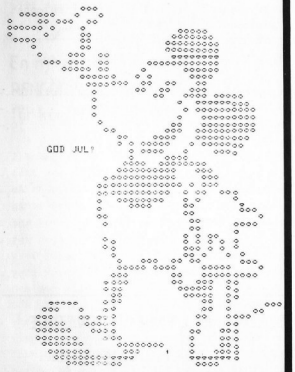
Claes PR ex

PROGRAM BITEN 81.4



Innehåll

Dyre medlem!

Så fyller alltså Programbiten tre år - eller kanske rättare tre årgångar. Längre tid är det inte som vi har hållit på med att utbyta programmeringserfarenheter. För det är ju en förening vi är med i och ett utbyte vi sysslar med, inte att erbjuda en färdig tidning. De fyra numren har ändå kommit någorlunda i rätt tid i år, och vi hoppas vi kan fortsätta med det.

I år har vi också börjat träffas personligen i litet större utsträckning. Det hör ju också till en förening att man lär känna varandra. Ett medel att göra detta vore ju också att sätta ut adress och telefon till den som inför ett bidrag. Om Du inte har något emot att bli uppringd av den som vill diskutera något med Dig så lämna de uppgifterna ihop med bidraget så tar vi in dem.

Det här året har också medfört ökade utlandskontakter. I år har det främst gällt programbiblioteket, men vi har ju också haft fortsatt tillgång till vad de stora kanonerna i USA presterat genom TI PPC Notes.

Nästa nummer tillhör alltså vår fjärde årgång. Vi skall försöka fortsätta på samma sätt som i år. Skulle det dessutom hända något nytt på räknarfronten (vilket väl många väntar på) så kan Du vara förvissad om att vi kommer att täcka in det.

Till sist ett tack till alla som bidragit till Programbiten under 1981, både som författare och i redaktionsarbetet!

Trevlig helg!

Lars Hedlund, ordf

Årsmöte

lördagen den 30 januari 1982 kl. 13.00
i Folke Johanson Ingenjörsbyrå AB:s lokaler i tunnelbanehuset, Farsta centrum (ingång i det runda trapphuset på SO-sidan, dörren är låst - ring så öppnar vi!).

På dagordningen står gängse föreningsangelägenheter, dvs ansvarsfrihet och ny styrelse samt den kommande verksamheten under 1982.

Ärende som medlem önskar få behandlat på årsmötet skall vara styrelsen tillhanda senast den 11 januari 1982.

Efter årsmötet anordnar vi

Medlemsträff

i samma lokaler fram till kl. 18. Tillfälle att lära känna Dina klubbkamrater och att diskutera programmering!

NYA PRISER FÖR GAMLA ÅRGÅNGAR - FÖRENINGENS TILLBEHÖRSFÖRSÄLJNING

Vi har satt ner priserna på gamla årgångar! Principen är 20 kr nedsättning per år räknat från medlemsavgiften 120 kr, ned till min 60 kr, och vi tillämpar fr o m nu 1982 års priser. Beställ genom att sätta in beloppet på postgiro 430 01 59 - 3!

För övrigt är priserna oförändrade, dvs

Sourcebook for Programmable Calculators	175,-
40 st tomma magnetkort och plånbok	108,-
Tom magnetkortsplånbok	10,-
Föreningens programmeringsblanketter, olika typer (se PB 79-3&4), block om 50 blanketter	11,-
Programbiten årgång 1978/79	60,-
Programbiten årgång 1980	80,-
Programbiten årgång 1981	100,-
Patenthandlingar TI-59	25,-
Astronomical Formulae (kan beställas även nu, viss väntetid)	55,-

MEDLEMMAR UTANFÖR SVERIGE !

Välkomna med ny medlemsavgift - men det måste vara svenskt postgiro eller bankcheck! Betalning genom bank kan vi inte godta!

"... Jag skulle vilja ge ett förslag angående PB. Jag tror att PB skulle bli mer vänlig (inte för att den inte är det) om det fanns bilder på dem som redigerar de avdelningar som ständigt förekommer (Utmaningen, Programbiblioteket etc). Åtminstone en av Björn Gustavsson. Han är en sådan fantastisk programmerare att han nästan kunde sättas på utställning!"

Markús Sveinn Markússon

Björn svarar att han tycker att fler, t ex Markús, gärna kunde visas i bild. Av trycktekniska skäl är det emellertid svårt att ha fotografier på mer än en sida. Att däremot visa just dem som Markús föreslår på s. 2 tycker jag är en mycket god idé. Vi skall med lämpor och övertalning försöka få vederbörande att lämna ifrån sig ett foto till nästa nummer.

En ny sorts RENGÖRINGSKORT för kortläsaren i TI 59

I belgiska TISofts medlemsblad som vi just fått kom ett gratisprov och en beskrivning på en ny produkt som Texas Instruments godkänt: Ett belgisktillverkat rengöringskort för kortläsaren av typ "våtservett", avsett för engångsbruk. Det är dyrt (skulle kosta över 10, kanske 15 kr/st i Sverige) men anses vara så effektivt att TI numera packar med dem med sina TI-59. Det har ingen slipverkan och skadar ej magnethuvudet. Det innehåller ett mycket lättflyktigt (brandfarligt) lösningsmedel och förpackningen måste därför öppnas omedelbart före användandet. Kortet torkar så snabbt att den kan användas en andra gång för en mer mekanisk rengöring.

TI Service i Sverige känner till kortet och anser att det är bra. En liknande effekt kan man faktiskt uppnå med mer primitiva medel. Om man besprutar det vanliga rengöringskortet av kartong med ett medel som heter OXYD EX-61 (svenskt avfettnings och rengöringsmedel) får man en riktigt hygglig verkan. Observera att andra medel inte får användas (t ex "5-56" innehåller fett vilket är absolut förbjudet!)

