UTMATNING PÅ PLOTTER.

LENNART HAGBJER.

RAPPORT RE-93, APRIL 1971

Utmatning på plotter.

Examensarbete i Reglerteknik.

Lennart Hagbjer.

Handledare: Johan Wieslander.

Copyright L.H./J.W.

Innehåll.

Introduction.

The task was to write a program for plotter output on a ink jet plotter built at Inst. för Elmät, LTH. This plotter puts dots of ink on a paper fastened to a rotating drum and it must have, for every x-coordinate in its turn, the y-coordinates in rising order where it shall put a dot. $0 \leq x,y < 1024$.

Indata exists in dump mode on DEC-tape, and consists of files of starting points followed by step increments. Each file has to be changed into pairs of xy-coordinates and then sorted and arranged in a suitable way for output.

The intermediate results are stored on disk. On account of the rather small core memory and the mass of coordinates, the sorting procedure is made in two steps. First a merge of several files sorted after x is made, then the y´s are sorted for each x-coordinate.

The program consists of six subroutines, linked by a program PLOTUT, written in FORTRAN.

CHANGE changes indata into several files of coord. sorted after the x-value.

UNITE merges those files into one.

YSORT sorts y-values and transforms the file in a way suitable for the plotter.

OUTPUT takes care of the actual output of data to the plotter.

MSORT does the actual sorting, called from CHANGE, YSORT.

INUT handles the transfer of data between disk and core memory.

The program is written to be run on a PDP-15 computer, and the subroutines are written in Macro-assembler.

Inledning.

Uppgiften avser att skriva ett datamaskinsprogram för utmatning på en vid Inst. för Elmät. konstruerad bläckstråleskrivare. Då denna sätter punkter på ett papper fastsatt på en roterande trumma skall utdata bestå av, för varje x-koordinat i tur och ordning, en stigande följd av de y-koordinater som skall utmärkas. $0 \leq x,y < 1024$.

Indata utgörs av följder av startpunkter samt serie av inkrement från startpunkten. Dessa indata skall alltså först omvandlas till xy-koordinater och sedan sorteras i rätt ordning för utmatning.

För lagring av mellanresultat används ett skivminne. På grund av begränsat utrymme i kärnmimmet sker sorteringen i två steg, först en sammansmältning av ett antal filer sorterade efter x, och sedan en sortering av y-värden för varje x-värde.

Programmet består av sex subrutiner, sammanhållna av ett huvudprogram, PLOTUT, skrivet i FORTRAN.

CHANGE	omvandlar indata till flera följder xy-koordinater
UNITE	sammansmälter följderna till en enda.
YSORT	sorterar y-värden och omformar för utmatning
OUTPUT	gör själva utmatningen på plottern
MSORT	rutin för sortering, används i CHANGE och YSORT.
INUT	sköter in- och utmatning till och från skivminnet.

Programmet är avsett att köras på en PDP-15 och subrutinerna är skrivna i PDP-assembler.

En kort beskrivning av plotterns arbetssätt.

Plotterns papper sitter fastspänt på en vals, som roterar 3000 varv/min. Längs denna vals rör sig ett organ som antingen sätter eller inte sätter en bläckprick i en viss position. Då papperet är uppdelat i 1024x1024 positioner och valsen snurrar ett varv för varje x-koordinat, sker alltså en fullständig utmatning på c:a 20 sekunder.

Valet prick - inte prick styrs av en räknare och ett register. Räknaren nollställs för varje varv av valsen. Det sker sedan ständig jämförelse mellan räknaren och registret, och när de har samma innehåll sätts en prick på papperet. Samtidigt hämtar registret ett nytt innehåll från datamaskinens kärnminne.

Till registret skall alltså ges succesivt växande y-koordinater för varje x-koordinat. Då plottern ej har någon räknare för x-värden bör dessutom sättas en prick i y 0 för varje x-koord. för att få korrekt framstegning i x-led.

För att täcka plotterns behov räcker det med 10 positioner i registret ($2^{10} = 1024$). Då datamaskinen arbetar med 18 pos. ord kan de sex första pos. användas för andra order. Ettor i pos. 5, 6 och 7 t. ex. medför en heldragen linje fram till och med det y-värde i vilket de finns från föregående y-värde. Detta används i YSORT.

Utmatningen från datamaskinen styrs av innehållet i de tre celler i kärnminnet med adresserna 34, 35 och 70 oktalt. Cell 34 innehåller negativa antalet ord som är kvar att lägga ut från blocket av celler i minnet och cell 35 innehåller adressen till cellen innan den som skall läggas ut.

Utmatning av data till en prick går till så att när begäran från plottern om nytt registerinnehåll kommer ökas innehållet i cell 34 med en enhet. Om nu innehållet blir 0 betyder det att

sista ordet i blocket håller på att matas ut. Vidare utmatning stoppas och en begäran om API-avbrott ges genom att overflow flag sätts (ettställs). I båda fallen ökas sedan innehållet i cell 35 och detta används så som adress till den minnescell vars innehåll matas ut till plotterregistret via datakanal.

När API-avbrottet kommer hoppar datorn till den plats i minnet som bestäms av innehållet i cell 70, och exekvering fortsätter därifrån vid hög prioritet till dess en DBR- eller DBK-order (de-break) ges. man måste dessförinnan även ge en order som avlägsnar (nollställer) overflow flag, så utmatning kan fortsätta utan nytt API-avbrott.

Bläckmatningen är ännu lite trög, så man bör inte ha någon signifikant information de första 10-20 x-värdena.

Indata.

Indata består av x- och y-koordinat för en utgångspunkt samt en rad steg från utgångspunkten. Se nedan för typiskt ex.

Ett steg till en grannpunkt kan i ett rätvinkligt koordinatsystem tas i 8 riktningar. Det skulle alltså räcka med 3 binära positioner för att lagra ett steg. För att få enklare kodning används dock 4 positioner. Man kan då låta dem betyda, i tur och ordning, steg i positiv och negativ x-led, och steg i pos. och neg. y-led. Ett steg 0110 skulle då betyda ett steg snett upp till vänster från föregående position.

Med denna kodning blir vissa kombinationer onödiga eller omöjliga. Dessa kan användas för ytterligare upplysningar. Jag har använt två av dem. Först 0000, som får betyda att förra steget var det sista i raden, nu kommer nya startkoordinater. Sedan får llll betyda att förra steget var det sista i plotterbilden och att alla data är inne.

