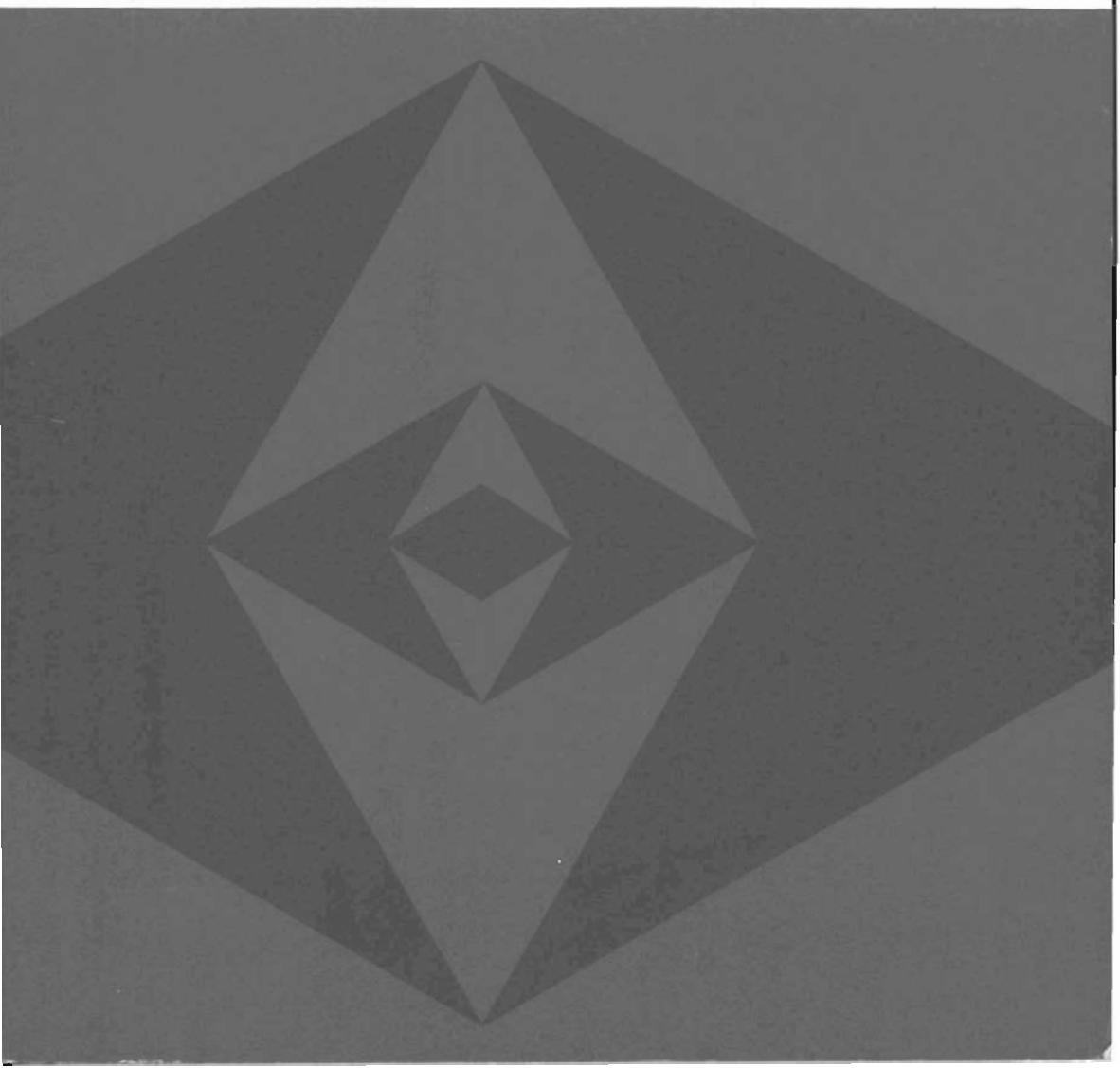
Tomas Ohlin

Knuth's förslag till in/utmatning i ALGOL

STUDENTLITTERATUR



Tomas Ohlin

Knuth's förslag till in/utmatning i ALGOL

STUDENTLITTERATUR
LUND

AKADEMISK FORLAG
KØBENHAVN

UNIVERSITETSFORLAGET
OSLO, BERGEN

I denna upplaga har bl a tillkommit Appendix 3.

Andra upplagan
Studentlitteratur
Lund 1968

INNEHÅLL

Kap. 1. INLEDNING

Inom den 'Revised Report on the Algorithmic Language ALGOL', som allmänt definierar programmeringsspråket ALGOL, har avsiktligt underlåtits att föreskriva hur in/utmatning av storheter bör ske. Detta förhållande har fört med sig att olika implementörer (förverkligare) av språket av datamaskintekniska skäl har valt olika metoder härför. Man har därmed måst konstatera dålig kompatibilitet (utbytbarhet) mellan olika ALGOL-versioner. Detta faktum har i någon mån kommit att ligga hela språket till last.

Erfarenhet har emellertid visat att det är möjligt att på ett ALGOL-bundet sätt definiera faciliteter för in/utmatning utan att ge avkall på maskinoberoendet. En underkommitté till 'ACM Programming Language Committee', under ordförandeskap av D. E. Knuth (CIT, Californien, USA), har i detta syfte utarbetat ett förslag, vilket först publicerades i 'Communications of the ACM', maj 1964. Detta förslag har utformats som en samling procedurer, avsedda att möjliggöra en mycket flexibel in/utmatning. Förslaget, som numera allmänt går under benämningen 'Knuth's förslag', har i en något modifierad form framlagts för International Organization for Standardization (ISO), vilken organisation dock ännu (mars 1967) ej har fattat något beslut angående dess standardisering.

Dessa utdragna förhandlingar har emellertid inte hindrat flera dominerande datamaskinleverantörer från att implementera det i sina ALGOL-kompilatorer. Sålunda existerar det idag för datamaskinerna i IBM:s 360-serie, Control Data:s 3000-serie, samma företags 6000-serie, samt General Electric:s 600-serie. Man kan säga att det nu ligger tyngd och erfarenhet bakom förslaget, såväl teoretiskt som praktiskt.

För undervisningsändamål har hittills på svenska språket icke funnits någon lämplig beskrivning av förslaget. Man har varit hänvisad antingen till ursprungsartikeln i CACM eller till ISO-publikationen. Då båda dessa är något svårlästa, har föreliggande skrift ansetts motiverad. Den beskriver Knuth's förslag i dess för ISO framlagda något modifierade form. Som referens syftas främst på ISO-publikationen.

Som utgångskunskaper för att kunna tillgodogöra sig innehållet i denna skrift förutsätts tämligen god kännedom om ALGOL 60. Det tillrådes därför att lägga undervisning om språkets allmänna uppbyggnad före genomgång av detta förslag.

För dem, som avser att nöja sig med inlärning av en begränsad del av förslagets faciliteter är en uppdelning gjord på det viset att Kap. 2 bör innebära tillräckligt material. Där presenteras ett antal principer och procedurer, som möjliggör in/utmatning av numeriska begrepp och text. Ett fortsatt studium är dock rekommendabelt för att få grepp om förslagets verkliga flexibilitet.

Av pedagogiska skäl används nedan endast stora bokstäver för ALGOL-symboler. Man kan därvid på ett klart sätt skilja programexempel o.d. från beskrivande text. Denna beskrivning är 'teoretiskt' hållen, d. v. s. inga hänsyn har tagits till avvikande implementörversioner av förslaget. För att något bespara användare av idag arbetet att uppleta modifikationer avseende aktuell datamaskinleverantörs hardwaremotiverade avvikelser, har emellertid tre kortfattade appendix bifogats. Dessa rör Control Datas, General Electrics samt IBM:s versioner.

Av erfarenhet är bekant att ett antal läsare endast 'skummar' en skrift av denna typ. För att dessa skumma individer på ett minimum av tid skall få möjlighet bilda sig någon uppfattning om förslaget uppträder redan här ett exempel på ett enkelt ALGOL-program, som använder sig av några av förslagets faciliteter. Detta program utmatar de konsekutiva naturliga talen 1, 2, ..., 500 med automatisk rad-, sid- och rubrikväxling. För beskrivning av variabeln NR se sidan 12, överst.

```
BEGIN
INTEGER     SID;
PROCEDURE   LAYOUT;
BEGIN
            H LIM (10, 112) ;
            V LIM (8, 48) ;
            V END (NYSIDA, NYSIDA, DUMMY) ;
            FORMAT (⌜↑ , (2/, 10 (-2ZD6B))⌝)
END ;
PROCEDURE   DUMMY ; ;
PROCEDURE   NYSIDA ;
BEGIN
            SID := SID + 1 ;
            OUTPUT 1 (NR, ⌜106B ⌜SID⌝ BZD⌝ , SID)
END ;
PROCEDURE   LISTA (UTMATA) ;
PROCEDURE   UTMATA ;
BEGIN
INTEGER     I ;
            FOR I := 1 STEP 1 UNTIL 500 DO UTMATA (I)
END ;
            SID :=0;
            OUTLIST (NR, LAYOUT, LISTA)
END
```

Man får hoppas att detta programexempel inbjuder till studium även av nästkommande sidor, och ej till motsatt aktion, i vilket fall författaren gjort sig skyldig till en pedagogisk felbedömning.

Tomas Ohlin

Kap. 2. ENKEL IN/UTMATNING

2.1 Format

Grundläggande för förslaget är användning av format för avbildning av storheter. Dessa format anges i strängar som används som aktuella parametrar i nedan beskrivna standardprocedurer. I detta kapitel behandlas endast talformat, standardformat samt rubrikformat.

DEFINITIONER:

1. En formatsträng består av ingen, en eller flera formatnotiser skilda åt av kommatecken, ytterst placerat mellan strängparenteser. Varje formatnotis består av ett format och/eller inställningskoder

2. En formatsträng tolkas från vänster till höger, svarande mot den lista av dataelement, som skall överföras (läsas/skrivas).

2.1.1 Talformat

2.1.1.1 Heltal

1. Formatet består av en följd av bokstäver Z, eller en följd av bokstäver D, eller en följd av bokstäver Z följda av bokstäver D, varvid varje bokstav avbildar en sifferposition i heltalet. Ett frivilligt plus- eller minustecken (förtecken) kan inleda eller avsluta formatet.

2. Bokstaven D innebär att en siffra alltid skall tryckas (även om den är 0). Ex. Talet 385 utmatat med formatet DDDD kommer att uppträda som 0385.

3. Bokstaven Z innebär att motsvarande siffra skall undertryckas om den är inledande nolla. I detta fall ersätts nollan med tecknet blank (␣). Ex. Talet 21 utmatat med formatet ZZD kommer att uppträda som ␣ 21.

4. Följder av bokstäver Z eller D kan skrivas NZ resp. ND, där N är ett konstant upprepande heltal (en replikator). Ex. 3Z och ZZZ är ekvivalenta begrepp.

5. Det eventuellt inledande eller avslutande förtecknet medför:
   a) Om det saknas, förmodas talet vara positivt.
   b) Om det är ett plustecken, uppträder det korrekta talttecknet alltid.
   c) Om det är ett minustecken, uppträder positiva tal med ett blankt tecken i st. f. förtecken, samt negativa med sitt minustecken.