TI Sverige överväger att ta in det belgiska "våtservettkortet" och vi kunde eventuellt också ta in det genom förmedling av TISoft. Även 15 kronor är ju trots allt en måttlig kostnad jämfört med att sända in räknaren för rengöring.

SKRIVARPAPPER

I PB 80-1 fanns en sida om skrivarpapper. Där rekommenderas bl a TI:s eget papper som har beteckningen TP 30250.

Nu har det emellertid upptäckts att det finns två olika papperstyper med denna beteckning! Det ena är tillverkat i Frankrike och ger blå skrift medan det andra är tillverkat i Belgien och ger svart skrift. Det "blå" (franska) papperet är dessvärre omöjligt att limma med vissa typer av kontorsklister (allt blir blått; däremot fungerar det lim bra som heter Henkel Pritt Multi-Limcreme!) och dessutom medför tejp över ett tecken att detta försvinner på ett par dagar. Det "svarta" (belgiska) papperet har inte dessa egenskaper.

På TI Sverige tror man att det "blå" franska papperet är en kvalitet som råkat bli kvar länge hos någon återförsäljare. Eftersom sådant ju dessvärre händer när man minst anar det så skadar det alltså inte att känna till skillnaden om man är "blåkänslig".

PROGRAMBITEN och FÖRENINGEN PROGRAMBITEN
c/o Hedlund
Årstavägen 27 6 tr
121 68 JOHANNESHOV

För KOMMERSIELLT 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

UTMANINGEN!

Redaktörens adress:

Björn Gustavsson
Nordlandervägen 12 A

777 00 SMEDJEBACKEN

Hej.

Markús Sveinn Markússon har skickat in ett program för DSE-funktion (problem 11); se pgm 1. Det använder T-registret och två vilande operationer. Före användning skall R_0 laddas med ett tal på formen ttttt.sssssfff, där alltså ttttt är talet som räknas ned, sssss är stopptal och fff förändringen. ttttt kommer att minskas med fff ända tills ttttt ≤ sssss.

Gör så här för att prova rutinen: Mata in programmet. Gå sedan till steg 000. Skjut där in den rutin som skall skall ingå i DSE-loopen. Som demonstration är RCL 00 INT PRT lämpligt. Lagra sedan ett tal i R_0 och tryck RST R/S.

Den här rutinen kan inte användas som subrutin i t ex en modul. Därför skrev jag pgm 2 efter Markús' metod men gjorde om rutinen så att flagga 6 nollställs när loopen är färdig och ställer flaggan annars. Om vi antar att rutinen används i en modul och har pgm-nummer 02 kan den anropas som i pgm 03.

Den som kommer på något smartare sätt att ordna samordning mellan huvudprogram och modulprogram, var vänlig skriv till Utmaningen och berätta.

```
000  53  (      008  05   5     016  59 INT     024  59 INT
001  43 RCL     009  22 INV     017  65  ×      025  77  GE
002  00  00     010  28 LOG     018  03   3     026  00  00
003  75  -      011  85  +      019  22 INV     027  00  00
004  53  (      012  32 X:T     020  28 LOG     028  92 RTN
005  22 INV     013  00   0     021  54  )
006  59 INT     014  54  )      022  42 STO
007  65  ×      015  22 INV     023  00  00
```

Pgm 1. (Markús S. Markússon.) Problem 11: DSE-funktion.

```
000  76 LBL     009  65  ×      018  59 INT     027  77  GE
001  11  A      010  05   5     019  65  ×      028  00  00
002  53  (      011  22 INV     020  03   3     029  31  31
003  43 RCL     012  28 LOG     021  22 INV     030  22 INV
004  00  00     013  85  +      022  28 LOG     031  86 STF
005  75  -      014  32 X:T     023  54  )      032  06  06
006  53  (      015  00   0     024  42 STO     033  92 RTN
007  22 INV     016  54  )      025  00  00
008  59 INT     017  22 INV     026  59 INT
```

Pgm 2. (Red.) Problem 11: DSE-funktion för användning i modul.

Markús har också skickat ett snabbare program (pgm 4) till problem 13. Metoden är den samma som i mitt pgm 6 i PB 81-3, men den inre loopen, som genomlöps väldigt många gånger, är betydligt kortare.

Det kan kanske vara på plats att förklara metoden som används. Den kallas "insticksmetoden" i fri översättning och går till så att ett register i taget placeras in i den sorterade listan. Den listan består från början av ett register. Efter första insättning består således listan av två register. Ett tredje register jämförs med listan och placeras in på rätt ställe. Så fortsätter det tills alla register finns på rätt ställe i den sorterade listan.

Här nedan visas ett exempel på hur sortering går till. Talen till vänster om kolonet bildar den sorterade listan. Talet precis till höger om kolonet är nästa tal som skall sättas in i listan. Pilen markerar var insättningen skall göras. Talen inom parentes är den helt osorterade listan.

```
^503 : 087 ( 512   061   908   170   987 )
 087   503^: 512 ( 061   908   170   987 )
^087   503   512 : 061 ( 908   170   987 )
 061   087   503   512^: 908 ( 170   987 )
 061   087^  503   512   908 : 170 ( 987 )
 061   087   170   503   512   908^: 987
 061   087   170   503   512   908   987
```

Programmen söker uppifrån i den sorterade listan tills de finner ett tal som är mindre än eller lika med tal som skall sorteras in och sätter in det. Samtidigt som rätt ställe söks, flyttas registren om, så när rätt ställe har hittats, finns det ett tomt register att placera talet i.

```
000  76 LBL     008  00   0     016  76 LBL     024  02  02
001  11  A      009  00   0     017  24 CE      025  11  A
002  25 CLR     010  03   3     018  43 RCL     026  87 IFF
003  01   1     011  00   0     019  00  00     027  06  06
004  05   5     012  00   0     020  59 INT     028  24 CE
005  93   .     013  01   1     021  72 ST*     029  25 CLR
006  00   0     014  42 STO     022  00  00     030  92 RTN
007  00   0     015  00  00     023  36 PGM
```