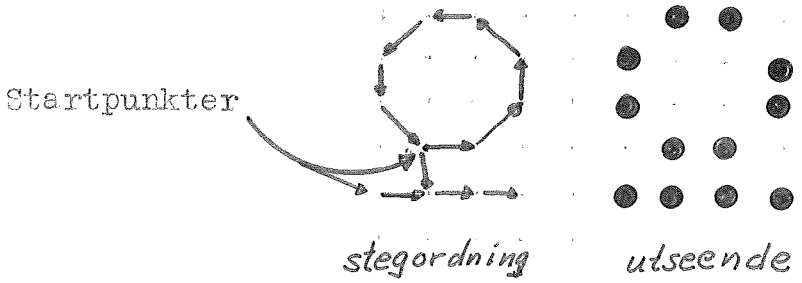
Maskinen använder ord med 18 binära positioner. Detta betyder att ett ord rymmer 4 st. steg. Man får även två positioner över, vilka kan användas till att skilja ord som innehåller begynnelsevärden från ord som innehåller steg. Då plottern använder pos. 8-17 i registret och räknaren, och då det är enklast att testa om ett värde är negativt eller positivt, används pos. 0 och 1 för denna information. En etta i position 0 innebär att ordet är ett y-värde och en etta i pos. 1 innebär att ordet är ett x-värde. Av praktiska skäl (se CHANGE) skall x-värdet alltid komma före y-värdet.

Indata finns lagrat på magnetband (DEC-tape) skrivet i dump mode där de olika filerna har namnen XXXXXA, XXXXXB, XXXXXC osv. (de 5 första bokstäverna godtyckliga enligt användarens önskan) samt en extension PLT.

Exempel på indata, filnamn EXEMPA PLT.

binärt	oktalt	vad det betyder.
0100000000001000000 märke	200100	startposition x-värde
1000000000001000000	400100	d:o y-värde
001000101000100110 steg	105046	4 steg, resp. pos. x-led; pos.x, pos.y; pos.y; neg. x, pos.y.
000100010100011001	042431	4 steg, resp. neg.x; neg.x, neg.y; neg.y; pos.x, neg.y.
000100000000000000 godtyckligt	040000	1 steg, neg.x-led samt varsel nästa värde nytt x-värde.
0100000000000111111	200077	x-värde
1000000000000111111	400077	y-värde
001000100010001111	104217	3 steg pos.x-led samt slutmärke.

Plotterfigur.



MSOPT.

Subrutinen MSOPT används för sortering av x-värden i CHANGE och y-värden i YSOPT. Den är översatt till Macro assembler från ett algolprogram Mergesort algorithm 45.Comp. Journ. vol 13 nr 1.

MSOPT har 5 variabler i anropet. N anger antalet värden som skall sorteras. Al är adressen till första cellen i den minnesarea A som innehåller de värden som skall sorteras. Ll är adressen till första cellen i den minnesarea där ordningsnummer mellan de sorterade värdena läggs in. STAPT innehåller vid uttrådet ordningsnumret för det minsta värdet i A. HDS slutligen innehåller adress till första cellen i en hjälparea.

Subrutinen kan sortera ett eller flera positiva eller negativa heltalsvärden, och sorteringen sker efter följande princip. Först indelas värdena i stigande eller fallande serier, där HDS används för att lagra ordningsnumret i A för det minsta värdet i varje serie. Ordningsnumret för nästa större värde i serien i A läggs sedan på samma plats i L som värdet ligger på i A. Se ex. fas 1.

I nästa fas slås serierna samman två och två till dess man har en enda följd. Numret på dess minsta värde läggs i START varefter uthopp sker. Se ex. fas 2.

Om vi antar att STAPT har värdet 7 så ligger alltså det minsta värdet i A på sjunde plats. På samma plats i L finns sedan platsnumret för närmast större värde osv.(i ex värde3).

Det största numret i en serie markeras med att på dess plats i L står en nolla. I ex fas 1 plats 1, 6, 8.

Det faktum att värdena i A ej flyttas gör denna sorteringsrutin mycket lämplig att använda fär sorteringen efter x-värden i CHANGE.

Exempel.

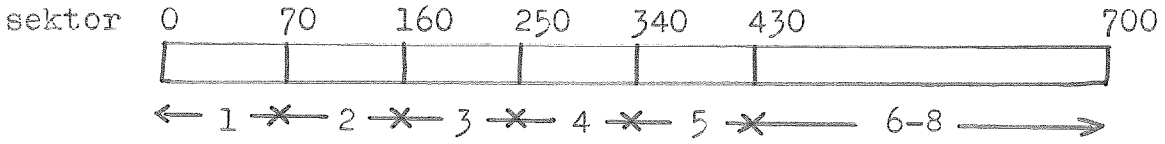
Nr	A	L	HDS	L	START
1	7	0	3	4	7
2	1	1	4	1	
3	3	2	7	8	
4	8	5		5	
5	12	6		6	
6	15	0		0	
7	-2	8		3	
8	4	0		2	
	första fasen			vid uthopp	

Skivminnets användning, INUT.

Tillgängligt för upplagring av information fanns när programmet skrevs en skivsida med 700 sektorer om vardera 400 ord. (alla siffervärden i detta avsnitt oktala). Av detta behövs hälften för in- och hälften för utinformation till subrutinerna, vilket medför att 340 sektorer finns tillgängliga för informationslagring.

CHANGE lägger ut block om 4000 ord i taget, vilka vardera motsvarar 10 skivsektorer. I UNITE skall alltså 340/10 = 34 filer förenas till en. Pga. minnets storlek kan ej alla förenas på en gång (34 x 2 in- + 2 utblock a 400 celler ger 35000 celler) utan sammansmältningen måste ske i två pass. Faktoruppdelning ger: 34 = 2 x 2 x 7 varför jag i första passet sammansmälter 7 filer åt gången till totalt 4 större filer. I andra passet framställs av dessa en stor fil.