6. Ett inledande förtecken kommer att på det yttre mediet placeras omedelbart till vänster om den första överförda siffran.

7. Exempel:

TAL	FORMAT	RESULTAT
2176	+ZZDDD	␣+2176
3	-ZZZDD	␣␣␣03
-45	-DDDD	-0045
0	ZZZ	␣␣␣
0	ZZD	␣␣0
390	2Z4D+	␣␣0390+

2.1.1.2 Tal med decimalpunkt

1. Formatet består här av bokstäver Z och/eller bokstäver D (med samma betydelse som för heltal), en punkt (.) eller bokstaven V, indikerande decimalpunkten, samt ett frivilligt plus- eller minustecken (med samma betydelse som för heltal).

2. Bokstäver Z får endast förekomma till vänster om decimalpunkten. Denna inskränkning gäller ej bokstäver D.

3. Konstanta replikationer före bokstäver Z resp. bokstäver D får förekomma på båda sidor av decimalpunkten.
Ex. 4ZD.3D och ZZZZD.DDD är ekvivalenta begrepp.

4. Bokstaven V indikerar endast var decimalpunkten skall befinna sig, och inget utrymme används på det yttre mediet. V har mening snarare vid inmatning än vid utmatning.

5. Exempel:

TAL	FORMAT	RESULTAT
146.776	ZZDD.DD	␣146.78
1.2	-3D.D	␣001.2
.0042	+3Z.3D	␣␣␣+.004
-142.78	-ZZDVD	-1428
-3394.7	ZZ4D.DD-	␣␣3394.70-
29.756	ZZD	␣30
-.0254	.3D-	.025-

2.1.1.3 Tal med exponent

1. Detta format innebär samma format som för tal med decimalpunkt, men här följt av en exponent-del (med basen 10). Exponentdelen består av Algol-grundsymbolen 'nedsänkt tia' ($_{10}$), följd av ett frivilligt förtecken samt en följd av bokstäver Z och/eller bokstäver D.

2. Reglerna för exponentdelens uppbyggnad (efter tecknet$_{10}$) är samma som för heltal. Förtecken till exponentdel får dock ej placeras efter exponenten.

3. Exempel:

TAL	FORMAT	RESULTAT
3075.2	$3D.DD_{10}+DD$	$307.52_{10}+01$
.021758	$ZZD_{10}+ZD$	$218_{10}\,␣\,-4$
917.2	$.3D_{10}+2D$	$.917_{10}+03$
.000312	$.DD_{10}-ZZZ$	$.31_{10}\,␣␣\,-3$

2.1.1.4 Trunkering

1. Heltalsformat eller format för tal med decimalpunkt kan få följas av bokstaven T för att indikera att utmatning skall ske trunkerat i stället för avrundat. Avrundning inträffar när inte trunkering är specificerad.

2. Exempel:

TAL	FORMAT	RESULTAT
-12.719	-2Z3D.2DT	␣␣-012.71
145.6	3ZDT+	␣145+
.012537	$-Z.DT_{10}+ZZ$	$␣\,1.2_{10}\,␣\,-2$

3. Följande definitioner gäller, när talet X skall nedbringas till d decimaler:

   Avrundning $10^{-d} \cdot entier(10^{d} \cdot X + 0.5)$

   Trunkering $10^{-d} \cdot sign(X) \cdot entier(10^{d} \cdot abs(X))$

2.1.1.5 Insättningar i talformat

1. Blanka tecken eller strängar får insättas på valfritt ställe inom ett talformat. Dessa insättningar kommer att uppträda på det yttre mediet.

2. Ett blankt tecken betecknas med bokstaven B. Denna bokstav får föregås av en konstant replikator.
   Ex. 3B är ekvivalent med BBB.

3. En insatt sträng inneslutes mellan strängparenteser. Även en insatt sträng får föregås av en konstant replikator.

4. Om vid inmatning strängar eller blanka tecken är insatta i talformat, kommer motsvarande antal karaktärer på inmatningsmediet att överhoppas.

5. Inställningskoder (se avsnitt 2.1.4) får inleda och/eller avsluta talformat.

6. Exempel:

TAL	FORMAT	RESULTAT
3972	D2B3D	3␣␣ 972
271	⌜SVAR=⌝ 4D	SVAR=0271
-195.763	⌜HELT⌝ -4ZVB ⌜DEC⌝ BDD	HELT␣-195␣DEC␣76
44865.5	2ZB2D. DBT$_{10}$+DD	44␣ 86.5 ␣$_{10}$ +01
56	-Z2⌜PA⌝D	␣5PAPA6

2.1.2 Standardformat

1. Om vid överföring av tal något av nedanstående tre fall föreligger kommer talet att överföras i enlighet med ett standardiserat format:
   a) om inget format anges i formatsträngen.
   b) om angivet format ej räcker till.
   c) om bokstaven N anges såsom format.

2. Vid inmatning definieras tal enligt reglerna för ALGOL 60. Som avgränsare mellan tal räknas endera av
   a) en för tal i ALGOL 60:s mening ej giltig karaktär, t.ex. kommatecken, någon bokstav etc.
   b) K eller flera blanka tecken. (En följd av K-1 eller färre blanka tecken ignoreras.) Heltalet K, som skall vara större än eller lika med 1, specificeras av implementören, och må alltså variera från maskin till maskin.
   c) slut på rad. Om emellertid aktuell rad endast innehåller blanka tecken eller karaktärer som inte ingår i talet självt, betraktas talet ej avslutat förrän vid uppträdande på någon nästkommande rad av en avgränsare enligt a) eller b).

3. Ett tal av reell typ utmatas som decimalbråk med exponent.

4. Ett tal av heltalstyp utmatas som heltal.

5. Med standardformat utmatade tal skall kunna läsas in igen med standardformat, d.v.s. varje tal skall vid utmatningen vara omedelbart följt av minst K blanka tecken.

6. Inställningskoder får inleda och/eller avsluta standardformat.

2.1.3 Rubrikformat

1. Detta format används utan syfte på överföring av värdet hos någon ALGOL-symbol.

2. Formatet består av insättningar (strängar och/eller blanka)

3. Vid utmatning kommer överföring av de insatta symbolerna att ske.

4. Vid inmatning kommer symbolerna endast att överhoppas.

5. Inställningskoder får inleda och/eller avsluta rubrikformat.

6. Exempel:

   ⌜BRA⌝ B ⌜EXEMPEL⌝ / kommer vid utmatning att innebära överföring av texten BRA ␣ EXEMPEL, följt av ny rad.

2.1.4 Inställningskoder

1. Följande inställningskoder existerar:

SYMBOL	INNEBÖRD
/	ny rad (radvisaren ökas med en enhet och positionsvisaren inställes på noll [för definition se avsnitt 4.2]).
↑	ny sida (radvisaren förflyttas till ny sida och positionsvisaren inställs på noll).
J	förflyttning (av positionsvisaren) till nästa tabulatorposition.

2. Symbolerna / , ↑ och J får föregås av replikatorer.

3. En formatnotis kan bestå av endast inställningskoder.

4. En formatnotis kan vara uppbyggd av inställningskoder samt ett godtyckligt format. Härvid gäller dock att inställningskoderna måste uppträda först och/eller sist i formatnotisen.

5. Exempel:

   Formatnotisen ↑ -ZD.D/ innebär ny sida, talformatet -ZD.D samt ny rad.

2.1.5 Formatupprepning

1. En formatnotis eller en godtyckning grupp av formatnotiser kan upprepas ett konstant antal gånger genom att notisen resp. gruppen av notiser innesluts mellan parenteser och den inledande (vänster-) parentesen omedelbart föregås av en konstant replikator. Om ingen sådan replikator föregår vänsterparentesen indikeras oändlig upprepning av notisen resp. gruppen av notiser.

2. Inställningskoder som föregår eller efterföljer ett upprepat format skall vara avskilda därifrån med kommatecken, d.v.s. utgöra egna formatnotiser. Se exempel i Kap. 1, proceduren FORMAT.

3. Exempel:

FORMAT	INNEBÖRD
2(-ZDD.D)	-ZDD.D, -ZDD.D
(3D2 ⌜PUH⌝ B,-ZD)	oändlig upprepning av: de två talformaten 3D och -ZD samt mellan dem insättningen PUHPUH ␣ .

2.2 Icke formatbunden in/utmatning av tal

Nedan beskrivs två par standardprocedurer för in/utmatning av reella tal och fält. I dessa och i ett stort antal därefter beskrivna procedurer förekommer som första parameter den formella parametern NR. Dess aktuella motsvarighet skall vara ett aritmetiskt uttryck. Värdet av detta uttryck skall vara kodbeteckningen för (ifrågavarande fil på) den yttre enheten. Denna kods utseende varierar från det ena maskinsystemet till det andra. För varje system finns dock en standardbeteckning för inmatning (normalt via kortläsare) samt en standardbeteckning för utmatning (normalt på radskrivare).

OBS! I nedanstående programexempel anses en adekvat kodbeteckning insatt i st. f. NR. Någon variabel NR är alltså ej deklarerad.

2.2.1 Procedurerna INREAL och OUTREAL

De två procedurerna INREAL och OUTREAL innebär in- resp. utmatning av ett tal per proceduranrop.

```
INREAL (NR, VARIABEL)
OUTREAL (NR, VARIABEL)
```

Regler:

1. Aktuell motsvarighet till VARIABEL skall vara av reell eller heltalstyp.

2. INREAL inmatar från det yttre mediet närmast kommande tal i ALGOL 60:s mening. Den som andra procedurparameter förekommande aktuella variabeln tilldelas detta tals värde.

3. OUTREAL utmatar på det yttre mediet värdet hos det som andra parameter förekommande aktuella aritmetiska uttrycket.

4. Med proceduren OUTREAL utmatade tal skall kunna läsas av proceduren INREAL.

5. Exempel:

```
BEGIN
REAL A;
     INREAL (NR, A);
     OUTREAL (NR, A)
END
```

Det inlästa talet trycks härmed. Formatet för det tryckta talet är ej fixerat i Knuth's rapport. Jämför dock regel 4 ovan.

2.2.2 Procedurerna INARRAY och OUTARRAY

De två procedurerna INARRAY och OUTARRAY innebär in- resp. utmatning av ett fält per proceduranrop.

```
INARRAY (NR, FÄLT)
OUTARRAY (NR, FÄLT)
```

Regler:

1. Aktuell motsvarighet till FÄLT skall vara av reell eller heltalstyp.

2. INARRAY inmatar från det yttre mediet en följd av tal (i ALGOL 60:s mening) och placerar dem i fältet med angiven aktuell beteckning. Antalet inmatade tal bestäms av fältdeklarationen. Hela detta fält fylls således med tal. Skulle fältdeklarationen vara flerdimensionell, fylls de inmatade talen radvis i den flerdimensionella matrisen. Härmed menas att högerindex varierar snabbare än vänsterindex.

3. OUTARRAY utmatar på det yttre mediet samtliga de tal, som finns lagrade i fältet med angiven aktuell beteckning. Är fältdeklarationen flerdimensionell utmatas talen radvis.

4. Med proceduren OUTARRAY utmatade tal skall kunna läsas av proceduren INARRAY.

5. Med proceduren OUTREAL utmatade tal skall kunna läsas av proceduren INARRAY, och motsvarande beträffande OUTARRAY och INREAL.