Pgm 3. (Red.) Problem 11: Exempel på anrop av pgm 2.

```
000  76 LBL   ▷ 012  82 HIR     024  73 RC*     036  00  00
001  11  A      013  14  14     025  01  01     037  27  27
002  82 HIR     014  32 X:T     026  32 X:T     038  69 OP
003  04  04     015  01   1   ▷ 027  69 OP      039  21  21
004  03   3     016  85  +      028  30  30     040  32 X:T
005  82 HIR     017  67  EQ     029  73 RC*     041  72 ST*
006  34  34     018  00  00     030  00  00     042  01  01
007  85  +      019  46  46     031  72 ST*     043  61 GTO
008  00   0     020  42 STO     032  01  01     044  00  00
009  23 LNX     021  00  00     033  69 OP      045  12  12
010  42 STO     022  42 STO     034  31  31   ▷ 046  25 CLR
011  02  02     023  01  01     035  77  GE     047  92 RTN
```

Pgm 4. (Markús S. Markússon.) Problem 13: Snabbare sortering.

Markús har på ett utmärkt sätt visat hur man kan snabba upp ett program genom att ersätta två tester med ett test. Genom att placera det minsta tal som kan tänkas i det lägsta registret kan DSZ-testen tas bort eftersom det andra testet alltid lyckas till slut. Det minsta talet som finns, kallat $-\infty$, är förstås -9.999999999 99. För att ytterligare snabba upp programmet har Markús använt både R_0 och R_1 för indirekt adressering. 97 register kan alltså sorteras med hans program.

Tyvärr borde testet i steg 035-037 vara GT 0 27 (x>t 0 27) istället för GE 0 27. Det testet finns ju inte på 59:an så det enda som kan göras är att byta plats på x och t och använda INV GE test istället. Det har jag gjort i program 5.

Vad får då användningen av GE-testet för verkan? Dels kan programmet fastna i en oändlig slinga om ett av talen är -9.99... 99, dels går det långsammare om flera av talen skulle vara lika, eftersom programmet då söker längre än nödvändigt.

```
  000  76 LBL     013  14  14   ▷ 026  32 X:T     039  26  26
  001  11  A      014  32 X:T     027  69 OP      040  69 OP
  002  82 HIR     015  01   1     028  30  30     041  21  21
  003  04  04     016  85  +      029  73 RC*     042  72 ST*
  004  03   3     017  67  EQ     030  00  00     043  01  01
  005  82 HIR     018  00  00     031  72 ST*     044  61 GTO
  006  34  34     019  47  47     032  01  01     045  00  00
  007  85  +      020  42 STO     033  69 OP      046  12  12
  008  00   0     021  00  00     034  31  31   ▷ 047  25 CLR
  009  23 LNX     022  42 STO     035  32 X:T     048  92 RTN
  010  42 STO     023  01  01     036  22 INV
  011  02  02     024  73 RC*     037  77  GE
▷ 012  82 HIR     025  01  01     038  00  00
```

Pgm 5. (Red.) Problem 13: Säkrare version av pgm 4.

```
  000  00   0     046  28 LOG     092  00   0     138  52 EE
  001  00   0     047  75  -      093  95  =      139  01   1
  002  00   0     048  01   1     094  59 INT     140  00   0
  003  76 LBL     049  95  =      095  44 SUM     141  22 INV
  004  11  A      050  32 X:T     096  03  03     142  44 SUM
  005  36 PGM   ▷ 051  43 RCL     097  22 INV     143  05  05
  006  02  02     052  00  00     098  52 EE      144  43 RCL
  007  71 SBR     053  49 PRD     099  52 EE      145  06  06
  008  02  02     054  02  02     100  01   1     146  55  ÷
  009  39  39     055  49 PRD     101  00   0     147  01   1
  010  09   9     056  03  03     102  22 INV     148  52 EE
  011  00   0     057  49 PRD     103  44 SUM     149  01   1
  012  22 INV     058  04  04     104  02  02     150  00   0
  013  58 FIX     059  75  -    ▽ 105  97 DSZ     151  95  =
  014  22 INV     060  43 RCL     106  01  01     152  59 INT
  015  57 ENG     061  01  01     107  00  00     153  44 SUM
  016  25 CLR     062  95  =      108  51  51     154  07  07
  017  42 STO     063  22 INV     109  43 RCL     155  22 INV
  018  05  05     064  77  GE     110  02  02     156  52 EE
  019  42 STO     065  01  01     111  44 SUM     157  52 EE
  020  06  06     066  05  05     112  05  05     158  01   1
  021  42 STO     067  43 RCL     113  43 RCL     159  00   0
  022  07  07     068  03  03     114  03  03     160  22 INV
  023  02   2     069  55  ÷      115  44 SUM     161  44 SUM
  024  00   0     070  01   1     116  06  06     162  06  06
  025  42 STO     071  52 EE      117  43 RCL     163  25 CLR
  026  00  00     072  01   1     118  04  04     164  98 ADV
▷ 027  43 RCL     073  00   0     119  44 SUM     165  43 RCL
  028  00  00     074  95  =      120  07  07     166  07  07
  029  42 STO     075  59 INT     121  97 DSZ     167  99 PRT
  030  02  02     076  44 SUM     122  00  00     168  43 RCL
  031  75  -      077  04  04     123  00  00     169  06  06
  032  01   1     078  22 INV     124  27  27     170  99 PRT
  033  95  =      079  52 EE      125  43 RCL     171  43 RCL
  034  42 STO     080  52 EE      126  05  05     172  05  05
  035  01  01     081  01   1     127  55  ÷      173  99 PRT
  036  00   0     082  00   0     128  01   1     174  98 ADV
  037  42 STO     083  22 INV     129  52 EE      175  98 ADV
  038  03  03     084  44 SUM     130  01   1     176  98 ADV
  039  42 STO     085  03  03     131  00   0     177  98 ADV
  040  04  04     086  43 RCL     132  95  =      178  98 ADV
  041  01   1     087  02  02     133  59 INT     179  98 ADV
  042  02   2     088  55  ÷      134  44 SUM     180  91 R/S
  043  55  ÷      089  01   1     135  06  06     181  61 GTO
  044  43 RCL     090  52 EE      136  22 INV     182  01  01
  045  00  00     091  01   1     137  52 EE      183  62  62
```