Dispositionen av minnet blir följande:



CHANGE fyller 10 sektorer i taget från och med sektor 70 och framåt. I UNITE, första passet, sammansmälts de 7 filerna i område 2 och resultatet läggs i område 1, filerna i område 3 läggs sammansmälta i område 2 osv. I andra passet sammansmälts filerna i område 1, 2, 3, och 4 och resultatet läggs i område 5-8 dvs sektor 340 och framåt.

YSOPT omändrar sedan denna fil för utmatning och lägger resultatet i omr. 1-4, varifrån OUTPUT hämtar det för utmatning på plottern.

Lägg märke till att hela skivminnet används. Även disk directory fylls alltså med data och när sedan efter programmets körning andra data t.ex. via PIP skall läggas på skivan måste new directory specifieras (se User's Guide).

Själva in- och utmatningen sker med hjälp av den lilla subrutinen INUT. Den är uppbyggd efter mönster av macron .TRAN. Då .TPAN bara kan flytta hela sektorer bör minsta in- och utarean i minnet bestå av 400 ord.

INUT har 4 argument. Ett anger om det är in- eller utmatning. Nästa anger antal ord som skall förflyttas och de två sista ger resp. sektoradress på skivan och adress till första ord i minnesarean. Med hjälp av dessa värden byggs ett monitoranrop (CAL) upp och exekveras efter en test om in- resp. utkanalen är ledig (.WAIT).

Anledningen till att jag använder dubbla in- och utblock är att då kan datatransport pågå mellan skivan och det ena blocket samtidigt med att programmet arbetar med data i det andra. Datatransport till eller från skivan är en mycket långsammare process än vad exekveringen av en order i programmet är.

CHANGE. (flödesschema sid 20)

Change är den subrutin som sköter om omvandlingen från startvärde och steg till xy-koordinater, vilka läggs ut i lämpligt stora block.

Tillgängligt för dataförvaring i kärnminnet finns ett område på c:a 10-12000 minnesceller (som förut är alla siffror gällande minnet oktala).

Till sorteringsrutinen MSOPT åtgår här lika många hjälpceller som de som skall sorteras. Om man väljer att sortera 2000 koordinater i taget krävs 2000 celler vardera till minnesarean A (för x-värden) och hjälparean L.Hjälparean HDS används bara inom MSOPT och ej vid utläggning av sorterade värden, varför man kan utnyttja utfilsutrymme därtill. Dessutom behövs 2000 platser för tillhörande y-värden.

Inblocken kan vara små, 400 celler var, vilket ger 1000 celler.

Utmatning skall ske av mellan 2001 (ett x- 2000 y-värden, en lodrät linje) och 4000 (t.ex. en vågrät linje) värden. Om man lägger ut 2000 värden åt gången kommer alltid per sortering 2 omgångar att läggas ut, vilket gör att de sorterade filerna kommer i bestämda sektorer på skivan som önskat för att UNITE skall kunna hitta dem. Det behövs alltså 2000 celler för utfilen U (och HDS).

Totalt ger detta 11000 minnesceller. Om det inte skulle finnas så många minnesceller tillgängliga skrivs FULL MEM ut varefter uthopp sker.

Efter minnestilldelning initieras först drivrutiner för in- och utmatning för skivan. Sedan får kontrollvariabler startvärden varefter inblocken fylls.

Om inföljden börjar direkt med steg antas utgångskoordinaterna vara (0,0).

Varje ord som behandlas testas först om det är negativt, dvs. ett y-värde. Sådana läggs direkt i YY, en minnescell för y-koord. Annars skiftas det ett steg vänster. Om det då är negativt är det ett x-värde. Det skiftas tillbaka, märket avlägsnas och värdet läggs i XX. Efter varje mottaget y-värde läggs också XX och YY ut i sorteringsareorna A resp. Y. Därför måste alltid x-värdet komma före y-värdet.

Efter denna test, om ordet alltså bestod av steg, skiftas det vänster ännu en gång och sparas i sparcell OPD. Steget i position 0-3 isoleras så och testas om det är 0000 eller 1111. I första fallet tas ett nytt ord in, i det andra sker en sista sortering och utmatning varefter uthopp sker.

Utvinning av steginformationen tillgår så att hela ordet testas om det är negativt (dvs. har etta i pos. 0) och sedan skiftas vänster ett steg. Om ordet är negativt sker lämplig förändring av XX resp. YY. Observera att båda kan ändras i samma steg.

När test och skift skett 4 ggr är alla positioner i steget testade och ordet läggs ned i sin sparcell som det är. Nu ligger alltså nästa steg i pos. 0-3 och kan testas på samma sätt. En räknare kontrollerar när alla stegen i ett ord är testade.

När XX och YY ändrats, antingen genom ett steg eller via nya startvärden, läggs det nya xy-paret på samma plats i A resp. Y. Efter sorteringen anger alltså L även ordningen i Y. Om nu A och Y är fulla sker hopp till MSORT, annars sker återhopp till framtagning av steg resp. intagning av nytt ord.

Sorteringen av A ändrar ej ordningsföljden inom A utan den rätta ordningen anges av hjälpfilen L. Därför kan, när sorteringen är färdig, x-värdet och tillhörande y-värde, vilket ju ligger på samma plats i Y som x-värdet gör i A, läggas ut sorterade efter växande x i U. Varje x-värde läggs dock ut endast en gång. Som marke-

ring av slutet på U-filen läggs sist ett värde x = 4000.

På det sätt som tidigare beskrevs läggs så U ut på skivan (två omgångar) varpå återhopp sker, till intagning av ord eller framtagning av steg, eller möjligen, om det var sista sorteringen, sker ett uthopp ur CHANGE.

För varje sortering och utläggning av fil som sker, ökas en variabel ABL med en enhet. Denna variabel talar sedan om för UNITE hur många filer som finns utlagda på skivan för sammansmältning.

Innan uthoppet stängs drivrutinen för DEC-tape, vilken nu ej behövs mer.

Exempel. Utseende efter sortering av indata sid **6.**

A	Y	Nr(okt.)	E	U	START: 10
000100	400100	1	6	000077	
000101	400100	2	5	400101	
000102	400101	3	4	400102	
000102	400102	4	16	400077	
000101	400103	5	15	000100	
000100	400103	6	12	400100	
000077	400102	7	13	400103	
000077	400101	10	7	400077	
000100	400100	11	14	400100	
000100	400077	12	11	400077	
000077	400077	13	1	000101	
000100	400077	14	2	400100	
000101	400077	15	3	400103	
000102	400077	16	0	400077	
				000102	
				400101	
				400102	
				400077	

UNITE.(flödesschema sid 21)

Unite skall sammansmälta maximalt 7 filer i taget varför det behövs en minnesarea för 2 x 7 inblock och 2 utblock eller totalt 20 block om vardera 400 celler vilket ger en minnesarea på 10000 celler, vilket är acceptabelt.