6. Exempel:

```
BEGIN
ARRAY   A [1:3, 1:2];
        INARRAY (NR, A) ;
        OUTARRAY (NR, A)
END
```

Programmet innebär inläsning av de sex talen i fältet A, samt omedelbar utmatning av desamma. Formatet för utmatningen är implementörberoende.

2.3 Formatbunden in/utmatning av tal och text

Nedan beskrivs en familj av standardprocedurer bl.a. för in/utmatning av tal och textsträngar. Den i procedurerna förekommande parameter NR har samma betydelse som i avsnitt 2.2.

Procedurerna har utseendet

```
INPUT N (NR, FORMATSTRÄNG,X1, X2, ...., XN)
OUTPUT N (NR, FORMATSTRÄNG, X1, X2, ....., XN)
```

N kan anta värdena 0, 1, 2, ..., 9. Det är att notera att N före vänsterparentesen ovan icke är en parameter utan en del av proceduridentifieraren. Familjen består alltså av 20 helt separata procedurer. (Grammatiken för ALGOL 60 förbjuder procedurer med ett variabelt antal parametrar.)

I formatsträngen anges formaten för de tal eller textsträngar etc., vilka förekommer som de aktuella parametrarna X1, X2, ..., XN. Det kan noteras, att med denna procedurfamilj kan överföras även logiska storheter. Beskrivning häröver återfinns från och med kap. 3.

Om formatsträngen är tom, d.v.s. har utseendet ⌜⌝, använder formatkoden N eller helt saknas (kommatecken måste dock utsättas för att antalet procedurparametrar skall bli korrekt), kommer standardformat till användning. Detta blir även fallet för resterande storheter om formatsträngen ofullständigt beskriver storheterna X1, X2, ..., XN.

Exempel:

```
BEGIN
REAL A, B, C, D;
     A:= 14.7; B:= 5.98; C:= 115.977; D:= 1.2;
     OUTPUT 4 (NR, ⌜↑,2(2ZD.DD)4B, 4Z.2D⌝ , A, B, C, D)
END
```

Programmet medför på ny sida följande utmatning:

␣14.70 ␣␣ 5.98 ␣␣␣␣␣ 115.98+1.200000000$_{10}$ +000

Det sista talet utmatas med standardformat (här exemplifierat med D.9D$_{10}$ +3D) eftersom formatsträngens notiser ej räcker till mer än 3 tal.

Exempel:

Utmatning av rubrik samt kvadratisk matris, vars storlek samt element först läses in.

```
BEGIN
INTEGER   N;
          INPUT1(NR, ⌜D⌝, N);
BEGIN
INTEGER   I, J;
ARRAY     A [1:N, 1:N];
          INARRAY (NR, A);
          OUTPUT 0(NR, ⌜↑⌝⌜EXEMPEL ␣ PÅ ␣ MATRISUTMATNING⌝//
                    ⌜NR⌝ B⌝ );
          FOR I:= 1 STEP 1 UNTIL N DO OUTPUT1 (NR, ⌜4BD⌝ , I);
          FOR I:= 1 STEP 1 UNTIL N DO
BEGIN
          OUTPUT1(NR, ⌜// D 2B⌝ , I);
          FOR J:= 1 STEP 1 UNTIL N DO
          OUTPUT 1 (NR, ⌜2BD.D⌝ , A [I, J] )
END
END
END
```

I detta exempel är ett par antaganden gjorda:

1. Matrisens storleksordning är högst 9 * 9.

2. Matriselementen är positiva, med en heltalssiffra och en decimal.

Aktuell motsvarighet till NR kan givetvis icke överensstämma både vid in- och utmatningen i exemplet ovan.

För att möjliggöra mer avancerad matrisutmatning, främst avseende layout, hänvisas till de följande kapitlen.

Kap. 3 YTTERLIGARE FORMATTYPER

3.1 Strängformat

Detta format används för utmatning av strängar. För inmatning av strängar skall i stället alfaformat användas.

Regler:

1. Formatet består av en eller flera bokstäver S. En konstant replikator får föregå bokstaven.
   Ex. 4S har samma innebörd som SSSS.

2. Varje bokstav S motsvarar en överförd grundsymbol.

3. Om den aktuella strängen är längre än det indikerade antalet bokstäver S, överförs endast motsvarande antal grundsymboler från vänster räknat. Om strängen är kortare, tilläggs blanka tecken till höger i strängen.

4. Insättningar i form av blanka tecken eller strängar får göras i strängformat. Härvid gäller samma regler som för talformat. Inställningskoder får inleda och/eller avsluta strängformat.

5. Exempel:

STRÄNG	FORMAT	RESULTAT
⌜A⌝	S	A
⌜POVEL⌝	5S	POVEL
⌜PROGRAM⌝	SSSS	PROG
⌜AJ⌝	4S	AJ ␣␣
⌜12345⌝	B3SBB2S	␣ 123␣␣ 45
⌜SVARA⌝	4S ⌜=⌝ SB	SVAR = A␣

I många fall kan alfaformat användas i stället för strängformat. I nedanstående exempel är detta dock ej möjligt:

```
BEGIN
PROCEDURE  UT MED DIG (R) ;
STRING     R ;
           OUTPUT 1 (NR, ⌜4S⌝, R) ;
           .
           .
           UT MED DIG (⌜SVEN⌝) ;
           .
           .
           UT MED DIG (⌜ZON⌝) ;
           .
END
```

Programmet medför utmatning av texten SVENZON␣ .

3.2 Alfaformat

Detta format används för överföring av ALGOL-grundsymboler. Grundsymbolerna lagras i maskinen som heltal enligt en viss intern kod, varierande från maskin till maskin. I detta lagringssätt ligger alltså skillnaden mellan alfaformat och strängformat. Eftersom intern kodrepresentation varierar mellan olika maskiner, blir allt aritmetiskt arbete utom vid användning av = och ≠ maskinberoende. Maskinoberoende arbete med grundsymboler möjliggörs emellertid av standardfunktionen EQUIV. Denna är deklarerad som en INTEGER PROCEDURE , och dess parameter är en sträng, som skall bestå av 1 grundsymbol. Alfaformat kan användas både för in- och utmatning.

Regler:

1. Alfaformat består av bokstaven A.

2. Bokstaven A motsvarar en överförd grundsymbol.

3. Inställningskoder får inleda och/eller avsluta alfaformat.

4. Exempel (vid inmatning):

GRUNDSYMBOL	FORMAT	RESULTAT (lagrat som heltal enligt intern kod)
B	A	B
;	A	;
BEGIN	A	BEGIN

(Det externa typografiska utseendet av BEGIN blir beroende av aktuell datamaskins representation av 'understrykning'). Om grundsymbolen kolon (:) avses läses till variabeln U med alfaformat, kommer därefter satsen

```
IF U ≠ EQUIV (':') THEN GOTO FEL;
```

att möjliggöra kontroll att läsningen skett korrekt.

3.3. Booleskt format

Detta format används för att överföra booleska storheter.

Regler:

1. Booleskt format består av bokstaven P eller bokstaven F.

2. Om P används och den booleska storheten har värdet 'sann' överförs siffran 1, i motsatt fall siffran 0 (noll).

3. Om F används och den booleska storheten har värdet 'sann' överförs grundsymbolen TRUE, i motsatt fall grundsymbolen FALSE. Det exakta externa typografiska utseendet av TRUE resp. FALSE är beroende av aktuell datamaskins representation av 'understrykning'.

4. Insättningar i form av blanka tecken eller strängar får göras i booleskt format. Härvid gäller samma regler som för talformat. Inställningskoder får inleda och/eller avsluta booleskt format.

5. Exempel:

SANNINGSVÄRDE	FORMAT	RESULTAT
TRUE	2BPB	␣␣1␣
FALSE	⌜SVARET␣ÄR⌝ BF ⌜,⌝	SVARET␣ÄR␣FALSE.

3.4 Ickeformat

Detta format används för att överföra storheter med bibehållande av intern maskinrepresentation. Storheter som är utmatade med ickeformat måste läsas in med samma format.

Regler:

1. Ickeformat består av bokstaven I, bokstaven R eller bokstaven L.

2. Bokstäverna har följande betydelser:
   I för överföring av heltal (variabler deklarerade som INTEGER).
   R för överföring av reella tal (variabler deklarerade som REAL).
   L för överföring av booleska värden (variabler deklarerade som BOOLEAN).

3. Inställningskoder får inleda och/eller avsluta ickeformat.

4. Citat från ursprungsförslagets text:
   "The precise behaviour of this format, and particularly its interaction with other formats, is undefined in general".

5. Exempel:

```
BEGIN
REAL A ;
     A := 3.14159 ;
     OUTPUT 1 (NR, ⌜↑⌜INTERN ␣MASKINREPRESENTATION ␣AV ␣ ETT
     ␣ NÄRMEVÄRDE␣ TILL ␣ TALET ␣ PI ␣ HAR ␣ UTSEENDET␣⌝R⌝,
     A)
END
```

3.5 Sammanställning över formatkoder

SYMBOL	BETYDELSE
A	Alfaformat
B	Blankt tecken
D	Siffra (utan nollundertryckning).
F	Booleskt format (TRUE / FALSE)
I	Ickeformat, INTEGER
J	Tabulatormärkning
L	Ickeformat, BOOLEAN
N	Standardformat
P	Booleskt format (1/0)
R	Ickeformat, REAL
S	Strängformat
T	Trunkering
V	Underförstådd decimalpunkt
X	Variabel replikator
Z	Nollundertryckt siffra
+	Förtecken + eller -
-	Förtecken ␣ eller -
10	Exponentdelindikator (nedsänkt tia)
( )	Upprepad formatnotis
,	Formatavskiljare
/	Ny rad
↑	Ny sida
⌜⌝	Strängparenteser
.	Decimalpunkt

Kap. 4. LAYOUT-PROCEDURER

Som framgått av det tidigare finns ett antal standard-förfaranden givna vid in/utmatning. Med den ovan beskrivna formathanteringen kan dessutom i procedurerna INPUT N/OUTPUT N tämligen avancerad in/utmatning dirigeras. Full flexibilitet erhåller man emellertid först vid användning av begreppet listprocedurer, vilket genomgås i kap. 5.

Listprocedurer beskriver listor av element och befattar sig å priori inte med utseendet av elementen vid överföringen. Detta utseende, layouten, beskrivs i stället separat i en layout-procedur, som anropas när aktuell överföring skall ske.
En dylik layout-procedur skall ej ha procedurparametrar.

Det måste understrykas, att nedan genomgångna beskrivande procedurer, avsedda att möjliggöra flexibel uppbyggnad av en av användaren deklarerad layout-procedur, endast kan användas i samband med de i kap. 5 beskrivna procedurerna INLIST/OUTLIST. Ett i ett program helt fristående anrop av t. ex. den beskrivande proceduren NO DATA får alls icke avsedd verkan. Endast om anropet förefinns i en av användaren deklarerad layoutprocedur (använd som parameter i INLIST/OUTLIST, varom mera senare), utför det sitt rätta arbete.

En beskrivande procedur utför aldrig något explicit överföringsarbete.