Pgm 7. (Patrik Johansson.) Problem 15: Beräkning av summan $1^1+2^2+...+20^{20}$ exakt.

Patrik Johansson har gjort ett program för problem 15, beräkning av $1^1 + 2^2 + ... + 20^{20}$ (pgm 7). Så här skriver Patrik:

> Det är ganska enkelt att göra ett program som beräknar summan i problem 15. Därför har jag också försökt optimera programmet. Programmet är som synes ganska långt men i gengäld är det rätt så snabbt. Det körs i Fast Mode och beräkningen tar 2 min 45 s. Körningen sker genom att man läser in kortet, trycker A och läser in kortet igen. Vill man ha en till utskrift så trycker man R/S. Den som inte har skrivare trycker lämpligen RST och kontrollerar registren R_7-R_5.

Patriks program ger resultatet i figur 1, alltså $1^1 + 2^2 + ... + 20^{20}$ = 1068762122000 59554303215024. Samma resultat gav programmet som publicerades som i TISoft, förstås. Men det programmet tog 14 och en halv minut på sig! Patriks program tar som konstaterats ovan 2 min 45 s. (Till TISoft's programmets försvar kan sägas att det inte gick i fast mode och det var optimerat med avseende på steg och alltså var kortare.)

Patrik summerar totalresultet till R_7-R_5. R_4-R_2 används som "multiplikationsregister", dvs där beräknas n^n som sedan summeras till totalregistret. Han spar tid genom att inte direkt överföra totalregistrets minnessiffror till rätt register. Vid beräkning av n^n spars tid genom att ta reda på hur många gånger multiplikationen kan göras innan det bildas en minnessiffra till nästa register. Om ingen minnessiffra finns hoppas helt enkelt rutinen som tar hand om minnessiffror över. Det görs genom att beräkna 12/lg n, som talar om hur många gånger multiplikationen kan göras utan risk för minnessiffra. Men det skulle gå snabbare att på en gång beräkna antal gånger multiplikationen skall göras utan minnessiffra och använda det talet för ett Dsz-register. ▶

NÄR DU SKICKAR PGM TILL UTMANINGEN

var då vänlig att skicka med magnetkort om du har en TI-59. Det spar tid åt mig. Jag kommer givetvis att skicka dig blanka magnetkort som kompensation. (Skicka alltså inte originalkorten, utan gör en kopia.)

En enkel gardering mot felläsningar är att spela in programmet på båda sidor av kortet, om det är kortare än 241 steg (går in i ett block).

I Utmaningen i PB 81-2 gav jag problem 10, att rätta till ett labyrintprogram med en eller annan bug. Här i Sverige har ju en lösning inkommit. I USA har ingen lösning inkommit. Men Maurice Swinnen hittade ett labyrintprogram i en västtysk programkatalog. Han fick en kopia av programmet.

Underligt nog, var det programmet väldigt likt det "gamla" programmet, utom att det fungerar som det ska. (Jag vet inte vad det finns för samband mellan programmen.) Programförfattaren heter Andras Biek. (TI PPC Notes, v6n9/10 p17-18.)

Bruksanvisning:

Inmatning: Tryck 3 OP 17, mata in ,. . 9 och spela in block 1, 2 & 3 i den uppdelningen. (En ny "utmaning": Tag bort några av noparna och sätt in en rutin för automatisk uppdelningsändring; då kan man läsa in korten i normaluppdelning. Det bör räcka att ta bort noparna i slutet av programmet - efter steg 593 och framåt.)

Pgm 9. (Andreas Biek.) Problem 10: Labyrintprogram.

