PROGRAM nittinian BITEN 24-25

Spara skärmen på kassett

Konsten att skriva spel i EX-basic

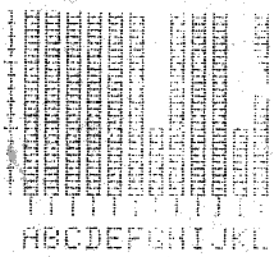
Dataöverföring i praktiken

Testbild f justering

PåskSöndagen och *ÄR TIDNINGEN TUNN*

ETT SUBPROGRAM *SAKNAR VI JUST*

HURRA vad vi är bra !!! **DITT TILLBEHÖRs-FÖRSÄLJNING BIDRAG**



Innehåll

ISSN 0281-1146

Redaktören...

Det är ett nöje att skriva dessa rader ty tidningen börjar nu få en profil och ett innehåll som passar våra olika behov och kunskaper. Det är min förhoppning att ni upplever detsamma, d.v.s. att tidningens innehåll passar era behov och er strävan att få mer kunskap om datorers möjligheter och då tänker jag speciellt på 99:an. Jag tror också att alla 59:e ägare utvecklar sig på sina räknare. Vi känner nog alla att våra kunskaper ökar mer och mer och därmed användandet av våra maskiner.

Utmaningen tar upp Procedurer, Definitioner och Subprogram i Extended Basic. Artikeln av Anders Persson är mycket informativ och lärorik, åtminstone ger den mig en hel del. Artikeln tar upp dessa möjligheter i Extended Basic på ett så ingående sätt att vi har gjort två artiklar av den. Del två kommer i PB nr.4 årets sista nummer, som vi hoppas kunna ge ut i mellandagarna.
Jag vågar inte lova för mycket men vi strävar efter att ha den klar innan Lucia den 13 dec.

Information om den engelska användarföreningen TI*MES tas upp på sidan 9.

Ordbehandling i TI-BASIC av Börje Häll på sid 10 tycker vi inom redaktionen att ni speciellt beaktar och knappar in. För att användas till nytta och nöje för oss alla. På sid 11 finner ni också Tekniska avdelningen.

Funderar ni på att köpa en bok att läsa under ledigheterna i jul rekommenderar jag er att läsa bokrescensionerna om Basicprogrammering på sid 12 och på sid 13 programmering i Forth.

Har du Mini-Memory-Modulen eller kanske Editor/Assembler-Modulen och vill lära dig Assemblerprogrammering finner du en rescension om detta på sid 25.

Tips och Tricks finns förstås också med i detta nummer. Den tar upp underligheter i Extended-Basic och vidare hur du överför Personal Record Keeping-filer till TI-WRITER-filer.

På sid 17 och framåt finner ni 59:e-program.
Först ett program som översätter färgmarkeringen på motstånd, resistorer till dess resistansvärde.
Programmet Trippelpunkter på sid 18 borde sysselsätta er hjärna ett bra tag framöver.
På sidan 20 beskrivs ett program för tennisdomare.
Uträkning av Påsksöndagar beskrivs på sid 22.
Alla dessa program är för 59:an.

Vi har ett assemblerprogram av Börje Häll, vilket tar upp en subrutin för inmatning av data från tangentbordet.

Har du börjat tröttna på färdiga Adventurespel ? Skriv ett själv vet'ja. Love Feuer skriver på sid 26 om hur du skapar dina egna Adventurespel. Ett färdigt Adventurespel listas och förklaras.
Kommer vi nu att få spännande och fascinerande spel till Programbanken ?

Längtar du efter att bygga ut ditt system eller funderar du kanske på att köpa avancerade ordbehandlingsprogram, registerprogram eller liknande ? Siktar du ännu högre möjligen att göra egna spel överförda på modul. Läs då Bengt Fahlgrens marknadsöversikt. Du hittar den på sidorna 30-31.

Några korta men lärorika TI-BASIC program finns även med i detta nummer "SCREENSAVER" och "TESTBILD". du finner dem på sid 32-33.

Dataöverföring i praktiken är en upplysande artikel om dataöverföring med hjälp av modem och dator. Själv har jag tyvärr ännu inte lyckats spara ihop till ett modem men tekniken fascinerar mig. Tänk er att ni har en kamrat i Danmark eller Norge och på några sekunder byter ni program och text med varandra. Till och med finns det möjligheter att skriva direkt till varandra s.k. "FULL DUPLEX".

Lämnad till tryck 1984-12-03

Till slut om ni är uppmärksamma ser ni att programbanken har sitt gamla utseende. Vi har inte hunnit med att bli klara med vår revidering av banken.
Vår programbanksförmedlare sitter och filar på på utformningen och uppläggningen av "nya" programbanken.
Alla program som har och kommer att publiceras i tidningen kommer att finnas tillgängliga i banken.
Vi överför alltså alla publicerade program till programbanken.
Ni kan alltså köpa dem genom föreningen.

Jag önskar er givande stunder med tidningen och dess innehåll.

Göran Nygren

Göran Nygren.

I REDAKTIONEN:

Redaktör	Göran Nygren
Biträdande redaktör	Michael Öhman
Utmaningsredaktör	Anders Persson
Programförmedlare	Björn Mårtensson
Allt-i-allo	Claes S

Föreningens och redaktionens adress:

Föreningen Programbiten
c/o Schibler
Wahlbergsgatan 6 1 tr ned
121 46 Johanneshov

Postgiro 19 83 00 - 6

DATAINSPEKTIONENs LICENSNUMMER 82100488

Medlemsavgift för 1984 är ..... 120 SEK
Nittinian, årgång 1983 (nr 1, 2, 3, 4/5) kostar 80 SEK

ANVÄNDARTIPS MED MINI MEMORY

Denna bok skrevs av Björn Gustavsson på uppdrag av Texas Instruments. Precis när boken var klar lade TI ner hemdatorn. Föreningen fick då rätten till boken och har tryckt upp den. Boken innehåller 60 sidor och kostar för medlemmar 60 SEK som sätts in på föreningens postgiro 19 83 00 - 6.

VIKTIGT MEDDELANDE

Föreningen kan från och med nr 84-1 inte lämna någon ersättning för artiklar och program enligt notis i Nittinian 2.

Annonser, insatta av enskild medlem (ej företag), som gäller försäljning av moduler eller andra tillbehör i enstaka exemplar är gratis.

Övriga annonser kostar 2000 SEK per helsida, förutom baksidan som kostar 3000 SEK.

ORDFÖRANDEN har ORDET

Genom brev och genom det allmänna medlemsmötet den 15 september i Stockholm har jag fått en litet tydligare bild av vad som intresserar medlemmar i föreningen. Högt på listan kommer

- tillbehörsmarknaden
- basmaskinen (16K, kassett, Basic/X-Basic)

Vi skall försöka möta detta intresse och börjar redan i detta nummer så vackert. Det är svårast att få fram material om basmaskinen av skäl jag redan redovisat(de som skriver är i allmänhet de som fortast tränger vidare mot dyrare och mera avancerade områden). Därför, Du som kan och vill: Skriv saker om basmaskinen som kan publiceras i tidningen - om Dig och Din maskin, om problem Du löst eller vill ha lösta, om saker Du känner till som kan intresserar andra (och det är i allmänhet mer än vad Du först föreställer Dig), om saker Du vill briljera med mm. Folk är faktiskt också ofta intresserade av lärorika misstag. Skriv gärna om sådana!

Vi ordnar ett nytt allmänt medlemsmöte den 15 december igen. Samma lokal - Militärhögskolan i Stockholm - och samma tid i veckan - kl 14 lördag em. Tre tekniska ämnen förbereds just nu - 99-an för filhantering, för ordbehandling, för datakommunikation - och ett organisatoriskt, nämligen SIG. SIG betyder Särskild IntresseGrupp och är en mindre grupp inom föreningen som samlas kring ett speciellt ämne för att ägna sig åt det. De som samlas bestämmer själva vad de vill göra: t ex hålla diskussionsmöten hos varandra, växla brev, byta program och prylar, skaffa en lärare och ordna kurs mm. Den 15 december tänker jag bara försöka se till att de som har ett gemensamt intresse träffas. Ämnen jag gissar kan vara intressanta för SIG är t ex FORTH, basmaskinen, spel, bygge och Assembler. Med bygge menar jag då lödning, EPROM-bränning o dyl.

För säkerhets skull: att vi vill intressera oss för basmaskinen betyder inte ett mindre intresse för det avancerade. Det är ju där det lockande finns när det basala fungerar!

Nu något helt annat. En brist, tycker många jag har kontakt med, är att det till 99-an inte funnits en "marknad" med piratprogram som kan bytas mellan "hackers". Med piratprogram menar jag då ett kommersiellt tillverkat program som av tillverkaren skyddats för kopiering, men som gjorts tillgängligt - "knäckts" - av hackers. TI hade ett effektivt skydd för sina program på modul. Det krävs avancerad utrustning för att knäcka dem. Nu börjar komma andra program på den öppna marknaden som tydligen går att knäcka. En bytesmarknad växer upp, har jag förstått. Om jag begripit juridiken rätt är det olagligt att ta betalt för sådana program. Däremot kan man lagligen ge bort eller byta med varandra. Lagligt eller inte - det är klart att man berövar tillverkare och lagliga distributörer en del av deras tänkta inkomst med detta. Föreningen bör därför inte, enligt min mening, aktivt medverka i saken. Vi kan heller inte gärna aktivt motarbete att medlemmarna byter program med varandra. Alltså rapporterar jag om saken så här.

Slutligen: det är snart dags för medlemmarna att ställa styrelsen till svars för det gångna året och välja en ny styrelse. Vi planerar årsmötet till

- 23 februari 1985 i Stockholm

En del styrelsemedlemmar vill använda sin tid till annat - programmering t ex. Det behövs därför en del nya namn. Anmäl Dig eller den Du vill se i styrelsen till valnämnden i god tid. Observera: det måste finnas några som gör det där styrelsearbetet om det skall bli tidning, möten mm! Vidra så gott Du kan!

Lennart Lindberg,ordförande

Valnämnd är Tony Rogvall, tfn 08/86 13 50 och Bo Nordlin, tfn 08/42 05 04.

FÖRENINGENs TILLBEHÖRs-FÖRSÄLJNING

Följande finns att köpa för medlemmar genom att motsvarande belopp sätts in på postgiro 19 83 00 - 6.

Användartips med Mini Memory		60:-
FORTH Olika versioner, se annons på annan plats i tidningen (sid 23)		
Nittinian T-tröja		40:-
99'er Magazine nr 12/82 nr 1-5,7-9/83 (per styck)		20:-
Nittinian, årgång 1983		80:-
Astronomical Formulae for Calculators		Slut
Programbiten, årgång 1983	Kalkylatorer	80:-
Programbiten, årgång 1982	-"-	80:-
Programbiten, årgång 1981	-"-	60:-
Programbiten, årgång 1980	-"-	60:-
Programbiten, årgång 1978/79	-"-	60:-
Programbiten, fem årgångar 1978-1983		280:-
Katalog med belgiska och engelska program för räknare TI-57, TI-58, TI-59		20:-
Föreningens programmeringsblanketter (TI-59), olika typer, block om 50 blanketter (se pb 83-1 sidan 30)		11:-
Patenthandlingar TI-59		25:-
40 st tomma magnetkort med plånbok		150:-
Tom magnetkortsplånbok		10:-

DATAVISION MODUL
Modulen gör att du kan kommunicera med DATAVISION eller liknande baser.
Utskrift för printer finns tillgänglig, sparning av skärmar på kassett eller diskett.
Nödvändig utrustning:
RS232 interface
Televerkets modem eller liknade
Modulen kommer med instruktioner om hur du kopplar upp dig med Televerkets modem.

Pris: 565:-

16 KRAM CMOS MODUL
utan hölje. Modulen kan simulera ett ROM, inladdning sker från T.ex FORTH vilken medför att minimum utrustning är X-basic, MINIMENORY, eller ED/ASM och 32 K minnesutökning. Ram'et är batteribackupat så programmet ligger kvar minst 2 veckor. Denna modul används i princip för att testa ut hur egna program i modulform fungerar innan man bränner prommar vilket blir klart jobbigare.

Pris: 875:-

UTMANINGEN!

Redaktörens adress:
Anders Persson
Kämnärsvägen 4:1078
222 45 LUND

Jaha, det ärväl bara att gratulera. Programbitens medlemmar tycks nämligen vara en av de mest bekymmerslösa folksamlingar som finns. Åtminstone om man ska döma av den brist på anstormning av problem som föreligger. Det är ju bra på sitt sätt, men det medför att jag får svamla alldeles på egen hand i Utmaningen. Vilket inte var meningen från början.

Såvitt jag kan förstå är en av anledningarna till insändartorkan att en del av er som inte är så väldigt avancerade i er programmering anser att det inte är någon id att ni kommer med era futtiga problem och stör storfräsarna i deras egen himmel. Ack ja, så beklagligt. Och så fel.

Nu råkar det vara så här, att vi driver den här föreningen av den enda anledningen att vi ska kunna hjälpa varandra. Avsikten är inte att tidningen Programbiten ska vara ett organ för redaktionens inbördes beundran. Enda sättet att undvika det är att ANDRA också skriver till tidningen. Personligen vägrar jag fortfarande att tro att ni saknar problem så totalt som ni försöker inbilla mig. Så kom igen nu, va?

Efter denna utflykt i kåserandets konst ska jag nu försöka ägna mig åt det jag borde, dvs programmeringsproblem.

Robert Prins hörde av sig efter det att jag hade skrivit förra Utmaingen. När det gäller beräkningen av ln(x) med "många" siffror efterlyser han en bok med tabeller med fler än 13 siffror på ln(x). En sådan bok är Handbook of Mathematical Functions av Milton Abramowitz och Irene A. Stegun. Den refereras det flitigt till i mycket av litteraturen som behandlar programmering av mer eller mindre avancerad matematik. Titta till exempel i bruksanvisningen till en del moduler till 59:an, ni som har en sådan.

I Handbook of... finns det en hel del tabeller på olika funktioner, med 15-25 siffrors upplösning. Dessutom finns det massor av primitiva funktioner, derivator, transformationer och serieutvecklingar (t. ex. de som Robert kallade inverterade serier i förra numret).

Jag kan inte låta bli att undra vad han ska använda alla siffror till när han väl har räknat ut dem, men det får väl vara Roberts problem.

Eftersom er aktivitet har varit så liten sen sist (klaga är jag duktig på) får jag nu hitta på något annat att skriva om. Då det är min uppfattning att en och annan har fått det här med underprogram lite om bakfoten tänkte jag försöka reda ut begreppen lite. Kanske det blir krångligare i stället, men... ja, allt har ju sina risker.

Först lite definitioner, så att vi vet vad vi talar om.

EXEKVERING betyder att ett programs instruktioner utföres av datorn. Ett BASIC program exekveras av datorn när man skriver RUN.

ALGORITM är den metod som ett program utnyttjar för att utföra sin uppgift. Algoritmen är helt enkelt lösningen på ett problem, och kan utnyttjas inte bara av en dator för att utföra lösningen. Även en människa kan följa algoritmen och komma fram till samma resultat, om än inte lika fort.

HUVUDPROGRAM kallas den del av ett program som exekveringen startar och, förhoppningsvis, slutar i. Huvudprogrammet bestämmer i vilken ordning olika delar av programmet skall exekveras.

UNDERPROGRAM är program inuti ett huvudprogram. Underprogram använder man av olika anledningar, i regel en eller flera av dessa:

- Likartade avsnitt återkommer på flera ställen i algoritmen.

- Vissa avsnitt är av intresse även för andra problem.

- Algoritmen är så stor att den är svår att överblicka i ett sammanhang.

Underprogram kan användas i alla program, oberoende av vilket språk programmet kodas i. Vissa krav måste naturligtvis ställas på programspråket, men de uppfylles av alla språk som användes numera. Skillnaden är att vissa uppfyller kraven i högre grad än andra.

Det som måste finnas är en möjlighet att anropa ett underprogram från huvudprogrammet och sedan låta underprogrammet återvända till satsen efter anropet. Dessutom måste underprogram kunna anropa varandra.

Man skiljer på olika sorters underprogram. Det finns i princip två typer, nämligen de som ger ett värde tillbaka till det anropande programmet och de som inte gör det. Härav dessa två definitioner:

FUNKTIONER är underprogram som vid anropet ger ett värde tillbaka till det anropande programmet. Hur mycket arbete som funktionen måste göra för att komma fram till detta värde är likgiltigt.

PROCEDURER kallas de underprogram som inte ger ett visst värde tillbaka, utan "bara" uträttar någonting. En sorteringsrutin (nu är vi där igen!) är ett bra exempel.

Hur utnyttjar man nu dessa för programmeringen så vesäntliga verktyg i 99:ans BASIC?

* FUNKTIONER

Låt oss titta på några funktioner.
I TI BASIC finns det ingen standardfunkton som ger värdet på konstanten pi. Om man vill kan man lätt DEKLARERA en sådan. Det innebär helt enkelt att man talar om för datorn hur man vill att funktionen PI ska se ut. I BASIC tar det sig ut så här:

```
DEF PI=3.1415926539
```

Ett program som utnyttjar deklarationen av PI kan se ut så här.

```
100 DEF PI=3.1415926539
110 INPUT "RADIE? ":RADIE
120 PRINT "CIRKELAREAN ÄR";PI*RADIE2
```

Kan man inte skriva

```
100 PI=3.1415926539
```

i programmet då? Jo, det kan man, men då är konstanten pi representerad av en variabel i programmet, vilket inte helt rimmar med definitionen på en konstant.

Antag att man någonstans av misstag råkar skriva

```
PI=PI*RADIE2
```

Då har plötsligt konstanten pi fått ett helt nytt värde, vilket säkerligen INTE är det som programmeraren hade tänkt sig. Om däremot PI deklareras med hjälp av DEF kommer datorn att protestera mot att PI ändrar värde. Ett fel mindre för programmeraren att leta upp, alltså.

PI är ett exempel på en konstant funktion. Ofta vill man att en funktions värde ska bero på ett annat värde. Om vi i exemplet ovan ska skriva ut en tabell med olika cirkelareor för radier mellan 1 och 20 meter, kan det vara bra att ha en funktion som ger arean som resultat om man talar om för funktionen vilken radie den ska räkna med. Det behövs alltså någon möjlighet för det anropande programmet att skicka med ett värde i anropet, en så kallad PARAMETER. Låt oss titta på en utvecklad variant av programmet ovan, som skriver ut arean för alla radier mellan 1 och 20 meter.

```
100 DEF PI=3.1415926539
110 DEF CIRKELAREA(CIRKELRADIE)=PI*CIRKELRADIE2

120 FOR RADIE=1 TO 20
130 PRINT "Radien";RADIE;"ger arean";
    CIRKELAREA(RADIE)
140 NEXT RADIE
```

Programmet ovan innehåller två funktioner, en utan parameter och en med en parameter. Här är det av intresse att man skiljer på olika sorters parametrar.

FORMELL PARAMETER kallas den parameter som finns i deklarationen av ett underprogram. I funktionen CIRKELAREA är det CIRKELRADIE som är den formella parametern. Observera att en formell parameter inte är en variabel i vanlig mening! Det finns ingen variabel i programmet ovan som heter CIRKELRADIE. CIRKELRADIE används bara för att man vid deklarationen av funktionen ska kunna ange hur parametern utnyttjas vid beräkningen av funktionens värde. Det är först när funktionen anropas som det framgår vilken variabel som ska utnyttjas av funktionen.
Dessutom är en formell parameters namn LOKALT i det underprogram där parametern är deklarerad. Det innebär att även om det HADE funnits en variabel i programmet ovan som hette CIRKELRADIE, hade den inte haft något med den formella parametern med samma namn att göra. När den formella parametern CIRKELRADIE användes för att referera till ett visst värde på RADIE, behåller ändå en eventuell variabel med namnet CIRKELRADIE sitt värde oförändrat.

AKTUELL PARAMETER kallas den variabel som utgör parameter vid anropet av ett underprogram. I programmet ovan är det RADIE, vilken är en variabel, som är den aktuella parametern. Namnen på aktuella och formella parametrar har inte med varandra att göra. I programmet ovan heter parametern CIRKELRADIE i deklarationen men RADIE i anropet. De kunde hetat XYZ respektive K2_318X i stället, men det hade knappast underlättat förståelsen för programmet. De kunde också haft samma namn, utan att det hade givit några problem.

Det här kanske kan tyckas vara onödigt tjat, men man måste ha förstått skillnaden på formella och aktuella parametrar när vi kommer in på SUB program senare.

Hittills har vi sett två numeriska funktioner, alltså funktioner som ger ett tal som resultat. Funktioner måste inte vara numeriska, utan kan vara av olika typer. I BASIC erbjudes tre varianter. Den numeriska har vi redan sett exempel på. Dessutom finns det möjlighet att använda strängfunktioner, vilka ger en text som resultat, och logiska funktioner, vilka ger ett av två värden som svar. Sant eller falskt.

En logisk funktion kan man använda exempelvis om man i ett program ofta vill se om olika värden överstiger ett visst fastställt värde. Funktionen OVER100 i programmet nedan talar om ifall den aktuella parameterns värde överstiger 100.

```
100 DEF OVER100(X)=X>100
110 INPUT "TAL? ":NUMBER
120 IF OVER100(NUMBER) THEN PRINT "TALET AR STÖRRE ÄN
    100"
130 GOTO 110
```

Nu undrar ni kanske vad det är för finess med att skriva OVER100(NUMBER) i stället för NUMBER>100 på rad 120. I och för sig undrar väl jag det också. I det här exemplet är det snarare så att den sista varianten är bättre. Fördelen med den första är bara att den illustrerar en logisk funktion.
Men ibland kan det vara bättre att definiera en logisk funktion än att skriva ut villkoret i varje IF-sats. Det finns ju möjlighet att krångla till uttrycket i deklarationen av funktionen i stort sett efter behag. På det sättet kan man uttrycka ett komplicerat villkor med ett kort namn i IF-satsen. Därigenom kan man få plats med fler satser på raden, och kanske slippa splittra upp satserna efter THEN och ELSE på flera rader, med därtill hörande GOTO och annat elände.

Deklarationen av en logisk funktion ser ut exakt som om det vore en numerisk funktion. Hur kan då datorn veta vilket det ska vara?
Jo, det är helt enkelt så att datorn betraktar talet noll som falskt och alla andra tal som sant. Funktionen OVER100 ger som resultat noll (0) om den aktuella parametern är <= 100, och -1 om den är > 100. Alla jämförelser som datorn gör ger endera av dessa två värden som resultat. Prova med att skriva in satser i stil med

```
PRINT 500>100
PRINT "ANDERS">"PERSSON"
```

så får ni se. Eftersom datorn inte skiljer på det numeriska värdet -1 och det logiska värdet sant, kan man blanda dessa som man vill. Den här möjligheten att blanda hur som helst gör naturligtvis att man får se upp, så att man själv vet vilket som är vilket.

Om man ofta i ett program behöver ta ut det första ordet i en strängvariabel, kan man deklarera en funktion som ger de bokstäver som står före det första blanktecknet. En sådan funktion kan se ut så här:

```
DEF FIRSTWORD$(WORD$)=
    SEG$(WORD$,1,POS(WORD$," ",1)-1)
```

Problemet med en sådan funktion är att om det inte finns något blanktecken i den aktuella parametern, eller om det första tecknet är blankt, får man ett exekveringsfel.

Om man hade kunnat använda en IF-sats i deklarationen av FIRSTWORD$, kunde man låta funktionen testa att det verkligen går att plocka ut ett ord innan den gör det, och annars ge en tom sträng som resultat. Varken TI BASIC eller Extended BASIC tillåter emellertid något annat än en tilldelning när man deklarerar en funktion. Nu kan man faktiskt skriva en sådan funktion ändå, genom att utnyttja det faktum att datorn inte ser någon skillnad på logiska och numeriska värden. I Extended BASIC är det ännu enklare, tack vare att det finns fler fördefinierade funktioner att tillgå. Funktionen MAX, som ger det största av två tal, tillåter en konstruktion av det här slaget:

```
DEF FIRSTWORD$(WORD$)=
    SEG$(WORD$,1,MAX(POS(WORD$," ",1)-1,0))
```

Tack vare MAX-funktionen är det lätt att ta fram det största av två tal i deklarationen av FIRSTWORD$. Som funktionen ser ut nu ger den alltså det första ordet i den aktuella parametern om det finns mer än ett ord, och annars en tom sträng. Genom att krångla till det ytterligare kan man få funktionen att ge det första ordet i den aktuella parametern, oberoende av om det finns ett eller flera ord i parametern. Det är nog den i praktiken mest användbara varianten.

```
DEF FIRSTWORD$(WORD$)=SEG$(WORD$,1,MAX(POS(WORD$,
    " ",1)-1,0)-(POS(WORD$," ",1)=0)*LEN(WORD$))
```

Deklarationen ovan fungerar faktiskt, även om den förvisso kräver en del dokumentation för att man omedelbart ska inse hur den fungerar när man tittar på sitt program sex månader senare. Kärnan är uttrycket

```
POS(WORD$," ",1)=0
```

vilket antingen blir noll (0) eller -1, beroende på var det finns ett blanktecken i strängen.

Genom att använda en del knep kan man alltså ibland komma runt bristen att man inte kan ha något annat än en tilldelning i en deklaration av en funktion. Om man däremot behöver en FOR-NEXT loop torde det vara omöjligt.

