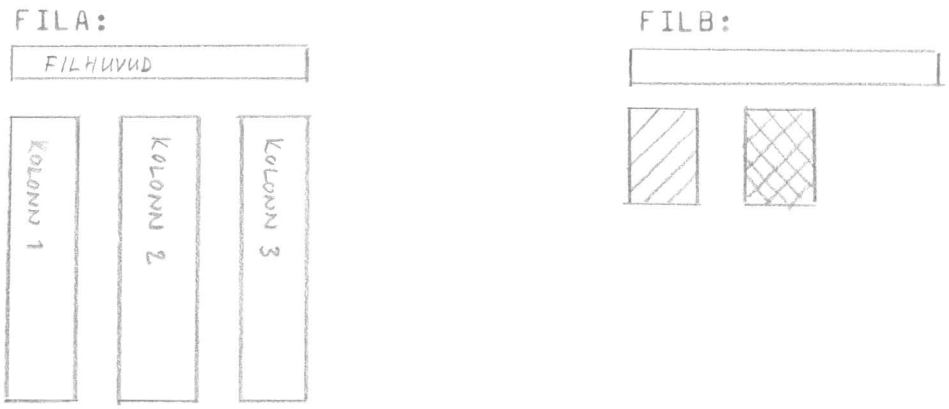
5.3. MOVE.

Med kommandot

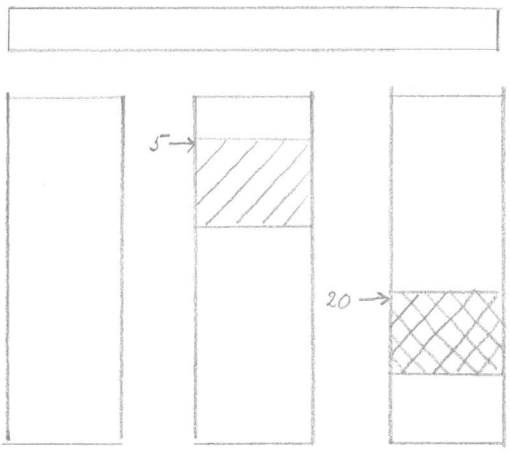
MOVE recnr filut←filin

kan man flytta en kolonn från en kort fil (filin) till valfri plats som anges av recnr i en lång fil (filut).

Ex. Filerna FILA och FILB har följande utseende:



Efter kommandona MOVE 5 FILA(2)←FILB och
MOVE 20 FILA(3)←FILB(2) ser FILA ut så här:



Gamla värden i de delar av FILA till vilka flyttning sker går förlorade.Man kan även flytta data till kolonnen närmast till höger om de redan befintliga kolonnerna.(I ex. ovan FILA(4).)Kommandot MOVE används lämpligen i de fall då det inte räcker att ha 50 element i en output-vektor fil.I efterhand kan man då bunta ihop flera kolonner till en enda lång kolonn.Jfr. ex. på sid. 36 .

5.4 END.

Kommandot END stoppar programmet utan vidare åthävor.

6. BESKRIVNING AV DE RUTINER SOM TOLKAR OCH EXEKVERAR KOMMANDON.

För att tolka och utföra de beräkningar som anges av kommandona har ett antal subrutiner skrivits.Strukturen hos dessa visas i fig. 2.En kortfattad beskrivning följer nedan:

MAIN
sköter kommunikationen med användaren,avläser första ordet i kommandot och anropar med ledning av detta någon underliggande subrutin.

LFINAM
avläser filnamnen i ett kommando och deras kolonner samt lagrar detta i COMMON-arean FILES.

COMPIL
sköter kompileringen av beräkningsfilen genom att för varje rad anropa COMP (se fig 1).

GENKOD
läser först filnamnen i anropet via LFINAM och kompilerar sedan beräkningsfilen via COMPIL.Därefter bildas den s.k. pseudofilen genom att pseudokod,variabel-,värde- och parametertabeller skrivs ut i den output-fil som angetts i kommandot.Om man vill använda samma beräkningsfil vid flera olika tillfällen räcker det alltså att kompilera den första gången och sedan spara pseudofilen.

