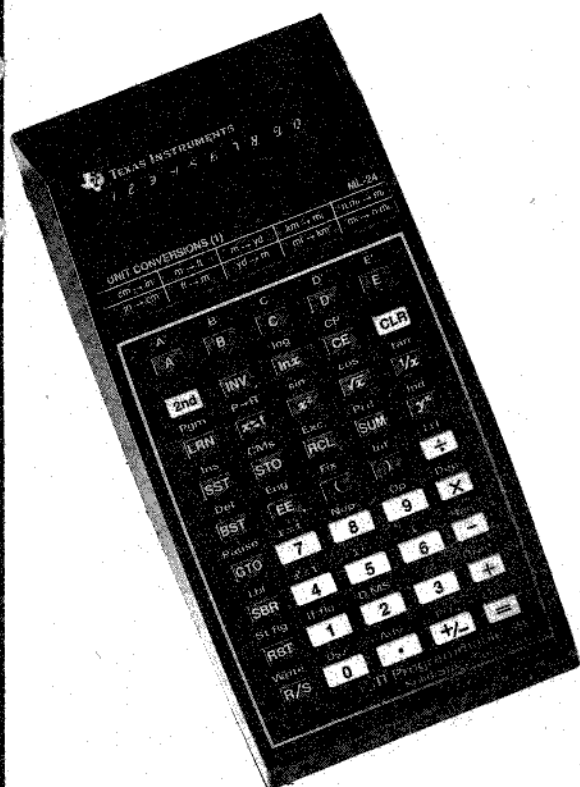
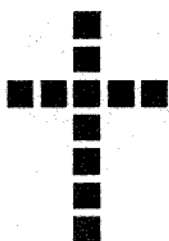


FORGÅM BILLEN SÖS



Innehåll

Insänt	3
Kontrollciffr	4
Egen modul? Nej, tyvärr!	5
Utmaningen	6
Numerisk integration 6; Numerisk till alfakonverterare 8; Bachetrepresenta- tion 8; Påskens datum 8; Problem 9	
Hexkoder på TI-57	12
Minfält	14

DYRE MEDLEM

Så har då tillverkningen av TI-59 lagts ned. Man kan undra om det inte gjorts i förtid. Efterfrågan finns fortfarande, och åtminstone TI själva har ännu ingen likvärdig eller bättre efterföljare. CC-40 ligger fortfarande i praktiken på is eftersom tillbehören inte är klara och program således inte kan lagras.

Tyngdpunkten i föreningens verksamhet ligger nu alltmer på hemdatorn TI-99/4A, som ju har ett eget medlemsblad. Personligen hoppas jag ändå att vi skall kunna fortsätta med att i någon form behandla våra gamla räknare, och vid ett samtal med Robert Prins för inte så länge sedan ställde han i utsikt att i mån av tid fortsätta med "Utmaningen".

På årsmötet i januari väljer vi styrelse för 1984 för hela Programbiten. Tyvärr blir det ganska många i den gamla styrelsen inkl. mig själv som inte har möjlighet att ställa upp för återval. Eftersom ju föreningens funktion ju beror på styrelsen hoppas jag både gamla och nya medlemmar kan åta sig en post i den.

Vi har just nu framför oss den vädermässigt sämsta delen av året. Låt oss göra den till den datorhobbymässigt bästa delen!

Med bästa hälsningar

Lars Hedlund

Lars Hedlund, ordf

* LICENSNUMMER *
* 82100488 *
* *****

PROGRAMBITEN och FÖRENINGEN PROGRAMBITEN
c/o Hedlund

Årstavägen 27 6 tr
121 68 JOHANNESHOV

För KOMMERSELLT bruk gäller följande:

Mångfaldigandet av innehållet i denna skrift, helt eller delvis, är enligt lag om upphovsrätt av den 30 december 1960 förbjudet utan medgivande av FÖRENINGEN PROGRAMBITEN. Förbudet gäller varje form av mångfaldigande genom tryckning, duplicering, stencilering, bandinspelning, magnetkortsinspelning etc.

Tryck: K-TRYCK AB STOCKHOLM

Redaktören...

32 - 24 - 16 är en dyster talserie. Den visar utvecklingen av sidantalet i Programbiten under mitt redaktörskap. Skuldfrågan låter jag andra diskutera. Jag vill bara säga att jag har upplevt det svårare att få in originalmanus, än vad jag hade hoppats.

Jag vill även meddela att platsen som redaktör för Programbiten är ledig fr o m nyår. Orsakerna till att jag slutar är flera, bl a har mitt intresse för TI 58/59 svalnat betydligt sedan jag jag började använda datorer. Om Programbiten ska fortsätta att ges ut, är det viktigt att den nye redaktören tillsätts snabbt. Intresserade bör därför snarast ge sig till kända.

Bo Nordh

MEDLEMS ANNONSER

Säljes: TI-59 + PC 100 C + M/U-modul 2.000:-.
Endast skrivaren 900:-. Tel 0303-101 82 Jan.

Medlemsträff

Nu är det dags igen! Lördagen den tredje december 1983 klockan 14 träffas vi i Dalens träffcenter. Dit kommer du enklast med tunnelbana till Sandborg. Den här gången är även Nittionio-folket med. Det kan ge rika tillfällen till diskussion om föreningens kommande inriktning. Väl mött!

FÖRENINGENS TILLBEHÖRSFÖRSÄLJNING

Följande finns att köpa för medlemmar genom att motsvarande belopp insättes på postgiro 430 01 59 - 3:

Programbiten, årgång 1982	100,-
-"- 1981	80,-
-"- 1980	60,-
-"- 1978/79	60,-
-"- fyra årgångar 1978/79 - 1982	250,-
Patenthandlingar TI-59	25,-
Katalog med belgiska och engelska program	20,-
Astronomical Formulae for Calculators (viss leveranstid)	55,-
40 st tomma magnetkort och plånbok	(OBS HÖJT PRIS!) 130,-
Tom magnetkortsplånbok	10,-
Föreningens programmeringsblanketter, olika typer, block om 50 blanketter (se PB 79-3&4)	11,-
Sourcebook är slutsåld.	

INSÄNT



Föreningen Programbiten
c/o Hedlund
Årstavägen 27 6 tr
121 68 JOHANNESHOV

Då vi i Programbiten 83-1 fick syn på artikeln "Benchmarks" frapperades vi utav den undermåliga HP-programmeringen. När sedan Programbiten 83-2 kom ut så var Lars Nilssons program en klar förbättring. Men vi har en del tips att komma med, både till Lars Nilsson och andra HP-bitna TI-fantaster. T.ex. BM 1 kan faktiskt snabbas upp ytterligare 14.5 sekunder (15%). I BM 6 blir tidsvinsten vid optimering hela 34 sekunder (3%).

För att ni skall få lite mer inblick i hur vi har gått tillväga vid optimeringen av HP 41C/CV programmen måste en del begrepp förklaras:

1. "LILLA B"

HP 41 C/CV har, liksom alla programmerbara räknare och datorer, en mängd interna register. "Lilla b" är det register som på en dator motsvaras av programpekaren (samt delar av returnadress-stacken). Genom kommandot Rcl b fås värdet av programpekaren, som nu innehåller nästa radnummer. Om vi senare i vårt program tar detta värde och på nytt lagrar det i b-registret genom Sto b, får vi ett blixtnabbt hopp till en simulerad label (egentligen absolutadress).

2. "E 3"

Detta är helt enkelt syntetisk programmering av instruktionen "1 E 3". Den syntetiska varianten sparar en byte och går något fortare än sin konventionella motsvarighet, då anrop av siffror tar lång tid på HP 41C/CV.

3. " " " "

Denna funktion går under benämningen "Text 0" och fyller samma uppgift i HP:n som "Nop" i TI 57-59. Vi har låtit "Text 0" ersätta CLA i BM 3 - BM 8. Detta av två orsaker:
Dels för att ALPHA-registret inte raderas och dels därför att den exekveras snabbare. "Text 0" meddelar räknaren att en teckensträng med längden noll följer i programminnet.

4. TIDSSKILLNAD

Om vi nu sätter in ovan nämnda funktioner, på alla ställen där det ger en positiv effekt, får vi nya exekveringstider på BM 1 - BM 8.