Ett korrekt anrop av en beskrivande procedur får till följd att ett antal 'dolda variabler' tilldelas vissa värden. Dessa värden styr sedan den aktuella överföringen. De dolda variablerna är sannolikt tillkomna endast på teoretisk grund. (De beskrivs också i ursprungsförslaget med attributet 'mytiska'.) Man får därvid möjlighet beskriva aktuella förlopp på ett ALGOL-liknande sätt. Resultaten innebär att i maskinspråk ett antal maskinadresser insätts på adekvata ställen och det kan därför synas överflödigt för en användare att befatta sig med dessa 'dolda variabler'. De lämnas därför i detta sammanhang helt därhän. Den specialintresserade finner emellertid en hel del om dem i ursprungsförslaget.

De för layoutprocedurer till buds stående beskrivande procedurerna är:
FORMAT, H LIM, V LIM, H END, V END, TABULATION samt NO DATA.

4.1 Procedurfamiljen FORMAT/FORMAT N

Denna procedurfamilj består av 10 skilda procedurer:
FORMAT 0, FORMAT 1, FORMAT 2, ..., FORMAT 9.

Beteckningen FORMAT (utan efterföljande siffra) får användas synonymt med den första individen FORMAT 0.

Proceduren FORMAT N har N+1 parametrar. Det är att observera att N här icke är någon variabel storhet, utan en del av procedurnamnet. Den aktuella bokstaven N kan explicit ej användas, utan står här för någon av siffrorna 0, 1, 2, ... 9.

Låt oss emellertid tillåta oss att skriva

FORMAT N (STRÄNG, X1, X2, ..., XN)

Härur framgår att som första aktuell parameter skall förekomma en sträng. Denna skall vara en formatsträng, uppbyggd efter regler som angivits i kap. 2 och 3, med ett intressant tillägg. Före genomgång av detta tillägg kan emellertid konstateras att strängen (som enda parameter) i proceduren FORMAT 0/ FORMAT tillåts syfta på en formatsträng. Den kan alltså i detta enda fall vara en formell procedurparameter, varvid dock motsvarande aktuella parameter måste vara en formatsträng (i detta fall utan X-replikatorer, se nedan).

Tillägget till uppbyggnaden av formatsträngen som första aktuell parameter i FORMAT N, i själva verket ett av motiven för hela denna procedurfamiljs existens, innebär att en eller flera (upp t o m 9 st) variabla replikatorer kan insättas. Dessa s.k. X-replikatorer beskrivs med bokstaven X. De skall till antalet i ett anrop av FORMAT N vara exakt N st. Den aktuella formatsträngen i t.ex. proceduren FORMAT 3 skall alltså innehålla 3 st. X-replikatorer, varken fler eller färre. En X-replikator får insättas varhelst en konstant replikator tillåts insättas, se kap. 2 och 3.

De aktuella motsvarigheterna till parametrarna X1, X2, ..., XN ger värden åt motsvarande X-replikatorer, från vänster räknade, vid utförande av programmet. Dessa aktuella parametrar skall vara icke-negativa heltal. De kan även via någon yttre procedur syfta på icke-negativa heltal. I procedurerna FORMAT N är variablerna X1, X2, ..., XN emellertid värdeanropade, varför man bör se upp med att ändra deras värden.

I det enklaste fallet är X1, X2, ..., XN konstanter eller heltalsvariabler.

Exempel:

```
R := 7;
FORMAT 2 ('XB.XD10 + DD', 5, R);
```

är ekvivalent med FORMAT ('5B.7D10 + DD');

Med hjälp av de ovan beskrivna 10 procedurerna löses alltså problemet med 'variabelt format', annars en stötesten i vissa programmeringsspråk.

4.2 Procedurerna H LIM och V LIM

För styrning av överföring vad beträffar rad- resp. sidslut finns 4 procedurer att tillgå, varav de två första har utseendet

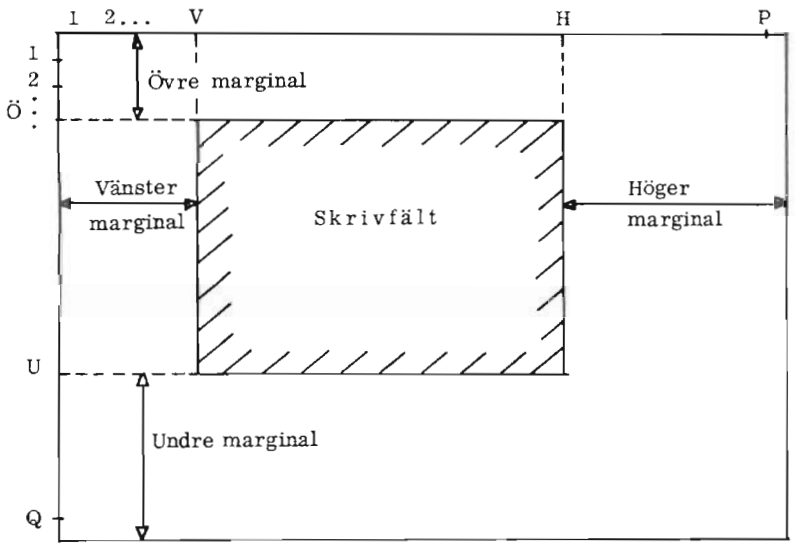
```
H LIM (V, H)
V LIM (Ö, U)
```

Dessa procedurer har mening för olika yttre medier men är sannolikt här mera lättsmälta vid syftning på utmatning på radskrivare. De aktuella motsvarigheterna till parametrarna V, H, Ö, U indikerar då avgränsning på utmatningspapperet av det fält, inom vilket skrivning tillåts ske. Det föreligger symmetri vid användning av horisontell marginalstyrning, med H LIM, samt vertikal, med V LIM.

Horisontellt format på papperet beskrivs med V, H, P. Vertikalt format i sin tur beskrivs med Ö, U, Q. Dessa har följande betydelser:

V	vänster horisontell marginal
H	höger " "
P	höger absolut (fysikalisk) marginal
Ö	övre vertikal marginal
U	undre " "
Q	undre absolut (fysikalisk) marginal

Av dessa sex parametrar används som ovan synes fyra i procedurerna H LIM och V LIM. De resterande två, P och Q, är för varje givet yttre fysiskt medium konstanta. Parametern P talar om maximala antalet karaktärer (tecken) som ryms på en horisontell rad på papperet (t. ex. 132), samt parametern Q det maximala antalet rader per sida på papperet (t. ex. 66). Dessa två parametrar kan inte nås av en användare på annat sätt än genom anrop av proceduren SYSPARAM (varom mera nedan), och detta kan förmodas vara aktuellt mera i undantagsfall. Betydelsen av de sex parametrarna framgår ur nedanstående figur.



Aktuella motsvarigheter till samtliga parametrar V, H, Ö och U skall vara aritmetiska uttryck, ur vilka kan utvärderas icke-negativa heltal. Aktuellt V måste väljas mindre än aktuellt H för att något fysiskt utrymme verkligen skall definieras, där skrivning skall ske. Av samma skäl måste aktuellt Ö väljas mindre än aktuellt U. Eftersom användaren normalt inte håller ordning på samtliga mediers P och Q kan fall inträffa då aktuellt H är större än P resp. aktuellt U är större än Q. Härvid kommer att som högermarginal väljas det minsta av aktuellt H och P, samt som undre marginal det minsta av aktuellt U och Q.

Man har alltså:

Vänster marginal	= V
Höger marginal	= Min (H, P)
Övre marginal	= Ö
Undre marginal	= Min (U, Q)

Om inte någon av procedurerna H LIM eller V LIM anropas, kommer följande värden att gälla:

V = Ö = 1
H = U = ett givet stort tal

Detta innebär att hela papperet kommer att användas.

Hittills har i detta avsnitt endast radskrivareutmatning beaktats i samband med H LIM och V LIM. Andra yttre medier nödvändiggör något annorlunda tolkning av de sex parametrarna.

Hålkort med 80 kolumner har P = 80 och Q oändlig. (Vertikal kontroll kan här förefalla meningslös, men ↑ kan t.ex. av implementören användas för att insätta ett speciellt hålkort, av styrtyp e.d.).

För magnetbandshantering, i de fall då godtyckligt långa block tillåtes, kan både P och Q betraktas som oändliga.

För hålremsa kan det förefalla svårt att ge någon mening åt vertikal kontroll, P skulle dock möjligen kunna innebära antalet karaktärer, som läses åt gången. Varje utformare av en ALGOL-kompilator har uppenbarligen tämligen fria händer att för vissa medier välja en fysisk tolkning av P och Q.

Sedan nu i användarens program fysiska gränser satts för aktuellt medium med hjälp av H LIM och V LIM, dirigeras automatiskt rad- och sidkontroll på följande vis:

Ny rad kommer att utföras vid ettdera av följande fall:

1. normal begäran om radbyte, angivet med ett snedstreck (/) i formatet.

2. H-spill, som inträffar när en eller flera karaktärer överföres (eller en förflyttning till nästa tabulatormärke begäres), som skulle passera den angivna H-gränsen.

3. P-spill, som inträffar när en eller flera karaktärer överföres (eller en förflyttning till nästa tabulatormärke begäres), som skulle passera P-gränsen, men icke H-gränsen. I detta fall förutsättes alltså H > P.

Ny sida kommer att utföras vid ettdera av följande fall:

1. normal begäran om sidbyte, angivet med en pil (↑) i formatet.

2. U-spill, som inträffar när en eller flera karaktärer överföres, som skulle uppträda på rad nr U + 1.

3. Q-spill, som inträffar när en eller flera karaktärer överföres, som skulle uppträda på rad nr Q + 1. I detta fall förutsättes alltså U > Q.

Det finns skäl att här införa begreppet positionsvisare, innebärande en tänkt visare, som pekar på den position på t.ex. raden på papperet, där aktuell karaktär just har tryckts. Positionsvisaren, som kan betecknas POSV, är alltså en heltalsvariabel, som lyder följande restriktion:

$$0 \leq V \leq POSV \leq \min (H, P)$$

På motsvarande sätt införes begreppet radvisare, betecknad RADV, och innebärande en visare, som pekar på t.ex. den rad på papperet, där tryckning just har skett. Radvisaren lyder restriktionen:

$$0 \leq Ö \leq RADV \leq \min (U, Q)$$

Dessa visare kommer att vara till nytta senare i detta kapitel.

Exempel på anrop:
```
H LIM (5,85);
V LIM (3,50);
```

Innebörd: vänster marginal sättes till 5, höger marginal till 85, övre marginal till 3 samt undre marginal till 50.

Det bör noteras att angivna marginalvärden ingår i skrivfältet, d.v.s. angivna intervall är slutna.