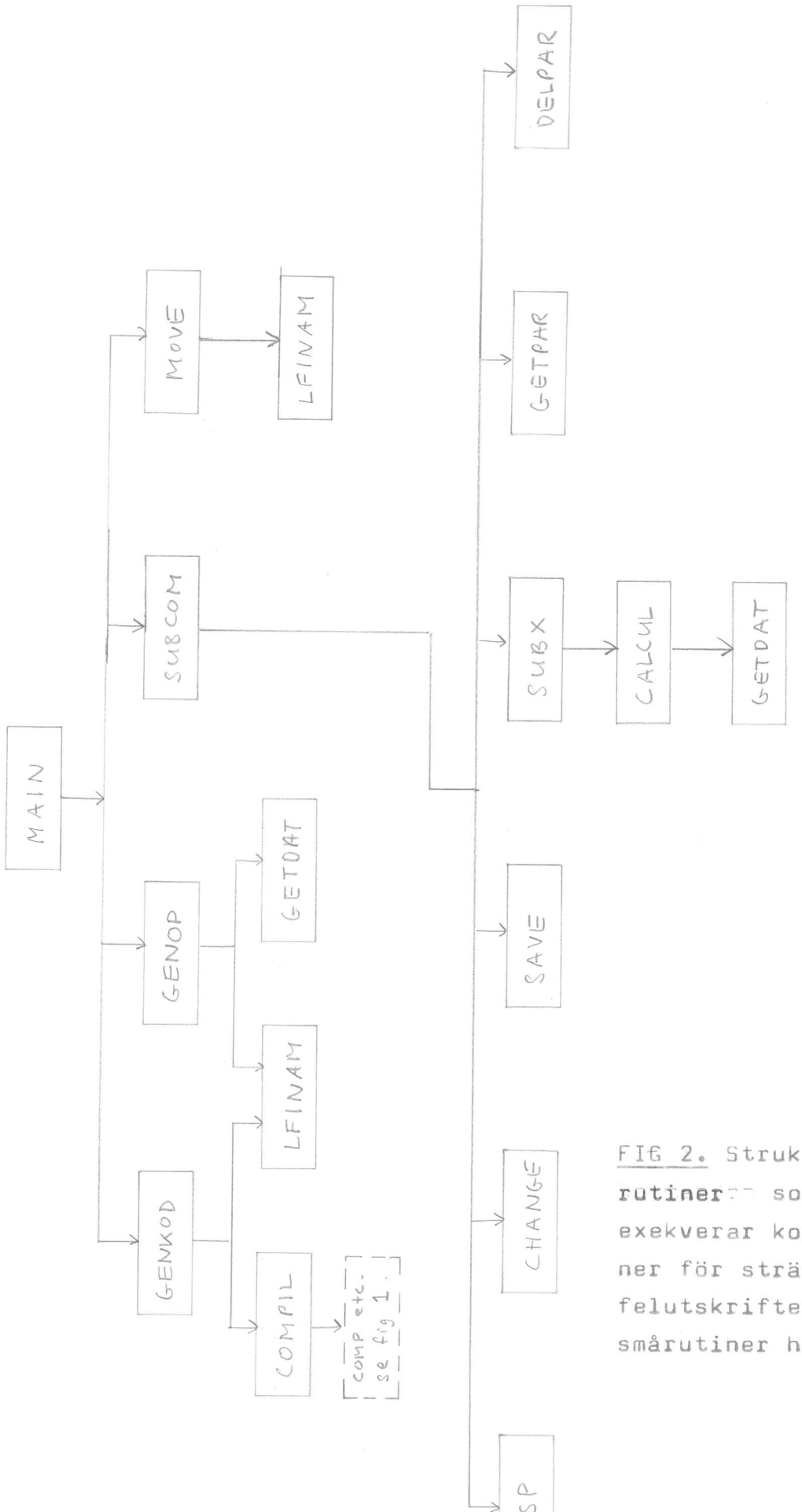


FIG 2. Strukturen hos de rutiner som tolkar och exekverar kommandon. Rutiner för stränghantering, felutskrifter och en del smårutiner har utelämnats.