När vi nu skall sammansmälta 7 st. filer på en gång måste vi hålla reda på 7 olika inadresser till skivan. det sker i speciella minnesceller. Resp. adress ökas sedan med en enhet varje gång ett block hämtas in från motsvarande fil.

Dessa minnesceller fylls med en adress i taget, varefter invärdet ABL minskas med en enhet och därefter ANT ökas med en enhet per gång. När ANT blir 7 och man alltså gett 7 skivadresser är det klart att fylla inblocken. Innan dess ökas FAST, vilket indikerar att två pass är nödvändiga (se avsnittet om skivminnet). Om innan ANT blivit 7 ABL blivit 0, dvs man har högst 7 filer att sortera, behövs bara ett pass. Man ändrar då ej FAST och lägger dessutom ut resultatet direkt till skivsektor 340 och framåt.

När man sedan har fyllt inblocken och samtidigt gett startvärdet till upp till 7 arbetsadresser sätter man en indikator XX lika med noll. Så hämtar man ett x-värde från första infilen (första arbetsadressen) och jämför det med XX. Om de inte är lika hämtar man ett värde från andra infilen osv. Om ingen infil har ett x-värde lika med XX ökar man XX med en enhet och börjar om igen.

När man hittar ett x-värde lika med XX läggs det i utblocket och samtidigt i ett x-minne varpå arbetsadressen ökas. Man hämtar in nästa värde från arbetsadressen. Om det är ett y-värde läggs det ut, arbetsadressen ökas och nästa värde hämtas in.

När man hämtat in ett x-värde ändras ej arbetsadressen utan man går till nästa fil och testar dess x-värde. På detta sätt kommer jämförelse med det nya x-värdet att ske när man kommer tillbaka

till filen nästa varv.

Nästa gång man hittar ett x-värde lika med XX jämförs det först med x-minnet innan det läggs ut, så man ej får några dubbletter av x-värden i utfilen. Varje x som läggs ut läggs också i x-minnet.

Man fortsätter så till dess XX blir 2001. Då påläggs även här en slutmarkering (ett x-värde större än 2000) och sista blocket läggs ut. Så ökas en annan räknare, LIMP, som varit nollställd från början. Den håller reda på hur många filer som läggs ut under första passet att sammansmältas under det andra.

Sedan testas om alla filer från Change genomgått en första förening (ABL = 0). Om så ej är fallet hämtas högst 7 nya filer in och man fortsätter som ovan. Om däremot första passet är klart testas om ett andra pass är nödvändigt. Om inte, så görs uthopp ur subrutinen, annars ges först nya värden till minnescellerna för skivadresser (första skivsektor nu 0, 70, 160 och 250) varpå indikatorn för sista pass, FAST, nollställes och ABL får värdet av LIMP. På nytt fylls ABL par inblock och sista sammansmältningen startar. Den tillgår på samma sätt som föregående.

Ordräknare ser till att block skiftas och läggs ut resp. tas in när ett utblock är fyllt eller ett inblock tömt.

YSORT. (flödesschema sid 22)

Ysort skall göra den slutliga förberedelsen för utmatning. Däri ingår att sortera y-värdena. För ett visst x-värde kan det finnas en y-axel samt några kurvor som korsar den. Då kan det bli över 2000 (talen fortfarande oktala) y-värden som skall sorteras. Därför har jag valt att reservera en sorteringsfil A och en hjälpfil L på vardera 2200 minnesceller. Vidare har jag reserverat 400 celler för hjälpblocket EDS.

Två inblock behövs, 400 celler var ger 1000 celler.

Beträffande utblockens storlek gäller att de måste rymma alla y-värden för ett visst x-värde. Det är nämligen ej troligt att i OUTPUT skiftet i plotterutmatning från en minnesarea till en annan skulle hinna ske under den tid plottern går från ett y-värde till nästa. Därför bör skiftet ske under den tid plottern går från ett x-värde till nästa, vilken tid är c:a 60 ggr längre. Om det sista värdet i varje block således är en nolla säkerställs minst 1,6 ms. från tiden overflow flag sätts till dess nästa data channel request kommer. I ett utblock måste alltså en hel kolonns y-värden få plats. Det ogynnsammaste fallet inträffar om vart tredje y-värde fattas. För att säkert täcka det har jag valt utblockens storlek till 2000 celler var.

Totalt åtgår alltså en minnesarea på 12000 celler.

När minnet reserverats och inblocken fyllts börjar programmet med att nollställa ett jämförelsevärde XPEG. Ett x-värde hämtas in, läggs i XSPAR och jämförs med XPEG. Om de inte är lika läggs en nolla i utblocket, XPEG ökas och jämförs åter med XSPAR. För varje nolla som läggs i rad i utblocket kommer alltså utmatning på plottern av i princip en blank kolonn att ske, och därmed en frammatning i x-led.

När så XSPAR blir lika med XPEG inhämtas efterföljande y-vär-

den från infilen ett och ett, märket i position 0 avlägsnas, och värdet placeras i sorteringsarean A samtidigt som ordräknaren för MSORT N stegas fram. Så småningom får man in nästa x-värde, vilket läggs i XSPAR, och sortering av A sker. Om antalet kvarvarande platser i utblocket är färre än antalet y-värden i A börjar det utblocket tömmas och y-värdena läggs i det andra, tomma blocket.

Innan utläggning ur A påbörjas nollställs en hjälpcell Y. När ett sorterat y-värde sedan hämtas från A jämförs det först med Y. Om de är lika hämtas ett nytt värde direkt. Om det hämtade värdet är en enhet större än Y sätts ett minne V varefter Y läggs ut för att markera startpunkten i den växande följden och det nya värdet läggs i Y. Så länge följden fortsätter att växa läggs bara det nya värdet i Y, inget läggs i utblocket. När ett nytt y-värde är två eller fler enheter större än Y testas på V om Y är sista värdet i en växande följd. Om så är fallet nollställs V och Y läggs ut "märkt" i positionerna 5,6 och 7 (se beskr. av plottern), annars läggs Y ut omärkt. Det nya värdet läggs i Y och proceduren fortsätter. När alla y-värden är tagna från A läggs ett sista Y ut, XSPAR plockas in igen och jämförelsen med XREG fortsätter.