4.3 Procedurerna H END och V END

Med hjälp av de ovan beskrivna procedurerna H LIM och V LIM automatiseras alltså rad- resp. sidbyten. Man kan t.ex. tänka sig att mata ut ett antal tabeller med tal eller text enbart genom att skriva en enda FOR-sats, utan att alltså bry sig om rad- resp. sidslut. Hur detta tillgår kommer att framgå nedan. I många fall kan emellertid vissa speciella åtgärder vara önskvärda just i samband med dessa byten. Det kan t.ex. röra isättning av radnummer längst till vänster på varje ny rad eller tryckning av rubrik eller sidnummer högst upp på varje ny sida. För dylik hantering finns ytterligare två procedurer till förfogande:

```
H END (PR, PH, PP)
V END (PS, PU, PQ)
```

Dessa procedurer behöver ej anropas för att rad- resp. sidbyte skall ske enligt ovan, men de möjliggör alltså utförandet av speciella åtgärder i samband därmed. Procedurerna (som alla dessa beskrivande procedurer) anropas i en layout-procedur för att få korrekt verkan.+) Samtliga parametrar PR, PH, PP, PS, PU och PQ är i sin tur specificerade som procedurer. Dessas aktuella motsvarigheter skall tillhandahållas av användaren, och alltså deklareras bland programmets övriga deklarationer. Den inbördes ordningen mellan deklarationer är som bekant godtycklig. De sex formella parametrarnas aktuella motsvarigheter aktiveras (anropas och genomlöps) enligt nedan:

Formell parameter	Motsvarande aktuell procedur aktiveras
PR	Efter det att normalt radbyte skett, orsakat av uppträdande av ett snedstreck (/) i formatet.
PH	Efter det att radbyte skett, som orsakats av H-spill.
PP	Efter det att radbyte skett, som orsakats av P-spill.
PS	Efter det att normalt sidbyte skett orsakat av uppträdandet av en pil (↑) i formatet.
PU	Efter det att sidbyte skett, som orsakats av U-spill.
PQ	Efter det att sidbyte skett, som orsakats av Q-spill.

Kod för utförande av särskilda åtgärder vid ovan nämnda tillfällen tillhandahålles alltså av användaren i egna procedurer. Det är sannolikt mera sällsynt att en användare skulle vilja utföra sådana åtgärder vid samtliga ovan beskrivna tillfällen. Det är då lämpligt att använda sig av en s.k. dummy-procedur vid de tillfällen, då inga särskilda åtgärder kräves. En sådan procedur har utseendet

```
PROCEDURE DUMMY ; ;
```

Denna procedurs kropp är som synes tom, d.v.s. består av en tom sats (mellan de båda tecknen semikolon). Proceduren utför alltså inget arbete, men är likafullt mycket användbar. (Vissa kompilatorer tillhandahåller denna procedur, under just ovanstående namn, och då behöver alltså användaren icke själv deklarera den.)

I vissa fall skall samma åtgärd utföras vid flera av ovan nämnda tillfällen. I dessa fall används också lämpligen samma aktuella procedur på flera ställen. De sex parametrarna ovan behöver alltså givetvis icke ha olika aktuella motsvarigheter.

Det är att notera, att de av användaren tillhandahållna procedurer, som används som aktuella motsvarigheter till PR, PH, ..., icke kan ha egna procedurparametrar.

Nedanstående program exemplifierar användning av de ovan beskrivna procedurerna. Endast layout-delen är här medtagen. Man kan tänka sig en fortsättning på programmet, där utmatning av text begärs, som skulle sträcka sig över ett antal sidor. För

+) jfr dock kap. 5: de allra sista meningarna.

att fullständiga exemplet behövs emellertid tillgång till begreppet listprocedur , vilket beskrivs först i kap. 5. Det förutsätts i exemplet att aktuell yttre enhet, beskriven av NR, har P > 110 och Q > 60.

```
BEGIN
INTEGER     RADNR, SIDNR;
PROCEDURE   DUMMY; ;
PROCEDURE   RADBYTE;
BEGIN
            RADNR := RADNR + 1;
            OUTPUT 1 (NR,'' RAD' BZZD3B', RADNR)
END;
PROCEDURE   SIDBYTE;
BEGIN
            SIDNR := SIDNR + 1;
            OUTPUT 1 (NR,' 100B'SID' BZD', SIDNR)
END;
PROCEDURE   LAYOUT;
BEGIN
            FORMAT ('↑');
            H LIM (3, 110);
            V LIM (4, 60);
            H END (RADBYTE, RADBYTE, DUMMY);
            V END (SIDBYTE, SIDBYTE, DUMMY)
END;
            .
            .
            .
            .
```

Detta program skulle som inledning, efter samtliga deklarationer, ha en sats med innebörden att variablerna RADNR och SIDNR ges värdet 0 (noll). Detaljer lämnas emellertid här, då exemplet endast bör ses principiellt. Utmatningen från programmet innebär att exempelvis sida 7, då 348 rader redan utmatats, skulle börja:

```
                                                    SID ␣␣ 7
RAD␣ 349 ␣␣␣ TEXT ETC ....
RAD␣ 350 ␣␣␣ .....
```

4.4 Proceduren TABULATION

Användning av bokstaven J i ett format syftar på förflyttning av positionsvisaren (se 4.2 för definition) till nästa tabulatormarkering. Dessa markeringar sätts med anrop av proceduren

```
TABULATION (N)
```

Aktuell motsvarighet till parametern N skall vara ett aritmetiskt uttryck, som utvärderat anger vidden på tabulatorfältet. Om proceduren TABULATION inte anropas, betraktas tabulatorfältets vidd vara 1 (en) tryckposition, och användning av J

får samma innebörd som användning av B.

Exempel:

```
TABULATION (6) ;
```

I och med detta anrop kommer tabulatormarkeringar att insättas vid tryckpositionerna V + 6, V + 12 osv på raden.

4.5 Proceduren NO DATA

Vid inmatning kan kontinuerlig läsning utan tanke på eventuellt slut på data avklaras med hjälp av proceduren

```
NO DATA (L)
```

Aktuell motsvarighet till parametern L är här ett läge. Identifieraren för det läge i programmet, dit hopp skall ske om data tagit slut på inmatningsmediet, insättes som aktuell parameter. Proceduren NO DATA har mening endast vid inmatning.

Exempel:

```
PROCEDURE    LAYOUT ;
BEGIN
                  .
                  .
                  NO DATA (ODDFELLOW)
END ;
                  .
                  .
                  .
   ODDFELLOW :
BEGIN
COMMENT      HÄR ÄR HELA DATAMÄNGDEN INLÄST ;
                  .
                  .
                  .
                  .
```

Om proceduren NO DATA inte anropas i ett program, och inmatning begärs även sedan data tagit slut på inmatningsmediet, kommer hopp att ske till ett tänkt läge omedelbart före programmets sista END. Utförandet av programmet kommer då alltså att avslutas.

Det bör observeras att de procedurer och lägen, som tillhandahålls som aktuella parametrar till de genomgångna beskrivande procedurerna måste vara deklarerade utanför layoutproceduren. Brott mot denna regel kan orsaka svårfångade urspårningar.

4.6 Exempel på layoutprocedurer

Nedan ges två exempel på utseende av layoutprocedurer, samt kommenteras de båda exemplen.

Exempel 1:

```
PROCEDURE LAYOUT ;
BEGIN
          FORMAT ( /^) ;
          IF B1 THEN
BEGIN
          FORMAT 1 (^XB^, Y + 10) ;
          NO DATA ( L32 )
END ;
          H LIM (IF B1 THEN 1 ELSE 10, 30)
END ;
```

I denna layoutprocedur refereras till tre i programmet globala storheter B1, Y och L32. Om Y har värdet 3, kommer ett anrop av proceduren att innebära:

Procedur	Om B1 är sann	Om B1 är falsk
FORMAT	13B	/
H LIM	(1, 30)	(10, 30)
V LIM	(1,∞)	(1,∞)
TABULATION	1	1
NO DATA	L32	end program

Man noterar här att i fallet variabeln B1 är sann kommer formatet 13B att överskrivas det tidigare satta formatet /. Varje nytt anrop av en beskrivande procedur förstör tidigare anrop, som rör samma beskrivning.

Exempel 2:

```
PROCEDURE LAYOUT ;
BEGIN
          FORMAT 1 (^↑, (X (BB-4Z.DD), //)^ , N) ;
          H LIM (11, 110) ;
          H END (K, L, L)
END ;
PROCEDURE K ;
          H LIM (11, 110) ;
PROCEDURE L ;
          H LIM (16, 105) ;
          .
          .
          .
          .
```

Dessa procedurer får till följd att de horisontella marginalerna (16, 105) sättes närhelst någon form av horisontellt spill inträffar. Normal begäran om ny rad, med ett snedstreck i formatet, kommer att innebära marginalerna (11, 110).

Kap. 5. LISTPROCEDURER

Begreppet listprocedur är inte någon ny företeelse. Det existerar redan i ALGOL 60, men har aktualiserats i samband med tillkomsten av Knuth's rapport. Begreppet kan med fördel användas för annat än vad denna skrift rör, d.v.s. in/utmatning.

5.1 Listbegreppet samt definition av listprocedur

För klargörande av begreppet ges först följande exempel:

```
PROCEDURE   LISTA (BEHANDLA) ;
PROCEDURE   BEHANDLA ;
BEGIN
            BEHANDLA (A) ;
            BEHANDLA (B) ;
            BEHANDLA (C)
END
```

Proceduren LISTA definierar här att något skall utföras på listan med elementen A, B och C.

```
PROCEDURE   LISTA (BEHANDLA) ;
PROCEDURE   BEHANDLA ;
BEGIN
INTEGER     I ;
            FOR I := 1 STEP 1 UNTIL 14 DO
            BEHANDLA (A[I])
END
```

Här består listan av de 14 elementen A[1], A[2], ... , A[14] .

Utmärkande för bearbetningen är här att en och samma process skall utföras på vart och ett av listans element. Man kan nu tänka sig proceduren LISTA anropad med den aktuella parametern (proceduren) ARBETA. Denna skall då i sin kropp innehålla koden för den algoritm, som definierar det arbete, som skall utföras i det aktuella fallet. Man skriver alltså anropet

```
LISTA (ARBETA) ;
```

och får därmed den aktuella parametern ARBETA insatt på samtliga ställen där den formella parametern BEHANDLA uppträder. Därefter genomlöps proceduren LISTA, varvid det aktuella arbetet utförs.

Följande benämningar införs: Proceduren LISTA kallas listprocedur, proceduren BEHANDLA elementprocedur samt A, B och C resp. A [I] element .

Definition:

En listprocedur utgöres av en procedur, vars parameter är en elementprocedur. Noll, ett eller flera anrop av denna elementprocedur skall uppträda i listprocedurens kropp. Elementprocedurens aktuella parameter vid vart och ett av dessa anrop utgör ett element i den av listproceduren definierade listan. Ordningsföljden mellan listans element överensstämmer med ordningsföljden mellan parametrarna i elementproceduranropen i listprocedurens kropp.

Handnot (innebärande en not, som inte nödvändigtvis behöver uppträda längst ner på papperet):

Det är i och för sig inte nödvändigt att föreskriva något om antalet procedurparametrar ovan. Listproceduren eller elementproceduren må teoretiskt ha flera parametrar. I Knuth's förslag saknar emellertid en dylik generalisering intresse, varför även i fortsättningen endast en-parameterfallet kommer att beaktas.

Konvention:

I ALGOL 60 behöver endast specificering av värdeanropade procedurparametrar ske. Av pedagogiska skäl specificeras emellertid här ständigt samtliga parametrar.