GENOP

läser först av pseudofilen och flyttar informationen till COMMON-areor.Vidare bildas en vektor INDEX(8),som har stor betydelse för en del av de följande rutinerna,med följande innehåll:

- INDEX(1) - =1 om output-variabeln är enkel,=2 om den är indicerad.
- INDEX(2) - pekar på outputvariabeln i VATB1-3 (samma innehåll som i VARTB1-3 .)
- INDEX(3) - =1 om första input-variabeln är enkel,=2 om den är indicerad.
- INDEX(4) - pekar på första input-variabeln i VATB1-3.
- INDEX(5), INDEX(6) } andra input-variabeln
- INDEX(7), INDEX(8) } tredje input-variabeln

Till sist registreras det vilka variabler som är input-vektorer.För var och en av dessa görs följande:

1) Första delen av datafilen flyttas till värdetabellen m.h.a. GETDAT.
2) I arean DSTORE registreras en pekare till datafilen som anger det första elementet som f.n. ligger i värdetabellen,hur många värden som flyttats till värdetabellen samt läget av det första av dessa värden i tabellen.

GETDAT

flyttar en del av en datafil till värdetabellen.Rutinen användes på input-vektorfiler.

SUBCOM

avläser ett subkommando (> ...) och anropar med ledning av detta någon underliggande subrutin.

DISP

skriver ut aktuella parametrar och deras värden på skärmen.

CHANGE
tolkar kommandot CHANGE PAR och utför ändringen av parameter-värdet(-ena).Lägg märke till att om man kompilerar beräknigs-filen en andra gång gäller de parametervärden som har angetts i filen,således inte de som under en tidigare körning har ändrats med CHANGE PAR.

SAVE
skriver ut aktuella parametrar och deras värden i den fil som anges i kommandot.

GETPAR
tilldelar parametrar tillhörande ett parameterblock värden enligt den grupp som anges i kommandot.

DELPAR
upphäver tilldelningen enligt senaste GET PAR-kommando.

SUBX
organiserar de beräkningar som ska göras med datafilerna. Värden från datafiler flyttas till värdetabellen,beräkningar utföres och resultatet skrivs tillbaka i datafilerna.Själva räkningarna görs av CALCUL (se nedan).Den exakta gången i programmet och de ställen där data ska hämtas och läggas bestämmes väsentligen av innehållet i vektorn INDEX,som bildades i GENOP.

CALCUL
utför de beräkningar som anges av pseudokoden.Om en indicerad input-variabel påträffas,kontrolleras att dess index hör till den del av filen som f.n. finns i värdetabellen.Om så behövs hämtas rätt del av datafilen m.h.a. GETDAT.

MOVE
flyttar en kolonn i en kort datafil till valfri plats i en kolonn tillhörande en lång datafil.

7. LÄNKNING,UPPSTART OCH SYSTEMBEROENDE RUTINER.

Länknigen görs enklast genom att läsa in en BATCH-remsa med följande innehåll:

```
$JOB
A RK <NEW> -1
CHAIN
MAINGO
SZ
MAIN,IFAC,RIFF,RBUFF,RFP,SIGN1,DIGITS,RABC,GAC,PAC,LENGTH,
LCOMPV,LCOND,WERROR,HSTORV,RCHAR,LFINAM,
LFIND
LKOD=GENKOD,COMPIL,
ICOMNT,PARA,PARDEF,COND,LITT,ALLOC,OPER,CONDEF,CODE,
ENDCOM,RELOC,LOOP,CMPSTA,DECL,RESW,LIFO,COMP,SEARCH,
DEFVAR,LEFTP,VARIAB,CHECK,REPONO,BOOLW,FUNC,INSTR,DEPOS,
ERROR
LOP=GENOP,MOVE,GETDAT,SUBCOM,DISP,CHANGE,SAVE,GETPAR,
DELPAR,SUBKIL,CALCUL,SUBX
LKOD:LOP

$JOB
ABS
MAINGO
$EXIT
```

RESIDENT CODE

LINK LKOD

LINK LOP

Uppstart av programmet sker sedan med följande monitorkommandon:

```
$ A RKA 3
$ BUFFS 5
$ E MAINGO
```

I en del rutiner har PDP-15 delordsoperationer använts,varför dessa alltså blir systemberoende.Dessa rutiner är stränghanteringsrutinerna GAC,PAC och LCOMPV.I de flesta av rutinerna i fig 2 har filhanteringrutiner för PDP-15 använts (SEEK, ENTER,CLOSE etc.)och även delordsoperationer förekommer i samband med bildandet av filnamnen.Dessa momenten i rutinerna är således systemberoende.