000 76 LBL	090 01 01	180 59 INT	270 42 STO	360 10 10	450 32 X:T	540 77 GE	630 00 0
001 19 D'	091 22 22	181 22 INV	271 15 15	361 09 9	451 43 RCL	541 04 04	631 95 =
002 53 (	092 22 INV	182 52 EE	272 65 ×	362 42 STO	452 00 00	542 31 31	632 69 OP
003 43 RCL	093 97 DSZ	183 97 DSZ	273 22 INV	363 00 00	453 67 EQ	543 61 GTO	633 06 06
004 19 19	094 00 00	184 16 16	274 74 SM*	364 73 RC*	454 05 05	544 05 05	634 04 4
005 23 LNX	095 01 01	185 01 01	275 00 00	365 00 00	455 46 46	545 74 74	635 06 6
006 33 X²	096 15 15	186 65 65	276 73 RC*	366 59 INT	456 19 D'	546 93 .	636 02 2
007 42 STO	097 87 IFF	187 92 RTN	277 00 00	367 72 ST*	457 45 Y×	547 01 1	637 07 7
008 19 19	098 01 01	188 01 1	278 22 INV	368 00 00	458 53 (	548 65 ×	638 03 3
009 65 ×	099 01 01	189 01 1	279 59 INT	369 97 DSZ	459 53 (	549 73 RC*	639 01 1
010 01 1	100 04 04	190 42 STO	280 69 OP	370 00 00	460 73 RC*	550 00 00	640 69 OP
011 00 0	101 71 SBR	191 00 00	281 10 10	371 03 03	461 15 15	551 59 INT	641 04 04
012 00 0	102 01 01	192 25 CLR	282 65 ×	372 64 64	462 85 +	552 69 OP	642 43 RCL
013 00 0	103 31 31	193 72 ST*	283 43 RCL	373 18 C'	463 73 RC*	553 10 10	643 18 18
014 54)	104 97 DSZ	194 00 00	284 12 12	374 92 RTN	464 00 00	554 95 =	644 69 OP
015 22 INV	105 15 15	195 97 DSZ	285 85 +	375 43 RCL	465 54)	555 74 SM*	645 06 06
016 59 INT	106 00 00	196 00 00	286 01 1	376 12 12	466 55 ÷	556 00 00	646 02 2
017 92 RTN	107 65 65	197 01 01	287 93 .	377 32 X:T	467 02 2	557 01 1	647 22 INV
018 76 LBL	108 71 SBR	198 93 93	288 01 1	378 43 RCL	468 00 0	558 44 SUM	648 44 SUM
019 15 E	109 01 01	199 92 RTN	289 95 =	379 10 10	469 95 =	559 12 12	649 18 18
020 55 ÷	110 22 22	200 19 D'	290 42 STO	380 15 E	470 32 X:T	560 43 RCL	650 22 INV
021 00 0	111 97 DSZ	201 65 ×	291 16 16	381 42 STO	471 93 .	561 12 12	651 49 PRD
022 32 X:T	112 00 00	202 08 8	292 22 INV	382 00 00	472 05 5	562 32 X:T	652 18 18
023 22 INV	113 00 00	203 95 =	293 97 DSZ	383 22 INV	473 22 INV	563 08 8	653 03 3
024 28 LOG	114 65 65	204 59 INT	294 16 16	384 67 EQ	474 87 IFF	564 77 GE	654 05 5
025 95 =	115 69 OP	205 22 INV	295 03 03	385 02 02	475 02 02	565 04 04	655 03 3
026 22 INV	116 05 05	206 28 LOG	296 40 40	386 54 54	476 04 04	566 03 03	656 01 1
027 59 INT	117 22 INV	207 95 =	297 19 D'	387 71 SBR	477 79 79	567 09 9	657 01 1
028 65 ×	118 86 STF	208 52 EE	298 65 ×	388 01 01	478 01 1	568 32 X:T	658 06 6
029 01 1	119 01 01	209 22 INV	299 02 2	389 47 47	479 22 INV	569 43 RCL	659 69 OP
030 00 0	120 92 RTN	210 52 EE	300 95 =	390 43 RCL	480 77 GE	570 10 10	660 04 04
031 95 =	121 68 NOP	211 42 STO	301 59 INT	391 13 13	481 05 05	571 15 E	661 43 RCL
032 59 INT	122 00 0	212 10 10	302 75 -	392 42 STO	482 46 46	572 42 STO	662 19 19
033 92 RTN	123 48 EXC	213 65 ×	303 19 D'	393 15 15	483 43 RCL	573 00 00	663 69 OP
034 76 LBL	124 16 16	214 01 1	304 45 Y×	394 10 E'	484 00 00	574 93 .	664 06 06
035 10 E'	125 84 OP*	215 00 0	305 05 5	395 61 GTO	485 32 X:T	575 01 1	665 98 ADV
036 01 1	126 00 00	216 85 +	306 95 =	396 03 03	486 43 RCL	576 74 SM*	666 71 SBR
037 74 SM*	127 05 5	217 01 1	307 50 I×I	397 40 40	487 15 15	577 00 00	667 02 02
038 00 00	128 42 STO	218 95 =	308 65 ×	398 76 LBL	488 77 GE	578 65 ×	668 00 00
039 43 RCL	129 15 15	219 42 STO	309 53 (	399 17 B'	489 04 04	579 44 SUM	669 08 8
040 00 00	130 92 RTN	220 14 14	310 43 RCL	400 01 1	490 93 93	580 14 14	670 06 6
041 65 ×	131 07 7	221 18 C'	311 12 12	401 42 STO	491 48 EXC	581 43 RCL	671 42 STO
042 43 RCL	132 04 4	222 92 RTN	312 85 +	402 12 12	492 00 00	582 14 14	672 01 01
043 15 15	133 65 ×	223 76 LBL	313 93 .	403 43 RCL	493 65 ×	583 59 INT	673 17 B'
044 22 INV	134 53 (	224 16 A'	314 09 9	404 12 12	494 93 .	584 42 STO	674 16 A'