Som ett mera fullständigt exempel för att klargöra funktionssättet hos en listprocedur betraktas här en situation, då en summering skall utföras. Varje delsummering kan då äga rum i elementproceduren:

```
PROCEDURE   ADDTERM (X) ;
REAL        X ;
            S := S + X ;
```

Summeringen kan nu definieras i en funktionsprocedur:

```
REAL PROCEDURE   SUM (LIST) ;
PROCEDURE        LIST ;
BEGIN
REAL             S ;
PROCEDURE        ADDTERM (X) ;
REAL             X ;
                 S := S + X ;
COMMENT          HÄR STARTAR PROCEDURKROPPEN FÖR SUM ;
                 S := O ;
                 LIST (ADDTERM) ;
                 SUM := S
END
```

I huvudprogrammet uppträder nu t.ex. satsen:

```
Y := SUM (LISTA) ;
```

där LISTA antages vara den överst i detta avsnitt deklarerade proceduren. Vid det aktuella anropet av SUM utbyts i dess procedurkropp den formella parametern LIST

mot den aktuella LISTA. Därefter genomlöps SUM, varvid satsen LIST (ADDTERM) ; innebär anrop av proceduren LISTA. I denna procedur utbyts i sin tur den formella parametern BEHANDLA mot den aktuella ADDTERM. Detta innebär att då LISTA genomlöps, utförs anrop av ADDTERM för i tur och ordning elementen A, B och C. Detta medför utförande av satserna:

```
S := S + A ;
S := S + B ;
S := S + C ;
```

en för varje anrop. Därefter är genomlöpningen av LISTA klar, och återhopp sker till procedurkroppen för SUM, där funktionsvärdet sättes lika med S, som nu erhållit värdet A + B + C. Till sist erhåller så Y detta värde.

Handnot: Det gives i ALGOL 60 enklare sätt att summera tre tal.

I ett användarprogram krävs ej procedurhantering av ovanstående vidlyftighetsgrad. För möjlighet till användning av systemprocedurerna INLIST/OUTLIST (beskrivna nedan) krävs endast deklaration av en listprocedur i överensstämmelse med de två procedurerna LISTA överst i detta avsnitt. Elementen A, B och C resp. A[I], I = 1, 2, ..., 14 utgör då de element, som skall läsas/skrivas. Det måste emellertid noteras att den av användaren tillhandahållna listprocedurens parameter (som alltså i sig är en procedur) endast får ha en parameter. Skälet till detta kommenteras i samband med beskrivning av INLIST/OUTLIST.

5.2 Procedurerna INLIST och OUTLIST

De mest generella överföringsprocedurerna i Knuth's förslag har utseendet:

```
INLIST (NR, LAYOUT, LISTA)
OUTLIST (NR, LAYOUT, LISTA)
```

Parametrarna har följande innebörd:

Formell parameter	Aktuell parameter
NR	Ett aritmetiskt uttryck, som utvärderat anger koden för den för överföringen aktuella yttre enheten.
LAYOUT	Identifieraren för en av användaren tillhandahållen layoutprocedur, där format etc. för de överförda begreppen anges.
LISTA	Identifieraren för en av användaren tillhandahållen listprocedur, vars element utgör det som skall överföras.

När en överföring med INLIST/OUTLIST utförs, anropas först layoutproceduren och därefter listproceduren.

Exempel 1: INLIST (NR, MAKEUP, ÖGON) ;

Denna procedursats får till följd att elementen specificerade i listproceduren ÖGON

kommer att läsas från yttre enhet med koden NR i överensstämmelse med format, som angivits i layoutproceduren MAKEUP.

Exempel 2:

```
BEGIN
PROCEDURE P ;
          FORMAT ( ´3D.D, BZZD, 2B4S /`) ;
PROCEDURE UT (A) ;
PROCEDURE A ;
BEGIN
          A (TOTAL) ;
          A (HELTAL) ;
          A (´SVAR`)
END ;
          .
          .
          .
          OUTLIST (NR, P, UT) ;
          .
          .
          .
END
```

Detta program medför utmatning av variablerna TOTAL och HELTAL samt strängen ´SVAR` på yttre enhet motsvarande koden NR.

Arbetssättet vid själva överföringen har ännu inte diskuterats. Det kan synas besynnerligt att i exemplet ovan någon utmatning verkligen kommer att ske, eftersom elementproceduren A av användaren lämnas obeaktad. Någon aktuell procedurparameter tillhandahålls ju ej, att substitueras för A vid utförandet. Detta förhållande kan förklaras på följande sätt:

Det antas i ALGOL-kompilatorn existera två för användaren oåtkomliga 'atomära' procedurer INITEM och OUTITEM, som kommer till användning i samband med anrop av INLIST och OUTLIST. I och med anropet

OUTLIST (NR, P, UT) ;

kommer att som aktuell parameter till proceduren UT, alltså motsvarande den formella parametern A, att insättas OUTITEM. Det är denna procedur, som utför själva överföringsarbetet. Denna substitution (A ersätts med OUTITEM) utförs automatiskt i och med anropet av OUTLIST, och ligger utanför användarens kontroll. Häri ligger nu skälet till att den av användaren deklarerade listproceduren (UT i exemplet ovan) måste ha en och endast en parameter. För vidare diskussion kring detta förlopp hänvisas till ursprungsförslaget. Förståelsen underlättas emellertid av ett detaljerat studium av det i avsnitt 5.1 beskrivna exemplet med proceduren SUM. Man bör där kunna få en känsla för hur kontrollväxlingen verkligen utförs.

Man bör notera att varje nytt anrop av INLIST/OUTLIST återställer alla interna beskrivande parametrar (de beryktade 'dolda variablerna') till sina normalvärden.

Exempel 3:

```
BEGIN
PROCEDURE L ;
BEGIN
            H LIM (1, 80) ;
            H END (EXIT, EXIT, EXIT)
END ;
PROCEDURE MÅNGA (ATOM) ;
PROCEDURE ATOM ;
            ATOM ( A[N + 1]) ;
PROCEDURE EXIT ;
            GOTO LÄGE ;
            N := 0 ;
      BACK: INLIST (NR, L, MÅNGA) ;
            N := N + 1 ;
            GOTO BACK ;
      LÄGE:
            .
            .
            .
END
```

I detta exempel, där deklaration av fältet A samt heltalsvariabeln N utelämnats, utförs inläsning av data i standardformat från t.ex. kortläsaren (som då förmodas motsvara koden NR). Ett enda hålkort läses, och vid läget LÄGE i programmet har antalet inlästa tal placerats i variabeln N. Talen är då inlästa till fältelementen A[1], ... , A[N].

Som ytterligare exempel rekommenderas på det här stadiet en genomgång av det första programexempel, som anges i denna skrift. Här åsyftas alltså exemplet med utmatning av de naturliga talen 1, 2, 3, ..., 500.

Det bör påpekas att givetvis ingenting föreskriver att endast en layoutprocedur kan deklareras i ett program. För styrning av olika layouter är det mycket lämpligt att deklarera flera, kanske för användning i samband med samma lista av element vid olika tillfällen.

Det kan noteras att det faktiskt är tillåtet att vid användning av INLIST/OUTLIST även i listproceduren inkludera anrop av beskrivande procedurer. Sålunda kan man t.ex. ändra format för ett visst tal beroende på värdet av ett tidigare inläst tal. Som en konsekvens av detta förhållande skulle man kunna draga slutsatsen att layoutprocedurer över huvud taget är onödiga. Något sådant kan vara sant, men detaljer kan vara beroende av aktuell implementering. Användning av beskrivande procedurer i listprocedurer får dock förmodas inträffa mera i undantagsfall.

Kap. 6. ÖVRIGA PROCEDURER

6.1 Proceduren SYSPARAM

I Knuth's förslag existerar ett antal parametrar, vilka kan betraktas som 'system-parametrar' och vilka inte är direkt åtkomliga för användaren. Dessa parametrar är:

1. Parametrarna P och Q, innebärande fysiska gränser för olika yttre enheter.

2. Antalet blanka tecken, betecknat K, som fungerar som avskiljare mellan tal i standardformat.

3. Löpande värden för positionsvisaren, betecknad POSV, samt radvisaren, betecknad RADV.

Med användning av proceduren

SYSPARAM (NR, F, KV)

kan tillgång till dessa parametrars värden erhållas. Dessa värden kan dessutom ändras. Aktuell parameter till NR har samma betydelse som tidigare, alltså kod för aktuell yttre enhet. De båda övriga parametrarna lyder följande samband:

Värde på F	Innebörd
1	KV := POSV ;
2	POSV := KV ;
3	KV := RADV ;
4	RADV := KV ;
5	KV := P ;
6	P := KV ;
7	KV := Q ;
8	Q := KV ;
9	KV := K ;
10	K := KV ;

Man lägger här märke till att anrop av SYSPARAM med F lika med 1, 3, 5, 7 eller 9 medför att en upplysning ges åt programmet. I dessa fall skall alltså aktuell motsvarighet till KV vara en variabel, vars i och med proceduranropet tilldelade värde senare på något sätt skall användas i programmet. (Det är uppenbart att KV i procedurkoden för SYSPARAM är en namnanropad formell parameter.)

Exempel: SYSPARAM (NR, 1, PLATS) ;

Detta anrop medför att den av användaren deklarerade variabeln PLATS tilldelas det värde positionsvisaren för ögonblicket (vid anropet) har, t. ex. den position på en tänkt utmatad rad dit utmatningen just hunnit.

I övriga fall, d.v.s. då F har värdena 2, 4, 6, 8 eller 10, kommer resp. parameter POSV, RADV, P, Q eller K att tilldelas värdet av uttrycket (avrundat till närmaste heltal), som uppträder som tredje procedurparameter.

Exempel: SYSPARAM (NR, 4, 51);

Här kommer radvisaren att förflyttas till rad 51, d.v.s. utmatning kommer att fortsätta på denna rad, oberoende av var radvisaren tidigare befann sig. Observera att begäran om förflyttning bakåt på raden vid F = 2 kommer att innebära att först ny rad utförs. På motsvarande sätt erhålles först ny sida om begäran om förflyttning uppåt sker med F = 4.

I fallen F = 6 resp. F = 8 kommer fysiska gränser att ändras när så är möjligt. Detta kan vara fallet rörande blocklängd på magnetband, radskrivare med flera radbredder, eller när en minskning från standardvärden begärs för andra yttre medier. I det fall fysikaliskt omöjliga ting begärs kommer proceduranropet att betraktas som en tom sats, och ingen åtgärd kommer alltså att vidtagas.

6.2 In/utmatning av grundsymboler

För att associera grundsymboler i ALGOL 60 med särskilda heltal finns till förfogande procedurerna

INSYMBOL (NR, S, MÅL)
OUTSYMBOL (NR, S, KÄLLA)

Dessa procedurer möjliggör arbete med grundsymboler på sätt som nedan beskrivs.

6.2.1 Proceduren INSYMBOL

För parametrarna till INSYMBOL gäller:

Formell parameter	Aktuell parameter
NR	Kod för aktuellt inmatningsmedium.
S	Sträng, uppbyggd av ett antal grundsymboler.
MÅL	Heltalsvariabel, deklarerad i användarens program.

Ett anrop av INSYMBOL leder till följande:

1. Den grundsymbol, som står närmast i tur på inmatningsmediet, läses in.
2. Om denna grundsymbol återfinnes i strängen som andra aktuell parameter, tilldelas aktuell heltalsvariabel till MÅL det värde, som innebär grundsymbolens numrerade placering i strängen. Härvid räknas numreringen från vänster till höger med början med ett.