8. NÅGRA EXEMPEL.

I detta kapitel visas beräkningsfiler,kommandon och datafiler för några valda exempel.

8.1. Generering av en funktion.

Frekvensfunktionen för normalfördelningen ska genereras.

Beräkningsfil:

```
INPUT IN
OUTPUT UT
PI:3.14159
UT=EXP(-IN*IN/2)/SQRT(2*PI)
END
```

Kommandon:

```
GENKOD PSFIL←BERFI
GENOP FDAT3←PSFIL FDAT3(3)
>X
```

FDAT3(3):

0.0500	
0.1000	
0.1500	1.5500
0.2000	1.6000
0.2500	1.6500
0.3000	1.7000
0.3500	1.7500
0.4000	1.8000
0.4500	1.8500
0.5000	1.9000
0.5500	1.9500
0.6000	2.0000
0.6500	2.0500
0.7000	2.1000
0.7500	2.1500
0.8000	2.2000
0.8500	2.2500
0.9000	2.3000
0.9500	2.3500
1.0000	2.4000
1.0500	2.4500
1.1000	2.5000
1.1500	2.5500
1.2000	2.6000
1.2500	2.6500
1.3000	2.7000
1.3500	2.7500
1.4000	2.8000
1.4500	2.8500
1.5000	2.9000

FDAT3(1):

0.3984	
0.3970	
0.3945	0.1200
0.3910	0.1109
0.3867	0.1023
0.3814	0.0940
0.3752	0.0863
0.3683	0.0790
0.3605	0.0721
0.3521	0.0656
0.3429	0.0596
0.3332	0.0540
0.3230	0.0488
0.3123	0.0440
0.3011	0.0395
0.2897	0.0355
0.2780	0.0317
0.2661	0.0283
0.2541	0.0252
0.2420	0.0224
0.2299	0.0198
0.2179	0.0175
0.2059	0.0154
0.1942	0.0136
0.1826	0.0119
0.1714	0.0104
0.1604	0.0091
0.1497	0.0079
0.1394	0.0069
0.1295	0.0060

8.2.Polynomberäkning.

UT=P(IN),där P är ett 3:e grads polynom vars parametrar kan varieras i ett parameterblock.

Beräkningsfil:

```
      OUTPUT UT
      INPUT IN
      PAR GR1 GR2 GR3
      A0:  3   3   3
      A1:  3   3   3
      A2:  2  -1  -1
      A3: -1   0   1
      ENDPAR
      UT=A0+IN*(A1+IN*(A2+IN*A3))
      END
```

Kommandon:

```
      ENTER COMMAND:

      GENKOD PSF2←HEJ2
      WELL DONE - COMPILATION WAS SUCCESFUL

      ENTER COMMAND:

      GENOP FDAT3(3)←PSF2 FDAT3
      >GET PAR GR2
      >DISPLAY PAR
      A0          =      3.0000
      A1          =      3.0000
      A2          =     -1.0000
      A3          =      0.0000
      >X
      >
      ENTER COMMAND:

      GENOP FDAT3(2)←PSF2 FDAT3
      >GET PAR GR3
      >DISPLAY PAR
      A0          =      3.0000
      A1          =      3.0000
      A2          =     -1.0000
      A3          =      1.0000
      >X
      >

      ENTER COMMAND:

      END
```

FDAT3:

```
 1.0000     6.0000     5.0000
 2.0000    13.0000     5.0000
 3.0000    30.0000     3.0000
 4.0000    63.0000    -1.0000
 5.0000   118.0000    -7.0000
 6.0000   201.0000   -15.0000
 7.0000   318.0000   -25.0000
 8.0000   475.0000   -37.0000
 9.0000   678.0000   -51.0000
10.0000   933.0000   -67.0000
11.0000  1246.0000   -85.0000
12.0000  1623.0000  -105.0000
13.0000  2070.0000  -127.0000
14.0000  2593.0000  -151.0000
15.0000  3198.0000  -177.0000
16.0000  3891.0000  -205.0000
  ...       ...        ...
```

8.3. RÄTTNING AV DATA MED LINJÄR INTERPOLLERING.

Felaktiga värden (här nollor) rättas med linjär interpollering i datafilen,värdena flyttas till en kort temporär fil och läggs slutligen tillbaks i datafilen.