Så småningom blir XPEG 2000, och då är alla värden behandlade. Det sista utblocket läggs på skivan.

Vid utläggning av block på skivan läggs först i första positionen av blocket antalet signifikanta celler i blocket negerat. Detta värde skall i OUTPUT användas att lägga i cell 34 för word count till plottern.

Som tecken till OUTPUT att utmatningen är färdig läggs slutligen ett block ut med ett positivt värde i första ordet. Sedan görs ett hopp ut ur subrutinen.

OUTPUT. (flödesschema sid 23)

Output behöver bara minnesutrymme för 2 inblock om vardera 2000 ord. Dessa block fylls meddde två första omgångarna som skall läggas ut.

Så läggs i cell 70 adressen till API-avbrottsprogrammet, varpå i cell 34 läggs innehållet i första ordet i första blocket och i cell 35 läggs adressen till samma ord (se beskr. av plottern).

Nu är allt klart för utmatning och den startas genom att med instruktionen 702544 flytta en etta i pos. 17 i AR till ett instruktionsregister i plottern. Bläckmatning och strålförflyttning slås till och första instruktionen hämtas. Programmet ligger nu i en loop och väntar till dess plotterutmatningen är klar.

När sedan innehållet i cell 34 (word count) blivit 0, API-avbrott begärts och erhållits görs ett hopp till ett API-program. Detta bestämmer vilket block som har tömts. Första ordet i nästa block tas in. Om det är negativt läggs det i 34 och dess adress i 35. Återställning av overflow flag och API-nivå (DBK) görs och det tömda blocket börjar fyllas på från skivan. Denna påfyllning är snabbare än plotterns behov av värden och det finns alltid ett fullt block berett när nästa API-avbrott kommer. Så sker återhopp till loopen.

Om däremot första ordet är positivt är alla data ute och plottern stoppas genom att överföra ett nollställt AR till plotterns instruktionsregister. Återställning av overflow flag och API-nivå görs och sedan görs uthopp ur subrutinen.

PLOTUT.

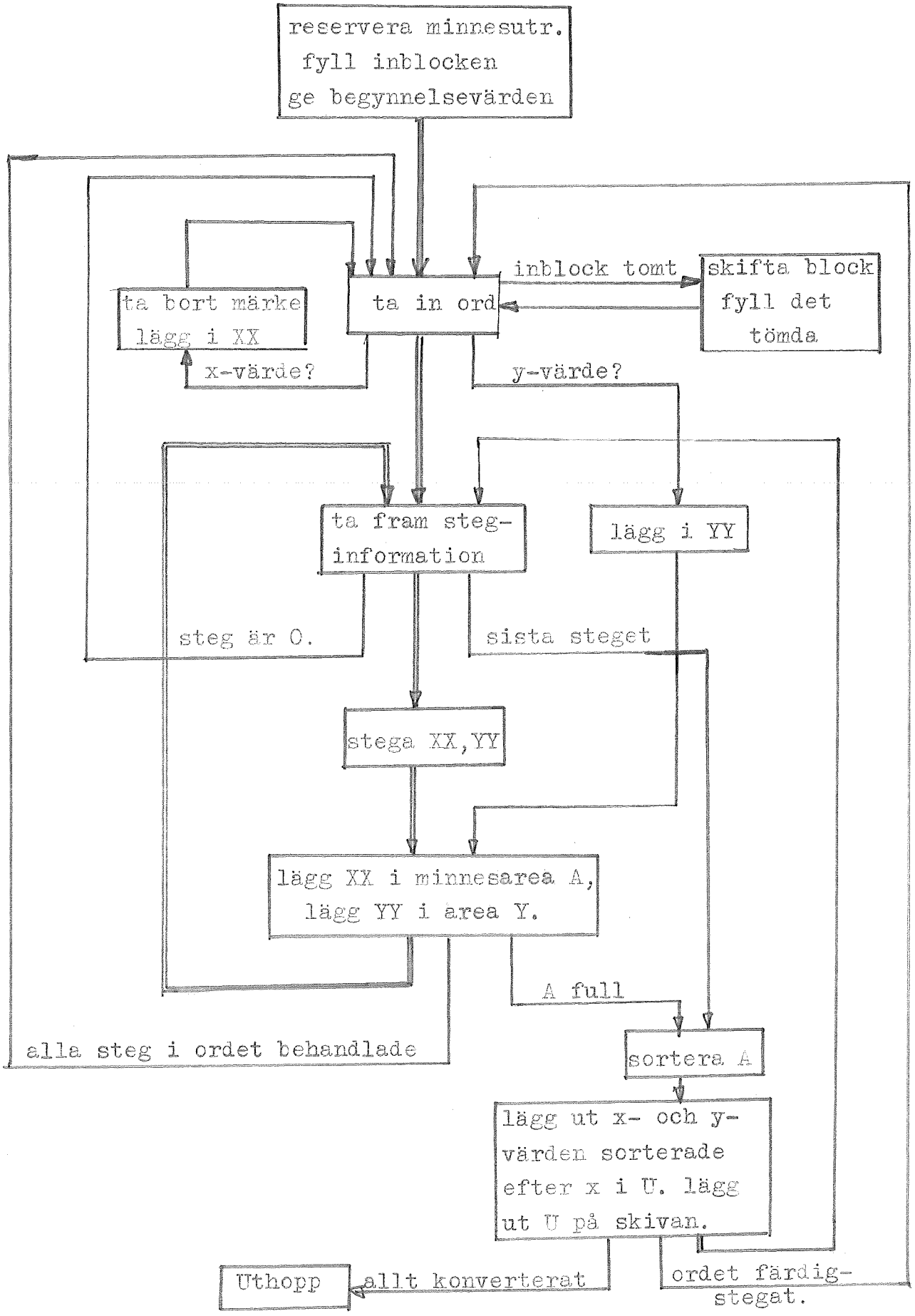
Plotut är det huvudprogram, skrivet i FORTRAN, som styr körningen genom tidigare beskrivna subrutiner.