045 28 LOG	135 05 5	225 01 1	315 75 -	405 32 X:T	495 01 1	585 14 14	675 97 DSZ
046 52 EE	136 75 -	226 42 STO	316 43 RCL	406 43 RCL	496 49 PRD	586 95 =	676 18 18
047 22 INV	137 43 RCL	227 12 12	317 15 15	407 10 10	497 00 00	587 59 INT	677 06 06
048 52 EE	138 15 15	228 43 RCL	318 32 X:T	408 15 E	498 95 =	588 42 STO	678 69 69
049 95 =	139 54)	229 12 12	319 43 RCL	409 42 STO	499 32 X:T	589 11 11	679 86 STF
050 44 SUM	140 22 INV	230 42 STO	320 13 13	410 00 00	500 43 RCL	590 86 STF	680 02 02
051 14 14	141 28 LOG	231 13 13	321 42 STO	411 22 INV	501 10 10	591 01 01	681 17 B'
052 92 RTN	142 33 X²	232 32 X:T	322 15 15	412 67 EQ	502 71 SBR	592 18 C'	682 71 SBR
053 76 LBL	143 95 =	233 43 RCL	323 95 =	413 04 04	503 01 01	593 92 RTN	683 02 02
054 18 C'	144 44 SUM	234 11 11	324 59 INT	414 18 18	504 60 60	594 76 LBL	684 00 00
055 04 4	145 16 16	235 15 E	325 44 SUM	415 71 SBR	505 42 STO	595 11 A	685 97 DSZ
056 42 STO	146 92 RTN	236 67 EQ	326 15 15	416 01 01	506 10 10	596 02 2	686 20 20
057 00 00	147 01 1	237 03 03	327 43 RCL	417 47 47	507 43 RCL	597 00 0	687 06 06
058 71 SBR	148 00 0	238 75 75	328 15 15	418 09 9	508 14 14	598 42 STO	688 00 00
059 01 01	149 42 STO	239 42 STO	329 67 EQ	419 87 IFF	509 71 SBR	599 17 17	689 92 RTN
060 22 22	150 00 00	240 00 00	330 02 02	420 02 02	510 01 01	600 22 INV	690 76 LBL
061 87 IFF	151 69 OP	241 01 1	331 92 92	421 04 04	511 60 60	601 86 STF	691 12 B
062 01 01	152 30 30	242 44 SUM	332 10 E'	422 29 29	512 42 STO	602 02 02	692 42 STO
063 01 01	153 73 RC*	243 12 12	333 93 .	423 65 ×	513 14 14	603 71 SBR	693 19 19
064 04 04	154 00 00	244 43 RCL	334 01 1	424 93 .	514 32 X:T	604 01 01	694 92 RTN
065 43 RCL	155 22 INV	245 12 12	335 74 SM*	425 07 7	515 42 STO	605 88 88	695 76 LBL
066 14 14	156 67 EQ	246 32 X:T	336 00 00	426 65 ×	516 15 15	606 01 1	696 13 C
067 55 ÷	157 01 01	247 43 RCL	337 61 GTO	427 19 D'	517 01 1	607 44 SUM	697 42 STO
068 01 1	158 51 51	248 11 11	338 02 02	428 95 =	518 00 0	608 17 17	698 20 20
069 00 0	159 92 RTN	249 15 E	339 92 92	429 42 STO	519 49 PRD	609 68 NOP	699 02 2
070 75 -	160 65 ×	250 22 INV	340 01 1	430 13 13	520 00 00	610 73 RC*	700 01 1
071 59 INT	161 09 9	251 67 EQ	341 44 SUM	431 22 INV	521 49 PRD	611 17 17	701 42 STO
072 42 STO	162 42 STO	252 02 02	342 12 12	432 97 DSZ	522 15 15	612 68 NOP	702 17 17
073 14 14	163 16 16	253 41 41	343 43 RCL	433 13 13	523 00 0	613 68 NOP	703 43 RCL
074 95 =	164 01 1	254 19 D'	344 12 12	434 05 05	524 63 EX*	614 68 NOP	704 17 17
075 69 OP	165 65 ×	255 45 Y×	345 32 X:T	435 46 46	525 15 15	615 42 STO	705 75 -
076 10 10	166 93 .	256 53 (	346 09 9	436 43 RCL	526 74 SM*	616 18 18	706 02 2
077 75 -	167 01 1	257 73 RC*	347 77 GE	437 12 12	527 00 00	617 98 ADV	707 00 0
078 01 1	168 85 +	258 00 00	348 02 02	438 85 +	528 43 RCL	618 02 2	708 95 =
079 95 =	169 22 INV	259 55 ÷	349 28 28	439 01 1	529 12 12	619 07 7	709 91 R/S
080 50 I×I	170 59 INT	260 06 6	350 44 SUM	440 95 =	530 42 STO	620 01 1	710 72 ST*
081 65 ×	171 22 INV	261 00 0	351 14 14	441 32 X:T	531 15 15	621 03 3	711 17 17
082 71 SBR	172 67 EQ	262 54)	352 43 RCL	442 43 RCL	532 10 E'	622 01 1	712 01 1
083 01 01	173 01 01	263 65 ×	353 14 14	443 10 10	533 01 1	623 04 4	713 44 SUM
084 31 31	174 77 77	264 01 1	354 55 ÷	444 15 E	534 44 SUM	624 69 OP	714 17 17
085 97 DSZ	175 43 RCL	265 93 .	355 01 1	445 42 STO	535 12 12	625 04 04	715 61 GTO
086 15 15	176 00 00	266 09 9	356 00 0	446 15 15	536 43 RCL	626 43 RCL	716 07 07
087 00 00	177 52 EE	267 75 -	357 95 =	447 67 EQ	537 12 12	627 17 17	717 03 03
088 97 97	178 09 9	268 93 .	358 59 INT	448 05 05	538 32 X:T	628 75 -	
089 71 SBR	179 95 =	269 01 1	359 42 STO	449 28 28	539 08 8	629 02 2	