En annan begränsning är att en funktion bara får ha en parameter. Ibland behöver man fler än så. Om man i exemplet med OVER100 ovan vill kunna variera gränsen, sätta den till något annat än 100, kanske man vill utnyttja en funktion som ser ut så här:

```
DEF OVERLIMIT(VALUE,LIMIT)=VALUE>LIMIT
```

Det kan man dock INTE göra, eftersom det bara får förekomma en parameter. Nu kan man kringgå den begränsningen genom att skriva programmet som det ser ut här nedan, men det för andra problem med sig.

```
100 DEF OVERLIMIT(VALUE)=VALUE>LIMIT
110 INPUT "Temp. 1? ":TEMP1
120 INPUT "Temp. 2? ":TEMP2
130 LIMIT=100
140 IF OVERLIMIT(TEMP1) THEN
    PRINT "Temp. 1 är för hög"
150 LIMIT=130
160 IF OVERLIMIT(TEMP2) THEN
    PRINT "Temp. 2 är för hög"
```

(Ja ja, jag vet att det finns bättre sätt att skriva det här på. Det är bara ett exempel.)

Problemet här är att LIMIT inte är en parameter, utan en vanlig variabel. LIMIT är alltså inte lokal, utan GLOBAL. Om man ändrar LIMIT på ett ställe, kommer den att ha det nya värdet nästa gång man anropar OVERLIMIT, vare sig det var det man hade tänkt sig eller ej.

Sammantaget gör dessa brister att funktioner inte är så praktiska i 99:ans BASIC som de hade kunnat vara. Extended BASIC innebär inte någon förbättring i det avseendet.

* PROCEDURER

Nå, hur är det med den andra typen av underprogram då? Även här gäller att TI BASIC inte har mycket nytt under solen. Det är den vanliga GOSUB - RETURN varianten som bjuds. För att se hur det fungerar kan vi titta på ett exempel som fanns i Nittinian 83-4/5. På sidan 20 fanns två sorteringsprogram (känns det igen?). Det första såg ut som raderna 1000-1140 i programmet nedan.

```
 100 DIM A(100)
 110 RANDOMIZE
 120 INPUT "Hur många tal? ":ANTAL
 130 REM Generera slumptalen
 140 FOR I=1 TO ANTAL
 150 A(I)=INT(RND*10000)
 160 NEXT I
 170 REM Skriv ut de osorterade talen
 180 PRINT "Före sorteringen": :
 190 FOR I=1 TO ANTAL
 200 PRINT A(I)
 210 NEXT I
 220 REM Sortera
 230 GOSUB 1000
 240 REM Skriv de sorterade talen
 250 PRINT :"Efter sorteringen": :
 260 FOR I=1 TO ANTAL
 270 PRINT A(I)
 280 NEXT I
 290 REM Slut på huvudprogrammet
 300 END

1000 REM SHELLSORT
1010 IF ANTAL<=1 THEN RETURN
1020 GAP=INT(ANTAL/2)
1030 BYTE=0
1040 FOR INDEX=1 TO ANTAL-GAP
1050 IF A(INDEX)<=A(INDEX+GAP) THEN 1100
1060 BYTE=-1
1070 SWAP=A(INDEX)
1080 A(INDEX)=A(INDEX+GAP)
1090 A(INDEX+GAP)=SWAP
1100 NEXT INDEX
1110 IF BYTE THEN 1030
1120 GAP=INT(GAP/2)
1130 IF GAP>=1 THEN 1030
1140 RETURN
```

Programmet lagrar ett visst antal slumptal i fältet A, skriver ut dem, sorterar i växande ordning och skriver ut talen igen. För att det ska fungera måste huvudprogrammet och underprogrammet vara överens. Om Shellsort sorterar fältet A måste huvudprogrammet ta hänsyn till detta och lagra talen som ska sorteras i just fältet A. Dessutom måste huvudprogrammet se till att låta bli att använda variabler som används vid sorteringen. Sådana är INDEX, GAP, BYTE m.fl.

Som vi ser finns det inga parametrar. Dessutom är alla variabler i sorteringen globala, vilket ställer till problem om man råkar använda samma variabelnamn på något annat ställe i programmet. Även om man bortser från risken för att blanda ihop namnen uppstår andra problem. Antag att man önskar använda samma sorteringsrutin till att sortera flera olika fält med olika namn och olika många element. Då har man det så lagom roligt. Ytterligare ett problem är att om vi inte hade satt ett END före rad 1000 hade datorn glatt fortsatt ner i underprogrammet, trots att vi inte anropat det med GOSUB. När datorn då hade kommit till RETURN, hade den inte vetat var den skulle ta vägen i programmet. BÖÖP och felmeddelande som resultat. Den här sortens procedurer är alltså inte självständiga program inuti ett större. Ovanpå allt elände säger en rad som GOSUB 1000 inte ett skvatt om vad som egentligen händer på rad 1000. Jämför med anropen av funktionerna, där ett lämpligt val av namn på funktionen gör att man genast kan säga vad det är för värde funktionen beräknar. I det lilla programmet ovan är det lätt att överblicka allt som händer, men när utskriften av programmet är tjock som en värre roman blir det omöjligt.
GOSUB - RETURN är ingen lyckad variant av underprogram, men ändå ofta det enda som finns i BASIC.

* SUB-PROGRAM

Välvilligt nog finns det i Extended BASIC en konstruktion som tillåter deklaration av riktiga procedurer. Dessa procedurer har alla egenskaper som funktionerna har och halva himlen därtill.

Det magiska ordet är SUB.
Liksom DEF inleder deklarationen av en funktion, inleder SUB deklarationen av en procedur i Extended BASIC.

En SUB har följande egenskaper:

- Den har ett namn, liksom funktionen, och anropas med sitt namn.

- Deklarationen kan, men behöver inte, innehålla en eller flera formella parametrar, vilka kan vara av olika typer.

- En SUB kan innehålla ett obegränsat antal satser, så länge minnet räcker till.

- Alla variabler som en SUB använder är lokala, och påverkar alltså inte andra variabler utanför SUB:en, även om dessa råkar ha samma namn.

- En SUB:s satser är också privata. Huvudprogrammet kan inte komma åt raderna inuti en SUB på något annat sätt än genom det avsedda anropet.

Sammantaget ger detta fördelar som man sällan hittar i BASIC-språk. En SUB är i praktiken en förenklad variant av Fortrans PROCEDURE.

Innan jag ger mig in på en detaljerad förklaring av hur ett SUB program deklareras och hur man skickar parametrar till en SUB kan vi se på hur exemplet med GOSUB ovan kan ta sig ut om man använder SUB i stället.

```
 100 DIM B(100)
 110 RANDOMIZE
 120 INPUT "Hur många tal? ":ANTAL
 130 REM Generera slumptalen
 140 FOR I=1 TO ANTAL
 150 B(I)=INT(RND*10000)
 160 NEXT I

 170 REM Skriv ut de osorterade talen
 180 PRINT "Före sorteringen": :
 190 FOR I=1 TO ANTAL
 200 PRINT B(I)
 210 NEXT I

 220 CALL SHELLSORT(B(),ANTAL)

 230 REM Skriv de sorterade talen
 240 PRINT :"Efter sorteringen": :
 250 FOR I=1 TO ANTAL
 260 PRINT B(I)
 270 NEXT I
 280 REM Slut på huvudprogrammet

1000 SUB SHELLSORT(A(),ANTAL)
1010 IF ANTAL<=1 THEN SUBEXIT
1020 GAP=INT(ANTAL/2)
1030 BYTE=0
1040 FOR INDEX=1 TO ANTAL-GAP
1050 IF A(INDEX)<=A(INDEX+GAP) THEN 1100
1060 BYTE=-1
1070 SWAP=A(INDEX)
1080 A(INDEX)=A(INDEX+GAP)
1090 A(INDEX+GAP)=SWAP
1100 NEXT INDEX
1110 IF BYTE THEN 1030
1120 GAP=INT(GAP/2)
1130 IF GAP>=1 THEN 1030
1140 SUBEND
```

Skillnader?
För det första har underprogrammet inte riktigt samma utseende. Deklarationen inleds med SUB, följt av procedurens namn och parameterlistan. I parameterlistan finns formella parametrar motsvarande den enda information sorteringsrutinen verkligen behöver utifrån. A() representerar ett endimensionellt numeriskt fält, ANTAL ett numeriskt värde. Vid anropet skall de aktuella parametrarna vara av samma typ som motsvarande formella parametrar.

Deklarationen av proceduren avslutas med SUBEND. Alla rader mellan SUB och SUBEND tillhör proceduren. Det END som tidigare stod framför underprogrammet behövs inte längre, eftersom datorn slutar exekvera huvudprogrammet när den kommer till deklarationen av ett SUB program. Härav följer att SUB program måste deklareras efter huvudprogrammet. Ett program kan innehålla flera olika procedurer, vilka kan anropa varandra. Raderna mellan SUBEND i en procedur och SUB i nästa får endast innehålla kommentarer.

När datorn kommer till SUBEND under exekvering av ett underprogram, hoppar den tillbaka till satsen efter anropet. Om man vill kan man hoppa tillbaka innan man kommit till SUBEND, genom att använda instruktionen SUBEXIT. Ett SUB program kan innehålla flera olika SUBEXIT, men bara ett SUBEND.

Nu kommer säkert vissa personer att protestera mot ett sådant förfarande. Man brukar numera säga att ett underprogram skall ha en ingång och en utgång, varken fler eller färre. I princip håller jag med om detta, men eftersom Extended BASIC inte tillåter hur långa IF - THEN - ELSE satser som helst, blir det ganska krångligt att följa den regeln i SUB program. Det finns ju inte WHILE - DO eller REPEAT - UNTIL heller. I rutinen ovan skulle man kunna skriva

```
IF ANTAL<=1 THEN 1140
```

på rad 1010, men det blir det inte lättare att förstå av. Därför anser jag att läsligheten snarare ökar om man använder SUBEXIT i stället för en GOTO till raden med SUBEND.

Alla variabler som används inuti SHELLSORT är nu lokala, även om det inte märks i programmet ovan. Där finns ju inga variabler utanför SUB programmet med samma namn, varför konflikter inte kan uppstå. Men om det hade funnits sådana variabler, hade det ändå inte gjort någon skada. Detta beror tekniskt sett på att variablerna inuti ett SUB program har ett eget minnesutrymme. En konsekvens av detta är att en lokal variabel som har ett visst värde när man lämnar ett SUB program, har kvar detta värde nästa gång underprogrammet anropas.

Eftersom A() och ANTAL i SHELLSORT nu är formella parametrar behöver de inte ha samma namn som motsvarande aktuella parametrar. Jag har avsiktligt bytt namn på fältet i huvudprogrammet, från A till B, bara för att visa att det fungerar ändå. Detta innebär också att underprogrammet lätt kan sortera flera olika fält, med olika namn, i samma program. Det enda som behöver ändras är de aktuella parametrarna vid anropen.

Anrop ja. Sorteringen anropas på rad 220, med instruktionen CALL. Efter CALL skall namnet på underprogrammet stå, och sedan parameterlistan. Vid deklarationen är det en lista på formella parametrar, men vid anropet är det en lista på aktuella parametrar. Observera att ordningen på parametrarna vid anropet måste motsvara den vid deklarationen.

Nu är det nog bäst att vi klarar ut det här med formella parametrar i deklarationen av ett SUB program.

Parametrarna kan vara av flera olika typer. Två huvudtyper är numeriska och strängar. Dessa kan man sedan dela upp i enkla parametrar och fält. Ett fält kan ha en eller flera dimensioner, upp till maximala sju.

Numeriska parametrar skriver man precis som vanliga numeriska variabler. Ett exempel är ANTAL i SHELLSORT. Detta är en enkel numerisk parameter, eftersom det är frågan om bara ett värde. A() i SHELLSORT är också en numerisk parameter, men den är inte enkel, utan ett fält. Parantesen efter namnet talar om att det är ett fält med en dimension. Om man vill använda ett fält med flera dimensioner skriver man ut ett komma för varje extra dimension.
ARRAY(,,,) är ett numeriskt fält med fyra dimensioner. Om man vill ha tag i ett värde i en sådan parameter kan man skriva

```
PRINT ARRAY(14,2,4,3)
```

Det syns här att vid deklarationen tar man bara bort själva indexen i fältet. Parantesen och kommatecknen blir kvar.

Orsaken till att man inte har några index i deklarationen av underprogrammet är att det inte finns något som säger hur många olika element som fältet ska innehålla. Vid sorteringen ovan har den aktuella parametern B() 101 element, numrerade från 0 till 100. Element nummer noll (0) behövs inte, eftersom SHELLSORT bara sorterar från och med element ett till och med elementet som anges av parametern ANTAL. Om man på ett annat ställe i programmet vill sortera ett fält med 200 element går det bra.

Både A() och ANTAL är numeriska parametrar. Om man vill deklarera formella parametrar av strängtyp gör man precis som när man använder strängvariabler. Namnet avslutas med ett dollartecken. På sidan 8 i Programbiten 84-2 finns två SUB program som använder parametrar av både sträng- och numerisk typ. I underprogrammet INS är STRING$ och INSERT$ av strängtyp, medan POSI är av numerisk typ.

Eftersom datorn inte ser någon skillnad på en numerisk och en logisk variabel, ser den heller inte någon skillnad på en numerisk och en logisk parameter. Logiska parametrar deklareras precis som de numeriska.

Vid anrop av ett SUB program måste de aktuella parametrarnas typ motsvara de formellas. Vid anropet av SHELLSORT är den första parametern ett numeriskt endimensionellt fält och den andra av enkel numerisk typ. Om de aktuella parametrarna är av andra typer, kommer datorn att protestera.

Att parametrarnas typer måste motsvara varandra kan ju tyckas vara ganska givet, mest beroende på att det ÄR ganska givet. Men vid anropet kan den aktuella parametern uppfylla typvillkoret på flera olika sätt. Titta på programmet nedan. Där finns ett SUB program PRINTSQR som inte gör något annat (eller gör det?) än räknar ut kvadratroten av den aktuella parametern och skriver ut resultatet.

```
100 DIM TABLE(2,11,3,4)
110 XX=128
120 CALL PRINTSQR(XX)
130 CALL PRINTSQR(XX)
140 CALL PRINTSQR(500)
150 XX=128
160 CALL PRINTSQR(XX+0)
170 CALL PRINTSQR((XX))
180 CALL PRINTSQR(XX)
190 TABLE(1,2,1,5)=312
200 CALL PRINTSQR(TABLE(1,2,1,5))

210 SUB PRINTSQR(VALUE)
220 VALUE=SQR(VALUE)
230 PRINT VALUE
240 SUBEND
```

Själva SUB programmet är enkelt. Att utskriften blir kvadratroten av den aktuella parametern torde det inte råda något tvivel om. Att rad 120 ger 11.31 (ungefär) som resultat var alltså väntat. Men när samma sak göres en gång till på rad 130 blir resultatet 3.36! Varför nu det då? Har XX bytt värde?

Ja, det är just det som har hänt. I PRINTSQR tilldelas VALUE kvadratroten av VALUE. Eftersom VALUE här refererar till den aktuella parametern XX blir det i själva verket XX som tilldelas kvadratroten av XX. Variabler som används inuti SUB program är lokala, liksom NAMNEN på formella parametrar är lokala, men vid anropet motsvarar den formella parametern en aktuell sådan. Därför medför en ändring av den formella parametern att den aktuella OCKSÅ ändrar värde. Det är denna mekanism som gör att en procedur kan ge värden tillbaka till det anropande programmet. Efter sorteringen i SHELLSORT ska den första aktuella parametern ju vara sorterad. Om underprogrammet inte kunde ändra värdet på den aktuella parametern vore hela idn omöjlig.

I vårt exempel för det dock med sig den oönskade effekten att värdet av XX ändras. Vad göra åt detta? En möjlighet är naturligtvis att ändra i PRINTSQR. Men om man i en annan del av huvudprogrammet verkligen vill att den aktuella parametern ska ändra värde är det ingen bra lösning. Det finns dock en annan möjlighet. Låt oss titta vidare på programmet, så ska vi se hur det kan ta sig ut.

På rad 140 anropas PRINTSQR med den aktuella parametern 500. Parametern är alltså en konstant. Det vore inte bra om konstanten 500 kunde få ett annat värde bara för att man har exekverat ett underprogram. Tursamt nog får den inte det heller. Överhuvudtaget gäller att alla uttryck behåller sina värden när de utgör aktuella parametrar, även om deras formella motsvarigheter ändrar värde i underprogrammet. Som kuriosa kan nämnas att Fortran- kompilatorer ofta missar den här konstruktionen, varför man där faktiskt kan få en konstant att ändra värde i programmet. Jätteskojigt blir det att förstå vad det är som är fel i ett program om alla konstanter med värdet 10 plötsligt får värdet 32. Ännu roligare brukar det bli om man ändrar värde på konstanten 1 (ett), eftersom den används på så många ställen i de flesta program.

I och med att värdet av uttryck inte ändras kan man skriva som det står på rad 160. Ännu elegantare är varianten på rad 170, vilken också föreslås av TI i bruksanvisningen till Extended BASIC. Att det fungerar beror på att Extended BASIC tolkar parentesen runt den aktuella parametern som ett uttryck.
Det är inte förrän på rad 180 som XX ändrar värde igen.

Rad 200 illustrerar en annan variant av parameter. Här är TABLE visserligen ett fält och den formella parametern av enkel typ. Att det ändå fungerar beror på att den aktuella parametern utgöres av ett element i fältet. Eftersom ett numeriskt fält är sammansatt av en massa enkla numeriska variabler, är ett element i fältet av enkel numerisk typ. Följaktligen passar parametrarna ihop.

Nu har vi faktiskt gått igenom det mesta av det man behöver veta för att kunna använda funktioner och procedurer i Extended BASIC. Det finns ytterligare några saker att tänka på, men de visar sig i de exempel som kommer här efter.

Att Extended BASIC har en del färdiga procedurer lär väl inte ha undgått någon. Ta CLEAR, CHARSET eller SPRITE t.ex. Att man kan ändra på funktionen hos dessa procedurer är däremot inte så uppenbart, men det kan man faktiskt. Hur då? Jo, helt enkelt genom att deklarera sig en egen procedur med samma namn som den befintliga.
Jaha, och vad ska nu det vara bra för då, tänker ni. Antag att ett program utnyttjar joysticks. Då finns det satser av typen

```
CALL JOYST(1,X,Y)
```

på en del ställen i programmet. Om man nu inte har någon joystick måste man alltså ändra i programmet. I regel försöker man ordna till något sätt att använda tangenterna i stället för en joystick. Om man då ska ändra på alla ställen där CALL JOYST finns blir det lätt en faslig massa ändrande. Enklare är det då att skriva en egen

```
SUB JOYST(ENHET,X,Y)
```

Inuti denna procedur utnyttjar man det inbyggda underprogrammet KEY för att läsa av tangentbordet och räkna om tangentkoderna till de som JOYST brukar ge ifrån sig.

De flesta har väl lagt märke till att underprogram som SOUND, CHAR, SPRITE m.fl. kan ha olika många aktuella parametrar. Sådana konstruktioner kan man dock inte göra med SUB. Det går att skriva egna underprogram som tillåter olika antal aktuella parametrar i anropen, men då måste dessa procedurer skrivas i assembler. Anropen måste göras med CALL LINK.

Jag skrev tidigare att ett SUB programs satser är privata, dvs man kan inte komma åt dem på något annat sätt än den avsedda CALL instruktionen. Om man försöker sig på GOTO eller GOSUB in i en SUB deklarerad procedur, får man ett felmeddelande. Det omvända förhållandet gäller också. Man kan inte inifrån en SUB komma åt rader utanför proceduren. Detta gäller oberoende av om man vill hoppa till dem med GOTO, GOSUB, IF-THEN eller ON ERROR.

Som vanligt gäller att alla regler har undantag. Icke exekverbara satser av DATA- och IMAGE-typ kan man komma åt inifrån ett SUB program, även om dessa rader finns inuti ett annat SUB program. Dessutom kan man utnyttja öppna filer i en procedur, även om filen inte öppnades i proceduren ifråga.

En svaghet med den form av underprogram som SUB utgör är att det inte finns någon möjlighet att komma åt variabler utanför underprogrammet, såvida de inte är parametrar. I ett program där många av underprogrammen använder ett antal variabler i huvudprogrammet, gör detta att parameterlistorna blir långa och otympliga. I Fortran har man löst detta problem genom att man kan deklarera variabler som COMMON. De variabler som finns i ett särskilt COMMON block i minnet kan då nås av alla underprogram.
Att parameterlistorna ibland kan bli ganska omfattande visar sig i det exempel som får avsluta denna utflykt i programmeringskonsten men det exemplet tar vi som del II i nästa nummer (84/4).

■

UPPROP

ÄR TIDNINGEN TUNN ?

SAKNAR VI JUST

DITT

BIDRAG

och

Tar DU KONTAKT med andra

MEDLEMMAR i din närhet

med hjälp av MATRIKELN

?

Ingen telefon ?! !?

DU får väl SKRIVA då

TI*MES

ENGELSK Användareförening för TI 99/4A

Av Peter Odelryd

Det finns i England en förening som heter TI 99/4A Exchange. Föreningens namn kommer av att den bl.a. driver en utbytesservice för moduler. Den har också startat en programbank (sänd in ett - få två i utbyte). I våras hade föreningen cirka 800 medlemmar. Den ger ut en tidning som heter TI*MES och kommer ut kvartalsvis. Det senaste numret, daterat oktober -84, är på ett 60-tal sidor. Innehållet består till en rätt stor del av fasta sidor som skrivs av samma författare i varje nummer. Sidorna brukar vara matnyttiga, med tips för både nybörjare och mer avancerade läsare. De innehåller ofta programmeringstips samt rätt mycket programrecensioner. En del av materialet är godbitar som tidigare publicerats, bl.a. i australiska 99-tidningar. Exempel på material ur senaste numret är en artikel hur man får upp hastigheten på program som innehåller tal genom att "poka" i minnet (CALL LOAD).

Annonser finns det också en hel del av. Produkter som annonserats i de senaste numren och som kanske kan vara av intresse är bl.a. följande.
* En firma erbjuder sig att för 7 pund byta ut batteriet i Mini Memory-modulen mot ett utbytbart sådant.
* Milton Bradleys MBX-system med röstinmatning till speciella spelmoduler säljs av en postorderfirma i Manchester. (MBX beskrevs i korthet i 99'er november 1983). 10 spelmoduler finns att tillgå till detta. Dock går det inte att använda i egna program. Tillverkaren släpper inte ut någon information om styrprogrammet som finns i modulerna. Pris för MBX-systemet cirka 90 pund.

Ett stort antal spelmoduler finns uppenbarligen att köpa på postorder i England. Extended Basic, Mini Memory, Terminal Emulator II, Editor/Assembler, Multiplan, TI Writer och Logo II verkar också gå att få tag på, liksom kort för expansionsboxen från flera tillverkare. Spelkassetter band alltså verkar det finnas ett större utbud på än här hemma.

Innan man börjar jämföra priserna i England bör man tänka på ett par saker. För det första kan man om det gäller dyrare varor slippa den engelska momsen (VAT). Den är på 15 %. Sen måste man också räkna med att få betala svensk moms och tull vid införseln. Det kan bli totalt cirka 30 %. Frakt tillkommer också. De firmor som annonserar i TI*MES kan man nog räkna med är seriösa jag har själv handlat från de som nämns nedan.

Skriver man och begär information om vad de har i lager måste man bifoga minst en internationell svarskupong för att betala portot för katalogen. Dessa finns att köpa på posten och kostar 4 kr/styck. Att ringa går förstås också bra. Ring upp och fråga om de har det Du söker i lager. Be dem i så fall räkna ut vad det kostar, inklusive frakt (och ev. frändraget VAT). Be att få återkomma om en stund när de har kollat det.

Adresser:

Parco Electrics
4 Dorset Place
New Street
Honiton
Devon EX14 8QS
Tel: 0404 44425

TI 99/4A Exchange
40 Barrhill
Patcham
Brighton
East Sussex BN1 8UF

Arcade Hardware
211 Horton Road
Fallowfield
Manchester M14 7QE
Tel: 061 225 2248

■

ORDbehandling i TI-Basic

Av Börje Häll (programmerare) och Göran Nygren (hjälpprogrammerare och artikelförfattare).

Inom Programbitens redaktion har vi länge tyckt att vi i onödan suttit på kvällar och helger och skrivit in på diskett de texter som flitiga och ambitiösa medlemmar sänt oss för publicering i Programbiten - hand-och maskinskrivna, ofta mycket snyggt, men i behov av transformering till datormedium innan de kan tryckas. Det tar lång tid, tidningen blir lätt försenad och vi blir lätt trötta och gör en massa fel. Om vi fick texterna oss tillsända på datormedium - kassett eller disk - skulle mycket vara vunnet.

Tänk om vi kunde publicera ett ordbehandlingsprogram i tidningen som kunde användas för just detta - skrivet i BASIC och utan andra krav på maskin än 16K och kassettbandspelare!

Nu kan vi det. Här gör vi det.

Grunden har lagts av Börje Häll, som efter ett telefonsamtal satte igång de små grå cellerna. Strax hade jag i brevlådan ett kvadratiskt kuvert med en programdiskett. Jag tog vid och kompletterade. Resultatet blev de två program som listas här bredvid - P:TEXTIN och P:TEXTUT.

Vi hoppas att programmen skall vara i stort sett självförklarande. Ändra bara inte på filernas uppläggning och textens dimensionering. Vi redigerar med mera kvalificerade ordbehandlingsprogram som kräver den filstruktur de här programmen ger.