BM	Ver. 1	Ver. 2	Ver. 3
1	97	96	81.5
2	315	98	98
3	620	360	347
4	940	570	556
5	1010	630	626
6	1590	1010	976
7	1930	1230	1215
8	1230	1050	1041
medel	966	630.5	618.5

BM 1: E 3 RCL b DSE Y STO b END

BM 2: E 3 ENTER CLX LBL 00 ISG X
" " ENTER ENTER 2 / 3 * 4 + 5 -
STO 6 RCL b DSE L STO b R! R!
X<Y? GTO 00 END

Då syntetisk programmering på HP 41C/CV är avsevärt mycket svårare att förklara och utföra än på TI 58/59, visar vi bara hur koderna skapas, inte varför. För djupare studier i ämnet rekommenderas boken "Synthetic programming on the HP 41C" av William C Wicks å det varmaste.

I. Vi skapar en så kallad "byte-grabber"

a. Master clear, asignera "+" till "Log".

b. Asignera "Del" till "Log".

KONTROLL- SIFFRA

Angående Hans Lindgrens utomordentliga program för beräkning av kontrollsiffra enligt Modul-10 metoden. Här bifogas två varianter av Hans program. Den första är gjord så kort och så snabbt som jag kunnat. Medan den andra är en s.k. "äkta subrutin" dvs stör ej pågående beräkningar. Förbättringen ligger i att inga minnen (förutom T-registret) används. Detta medför att programmet blir en aning snabbare.

VARIANT 1

```
000. Lbl A   x   .   0   1   x   INT X:T 7
010. 9   0   +   .   4   )   INT +   0   X:T
020. INV EQ A   )   x   .   1   )   INV INT
030. x   1   0   )   RTN
035.
```

VARIANT 2

```
000. Lbl A   (   (   CE x   .   0   1   x
010. INT X:T 7   9   0   +   .   4   )   INT
020. +   0   X:T INV EQ 0   03   -   (   CE
030. x   .   1   )   INT x   1   0   )   RTN
040.
```

Sven-Arne Wallin
N. Parkg 5
343 00 ALMHULT

3 c. Prgm catalog 1, R/S (omedelbart så att ingenting hinner visas) ALPHA <- ALPHA GTO .005 USER Log (som egentligen är "Del") 003 ALPHA ?AAAAAA ALPHA PRGM GTO ..

d. I user-mode skall nu "Ln" visa XROM 28,63.

II. För att vi nu skall kunna skapa någon syntetisk instruktion, krävs det att minst en byte är upptagen, nämligen den innan position där den syntetiska funktionen skall placeras. Detta villkor gäller endast för den första syntetiska instruktionen i en grupp.

III. Sekvenserna för att skapa de olika syntetiska funktionerna är:

PRGM (t.ex. ENTER)

a. EEX 3 Bst USER Ln Ln <- <- <- SST
Nu har "E 3" skapats.

b. STO IND T Bst USER Ln <- SST " " "
har nu skapats.

c. STO IND 16 MEAN Bst Bst Ln <- SST
Nu har Rcl b kommit till.

d. STO IND 17 MEAN BST BST LN <- SST
Nu har vi även fått fram Sto b.

IV. Om vi till exempel vill slå in BM 6: Utför steg I a - d, PRGM, steg III a - d. Du har nu knappat in samtliga syntetiska funktioner i följden E 3 " " Rcl b Sto b. Därefter är det bara att

mata in de vanliga kommandona och utföra Sst när någon av de syntetiska instruktionerna dyker i displayen.

Vi hoppas att vi har gett HP 41C/CV lite mer rättvisa, samt att vi har lyckats belysa vissa delar av den relativt invecklade syntetiska programmeringen. Förhoppningsvis har detta bidrag varit till nytta och nöje för någon flitig hacker.

Vid frågor och/eller behov av listnigar, skriv till:

Krister Norrman
Gräslösstigen 12
296 00 ÅHUS

Claes Carlstedt
Mandelblomsstigen 2
296 00 ÅHUS

Bifoga ett frankerat svarskuvert.

REDAKTIONENS KOMMENTAR

Jag vill tacka bidragslämnarna för deras brev. Skulden för den "undermåliga HP-programmeringen" får läggas på mina axlar. Den torde bl.a. bero på att jag aldrig har varit i närheten av HP 41-manualen. Lars Nilsson kände till möjligheterna med syntetisk programmering men valde att inte använda sig av dem.

I ovanstående brev har vissa specialtecken blivit utbytta: pil snett upp till höger har ersatts med "!" och pil åt vänster återges som "<-".

Egen modul? Nej tyvärr!

Lars Hedlund

Det blev alltså aldrig någon TI-59-användarnas "egen modul". Följande rader får ses som en nostalgisk betraktelse över vad vi "skulle kunna ha gjort" i form av några uppgifter om vad HP-41-användarna presterat.

I Programbiten tog vi första gången upp idén hösten 1980. TI-59 var då tre år gammal och några utländska klubbar hade tydligen haft frågan uppe men avstått. Maurice Swinnen i USA sände dock efter brevväxling med oss ut en förfrågan till sina medlemmar, och det resultatet fick vi ta del av och det publicerades i nr 81-1. Men sedan dog intresset ut; vi märkte här att vi inte orkade driva saken och i USA avvaktade Maurice Swinnen då han hört rykten om att en ny räknare var på gång - den sedermera nedlagda TI-88.

Det är flera saker jag måste tillstå, bl a att jag redan från början borde ha läst PPC Calculator Journal och följt HP-arnas framsteg och vedermödor, och att jag tydligen hade mycket orealistiska idéer om hur mycket tid som skulle gå åt för att sammanställa en modul. När jag sent omsider läst om HP-modulens historia kommer jag till misstanken att det kanske var lika bra att vi aldrig satte i gång! Richard Nelson anser t ex att han ensam kan redovisa 20.000 timmar!

Richard Nelson, nu redaktör för HP-användarnas PPC Calculator Journal, hade redan i början av 1977 funderingar på moduler, långt innan TI-59 hade släppts ut och ännu mer före HP-41C som först presenterades i juli 1979. (Det gällde då National NS 7100 vars produktion lades ned då TI-59 presenterades i juli 1977.) Richard hade t o m kontakt med TI:s "software manager" som var allvarligt intresserad. Men det ledde inte till något.

När HP-41C sedan kom inriktades strävandena redan från början på att användarna skulle bestämma hur "modulen" skulle se ut, och förslaget publicerades i PPC Journal i augusti 1979. Under två år skrevs det 70 sidor text som hade samband med modulen (finstilt, över 10.000 tecken per sida!).

När till sist packningen av de 15 tonnen (!) moduler och manualer gjordes på luciadagen 1981 krävde bara den saken ett smärre folkuppbåd plus diverse maskinella hjälpmedel.

Priset inkl manual på 500 sidor ungefär A4 är \$75, dvs knappt 600 kr (portot hit går dessutom på nära \$20!).

Och vad finns då i modulen? En sak som man lägger märke till i programmen är den avancerade användningen av "syntetisk programmering" med vars hjälp avsevärda genvägar kunnat tas i handhavandet av HP-41 (ingrepp i och användning av alfaregistren för andra ändamål; kontroll av uppdelning motsvarande vårt Op 16; kontroll av s k tillordning av tangenter för bestämda ändamål etc). Dessutom kan modulen användas som hjälp för egen syntetisk programmering. Däremot har HP-41 inte någon mot-svarighet till vår "Fast Mode".

En del program liknar dem som vi redan har i ML-modulen, t ex datum till julianskt dagnummer (och tvärtom!), ränte- och amorteringsberäkningar, kombinationer och permutationer, slumpalgsgenerator och (vilket HP-41 alltså inte har som standard!) nollställning av alla flaggor.

De andra programmen är mestadels sådana som man också hittar i vår lista i PB 81-1. Dit hör hantering av registerblock (t ex flyttning av innehållet i R5-R12 till R18-R25; man har avstått från att försöka flytta program-avsnitt!), viss matrishantering (summering av rader, multiplikation av rad med konstant, framtagnings av element), baskonvertering, funktioners första derivata, flaggtest och sortering. Det finns en "superplot" för upp till 9 funktioner samtidigt och en do. med hög upplösning (naturligtvis långsammare) samt histogram i hög upplösning med och utan axel. Så finns det ett packningsprogram för register som är samma som Björn Gustavssons i PB 81-4 s.9. Vidare finns en formateringsrutin för snygg tabellutskrift. Och som utfyllnad på några steg som blev över: rutinen "Paper Out" som består av fem "Adv"!