Exempel:

INSYMBOL (NR, ⌜ABC IF 7K⌝, AD) ;

Om vid detta anrop den inlästa grundsymbolen är 7, kommer variabeln AD att tilldelas värdet 5, eftersom siffran 7 uppträder som femte grundsymbol i strängen, från vänster.

3. Om den inlästa grundsymbolen inte återfinns i strängen, kommer värdet 0 (noll) att tilldelas aktuell motsvarighet till variabeln MÅL.

4. Om den inmatade symbolen inte är en grundsymbol i ALGOL 60:s mening, kommer ett korresponderande heltalsvärde med negativt tecken att tilldelas aktuell motsvarighet till variabeln MÅL. Denna korresponderande kod (med negativa tal) är ej fixerad i Knuth's rapport, utan lämnas för bestämning till implementören.

6.2.2 Proceduren OUTSYMBOL

För de två första aktuella parametrarna till OUTSYMBOL gäller samma innebörd som för INSYMBOL, medan den tredje aktuella parametern till KÄLLA skall vara ett aritmetiskt uttryck.

Ett anrop av OUTSYMBOL innebär:

1. Det aritmetiska uttryck som står som aktuell tredje parameter, alltså i stället för den formella parametern KÄLLA, evalueras och avrundas till närmaste heltalsvärde.

2. Om strängens (den andra parameterns) antal grundsymboler är större än eller lika med detta heltalsvärde, kommer motsvarande grundsymbol att matas ut på det yttre mediet. Härvid numreras strängens grundsymboler från vänster till höger.
   Exempel:

   OUTSYMBOL (NR, ⌜P BEGIN QAC⌝, 5 + I) ;

   Om variabeln I vid detta anrop har värdet -1, kommer den fjärde grundsymbolen att matas ut, alltså bokstaven A.

3. Lämplig åtgärd i det fall den tredje aktuella parameterns närmaste heltalsvärde är 0 (noll) eller större än antalet grundsymboler i strängen överlåts i Knuth's förslag åt implementören att bestämma. Vid negativa värden skall emellertid samma korresponderande kod användas som vid motsvarande fall för INSYMBOL.

6.3 Mellanlagring av data

För att kunna mellanlagra data på ett lämpligt yttre medium finns till förfogande proceduren

PUT (N, LISTA)

Här står aktuellt värde på N för en heltalsparameter och aktuell motsvarighet till

LISTA skall vara identifieraren för en av användaren deklarerad listprocedur. Ett anrop av proceduren medför att de värden som beskrivs i listproceduren lagras tillsammans med det aktuella identifikationstalet N. Om något tidigare mellanlagrats med samma identifikationstal, förstörs det i och med anropet. Proceduranropet förändrar icke värden hos i listan förekommande variabler.

Med proceduren PUT mellanlagrade data kan återhämtas med proceduren

GET (N, LISTA)

Parametrarna har här samma betydelse som ovan. Listprocedurens element måste emellertid i detta fall vara variabler. Ordningsföljden mellan data är densamma vid återläsning som vid lagring med PUT. Dessutom skall resp. typer överensstämma (såvitt ej särskilda konverteringsrutiner inlagts). Om läsning sker till ett mindre antal element än som skrivits med ett anrop, kommer endast det vid läsningen begärda antalet att överföras. I det motsatta fallet anses situationen odefinierad.

Ett anrop av GET förändrar ej data på det yttre mediet.

Exempel:

```
BEGIN
REAL        A, B, C, K, L, M;
PROCEDURE   L1 (PRO) ;
PROCEDURE   PRO ;
BEGIN
            PRO (A) ;
            PRO (B) ;
            PRO (C)
END ;
PROCEDURE   L2 (JOBBA) ;
PROCEDURE   JOBBA ;
BEGIN
            JOBBA (K) ;
            JOBBA (L) ;
            JOBBA (M) ;

END ;
            .
            .
            .
            A := 3.6 ;
            B := 17.5 ;
            C := 269.153 ;
            PUT (3, L1) ;
            .
            .
            GET (3, L2) ;
            .
            .
END
```

I och med anropet av GET ges K värdet 3.6, L värdet 17.5 och M värdet 269.153.

Det överlämnas åt implementören att fixera detaljer kring dessa procedurer, exempelvis metod för val av lämpligt yttre medium för lagringen o.d.

6.4 Procedurerna STRINGELEMENT samt LENGTH

För att möjliggöra avkänning (scanning) av en given formell eller aktuell sträng på ett maskinoberoende sätt har införts proceduren:

STRINGELEMENT (S1, I, S2, J)

Här är aktuella motsvarigheter till parametrarna S1 och S2 strängar, medan aktuella motsvarigheter till I och J skall vara heltalsvariabler. Ett anrop av proceduren medför:

1. Aktuellt värde på variabeln I bestämmer vilken symbol i strängen S1 som avses. Om I = 1 är det symbolen längst till vänster, om I = 2 symbolen näst längst till vänster o.s.v.

2. Denna symbols plats i aktuell motsvarighet till strängen S2 (den skall alltså finnas där) bestämmer det värde som tillordnas variabeln J. Härvid räknas placering även från vänster till höger. Jfr f.ö. proceduren INSYMBOL.

Det är uppenbart att STRINGELEMENT har sin största betydelse när den ena av de aktuella parametrarna till S1 och S2 är en formell parameter i en användarprocedur. Med båda strängarna aktuella föreligger knappast lika stor användbarhet.

Exempel:

```
PROCEDURE    KÄNN (STR, VAR);
STRING       STR;
INTEGER      VAR;
             STRINGELEMENT (STR,3, ⌜ABCDEFGH⌝,
                                   VAR);
```

Anropet KÄNN (⌜GAFDE⌝, PL); medför att den tredje symbolen i strängen ⌜GAFDE⌝, d.v.s. bokstaven F, via sin placering i strängen ⌜ABCDEFGH⌝ medför att variabeln PL tilldelas värdet 6.

Det föreligger möjlighet att få reda på av hur många grundsymboler en given sträng är uppbyggd. Detta utförs med ett anrop av proceduren

LENGTH (S)

Denna procedur är en heltalsfunktion (d.v.s. deklarerad som INTEGER PROCEDURE) och ett anrop medför att funktionen ges värdet av antalet grundsymboler i aktuell motsvarighet till strängen S. Härvid räknas ej det yttersta paret strängparenteser.

Exempel: B := LENGTH ('732B5AC DO 6') ;

Denna sats tillordnar variabeln B värdet 9. Proceduren LENGTH har sin största användbarhet då strängen S är en formell parameter i en användarprocedur. I exemplet ovan kunde ju lika gärna från början antalet grundsymboler räknats för hand.

6.5 Procedurerna NAME och TYPE

Det förekommer fall då det kan vara önskvärt att 'komma ihåg' en proceduridentifierare eller en lägesidentifierare. I ALGOL 60 tillhandahålls en möjlighet här med hjälp av den aktuella-formella motsvarigheten vid namnanropade parametrar. För att utvidga denna möjlighet till variabler såväl som parametrar introduceras proceduren:

NAME (F, V, L, P)

För parametrarna gäller:

Formell parameter	Aktuell parameter
F	Aritmetiskt uttryck.
V	Heltalsvariabel.
L	Lägesidentifierare.
P	Proceduridentifierare.

Ett anrop av denna procedur medför:

Värde på uttrycket F	Aktion
1	Ett heltalsvärde motsvarande läget L tillordnas variabeln V.
2	Hopp till det läge, som motsvaras av aktuellt värdet på variabeln V. Denna måste ha erhållit detta värde i ett tidigare anrop av NAME. Om aktuell variabel till V har värdet 0 (noll) kommer här hopp att ske till ett tänkt läge omedelbart före programmets sista END. Aktionen innebär alltså att i detta fall utförandet av programmet avslutas.
3	Ett heltalsvärde motsvarande aktuell motsvarighet till proceduren P tillordnas aktuell variabel V.
4	Anropet av NAME kommer att innebära anrop av av den procedur (som skall vara utan formella parametrar), vars identifierare i ett tidigare NAME-anrop erhållit värdet motsvarande aktuellt värde på variabeln V. Om aktuellt V här har värdet 0 (noll) kommer proceduren att vara PROCEDURE DUMMY ; ;

Det bör noteras att tillordningen av heltalsvärden till läges- eller proceduridentifierare enbart gäller inom det block i programmet där resp. identifierare deklarerats. Reglerna för blockstrukturgiltighet i ALGOL 60 följs alltså.

Vid utmatning av tal i standardformat har angivits formen 'heltal' eller 'decimalbråk med exponent', beroende på om det utmatade talet deklarerats som INTEGER eller REAL. För att vid programmering av utmatningen kunna få fram önskvärd layout för båda dessa fall tillhandahålls proceduren

TYPE (FL, VA)

Ett anrop av proceduren, där aktuell motsvarighet till FL är en heltalsvariabel och aktuell motsvarighet till VA en variabel (eller en sträng), innebär:

Om VA är av typen	ges FL värdet
INTEGER	1
REAL	2
(BOOLEAN	3)
(STRING	4)

De två sista är placerade inom parentes eftersom här Knuth's förslag endast rekommenderar ovan nämnd aktion, detta p.g.a. att en dylik distinktion inte är möjlig i vissa implementeringar av ALGOL 60.

APPENDIX 1

Några kommentarer rörande Control Data:s implementation av förslaget.

1. För parametern NR (avseende aktuell yttre enhet) gäller följande standardvärden:

	NR
Inmatning via kortläsare	60
Utmatning via radskrivare	61

2. Såsom insättning i talformat får ej förekomma en sträng, omedelbart föregången av en replikator.

3. Vid utmatning med standardformat kommer en asterisk att uppträda på båda sidor om talet, om
   a) formatet är angivet utan förtecken och det utmatade talet är negativt.
   b) det utmatade talet är för stort.

4. Normalt standardformat för utmatning innebär:
   a) för heltal formatet +15ZD
   b) för reella tal formatet $+D.9D_{10}+3D$

5. Ett ALGOL-ord (ex. BEGIN) betraktas ej som en enda grundsymbol vid in/utmatning av strängar.

6. Formatkoden M tilläggs för ickeformat, innebärande att ett tal överförs precis som det ser ut externt eller internt, oberoende av typ.

7. Antalet blanka tecken, betecknat K, fungerande som avgränsare mellan tal, har som standard värdet 2.

8. Beteckningen FORMAT ersätter generellt FORMAT N. Man skriver alltså t.ex.

   FORMAT (STRÄNG, X1, X2) i st.f. FORMAT 2 (STRÄNG, X1, X2)

   Antalet variabler X1, X2, ... är ökat från 0 - 9 till 0 - 30.

9. Av systemtekniska skäl ökas värdena på parametrarna V och H med en enhet när proceduren H LIM genomlöps.

10. Proceduridentifierarna INSYMBOL/OUTSYMBOL ersätts med INCHARACTER/OUTCHARACTER, vid vilkas användning gäller att ett ALGOL-ord ej betraktas som en enhet.

11. Proceduren OUTREAL utmatar tal i standardformat.

12. Beteckningarna INPUT N/OUTPUT N ersätts generellt av INPUT/OUTPUT. Man skriver t.ex.:

OUTPUT (NR, FORMATSTRÄNG, X1) i st.f. OUTPUT 1 (NR, FORMATSTRÄNG,X1)

Antalet variabler X1, X2, ... är ökat från 0 - 9 till 0 - 61.