Fortsätt nu att skriva bidrag till tidningen - eller börja om det är det som gäller - och skriv på kassett eller disk. Lägg märke till att Du inte behöver skrivare. Du kan kontrollera texten på skärmen, föra över den till kassetten och sedan skicka kassetten till oss.

Börje och jag - och alla andra som jobbar med tidningen ser fram emot att få börja redigera alla de kassetter som nu bör strömma in. Ös på!

Trevlig hacking!

PS Även om det är otroligt kan det ju finnas buggar i programmen. Avslöja dem då snabbt så skall vi skoningslöst döda dem! DS

```
100 REM P:TEXTIN
110 REM
120 REM Börje Häll
130 REM Vasavägen 101
140 REM 175 32  JÄRFÄLLA
150 REM
160 REM  Program för att mata in text, som ska sändas
    till 99'an
170 REM
180 REM  Dimensionera textvariabeln
190 DIM TEXT$(63)
200 REM  Bestäm skärmfärg och rensa skärmen
210 CALL SCREEN(8)
220 CALL CLEAR
230 REM Definiera svenska tecken
240 REM Stora bokstäver
250 CALL CHAR(91,"00280038447C4444")
260 CALL CHAR(92,"0028007C4444447C")
270 CALL CHAR(93,"00382838447C4444")
280 REM Små bokstäver
290 CALL CHAR(123,"0000280038447C44")
300 CALL CHAR(124,"000028007C44447C")
310 CALL CHAR(125,"0000382838447C44")
320 REM Fråga efter kassett eller flexskiva
330 INPUT "Kassett eller flexskiva? K/F":M$
340 IF (M$<>"K")+(M$<>"k")+(M$<>"F")+(M$<>"f")=-4 THE
    N 330
350 CALL CLEAR
360 REM   Initialisera räknaren för filer på flexskiv
    a
370 FILE=1
380 REM Instruktion
390 PRINT "Om kommatecken (,) ska matas"
400 PRINT "in måste texten börja och"
410 PRINT "sluta med citationstecken ""."
420 PRINT :"Avsluta inmatningen med"
430 PRINT "ENTER utan att skriva någon"
440 PRINT "text.": : : : :
450 REM Mata in texten
460 FOR I=0 TO 63
470 REM  Rad 480 får inte ändras
480 INPUT "Skriv in text (blankt för avsluta) ":A$
490 PRINT
500 REM  Om A$ är tom så ska inte mer text matas in
510 IF A$="" THEN 560
520 TEXT$(I)=A$
530 NEXT I
540 REM I är 1 mer än antal inmatade rader
550 REM därför måste I minskas med 1
560 I=I-1
570 REM Ska kassett eller flexskiva användas?
580 IF (M$<>"F")+(M$<>"f")=-2 THEN 650
590 REM   Öppna file på flexskiva. Rad 600 får inte ä
    ndras
600 OPEN #1:"DSK1.TEXT"&STR$(FILE),SEQUENTIAL,DISPLAY
     ,OUTPUT,FIXED 105
610 REM Öka räknaren
620 FILE=FILE+1
630 GOTO 670
640 REM  Öppna file på kassett. Rad 650 får inte ändr
    as
650 OPEN #1:"CS1",SEQUENTIAL,DISPLAY ,OUTPUT,FIXED 12
    8
660 REM Skriv texten på kassett/flexskiva
670 FOR J=0 TO I
680 PRINT #1:TEXT$(J)
690 NEXT J
700 REM   Skriv en tom sträng för att markera fileslu
    t och stäng filen
710 PRINT #1:""
720 CLOSE #1
730 REM  Fråga om mer text ska matas in
740 INPUT "Ska mer text matas in? J/N ":SVAR$
750 IF (SVAR$<>"J")+(SVAR$<>"j")+(SVAR$<>"N")+(SVAR$<
    >"n")=-4 THEN 740
760 IF (SVAR$="N")+(SVAR$="n")<0 THEN 790
770 CALL CLEAR
780 GOTO 390
790 CALL CLEAR
800 END
```

```
 10 REM P:TEXTUT
 20 REM Göran Nygren
 30 REM Uppingegränd 10
 40 REM 163 61  Spånga
 50 REM
 60 REM Program för att lösa/skriva ut text skapat me
    d P:TEXTIN
 70 REM Ta bort alla REMarks annars blir det MEMORY F
    ULL IN XXX !!
 80 REM Rensa skärmen, fråga om skrivare finns annars
     utskrift på skärmen
100 CALL CLEAR
110 INPUT "Har du printer/skrivare?    (J/N):":VAL$
120 IF (VAL$<>"J")+(VAL$<>"j")+(VAL$<>"N")+(VAL$<>"n"
    )=-4 THEN 110
130 IF (VAL$<>"J")+(VAL$<>"j")=-2 THEN 180
135 REM Val av printer och hastighet, öppna filen
140 INPUT "Printer specifikation:":PR$
```

```
150 OPEN #2:PR$,OUTPUT
155 REM Val av teckenbredd antal tecken per tum, för
    utskrift avpassad till PB
160 INPUT "     Spaltbredd (46/TPT el. 55/TPT):":RL
165 REM Nollställ vissa variabler och ge andra rätt v
    ärde, RL=radlängd, FL=flagga för fler filer, ANT=
    antal filer.
170 RL=RL-2
180 FL=0
190 ANT=1
195 REM Möjlighet att skriva ut flera filer i rad
200 INPUT "Skall flera filer skrivas ut? (J/N):":FLER
    $
210 IF (FLER$<>"J")+(FLER$<>"j")+(FLER$<>"N")+(FLER$<
    >"n")=-4 THEN 200
220 IF (FLER$<>"J")+(FLER$<>"j")=-2 THEN 250
225 REM FL ges värdet 1 om fler filer skall skrivas u
    t
230 FL=1
235 REM Variabel för att rätt antal filer läses in, F
    ILE=FILE+1, FILE>ANT
240 INPUT "Antal filer:":ANT
245 REM val av kassett eller fleskiva
250 INPUT "Kassett eller flexskiva?    (K/F):":M$
260 IF (M$<>"K")+(M$<>"k")+(M$<>"F")+(M$<>"f")=-4 THE
    N 250
265 REM Ge FILE rätt startvärde
270 FILE=1
275 REM Kontrollera att rätt antal filer läses, för a
    tt inte få ERROR och BREAKPOINT
280 IF FILE>ANT THEN 480
285 REM Skall flexskiva eller kassett användas
290 IF (M$<>"F")+(M$<>"f")=-2 THEN 320
295 REM Rad 300 får inte ändras
300 OPEN #1:"DSK1.TEXT"&STR$(FILE),SEQUENTIAL,DISPLAY
     ,INPUT ,FIXED 105
310 GOTO 330
315 REM Rad 320 får inte ändras
320 OPEN #1:"CS1",SEQUENTIAL,DISPLAY ,INPUT ,FIXED 12
    8
325 REM Starta räknaren för file och öka med ett
330 FILE=FILE+1
335 REM Start av inläsning och utskrift, ändra ej des
    sa rader 340-470
340 INPUT #1:TEXT$
350 IF TEXT$="" THEN 460
355 REM Utskrift skärm eller printer?
360 IF (VAL$="J")+(VAL$="j")=0 THEN 530
365 REM Rad 400-450, kontroll att rätt längd/textbred
    d skrives ut
370 IF LEN(TEXT$)>RL THEN 400
380 PRINT #2:TEXT$
390 GOTO 450
400 PRINT #2:SEG$(TEXT$,1,RL)
410 IF LEN(TEXT$)-RL<RL THEN 440
420 PRINT #2:SEG$(TEXT$,RL+1,RL):SEG$(TEXT$,(RL+RL)+1
    ,LEN(TEXT$))
430 GOTO 450
440 PRINT #2:SEG$(TEXT$,RL+1,LEN(TEXT$))
450 GOTO 340
460 CLOSE #1
465 REM Om FL=1 läs då in nästa fil
470 IF FL=1 THEN 280
475 REM Skall annan text läsas in eller skall program
    met stanna
480 INPUT "Ska mer text skrivas ut?    (J/N):":SVAR$
490 IF (SVAR$<>"J")+(SVAR$<>"j")+(SVAR$<>"N")+(SVAR$<
    >"n")=-4 THEN 480
500 IF (SVAR$="N")+(SVAR$="n")<0 THEN 520
510 GOTO 180
520 END
525 REM Utskrift av texten på skärmen
530 PRINT
540 PRINT TEXT$
550 CALL KEY(0,K,S)
560 IF S=0 THEN 550
570 GOTO 340
```

■

TEKNISKA Avdelningen

av Bengt Fahlgren

KONTAKTPROBLEM

Om du är i tagen att bygga nätaggregat till drive No 2, vill du kanske veta var du kan få tag på den vita 4-pol strömförsörjningskontakten ?

AMP Svenska AB OBS! Grossist
Avd AMPLIVERSAL
Box 512
S-175 26 JÄRFÄLLA

Ordertfn. 0758-10410

Hankontakt vit Nr: 1-480424-0
Kontaktstift till ovan. Nr: 163304-2

MONTERING AV DRIVAR

Monteringsskruvar till Floppydisk Fab. SHUGART,PERTEC kan vara svåra att få tag på i järnhandeln då det är tumgänga.
Generalagenten för PERTEC har dessa skruvar.

SCANDIA METRIC AB
Banvaktsvägen 20
Box 1307
171 25 SOLNA tel Vx 08/820400

MERA KONTAKTER

En annan kontakt som kan vara knepig att få tag på är PIO(Centronics) kontakten på RS232 kortet.

ITT Multikomponent OBS! Grossist
Ankdammsg.32
Box 1330, 171 26 SOLNA Vx 08/830020

Ordertfn.Komponenter etc Vx 08/835150

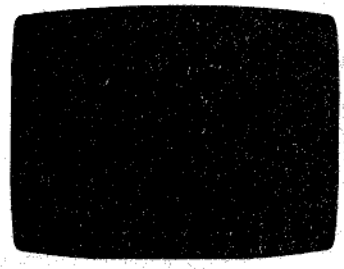
16-pol blockkontakt hona
Nr E 137202 ? Kolla detta!

Blijenburgh Electronics
DATABUTIKEN
Drottningatan 81
tfn 08/205054

ANSLUT TI-99/4A TILL FÄRGMONITOR/VIDEO

Det finns möjlighet att ansluta TI-99/4A till RGB via ett interface som heter PHA 2037.Denna burk liknar PHA 2036 men från interfacet går en kabel med en sk. SCART Video-kontakt.Pris ca 875:-
Tillgång osäker.

■



EMPTY SCREEN = TOM SKÄRM

BASICprogrammering

Lennart Lindberg

Basic-programmering -
inte bara att rada upp
satser utan 'SYNTAX ERROR'!

Den som skaffar en hemdator börjar antagligen i de allra flesta fall att utnyttja den för spel. Action-spel, adventure, intelligensspel av schacktyp - vilket det blir beror nog på läggning och på tillfälligheter.

Vägen vidare tror jag pekar åt två olika håll.
Det ena innebär fortsatt anskaffning av mera avancerad, färdig programvara. Det kan röra sig om andra spel och om nyttoprogram för ordbehandling, kalkylering, registerföring, datakommunikation etc.
Det andra innebär tillverkning av egna program.

För den som fortsätter att skaffa färdiga program är nog datorn i första hand ett redskap som utför ett arbete - för nytta (skriva, räkna, föra register etc) eller nöje (främst spel). Jag tror att huvuddelen av alla användare av hemdatorer tillhör och kommer att tillhöra den kategorin. Samma gäller dem som har eller kommer att ha persondatorer på sina arbeten. De kommer att ställa allt större krav emellertid på dessa färdiga programs prestationer, flexibilitet, snabbhet, enkelhet i användningen etc. De kommer också att lära sig att det kostar att skaffa sådana här program, men de kommer att betala om programmen svarar mot deras behov.

Detta ger den andra kategorin - de som ville tillverka program själva - en möjlighet. Ty de kommer - precis som nu sker - att kunna förena nytta med nöje och tänka ut, tillverka och sälja program till de andra. Samtidigt tror jag att drivkraften hos många programtillverkare från början inte alls är kommersiell. Vad de egentligen är ute efter är att förstå och behärska datorn. För dem är datorn en teknisk och intellektuell utmaning och/eller ett verktyg, men ett verktyg som de själva vill utveckla för egna ändamål.

En del av dem som programmerar själva blir 'hackers' - entusiaster som tillbringar timmar och åter timmar vid tangentbordet och tränger allt djupare in i maskinen via assembler och med hjälp av tjocka böcker med tekniska beskrivningar, ritningar etc. Deras språk är fyllt av ord och uttryck på en svensk-engelsk jargong som jag misstänker att många av dem ibland har svårt att förstå själva.

En annan del av dem som programmerar själva har en mera nyttobetonad inställning och har som huvudintresse att producera program som utför just det arbete som programmeraren av någon anledning vill att maskinen skall utföra - antingen det innebär nytta eller nöje.

Det är för dessa senare den här artikeln är skriven.

Jag tror att de som har den inställning jag beskrev har ett behov av att med en viss effektivitet kunna tillverka program. De behöver då för detta tillägna sig tekniker för att kunna

- analysera och definiera krav på programmet
- planera utformningen av programmet, först grovt och sedan i detalj
- tillverka programmet
- finslipa och optimera programmet
- prova ut programmet och korrigera fel

När ett program färdigställts dröjer det erfarenhetsmässigt inte alls länge innan programmeraren finner att det dyker upp nya krav på programmet. Han behöver då kunna ändra i programmet och göra tillägg och justeringar utan att göra om alltihopa från början.

Den som skaffar en hemdator och lär sig Basic genom att läsa i bruksanvisningen (eller Users Reference Guide, ofta förkortat kallad URG) hittar inget om dessa tekniker där. Han hittar säkert inget heller i den stora mängd böcker och tidningar som generöst bjuder på programlistningar som man kan skriva av. De flesta av dessa program är nämligen inte skrivna på ett sådant sätt att de svarar mot rimliga krav i de hänseenden jag nämnde.

Vad jag vill ha sagt med denna kritik är att det visserligen - naturligtvis - är nödvändigt att kunna programmeringsspråkets syntax för att kunna skriva program men att det på intet sätt är tillräckligt om man vill skriva bra program. URG lär bara ut språkets syntax. Det andra får man lära sig på annat sätt. Här följer några tips.

Det är tre stycken böcker jag vill framhålla :

- 'Starta med struktur' av Sven Burman och Bo Bäckman, Liber 1983, c:a 130:-kr
- 'Basic i praktiken' av Jean-Pierre Lamoitier, Pagina 1983 (i Frankrike 1980), c:a 150:- kr
- 'Some Useful Basic Subroutines' av Ian R. Sinclair, Newnes Technical Books 1983, c:a 120:-kr

Det överordnade budskapet är att program bör tillverkas i 'moduler' - dvs byggklossar, där varje byggkloss är en väl angränsad helhet med mycket väldefinierade gränser mot andra byggklossar och mot resten av programmet överhuvdtaget. De här modulerna får vanligen i ett Basic-program formen av sub-rutiner, som man begagnar med hjälp av 'GOSUB radnr', där radnr inleder en del av programmet som avslutas med RETURN.I X-Basic kan man också använda 'CALL XYZ', där XYZ är ett sub-program, som inleds med SUB och avslutas med SUBEND. Jag vill fö notera att subprogram-möjligheten i X-Basic är en avancerad finess, som är väl värd uppmärksamhet. Man kallar alltså inte med radnummer utan med namnet på sub-programmet.

En stor del av hemligheten med 'strukturerad programmering', som den teknik som beskrivs i 'Starta med struktur' kallas, är att kunna avgränsa sub-rutiner väl. Jag har här bredvid tagit med ett exempel ur boken på ett program som i dialogform beräknar moms på ett inmatat belopp för att visa hur programlistan kan se ut när man följer denna metod.

Metoden kallas också 'JSP' (=Jackson Systematic Programming) efter M A Jackson, som energiskt propagerat för metoden och framgångsrikt spritt den. Den innebär framför allt att man planerar sitt program väl innan man börjar koda. Det gäller att undvika vad som kallas 'spagetti-program' med långa, oöverskådliga sekvenser av satser där exekveringen styrs med GOTO-hopp. Program enligt JSP bör styras med GOSUB (motsvarande) och med IF-THEN-ELSE (motsvarande). Men allt det där står i boken, som är en ren lärobok med mycket pedagogisk framställning och gott om övningsexempel. Alla exempel är skrivna för ABC 80/800 men översättningen till TI Basic tycker jag inte är så svår.

Ett utmärkt komplement till 'Starta med struktur' är 'Some Useful Basic Subroutines', vilket kanske är lätt att förstå. Boken beskriver helt enkelt ett antal sub-rutiner, kodade i Basic. Dem kan man plocka in i sitt strukturerade program och använda med GOSUB eller med CALL XYZ, om man använder X-Basic. Nu är de i boken inte skrivna i TI Basic utan Microsoft Basic, vilket kräver ett visst översättningarbete. Det bör väl dock inte vålla oöverkomliga svårigheter. Den som skriver av program från t ex 'Basic Computer Games' av David H. Ahl har exakt samma problem.

Sub-rutinerna i Sinclairs bok handlar om

- skärmhantering (t ex en blinkande rubrik)
- inmatning (t ex siffervalidering, J/N-svar)
- sökning och sortering
- matris-hantering
- grafik (t ex understrykning, rörlig figur)
- kassett-rutiner

Boken är på engelska.

▶ 15

Programmera i FORTH ?

Lennart Lindberg

Hur lär man sig att programmera FORTH?

Den första betydelsefulla frågan blir nog
- varför skall man lära sig att programmera FORTH?
Mitt svar blir
- därför att det är ett utmärkt sätt att kunna nästan maximalt utnyttja de inneboende möjligheterna i en mikrodator som TI 99/4A. Man får program som
    * tar mycket liten plats i minnet
    * är mycket snabba
- därför att FORTH är ett okonventionellt språk som gör det möjligt att skapa alldeles egna instruktioner (kallas ord i FORTH) och t o m skapa ord som skapar andra ord! Man kan alltså själv välja programmeringsnivå - från manipulation av enskilda bitar i vissa bytes till maximalt kraftfulla makro-instruktioner.
- sammanfattningvis därför att FORTH är ett mycket intressant alternativ och komplement till samt ett stimulerande utvecklingssteg efter Basic och X-Basic.

Nästa fråga misstänker jag kan vara
- vad behöver man för utrustning till 99-an för FORTH?
Svaret blir
- basmaskin plus minnesexpansion plus ett lagringsmedium. Bäst är att ha diskar men föreningens FORTH kan användas även med kassett. Dock har FORTH en grundfilosofi som baseras på disk. Till denna hårdvara behöver man sedan en programmodul. TI Forth kräver Editor/Assembler-modulen medan föreningens FORTH dessutom fungerar med X-Basic eller Mini Memory. För att det skall bli något av det här sedan måste man ha ett FORTH kärnprogram på disk eller kassett med tillhörande dokumentation. Det är detta senaste som säljs genom föreningen. Det finns två varianter
    * föreningens FORTH, som utgörs av en naken TI FORTH version 2.0, skickligt och i hög grad förbättrad i Sverige av Björn Gustavsson (bl a så att den kan köras med kassett och blivit tillförd såväl dekompilator som disassemblator, med vars hjälp man alltså från kompilerad/assemblerad kod kan ta fram den använda källkoden) och försedd med enkel, delvis ofullständig dokumentation (som alltså ställer krav på användaren)
    * TI FORTH version 3.0 med flyttalsmöjligheter (FORTH är i kärnan ett språk för heltalsräkning), filhantering, grafik i högupplösning, vokabulär för assemblerprogrammering, testhjälpmedel mm och god dokumentation

Det är inte nödvändigt, men man har god nytta av en skrivare.

Den som nu är motiverad för FORTH och har eller tänker skaffa de grejor som behövs kan fortsätta att läsa här.

Hur lär man sig då att programmera FORTH?

Jag har lärt mig genom att växelvis
- läsa i böcker
- öva vid maskinen
Jag har läst ganska mycket och jag har läst samma text många gånger. Jag tror metoden är allmänt sett bra. Jag tror också att ett mycket lämpligt komplement är att föra utvecklande och klargörande samtal med likasinnade om olika problem man stöter på i böckerna och vid tangentbordet.

De intressanta böckerna är på engelska. På svenska har jag bara sett en bok i bokhandeln och den lockade inte just då i alla fall.

Den bokhandel där jag hittat det bästa sortimentet av FORTH-litteratur är Esseltes bokhandel 'Bokhuset' på Gamla Brogatan, vid PUB, i Stockholm. Den förefaller f ö vara mest välsorterad överhuvudtaget på böcker som kan vara av intresse för 99-hackers.

Grundboken framför andra är nog 'Starting FORTH' av Leo Brodie. Den köps dock billigast hos ABC-klubben, som skaffat ett lager som man säljer av även till icke-medlemmar. Kontakta Claes Schibler, som är medlem i ABC-klubben. Boken är i mitt tycke utmärkt - välskriven, rolig att läsa inte minst genom Brodies egna teckningar, pedagogiskt lyckad, goda övningsuppgifter. Man kan komma långt med hjälp av den. Ett skäl att skaffa just den är också att såväl TI FORTH som föreningens FORTH i dokumentationen har hänvisningar till Brodies bok och kommenterar skillnaderna mellan den FORTH-variant som Brodie beskriver (kallad polyFORTH) och den som implementeras av TI (kallad FIG FORTH). De där varianterna förstår man när man kommit en bit på väg med FORTH.

En ganska annorlunda bok är 'Exploring FORTH' av Owen Bishop. Leo Brodie är amerikan och var när han skrev boken professionell FORTH-specialist i ett speciellt företag som lever på att utveckla FORTH. Det präglar hans bok. Owen Bishop är en flitig ADB-journalist i England och har skrivit ett fyrtiotal böcker om ADB ur alla möjliga synvinklar - nu även ur FORTH-vinkeln, så att säga. Detta präglar hans bok.

Bishop kommer inte alls lika långt och djupt som Brodie. Han närmar sig däremot FORTH på ett enklare och mera självklart sätt och presenterar ganska fort i boken program som skapar rörelse på skärmen. Han är helt enkelt inte så renlärigt frälst som Brodie är utan FORTH blir bara ytterligare ett nytt språk som kan användas till att få resultat. Han kommer t ex tidigt i boken och på ett enkelt sätt in på såväl stränghantering vid input - som ju är så enkelt i Basic men tarvar litet mera eftertanke i FORTH tills man utformat de ord man själv vill ha för stränghanteringen - och spel med grafik. Han visar också enkla sorteringsrutiner och - vilket jag tyckte var mycket lärorikt - ger tips om hur man undersöker ordlistan i FORTH för att komma underfund med hur de ord man skapar ser ut i lagrat skick. Hans litet annorlunda vinklade framställning gör också att jag upplevde det som stimulerande att växla emellan de två böckerna. En speciell knorr tycker jag är att Bishop gärna diskuterar FORTH på maskiner som är intressanta i England men som för mig framstår som litet exotiska - BBC Microcomputer och Jupiter Ace t ex.

För den som känner sig lockad av det där med grafik vill jag bara påpeka att just grafiken är mycket maskinspecifik. Det kan alltså krävas en del maskinkunnande för att överföra grafiken från Bishops bok till 99-an. TI FORTH version 3.0 har ord för 99-ans grafik.

Vill man gå ett steg till i sina FORTH-studier kan man skaffa något som heter 'FORTH Fundamentals'. Det är en bok i två volymer, skriven av C. Kevin McCabe, som är en advokat i USA med ingenjörsbakgrund i flygindustrin, där han tydligen sysslade med datorer. Han ägnar sig nu mycket åt FORTH-språket och dess utveckling och har i den här boken sammanställt en klar, väldokumenterad, noggrann och detaljerad beskrivning av FORTH. Det finns inga teckningar och inga övningsexempel utan bara tydlig, välskriven text som avslöjar de flesta av FORTH's hemligheter. Fotnötter och litteraturhänvisningar förekommer rikligt. Han beskriver FIG FORTH - han är själv medlem i FORTH Interest Group, som FIG förkortar - men också FORTH-79, som är ett standariseringsförsök, och de utökningar i FIG FORTH som gjorts för den mycket använda mikroprocessorn Z80. Den första volymen - 240 sidor - beskriver FORTH som ide och språk ur alla intressanta aspekter men med en annan uppläggning än Brodie, vilket ökar inlärningsmöjligheterna när man läser båda. Den andra volymen är en alfabetisk förteckning med utmärkta förklaringar av alla de FORTH-ord McCabe tagit upp i första volymen.Det är en förträfflig uppslagsbok. McCabe har en överlägsen teknik när det gäller att ställa upp förklaringar till FORTH-orden så att deras funktion blir begriplig. Jag har därigenom kunnat ta delar av ord han förklarat och använda som förlaga för något problem jag velat lösa.

Tips och tricks

WARNING!!!!!

Man kan frambringa FCTN = (QUIT) genom att samtidigt trycka ned följande tangenter...

<FCTN>,<SPACE>,<M>,<,>,<K>

eller
<FCTN>,<SPACE>,<,>,<.>,<L>

eller
<FCTN>,<SPACE>,<N>,<M>,<J>

Tyvärr är detta byggt på erfarenhet.

X-Basic Bug

Mata in följande program i X-Basic:

```
  100 ACCEPT SIZE(248):A$
  110 ACCEPT SIZE(248):B$
 (120 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX)
```

Rad 120 frivillig förklaras nedan.