Man ser alltså att önskemålen om rutiner i en modul är ganska lika antingen man äger en HP-41 eller en TI-59. Det är naturligtvis tråkigt att konstatera nu när TI-59 inte längre tillverkas, att vi inte klarade vad HP-folket klarade, trots att vi hade två års försprång och trots att åtskilligt arbete redan var gjort i form av lämpliga program framställda av in-

UTMANINGEN!

Redaktörens adress:
Robert Prins
Alfred Nobellaan 112
NL-3731 DX De Bilt
Holland

Det här avsnittet av Utmaningen är inskrivet med hjälp av dator. Det har inneburit en del problem med utskriften av formler. Upphöjt till skrivs med två astersiker (**), 2^3 blir alltså $2**3$. w_q skrivs genomgående wq. Större än eller lika med skrivs $\geq$. På grund av ordbehandlingsprogrammet har radavståndet blivit större än normalt.

I skrivande stund, första september, har fem lösningar inkommit: två till problem 30, en för problem 39 och åter igen två för problem 40. Förutom det har jag fått ett stort brev från Bo med lösningar till en del äldre problem, som Björn skickade till honom när han slutade med Utmaningen.

Vi börjar med de tre lösningarna till problem 30 (ja, tre, eftersom jag också skrev ett program). Den här gången ger jag alla tre lösningarna, på grund av de helt skilda metoder som de tre programmerarna löste den tredubbla summeringen, användandet av faktorn (-1) och pekaren till de olika wq.

Den första lösningen kommer från Martin Harnevie, Målindal, förutom att jag har lagt till automatisk uppdelning. Martins lösning består av endast 101 steg. Han har placerat wq i R90-R99 vilket lämnar 59 steg åt funktionen.

Instruktionerna för programmet är:

1. Läs in block 1 i normaluppdelning.
2. Tryck E. Räkaren går in i programmeringsläge vid steg 101 och du kan då mata in din funktion. Du får använda INS, DEL och "=". När funktionen anropas finns x i displayen. Om du behöver värdet mer än en

gång kan det lagras i något av registren R8-R89. Avsluta funktionen med INV SBR och tryck LRN.

3. Mata in den nedre gränsen a och tryck A.
 4. Mata in den övre gränsen b och tryck B.
 5. Mata in antal intervall och tryck C.
- Efter en stund visas integralens värde.

Den andra lösningen kommer från Conny Bonet. Hans program är längre (det var 121 steg, men jag lade till tre steg för att göra det mer användarvänligt och också lite snabbare). Förutom det lyckades han inte få in program och konstanter i ett block. Hans program är - på grund av att x räknas ut varje gång - ungefär 60% långsammare än Martins. Men det viktigaste var att han skrev ett program!

Instruktionerna för programmet är:

1. Läs in block 1 och 4 i normaluppdelning.
 2. Tryck E. Räkaren går in i programmeringsläge vid steg 124 så att du kan mata in funktionen. (Det är ändringen som jag gjorde eftersom Connys program anropade E varje gång, vilket naturligtvis tar mer tid än SBR 124.) Du får använda "=" och alla register från R19 och uppåt. Som väntat finns x i displayen när funktionen anropas.
 - 3a. Mata in den nedre gränsen a och tryck A.
 - 3b. Mata in den övre gränsen b och tryck B.
- Stegen 3a och 3b måste utföras i den ordningen!
4. Mata in antal intervall och tryck C.
- Efter ett tag visas integralens värde i displayen.

Det tredje programmet kommer från mig. Jag skrev det för tre år sedan och anledningen till att jag tog upp problemet var att jag trodde att ingen skulle slå mitt programs korthet. Martin drog upp mitt huvud ur sanden genom att skriva ett både kortare och något snabbare program. Versionen som

presenteras här är 159 steg. Det kanske verkar långt, men programmet innehåller också en formel för beräkning av integraler från en nedre gräns till oändligheten. Jag hittade den formeln i HP-41C-boken "High level mathematics". Här är den:

$$\int_a^{\infty} f(x) dx = 2 \sum_{j=1}^2 \sum_{q=1}^5 \frac{w_{2q}}{1+(-1)^j \cdot w_{2q-1}} \times f \left[\frac{2}{1+(-1)^j w_{2q-1}} + a-1 \right]$$

Instruktionerna för programmet är:

1. Mata in din funktion under label A' med början vid steg 000. Du får använda "=" och alla register mellan R10 och R59. Som vanligt finns x i displayen när funktionen anropas.

2. Mata/läs in programmet och konstanter som alla ligger i block 2, steg 240-399 och R60-R69. Genom att lagra program och data på detta sätt får man ett stort utrymme, 240 steg, för lagring av funktionen och labelsökningen går snabbare, eftersom label A' (eller någon annan label) finns vid steg 001.

3a. Mata in den nedre gränsen a och tryck A.

3b. Mata in den övre gränsen b och tryck B.

Det är nödvändigt att utföra stegen 3a och 3b i ordning.

4. Mata in antal intervall och tryck C.

5. Beräkna integralen genom att trycka D.

Om du vill beräkna integraler från a till oändligheten gäller samma instruktioner med två undantag:

1. Mata inte in b (det ger fel).

2. Mata inte in antal intervall (det är meningslöst).

Nu till de tidigare nämnda skillnaderna i beräkningen av $(-1)^j$ upphöjt till j och pekaren till wq:

Den första lösningen kommer från Martin: han använder två anrop till E', steg 070 - 074, den ena med ett och den andra med minus ett. För att få den korrekta wq använder han två pekare, R6 & R7 vilka sätts till 99 och 94 och minskas för varje q. Det innebär att w29 lagras i R90-R94 och w29-1 i R95-99. Conny Bonet använde verkligen tredubbel summering. Titta på Dsz på stegen 081-084, 085-088 och 089-092. Han beräknade $(-1)^j$ upphöjt till j med den matematiskt inkorrekta formeln $(-1)^{j*2} = (3-2*j)$ (steg 042-049), eftersom hans j=1 ger +1 och hans j=2 ger -1. Det samma gäller beräkningen av registren som innehåller w29 och w29-1 (R1-R10).

Slutligen använde jag en flagga för att hålla reda på faktorn ett eller minus ett gånger j. Det har fördelen att inte behöva något register, men nackdelen att flaggan måste testas och varannan gång måste flaggan sättas respektive nollställas: stegen 280-282 och 321-329. Dessutom använder jag bara ett register för att peka på wq.

Efter den djupdykning i programmen kan man förstås fråga sig vilket som är bäst? Fråga inte mig, eftersom Martins är kortast och snabbast, men endast har plats för "korta" funktioner (trots att jag anser det vara mycket svårt att göra en funktion med mer än 59 steg). Connys program är väldigt lätt att följa och tillåter mycket långa funktioner, men är ganska långsamt. Slutligen mitt program: det är både långsammare och längre än Martins, men tillåter längre funktioner. (Jag betraktar inte möjligheten att betrakta integraler från a till oändligheten som en fördel eftersom jag inte gav formeln för det.)

En sista anmärkning: Conny skrev i sitt brev att han inte använde E(x)-termen. Det är naturligtvis inget problem eftersom den

bara ger det maximala felet som för det mesta är mycket litet. Den exakta fel-formeln för ett intervall är:

$$E(x) = \frac{(b-a)^{21} \cdot (10!)^4}{21 \cdot (20!)^3} \cdot f^{20}(\xi)$$

Där ξ ligger i intervallet (a,b)

Ett test för kontroll att program är riktigt:

$$\int_0^{\infty} x + \sin^2 x \, dx = 6.505598527$$

med 4 intervall (och vinkelmåttet Rad förstås).

Nästa problem var numerisk till alfakonverterare. Jag trodde att mitt program var kort (41 steg) tills jag såg Jan Björklunds lösning med bara 29 steg! Han använder formeln:

Skrivkod för den i:te bokstaven $Sk(i) = \text{Int}(2.5 \times \text{Int}(1 + (i + 2.8)/7.9) + 12.5)$

När jag såg formeln provade jag följande ändring:

$Sk(i) = \text{int}(2.5 * \text{int}(1 + (i + 42.3)/7.9))$

Det fungerar om man tänker på

- a. $12.5 = 5 \times 2.5$ och
- b. $\text{int}((a+b)/c) + d = \text{Int}((a+b+cd)/c)$ för alla heltalsvärden på d.