Beräkningsfilen:

```
VECTOR IN[25] UT[10]
OUTPUT UT
INPUT IN
GALET:0
ISTART:5
NDATA:10
I=ISTART
J=0
WHILE I<NDATA+ISTART DO
J=J+1
UT[J]=IN[I]
IF NOT IN[I]>0 AND NOT IN[I]<0 THEN
N=I-1
M=I+1
UT[J]=(IN[N]+IN[M])/2
ELSE
UT[J]=IN[I]
ENDIF
I=I+1
ENDWHILE
X=ISTART+NDATA-1
UT[X]=IN[X]
END
```

Kommandon:

```
GENKOD PSF4←HEJ4
WELL DONE - COMPILATION WAS SUCCESFUL

ENTER COMMAND:

GENOP FDAT1←PSF4 FDAT3(4)
>DISPLAY PAR
GALET     =      0.0000
ISTART    =      5.0000
NDATA     =     10.0000
>X
>

ENTER COMMAND:

GENOP FDAT1(2)←PSF4 FDAT3(4)
>

>X
>

ENTER COMMAND:

GENOP FDAT1(2)←PSF4 FDAT3(4)
>CHANGE PAR ISTART:15
>DISPLAY PAR
GALET     =      0.0000
ISTART    =     15.0000
NDATA     =     10.0000
>X
>

ENTER COMMAND:

GENOP FDAT1(3)←PSF4 FDAT3(4)
>CHANGE PAR ISTART:25
>X
>

ENTER COMMAND:

GENOP FDAT1(4)←PSF4 FDAT3(4)
>CHANGE PAR ISTART:35
>X
>
```

```
ENTER COMMAND:

MOVE 5 FDAT3(4)←FDAT1

ENTER COMMAND:

MOVE 15 FDAT3(4)←FDAT1(2)

ENTER COMMAND:

MOVE 25 FDAT3(4)←FDAT1(3)

ENTER COMMAND:

MOVE 35 FDAT3(4)←FDAT1(4)

ENTER COMMAND:

END
```

FDAT1:

475.0000	425.0000	375.0000	325.0000
564.0000	504.0000	444.0000	384.0000
651.0000	581.0000	511.0000	441.0000
736.0000	656.0000	576.0000	496.0000
819.0000	729.0000	639.0000	549.0000
454.0000	404.0000	354.0000	304.0000
89.0000	79.0000	69.0000	59.0000
176.0000	156.0000	136.0000	116.0000
261.0000	231.0000	201.0000	171.0000
344.0000	304.0000	264.0000	224.0000

FDAT3(4) före MOVE:

99.0000
196.0000
291.0000
384.0000
475.0000
564.0000
651.0000
736.0000
819.0000
0.0000
89.0000
176.0000
261.0000
344.0000
425.0000
504.0000
581.0000
656.0000
729.0000
0.0000
79.0000
156.0000
231.0000
304.0000
375.0000
444.0000
511.0000
576.0000
639.0000
0.0000
69.0000
136.0000
201.0000
264.0000
325.0000
384.0000
441.0000
496.0000

FDAT3(4) efter MOVE:

99.0000
196.0000
291.0000
384.0000
475.0000
564.0000
651.0000
736.0000
819.0000
454.0000
89.0000
176.0000
261.0000
344.0000
425.0000
504.0000
581.0000
656.0000
729.0000
404.0000
79.0000
156.0000
231.0000
304.0000
375.0000
444.0000
511.0000
576.0000
639.0000
354.0000
69.0000
136.0000
201.0000
264.0000
325.0000
384.0000
441.0000
496.0000

8.4. LETNING OCH INTERPOLLERING I TABELL.

Givet tabellvärden i (X_i, Y_i). Det intervall $[X_i, X_{i+1}]$ till vilket indata hör letas upp varefter utdata beräknas med linjär interpollering i tabellen.