Kör programmet. Mata in några tecken, tryck <ENTER>. Edge-tecknet(ASCII(31)) blir omdefinierat till ("8080808080800000"). Man ser detta p.g.a. ASCII(31) är placerat i kolumn 1,2,31,32.

Mata in några fler tecken. Datorn skriver ut ett antal <SPACE>. Tryck på <ENTER>. Ett konstigt mönster uppstår. Tecken-karaktärerna har blivit omdefinerade och några bokstäver syns ej längre. Minnet är fullt!

Man kan se spår av markören längst ner till vänster. Om du matade in rad 120 så ta bort den <1><2><0><ENTER> så att du får några bytes att arbeta med.

Mata in följande kommando:

```
FOR I=30 TO 143 :: CALL HCHAR(1,1,I,768) :: NEXT I
```

Tryck <ENTER>

Datorn visar nu ett antal skärmsidor med de nya karaktärer som finns i minnet.Genom att framkalla MEMORY FULL kan man återfå Standard-karaktärerna på skärmen med programmet kvar i minnet.

Mata in följande programrad:

```
130 GOSUB 130
```

Tryck <ENTER>

Mata in följande kommando:

```
RUN 130
```

Tryck <ENTER>

Nu ser "troligen" skärmen ut som vanligt med två undantag.

```
(1) ASCII(31)=("8080808080800000")
(2) ASCII(32)=("8080808080808080").
```

CALL CHARSET FUNGERAR EJ!

Ibland låser sig processorn och då är det bara att stänga av datorn. Vissa villkor:

SIZE>=228
ERASE ALL får ej ingå i programraden.

TIPS I EXTENDED BASIC !

Av Johan Pihlsgård

med viss hjälp av Micael Dahlquist

Följande program illustrerar en mycket oväntad effekt i EXTENDED BASIC. Fenomenet uppstår på samma sätt som 'film' (?).

```
100 CALL CLEAR :: DISPLAY AT(12,12):"GAME OVER"
110 CALL SCREEN (2) :: CALL SCREEN (11) :: GOTO 110
```

(OBS) Detta program kan ej modifieras till TI BASIC på grund av att TI BASIC är alldeles för långsamt.

Och nu samma rutin som ovan, men i ASSEMBLER !

```
   LI   R0, >8043
   LI   R1, >1000
   MOVB R0,@>8C02
   SWPB R0
   MOVB R0,@>8C02
L1 MOVB R1,@>8C00
   INC  R0
   CI   R0, >43A0
   JL   L1
   LI   R0, >1187
   MOVB R0,@>8C02
   SWPB R0
   MOVB R0,@>8C02
   BL   @S1
   LI   R0, >1A87
   MOVB R0,@>8C02
   BL   @S1
   JMP  $-36
S1 LI   R0, >6C
L2 DEC  R0
   JNE  L2
   LIMI 2
   LIMI 0
   B    *R11
```

Programmet kan köras från exempelvis MINI MEMORY (glöm ej REF/DEF table) och om du kallar på rutinen från MINI MEMORY, så måste CALL SCREEN(1) skrivas innan du kallar på rutinen.

EX.

```
10 CALL SCREEN(1)
20 CALL LINK("namn")
```

Mycket nöje !

2.

(Saxat ur TI*MES, Nr. 5/1984 sidan 22-23, den engelska användarföreningen, av Göran Nygren)
Citat: "I det sista numret nämnde jag ett TI-häfte som beskriver extra CALL subrutiner, som går att använda i TI-BASIC när PRK-Modulen (Personal Record Keeping) och Statistik-Modulen sitter inpluggade i cartridgesloten, det häftet finns fortfarande att köpa". Vi skall försöka få tag på ett exemplar, hoppas vi!

Överföring av PRK filer till TI-WRITER kompatibla filer.

Starta med att stoppa in PRK-Modulen och välj sedan TI-BASIC

1. Är flexskiveenheten inkopplad? Använd CALL FILES(1) och därefter NEW.

2. Använd koden CALL P(10900) och därefter NEW.

Nu är konsolminnet preparerat, initierat.

Restriktioner:

Flexskiveenheten och dess system tar upp en del minne, du kanske får ta bort några poster (records) från PRK filerna för att kunna överföras,om PRK filerna är fulla kan arbetsminnet bli fullt.

```
100 CALL L("DSK1.PRKFILE",Y) * Ladda PRK fil
110 IF Y=0 THEN 310 * Är filen inläst?
120 OPEN #1:"DSK1.TIW_FILE",DISPLAY,VARIABLE 80 *
```

öppnar fil för TI-WRITER, öppna inte filen innan CALL L!

```
130 CALL H(1,5,0,F) * Hur många fält i varje post?
140 PRINT "FIELDS";F
150 CALL H(1,6,0,R) * Hur många poster (sidor) ?
160 PRINT "RECORDS";R
170 FOR I=1 TO R * Nu gå igenom varje post
180 FOR T=1 TO F * Och varje fält i tur och ordning
190 CALL H(1,10,T,TP) * Är fältet numeriskt eller
alphanumerisk?
200 IF TP=1 THEN 240
210 CALL G(1,I,T,Z,RD) * För numerisk data plocka data
i post I, fält T
220 PRINT #1:RD * Och skriv ut det på flexskivan
230 GOTO 270
240 CALL G(1,I,T,Z,RD$) * För alphanumerisk data
250 PRINT I;T;RD$
260 PRINT #1:RD$
270 NEXT T
280 PRINT #1:"  ":: * Skriv mellanrum (spacing) vilket
fordras
290 NEXT I
300 END
310 PRINT "NOT LOADED"
```

Citat igen, "Om det fordras, du anser det nödvändigt, kan alpha och numerisk data förkortas till lämpligt format innan det skrives ut till flexskivan. Läs varje utvalt fält, avkorta och sedan skriv ut.
Varje postutskrift skall anses som en TI-WRITER rad.
Användande av pågående, oavgjord postutskrift (Pending outputs) bör undvikas samma gäller användandet av semikolon som utskrifts skiljare beroende på komplexiteten av DISPLAY formsat. Jag misstänker att det är möjligt att göra mycket intressanta saker med dessa moduler. De möjliggör för dig att spara data i binärt bild format (IMAGE FORMAT), mycket snabbare och använda mindre lagringsminne. Det andra sättet är att använda MINI-MEMORY likt en elektronisk flexenhet och dumpa innehållet i MiniMem på bildbinärt, memory image , sätt genom Easy Bug's 'S'."

Finns det någon därute som kanske har boken redan isåfall är vi i redaktionen intresserade att få titta i den och kanske ordna till det så att vi får kopiera den, med tillåtelse av författaren. Annars om ni har mer vetskap om moduler och länkmöjligheter med TI-BASIC, hör av er. Jag tror att många har ett stort intresse av dessa "nya" möjligheter. Om du fick uppslag av denna artikel och kommer på någonting nytt. Håll inte på det! Dela med dig av din kunskap.

Det får väl räcka för den här gången. Vi måste få spara en del go'bitar till ett annat nummer och du bör få tid att arbeta och studera andra artiklar i detta nummer, eller hur?

Hälsningar från REDAKTIONEN ■

12 ▶

BASICprogrammering

'Basic i praktiken' är en bok för den mera avancerade programmeraren. Den ligger på en högre teoretisk nivå och tar framför allt sina exempel från matematik, statistik och operationsanalys, vilket nog kan avskräcka många. För den som inte avskräcks är boken intressant och ger många tips i programmeringskonsten - mest då förstås inom de områden jag nämnde ovan. Alla exempel gäller ABC80/800.

Sluligen: för den som inte vill översätta från andra Basic-dialekter har jag ett tips: 'Programs for the TI Home Computer' av Steve Davis, Prentice-hall 1983, c:a 300:-kr. Närmare 50 program av alla möjliga sorter att skriva av. Tyvärr tycks författaren inte ha läst sin Michael Jackson. De flesta programmen fungerar på en enkel 16K-maskin.

LISTNING AV PROGRAM FöR MOMS-BERäKNING.

Programmet är skrivet i TI Basic enligt JSP-metod.
Det är hämtat ur 'Starta med struktur'.
Listningen illustrerar enbart modul-tekniken och
hur den återspeglas i programmets uppläggning.
Lägg märke till hur REM-satserna ökar läsbarheten.

```
100 REM    ************************************MOMS
110 REM    ***INITIERING***
120 MOMSPROCENT=.2346
130 GOSUB 200
140 REM  200=INMATNING
150 GOSUB 300
160 REM  300=BERAEKNING
170 GOSUB 400
180 REM  400=UTSKRIFT
190 END
200 REM    ********************************INMATNING
210 INPUT "BELOPP         ";BELOPP
220 RETURN
300 REM    ********************************BERAEKNING
310 MOMS=MOMSPROCENT*BELOPP
320 TOTAL=BELOPP+MOMS
330 RETURN
400 REM    ********************************UTSKRIFT
410 PRINT "MOMS=          ";MOMS
420 PRINT "BELOPP+MOMS= ";TOTAL
430 RETURN
```
■

13 ▶

Programmera i FORTH ?

'Starting Forth' är utgiven på Prentice-Hall 1981.
'FORTH Fundamentals' är utgiven på dilithium Press 1983.
'Exploring FORTH' är utgiven på Granada 1984.

'Exploring FORTH' är billigast med ett bokhandelspris på c:a 130:-kr. 'FORTH Fundamentals' kostar c:a 200:-kr per volym medan 'Starting FORTH' i bokhandeln kostar över 300:- och i ABC-klubben kring 200:-kr. Så det kan visserligen vara sant att kunskap är en lätt börda att bära men kostnaden för att skaffa den kan vara tung.

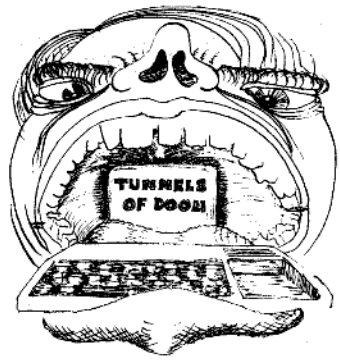
Jag har sett åtminstone tre böcker till om FORTH på svenska bokhandelsdiskar - t ex två engelska: 'The Complete FORTH' av Alan Winfield, Sigma Technical Press 1984, och 'FORTH for Micros' av Steve Oakley,Newnes Technical Books 1984. Dem har jag bara bläddrat i. En jag tänker läsa däremot är Leo Brodies nya 'Thinking FORTH', Prentice Hall 1984. Den verkar mycket spännande!

■

MER

ORDLISTA

till



ORDLISTA TILL TUNNELS OF DOOM
av Ung man, överlämnat på medlemsträffen den 15 september 1984.

A

ACID SPRAY	*	Syrabesprutning, minskar spelarens rustningsskydd.
ARCH DEVIL		Ärkedjävul.

C

CHRUSHING CHOMP		Krasande jättetugga, ökar spelarens skador.
CLOAK OF HIDING		Osynlighetsmantel, den bästa skölden.
CONSUMING BEAM		Förtärande stråle, minskar monstrens träffpoäng.
CURSE	*	Förbannelse, minskar spelarens tur, vapenbonus och rustningsskydd.

D

DISINTEGRATE		Förstörelse, minskar alla monstrens träffpoäng.

E

EARTH QUAKE		Jordbävning, ökar alla spelarens skador.
ELEMENTAL		Elementar ande (eld, luft, jord och vatten).
ELVEN BLADE	*	Alvblad.
EVIL MANE	*	Elak man.
EXPLOSION		Explosion, ökar alla spelarens skador.

F

FUNGUS		Rötsvamp, minskar spelarens träffpoäng.

G

GHOUL		Gravskändare.
GNOLL	*	Blandning av troll och gnom.

K

KOBOLD		Spöke som lever i gruvor.

L

LAND SHARK		Landhaj.

M

METAL MOULD		Metallmögel, minskar spelarens rustningsskydd.
MINOTAUR	*	Minotaur, hälften människa hälften tjur.

O

ORGE	*	Elak man, ofta jätte.

P

PHASE SHIFT		Fas för skjutning ökar monstrets rustningsskydd.
PIT FIEND		Avgrundsdjävul.
PITCHFORK		Treudd, minskar spelarens träffpoäng.
PRANK-BUS		Minskar spelarens tur.

Q

QUARREL	*	Armborstpil, även lod, skjuts med armborst men är små blykulor.

R

RAGE		Raseri ökar monstrens attackklass.
REGENERATION		Självläkning, ökar monstrens träffpoäng.

S

SWORD KING		Svärdkung.

T

TOUNGE OF FLAME		Eldtunga, flammande tunga, ökar spelarens skador.
TRANSMUTE	*	Förvandling, minskar gruppens matransoner.

V

VULNERABILITY		Sårbarhet, minskar monstrens magiska motstånd.

W

WARRIOR MAIL		Riddarrustning.
WILL O WISP		Lyktgubbe, irrbloss.

Med * märkt ord finns även beskrivet i NN 83-4/5

■

EMPTY SCREEN = TOM SKÄRM

EMPTY SCREEN, VADDÅDÅ?????????
KLASSE TYCKER ENLIGT FÖLJANDE :

LITET MED TEXT GER UTSPRIDNING AV TEXTEN BÅDE I SIDLED OCH I HÖJDLED (10 TECKEN / TUM OCH RADHÖJD 1½ ELLER TREKUGG)
GÄLLER EJ 59-SIDORNA SOM SKA SE UT SÅ, DET HAR DE ALLTID GJORT.
MEN NI SOM ÄR 99-ÄGARE-LÄSARE OCH GNÄLLER SKÄMS PÅ ER!!!!!!!!
KOM **IGEN MED MER MATERIAL TILL TIDNINGEN NU !!!!!!!!!!!**

A DUUUUUUUUU !

VI INOM REDAKTION OCH STYRELSE I FÖRENINGEN **PROGRAMBITEN** ÄR **SYNNERLIGEN GLADA ÖVER MATERIAL BÅDE FRÅN 99-FOLK SOM 59-FOLK !!!!**

TI 59

Motstånds färgkoder

Martin Harnevie
Mölndal

Bidrag till programbiten

Följande program anser jag vara lika enkelt som användbart.
Genom att mata in ett motstånds färgmarkeringar såsom följder av tre tangentnedtryckningar får man motståndets resistans.
Man slipper trasslet med färgkoder.
På mitt eget magnetkort har jag dessutom målat rutorna i respektive färg:

med tippex för vitt
med blyerts för grått
och med tusch för övriga färger.

För silver och guld använder jag samma labels som för grått respektive vitt, vilket kan kommas ihåg genom att anteckna detta ovanför dessa rutor.

Instruktioner

1. Initiera med SBR CLR (tar bort Fix och Eng, samt nollställer flaggor).

2. Tryck in färgkod såsom följder av tre tangentnedtryckningar. Värdet visas. Nu kan 2. upprepas utan ny initiering.

Martin Harnevie
Mölndal

För att erhålla kod 46 på Rad 010 gör man RCL 46 BST BST DEL SST.

	100. 00
	1. 03
	2.2 03
	9.9 00
	0. 00
	1. 00
	880. -03
	9.9 00
	4.4 00
	440. -03

BBB	110. 00
BBE	110. 03
EEC´	440. 06
E´E´E´	9.9 00
D´D´D´	880. -03
ED´E	480. 03
ED´E´	4.8 00
ED´E	480. 03

1	PROGRAMMERARE: HARNEVIE			
Färgkoder motstånd			SILVER	GULD
GRÖN	BLÅ	VIOLETT	GRÅ	VIT
SVART	BRUN	RÖD	ORANGE	GUL

000	91	R/S		061	76	LBL	
001	76	LBL		062	14	D	
002	69	OP		063	24	CE	
003	87	IFF		064	03	3	
004	02	02		065	61	GTO	
005	88	DMS		066	69	OP	
006	87	IFF		067	76	LBL	
007	01	01		068	15	E	
008	87	IFF		069	24	CE	
009	53	(		070	04	4	
010	46	INS		071	61	GTO	
011	65	×		072	69	OP	
012	01	1		073	76	LBL	
013	00	0		074	16	A'	
014	54	)		075	24	CE	
015	42	STO		076	05	5	
016	01	01		077	61	GTO	
017	01	1		078	69	OP	
018	94	+/-		079	76	LBL	
019	86	STF		080	17	B'	
020	01	01		081	24	CE	
021	22	INV		082	06	6	
022	57	ENG		083	61	GTO	
023	92	RTN		084	69	OP	
024	76	LBL		085	76	LBL	
025	87	IFF		086	18	C'	
026	44	SUM		087	24	CE	
027	01	01		088	07	7	
028	02	2		089	61	GTO	
029	94	+/-		090	69	OP	
030	86	STF		091	76	LBL	
031	02	02		092	19	D'	
032	92	RTN		093	24	CE	
033	76	LBL		094	02	2	
034	88	DMS		095	94	+/-	
035	22	INV		096	87	IFF	
036	28	LOG		097	02	02	
037	49	PRD		098	69	OP	
038	01	01		099	08	8	
039	43	RCL		100	61	GTO	
040	01	01		101	69	OP	
041	57	ENG		102	76	LBL	
042	81	RST		103	10	E'	
043	76	LBL		104	24	CE	
044	11	A		105	01	1	
045	24	CE		106	94	+/-	
046	00	0		107	87	IFF	
047	61	GTO		108	02	02	
048	69	OP		109	69	OP	
049	76	LBL		110	09	9	
050	12	B		111	61	GTO	
051	24	CE		112	69	OP	
052	01	1		113	76	LBL	
053	61	GTO		114	25	CLR	
054	69	OP		115	22	INV	
055	76	LBL		116	57	ENG	
056	13	C		117	22	INV	
057	24	CE		118	58	FIX	
058	02	2		119	25	CLR	
059	61	GTO		120	81	RST	
060	69	OP					

TI 59

Trippelpunkten

Kommentar till följande program.

Några dagar innan programmet trippelpunkter ramlade in i brevlådan hade jag fått ett brev från Raul där han berättade litet om sig själv.
Raul slutar det brevet med:
Jag skall skicka ett sataniskt problem med lösning som går ut på att finna den punkt på jorden (φ,λ) som har samma avstånd till tre punkter (städer).
Detta är aktuellt i a) Nordsatprojektet
b) Polarisraketbaser.
Det senare kan hjälpa vid U-båtsjakt.
(Noggrannhet 158,2m dvs 0,1').

När programmet kördes erhölls körtid 11 minuter på den stora triangeln - med en etta i steg 131.
När jag gjorde INS ett steg 132 00 - vilket går bra då ingen direktaddressering förekommer - erhölls mycket kortare körtid.

Detta påpekar Raul redan i texten till programmet och även i senare brev till mig.

Pargas, 25/3-84

Hej Claes!

Om du vill kan du pröva på att lösa detta till synes lätta till "skolmatten hörande" problem att finna "tyngdpunkten till en sfärisk triangel" - i det följande kallad trippelpunkten. Av mina anteckningar att döma skedde renskrivningen den 3/4-83.
Dock minns jag än hur hopplöst det ett tag verkade.
De flesta matematiska problem brukar ganska snart konvergera mot en lösning efter det man uppskrivit de grundläggande kraven i formelform. Men i det här fallet blev ekvationen så lång att pappret inte ville räcka till. Dessutom var den transcendent så man i normalt fall skulle övergått till trivialiteter. Jag skall inte trötta ut dig med de tråkiga ekvationssystem som först måste lösas för att få följande:

$$\left(\frac{\cos\varphi_1}{\sin\varphi_1 - \sin\varphi_2} + \frac{\cos\varphi_1}{\sin\varphi_3 - \sin\varphi_1}\right)\cos(\lambda - \lambda_1) +$$

$$+ \frac{\cos\varphi_2\cos(\lambda - \lambda_2)}{\sin\varphi_2 - \sin\varphi_1} + \frac{\cos\varphi_3\cos(\lambda - \lambda_3)}{\sin\varphi_1 - \sin\varphi_3}$$

Ovanstående monster löses först numeriskt med Pgm 08 med avseende på longituden (λ).

Sedan ger

$$\operatorname{tg}\varphi = \frac{\cos\varphi_2\cos(\lambda - \lambda_2) - \cos\varphi_1\cos(\lambda - \lambda_1)}{\sin\varphi_1 - \sin\varphi_2}$$

latituden (φ) och de tre avståndsformlerna som låg till grund för ovanstående ger avstånden ℓ_ν tillika med kontroll att allt blev rätt.
Vid input envisas jag fortfarande med att anse att E=+ emedan jag bor öster om Greenwich. Amerikanerna är självfallet av en annan åsikt som jag väl påpekat tidigare.

PROGRAMBESKRIVNING:

! Mastermodulen ska sitta i. Pgm 08 används !
Minnena R1 - R8 samt de användardefinierade Labels A - E får ej ändras på grund av Pgm 08.
Om du tycker det är för mycket flaggviftande för att få in triangelns tre hörnpunktskoordinater (DDD.MMss) är det lätt via en INIT jämte STO IND eliminera flaggorna.

Programstegen	000 - 065	är inmatning
"	066 - 120	är koefficienter
	131	får ändras till 1 - 100 om exekveringen känns lång (normaltid 3 min)
	150	sökta longituden utskrives
	194	sökta latituden "
subrutin	D'	avståndsformeln
Label	A'	ekvationen som används i Pgm 08.

Observera speciellt att man måste vara försiktig kring datumgränsen och vidtaga lämplig transformation av longituden i detta program.
"Se de två exemplen.
Tokio (N36 E139) matas in (36,-41)
Melbourne (S38 E145) blir (-38,-35)
och Los Angeles (N35 W118) blir (35,62).
Latituden blir härvid omedelbart rätt men longituden E 10^{o} 14,5' blir W 169^{o} 45,5'. (Se TRACEningen)

1	PROGRAMMERARE: HOFMAN			2
anv	LAT	LONG	anv	anv
anv	anv	anv	anv	anv

TI 59

```
001   17 B'
014   18 C'
031   22 INV
041   24 CE
051   23 LNX
059   25 CLR
223   19 D'
264   16 A'
```

```
000   76 LBL
001   17 B'
002   99 PRT
003   88 DMS
004   87 IFF
005   01  01
006   22 INV
007   87 IFF
008   03  03
009   23 LNX
010   42 STO
011   11  11
012   91 R/S
013   76 LBL
014   18 C'
015   99 PRT
016   88 DMS
017   87 IFF
018   02  02
019   24 CE
020   87 IFF
021   04  04
022   25 CLR
023   42 STO
024   12  12
025   86 STF
026   01  01
027   86 STF
028   02  02
029   91 R/S
030   76 LBL
031   22 INV
032   42 STO
033   13  13
034   22 INV
035   86 STF
036   01  01
037   86 STF
038   03  03
039   91 R/S
040   76 LBL
041   24 CE
042   42 STO
043   14  14
044   22 INV
045   86 STF
046   02  02
047   86 STF
048   04  04
049   91 R/S
050   76 LBL
051   23 LNX
052   42 STO
053   15  15
054   22 INV
055   86 STF
056   03  03
057   91 R/S
058   76 LBL
059   25 CLR
060   42 STO
061   16  16
062   98 ADV
063   22 INV
064   86 STF
065   04  04
066   43 RCL
067   11  11
068   38 SIN
069   75  -
070   43 RCL
071   13  13
072   38 SIN
073   95  =
074   42 STO
075   17  17
076   43 RCL
077   11  11
078   38 SIN
079   75  -
080   43 RCL
081   15  15
082   38 SIN
083   95  =
084   42 STO
085   18  18
086   43 RCL
087   11  11
088   39 COS
089   55  ÷
090   43 RCL
091   17  17
092   75  -
093   43 RCL
094   11  11
095   39 COS
096   55  ÷
097   43 RCL
098   18  18
099   95  =
100   42 STO
101   19  19
102   43 RCL
103   13  13
104   39 COS
105   94 +/-
106   55  ÷
107   43 RCL
108   17  17
109   95  =
110   42 STO
111   20  20
112   43 RCL
113   15  15
114   39 COS
115   55  ÷
116   43 RCL
117   18  18
118   95  =
119   42 STO
120   21  21
121   43 RCL
122   12  12
123   36 PGM
124   08  08
125   11  A
126   43 RCL
127   16  16
128   36 PGM
129   08  08
130   12  B
131   01  1
132   36 PGM
133   08  08
134   13  C
135   93  .
136   00  0
137   00  0
138   00  0
139   08  8
140   36 PGM
141   08  08
142   14  D
143   36 PGM
144   08  08
145   15  E
146   22 INV
147   88 DMS
148   58 FIX
149   04  04
150   99 PRT
151   22 INV
152   58 FIX
153   88 DMS
154   42 STO
155   22  22
156   43 RCL
157   13  13
158   39 COS
159   65  ×
160   53  (
161   43 RCL
162   22  22
163   75  -
164   43 RCL
165   14  14
166   54  )
167   39 COS
168   75  -
169   43 RCL
170   11  11
171   39 COS
172   65  ×
173   53  (
174   43 RCL
175   22  22
176   75  -
177   43 RCL
178   12  12
179   54  )
180   39 COS
181   95  =
182   55  ÷
183   43 RCL
184   17  17
185   95  =
186   22 INV
187   30 TAN
188   42 STO
189   23  23
190   22 INV
191   88 DMS
192   58 FIX
193   04  04
194   99 PRT
195   98 ADV
196   71 SBR
197   19 D'
198   43 RCL
199   13  13
200   42 STO
201   11  11
202   43 RCL
203   14  14
204   42 STO
205   12  12
206   71 SBR
207   19 D'
208   43 RCL
209   15  15
210   42 STO
211   11  11
212   43 RCL
213   16  16
214   42 STO
215   12  12
216   71 SBR
217   19 D'
218   98 ADV
219   98 ADV
220   98 ADV
221   91 R/S
222   76 LBL
223   19 D'
224   53  (
225   53  (
226   53  (
227   43 RCL
228   23  23
229   38 SIN
230   65  ×
231   43 RCL
232   11  11
233   38 SIN
234   85  +
235   43 RCL
236   23  23
237   39 COS
238   65  ×
239   43 RCL
240   11  11
241   39 COS
242   65  ×
243   53  (
244   43 RCL
245   22  22
246   75  -
247   43 RCL
248   12  12
249   54  )
250   39 COS
251   54  )
252   22 INV
253   39 COS
254   65  ×
255   06  6
256   00  0
257   54  )
258   58 FIX
259   01  01
260   99 PRT
261   54  )
262   92 RTN
263   76 LBL
264   16 A'
265   53  (
266   42 STO
267   10  10
268   53  (
269   43 RCL
270   10  10
271   75  -
272   43 RCL
273   12  12
274   54  )
275   39 COS
276   65  ×
277   43 RCL
278   19  19
279   85  +
280   53  (
281   43 RCL
282   10  10
283   75  -
284   43 RCL
285   14  14
286   54  )
287   39 COS
288   65  ×
289   43 RCL
290   20  20
291   85  +
292   53  (
293   43 RCL
294   10  10
295   75  -
296   43 RCL
297   16  16
298   54  )
299   39 COS
300   65  ×
301   43 RCL
302   21  21
303   54  )
304   92 RTN
```

prov kørning
steg 131 til 236

```
  36.  Tokio
 -41.
 -38.  Melbourne
 -35.
  35.  Los Angeles
  62.

10.1430  λ'
 2.1807  φ

3481.0
3481.0
3481.0

 180.      -
10.143   λ'  DMS
  10.2
10.24166667   =
 169.8
 169.8     IDMS
 169.5
169.48     IFIX
169.48   λ
```

```
60.12 } OSlo
11.18 }
55.25 } København
 13.
59.21 } Seh.
18.06 }

13.5233  λ
57.5801  φ

155.7 }
155.7 } km
155.7 }
```

TI 59
TennisScore

Program för TI-59

Undertecknad har äntligen kommit mig för med ett program för TI-59, vilket är det enda som intresserar mig, då jag inte har tillgång till någon 99:a.
Nedanstående program, som jag tror är originellt och är mer komplicerat än vad man kan tycka, är ett program för tennisdomare!
Skrivare är nödvändig. Listning bifogas.