När jag hade testat den formeln försökte jag ta bort decimaldelen från 42.3. Den ändrade formeln fungerade för alla i förutom 21, där $(21+42)/7.9$ gav 7.9... i stället för 8. Uppenbarligen måste man försöka sänka 7.9 lite grann. Det första försöket, 7.85, visade sig gå bra, så den slutgiltiga formen som jag använde var:

$Sk(i) = \text{int}(2.5 * \text{int}(1 + (i + 42)/7.85))$

Programmet som gör den beräkningen är endast 26 steg långt, eller, om du utesluter de första två stegen "(Ins", endast 24!

Som sammanfattning vill jag säga att det är lite överdrivet att kalla Jan ett geni för att han fann formeln, men han gjorde verkligen ett fantastiskt jobb!

Det sista problemet som jag gav var Bachet-representationen av ett tal. Två lösningar kom: den första från Patrik Johansson, Ursviken, och den andra från Jan Björklund. Endast Patriks lösning tas med här, eftersom jag inte lyckades få riktiga svar från Jans program för värden $\geq 5E9$. Patriks program består av endast 58 steg, är snabbt och fungerar för alla heltal $\leq 10E13-1$. Naturligtvis hittade Patrik lösningens nyckel, nämligen att Bachet-representationen beror på bas 3-representationen av talet, och på det faktum att $2 \times 3^{**n} = 3^{**n+1} - 3^{**n}$. Det är litet för besvärligt att förklara, så låt oss ta talet 19 som ett exempel. Representationen i bas 3 fås genom att dela med 3 och ta resterna i omvänd ordning:

$19/3=6$ rest 1

$6/3=2$ rest 0

$2/3=0$ rest 2

Det betyder att 19 (bas 10) = 201 (bas 3). Det i kombination med det faktum att $23^{**n} = 3^{**n+1} - 3^{**n}$ ger lösningen till problemet:

1. Hitta talets motsvarighet i bas 3.
2. Ändra varje tvåa till -1 och addera ett till nästa högre potens (skulle en trea dyka upp skall den ersättas med en nolla och åter igen skall ett adderas till nästa högre potens av tre). 19 blir: 1, -1, 0, 1 (=27-9+1).

Detta kan naturligtvis utföras på många olika sätt, men Patriks lösning är verkligen väldigt koncis, och ger resultatet på ett snyggt sätt (men Jans resultat var faktiskt ännu snyggare).

Den sista lösningen gäller problem 27, påskens datum. Problemet ställdes först i

PGM 9, den danska Programklubbens tidskrift. Björn Gustavsson nämnde redan i PB 82-3 s 4 att en lösning hade kommit in.

Den lösningen presenteras här tillsammans med en lösning som fanns i PGM 11. Lösningen kommer från Krister Norrman och instruktionerna är:

1. Mata in programmet (48 steg).
2. Mata in registerinnehållet.
- 2a. 29 STO 4 Nödvändigt endast
 31 STO 5 en gång
- 2b. 25 STO 0 Dessa konstanter måste
 11 STO 1 lagras varje gång!
 19 X:T

Året-1900 STO 2 STO 3

3. Starta beräkningen med RST R/S.

Om ett positivt tal visas är det datumet i april. Om ett negativt tal visas måste du addera 31 för att få datumet i mars.

4. För ny beräkning gå till steg 2b.

Lösningen i PGM 11, av Henrik Klein och P. Korsgaard Johansen, är lite mer användarvänlig: det är inte nödvändigt att lagra en massa konstanter före varje körning och datumet visas i formatet dd.0m, där dd är datumet och m är månaden.

Instruktionerna för deras program är:

1. Mata in programmet (50 steg).
2. Mata in registerinnehåll (en gång).

0.8 STO 0
19 STO 3
25 STO 4
29 STO 5
31.03 STO 6

3. Tryck RST. Det behöver bara göras en gång.
4. Mata in året minskat med 1900 och tryck R/S. Påskens datum visas i ovan nämnda format.
5. För ny beräkning, gå till steg 4.

Det var alla lösningar den här gången. I nästa Utmaningen kommer en ny lösning till problem 30, Rombergs metod i Fast Mode, skrivet av Hans Lindgren. Förutom det har jag också fått en lösning till problem 7, permutationer och kombinationer, gjort av Håkan Svensson, och en allmän lösning till Damerna på schackbrädet skrivet av Björn Gustavsson. Hans program klarar av ett schackbräde av storleken 98×98 ! (Jag vet inte hur många lösningar ett sådant schackbräde har, men jag gissar att skulle ta många (hundratal eller tusental?) rullar av skrivarpapper att skriva ut alla, för att inte tala om tiden det skulle ta att beräkna alla (troligen flera år!))

Förutom det kommer jag att med ett ändrat program från PPC-Notes för beräkning av π med 1287 siffror.

Nu till några nya problem.

41. Ett ganska svårt problem går ut på att beräkna alla uppdelningar av ett heltal, dvs alla sätt som du skriva ett positivt heltal som en summa av positiva heltal. Till exempel:

2: 2 och 1+1

3: 3 och 2+1

4: 4 och 3+1 och 2+2 och 2+1+1 och 1+1+1+1

Antalet uppdelningar av n kallas $P(n)$. $P(0)$ och $P(1)$ har definierats som 1. Vad sägs om att skriva ett program som kan beräkna $P(n)$ för n upp till, låt oss säga, 100 ($P(100)=190569292$).

42. Skriv en subrutin som alstrar följande sekvens av tal: 37, 47, 24, 47, 24, 47, 24, 47, 24, 47, 37 ... Om du undrar vad för slags tal det är kan du försöka använda dem som skrivkoder och skriva ut med i Graphics Mode med hexkoden 25 vid steg 016. Du får en underbar skala, som kan användas i problem 43.

43. Skriv ett program som tar emot 19 värden och skriver ett stolpdiagram i Graphics Mode (och Fast Mode om du har lust). Helst bör programmet kunna avpassa värdena till en speciell maximal skallängd. Det skulle också vara trevligt att skriva ut både positiva och negativa värden och en horisontell linje vid en användardefinierad nivå om så önskas. Programmet skall skriva ett identifieringsmärke under varje kolumn i stolpdiagrammet. Jämför med mitt cirkeldiagram. Den första kolumnen på remsan skall innehålla skalan från problem 42.

44. Jag vet inte om du någonsin har haft någon nytta av (INV) DMS-funktionen i ett program i Fast Mode. Den går naturligtvis inte att använda direkt så du måste skriva din egen (INV) DMS-funktion, som helst bör vara kort. Gör ett försök!

45a. Det här är ett program som skall lösas i BASIC: Skriv ett program som läser in 59:ans tangentkoder och ger en listning liknande den på PC-100.

Alfakonverterare - Jan Björklund

```
000 25 LBL 008 93 . 016 59 INT 024 93 .
001 11 R 009 93 8 017 65 X 025 08 5
002 85 + 010 54 0 018 02 2 026 95 =
003 53 ( 011 55 + 019 93 . 027 59 INT
004 53 ( 012 07 7 020 05 5 028 92 RTH
005 2+ CE 013 93 . 021 85 +
006 85 + 014 09 9 022 01 1
007 92 2 015 34 0 023 02 2
```

Påskens datum - Krister Norrman

```
00 61 0 SBR 0      24 45 +
01 39 1 PRD 1      25 04 4
02 55 X            26 85 =
03 07 7            27 49 INT
04 75 +            28 34 2 SUM 2
05 01 :            29 32 5 RCL 5
06 85 =            30 34 2 SUM 2
07 45 =            31 33 2 RCL 2
08 33 7 RCL 7      32 61 0 SBR 0
09 85 =            33 34 0 INV SUM 0
10 49 INT          34 33 0 RCL 0
11 34 1 INV SUM 1  35 81 R/S
12 04 4            36 86 0 LBL 0
13 34 1 SUM 1      37 65 -
14 33 4 RCL 4      38 43 (
15 22 XIT          39 14 CE
16 33 1 RCL 1      40 45 +
17 61 0 SBR 0      41 33 7 RCL 7
18 84 +-           42 44 )
19 34 2 SUM 2      43 49 INT
20 34 0 SUM 0      44 55 X
21 07 7            45 33 7 RCL 7
22 22 XIT          46 44 )
23 33 3 RCL 3      47 61 INV SBR
```

45b. Lägg till ett program som gör om alla hopp till labels till absoluthopp och sedan listar det ändrade programmet.

45c. (Troligen väldigt svårt.) Skriv ett program i BASIC eller i assembler om du är ett riktigt geni, som simulerar en utökad TI-59, dvs ett program som uppför sig exakt som en TI-59 med 1000 register och 10000 steg. Det betyder att utökade koder kan vara t ex RCE (recall extended), där de två följande stegen ger minnesadressen eller GTE (GOTO extended). Du kanske undrar varför jag vill se ett sådant program: Tja, jag tycker det är mycket lättare att göra något i "59-language" än BASIC.