Körning: Ändra uppdelningen genom att trycka 3 OP 17 och läs in tre kortsidor. Mata in ett slumptalsfrö och tryck B. Mata in antal labyrinter och tryck C (högst 10 labyrinter). Mata in antal rader i den första labyrinten och tryck R/S. Gör likadant för övriga labyrinter. För att starta utskriften, tryck A.

Rättelse: Problem 5, pgm 2 Utmaningen PB 81-3 s 5. Markús har hittat ett fel i sitt eget program. Så här skriver han:

Tal i intervallet 0.99995, 1 $\cdot 10^n$; n=-4, -3,..., 3 rundas upp till 10^n utan att ändra antal siffror för och efter decimalkommat. Således blir skrivkoden för t ex 9.9999 1340010101 vilket betyder C.000 och inte 10.00.

I program 6 har han rättat till felet. Observera att rutinen inte får köras med tiopotens- eller teknisk notation inställd.

000 76 LBL	038 32 X:T	076 01 01	114 65 ×
001 99 PRT	039 22 INV	077 12 12	115 01 1
002 42 STO	040 77 GE	078 42 STO	116 00 0
003 07 07	041 00 00	079 10 10	117 00 0
004 50 I×I	042 51 51	080 01 1	118 49 PRD
005 55 ÷	043 69 OP	081 00 0	119 11 11
006 29 CP	044 28 28	082 00 0	120 34 ΓX
007 67 EQ	045 69 OP	083 49 PRD	121 85 +
008 00 00	046 29 29	084 11 11	122 01 1
009 24 24	047 55 ÷	085 04 4	123 85 +
010 53 (	048 01 1	086 00 0	124 32 X:T
011 28 LOG	049 00 0	087 44 SUM	125 09 9
012 22 INV	050 95 =	088 11 11	126 77 GE
013 77 GE	051 58 FIX	089 29 CP	127 01 01
014 00 00	052 04 04	090 43 RCL	128 30 30
015 19 19	053 52 EE	091 09 09	129 02 2
016 85 +	054 22 INV	092 67 EQ	130 93 .
017 01 1	055 58 FIX	093 01 01	131 75 -
018 54)	056 42 STO	094 00 00	132 59 INT
019 59 INT	057 10 10	095 42 STO	133 44 SUM
020 77 GE	058 43 RCL	096 08 08	134 11 11
021 00 00	059 08 08	097 71 SBR	135 95 =
022 24 24	060 32 X:T	098 01 01	136 97 DSZ
023 32 X:T	061 04 4	099 12 12	137 08 08
024 42 STO	062 22 INV	100 43 RCL	138 01 01
025 08 08	063 44 SUM	101 07 07	139 14 14
026 42 STO	064 09 09	102 29 CP	140 92 RTN
027 09 09	065 22 INV	103 77 GE	141 25 CLR
028 22 INV	066 77 GE	104 01 01	142 32 X:T
029 28 LOG	067 01 01	105 09 09	143 07 7
030 95 =	068 41 41	106 02 2	144 01 1
031 32 X:T	069 25 CLR	107 00 0	145 00 0
032 93 .	070 42 STO	108 32 X:T	146 00 0
033 09 9	071 11 11	109 43 RCL	147 00 0
034 09 9	072 67 EQ	110 11 11	148 00 0
035 09 9	073 00 00	111 92 RTN	149 92 RTN
036 09 9	074 85 85	112 43 RCL	
037 05 5	075 71 SBR	113 10 10	

Pgm 6. (Markús S. Markússon.) Problem 5: Rättat skrivkodsprogram.

I förra Utmaningen skrev jag litet grann om primtalstest med Fermats sats. Här skall jag beskriva en metod att bevisa att ett tal är primtal. Först några beteckningar:

Med x mod y menas resten när x delas med y. Exempel: 5 mod 3 = 2, 7 mod 16 = 7, 17 mod 3 = 2.

Fermats sats (se PB 80-4 s 22 och förra Utmaningen) kan formuleras på följande sätt:

Om p är ett primtal, och a inte är en multipel av p så gäller

a^{p-1} mod p = 1,

vilket kan utläsas: resten vid divisionen a^{p-1}/p är lika med ett.

Som jag skrev i förra Utmaningen kan detta utnyttjas för att ta reda på om ett tal är primtal. Om man finner att a^{n-1} mod n $\neq$ 1, där a är ett godtyckligt heltal och n är ett heltal som man vill testa, vet man att n inte är ett primtal.

Om n inte är ett primtal kan man ofta snabbt finna ett tal a, som gör att a^{n-1} mod n $\neq$ 1. För vissa värden på n händer att a^{n-1} mod n ofta blir lika med ett, trots att n inte är ett primtal, men sådana n-värden är sällsynta.

Hur gör man nu för att beräkna t ex a^x mod n? Det är enkelt att göra med programmet i PB 80-4 s 22. För dem som inte har den årgången återger jag programmet här (pgm 8). (Fast jag rekommenderar att de som inte har den köper den.) Bruksanvisningen är följande:

1. Mata in a och tryck A.
2. Mata in n och tryck R/S (eller B).
3. Mata in x och tryck R/S (eller C). Resten visas.

Pgm 8 klarar tyvärr endast av n-värden mindre än $2\sqrt{10^{13}}$, ungefär 6 millioner. I praktiken blir nog vanliga primtalsprovare snabbare, men om dubbel precision används kan den här metoden bli snabbare.

000 76 LBL	020 00 00	040 42 STO	060 00 00
001 16 A'	021 92 RTN	041 02 02	061 02 2
002 75 -	022 76 LBL	042 95 =	062 32 X:T
003 53 (	023 12 B	043 29 CP	063 43 RCL
004 24 CE	024 42 STO	044 67 EQ	064 02 02
005 55 ÷	025 01 01	045 00 00	065 77 GE
006 43 RCL	026 75 -	046 55 55	066 00 00
007 01 01	027 01 1	047 43 RCL	067 36 36
008 54)	028 42 STO	048 00 00	068 43 RCL
009 52 EE	029 03 03	049 65 ×	069 00 00
010 22 INV	030 54)	050 43 RCL	070 65 ×
011 52 EE	031 92 RTN	051 03 03	071 43 RCL
012 65 ×	032 76 LBL	052 16 A'	072 03 03
013 43 RCL	033 13 C	053 42 STO	073 16 A'
014 01 01	034 58 FIX	054 03 03	074 22 INV
015 95 =	035 00 00	055 43 RCL	075 58 FIX
016 92 RTN	036 55 ÷	056 00 00	076 61 GTO
017 76 LBL	037 02 2	057 33 X²	077 00 00
018 11 A	038 75 -	058 16 A'	078 16 16
019 42 STO	039 59 INT	059 42 STO	

Pgm 8. (Red.) Beräkning av a^x mod n.