Bruksanvisning:

Efter att programmet är inspelat, varom mera nedan, lägger man in 5 i register 00, om 5-setsmatch skall spelas.
Om 3-sets skall spelas företas härvid ingen åtgärd. Vidare, om första bollbyte ej skall ske efter 7 games, lägger man in den aktuella siffran, t.ex. 9 i register 01. Om första bollbytet alltså skall ske efter 7 games, företas ingen åtgärd. Ytterligare bollbyten sker automatiskt med ett tillägg av 2 games till den första siffran.
Spelet kan nu börja.
Den spelare eller den sida som servar först kallas A samt mottagaren för B. Dessa beteckningar följer spelarna genom hela matchen.
Om A t.ex. vinner en boll, trycker man på tangenten A.

Skrivaren registrerar:

15 A
0 B

Om B t.ex. vinner en boll, trycker man på tangenten B.

Skrivaren registrerar:

15 A
15 B

Skrivaren anger alltid servaren överst. Undantag för gamen och seten, som alltid visas med A överst. Skrivaren håller hela tiden reda på ställningen. Fördel anges med 50. Sålunda om servaren har fördel, i detta fall A, registrerar skrivaren:

50 A
40 B

Skrivaren anger bollar, games, bollbyte, set och match.

1,3 PROGRAMMERARE: FELLÄNDER				2,4
A	B			TIEBR ?

Vid ställningen 6-6 i games frågar skrivaren:
"Tiebreak? Tryck E".
Om långa set skall spelas trycker man alltså inte på E utan fortsätter som förut med att trycka på A om A vinner en boll, resp. på B om B vinner. Skall tiebreak tillämpas trycker man på E, varpå man fortsätter med att tryck på A resp. B.
Tiebreaksiffrorna redovisas med servaren överst.
Sidbyte efter var 6:te boll anges. Då tiebreaksetet är färdigspelat fortsätter programmet automatiskt utan åtgärd med vanlig tennisregistrering.

Inspelning av magnetkort. (Se bifogad listning).

Med normalindelning 6 2. op 17 fylles nedanstående register:

3	STO 00	
7	STO 01	
2	STO 02	
13	STO 07	
14	STO 08	
102	STO 11	
117	STO 12	
22133017	STO 14	GAME
3013371523	STO 15	MATCH
3724171435	STO 16	TIEBR
1713260071	STO 17	EAK ?
3145130014	STO 18	NYA B
3227271335	STO 19	OLLAR

Övergå därpå till indelning 0 2. op 17 samt spela in registren i block 4.
Ändra därpå indelningen till 3 2. op 17 samt knappa in programmet enligt listning, vilket spelas in på block 1, 2 och 3.
När man senare skall spela av magnetkorten börjar man med indelning 0 2. op 17 samt spelar av block 4. Därpå övergår man till indelning 3 2. op 17 samt spelar av block 1, 2 och 3.
Programmet arbetar med direktadressering, använder register 00-29, flagga 0 och 1 samt användardefinierade lablar A - E.

Lennart Felländer

TI 59

```
000  76 LBL    120  43 RCL    240  67  EQ    360  24  24    480  43 RCL    540  91 R/S    600  05  5     660  42 STO
001  15  E     121  04  04    241  02  02    361  42 STO    481  26  26    541  01  1     601  03  3     661  05  05
002  86 STF    122  69 OP     242  76  76    362  22  22    482  69 OP     542  44 SUM    602  07  7     662  92 RTN
003  01  01    123  06  06    243  43 RCL    363  00  0     483  06  06    543  03  03    603  01  1     663  43 RCL
004  00  0     124  71 SBR    244  16  16    364  48 EXC    484  71 SBR    544  44 SUM    604  07  7     664  11  11
005  42 STO    125  06  06    245  69 OP     365  06  06    485  06  06    545  09  09    605  00  0     665  48 EXC
006  09  09    126  83  83    246  02  02    366  48 EXC    486  88  88    546  61 GTO    606  00  0     666  12  12
007  91 R/S    127  43 RCL    247  43 RCL    367  29  29    487  43 RCL    547  05  05    607  00  0     667  42 STO
008  76 LBL    128  03  03    248  17  17    368  48 EXC    488  27  27    548  57  57    608  00  0     668  11  11
009  11  A     129  71 SBR    249  69 OP     369  27  27    489  71 SBR    549  86 STF    609  71 SBR    669  92 RTN
010  87 IFF    130  06  06    250  03  03    370  48 EXC    490  06  06    550  00  00    610  07  07    670  43 RCL
011  01  01    131  77  77    251  69 OP     371  25  25    491  77  77    551  71 SBR    611  07  07    671  20  20
012  05  05    132  87 IFF    252  05  05    372  42 STO    492  71 SBR    552  06  06    612  43 RCL    672  48 EXC
013  41  41    133  01  01    253  69 OP     373  23  23    493  06  06    553  49  49    613  09  09    673  21  21
014  43 RCL    134  01  01    254  00  00    374  71 SBR    494  83  83    554  61 GTO    614  55  ÷     674  42 STO
015  03  03    135  37  37    255  03  3     375  06  06    495  43 RCL    555  05  05    615  02  2     675  20  20
016  75  -     136  91 R/S    256  07  7     376  83  83    496  28  28    556  41  41    616  95  =     676  92 RTN
017  01  1     137  92 RTN    257  03  3     377  43 RCL    497  69 OP     557  43 RCL    617  22 INV    677  69 OP
018  05  5     138  76 LBL    258  05  5     378  28  28    498  06  06    558  03  03    618  59 INT    678  06  06
019  95  =     139  14  D     259  04  4     379  69 OP     499  71 SBR    559  75  -     619  67  EQ    679  71 SBR
020  77  GE    140  00  0     260  05  5     380  06  06    500  06  06    560  07  7     620  06  06    680  07  07
021  00  00    141  42 STO    261  01  1     381  71 SBR    501  88  88    561  95  =     621  25 CLR    681  11  11
022  28  28    142  03  03    262  05  5     382  06  06    502  43 RCL    562  22 INV    622  71 SBR    682  92 RTN
023  01  1     143  42 STO    263  02  2     383  88  88    503  29  29    563  77  GE    623  06  06    683  43 RCL
024  05  5     144  04  04    264  06  6     384  43 RCL    504  71 SBR    564  05  05    624  63  63    684  07  07
025  42 STO    145  71 SBR    265  69 OP     385  29  29    505  06  06    565  77  77    625  25 CLR    685  69 OP
026  03  03    146  06  06    266  02  02    386  71 SBR    506  77  77    566  43 RCL    626  91 R/S    686  04  04
027  13  C     147  63  63    267  01  1     387  06  06    507  71 SBR    567  03  03    627  22 INV    687  92 RTN
028  75  -     148  22 INV    268  07  7     388  77  77    508  06  06    568  75  -     628  86 STF    688  43 RCL
029  01  1     149  87 IFF    269  52 EE     389  61 GTO    509  83  83    569  43 RCL    629  01  01    689  08  08
030  05  5     150  00  00    270  06  6     390  01  01    510  43 RCL    570  04  04    630  00  0     690  69 OP
031  95  =     151  01  01    271  22 INV    391  98  98    511  05  05    571  75  -     631  42 STO    691  04  04
032  77  GE    152  56  56    272  52 EE     392  71 SBR    512  69 OP     572  02  2     632  03  03    692  92 RTN
033  00  00    153  71 SBR    273  71 SBR    393  06  06    513  06  06    573  95  =     633  42 STO    693  43 RCL
034  40  40    154  06  06    274  07  07    394  99  99    514  71 SBR    574  77  GE    634  04  04    694  14  14
035  03  3     155  56  56    275  07  07    395  43 RCL    515  06  06    575  06  06    635  01  1     695  71 SBR
036  00  0     156  01  1     276  25 CLR    396  15  15    516  88  88    576  27  27    636  44 SUM    696  07  07
037  42 STO    157  44 SUM    277  91 R/S    397  71 SBR    517  43 RCL    577  13  C     637  05  05    697  07  07
038  03  03    158  05  05    278  43 RCL    398  07  07    518  06  06    578  43 RCL    638  44 SUM    698  92 RTN
039  13  C     159  44 SUM    279  05  05    399  07  07    519  71 SBR    579  09  09    639  10  10    699  03  3
040  75  -     160  10  10    280  75  -     400  22 INV    520  06  06    580  55  ÷     640  71 SBR    700  06  6
041  01  1     161  71 SBR    281  43 RCL    401  87 IFF    521  77  77    581  06  6     641  06  06    701  01  1
042  00  0     162  06  06    282  06  06    402  00  00    522  05  5     582  95  =     642  63  63    702  07  7
043  95  =     163  93  93    283  75  -     403  04  04    523  01  1     583  22 INV    643  71 SBR    703  03  3
044  77  GE    164  43 RCL    284  02  2     404  14  14    524  69 OP     584  59 INT    644  06  06    704  07  7
045  00  00    165  05  05    285  95  =     405  22 INV    525  02  02    585  22 INV    645  93  93    705  00  0
046  52  52    166  75  -     286  77  GE    406  86 STF    526  05  5     586  67  EQ    646  61 GTO    706  00  0
047  04  4     167  06  6     287  02  02    407  00  00    527  01  1     587  06  06    647  02  02    707  69 OP
048  00  0     168  95  =     288  92  92    408  71 SBR    528  05  5     588  12  12    648  92  92    708  03  03
049  42 STO    169  77  GE    289  61 GTO    409  06  06    529  01  1     589  03  3     649  43 RCL    709  69 OP
050  03  03    170  02  02    290  01  01    410  56  56    530  52 EE     590  06  6     650  03  03    710  05  05
051  13  C     171  78  78    291  72  72    411  71 SBR    531  06  6     591  02  2     651  48 EXC    711  69 OP
052  43 RCL    172  22 INV    292  43 RCL    412  06  06    532  22 INV    592  04  4     652  04  04    712  00  00
053  04  04    173  87 IFF    293  00  00    413  70  70    533  52 EE     593  01  1     653  42 STO    713  98 ADV
054  75  -     174  00  00    294  67  EQ    414  43 RCL    534  71 SBR    594  06  6     654  03  03    714  25 CLR
055  43 RCL    175  01  01    295  03  03    415  22  22    535  07  07    595  01  1     655  92 RTN    715  92 RTN
056  03  03    176  83  83    296  15  15    416  85  +     536  07  07    596  04  4     656  43 RCL
057  95  =     177  22 INV    297  00  0     417  43 RCL    537  98 ADV    597  69 OP     657  05  05
058  67  EQ    178  86 STF    298  48 EXC    418  23  23    538  98 ADV    598  02  02    658  48 EXC
059  00  00    179  00  00    299  00  00    419  95  =     539  98 ADV    599  04  4     659  06  06

060  65  65    180  71 SBR    300  75  -     420  22 INV
061  77  GE    181  06  06    301  03  3     421  67  EQ
062  00  00    182  56  56    302  95  =     422  04  04
063  70  70    183  71 SBR    303  67  EQ    423  47  47
064  14  D     184  06  06    304  03  03    424  43 RCL
065  05  5     185  83  83    305  12  12    425  24  24
066  00  0     186  43 RCL    306  03  3     426  85  +
067  42 STO    187  05  05    307  42 STO    427  43 RCL
068  03  03    188  69 OP     308  13  13    428  25  25
069  13  C     189  06  06    309  61 GTO    429  95  =
070  04  4     190  71 SBR    310  03  03    430  22 INV
071  00  0     191  06  06    311  15  15    431  67  EQ
072  42 STO    192  88  88    312  02  2     432  04  04
073  04  04    193  43 RCL    313  42 STO    433  62  62
074  13  C     194  06  06    314  13  13    434  43 RCL
075  76 LBL    195  71 SBR    315  22 INV    435  26  26
076  12  B     196  06  06    316  87 IFF    436  85  +
077  87 IFF    197  77  77    317  00  00    437  43 RCL
078  01  01    198  43 RCL    318  03  03    438  27  27
079  05  05    199  01  01    319  23  23    439  95  =
080  49  49    200  75  -     320  71 SBR    440  22 INV
081  86 STF    201  43 RCL    321  06  06    441  67  EQ
082  00  00    202  10  10    322  70  70    442  04  04
083  71 SBR    203  95  =     323  01  1     443  77  77
084  06  06    204  22 INV    324  44 SUM    444  61 GTO
085  49  49    205  67  EQ    325  20  20    445  04  04
086  11  A     206  02  02    326  43 RCL    446  92  92
087  76 LBL    207  24  24    327  20  20    447  71 SBR
088  13  C     208  43 RCL    328  75  -     448  06  06
089  22 INV    209  18  18    329  43 RCL    449  83  83
090  87 IFF    210  69 OP     330  13  13    450  43 RCL
091  00  00    211  02  02    331  95  =     451  22  22
092  01  01    212  43 RCL    332  67  EQ    452  69 OP
093  00  00    213  19  19    333  03  03    453  06  06
094  22 INV    214  71 SBR    334  92  92    454  71 SBR
095  86 STF    215  07  07    335  22 INV    455  06  06
096  00  00    216  07  07    336  87 IFF    456  88  88
097  71 SBR    217  00  0     337  00  00    457  43 RCL
098  06  06    218  42 STO    338  03  03    458  23  23
099  49  49    219  10  10    339  49  49    459  71 SBR
100  83 GD+    220  48 EXC    340  22 INV    460  06  06
101  11  11    221  02  02    341  86 STF    461  77  77
102  71 SBR    222  44 SUM    342  00  00    462  71 SBR
103  06  06    223  01  01    343  71 SBR    463  06  06
104  83  83    224  43 RCL    344  06  06    464  83  83
105  43 RCL    225  05  05    345  56  56    465  43 RCL
106  03  03    226  75  -     346  71 SBR    466  24  24
107  69 OP     227  06  6     347  06  06    467  69 OP
108  06  06    228  95  =     348  70  70    468  06  06
109  71 SBR    229  22 INV    349  71 SBR    469  71 SBR
110  06  06    230  67  EQ    350  06  06    470  06  06
111  88  88    231  02  02    351  99  99    471  88  88
112  43 RCL    232  76  76    352  00  0     472  43 RCL
113  04  04    233  43 RCL    353  48 EXC    473  25  25
114  61 GTO    234  05  05    354  05  05    474  71 SBR
115  01  01    235  75  -     355  48 EXC    475  06  06
116  29  29    236  43 RCL    356  28  28    476  77  77
117  71 SBR    237  06  06    357  48 EXC    477  71 SBR
118  06  06    238  95  =     358  26  26    478  06  06
119  88  88    239  22 INV    359  48 EXC    479  83  83
```

```
001  15  E
009  11  A
076  12  B
088  13  C
139  14  D
```

```
3.          00
9.          01
0.          02
0.          03
0.          04
8.          05
7.          06
15.         07
14.         08
0.          09
0.          10
117.        11
102.        12
0.          13
22133017.   14
3013371523. 15
3724171435. 16
1713260071. 17
3145130014. 18
3227271335. 19
0.          20
0.          21
0.          22
0.          23
0.          24
0.          25
0.          26
0.          27
0.          28
0.          29
```

```
15.   A      15.   A
 0.   B       0.   B

30.   A      30.   A
 0.   B       0.   B

30.   A      40.   A
15.   B       0.   B

20.   A        GAME
30.   B

                2.  A
30.   A         1.  B
40.   B

          NYA BOLLAR
   GAME

                0.  B
 0.   A       15.  A
 1.   B

                0.  B
15.   B       30.  A
 0.   A

                0.  B
30.   B       40.  A
 0.   A

                15. B
30.   B       40. A
15.   A

   GAME

30.   B
30.   A         3.  A
                1.  B
30.   B
40.   A        15.  A
                0.  B
   GAME

                30. A
 1.   A        0.  B
 1.   B

                30. A
               15.  B

                40. A
               15.  B

   GAME

                4.  A
                1.  B
```

TI 59

PåskSöndagen

Del av brev

Pargas, den 7/4-84

Hej Claes!

''''',,,,,''''',,,,,

För en tid sedan fanns det en utmaning för TI-57 att beräkna påsksöndagen mellan (1800-2099). Det verkade så enkelt varför jag knappade in ekvationen direkt på min 59:a. Tji. Fel resultat. Listning. Hittade inte felet.
Papper och penna. Samma resultat som tidigare. Gick till biblioteket. Lånade 'Årets dagar'. Returnerade boken följande dag med en testremsa.
Här har du min version:

'Årtalet' tryck på E

Antagligen Gauss som gjort härledningen.

m = 24	i ovannämnda
n = 5	tidsintervall
a = årtalet	mod 19
b = -"-	mod 4
c = -"-	mod 7
d = 19a + m	mod 30
e = 2b + 4c + 6d + n	mod 7

om 22 + d + e <= 31 är det mars
annars ger d + e - 9 april såvida inte
1) d + e = 35 som ger 19 april eller
2) a > 10, d = 28 och d + e = 34 som ger 18 april

ps
ABC-80 klämde ur sig 10 på varandra följande års påsker när TI-59 höll på med texten! Jag provade. Fick samma värden.

Raul

Ett ord på vägen från Claes för er som vill överföra programmet till annan räknare eller dator.
Programmet rad 011 INT.
INT kapar decimaler rakt av: 3.9987996 blir 3.

I andra maskiner kan INT avrundas till närmaste heltal: 3.50001 blir 4 och 3.49999 blir 3.

Lika fast tvärs om INV INT kapar heltalsdelenoch lämnar decimaldelen orörd.

Ässå knappa in rätt tangenter: ett x som blev + tog en timme att hitta, å då först med hjälp av en TRACE-remsa.

```
000  53  (      045  29 CP     090  42 STO    135  06  06     180  69 OP     225  69 OP     270  03  3
001  43 RCL      046  01  1     091  05  05     136  67  EQ     181  04  04     226  02  02     271  02  2
002  00  00      047  09  9     092  85  +      137  01  01     182  43 RCL     227  03  3      272  07  7
003  75  -       048  65  x     093  43 RCL    138  44  44     183  06  06     228  02  2      273  69 OP
004  32 X:T      049  43 RCL    094  04  04     139  61 GTO     184  85  +      229  03  3      274  01  01      ANNO DOMINI
005  65  x       050  01  01    095  95  =      140  01  01     185  02  2      230  00  0      275  02  2          1981.     SKER
006  53  (       051  85  +     096  42 STO    141  58  58     186  02  2      231  02  2      276  07  7
007  32 X:T      052  02  2     097  06  06     142  01  1      187  95  =      232  04  4      277  01  1     INFALLANDET DEN
008  55  ÷       053  04  4     098  32 X:T    143  85  +      188  69 OP     233  03  3      278  03  3           19.     APRL
009  32 X:T      054  95  =     099  09  9      144  01  1      189  06  06     234  01  1      279  03  3
010  54  )       055  42 STO    100  77  GE     145  08  8      190  98 ADV     235  02  2     280  01  1
011  59 INT      056  00  00    101  01  01     146  95  =      191  91 R/S     236  04  4     281  01  1
012  54  )       057  03  3     102  70  70     147  42 STO    192  53  (      237  69 OP     282  06  6
013  92 RTN      058  00  0     103  03  3      148  07  07     193  69 OP     238  03  03     283  01  1      ANNO DOMINI
014  76 LBL      059  32 X:T    104  05  5      149  71 SBR    194  05  05     239  69 OP     284  07  7          1984.     SKER
015  15  E       060  71 SBR    105  67  EQ     150  01  01     195  01  1      240  05  05     285  69 OP
016  42 STO      061  00  00    106  01  01     151  92  92     196  03  3      241  69 OP     286  02  02     INFALLANDET DEN
017  00  00      062  00  00    107  42  42     152  43 RCL    197  03  3      242  00  00     287  03  3           22.     APRL
018  61 GTO      063  42 STO    108  43 RCL    153  07  07     198  03  3      243  69 OP     288  07  7
019  02  02      064  04  04    109  01  01     154  69 OP     199  03  3      244  05  05     289  00  0
020  07  07      065  29 CP     110  32 X:T    155  06  06     200  05  5      245  03  3      290  00  0
021  01  1       066  07  7     111  01  1      156  98 ADV    201  02  2      246  06  6      291  01  1
022  09  9       067  32 X:T    112  00  0      157  91 R/S    202  07  7      247  02  2      292  06  6      ANNO DOMINI
023  32 X:T      068  02  2     113  22 INV    158  71 SBR    203  69 OP     248  06  6      293  01  1          1985.     SKER
024  71 SBR      069  65  x     114  77  GE     159  01  01     204  04  04     249  01  1      294  07  7
025  00  00      070  43 RCL    115  01  01     160  92  92     205  54  )      250  07  7      295  03  3     INFALLANDET DEN
026  00  00      071  02  02    116  20  20     161  43 RCL    206  92 RTN    251  03  3      296  01  1           7.      APRL
027  42 STO      072  85  +     117  61 GTO    162  06  06     207  69 OP     252  05  5      297  69 OP
028  01  01      073  04  4     118  01  01     163  75  -      208  00  00     253  69 OP     298  03  03
029  29 CP       074  65  x     119  58  58     164  09  9      209  69 OP     254  04  04     299  69 OP
030  04  4       075  43 RCL    120  02  2      165  95  =      210  05  05     255  43 RCL    300  05  05
031  32 X:T      076  03  03    121  08  8      166  69 OP     211  01  1      256  00  00     301  69 OP
032  71 SBR      077  85  +     122  32 X:T    167  06  06     212  03  3      257  69 OP     302  00  00      ANNO DOMINI
033  00  00      078  06  6     123  43 RCL    168  98 ADV    213  69 OP     258  06  06     303  61 GTO
034  00  00      079  65  x     124  04  04     169  91 R/S    214  01  01     259  69 OP     304  00  00          1986.     SKER
035  42 STO      080  43 RCL    125  67  EQ     170  69 OP     215  03  3      260  00  00     305  21  21
036  02  02      081  04  04    126  01  01     171  05  05     216  01  1      261  69 OP          INFALLANDET DEN
037  29 CP       082  85  +     127  31  31     172  03  3      217  03  3      262  05  05     015  15  E           30.     MARS
038  07  7       083  05  5     128  61 GTO    173  00  0      218  01  1      263  02  2
039  32 X:T      084  95  =     129  01  01     174  01  1      219  03  3      264  04  4
040  71 SBR      085  42 STO    130  58  58     175  03  3      220  02  2      265  03  3
041  00  00      086  00  00    131  03  3      176  03  3      221  00  0      266  01  1
042  00  00      087  71 SBR    132  04  4      177  05  5      222  00  0      267  02  2
043  42 STO      088  00  00    133  32 X:T    178  03  3      223  01  1      268  01  1
044  03  03      089  00  00    134  43 RCL    179  06  6      224  06  6      269  01  1
```

FÖRENINGENs FORTH

och

TI FORTH

Denna beskrivning över programspråket
TI FORTH
erhållen från Maurice Swinnen, USA
kan köpas genom:

FÖRENINGEN PROGRAMBITEN, POSTGIRO: 19 83 00 - 6

för MEDLEMMAR	för ICKE MEDLEMMAR
190 SEK	270 SEK

Föreningen har även programmet på diskett, vilken kostar:

för MEDLEMMAR	för ICKE MEDLEMMAR
60 SEK	100 SEK

För beskrivning och diskett tillsammans:

för MEDLEMMAR	för ICKE MEDLEMMAR
250 SEK	370 SEK

Föreningen har även en egen FORTH framtagen av Björn Gustavsson, vilken inklusive kortfattad beskrivning kostar:

för MEDLEMMAR	för ICKE MEDLEMMAR
250 SEK	370 SEK

Vid SAMTIDIG beställning av TI FORTH och Föreningens FORTH

för MEDLEMMAR	för ICKE MEDLEMMAR
400 SEK	600 SEK

Medlem i FÖRENINGEN PROGRAMBITEN blir man enklast genom att betala in 120 SEK på POSTGIRO: 19 83 00 - 6.