Det var allt för den här gången. Jag hoppas att fler personer kommer att skicka lösningar, men jag skulle också vilja ha fler problem, eftersom jag knappast har några kvar.

På återhörande

Robert Prins

P.S. Skicka inga magnetkort.

Alfakonverterare - Jan Björklund och Robert Prins

```
000 25 LBL 007 46 INS 014 93 . 021 93 .
001 11 R 008 95 + 015 08 8 022 05 5
002 33 ( 009 04 4 016 05 5 023 54 )
003 46 INS 010 02 2 017 54 ) 024 59 INT
004 85 + 011 54 0 018 59 INT 025 92 RTH
005 53 ( 012 55 + 019 65 X
006 53 7 013 07 7 020 02 2
```

Påskens datum - Henrik Klein och P. Korsgaard Johansen

```
00 32 1 STD 1      25 61 0 SBR 0
01 89 R            26 34 1 INV SUM 1
02 49 INT          27 33 2 RCL 2
03 32 2 STD 2      28 34 1 SUM 1
04 33 3 RCL 3      29 07 7
05 61 0 SBR 0      30 61 0 SBR 0
06 19 C.T          31 33 4 RCL 4
07 32 1 STD 1      32 34 7 SUM 7
08 55 X            33 25 1/X
09 07 7            34 76 GE
10 75 +            35 33 6 RCL 6
11 02 2            36 34 7 SUM 7
12 34 2 SUM 2      37 22 XIT
13 23 XIT          38 81 R/S
14 39 1 PRD 1      39 71 RST
15 65 =            40 86 0 LBL 0
16 34 1 SUM 1      41 39 1 INV PRD 1
17 03 3            42 55 X
18 85 =            43 28 1 EXC 1
19 45 =            44 49 INV INT
20 33 3 RCL 3      45 85 =
21 85 =            46 42 EE
22 49 INT          47 42 INV EE
23 34 1 INV SUM 1  48 34 7 INV SUM 7
24 33 5 RCL 5      49 61 INV SBR
```

Numerisk integrering enlig Gauss -
Robert Prins

240	76	LBL	230	22	INV	320	90	90	360	01	01
241	14	D	281	86	STF	321	01	1	361	75	-
242	07	7	282	00	00	322	94	+/-	362	01	1
243	69	DP	283	05	5	323	49	PRD	363	54	)
244	17	17	284	04	9	324	07	07	364	16	A
245	87	IFF	285	42	STD	325	22	INV	365	53	(
246	01	01	286	05	05	326	87	IFF	366	46	INS
247	02	02	287	05	5	327	00	00	367	65	x
248	73	73	288	42	STD	328	02	02	368	43	RCL
249	53	(	289	08	08	329	81	81	369	02	02
250	43	RCL	290	69	DP	330	97	DSZ	370	33	X2
251	01	01	291	25	25	331	04	04	371	61	STD
252	75	-	292	53	(	332	02	02	372	03	03
253	53	(	293	73	RC*	333	76	76	373	09	09
254	53	(	294	05	05	334	43	RCL	374	76	LBL
255	94	+/-	295	87	IFF	335	07	07	375	11	A
256	95	+	296	01	01	336	49	PRD	376	42	STD
257	43	RCL	297	03	03	337	09	09	377	01	01
258	03	02	298	44	44	338	06	6	378	02	2
259	94	(	299	65	x	339	69	DP	379	42	STD
260	55	+	300	43	RCL	340	17	17	380	07	07
261	43	RCL	301	07	07	341	43	RCL	381	85	STF
262	04	04	302	85	+	342	09	09	382	01	01
263	55	+	303	43	RCL	343	92	RTN	383	43	RCL
264	42	STD	304	06	06	344	53	(	384	01	01
265	03	03	305	54	)	345	97	IFF	385	92	RTN
266	02	2	306	16	A	346	06	00	386	76	LBL
267	54	)	307	53	(	347	03	03	387	12	B
268	42	STD	308	46	INS	348	50	50	388	22	INV
269	07	07	309	65	x	349	94	+/-	389	86	STF
270	54	)	310	69	DP	350	85	+	390	01	01
271	42	STD	311	25	25	351	01	1	391	42	STD
272	06	06	312	73	RC*	352	54	)	392	02	02
273	00	0	313	05	05	353	35	1/X	393	92	RTN
274	42	STD	314	54	)	354	42	STD	394	76	LBL
275	09	09	315	44	SUM	355	02	02	395	13	C
276	43	RCL	316	09	09	356	65	x	396	42	STD
277	03	03	317	97	DSZ	357	02	2	397	04	04
278	44	SUM	318	08	08	358	85	+	398	92	RTN
279	06	06	319	02	02	359	43	RCL			

Bachet tal - Patrik Johansson

000	76	LBL	015	52	(	030	00	00	045	07	7
001	13	A	016	24	CE	031	52	52	046	69	DP
002	99	ADP	017	75	-	032	32	XIT	047	04	04
003	99	PRT	018	53	(	033	01	1	048	43	RCL
004	43	STD	019	24	CE	034	67	EQ	049	01	01
005	00	00	020	55	+	035	00	00	050	69	DP
006	00	0	021	03	3	036	44	44	051	06	06
007	42	STD	022	54	)	037	65	DP	052	69	DP
008	01	01	023	59	INT	038	20	20	053	21	21
009	29	CP	024	42	STD	039	02	2	054	61	STD
010	43	RCL	025	00	00	040	00	0	055	00	00
011	00	00	026	65	x	041	61	STD	056	09	09
012	67	EQ	027	03	3	042	00	00	057	92	RTN
013	00	00	028	54	)	043	46	46			
014	57	57	029	67	EQ	044	04	4			

ROBERT PRINS
DATA REGISTER

A	01
B	02
(B-A)/N	03
N (DSZ)	04
VISARE TILL U ₀	05
USED	06
(B-A)/2N	07
Q (DSZ)	08
SUM	09
.1488743389816	60
.2955242247148	61
.4333953941292	62
.26926671931	63
.679409568299	64
.219086362516	65
.865063366689	66
.1494513491506	67
.9739065285172	68
.0666713443087	69

CONNY BONET
DATA REGISTER

N (DSZ)	00
.1488743389816	01
.2955242247148	02
.4333953941292	03
.26926671931	04
.679409568299	05
.219086362516	06
.865063366689	07
.1494513491506	08
.9739065285172	09
.0666713443087	10
A	11
(B-A)/N	12
J (DSZ)	13
(A-B)/N	14
Q (DSZ)	15
VISARE TILL U ₂₀	16
VISARE TILL U ₂₀₋₁	17
SUM	18

Numerisk integrering enlig Gauss -
Martin Harnevie

000	76	LBL	026	95	=	052	03	03	078	37	37
001	10	E'	027	44	SUM	053	95	=	079	97	DSZ
002	65	x	028	02	02	054	42	STD	080	05	05
003	73	RC*	029	92	RTN	055	04	04	081	00	00
004	07	07	030	76	LBL	056	00	0	082	70	70
005	75	-	031	11	A	057	42	STD	083	97	DSZ
006	01	1	032	42	STD	058	02	02	084	03	03
007	85	+	033	01	01	059	09	9	085	00	00
008	02	2	034	92	RTN	060	09	9	086	59	59
009	65	x	035	76	LBL	061	42	STD	087	43	RCL
010	43	RCL	036	12	B	062	05	06	088	04	04
011	03	03	037	42	STD	063	09	9	089	65	x
012	95	=	038	03	03	064	04	4	090	43	RCL
013	65	x	039	92	RTN	065	42	STD	091	02	02
014	43	RCL	040	76	LBL	066	07	07	092	95	=
015	04	04	041	13	C	067	05	5	093	92	RTN
016	85	+	042	48	EXC	068	42	STD	094	76	LBL
017	43	RCL	043	03	03	069	05	05	095	15	E
018	01	01	044	75	-	070	01	1	096	01	1
019	95	=	045	43	RCL	071	10	E'	097	00	0
020	71	SBK	046	01	01	072	01	1	098	69	DP
021	01	01	047	95	=	073	94	+/-	099	17	17
022	01	01	048	55	+	074	10	E'	100	31	LRN
023	65	x	049	02	2	075	69	DP			
024	73	RC*	050	55	+	076	36	36			
025	06	06	051	43	RCL	077	69	DP			