13. Procedurerna GET och PUT är ej implementerade. Följande procedurer existerar:

    GET ARRAY (NR, MÅL)
    PUT ARRAY (NR, KÄLLA)

    Hela det fält, som är aktuell motsvarighet till MÅL/KÄLLA, läses/skrives radvis vid ett anrop. Detta sker helt formatfritt, till yttre enhet motsvarande beteckningen NR. På styrkortet CHANNEL skall bokstaven A medtagas.

14. Proceduren LENGTH (S) ersätts med CHLENGTH (S), vars funktionsvärde vid ett anrop blir antalet karaktärer (ej grundsymboler) i den aktuella motsvarigheten till strängen S.

15. För proceduren STRINGELEMENT räknas karaktärer i st. f. grundsymboler.

16. Proceduren NAME är ej implementerad.

17. Proceduren TYPE är ej implementerad.

APPENDIX 2

Några kommentarer rörande General Electric:s implementation av förslaget:

1. För parametern NR (avseende aktuell yttre enhet) gäller följande standardvärden:

	NR
Inmatning vid kortläsare	5
Utmatning via radskrivare	6

2. Proceduren BAD DATA (P), där P är specificerad som procedur, är tillagd för att giva möjlighet vidta åtgärd (inom den aktuella motsvarigheten till P) när ett tal försöker inmatas och referensfältet ej är i överensstämmelse med talet.

3. I talformat kan användas bokstaven O (som i Olof) för överföring av ett oktalt tal. Bokstaven får föregås av replikator.

4. Standardformat för utmatning innebär ständigt formatet $-.16D_{10}+DD$.

5. I ickeformat kan användas formatkoden E för tal som deklarerats som EXTENDED REAL , innebärande dubbel precision.

6. Procedurerna INREAL/OUTREAL samt INARRAY/OUTARRAY är ej implementerade.

7. Proceduren NO DATA har som parameter en procedur och ej ett läge. Den aktuella proceduren kommer att genomlöpas i st.f. hopp till nämnda läge.

8. Proceduren POSITION (NR, A, B) kompletterar proceduren SYSPARAM. Aktuellt värde på A innebär sidnummer och B radnummer. Ett anrop medför förflyttning till avsedd fysisk plats.

9. Procedurerna PUT och GET är ej implementerade.

APPENDIX 3

Några kommentarer rörande IBM:s implementation av förslaget:
För maskinsystemet IBM 360 har det ansetts tillräckligt att implementera en begränsad mängd av de beskrivna procedurerna. Denna mängd kan i vissa sammanhang anses alltför begränsad. Därför har på ett antal olika håll försiggått tilläggsprogrammering, så att nu (1968) bl a inom Sverige en mera fullständig procedurfamilj finns tillgänglig.
Nedan angives först de av IBM som standard tillhandahållna faciliteterna:

1. För parametern NR (avseende aktuell yttre enhet) gäller följande standardvärden:

	NR
Inmatning via kortläsare	0
Utmatning via radskrivare	1

För övrigt skall NR alltid ha något av värdena 0, 1, . . . , 15.

2. Följande procedurer existerar:

För inmatning:	INSYMBOL, INREAL, ININTEGER, INBOOLEAN, INARRAY, INTARRAY, INBARRAY.
För utmatning:	OUTSYMBOL, OUTREAL, OUTINTEGER, OUTBOOLEAN, OUTSTRING, OUTARRAY, OUTTARRAY, OUTBARRAY.
För styrning:	SYSACT
För mellanlagring:	PUT, GET

2.1 Procedurerna INSYMBOL och OUTSYMBOL överensstämmer helt med Knuth´s förslag så när som att programavbrott inträffar då ett inmatat tecken ej finns i strängen, eller heltalet vid utmatning ej pekar på något tecken i strängen.

2.2 Procedurerna INREAL och OUTREAL överensstämmer nära med Knuth´s förslag.
Märk dock: OUTREAL konverterar utmatat värde till standardformat. För IBM 360 gives två möjliga standardformat, kort eller långt. Val mellan dessa sker med parameter i EXEC-styrkortet.

kort standardformat	$+D.6D_{10}{}^{+}DD$
långt standardformat	$+D.15D_{10}{}^{+}DD$

Talet noll utmatas dock som ␣0 följt av 11 resp 20 blanka tecken.

2.3 Procedurerna ININTEGER och OUTINTEGER har tillkommit för heltalsbehandling. Uppbyggnaden är helt lik INREAL/OUTREAL. Konvertering sker med OUTINTEGER så att formatet +9ZD kommer till användning.

2.4 Procedurerna INBOOLEAN och OUTBOOLEAN överensstämmer även i stort med INREAL/OUTREAL. Överförda värden skall dock vara av boolesk typ. Värdet hos med OUTBOOLEAN överfört logiskt uttryck

utmatas som 'TRUE'␣ eller 'FALSE'.

2.5 Proceduren OUTSTRING (NR, S) utmatar alla i strängen S förekommande symboler (utom inledande och avslutande strängparentes).

2.6 Procedurerna INARRAY/OUTARRAY, INTARRAY/OUTTARRAY samt INBARRAY/OUTBARRAY överensstämmer i stort med Knuth's förslag INARRAY/OUTARRAY. Dock gäller för IBM 360:

INARRAY/OUTARRAY	överför fält av reell typ,
INTARRAY/OUTTARRAY	överför fält av heltalstyp,
INBARRAY/OUTBARRAY	överför fält av boolesk typ.

2.7 Proceduren SYSACT är motsvarighet till SYSPARAM, men med utökat arbetssätt. Vid anropet SYSACT (NR, F, KV) gäller full överensstämmelse med Knuth's förslag för F-värdena 1 t o m 10. Härutöver är emellertid av IBM tillagt 5 möjliga F-värden:

Värde på F	Innebörd
11	KV: = C;
12	C: = KV;
13	KV: = RADV samt intern aktion
14	Radförflyttning
15	Sidförflyttning

2.7.1 Vid F-värdet 11 gäller:
Aktuell motsvarighet till KV skall vara en variabel. Denna ges värdet 1 om aktuell datafil är öppen, värdet 0 om aktuell datafil är stängd, samt värdet -1 om aktuell datafil är uttömd. Detta sistnämnda faktum medger alltså möjlighet avgöra om vid inläsning inga fler data finns tillgängliga för inmatning, alltså en motsvarighet till NO DATA, vilken procedur ej existerar hos IBM.

2.7.2 Vid F-värdet 12 gäller:
Ett anrop av SYSACT är definierat endast om KV har värdet 1 eller 0. KV lika med 1 innebär att aktuell datafil öppnas (förutsatt att den dittills varit stängd), samt KV lika med 0 innebär att aktuell datafil stänges (förutsatt att den dittills varit öppen). För andra utgångssituationer med - för ett anrop ingen åtgärd. Beträffande öppna och stängda datafiler gäller:
En datafil öppnas genom endera av:

1. Anrop av SYSACT med F=12 och KV=1.
2. Anrop av valfri inmatningsprocedur.

En datafil stänges genom endera av:

1. Anrop av SYSACT med F=12 och KV=0
2. Vid slutet av aktuellt Algol-program.

2.7.3 Vid F-värdet 13 gäller:
Ett anrop av SYSACT förutsätter att aktuell datafil är öppen samt att aktuell motsvarighet till KV är en variabel. Proceduranropet medför två aktioner:

a) Samma som ett anrop med F=3 d v s aktuell motsvarighet till KV

ges det aktuella värdet hos RADV.

b) Aktuellt värde hos RADV lagras undan i ett internt index, som sedan kan användas vid ett SYSACT-anrop med F=4. I situationen bakåtförflyttning (t ex vid magnetband) med F=4 måste aktuell motsvarighet till RADV ha placerats i detta interna index vid ett SYSACT-anrop med F=13. Vid framåtförflyttning är detta ej nödvändigt men tidsbesparande.

2.7.4 Vid F-värdet 14 gäller:
Aktuell motsvarighet till KV måste vara större än 0. Ett SYSACT-anrop kommer att medföra att RADV ökas med värdet hos den aktuella motsvarigheten till KV. Förbipasserade rader kommer därvid att överhoppas eller fyllas med blanka tecken, beroende på om in- eller utmatning senast har utförts.
Anropet SYSACT (NR, 14, KV); motsvarar alltså genomlöpande av programmet:

```
BEGIN
INTEGER     V;
            SYSACT (NR, 3, V);
            SYSACT (NR, 4, V + KV)
END
```

Begäran om radframmatning över sidslut kommer ej att effektueras, utan enbart till första raden på nästföljande sida, även om detta innebär framматning ett mindre antal rader än vad som anges av aktuell motsvarighet till KV.

2.7.5 Vid F-värdet 15 gäller:
Ett anrop av SYSACT medför i tur och ordning:

a) Förflyttning till ny sida.

b) RADV: = aktuellt värde hos KV;

c) POSV: = 1;

d) Utfyllning av överhoppade positioner med blanka tecken (om vid utmatning).

Om ingen siduppdelning föreligger kommer anropet

```
SYSACT (NR, 15, KV) ;
```

att vara helt ekvivalent med anropet

```
SYSACT (NR, 14, KV);
```

2.8 Procedurerna PUT och GET arbetar helt i överensstämmelse med Knuth's förslag, så när som att automatisk konversion INTEGER/REAL ej sker.

3 I samarbete med IBM har under 1967 vid Institutionen för Informationsbehandling, KTH/SU, utarbetats ytterligare procedurer. I förhållande till Knuth's förslag gäller för dessa:

3.1 Procedurerna måste vid användning deklareras som kodprocedurer i programmet.

3.2 Procedurerna INPUT N, NO DATA, SYSPARAM, TYPE och STRING-ELEMENT existerar icke.

3.3 FORMAT N existerar endast för N = 0, 1, 2, 3, 4, 5.

3.4 Identifieraren OUTPUT N har ändrats till UTPUT N.

3.5 Vid alfaformat och strängformat motsvarar varje A eller S en karaktär.

3.6 Vid alfaformat och ickeformat måste listelement och formatnotis överensstämma till typ och sort.

3.7 Som insättning får ej förekomma en replikator omedelbart före en sträng.

3.8 Med UTPUT N utmatade storheter måste vara aritmetiska uttryck.

REFERENSER

A Proposal for Input-Output Conventions in ALGOL 60.
Communications of the ACM, maj 1964.

ISO Draft Proposal on the Algorithmic Language ALGOL,
Proposal for Input-Output Procedures for ALGOL 60 (ACM), mars 1965.

Tomas Ohlin Knuth's förslag till in/utmatning i ALGOL

I programmeringsspråket ALGOL existerar ursprungligen inga anvisningar om hur in/utmatning av storheter bör ske. År 1964 publicerade emellertid en av American Computing Machinery tillsatt kommitté ett förslag till utformning av ett antal procedurer för detta ändamål. Förslaget går numera under benämningen Knuth's förslag. Det har förverkligats av flera datamaskinleverantörer och har även framlagts för International Organization for Standardization för standardisering.

Denna bok beskriver förslaget och är avsedd dels för undervisning vid universitet och högskolor och dels som programmeringshjälp i andra sammanhang.

Läsaren förutsättes ha kunskaper i ALGOL.