Reservation mot prisändringar på grund av materialfördyrningar utanför vår kontroll.

Stockholm 1984-06-20

INPUT ETT SUBPROGRAM I ASSEMBLER

Av Börje Häll

Input är ett subprogram för inmatning av data från tangentbordet. Det är avsett att anropas från ett annat assembler program och kan inte användas fristående. I det anropande programmet måste REF INPUT finnas med. De tecken som kan matas in är ASCII-koderna 32-126, d.v.s från blank till tilde. Inga kontroller för att förhindra skrivning utanför skärmen finns med i programmet. Skärm-adresserna är 0-767 i GRAPHICS MODE och 0-959 i TEXT MODE. Det sker heller ingen kontroll att max antal tecken som får matas in är större än 0.

Förhoppningsvis förklarar kommentarerna på respektive rad hur subprogrammet fungerar. Det som kanske behöver förtydligas är JHE på raderna 115 och 142.
Talen i datorn lagras binärt med negativa tal i 2-komplementform. Det betyder att datorn behandlar tal i området -32768 t.o.m. 32767. JHE är en logisk jämförelse och det innebär att datorn vid jämförelse inte uppfattar tal i 2-komplementform som negativa. Det betyder i sin tur att datorn, vid logisk jämförelse, arbetar med talområdet 0 t.o.m. 65535. På rad 115 ska hopp ske om antal inmatade tecken är lika med eller större än max antal tecken som får matas in. Varken max antal tecken eller antal tecken som har matats in kan vara negativt och det innebär att vi bara jämför positiva tal. Då kan JHE användas i den vanliga betydelsen av större än eller lika med. Motsvarande gäller för rad 142.

Om emellertid något av talen eller båda talen råkar vara negativt så kan inte JHE användas utan man kan göra på följande sätt:

```
Antag R3=2
      R5=-5
      Hopp ska ske on R3>=R5

      C    R3,R5
      JEQ  A  * Inget hopp ty R3<>R5
      JGT  A  * Hopp ty R3>R5
```

Om JHE används så sker inget hopp eftersom datorn, vid logisk jämförelse, uppfattar 2 som mindre än -5. Programmet sparar inte data som matas in utan du måste göra en subrutin som sparar tecknen på skärmen.

```
*****************************************************
* Källkod: K:INPUT                 Använda register *
* Objektkod O:INPUT                R0  VDP access   *
*                                  R1  -"-          *
* Börje Häll                       R2  Status       *
* Vasavägen 101                    R3  Teckenräknare*
* 175 32  JÄRFÄLLA                 R4  Blinkräknare *
*                                  R5  Max antal tecken *
* 1984-04-23                           som får matas in *
* Modifierat 1984-05-27            R14 Överföring av data *
*                                      och returadress *
* Subprogram för inmatning                         *
*                                                  *
* Antal inmatade tecken returneras                 *
*        i anropande programs R10                  *
*                                                  *
* Anrop: BLWP @INPUT                               *
*        DATA Startadress på skärmen               *
*        DATA Max antal tecken                     *
*                                                  *
*****************************************************
```

```
IDT  'INPUT'
DEF  INPUT
REF  VSBW
*-------------------------
* Equates
*
KEY    EQU  >8375 * Här sparas nertryckt tangen
                    t
KEYBRD EQU  >8374 * Vilket tangentbord som ska
                    skannas
STATUS EQU  >837C
*-------------------------
* Konstanter
*
B0     BYTE 0
B127   BYTE 127   * För kontroll av godtagbart
                    tecken
B31    BYTE 31    * För kontroll av godtagbart
                    tecken
B32    BYTE 32    * Blanktecken
ENTER  BYTE >0D   * För kontroll av ENTER-tang
                    enten
PV     BYTE 8     * För kontroll av pil vänste
                    r-tangenten
EVEN
D0     DATA 0     * För kontroll av flaggor
D1     DATA 1     * För kontroll av flaggor
SET    DATA >2000 * För kontroll om någon tang
                    ent har tryckts ner
*-------------------------
* Variabler
*
OLDKEY BYTE 0     * För att spara gammalt teck
                    en
EVEN
BLINK  DATA 0     * 1=skriv tecken
DLJUD  DATA 0     * 1=ljud har avgivits
START  DATA 0     * Startposition på skärmen
STOPP  DATA 0     * Slutposition på skärmen
*-------------------------
* Workspace
*
INPREG BSS  >20   * Subprogrammets workspace
*-------------------------
* Programstart
*
INPUT  DATA INPREG        * Workspace pointer
DATA INP1          * Program counter
INP1   MOV  *R14+
MOV  *R14+
MOV  @STOPP
CLR  @BLINK        * 0-ställ blinkflaggan
CLR  @DLJUD        * 0-ställ ljudflaggan
MOVB @B32
CLR  R3            * 0-ställ teckenräknaren
CLR  R4            * 0-ställ blinkräknaren
A    @START
*                        * för att få slutpos
                   ition på skärmen
*                        * OBS slutpositionen
                    är 1 för mycket
DEC  @STOPP        * Justera slutpositionen ti
                   ll rätt värde
MOV  @START
INP2   C    @BLINK
JEQ  INP3          * Ja
LI   R1
INC  R4            * Öka blinkräknaren
CI   R4
JLT  INP4          * Nej
INC  @BLINK        * Ja
CLR  R4            * 0-ställ blinkräknaren
JMP  INP4          * Hoppa och skriv cursorn
INP3   MOVB @OLDKEY
INC  R4            * Öka blinkräknaren
CI   R4
JLT  INP4          * Nej
DEC  @BLINK        * Ja
CLR  R4            * 0-ställ blinkräknaren
INP4   BLWP @VSBW        * Skriv tecknet/curs
                   orn på skärmen
```

```
LIMI 2              * För att ljud ska höras
LIMI 0
CLR  @KEYBRD        * Tangentbord 0 ska skannas
BLWP @KSCAN         * Skanna tangentbord 0
MOVB @STATUS
COC  @SET
JNE  INP2           * Nej
CB   @KEY
JEQ  INP11          * Ja
INP5   CB   @KEY
JEQ  PILV           * Ja
CB   @KEY
JGT  INP6           * Nej
BL   @LJUD          * Ja
*                         * bad response tone
JMP  INP2
INP6   CB   @KEY
JLT  INP7           * Nej
BL   @LJUD          * Ja
JMP  INP2
*-------------------------
* Godtagbart tecken
*
INP7   C    R3
JHE  INP8       * Ja
INC  R3         * Nej
INP8   MOVB @KEY
BLWP @VSBW      * Skriv tecknet på skärmen
INC  R0         * Öka skrivpositionen
C    R0
JGT  INP9       * Ja
BLWP @VSBR      * Nej
MOVB R1
JMP  INP2
INP9   C    @DLJUD
JEQ  INP10      * Ja
BL   @LJUD      * Nej
INP10  MOV  @D1
MOV  @STOPP
BLWP @VSBR      * Läs ett tecken
MOVB R1
BLWP @VSBW      * Skriv tecknet
JMP  INP2
*-------------------------
* Pil vänster
*
PILV   MOVB @B32
BLWP @VSBW      * Skriv tecknet på skärmen
DEC  R3         * Minska teckenräknaren
DEC  R0         * Minska skrivadressen
C    R0
JHE  PILV1      * Nej
BL   @LJUD      * Ja
MOV  @START
CLR  R3         * 0-ställ teckenräknaren
PILV1  BLWP @VSBR      * Läs tecken från skärm
                  en
MOVB R1
BLWP @VSBW      * Skriv cursorn på skärmen
B    @INP2
*-------------------------
* Avsluta
*
INP11  MOVB @OLDKEY
BLWP @VSBW      * Skriv tecknet på skärmen
MOV  R3
*                       * anropande programs R
                  10
RTWP            * Hoppa till anropande progra
                  m
*****************************************
* Subrutin för att avge bad response tone *
*                                         *
* Anrop: BL   @LJUD                       *
*****************************************
LJUD   MOVB @B0
BLWP @GPLLNK    * Ge ljud
DATA >36        * Bad response tone
RT              * Hoppa till anropande rutin
END
```

■

BOKRESCENSION

Av Göran Nygren

Bokhuset på Gamla Brogatan har tagit hem några böcker vilka speciellt tar upp programmering på TI99/4A. Bokhuset är välsorterat med dator- och programmeringslitteratur

En bok jag fastnat för behandlar assemblerprogrammering med 99:an.

"Fundamentals of TI-99/4A Assembly language" skriven av M.S. Morley och utgiven på Tab Books Incorporated, med IBS nr. 0-8306-1722-1 (pocketupplagan) är en underbar bok för den som vill förstå assemblerprogrammering Den är i första hand skriven för att användas med Mini-Memory Modulen och Line-by-Line assemblatorn som följer med modulen. Den finns att köpa för priser ned till omkring 600 kr. Själv har jag betalat omkring 1000 kr.Det tycker jag fortfarande är billigt om man jämför med Editor/Assembler modulen plus expansionsminne.Tidningens skrift om Mini-Memory (Användartips med Mini-Memory 60 kr) ökar nöjet i hög grad. Modulen går även att använda tillsammans med TI-Basic program.

Boken är lättsamt skriven och mycket instruktiv för oss som inte är så slängda i maskinkodsprogrammering. Ett litet förord tar upp varför boken överhuvudtaget skrevs. Det fick mig att le jag kände igen tankegångarna.

I del 1 och 2 tar Morley upp grundläggande kunskaper om maskinkoder, mnemoniska koder, binära tal, TI-99/4A's alla enheter och deras adresser. Hur processorn är uppbyggd och mycket annat. Denna del i boken gav mig en god bild hur en dator och dess mjukvara är uppbygd.

I del 3 börjar själva programmeringen. Morley går stegvis från enkla operationer till mer komplexa. Varje programexempel förklaras ingående och instruktivt.

"Fundamentals of TI-99/4A Assembly Language" är ett förnuftigt inköp av den som vill lära maskinkarkitektur och assemblerprogrammering.

■

SÄLJES

Editor/Assembler Modul (TI-original)

Komplett

2 disketter Part A (Editor/Assembler routine) Part B (Save routine and Tombstone City source and object code)

Manual: Editor/Assembler

Pris: 700 kr.

Tel: 0510 - 60545

Johan Bleeker

SÄLJES

TI99/4A säljes med expansionsbox, diskdrive, diskkontrollkort, Extended Basic, Joysticks och massor av program (c:a 150 st), tidningar m.m.

Lars Ekeroth
Tel: 036/120649
Säkrast kvällar.

Konsten att skriva adventure spel i EX-basic

Av Love Feuer.

För adventurefantaster som kan ganska mycket EXTENDED BASIC.

En vacker dag insåg jag meningslösheten med att hålla på med vanliga spel.
Därför gick jag till min Texas-handlare och köpte adventure modulen.
Men när man löst Pirate adventure och har fått kopior på alla andra adventurespelen så börjar man fundera på att skriva ett eget äventyrsspel.

Sagt och gjort, min första skapelse blev klar för ungefär ett halvår sedan.
Men spelet var inte riktigt fulländat....
D.v.s. att programet satte upp en karta i ett X,Y system där t.ex. cykeln stod på ruta 3,5.
Det ända man kunde göra var att gå omkring på kartan, så ett adventurespel var det lite väl magstarkt att kalla det.
Så där höll jag på ett par veckor och gjorde grafikkartor och andra värdelösa finesser, Men jag kom inte vidare.
Men en dag så berättade en god vän för mig om den ädla konsten att göra effektiva och minnessnåla kartor som dessutom var mycket snabba.
Konsten var att ha fyra indexerade variabler för varje rum, en variabel för varje riktning.
Variabeln innehöll nummret på det rum man gick till.

Ett litet exempel skulle kanske vara på sin plats.....

Om vi namnger de fyra variablerna som:

N( ) för rummet åt norr.
S( ) för rummet åt söder.
E( ) för rummet åt öster.
W( ) för rummet åt väster.

Om vi står i rum 3 och ämnar gå norrut så kollar datorn variabeln "N(3)" och finner att den innehåller värdet "4", alltså det rum som du kommer till när du går norr.

Man kan också lägga in alltihop i en tvådimensionell variabel typ A(rum,riktning) där rum är vilket rum som man går ifrån och riktning är den nummret på den riktning man vill gå Ex.:

1 = norr.
2 = öster.
3 = söder.
4 = väster.

Om "A(2,3)" är 5 innebär det att man hamnar i rum 5 om man går söder ifrån rum 2.
Denna metod ger också stor rörelsefrihet angående specialrum som lönngångar, fallluckor, hissar O.S.V.

Själva rumsbeskrivningen lägger jag i en separat sträng som jag kallar RU$( ).
Innehållet i den kan vara t.ex "I EN MÖRK GROTTA", därmed behöver man bara skriva "JAG ÄR " och RU$( ) för att få en beskrivning av rummet.

När det gäller saker i spelet började jag med att ha en indexerad variabel för varje rum, variabeln innehöll nummret på det föremål som fanns där.
Det finns en fördel och två nackdelar med metoden.

Fördelen är att allting är mycket snabbt, man behöver bara lista sakerna för just det rummet, inget sökande alltså.

Nackdelarna är följande.:
1. Flexibiliteten blir klart lidande eftersom man måste ha onödigt många variabler för varje rum. Och ska en sak T.ex försvinna så blir det en massa saker som ska göras.
2. Det krävs klart mer minne än den metod som beskrivs nedan. Efter som man måste dimensionera upp varje variabel dels efer antal rum och dels efter hur många saker det finns i HELA spelet.
Om vi har 52 rum och 78 saker så skulle "DIM" satsen anta följande form:
10 DIM A(52,78)
Och varje rum har upp till 10 variabler!
En motsvarande basic rad skulle se ut som:
10 dim A(10,52,78)
Pröva den raden och skåda hur det lätt irriterande felmedelandet: "* MEMORY FULL" framträder, och det här är ju bara sakerna, kartan och resten av programet ska ha sitt också.

Nej, då föredrar jag den metod jag nu använder.

Denna andra (och mycket bättre) metod bygger på att varje sak får sin egen uppsättning variabler.
Det kanske låter lika minneskrävande som metod No 1 men det är inte sant.

Här nedan beskriver jag vilka variabler som jag brukar använda. Dessa kan naturligtvis varieras beroende på hur avancerat spelet ska vara.
Som föremål väljer vi en liten nyckel som ligger på golvet.

SAK$( ) = Hur saken skrivs ut på skärmen, Här blir det "en liten nyckel".
KALL$( ) = Vad saken kallas av spelaren, Denna sträng finns därför att man inte gärna skriver "TA EN LITEN NYCKEL".
Nej istället skriver man hellre "TA NYCKELN" eller "TA NYC" om man vill använda förkortningar. Mer om det senare.

EX$( ) är hur saken undersöks, om vi undersökte nyckeln (med "UNDERSÖK NYCKEL") så kanske vi skulle få medelandet "Den är varm som om någon hållit i den.". EX$( ) skulle alltså vara "Den är varm som om någon hållit i den.".

VAR( ) är i vilket rum saken befinner sig, om VAR( ) är 3 skulle nyckeln ligga i rum 3.
Men VAR( ) har fler funktioner.
Om VAR( ) är 0 så är saken ute ur spelet, men om den istället är -1 så bär spelaren saken på sig.

FL( ) är en flagga som håller reda på följande:
FL( )=1 är att saken skrivs ut och att man kan ta saken
FL( )=2 är att saken skrivs ut men att den är otagbar (Ex. en ståldörr).
FL( )=3 är att saken inte skrivs ut men är tagbar. När denna funktion utnyttjas beskrivs längre ner.

Nyckelns FL( ) skulle få värdet 1 efter som den skrivs ut och är tagbar.

Och till sist men inte minst:

KL( ) som är en variabel som visar om man kan klättra på saken eller ej.
Om man t.ex hittade en flaggstång och kanske ville klättra på den så skulle KL( ) innehålla nummret på det rum man kom till när man tagit sig upp i stången.
Denna variabel kan vara bra om man vill ha t.ex en trappa som ligger åt norr.
Man sätter då en utgång åt norr och samtidigt placerar man en trappa som ett vanligt föremål i rummet.
Men man låter trappan vara outskriven genom att sätta FL( ) till 3 (se ovan).
Nu kanske ni frågar er hur spelaren ska få veta att det finns en trappa i rummet.
Jo... Man skriver in trappan i rumsbeskrivningen.
När man gör detta får utskriften en mycket "proffsigare" design.

Om man låter rumsbeskrivningen vara "JAG ÄR I EN DRYPANDE VÅT KÄLLARE, TILL NORR LEDER EN STENTRAPPA UPP I MÖRKRET.", ger det ett snyggare utseende än "JAG ÄR I EN DRYPANDE VÅT KÄLLARE." och under det :"JAG SER EN STENTRAPPA.".
När man sedan vill göra det man behagar med saken, så behandlas den precis som ett vanligt föremål.
Om vi nu skulle vilja gå i trappan så använder man bara KL( ) för att få reda på i vilket rum man hamnar i.

Alla dessa variabler (inklusive kartan) läses lämpligen in genom datasatser eller genom disken om man har en.
Ett exempel på en rutin som läser in två rum och fyra saker finns i fig. 1, som också initsierar alla variabler och tar hand om huvudloopen.

När vi nu vet lite om hur vi ska göra kartor och placera saker i dessa kan det kanske vara på tiden att visa lite programeringsteknik.

För att göra det hela enkelt ska vi försöka ta efter Adventuremodulen i det längsta.

I Följande exempel använder jag de variabler som jag har beskrivit ovan plus en del nya som kommer här.:

R är det rum man är i.
ANT är hur många saker det finns i spelet.
RUM är hur många rum det finns i spelet.
V$ är komandot eller verbet. T.ex "TA". O.B.S. I förkortad form!
H$ är saken eller substantivet. T.ex "NYCKELN". O.B.S. I förkortad form!
A$ är meningen som skrivits in.

Vi börjar med utskriften....

När vi kommer in i ett rum så skriver datorn ut "JAG ÄR " och rumsbeskrivningen (som ligger i RU$( )).
Därefter gör datorn en loop som letar igenom alla saker i spelet och kollar om någon är i samma rum som man själv är i.(Detta görs genom att man jämför alla VAR( ) med R som är rummet man är i.)
Om den hittar någonting som är i det aktuella rummet så skriver den "JAG SER " och SAK$( ) (som är beskrivningen av föremålet ifråga.)
När maskinen är klar med loopen så tar den itu med utgångarna.
Den sätter då en flagga till ett och går sedan igenom alla riktningarna (variablerna N( ), E( ), S( ), W( )).
Om datorn hittar en riktning som skiljer sig från noll så skriver den ut den aktuella utgången och nollställer flaggan.
När den har gått igenom alla väderstrecken så kollar den om flaggan fortfarande är satt om den är den så finns det inga utgångar och maskinen medelar detta.

Allt detta görs lämpligen i ett subprogram finns listat i fig. 2.

Om ni undrar varför jag sätter flaggan till 1 och sedan nollställer den, så beror det på att om man skriver in en rad som "100 IF F THEN 100" så hoppar datorn till rad 100 om F är skilt från 0, man har då gallant sparat in två bytes. (Detta kallas BYTEJAKT!)

Redan nu kan jag medela att allting i det här spelet skrivs med STORA bokstäver... Detta gäller också inmatningen. ALPHA LOCK nere alltså!

När man skriver in ett kommando i ett Scott Adam's adventure har man två ord att röra sig med.
Det första ordet är vad man vill göra (t.ex "TA", "GÅ", "UNDERSÖK" O.S.V.), Det andra är med vad man vill göra något eller en riktning (t.ex "NYCKELN", "ASKEN", "DÖRREN", "NORR", O.S.V.).

Om man ska spela ett adventure en längre stund, kan man riskera att få fingrar som liknar ett par välkokta korvar, om man tvingas skriva ner hela orden, därför brukar datorn klara av förkortningar med de tre första bokstäverna av orden.
T.ex "NYCKEL" blir "NYC" och "KLÄTTRA" blir "KLÄ".

Nu har vi satt upp de saker som vi behöver för att göra en input rutin där man skriver kommandot i "A$" och får tillbaka de två orden i "V$" resp. "H$" i förkortad form, rutinen protesterar dessutom om fler än två ord skrivs in.
Alltihop visas med remsatser i fig. 3

När vi sen har skrivit in vårt kommando ska vi kolla om vi vill gå någonstans, om vi vill det så ska vi hamna i ett nytt rum.
Men här ska man också kunna skriva bara T.ex. "N" för att ta sig norrut.
Allt detta och felmedelanden tas omhand av subrutinen i fig. 4, Variablerna är de samma som beskrevs i stycket om karttekniken. Om-andra ordet inte är en riktning, så hoppar programet till subrutinen i fig. 8. (Se nedan.)

Vi fortsätter sedan med att ta upp föremål. Den rutinen visas i fig. 5, här finns också en felsökningsrutin som säger till om man : 1, inte ser den. 2, inte kan ta den. 3, redan har den.
Släppa....fig. 6, med nästan samma felsökning som i rutin 5.

Rutinen för att undersöka saker visas i fig. 7 (som subrutin), om datorn inte hittar något andra-ord så skriver den ut rumsbeskrivningen.

En subrutin för att klättra på saker visas i fig. 8. denna rutin kallas aldrig direkt av datorn, den funkar som en under rutin till subrutinen i fig. 5 och anropas om andra-ordet inte är en riktning. Därmed kan man både skriva "GÅ NORR" och "GÅ TRAPPA".

Och till sist, en rutin som visar vad man har. Den anropas med "UTRUSTNING" varvid allting man bär på sig visas. Rutinen finns i fig. 9.

Nu har jag tagit med grunderna för att göra sina egna adventurespel, men det finns mycket mer.
Det ska in en massa specialrutiner som är helt beroende av äventyret i sig.(T.ex rutiner för att vandra i mörker.)
Men sådana saker saker får ni göra själva.

Om ni har synpunkter, förbättringar och naturligtvis frågor så kan ni antingen ringa mig eller skriva ett brev till nedanstående adress.
Tänk på att även de mest "idiotiska" sakerna kan vara interesanta.

Adressen är: Love Feuer
Västerlånggatan 65
111 29 Stockholm

Tel. 08/21 62 78 efter kl.20.00

P.S. Om intresset visar sig vara stort för den här typen av program kanske jag kommer igen med nya rutiner, bland annat har jag skissat på en kartgenerator som med hjälp av en ordlista skapar egna kartor med saker slumpmässigt utplacerade.

Tack för att ni läste dessa 11004 bytes.

Upprop ADVENTURE Adventure

Adventure spelare har du också stött på patrull och inte kommer längre. Likadant här, kan vi hjälpa varandra ? Ring eller skriv till mig.