Numerisk integrering enlig Gauss -
Conny Bonet

000	76	LBL	031	95	=	062	75	-	093	43	RCL
001	13	C	032	42	STD	063	43	RCL	094	12	12
002	42	STD	033	17	17	064	00	00	095	55	+
003	00	00	034	73	RC*	065	85	+	096	02	2
004	32	HIR	035	17	17	066	01	1	097	65	x
005	08	08	036	65	x	067	54	)	098	00	0
006	22	INV	037	43	RCL	068	65	x	099	48	EXC
007	49	PRD	038	12	12	069	43	RCL	100	18	18
008	12	12	039	55	+	070	12	12	101	95	=
009	22	INV	040	02	2	071	95	=	102	97	DSZ
010	49	PRD	041	65	x	072	71	SBK	103	76	LBL
011	14	14	042	53	(	073	01	01	104	11	A
012	02	2	043	03	3	074	24	24	105	42	STD
013	42	STD	044	75	-	075	65	x	106	11	11
014	13	13	045	02	2	076	73	RC*	107	42	STD
015	05	5	046	65	x	077	16	16	108	14	14
016	42	STD	047	43	RCL	078	95	=	109	94	+/-
017	15	15	048	13	13	079	44	SUM	110	42	STD
018	02	2	049	54	)	080	18	18	111	12	12
019	65	x	050	85	+	081	97	DSZ	112	91	R/S
020	33	(	051	43	RCL	082	15	15	113	76	LBL
021	06	6	052	14	14	083	00	00	114	12	B
022	75	-	053	55	+	084	18	18	115	44	SUM
023	43	RCL	054	02	2	085	97	DSZ	116	12	12
024	15	15	055	85	+	086	13	13	117	22	INV
025	54	)	056	43	RCL	087	00	00	118	44	SUM
026	95	=	057	11	11	088	15	15	119	14	14
027	42	STD	058	85	+	089	97	DSZ	120	91	R/S
028	16	16	059	53	(	090	00	00	121	76	LBL
029	75	-	060	82	HIR	091	00	00	122	15	E
030	01	1	061	18	18	092	12	12	123	31	LRN

MARTIN HARNEVIE
DATA REGISTER

A	01
SUM	02
B/N/K	03
(B-A)/2N	04
Q	05
STYRREGISTER	06
STYRREGISTER	07
.1488743389816	90
.4333953941292	91
.679409568299	92
.865063366689	93
.9739065285172	94
.2955242247148	95
.26926671931	96
.219086362516	97
.1494513491506	98
.0666713443087	99

KRISTER NORRMAN
DATA REGISTER

25	0
11	1
ARET - 1900	2
ARET - 1900	3
29	4
31	5
--	6
19	7

HENRIK KLEIN -
P. KORSGAARD JOHANSEN
DATA REGISTER

0.8	0
--	1
--	2

HEXKODER på TI 57

DEJAN RISTANOVIĆ

Den lilla TI 57:an har fortfarande många hemligheter. Eftersom varje programsteg tar upp en byte, finns det 256 möjliga instruktioner (en byte kan anta 256 olika värden). Då 57:an inte använder sig av alla dessa instruktioner "fattas" det några instruktioner. Varje dator måste på något sätt reagera på varje instruktion, låt oss därför undersöka vad som händer när datorn stöter på någon av de instruktioner som "fattas". Kort sagt: låt oss studera följderna av "syntetisk programmering".

För att kunna skapa någon av de nya koderna på steg KK+4 matas först följande program in. (KK är ett referenssteg och kan beteckna valfritt stegnummer).

stegnummer	kod	tangent
KK+0	02	2
KK+1	38	Exc SST
KK+2	86 0	Lbl 0
KK+3	00	0
KK+4	00	0

Steg KK+1 innehåller den mest intressanta sekvensen. Tryck Exc SST och 57:an luras till att behålla kod 38 (Exc), men utan någon adressdel. Med ovanstående korta program i räknaren skapas den syntetiska koden på steg KK+4 genom att manuellt trycka GTD 2nd KK SST SST SST. Programpekaren kommer nu att vara på steg KK+4.

Här finns det tre sätt att fortsätta på:

- I Tryck LRN och därefter en siffra från 2 t.o.m. 7. Detta förfarande skapar en hexadecimal kod 0n, där n kan vara någon av de hexadecimala "siffrorna" (A, B, C, D, E eller F). Om exempelvis 2 tryckes kommer koden 0A att framträda. Denna kod fungerar som en vanlig sifferinmatning som matar in "siffran" A. Samma förhållande gäller även för de övriga "siffrorna" B - F. Detta kan vara mycket användbart då text kan skrivas i displayen, under förutsättning att den bara innehåller bokstäverna A - F, samt andra bokstäver som kan

liknas vid siffror (t.ex. 0 kan motsvara noll). Man kan exempelvis skriva FIASCQ som någon kommentar i ett spelprogram.

Den här möjligheten att få fram bokstäver i displayen är avsiktligt kvarlämnad av TI 57:s konstruktör. Beviset för detta är att A, C, E och F är stora bokstäver, medan B och D skrivs b respektive d, dvs. med små bokstäver. Detta för att de bokstäverna ska kunna skiljas från siffrorna 8 och 0. Tyvärr kan man inte räkna med dessa hexadecimala tal. T.ex. beter sig 1B som 21 (och inte 37), samma sak gäller för övrigt även TI 58/59.

- II Tryck EE LRN och mata in en siffra n mellan 0 och 7 (inklusive 0 och 7). Med denna procedur erhålls koden m2, där $m = n + 1$. Detta ger koderna för instruktionerna på tangentbordets andra kolumn (INV, STO, EE osv.). Talet m representerar raderna på tangentbordet. Exempel: Mata in siffran 3 ($= n$), detta ger $m = 4$, vilket resulterar i koden 42 som är lika med STO.
- III Tryck . LRN n, även här är n en siffra mellan 0 och 7. Detta ger en möjlighet att skapa koder m1, där $m = n + 1$. På detta sätt kan koderna för alla instruktionerna i första raden framställas: 2nd, LRN et.c.

Låt oss nu titta på några användbara hexkoder:

1. Instruktionen som fås av GTD SST eller SBR SST kommer att söka upp första förekomsten av kod 00 i programmet; det kommer sedan att fortsätta på steget efter kod 00.
2. Sekvensen 51 0n gör samma sak frånsett att programkörningen flyttas till den första 0n koden i programmet.
3. Fix SST 0n ($n = A, B, C, D, E$ eller F) har några användbara möjligheter. Om $n = F$ kommer räknaren att sättas i

"Fix -1" läge, vilket kan vara nyttigt för att spara energi då ett program måste ligga inne en längre stund (bara ett minustecken syns i displayen efter CLR =).

4. Sekvensen 11 = återkallar det tal som fanns i displayen innan CLR utfördes. Detta är ett väldigt intressant beteende men inte vidare användbart.
5. Sekvensen = 12 dividerar talet i displayen med 10 upphöjt till (9 - n). Räkaren skall vara i EE-läge med Fix n.
6. Sekvensen 12 11 EE Int plockar ut de tre sista siffrorna av talet i displayen, under förutsättning att räkaren befinner sig i EE och Fix 9 läge.
7. Sekvensen Clr 12 har några intressanta effekter, mestadels för att ha roligt med displayen. Om räkaren är i Fix -1 läge, tryck +/- +/- . och bara en punkt finns kvar i displayen.

För att demonstrera den praktiska nyttan med syntetisk programmering på TI 57 följer här ett program som genererar slumpstal mellan 0 och 999. Ingen, absolut ingen, kan göra det kortare utan hjälp av de nya koderna.

```
Lb1 0 RCL 0 Lnx 12 11 EE Int STO 0 RTN
```

Fröet i Register 0 skall vara positivt.

Inmatningsinstruktioner för ovanstående program:

Vi startar på KK+0 = 8.

Tryck GTO 2nd 08 LRN och mata in initieringsrutinen:

```
2 2nd Exc SST 2nd Lb1 0 LRN.
```

Tryck nu :

```
GTO 2nd 08 SST SST SST . LRN 0 BST
```

och se 12 11 i displayen, vilket visar att vi har skapat hex-koden 11 på steg 12.

Tryck 2nd Ins för att göra plats för nästa hex-kod, kod 12.