Den börjar med att presentera sig och fråga efter namnet på DEC-tapefilerna med indata. När det fått namnet via teletypen söker det upp första filen och gör klart för inläsning varefter Change anropas. Vid återkomsten testas ABL. Om det är noll räckte inte kärnminnet till minnesareorna och ingen bearbetning har skett. Full mem skrivs då ut och uthopp sker. Då ABL är större än noll används det i anropet av Unite och sedan anropas Ysort och Output. Varje gång man kommer ut ur Output och alltså en utmatning är klar frågar programmet vart man önskar fortsätta. Tre alternativ finns.

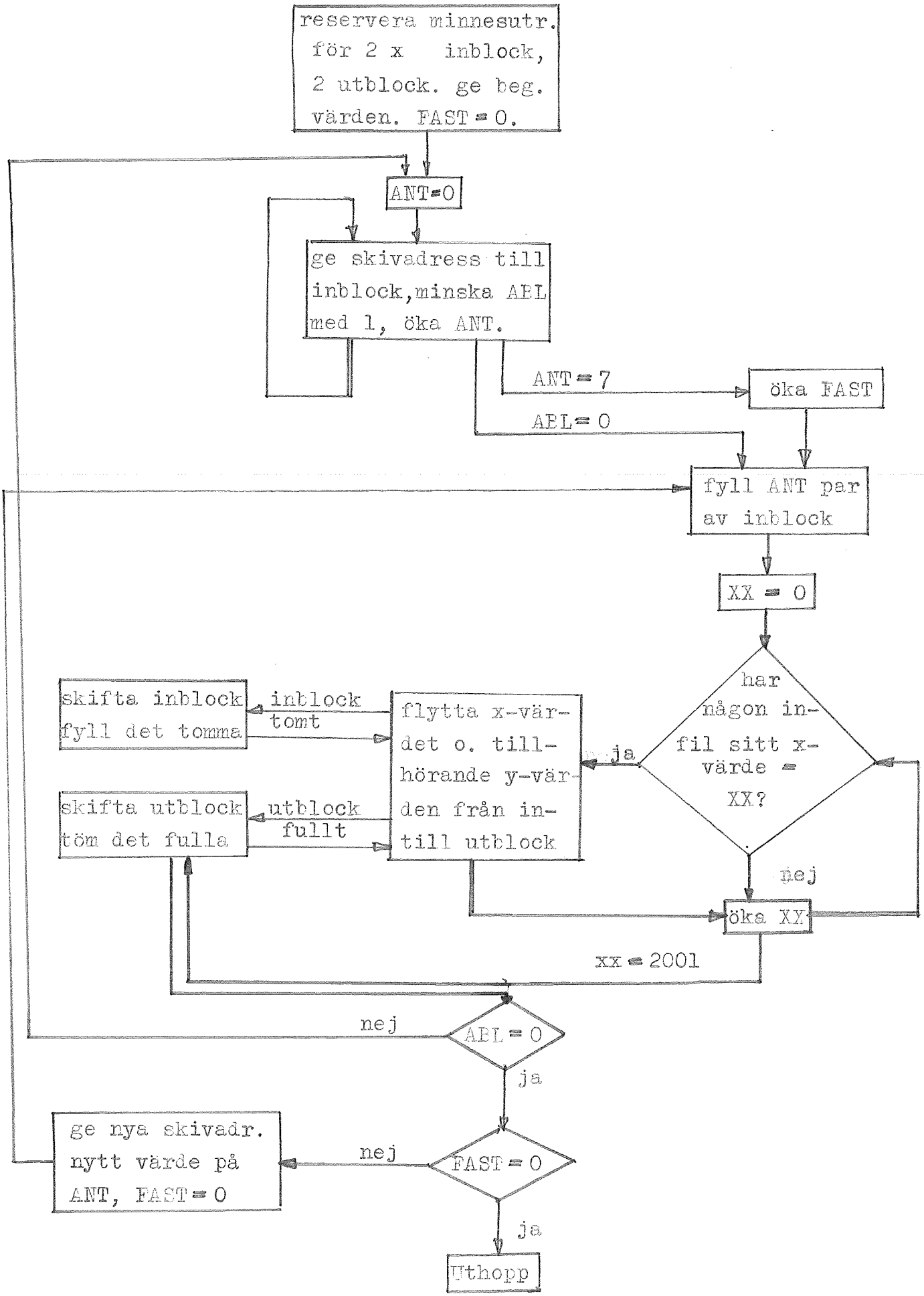
1) Ny utmatning av samma fil på plottern. Man anropar då Output på nytt.
2) Utmatning av nästa fil. Ordningsbokstaven i filnamnet ändras varpå återhopp sker dit där infilen uppsöks.
3) Uthopp från programmet och återgång till monitorn.