Johan Bleeker
Linjevägen 11
531 00 Lidköping
Tel: 0510-60545

```
100 ! *****************
110 ! * ADVENTURESPEL *
120 ! *               *
130 ! * AV LOVE FEUER *
140 ! *     1984      *
150 ! *****************
160 !
170 !
180 !  ***************
190 !  * INITIERING *
200 !  *     AV      *
210 !  *  VARIABLER  *
220 !  ***************
230 !
240 ! DIMENSIONERING AV
250 ! VARIABLER.
260 ! DESSA VÄRDEN FÅR
270 ! VARIERAS BEROENDE PÅ
280 ! ANTAL RUM & SAKER.
290 !
300 DIM RUM$(2),N(2),E(2),S(2),W(2)! ## RUMMEN
310 DIM SAK$(4),KALL$(4),EX$(4),VAR(4),FL(4),KL(4)! #
    # SAKERNA
320 !
330 ! INLÄSNING AV ANTAL
340 ! RUM & SAKER
350 !
360 READ RUM,ANT
370 ! LÄS IN VARIABLERNA
380 ! FRÅN DATA SATSERNA
390 !
400 ! ( FÖRST RUMMEN OCH
410 ! SEDAN SAKERNA. )
420 !
430 FOR I=1 TO RUM :: READ RUM$(I),N(I),E(I),S(I),W(I
    ):: NEXT I
440 !
450 ! SAKERNA
460 !
470 FOR I=1 TO ANT :: READ SAK$(I),KALL$(I),EX$(I),VA
    R(I),FL(I),KL(I):: NEXT I
480 !
490 R=1 :: CALL CLEAR ! RUMMET MAN ÄR I & SVENSKA TEC
    KEN
500 !
510 !
520 ! UTSKRIFT AV RUMMET
530 !
540 CALL PRINTOUT(R,ANT,VAR(),FL(),RUM$(),N(),E(),S()
    ,W(),SAK$())
550 !
560 ! INPUTRUTIN
570 !
580 CALL INP(V$,H$)
590 !
600 !*HÄR BÖRJAR HUVUDLOOPEN
610 !
620 ! VILL MAN GÅ ?
630 IF V$="N" OR V$="NOR" OR V$="Ö" OR V$="ÖST" OR V$
    ="S" OR V$="SÖD" OR V$="V" OR V$="VÄS" THEN H$=V$
     :: GOTO 3000
640 IF V$="GÅ" OR V$="KLÄ" OR V$="KRY" OR V$="SPR" TH
    EN 3000
650 !
660 ! VILL MAN TA EN SAK ?
670 !
680 IF V$="TA" THEN 3500
690 !
700 ! VILL MAN SLÄPPA?
710 !
720 IF V$="SLÄ" OR V$="DRO" OR V$="KAS" THEN 4000
730 !
740 ! VILL MAN UNDERSÖKA?
750 !
760 IF V$="UND" OR V$="TIT" OR V$="KOL" OR V$="SE" OR
     V$="EXA" THEN 4500
770 !
780 ! UTRUSTNING
790 IF V$="UTR" THEN 5500
800 PRINT "JAG KÄNNER INTE TILL VERBET." :: GOTO 580
810 !**********************
820 !* HÄR LÄGGER DU IN   *
830 !* EGNA RUTINER.      *
840 !*                    *
850 !* LYCKA TILL !       *
860 !*(NI KAN BEHÖVA DET.)*
870 !*                    *
880 !**********************
```

```
19950 !
19960 ! ** DATASATSER **
19970 !
19980 ! VI BÖRJAR MED ANTAL
19990 ! REM OCH SAKER
20000 DATA 2,4
20010 !
20020 ! HÄR LIGGER RUMS-
20030 ! BESKRIVNINGARNA.
20040 !
20050 DATA I EN STOR HALL MED EN TRAPPA ÄT NORR,2,0,0,0
20060 DATA "PÅ VINDEN I HUSET, ÄT SÖDER LEDER EN GÅNG N
      ERÅT",0,0,1,0
20070 !
20080 ! HÄR LIGGER SAKERNA.
20090 !
20100 DATA EN YXA,YXA,DEN ÄR BLODIG!,1,1,0
20110 DATA EN RING,RIN,DEN GLIMMAR SOM *GULD*,2,1,0
20120 DATA TRAPPAN,TRA,DEN LEDER UPPÅT.,1,3,2
20130 DATA GÅNGEN,GÅN,DEN LEDER TILL FÖRSTA RUMMET.,2,3
      ,1
20140 !
20150 ! HÄR KAN DU LÄGGA IN
20160 ! EGNA SAKER, MEN
20170 ! GLÖM INTE ATT ÄNDRA
20180 ! PÅ VÄRDENA I BÖRJAN
20190 ! AV PROGRAMET.
20200 !
```

```
29970 !
29980 ! SUBPROGRAMEN
29990 !
30000 SUB PRINTOUT(R,ANT,VAR(),FL(),RUM$(),N(),E(),S(),
      W(),SAK$())
30010 !
30020 ! SKRIV RUMMET
30030 !
30040 PRINT "JAG ÄR "&RUM$(R): :
30050 !
30060 ! LETA UTGÅNGAR
30070 !
30080 PRINT "SYNLIGA UTGÅNGAR ÄR " :: F=1
30090 IF N(R)THEN PRINT "NORR.";:: F=0
30100 IF E(R)THEN PRINT "ÖSTER.";:: F=0
30110 IF S(R)THEN PRINT "SÖDER.";:: F=0
30120 IF W(R)THEN PRINT "VÄSTER.";:: F=0
30130 IF F THEN PRINT "INGA!";
30140 PRINT : :
30150 !
30160 ! LETA SAKER
30170 !
30180 FOR I=1 TO ANT :: IF VAR(I)=R AND FL(I)<>3 THEN P
      RINT "JAG SER ";SAK$(I);"."
30190 NEXT I
30200 PRINT
30210 SUBEND
```

```
32200 !
32210 ! INPUT RUTIN
32220 !
32230 SUB INP(V$,H$):: H$="" :: V$=""
32240 INPUT "DINA ORDER ":A$
32250 PRINT
32260 IF A$="" THEN PRINT "SLUTA MUMLA!": : :: GOTO 322
      40
32270 !
32280 ! KOLLA OM ETT ORD
32290 !
32300 IF POS(A$," ",1)=0 THEN V$=A$ :: GOTO 32440
32310 P=POS(A$," ",1)
32320 !
32330 ! FLER ÄN 2 ORD ?
32340 !
32350 IF POS(A$," ",P+1)THEN PRINT "INTE FLER ÄN TVÅ OR
      D.": : :: SUBEXIT
32360 !
32370 ! SPLITTA UPP ORDEN.
32380 !
32390 V$=SEG$(A$,1,P-1)
32400 H$=SEG$(A$,P+1,LEN(A$)-P+1)
32410 !
32420 ! FÖRKORTA OM MÖJLIGT
32430 !
32440 IF LEN(V$)>3 THEN V$=SEG$(V$,1,3)
32450 IF LEN(H$)>3 THEN H$=SEG$(H$,1,3)
32460 SUBEND
```

```
2970 !
2980 ! FÖRFLYTTNINGSRUTIN
2990 !
3000 IF H$="" THEN PRINT "ETT ANDRA ORD TACK.": : :: G
     OTO 580
3010 !
3020 !VILL MAN GÅ PÅ SAKER?
3030 !
3040 IF H$<>"N" AND H$<>"NOR" AND H$<>"ÖST" AND H$<>"Ö
     " AND H$<>"SÖD" AND H$<>"S" AND H$<>"VÄS" AND H$<
     >"V" THEN 5000
3050 !
3060 ! KOLLA I VILKEN
3070 ! RIKTNING MAN VILL GÅ
3080 ! I OCH SKRIV FELMED-
3090 ! ELANDEN RESP OK.
3100 IF H$<>"N" AND H$<>"NOR" THEN 3130
3110 IF N(R)=0 THEN PRINT "JAG KAN INTE GÅ DITÅT.": :
     :: GOTO 580
3120 PRINT "OK.": : :: R=N(R):: GOTO 540
3130 IF H$<>"Ö" AND H$<>"ÖST" THEN 3160
3140 IF E(R)=0 THEN PRINT "JAG KAN INTE GÅ DITÅT.": :
     :: GOTO 580
3150 PRINT "OK.": : :: R=N(R):: GOTO 540
3160 IF H$<>"SÖD" AND H$<>"S" THEN 3190
3170 IF S(R)=0 THEN PRINT "JAG KAN INTE GÅ DITÅT.": :
     :: GOTO 580
3180 PRINT "OK.": : :: R=S(R):: GOTO 540
3190 IF W(R)=0 THEN PRINT "JAG KAN INTE GÅ DITÅT.": :
     :: GOTO 580
3200 PRINT "OK.": : :: R=W(R):: GOTO 540
```

```
3470 !
3480 ! TA RUTIN
3490 !
3500 IF H$="" THEN PRINT "ETT ANDRA ORD TACK.": : :: G
     OTO 580
3510 !
3520 ! LETA EFTER SAKEN
3530 !
3540 FOR I=1 TO ANT :: IF KALL$(I)=H$ THEN 3590
3550 NEXT I :: PRINT "JAG VET INTE VAD SAKEN ÄR.": : :
     : GOTO 580
3560 !
3570 ! HAR JAG DEN REDAN?
3580 !
3590 IF VAR(I)=-1 THEN PRINT "JAG HAR REDAN SAKEN.": :
      :: GOTO 580
3600 !
3610 ! SER JAG DEN?
3620 !
3630 IF VAR(I)<>R THEN PRINT "JAG SER INTE SAKEN.": :
     :: GOTO 580
3640 !
3650 ! TAGBAR?
3660 !
3670 IF FL(I)>1 THEN PRINT "JAG KAN INTE TA SAKEN.": :
      :: GOTO 580
3680 PRINT "OK.": : :: VAR(I)=-1 :: GOTO 580
```

```
3970 !
3980 ! SLÄPPA
3990 !
4000 IF H$="" THEN PRINT "ETT ANDRA-ORD TACK.": : :: G
     OTO 580
4010 !
4020 ! LETA EFTER SAKEN.
4030 !
4040 FOR I=1 TO ANT :: IF H$=KALL$(I)THEN 4060
4050 NEXT I :: PRINT "JAG VET INTE VAD SAKEN ÄR": : ::
      GOTO 580
4060 !
4070 ! HAR JAG DEN PÅ MIG?
4080 !
4090 IF VAR(I)<>-1 THEN PRINT "JAG HAR INTE SAKEN PÅ M
     IG.": : :: GOTO 580
4100 PRINT "OK.": : :: VAR(I)=R :: GOTO 580
```

```
4470 !
4480 ! UNDERSÖKA RUTIN
4490 !
4500 IF H$="" THEN 540
4510 !
4520 ! LETA EFTER SAKEN.
4530 !
4540 FOR I=1 TO ANT :: IF H$=KALL$(I)THEN 4560
4550 NEXT I :: PRINT "JAG VET INTE VAD SAKEN ÄR": : ::
      GOTO 580
```

```
4560 !
4570 ! SER JAG SAKEN
4580 !
4590 IF VAR(I)<>R AND VAR(I)<>-1 THEN PRINT "JAG SER I
     NTE SAKEN.": : :: GOTO 580
4600 PRINT EX$(I): : :: GOTO 580
4610
4620
```

```
4970 !
4980 ! KLÄTTRA PÅ SAKER.
4990 !
5000 ! LETA EFTER SAKEN.
5010 !
5020 FOR I=1 TO ANT :: IF KALL$(I)=H$ THEN 5070
5030 NEXT I :: PRINT "JAG VET INTE VAD DU VILL    KLÄT
     TRA PÅ": : :: GOTO 580
5040 !
5050 ! FINNS SAKEN HÄR?
5060 !
5070 IF VAR(I)<>R AND VAR(I)<>-1 THEN PRINT "JAG SER I
     NTE SAKEN HÄR.": : :: GOTO 580
5080 !
5090 ! KAN MAN KLÄTTRA?
5100 !
5110 IF KL(I)=0 THEN PRINT "JAG KAN INTE KLÄTTRA PÅ DE
     T.": : :: GOTO 580
5120 PRINT "OK.": : :: R=KL(I):: GOTO 540
5130
5140
```

```
5470 !
5480 ! UTRUSTNING
5490 !
5500 F=1 :: PRINT "JAG HAR :";
5510 !
5520 ! LETA IGENOM SAKERNA.
5530 !:
5540 FOR I=1 TO ANT :: IF VAR(I)=-1 THEN F=0 :: PRINT
     TAB(11);SAK$(I)
5550 NEXT I
5560 IF F THEN PRINT TAB(11);"INGENTING!"
5570 GOTO 580
5580
5590 STOP
```

```
100 CALL CLEAR
110 DIM TYPE$(5)
120 PR$="press space"
130 TYPE$(1)="DIS/FIX"
140 TYPE$(2)="DIS/VAR"
150 TYPE$(3)="INT/FIX"
160 TYPE$(4)="INT/VAR"
170 TYPE$(5)="PROGRAM"
180 OPEN #1:"DSK1.",INPUT ,RELATIVE,INTERNAL
190 INPUT #1:A$,I,J,K
200 DISPLAY AT(1,1):"DSK1-DISKNAME="&A$:"AVAILABLE="&
    STR$(K);" USED="&STR$(J-K)
210 IF I<>0 THEN DISPLAY AT(3,1):"PROTECTED" ELSE DIS
    PLAY AT(3,1):"NO PROTECTION"
220 DISPLAY AT(4,1):"FILENAME   SIZE   TYPE     P";"-
    --------- ---- --------- -";
230 FOR I=6 TO 24
240 INPUT #1:A$,A,J,K
250 IF LEN(A$)=0 THEN CLOSE #1 :: GOTO 330
260 DISPLAY AT(I,1):A$;TAB(11);J;TAB(17);TYPE$(ABS(A)
    )
270 IF ABS(A)<>5 THEN DISPLAY AT(I,24):STR$(K)
280 IF A<0 THEN DISPLAY AT(I,28):"Y"
290 IF I>23 THEN 300 ELSE 320
300 FOR I=1 TO LEN(PR$):: CALL HCHAR(I,31,ASC(SEG$(PR
    $,I,1))):: NEXT I
310 CALL KEY(0,K,S):: IF S=0 THEN 310 ELSE CALL HCHAR
    (6,1,32,608)
315 FOR I=1 TO LEN(PR$):: CALL HCHAR(I,31,32):: NEXT
    I :: GOTO 230
320 NEXT I
330 CALL KEY(0,K,S):: IF S=0 THEN 330
340 CALL INIT
500 DISPLAY AT(24,1):"PROGRAM NAME?"
510 ACCEPT AT(24,14)SIZE(10)BEEP:NA$
515 J=-35
520 FOR I=1 TO 10
540 IF I>LEN(NA$)THEN X=32 :: GOTO 560
550 X=ASC(SEG$(NA$,I,1))
560 CALL LOAD(J,X)
570 J=J+1
580 NEXT I
590 RUN "DSK1.AXXXXXXXXX"
```

MARKNADSÖVERSIKT

Av Bengt Falhgren

Jag har försökt att göra en sammanställning av tillbehör från tredje partstillverkare vilka är avpassade till TI99/4A.
Tyngdpunkten ligger på hårdvara, nyttoprogram samt programmeringshjälpmedel.
Materialet är bl.a. hämtat från HOME COMPUTER MAGAZINE.
En del av dessa tillbehör kan du köpa från firmor i Sverige.
T.ex. STIM99 (#27), BLIJENBURGH ELECTRONICS (#30), MAGNUSSONS DIGITALSERVICE (#29).

Uppläggning:
1. Minnesexpansion
2. RS232C Paralell(Centronics)
3. DOS (Diskoperativsystem)
4. EXPANSIONSYSTEM
5. OPERATIVSYSTEM
6. PROGRAMSPRÅK
7. MODULTILLVERKNING.
8. ÖVRIG HÄRDVARA
9. ORDBEHANDLINGSPROGRAM
10. DATABASPROGRAM
11. FILHANTERINGSPROGRAM
12. PROGRAMMERINGSHJÄLPMEDEL

****

1. MINNESEXPANSION

Fristående:

MULTICOM 32KRAM Stand alone $124.95
#2

Till X-BOX:

CORCOMP 32K CARD $199.9
#15

MULTICOM Mod 1000 32K CARD $99
#2

FOUNDATION 32K CARD $150
#3

FOUNDATION 128K CARD $230

2. RS232C - PIO

Fristående:

CORCOMP RS232+PIO $127
#15

MIKEL RS232C Stand Alone $99.95
#1

MULTICOM RS232C+PIO $144.95
#2

FÖR X-BOX:

CORCOMP RS232C+PIO $89.95
#15

3. DISKOPERATIVSYSTEM (DOS)

Fristående:

PERCOM DATA TX-99 SSSD INKL DR $289.95
#6

CORCOMP CC9900 MICRO EXP-SYS.32K,RS232, PIO,DOS $329
#15

CORCOMP CC99000 EXP-SYS 32K RS232 PIO DOS Utrymme för en drive. $?
#15

TIMPAC 32K RS232 DOS $499
#7

För X-BOX:

CORCOMP 9900 DISK CONTROLLER
CC DISK MANAGER $169.95
#15

4. EXPANSIONSSYSTEM

SE OVAN CC9900,CC99000,TIMPAC

5. OPERATIVSYSTEM

MORNINGSTAR CP/M PROCESSOR
CP/M-80 2.2 $595
BUSINESS MASTER PLUS $295
SUPERWRITER by Sorcim $295
PERSONAL PEARL DATABAS $295
THE RANDOM HOUSE STAVNINGKOL $50
CBASIC INTERPRETER $150
#4

6. PROGRAMSPRÅK

TI-FORTH SEK 250
#25

FÖRENINGENS FORTH SEK 250
#25

SST BASIC COMPILER $50
SST EXTENDED BASIC COMPILER $99
#17

7. MODULTILLVERKNING

KONVERTERING AV PROGRAM PÅ KASSETT OCH DISKETT TILL MODUL

MODEL I Peripheral Port Cartridge
8KB-48KB ROM
2KB RAM (DSR utrymme)

MODEL II Command Port Cartridge
10KB ROM eller
8KB ROM + 2 KB RAM
#10

CARTRIDGE PROGRAMMER $299.95
MANUAL $25.00
BLANK CARTRIDGES $19.95
#6

8. ÖVRIG HÄRDVARA

FOUNDATION 80*24 KOLUMN KORT
STATUSINFORMATION PÅ 25'E RADEN $240
#3

WIDGIT WD-01 PORT EXP. 3 MODULER $40

DISK-FIXER AS-02 (32K DISK-DRIVE) $40
#13

OPTICAL SCANNING READER OSCAR $79.95
STRECK-KOD SCANNER MED MJUKVARA
#6

WINCHESTER DISK 5 MBYTE $ 1899
10 MBYTE $ 2199
#14

9. ORDBEHANDLINGSPROGRAM

WORDMASTER X-BASIC $44.95
#8

TYPWRITER CAS $32
#21

PAGEWRITER CAS $19.95
#23

COMPANION 32K DISK XB $79
#26

MINI-WRITER MINI-MEM $19.95
#12

---

10. DATABASPROGRAM

COUNT-SIL SPREADSHEET CONSOLE $29.95
XB 32K $49.95
#5

NAME-IT MAIL-LIST DATABAS $32
#21

---

11. FILHANTERINGSPROGRAM

DISK-FIXER AS-02 $39.95
RECOVER LOST DATA
CHANGE ANY BYTE ON DISK etc.
#13

QUICK-COPIER 32K ED/ASSM $32.95
#12

MASTER DISK-FILE DISK $15
CATALOG DISKETTES/FILES
#21

DISK LIBRARY SYSTEM $19.95
CAP 1500 FILES 250 DISKS
#24

---

12. PROGRAMMERINGSHJÄLPMEDEL

SXB (SUPER EXTENDED BASIC) $99.95
(100 TMS9900 ASSEM. LANG. SUBROUTINES
FOR USE WITH XBASIC+32 KRAM+ED/AS)
NEWSLETTER INCLUDED IN PRICE
#16

HIGH LEVEL MATH EXT-003 $24.95
MATRISER DIFFEKVATIONER
MULTIPELINTEGRALER etc.
#18

DESIGN GRAPHICS $14.95
Sprite Editor
#19

SELECT A SPRITE $12.95
Sprite Editor
#20

SCREEN DUMP $12.00
#21

PROGRAM WRITER XB DISK $18.00
SKAPA BASIC-PROGRAM MED ORD-
BEHANDLARE.

DISASSEMBLER SPEC XB,MM,ED/AS $12.50
#22

****

FÖRSÄLJARE OCH TILLVERKARE

****

#1 MIKEL LABORATORIES INC.
17360 Gramercy Place GARDENA,
CA 90247-5212 (213) 532-3029

#2 MULTICOM INC.
P.O.Box 1693
SANDY, UTAH 84091 (801) 572-6272

#3 FOUNDATION COMPUTING
74 Claire Way
Tiburon, CA 94920 (415) 388-3840

#4 MORNINGSTAR SOFTWARE
4325 S.W. 109th Ave
Beaverton, OR 97005 (800) 824-2412

#5 SYSTEMS INTERFACE
1511 Merivale Road
Nepean, ONTARIO
K2G 3J3 CANADA (613) 232-2188

#6 UNISOURCE ELECTRONICS INC.
P.O.Box 62240
lubbock, TEXAS 79464

#7 TEX MICRO INC.
Titusville, FL 32783-5366
(305) 267-4513

#8 KCR Dept H2
Box 8128
Huntington WV 25705

#9 BACH COMPANY
760 San Antonio Road
PALO ALTO, CA 94303

#10 SUNWARE Ltd
Park Tower Bldg.,
Suite 140, 1617-27th St.
Lubbock, TEXAS 79405
(806) 747-2534

#11 CENTROPLEX COMPUTERS
RT.1 Box 458
Salado, TX 76571

#12 TEX-COMP
P.O.Box 33084
Granada Hills, CA 91344
(

#13 NAVARONE INDUSTRIES
510 Lawrence Expressway,
#800, Sunnyvale, CA 94086
(408) 866-8579

#14 MYARC INC.
P.O.Box 140
Basking Ridge, NJ 67920
(201) 766-1700

#15 TENEX
P.O.Box 6578
South Bend, IN 46660
(219) 277-7726

#16 J&KH SOFTWARE
2820 S.Abingdon St.
ARLINSTON, VA 22200
(703) 820-4131

#17 SST SOFTWARE INC
P.O.Box 26
Cedarburg, WI 53012
(414) 771-8415

#18 EUROWARE
117 Roxanne Ct.Suite 4
Walnut Creek, CA 94596

#19 SCOTT COMPWARE
5710 Lee Highway 18
Chattanooga, Tennessee 37421

#20 SMITH-WARE
6841 S.E. RAMONA St.
Portland, OR 97206

#21 EXTENDED SOFTWARE CO.
11987 Cedarcreek Dr.
Cincinnati, OH 45240
(513) 825-6645

#22 THE SOFTIES
7300 Gallagher, Suite 229
Edina, Minnesota 55435

#23 VMC SOFTWARE
P.O.Box 326
Cambaia Hts., NY 11411

#24 SEAGULL SOFTWARE
P.O.Box 611, Keyport
WASHINGTON 98345

#25 FÖRENINGEN PROGRAMBITEN
c/o Schibler
Wahlbergsgatan 6 nb
12146 JOHANNESHOV

#26 INTELPRO
5825 Baillargeon St.
Brossard,Quebec,
CANADA

#27 STIM 99
Hamiltons väg 54
S-302 41 HALMSTAD

#28 CBI (Computer Boss International)
Box 503
S-631 06 ESKILSTUNA 016/131020

#29 Magnussons Digitalservice
c/o Texas Intruments
Box 39103
S-100 54 STOCKHOLM 08/235480

#30 Blijenburgh Electronics
Drottninggatan 81
S-10420 STOCKHOLM 08/205054
08/541875 kont.

****

Till slut :
Tillverkningsrätten för alla TI-orginal delar är sålda till:
INTERTRANS CORPORATION
P.O.Box 61008,
DALLAS FORTH-WORTH,
TEXAS 75261
USA

■

Spara skärmen på kassett

av BENGT FAHLGREN

Detta program ger dig, som enbart har grundenheten, en möjlighet att spara en skärmsida på din kassett-bandspelare.

Programmet har i sin nuvarande form mycket begränsade möjligheter till redigering. Dessa kan liknas vid en skrivmaskin med raderingstangent.

Skärmen består av 32*24 tecken. Du kan förflytta markören mha piltangenterna. Markören kan dock tex inte flyttas direkt från kolumn 32 till kolumn 1.

Följande funktioner finns:
FCTN 7 = Lagra skärmen på CS1
FCTN 8 = Hämta skärmen från CS1

När du trycker på FCTN 7 läser programmet av skärmen med GCHAR och lagrar detta i fältet A(i,j). Eftersom det skulle ta 7-8 min att lagra dessa tal på kassett gör programmet om fältet till 8 st strängar A$(X) som sedan lagras på kassettbandet. Lagringen tar ca 4 min i TI-Basic och ca 3 min i X-Basic. Man kunde ha tänkt sig att direkt lagra texten i A$(X) men då uppstår två problem:

1) Utskriften av en skärmsida sker långsammare.
2) Det blir knepigare att lägga till fler funktioner.

Till sist...

Om du vill lägga till fler funktioner så tänk på att programmet reagerar allt långsammare.