Tryck nu :

```
LRN GTO 2nd 08 SST SST SST EE LRN 0 BST
```

och se 12 -12 i displayen. Vi har nu även

hex-kod 12 på plats i programmet. För att göra resten av programmet trycker vi BST följt av Ins, därmed får vi plats i programmet före hex-koderna. Nu återstår bara att mata in den "vanliga delen" av programmet.

Tryck nu :

```
RCL 0 Lnx SST SST EE Int STO 0 INV EE Pause  
GTO 0 LRN.
```

(med SST hoppar vi över hexkoderna)

För att köra programmet behövs ett frö i register 0. Om fröet 21 används blir resultatet att de tresiffriga talen 452, 368, 808, 456, 249 ... blinkar fram i displayen.

Som Du säkert märkt har programmet utökats med instruktionerna INV EE Pause GTO 0. Detta medför att tio-exponenten tas bort vid visning samt att programmet stannar upp en sekund för avläsning av displayen och sedan återupptar proceduren.

Jag hoppas att dessa hex-koder skall inspirera till en hel del efterforskningar inom området, så ta och plocka fram TI 57:orna ur de mörka byrålåden.

Dejan vill även tacka Sinisa Djurekovic i Zagreb, Jugoslavien, för att ha uppmärksammat dessa nya koder.

Redaktören för TI PPC-Notes, Palmer O Hanson, har provat de nya koderna och skrivit några kommentarer:

- A. Detta sätt att skapa nya koder fungerar på TI 57 men inte på TI 57 LCD.
- B. Koderna kan flyttas med Del och Ins utan att de normaliseras, detta är inte fallet på TI 58/59.
- C. När koder skapas på sätt II med n motsvarande antingen 0, 1, 2 eller 3, framstår koderna med ett minustecken framför sig, vilket vanligtvis indikerar inversen av instruktionen. För n = 4 till 7 fås 51 7, 51 4, 51 1 och 51 0. Adressen för dessa GTO-instruktioner motsvarar siffertangenterna på tangentbordets andra kolumn.
- D. Initieringsprogrammet fungerar lika bra med Prd SST eller Fix SST istället för Exc SST.
- E. Om tvåan på steg KK+0 i initieringsprog-

MINFÄLT

DEJAN RISTANOVIĆ

Från Dejan Ristanovic i Jugoslavien har vi fått spelprogrammet Minefield, eller Minfält som det torde heta på svenska. Programmet kräver en II 59:a för att kunna köras, där- emot är skrivare inte nödvändig men kan vara användbar.

Din uppgift är att ta dig över ett minfält, utan att utlösa någon mina. Minfältet består av 10 * 10 rutor, där ett slumpmässigt antal minor finns utlagda, dock högst en mina per ruta. Dessutom innehåller minfältet stora stenbumlingar, som hindrar dig från att beträda vissa rutor. Till din hjälp har du en mindetektor som talar om hur många minor som finns på de omkringliggande rutorna. Det finns en hake: mindetektorn drar en hel del ström, som du naturligtvis har begränsad tillgång till. När strömmen är slut tvingas du att famla fram i blindo. Näväl, du kan spara ström genom att stänga av mindetektorn när du tror att den inte behövs. Som tur är har du tillgång till en engångssköld, som gör det möjligt för dig att beträda en ruta som innehåller en mina, utan att du själv tar någon skada. Som sagt: du kan bara använda dig av skölden en gång per spel. Dessutom täcks mindetektorn av skölden, därför får du ingen information om antalet kringliggande minor när du använder skölden.

När (eller snarare om) du når fram till mål får du en poängsiffra som anger hur pass bra du har lyckats. Taktiken i det här spelet är uppenbarligen att söka sig fram längs vägar där det inte finns några omkringliggande minor. Förr eller senare stöter man på minor, då gäller det att undersöka bredvid liggande rutor och på så vis få en bättre uppfattning om minans exakta position. I princip blir det ett ständigt uppställande av "sannolikhetsskalkyler" ända till risken att det ska finnas en mina på den ruta du vill gå till har minimerats. Spelet kräver med andra ord ett flitigt tänkande.

Börja med att knappa in det 480 steg långa programmet. Observera att Nop:en på steg 439 fyller en funktion, mer därom senare. Därefter initieras programmet med E', gå nu och drick en kopp te eller något annat stärkande; programmet behöver nämligen 10 minuter

på sig för att göra i ordning spelplanen. När initieringen äntligen är till ända skrivs POS 00 N MINES ut, vilket innebär att du befinner dig på ruta 00 och att N antal minor kringgärdar dig. Lägg märke till att startrutan 00 är i övre högra hörnet och att målrutan 99 således befinner sig längst ner till vänster. Positionsangivelsen är alltid på formen YX, tiotals-siffran anger Y-position och entalssiffran X-positionen.

Förflyttning åstadkomms med A, B, C och D. A är i västlig riktning (vänster), B östlig riktning (höger), C motsvarar norrut (uppåt) och slutligen D söderut (nedåt). Efter varje drag dröjer det cirka 10 sekunder innan ny information ges på skrivaren. Om inget speciellt har inträffat ges den vanliga upplysning om YX position samt antalet kringliggande minor. Andra möjligheter är att ROCK, SHLD, BOOM eller ingenting alls skrivs ut. ROCK innebär att ruta dit du ville gå innehåller en sten, vilket gör det omöjligt att beträda ruta. Du tvingas stanna kvar på den tidigare ruta. SHLD är en förkortning av shield, du har aktiverat skölden och därmed kan mindetektorn inte ge något utslag. BOOM är självförklarande, du har gått på en mina och har förlorat. Om ingenting skrivs ut har du stängt av mindetektorn för att spara energi. SBR SBR är sekvensen för att aktivera skölden, men kom ihåg att detta bara är möjligt en gång per spel. E kopplar av mindetektorn.

Har du ingen skrivare går det ändå att använda programmet. Efter varje drag syns antalet minor i displayen. Om du ser en nia finns det en sten på ruta. En blinkande nolla indikerar att spelet är förlorat, du har stött på en mina. Ett tal större än nio är poängen du har fått efter att ha gått i mål.

Nop:en på steg 439, som nämndes tidigare, gör det möjligt att starta spelet på nytt efter att ha gått en mina. Det är bara för dig att trycka R/S. Med tanke på de 10 minuter som initiering tar kan det vara befogat med denna möjlighet till fusk. Om detta synes dig moraliskt anstötligt kan du byta ut Nop:en mot ett Cms.

13 ► HEXKODER på TI 57

rammet ersätts med en nolla fungerar EE LRN och . LRN n sekvenserna. Minustecknet i paragraf C finns inte med men andra effekter nämnda av Dejan dyker upp. Det gör ingen skillnad i ett program om hexkoden är positiv eller negativ (inversen). Med 0 på steg KK+0 fungerar inte första sättet att skapa hexkoder, dvs. det sättet som medger att skriva text i displayen.

Det är dock inte att rekommendera att använda 0 på steg KK+0 för att eliminera minustecknet och på så vis blanda koder gjorda på de olika sätten. Anledningen är