Beräkningsfilen:

```
VECTOR X[25] Y[25]
INPUT X Y IN
OUTPUT UT
I=1
J=2
WHILE IN<X[I] OR IN>X[J] DO
I=I+1
J=J+1
ENDWHILE
UT=(IN-X[I])/(X[J]-X[I])*(Y[J]-Y[I])+Y[I]
END
```

Kommandon:

```
ENTER COMMAND:

GENKOD PSF5←HEJ5
WELL DONE - COMPILATION WAS SUCCESFUL

ENTER COMMAND:

GENOP FDAT3(3)←PSF5 FDAT3(1 2 4)
>X
>

ENTER COMMAND:

END
```

FDAT3:

```
* * * * * * * * * * * *
100   4  Xi          Yi          uT          IN
      1.0000      5.8300      6.3300     11.0000
      2.0000      5.8800      6.8150     20.7000
      3.0000      5.9300      7.2900     30.2000
      4.0000      5.9800      7.7550     39.5000
      5.0000      6.0300      8.2100     48.6000
      6.0000      6.0800      8.6550     57.5000
      7.0000      6.1300      9.0900     66.2000
      8.0000      6.1800      9.5150     74.7000
      9.0000      6.2300      9.9300     83.0000
     10.0000      6.2800      5.8350      1.1000
     11.0000      6.3300      6.2800     10.0000
     12.0000      6.3800      6.7150     18.7000
     13.0000      6.4300      7.1400     27.2000
     14.0000      6.4800      7.5550     35.5000
     15.0000      6.5300      7.9600     43.6000
     16.0000      6.5800      8.3550     51.5000
     17.0000      6.6300      8.7400     59.2000
     18.0000      6.6800      9.1150     66.7000
     19.0000      6.7300      9.4800     74.0000
     20.0000      6.7800      5.8350      1.1000
     21.0000      6.8300      6.2300      9.0000
     22.0000      6.8800      6.6150     16.7000
     23.0000      6.9300      6.9900     24.2000
     24.0000      6.9800      7.3550     31.5000
     25.0000      7.0300      7.7100     38.6000
     26.0000      7.0800      8.0550     45.5000
     27.0000      7.1300      8.3900     52.2000
     28.0000      7.1800      8.7150     58.7000
     29.0000      7.2300      9.0300     65.0000
     30.0000      7.2800      5.8350      1.1000
     31.0000      7.3300      6.1800      8.0000
     32.0000      7.3800      6.5150     14.7000
     33.0000      7.4300      6.8400     21.2000
     34.0000      7.4800      7.1550     27.5000
     35.0000      7.5300      7.4600     33.6000
     36.0000      7.5800      7.7550     39.5000
     37.0000      7.6300      8.0400     45.2000
     38.0000      7.6800      8.3150     50.7000
      ...          ...         ...        ...
```

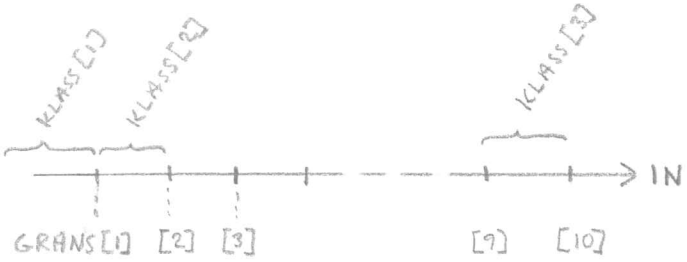
Den observante läsaren har säkert noterat att $Y_i=X_i/20+5.78$.

8.5. STATISTISK BEHANDLING AV MÄTDATA.

Först göres en klassindelning av mätdata.Därefter normeras klasserna varvid frekvensfunktionen erhålles.Ur denna beräknas sedan fördelningsfunktionen.