Kom gärna med förslag hur programmet kan förbättras.(När tiden går mot noll och programmet fungerar "filar" man inte längre.) Eller skicka in den där geniala lösningen du redan gjort.

```
100 REM --------------------
110 REM     SCREENSAVER
120 REM --------------------
130 REM  BY BENGT FAHLGREN
140 REM --------------------
150 DIM A(24,32),A$(8)
160 CALL CLEAR
170 PRINT "        SCREEN-SAVER": : : : : : : : : :
180 PRINT " C 1984   BENGT FAHLGREN"
190 FOR I=1 TO 100
200 NEXT I
210 CALL CLEAR
220 REM --------------------
230 REM START OF MAIN LOOP
240 REM --------------------
250 FOR I=1 TO 24
260 FOR J=1 TO 32
270 CALL KEY(0,K,S)
280 CALL GCHAR(I,J,C)
290 CALL HCHAR(I,J,30)
300 CALL HCHAR(I,J,C)
310 IF S=0 THEN 270
320 IF K>11 THEN 470
330 ON K GOTO 580,270,270,270,270,1100,270,410,440,38
    0,350
350 IF I<=1 THEN 270
360 I=I-1
370 GOTO 270
380 IF I>=24 THEN 270
390 I=I+1
400 GOTO 270
410 IF J<=1 THEN 270
420 J=J-1
430 GOTO 270
440 IF J>=32 THEN 270
450 J=J+1
460 GOTO 270
470 REM
471 CALL HCHAR(I,J,K)
480 NEXT J
490 NEXT I
500 GOTO 250
510 REM --------------------
520 REM END OF MAIN LOOP
530 REM --------------------
540 REM
550 REM --------------------
560 REM LAGRA DATA TILL CS1
570 REM --------------------
580 FOR M=1 TO 24
590 FOR N=1 TO 32
600 CALL GCHAR(M,N,D)
610 A(M,N)=D
620 NEXT N
630 NEXT M
640 C1=1
650 C2=3
660 X=1
670 GOSUB 1610
680 C1=4
690 C2=6
700 X=2
710 GOSUB 1610
720 C1=7
730 C2=9
740 X=3
750 GOSUB 1610
760 C1=10
770 C2=12
780 X=4
790 GOSUB 1610
800 C1=13
810 C2=15
820 X=5
830 GOSUB 1610
840 C1=16
850 C2=18
860 X=6
870 GOSUB 1610
880 C1=19
890 C2=21
900 X=7
910 GOSUB 1610
920 C1=22
930 C2=24
940 X=8
950 GOSUB 1610
960 OPEN #1:"CS1",INTERNAL,OUTPUT,FIXED 128
970 FOR P=1 TO 8
980 PRINT #1:A$(P)
990 NEXT P
1000 CLOSE #1
1010 FOR I=1 TO 24
1020 FOR J=1 TO 32
1030 CALL HCHAR(I,J,A(I,J))
1040 NEXT J
1050 NEXT I
1060 GOTO 250
1070 REM --------------------
1080 REM HAMTA DATA FRAN CS1
1090 REM --------------------
1100 OPEN #1:"CS1",INTERNAL,INPUT ,FIXED 128
1110 FOR P=1 TO 8
1120 INPUT #1:A$(P)
1130 NEXT P
1140 CLOSE #1
1150 C1=1
1160 C2=3
1170 X=1
1180 GOSUB 1700
1190 C1=4
1200 C2=6
1210 X=2
1220 GOSUB 1700
```

```
1230 C1=7
1240 C2=9
1250 X=3
1260 GOSUB 1700
1270 C1=10
1280 C2=12
1290 X=4
1300 GOSUB 1700
1310 C1=13
1320 C2=15
1330 X=5
1340 GOSUB 1700
1350 C1=16
1360 C2=18
1370 X=6
1380 GOSUB 1700
1390 C1=19
1400 C2=21
1410 X=7
1420 GOSUB 1700
1430 C1=22
1440 C2=24
1450 X=8
1460 GOSUB 1700
1470 FOR R=1 TO 24
1480 FOR S=1 TO 32
1490 CALL HCHAR(R,S,A(R,S))
1500 NEXT S
1510 NEXT R
1520 REM goto main loop
1530 GOTO 250
1540 REM -------------------
1550 REM END OF PROGRAM
1560 REM -------------------
1570 END
1580 REM -------------------
1590 REM SUBR FLD TO STRING
1600 REM -------------------
1610 FOR C=C1 TO C2
1620 FOR R=1 TO 32
1630 A$(X)=A$(X)&CHR$(A(C,R))
1640 NEXT R
1650 NEXT C
1660 RETURN
1670 REM -------------------
1680 REM SUBR STR TO FIELD
1690 REM -------------------
1700 FOR C=C1 TO C2
1710 FOR R=1 TO 32
1720 A(C,R)=ASC(A$(X))
1730 IF LEN(A$(X))=1 THEN 1780
1740 T$=SEG$(A$(X),2,LEN(A$(X)))
1750 A$(X)=T$
1760 NEXT R
1770 NEXT C
1780 RETURN
```

FULL SCREEN = FULL SKÄRM

Testbild för bildjustering

av Bengt Fahlgren

Ofta ligger en del av kolumn 1 i blockgrafiken utanför bildskärmen.
Dessa tvivelaktiga programrader tillverkades i syfte att underlätta injusteringen av min TV-bild och här ser ni mitt enkla program. Håll till godo.

```
100 REM ***************
110 REM *   TESTBILD   *
120 REM ***************
130 CALL SCREEN(8)
140 CALL CLEAR
150 PRINT
160 PRINT "       DATOR TESTBILD"
170 X=40
180 CALL CHAR(X,"FFFF010101010101")
190 CALL CHAR(X+1,"010101010101FFFF")
200 CALL CHAR(X+2,"808080FFFF808080")
210 CALL CHAR(X+3,"010101FFFF010101")
220 CALL CHAR(X+4,"FFFF808080808080")
230 CALL CHAR(X+5,"FFFF010101010101")
240 CALL CHAR(X+6,"010101010101FFFF")
250 CALL CHAR(X+7,"808080808080FFFF")
260 REM ram
270 CALL COLOR(2,2,1)
280 CALL HCHAR(1,1,44,1)
290 CALL HCHAR(1,2,40,30)
300 CALL HCHAR(1,32,45,1)
310 CALL VCHAR(2,1,42,22)
320 CALL VCHAR(2,32,43,22)
330 CALL HCHAR(24,1,47,1)
340 CALL HCHAR(24,2,41,30)
350 CALL HCHAR(24,32,46,1)
360 REM ****************
370 CALL CHAR(48,"080808FFFF080808")
380 CALL CHAR(49,"FFFFFFFFFFFFFFFF")
390 CALL COLOR(1,2,1)
400 CALL COLOR(14,6,6)
410 CALL COLOR(15,10,10)
420 CALL COLOR(16,4,4)
430 FOR X=3 TO 22
440 CALL HCHAR(X,7,49,20)
450 NEXT X
460 FOR X=4 TO 21
470 CALL HCHAR(X,8,32,18)
480 NEXT X
490 FOR X=136 TO 152 STEP 8
500 CALL CHAR(X,"FFFFFFFFFFFFFFFF")
510 NEXT X
520 REM colord boxes
530 FOR X=4 TO 9
540 CALL HCHAR(X,8,136,6)
550 NEXT X
560 FOR X=10 TO 15
570 CALL HCHAR(X,8,152,6)
580 NEXT X
590 FOR X=16 TO 21
600 CALL HCHAR(X,8,144,6)
610 NEXT X
620 FOR X=4 TO 9
630 CALL HCHAR(X,14,144,6)
640 NEXT X
650 FOR X=10 TO 15
660 CALL HCHAR(X,14,136,6)
670 NEXT X
680 FOR X=16 TO 21
690 CALL HCHAR(X,14,152,6)
700 NEXT X
710 FOR X=4 TO 9
720 CALL HCHAR(X,20,152,6)
730 NEXT X
740 FOR X=10 TO 15
750 CALL HCHAR(X,20,144,6)
760 NEXT X
770 FOR X=16 TO 21
780 CALL HCHAR(X,20,136,6)
790 NEXT X
800 REM UPPLOESNING
810 CALL CHAR(128,"AAAAAAAAAAAAAAAA")
820 CALL CHAR(129,"CCCCCCCCCCCCCCCC")
830 CALL CHAR(130,"9292929292929292")
840 CALL CHAR(131,"4949494949494949")
850 CALL CHAR(132,"2424242424242424")
860 CALL CHAR(133,"DBDBDBDBDBDBDBDB")
870 CALL CHAR(134,"6D6D6D6D6D6D6D6D")
880 CALL CHAR(135,"B6B6B6B6B6B6B6B6")
890 CALL COLOR(13,2,8)
900 FOR X=3 TO 5
910 CALL VCHAR(3,X,128,10)
920 NEXT X
930 CALL VCHAR(13,3,130,10)
940 CALL VCHAR(13,4,131,10)
950 CALL VCHAR(13,5,132,10)
960 FOR X=28 TO 30
970 CALL VCHAR(13,X,129,10)
980 NEXT X
990 CALL VCHAR(3,28,133,10)
1000 CALL VCHAR(3,29,134,10)
1010 CALL VCHAR(3,30,135,10)
1020 GOTO 1020
```

Dataöverföring i praktiken

Lennart Lindberg
tfn 08/87 80 50

Datavision, Teletex, modem, baud, handskakningsrutin - vi känner igen orden, som blir allt vanligare. Det handlar om överföring av data mellan datorer, vanligen via telenätet. Jag tänker här berätta hur det kan te sig i praktiken.

Jag är en av dem som gör det nämligen, och det är faktiskt inte alls märkvärdigt.

I mitt arbete använder jag dels en enkel mikrodator som med ett 1200 bauds modem från televerket är anslutet till telenätet, dels också en synkron höghastighetsterminal utan egen intelligens, som via ett lokalt nät är kopplad till en stor dator.

Hemma har jag en 99-a med ett RS232, som via ett av televerkets 300 bauds modem kan kopplas till telenätet. Det är den jag tänker berätta litet om. De första nämnde jag för att klargöra mina jämförelsemöjligheter.

Michael Öhman och jag talades vid på telefon för en tid sedan. Modemet hade då stått hemma ett tag med en anslutningssladd kopplad mellan modemet och RS232-kortet. Jag hade försökt få förbindelse med en stordator via ett nummer jag skaffat men inte lyckats. Michel får mig att dra isär kontakten till sladden och titta hur trådarna är kopplade. De låg fel! TI har en alldeles egen standard, som inte ger problem med TI's egna grejor men kan bli fel när man kopplar till andra grejor - som tex televerkets modem. Jag fick det omlött av en händig granne. Sedan körde vi.

Det börjar med att man har vanlig telefonförbindelse och talar med varandra. Det är tex lämpligt att komma överens om vem som skall vara A-kanal och vem som skall vara B-kanal vid överföringen. Sedan laddar man ett lämpligt program i sin maskin. Jag körde först med X-Basic-modulen i datorn och gjorde att litet program på några rader som öppnade en fil via RS232 och läste den filen med INPUT #. Michael gjorde ett program som sände via hans RS232. Därefter skrev vi båda RUN, dock utan att trycka på ENTER ännu. Först kopplade vi in våra modem - jag genom att trycka på en knapp och Michael genom att lägga sin telefonlur i modemet - han har ett akustiskt modem nämligen . Vi hade sett till att han kopplade sig som A och jag som B. När man trycker igång lyser ett par lampor på modemen som tecken på att det finns en fungerande bärvåg. Sedan tryckte vi på ENTER och programmen började arbeta. Michaels program sände och mitt tog emot. När sändningen var klar bröt Michael kontakten på sitt modem, varvid en lampa slocknade på mitt och jag tog telefonluren. På min skärm hade jag då den text Michael hade sänt!

Efter det här har vi gjort om det i större skala. I styrelsen jobbar vi med disketter en del för sammanträdesdokumentation, tidningsartiklar mm och Michael sände häromdagen till mig en ganska lång text som han hade på en diskett. Det tog åtta minuter. Alternativet hade varit posten. Nu kunde jag tjäna två dagar, vilket var viktigt just då. Jag använde min TI Writer-modul och gav helt enkelt kommandot 'LOAD FILE' samt definierade filen som 'RS232.LF'. Jag fick hela texten i minnet precis som om jag hade läst in den från min egen skivenhet och kunde direkt börja redigera texten!

Jag har också provat att koppla ihop mig med en kamrat till min grabb, som har en VIC 64. Inga problem alls! Vi använde ungefär samma procedur som Michael och jag. Det han skrev på sina tangenter kom på min skärm och tvärt om. Då använde jag Terminal Emulator II som sändande och mottagande program. Han använde VIC's nyligen utsläppta modem, som tydligen är ganska kvalificerat.

Och dyrt, tyckte han.

En annan gång med Michael kopplade vi upp oss med X-Basic, varefter Michael till mig sände över ett program han skrivit i Assembler. Jag förde över det på disk, bytte till Assembler-modulen, laddade och körde igång programmet. Michael hade samma program hos sig och vi körde en förbindelse i full duplex, där det han skrev visade sig på övre delen av min skärm och på nedre delen av hans och tvärtom. Full duplex innebar att vi kunde skriva samtidigt!

Jag låg sjuk hemma
härförleden i lunginflammation. Det blev träkigt efter ett par dagar och jag kopplade då upp mig med 99-an till den stordator jag använder i mitt arbete och kommunicerade med den hemifrån. Utmärkt bra och inga som helst problem! Jag använde Terminal Emulator II. Det där gör jag numera tämligen regelbundet och kan då hålla kontakt med det Portacom-system vi använder. Det innebär att vi skickar elektroniska brev till varandra via en stordator och jag kan alltså hålla den kontakten även hemifrån nu, vilket är bra när jag reser utan att åka till kontoret den dagen eller kanske på flera dagar.

När jag ringer till stordatorn behöver jag inte tala med någon på telefonen. Efter det att samtalet kopplats upp får jag en hög ton, som innebär att stordatorn är klar att ta emot mig. Jag kopplar då in mig som A-kanal och förbindelsen är etablerad. Sedan måste jag veta vilka koder jag skall sända för att bli accepterad av det program som ligger i stordatorn och snurrar. De där koderna leder också så småningom till att jag får en räkning på den tid jag använt stordatorn.

Svårare är det inte. Men det gäller att ha en sladd som är rätt kopplad.

■■

FÖRENINGENS TILLBEHÖRSFÖRSÄLJNING

Följande finns att köpa för medlemmar genom att motsvarande belopp sätts in på postgiro 19 83 00 - 6.

Användartips med Mini Memory		60:-
FORTH Olika versioner, se annons på annan plats i tidningen (sid 23)		
Nittinian T-tröja		40:-
99'er Magazine nr 12/82 nr 1-5,7-9/83 (per styck)		20:-
Nittinian, årgång 1983		80:-
Astronomical Formulae for Calculators		Slut
Programbiten, årgång 1983	Kalkylatorer	80:-
Programbiten, årgång 1982	-"-	80:-
Programbiten, årgång 1981	-"-	60:-
Programbiten, årgång 1980	-"-	60:-
Programbiten, årgång 1978/79	-"-	60:-
Programbiten, fem årgångar 1978-1983		280:-
Katalog med belgiska och engelska program för räknare TI-57, TI-58, TI-59		20:-
Föreningens programmeringsblanketter (TI-59), olika typer, block om 50 blanketter (se pb 83-1 sidan 30)		11:-
Patenthandlingar TI-59		25:-
40 st tomma magnetkort med plånbok		150:-
Tom magnetkortsplånbok		10:-

→→

HURRA vad vi är bra !!!

Östvik 22/11-84

Programbiten

Hurra va vi är bra
PETER ODELRYD och jag
NEDANSTÅENDE PROGRAM KÖRDES VIA MODEM 300 BAUD
DEN 21-11 1984 MELLAN ÖSTVIK (0910) OCH HANDEN (08)
TE-II MODULEN ANVÄNDES!

DET BÄSTA VAR ATT PETER BEHÖVDE EFTERÅT BARA SKRIVA RUN "DSK1. PROV" I EK.BASIC.
(PETER BEHÖVDE EJ SKRIVA PROGRAMMET FÖRST)
ÖSTVIK SÄNDE PROGRAMMET, HANDEN TOG EMOT.

```
90 CALL CLEAR
100 INPUT "ANGE:  'DEVICE. OCH FILNAMN!'":FILE$
120 OPEN #1:FILE$,INPUT ,FIXED 80,DISPLAY
130 INPUT "PRINTER (Y/N)? ":C$
135 CALL CLEAR
140 IF C$="N" THEN 205
150 DISPLAY AT(12,2)BEEP:"PRINTER:    RS232.BA=4800"
160 FOR I=1 TO 1000 :: NEXT I
165 CALL CLEAR
166 INPUT "ANTAL TECKEN PER RAD? ":ANTAL
167 CALL CLEAR
170 OPEN #2:"RS232.BA=4800",OUTPUT,DISPLAY
180 LINPUT #1:A$
190 IF EOF(1)THEN 260
200 PRINT #2:SEG$(A$,1,ANTAL)
202 GOTO 180
205 CALL CLEAR :: DISPLAY AT(12,1)BEEP:"TRYCK EN TANGENT=RADMATNING"
206 FOR I=1 TO 1000 :: NEXT I
207 CALL CLEAR
220 LINPUT #1:A$
230 IF EOF(1)THEN 260
235 T$="                40 MELLAN SLAG          "
236 TT$="                 - 4 -                  "
237 Y$=T$&TT$
238 IF SEG$(A$,40,80)=SEG$(Y$,40,80)THEN X=40 ELSE X=80
240 PRINT SEG$(A$,1,X)
242 CALL KEY(0,K,S)
244 IF S=0 THEN 242
250 GOTO 220
260 END
```

ÄR VI DOM FÖRSTA ATT HA HÖRT ÖVER ETT PROGRAM* MELLAN TVÅ TI-99/4A ? (I SVERIGE) OM INTE! VI ÄR VÄL FÖRSTA MED DET AVSTÅNDET 6/A 100 MIL.

* PROGRAM = PROGRAM SOM GÅR ATT KÖRA DIREKT AV DEN SOM MOTTAGIT DET.

Falle Andersson Östvik.
0910-22280 OM NÅGON VILL UTÖKA AVSTÅNDET!

DATAVISION MODUL
Modulen gör att du kan kommunicera med DATAVISION eller liknande baser.
Utskrift för printer finns tillgänglig, sparning av skärmar på kassett eller diskett.
Nödvändig utrustning:
RS232 interface
Televerkets modem eller liknade
Modulen kommer med instruktioner om hur du kopplar upp dig med Televerkets modem.

Pris: 565:-

16 KRAM CMOS MODUL
utan hölje. Modulen kan simulera ett ROM, inladdning sker från T.ex FORTH vilken medför att minimum utrustning är X-basic, MINIMENORY, eller ED/ASM och 32 K minnesutökning. Ram'et är batteribackupat så programmet ligger kvar minst 2 veckor. Denna modul används i princip för att testa ut hur egna program i modulform fungerar innan man bränner prommar vilket blir klart jobbigare.

Pris: 875:-

DISK DRIVAR

Fabriksnya drivar från Shugart. Kan direkt kopplas in i expansionslåda.

Enkelsidig: 1200:-
Dubbelsidig: 1600:-

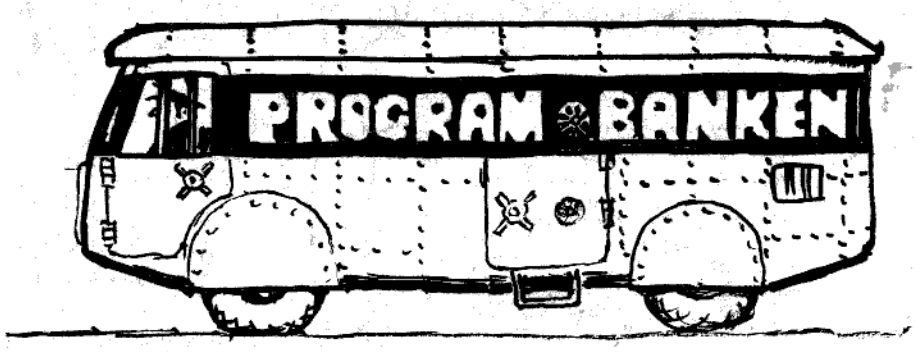
Specialerbjudande: Halvhöjdsdrivar. Två stycken får plats i expansionslådan. Grenkontakt och monteringsanvisning medföljer.

Pris per PAR: 3500:-

Även begagnade enkelsidiga drivar finns.

Eventuell frakt tillkommer.

Björn Gustavsson
Lagfartsvägen 46
145 58 Norsborg



Här visas en lista på de tillgängliga programmen i Programbanken. Programmen kan numera köpas. Priset är uppdelat i två delar, dels en startkostnad som täcker kostnaden för porto och mediet, dels en kopieringsavgift för varje program.

Startkostnad, kassett	**35:-**
Startkostnad, diskett	**55:-**
Kopieringsavgift, kassett	**15:-**
Kopieringsavgift, diskett	**10:-**

Som förut får du tre program i utbyte när du skickar in ett BASIC-program.

De två första siffrorna i programnumret anger vilken diskett programmet ligger på.
De nästa två siffrorna anger programkategori se nedan
Femte siffran anger ursprungsland:
1 = USA, 2 = Holland, 7 = Sverige.
De tre sista siffrorna är ett löpnummer och bokstaven som finns sist anger vilket språk som används i programmet: E = Eng., H = Holl., F = Fra., S = Svenska.
Bokstäverna efter numret talar om vilken utrustning som krävs: X = Ext. Basic, D = Diskett, P = Printer E = Minnesexp., J = Joystick, C = Call Files (1) måste användas om man har disk. Ett + anger att programmet är delat i flera delar.

Programkategori:
14 = Ekonomi
20 = Regression, Kurvanpassning
21 = Statistik, Varians
29 = Statistik, Sannolikhet
30 = Linjär algebra
39 = Allmän matamatik
65 = Elektronik
78 = Astronomi
90 = Praktiska programhjälpmedel
91 = Spel
92 = Utbildning
96 = Musik
97 = Demo och Musik
99 = Övrigt

```
01201001E ___ LINEAR REGRESSION
01201002E ___ TREND LINES ANALY
01391001E ___ 8 MATH PROGRAMS
01391002E ___ SIMULT. EQUATIONS
01651001E ___ ELECTRONICS
01781001E ___ GEOSTAT. SATELITE
01901001E __P WORD-PROCESSOR 1
01901002E ___ MAILING LIST
01911028H ___ CAR DRIVER
01911029H ___ ANIMALS
01911030H ___ SPACE INVADER
01921006H __X GENETICS
01971001H __C 99/4A DEMO
01991001E __X ANCESTRAL FILER
01991002H ___ CALENDAR W. PRINT
01992001E ___ MOON CYCLE
02911001E ___ MASTERMIND
02911002E __X SCHMOO
02911003E ___ STARTRECK
02911004E ___ AIR - SEA BATTLE
02911005E ___ QUEENS
02911006E ___ OTHELLO 2-SPELARE
02911007E ___ CHASE
02911008E ___ PUZZLE 15
02911009E ___ NUMBERS AWAY
02911010E ___ AWARI
02911011E ___ KENO
02911012E _+D STARGUARD
03911013E ___ MINER
03911014E __C YAHTZEE
03911015E ___ BACKGAMMON
03911016E ___ 3D TIC-TAC-TOE
03911017E __X BLACKJACK
03911018E __X NOT ONE
03911019E __X POKER
03911021E ___ DONKEY'S TAIL
03911023E __X SPACE GEMS
03911024E ___ SCORE FOUR
04911025E ___ STARTREK 2
04911026E __X CHICKEN HELPER
04911027E ___ CATAPULT
04911031E ___ ISOLA
04912001E ___ SQUARES
04912002E ___ ECHO CHAMBER
04921001E ___ COLORMATCH
04921002E ___ TEST TUBE
04921003E __X RELATIVE IQ TEST
04921004E ___ ALGEBRA
04921005E ___ PROJECTILE PROBL.
05912003 __X SEA AIR MISSILES
05962001 __C A STRAUSS WALTZ
05962002 ___ NEVER ON A SUNDAY
05962003 __X BERCEUSE
05962004 _XC BEETHOVEN #5
05962005 __X SERENADE
05962006 +DX AMAZING GRACE
05962007 ___ BEETHOVEN OPUS 27
05962008 +DX TIME IN A BOOTLE
05962009 +DX LIGHT UP MY LIFE
05962010 __N BUMBLE-BOOGIE
06211001E __C GEN. STATISTICS
06301001E ___ MATRIS INV/MULT.
06901003E __X TOTAL DISK KAT.
06901004E ___ DISK KAT M PRINT
06911032E ___ LAST ROBOT
06911033E ___ ANTI AIR GUN
06911034H __X L-GAME
06912004E ___ SLALOM
06912005E ___ KILLER
06962011 _+D FIDDLER ON ROOF
06962012 ___ BACH #3 MINUET
06971002E ___ KALEIDOSCOPE
06972001 ___ THE PINK PANTHER
06972002 ___ SINE WAVE
06972003 __P LOVE!
06972005 __X SPRITE DEMO #1
06972006 ___ SWEETHEART
07141001H __X ANNUITIES
07901005H ___ MORSE
07902002 __P BANNER (BIG TEXT)
07911035H __X WARI
07911036H ___ LABYRINTH
07911037H __X ELIZA
07911038H ___ SECRET THOUGHTS
07911039H __X CATCH THE CHAR.
07911040H __X BLACKJACK
07911041H __X SPACE BATTLE
07911042H __X MAGIC WORD
07911043H ___ JACKPOT
07911044H ___ LABYRINTH 2
07921007E ___ PAST TENSE
07962013 ___ LOOKING THROUGH U
07971003H ___ BIG CHARACTERS
08901006E _XE UNIV. TITLE PAGE
08902003 ___ LARGE CHARACTERS
08911039E __X SPRITE CHASE
08911046E __X CHECKERS
08911049E __X SWORDS & SORCERY
08911051E __X EGGWARS
08961001E __X KILLING ME SOFTLY
08962014 __X CHOPIN OP 10 #3
09291001H ___ PROBABILITY
09391003H ___ DIFFERENTIAL EQ.
09911052E ___ DEEP SPACE
09911053F ___ ATOMES
09911054H ___ PLANETARY LANDER
09911056H __X LUNAR LANDER
09911057F ___ TOWERS OF HANOI
09911058E ___ BREAKOUT
09911059H __J DRAWING ON SCREEN
09911060H _JX TREASURE HUNT
09912008F __X ASTEROID
09912009 ___ BATTLE STAR
09991003F ___ BIORYTHM
10392002 ___ PRIME NUMBERS
10901007H ___ LOTTO (HOLL.)
10901008E ___ VERY LARGE CHAR.
10901009E ___ KEY DEFINER 99/4
10901010E __C TEXTPRO (99/4)
10911061H __X KERMIT
10911062H _JX FIND THE GUN
10911063E ___ CHASE A TARGET
10911064H ___ RUSSIAN ROULETTE
10911065E ___ CAMEL
10921008E ___ COLOR FRACTIONS
11901011S ___ STAPELDIAGRAM
11911055S __X HANGNING
11917001S _JX INKRAEKTARNA
11917002S _JX PAERON
```