Det här programmet är inspirerat av John Ionidas' HP 41C program med samma namn, som är hämtat från HP Users' Library, Genève. Enligt Dejan är detta program roligare och lättare att använda. 59-programmet är egentligen en utvecklad version. Det ursprungliga programmet ändrades efter ett förslag av Lars Hedlund. Den första versionen krävde 50 sekunders väntetid mellan dragen, däremot var tiden för initiering kortare.

```

000 75 LBL      060 44 SUM      120 03 03      180 09 9
001 17 B'      061 03 03      121 01 1        181 19 D'
002 85 +      062 69 DP      122 00 0        182 69 DP
003 32 XIT     063 24 24      123 22 INV      183 33 33
004 08 8       064 43 RCL     124 44 SUM      184 02 2
005 77 GE      065 04 04      125 03 03      185 44 SUM
006 00 00      066 94 +/-     126 61 GTO      186 04 04
007 10 10      067 22 INV     127 00 00      187 03 3
008 03 3       068 28 LOG     128 92 92      188 42 STD
009 76 LBL     069 52 EE      129 76 LBL      189 05 05
010 01 1       070 65 X      130 11 A        190 03 3
011 95 =       071 73 RC*     131 69 DP      191 94 +/-
012 92 RTN     072 03 03      132 34 34      192 44 SUM
013 76 LBL     073 95 =      133 61 GTO      193 04 04
014 19 D'      074 82 HIR     134 95 =      194 03 3
015 82 HIR     075 03 03      135 76 LBL      195 42 STD
016 08 08      076 22 INV     136 12 B        196 00 00
017 71 SBR     077 59 INT     137 69 DP      197 22 INV
018 00 00      078 32 HIR     138 24 24      198 86 STF
019 48 48      079 53 53      139 61 GTO      199 00 00
020 42 STD     080 65 X      140 95 =      200 43 RCL
021 03 03      081 01 1      141 76 LBL      201 03 03
022 71 SBR     082 00 0      142 13 C        202 71 SBR
023 00 00      083 22 INV     143 69 DP      203 03 03
024 48 48      084 44 SUM     144 33 33      204 88 88
025 42 STD     085 03 03      145 61 GTO      205 43 RCL
026 04 04      086 95 =      146 95 =      206 04 04
027 85 +      087 82 HIR     147 76 LBL      207 71 SBR
028 43 RCL     088 02 02      148 14 D        208 03 03
029 03 03      089 59 INT     149 69 DP      209 88 88
030 95 =       090 82 HIR     150 23 23      210 87 IFF
031 29 DP      091 52 52      151 61 GTO      211 00 00
032 67 EQ      092 22 INV     152 95 =      212 02 02
033 00 00      093 52 EE      153 76 LBL      213 28 28
034 17 17      094 69 DP     154 15 E        214 08 8
035 08 8       095 34 34      155 96 STF      215 32 XIT
036 32 XIT     096 92 RTN      156 01 01      216 71 SBR
037 71 SBR     097 69 DP      157 92 RTN      217 00 00
038 00 00      098 24 24      158 76 LBL      218 58 58
039 58 58      099 95 +      159 10 E'       219 77 GE
040 77 GE      100 82 HIR     160 47 CMS      220 02 02
041 00 00      101 12 12      161 42 STD      221 28 28
042 17 17      102 95 =      162 09 09      222 85 +
043 82 HIR     103 55 +      163 01 1        223 01 1
044 18 18      104 01 1      164 00 0        224 95 =
045 61 GTO     105 00 0      165 42 STD      225 71 SBR
046 00 00      106 44 SUM     166 06 06      226 00 00
047 97 97      107 03 03      167 02 2        227 97 97
048 36 PGM     108 85 +      168 04 4        228 69 DP
049 15 15      109 32 HIR     169 42 STD      229 24 24
050 71 SBR     110 13 13      170 08 08      230 97 DSZ
051 88 DMS     111 95 =      171 09 9        231 00 00
052 65 X      112 65 X      172 42 STD      232 01 01
053 01 1      113 43 RCL     173 00 00      233 97 97
054 00 0       114 04 04      174 08 8        234 69 DP
055 95 =       115 22 INV     175 19 D'       235 33 23
056 58 INT     116 28 LOG     176 97 DSZ      236 97 DSZ
057 92 RTN     117 52 EE      177 00 00      237 05 05
058 01 1       118 95 =      178 01 01      238 00 01
059 00 0       119 72 ST*     179 74 74      239 90 90

```

klar: om minustecknet tas bort kan hex-koder skapas som ser likadana ut i LRN-läge men som betar sig olika vid programkörning. Exempel:

a) Med 2 på steg KK+0, genomför initieringen och tryck LRN 4. Därmed skapas koden 0C, som visas som 12 i LRN-läge och som skriver ut ett C i displayen vid körning.

b) Med 0 på steg KK+0, genomför initieringen och tryck EE LRN 0, på så vis skapas 12, som också visas som 12 i LRN-läge. Däremot skriver den inte ut ett C i displayen.

Jag önskar er trevlig hexadecimal programmering!

5 ► Egen modul? Nej tyvärr!

tresserade och skickliga programmerare. Men, som sagt, deras framgång har inte varit gratis - det ligger många sena kvällars jobb bakom framgången med PPC ROM. Det är dock en liten tröst, och bättre är att sluta med en applåd för vad HP-folket uppnått!

```

240 97 DSZ      300 69 DP      360 17 B'      420 69 DP
241 06 06      301 01 01      361 65 X      421 05 05
242 01 01      302 43 RCL     362 01 1        422 09 9
243 80 80      303 03 03      363 00 0        423 92 RTN
244 25 CLR     304 17 B'      364 00 0        424 01 1
245 42 STD     305 82 HIR     365 95 =      425 04 4
246 03 03      306 06 06      366 69 DP      426 03 3
247 42 STD     307 43 RCL     367 03 03      427 02 2
248 04 04      308 04 04      368 03 3        428 03 3
249 42 STD     309 17 B'      369 00 0        429 02 2
250 08 08      310 85 +      370 02 2        430 03 3
251 82 HIR     311 82 HIR     371 04 4        431 00 0
252 04 04      312 16 16      372 03 3        432 07 7
253 02 2       313 65 X      373 01 1        433 03 3
254 04 4       314 01 1      374 01 1        434 69 DP
255 42 STD     315 00 0      375 07 7        435 04 04
256 07 07      316 00 0      376 03 3        436 69 DP
257 76 LBL     317 95 =      377 06 6        437 05 05
258 95 =       318 69 DP      378 69 DP      438 25 CLR
259 69 DP      319 02 02      379 04 04      439 68 NDP
260 28 28      320 01 1      380 82 HIR      440 61 GTO
261 69 DP      321 32 XIT      381 12 12      441 61 GTO
262 00 00      322 82 HIR      382 29 DP      442 03 3
263 01 1       323 14 14      383 69 DP      443 06 6
264 08 8       324 67 EQ      384 05 05      444 01 1
265 32 XIT     325 04 04      385 61 GTO      445 05 5
266 43 RCL     326 57 57      386 04 04      446 03 3
267 03 03      327 71 SBR      387 76 76      447 05 5
268 85 +       328 00 00      388 65 X      448 04 4
269 43 RCL     329 58 58      389 53 53      449 00 0
270 04 04      330 82 HIR      390 40 IND      450 69 DP
271 95 =       331 02 02      391 75 -      451 04 04
272 67 EQ      332 32 XIT      392 09 9        452 43 RCL
273 04 04      333 09 9        393 95 =      453 08 08
274 42 42      334 67 EQ      394 32 XIT      454 69 DP
275 22 INV     335 04 04      395 00 0        455 06 06
276 86 STF     336 24 24      396 77 GE      456 91 R/S
277 00 00      337 08 8        397 04 04      457 03 3
278 43 RCL     338 67 EQ      398 01 01      458 06 6
279 03 03      339 04 04      399 86 STF      459 02 2
280 71 SBR     340 02 02      400 00 00      460 03 3
281 03 03      341 43 RCL      401 92 RTN      461 02 2
282 88 88      342 03 03      402 43 RCL      462 07 7
283 43 RCL     343 42 STD      403 01 01      463 01 1
284 04 04      344 01 01      404 42 STD      464 06 6
285 71 SBR     345 43 RCL      405 03 03      465 04 4
286 03 03      346 04 04      406 43 RCL      466 00 0
287 88 88      347 42 STD      407 02 02      467 69 DP
288 87 IFF     348 02 02      408 42 STD      468 04 04
289 00 00      349 25 CLR      409 04 04      469 69 DP
290 04 04      350 87 IFF      410 03 3        470 05 05
291 24 24      351 01 01      411 05 5        471 76 LBL
292 03 3       352 03 03      412 03 3        472 71 SBR
293 03 3       353 82 82      413 02 2        473 01 1
294 03 3       354 22 INV      414 01 1        474 82 HIR
295 02 2       355 97 DSZ      415 05 5        475 34 34
296 03 3       356 07 07      416 02 2        476 22 INV
297 06 6       357 03 03      417 06 6        477 86 STF
298 04 4       358 82 82      418 69 DP      478 01 01
299 00 0       359 32 XIT      419 04 04      479 92 RTN

```



Texas Instruments hemdator.

Texas Instruments hemdator är en riktig dator. Som hela familjen kan arbeta med, räkna på, lära av eller spela med.

Det finns mer än 1000 färdiga program från hela världen – från rymdspel och schack till budgetering av familjens ekonomi. Lägg bara in programmen och du är klar att börja.

När du själv vill programmera ger den utförliga handboken dig instruktioner, steg för steg. Spännande och inte alls svårt.

Texas Instruments hemdator har hela 16 Kbyte användarminne (RAM) redan från början men stannar inte vid detta – utvecklingsmöjligheterna är mycket stora. Du kan tex ansluta en talenhet som ger din dator röst.

Kom in och prova Texas Instruments hemdator – den är mer än värd sitt pris.



TEXAS INSTRUMENTS