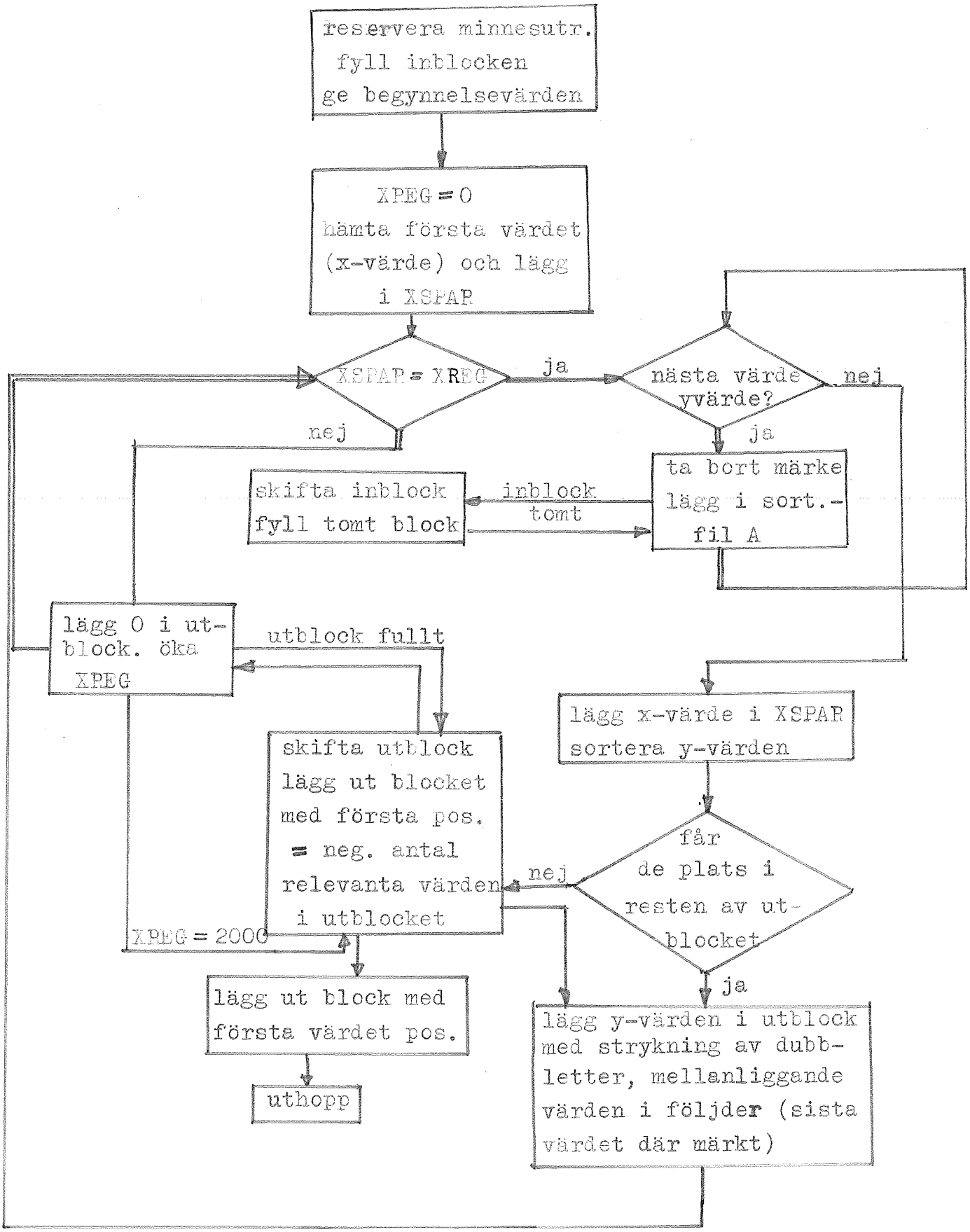
Ett påpekande: som namn skall ges bara de fem första bokstäverna, ej ordningsbokstaven. Programmet börjar alltid med A.

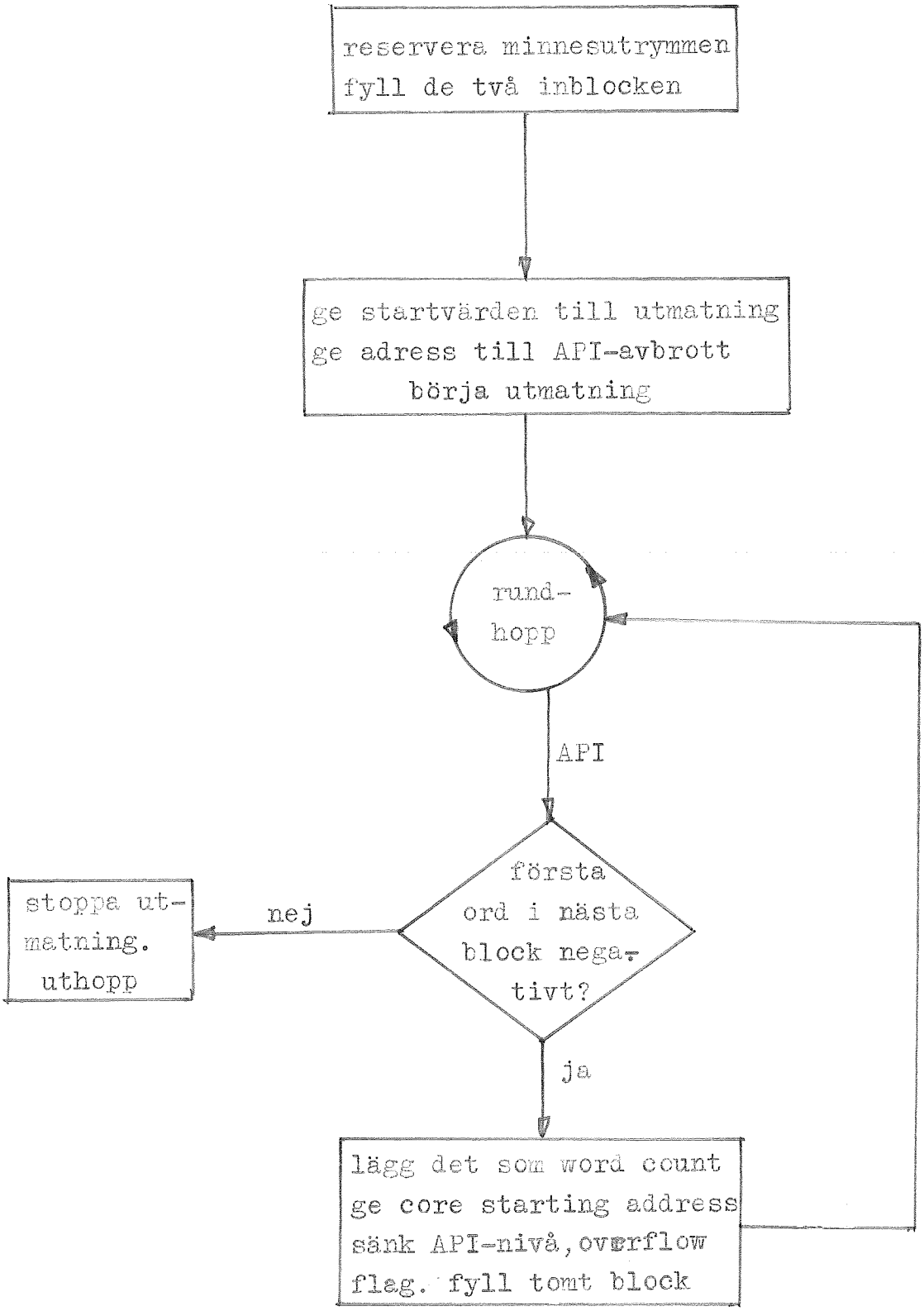
utvärde: Abl, antalet filer utlagda på skivan.