Beräkningsfil(klassindelning):

```
VECTOR GRANS[10] KLASS[10]
OUTPUT KLASS
INPUT IN GRANS
I=1
WHILE IN>GRANS[I] AND I<11 DO
I=I+1
ENDWHILE
KLASS[I]=KLASS[I]+1
END
```



Kommandon(klassindelning):

```
ENTER COMMAND:

GENKOD PSF6←HEJS
WELL DONE - COMPILATION WAS SUCCESFUL

ENTER COMMAND:

GENOP FDAT1←PSF6 FDAT3(1 4)
>X
>
```

Beräkningsfil(normering):

```
VECTOR KLASS1[10] KLASS2[10]
INPUT KLASS1
OUTPUT KLASS2
SUM=0
I=0
WHILE I<10 DO
I=I+1
SUM=SUM+KLASS1[I]
ENDWHILE
I=0
WHILE I<10 DO
I=I+1
KLASS2[I]=KLASS1[I]/SUM
ENDWHILE
END
```

Kommandon(normering):

```
ENTER COMMAND:

GENKOD PSF7←HEJ7
WELL DONE - COMPILATION WAS SUCCESFUL

ENTER COMMAND:

GENOP FDAT1(2)←PSF7 FDAT1
>X
>
```

Beräkningsfil(fördeln.funk.)

```
VECTOR FORD[10] FQ[10]
INPUT FQ
OUTPUT FORD
SUM=0
I=0
WHILE I<10 DO
I=I+1
SUM=SUM+FQ[I]
FORD[I]=SUM
ENDWHILE
END
```

Kommandon(fördeln.funk.)

```
ENTER COMMAND:

GENKOD PSF8←HEJ8
WELL DONE - COMPILATION WAS SUCCESFUL

ENTER COMMAND:

GENOP FDAT1(3)←FDAT1(2)\\\\\\\\PSF8 FDAT1(2)
>X
>

ENTER COMMAND:

END
```

FDAT3(1):

1.0000
2.0000
3.0000
4.0000
5.0000
6.0000
7.0000
8.0000
9.0000
10.0000
11.0000
12.0000
13.0000
14.0000
15.0000
16.0000
17.0000
18.0000
19.0000
. . .
96.0000
97.0000
98.0000
99.0000
100.0000

FDAT3(4):

2.0900
3.0600
4.0100
4.9400
5.8500
6.7400
7.6100
8.4600
9.2900
1.1000
1.9900
2.8600
3.7100
4.5400
5.3500
6.1400
6.9100
7.6600
8.3900
. . .
1.3400
1.3100
1.2600
1.1900
1.1000

FDAT1:

KLASS:	FQ:	FORD:
0.0000	0.0000	0.0000
35.0000	0.3500	0.3500
21.0000	0.2100	0.5600
13.0000	0.1300	0.6900
12.0000	0.1200	0.8100
6.0000	0.0600	0.8700
7.0000	0.0700	0.9400
3.0000	0.0300	0.9700
2.0000	0.0200	0.9900
1.0000	0.0100	1.0000

8.6. EXEMPEL PÅ FELUTSKRIFTER.

Antag att man vill göra en klassindelning och skriver följande beräkningsfil:

```
VECTOR GRANS[10] KLASS[10]
INPUT IN GRANS
OUTPUT KLASS
I=1
WHILE IN>X[I] AND I<11 DO          1.Skall stå GRANS[I]
I=I+1
ENDWHILE
UT[I]=UT[I]+1                      2.Skall stå KLASS[I]
END                                3.ENDWHILE saknas.
```

Felen 1-3 ger vid kompilering upphov till följande felutskrifter:

```
1. GENKOD PSFIL←HEJSA
   WHILE IN>X[I] AND I<11 DO
      YOU CAN DO BETTER THAN THIS, A VARIABLE
      IN RIGHT PART WAS NOT ASSIGNED


2. GENKOD PSFIL←HEJSA
   UT[I]=UT[I]+1
      DO NOT TRY WITH ME! YOU HAVE NOT DECLARED THE
      VECTOR IN LEFT PART


3. GENKOD PSFIL←HEJSA
   END
     YOU REALLY MAKE ME WORRIED.
     BAD NUMBER OF "WHILE"S OR "ENDWHILE"S
```