invärde: abl, är antal filer som skall förenas till en.




reservera minnesutr.
fyll inblocken
ge begynnelsevärden
XPEG = 0
hämta första värdet
(x-värde) och lägg
i XSPAR
XSPAR = XREG
ja
nej
nästa värde
yvärde?
nej
ja
ta bort märke
lägg i sort.-
fil A
inblock
tomt
skifta inblock
fyll tomt block
lägg 0 i ut-
block. öka
XPEG
utblock fullt
lägg x-värde i XSPAR
sortera y-värden
skifta utblock
lägg ut blocket
med första pos.
= neg. antal
relevanta värden
i utblocket
får
de plats i
resten av ut-
blocket
nej
ja
XPEG = 2000
lägg ut block med
första värdet pos.
uthopp
lägg y-värden i utblock
med strykning av dubb-
letter, mellanliggande
värden i följder (sista
värdet där märkt)


reservera minnesutrymmen
fyll de två inblocken
ge startvärden till utmatning
ge adress till API-avbrott
börja utmatning
rund-
hopp
API
första
ord i nästa
block nega-
tivt?
nej
stoppa ut-
matning.
uthopp
ja
lägg det som word count
ge core starting address
sänk API-nivå, overflow
flag. fyll tomt block

Appendix A: Programhuvuden:

```
C     TITLE PLOTUT
C
C     THIS PROGRAM TAKES FILES OF START COORDINATES AND STEPS AND
C     SHOWS THEM ON PLOTTER
C
C     AUTHOR LENNART HAGBJER
C
C     REFERENCE, EXAMINATION PAPER
C
C     INPUT FROM TELETYPE: NAME OF FILES, NR TO CHOOSE BETWEEN SAME
C     FILE OUTPUT, NEXT FILE OUTPUT, JUMP OUT
C
C     SUBROUTINES REQUIRED: CHANGE,UNITE,YSORT,OUTPUT,INUT,MSORT
C
```

```
      .TITLE CHANGE
      .
      /SUBROUTINE CHANGE(ABL)
      /
      /THIS SUBROUTINE CHANGES INPUT INFORMATION OF STARTING VALUES AND STEPS
      /TO XY-COORDINATES IN BLOCKS OF 2000 COORDINATES SORTED AFTER X.
      /
      /AUTHOR LENNART HAGBJER
      /
      /REFERENCE EXAMINATION PAPER
      /
      /ABL CONTAINS NR OF BLOCKS OUTPUT ON DISK WHEN LEAVING SUBROUTINE
      /
      /SUBROUTINES REQUIRED: INUT, MSORT
```

```
        .TITLE  UNITE
/
/TITLE UNITE(ABL)
/
/THE PROGRAM UNITES SEVERAL COORDINATE FILES SORTED AFTER X INTO
/ONE FILE
/
/AUTHOR LENNART HAGBJER
/
/REFERENCE, EXAMINATION PAPER
/
/ABL CONTAINS NR OF FILES TO BE UNITED
/
/SUBROUTINE REQUIRED IS INUT


        .TITLE  YSORT
/
/SUBROUTINE YSORT
/
/THIS SUBROUTINE CHANGES FILE OF COORDINATES SORTED AFTER X TO
/BLOCKS OF Y-VALUES IN SUITABLE FORM AND SIZE TO BE HANDLED BY
/OUTPUT FOR TRANSFER TO PLOTTER
/
/AUTHOR LENNART HAGBJER
/
/REFERENCE, EXAMINATION PAPER
/
/NO ARGUMENTS
/
/SUBROUTINES REQUIRED: INUT, MSORT.
/
```

```
        .TITLE  OUTPUT
/
/TITLE OUTPUT
/
/THIS SUBROUTINE STARTS PLOTTER, TRANSFERS BLOCKS TO BE OUTPUT FROM
/ DISK TO PLOTTER, SENSES WHEN EVERYTHING IS OUTPUT, AND STOPS PLOTTER
/
/AUTHOR LENNART HAGBJER
/
/REFERENCE, EXAMINATION PAPER
/
/NO ARGUMENTS
/
/SUBROUTINE REQUIRED: INUT.
/


        .TITLE  MSORT
/
/SUBROUTINE MSORT(N,A(1),L(1),START,HDS(1))
/
/THIS SUBROUTINE SORTS ARRAY A WHITOUT CHANGING ORDER OF KEYS WITHIN
/ARRAY. LOWEST KEY IN A IS A(START). THE KEY NEXT HIGHER OF A(I) IS
/A(L(I)), WHERE L IS A HELP ARRAY OF SAME SIZE AS A. THE HIGHEST
/KEY A(J) HAS L(J)=0. HDS IS A HELP ARRAY HALF THE SIZE OF A AND L.
/
/AUTHOR LENNART HAGBJER
/
/REFERENCE, COMPUTER JOURNAL VOL 13 NR 1 FEB 1970 ALGORITHM 45
/
/N IS NR OF KEYS IN A TO BE SORTED, A(1) IS FIRST CELL IN ARRAY A,
/L(1) IS FIRST CELL IN ARRAY L, START CONTAINS NR TO LOWEST KEY IN A,
/HDS(1) IS FIRST CELL IN HELP ARRAY HDS.
/
/SUBROUTINES REQUIRED: NONE
/
```

```
        .TITLE      INUT
/
/SUBROUTINE INUT(NR,CSA,WC,DEVA)
/
/THIS SUBROUTINE TRANSFERS ARRAYS OF DATA TO OR FROM DISK MEMORY.
/
/AUTHOR LENNART HAGBJER
/
/REFERENCES, EXAMINATION PAPER, EXPANSION OF .TRAN
/
/NR IS REPLACED WITH 0, IF INPUT FROM DISK WANTED, 1000 OCTAL (512 DEC)
/IF OUTPUT TO DISK WANTED. CSA IS CORE STARTING ADDRESS, WC IS NEGA-
/TIVE WORD COUNT, DEVA IS SECTOR ADDRESS ON DISK
/
/SUBROUTINES REQUIRED: NONE
/
```