Nu skall jag beskriva ett sätt att bevisa att ett tal n är primtal. Sedan tänkte jag också beskriva en metod som är mer lämpad för programmering. Jag ger inga bevis; den intresserade kan läsa i Knuths The Art of Computer Programming, vol II, s 374 ff.

Om det finns ett heltal x för vilket ordningen av x modulo n är lika med n-1, så är n ett primtal. Ordningen av x modulo n är det lägsta positiva heltal k sådant att x^k mod n = 1. Ordningen av x kommer att vara n-1 om och endast om

x^{n-1} mod n = 1 och

$x^{(n-1)/p}$ mod n ≠ 1 för alla primtal p som delar n-1.

Vi behöver alltså bara finna alla primfaktorer till n-1 och försöka finna ett x så att de två villkoren ovan är uppfyllda, och beviset är färdigt. Det går faktiskt att använda olika värden på x för varje primfaktor p, och beviset gäller fortfarande. Det är lämpligt att endast låta x anta primtalsvärden.

Vi skall nu försöka bevisa att talet 6324551 är ett primtal. Till detta behöver vi två program, dels pgm 8, dels en primfaktorfinnare (använd lämpligen en från PB 81-3 s 18-19).

Vi börjar med första villkoret ovan. Läs in pgm 8. Vi väljer x = 3 och beräknar alltså $3^{6324550}$ mod 6324551. Det görs så här:

Mata in 3 och tryck A.
Mata in 6324551 och tryck R/S. Displayen visar 6324550, så tryck bara R/S. Efter en stund visas 1.

Första villkoret gäller alltså. Nu går vi på andra. Vi läser in ett primfaktorprogram och faktoruppdelar n-1, dvs 6324550. (Den som har en TI-58 rekommenderas att göra faktoruppdelningen först.)

Faktorerna som fås är 2, 5, 5, 126491. För dessa faktorer skall vi försöka hitta ett värde på x så att andra villkoret uppfylls. Vi läser in pgm 8 igen och börjar undersökningen. Här är resultatet:

x	p	$x^{(n-1)/p}$ mod n
3	2	1
3	5	-399980
3	126491	-1863678
5	2	1
7	2	-1

Som synes misslyckades testet när p var lika med 2, medan det lyckades på en gång för övriga p. På slutet sökte vi efter x-värden som skulle uppfylla villkoret och när x var lika med 7 lyckades det. Vi har alltså bevisat att 6324551 är ett primtal.

Tyvärr blir det ingen tidsvinst med så här små tal. Med snabbaste faktorfinnaren som finns (se PB 81-3 s 18-19) tog det ca 3 min 20 s att konstatera att talet är ett primtal. Talet 6324551 är tyvärr också det största primtal som pgm 8 klarar av. Se därför problem 18 i slutet av den här avdelningen.

Den här metoden är tyvärr inte så bra för stora n heller. n-1 blir för svår att faktorera när n växer. Som tur är finns det en annan metod, som bör gå att programmera på TI-59, som med en viss sannolikhet säger att n är ett primtal. Den metoden hoppas jag kunna beskriva i nästa Utmaningen.

Jag hoppas att den som har synpunkter eller frågor om det här med primtalstest skriver till mig. Kanske tycker ni att metoden förklarades för dåligt. Hör av er i så fall.

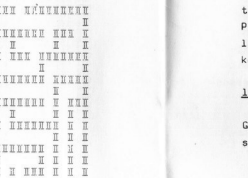
Markús Markússon har också sänt in en lösning till problem 14. Av utrymmeskäl tar jag med det programmet i nästa Utmaningen.

```
   1068762.                1.      LAB
1220005955.               30.      ZLN
4303215024.           913897.      RND
```

Figur 1. Utskrift av Pgm 7.

```
   01234567
00  0123456
10 789ABCDE
20 -FGHIJKL
30 MNOPQRST
40 .UVWXYZ+
50 ×÷fre(),
60 ↑%!/=*^R
70 ²?÷?∏▲Σ∑

   0123456789
00  012345678
10 789ABCDE78
20 -FGHIJKLMN
30 MNOPQRSTMN
40 .UVWXYZ+××
50 ×÷fre()×÷×
60 ↑%!/=*^R²?
70 ²?÷?∏▲Σ∑?
80  012345678
90 789ABCDE78
```

Figur 3. Skrivkodstabeller (problem 17).



Figur 2. Utskrift av Pgm 9 (samma som i PB 81-2 s 8 rättad).

Det är bara tre nya problem den här gången. Jag hoppas att läsarna skickar in problem till nästa gång.

16. Superprimtal.

"Talet 7193 är ett superprimtal. Det är nämligen ett primtal och det är också talen 719, 71 och 7, som man får genom att successivt stryka siffror i talet från höger. Skriv ett program som systematiskt bestämmer följden av superprimtal." (Lennart Råde, Äventyr med programmerbar miniräknare.)

Kommentar: Det finns endast ett ändligt antal superprimtal. TI-59:s noggrannhet är tillräcklig för att bestämma samtliga. Det är också möjligt att göra det på rimlig tid. (Jag har ett program, men inte i fast mode.)

Jag kräver att körtiden för alla bidrag anges. (Det enklaste sättet att lösa problemet är att testa ett udda heltal i taget och se om det är ett superprimtal. Den metoden använde jag när jag körde ett av mina första datorprogram på en stor, snabb dator under en praktikvecka. Det tog någon timme för att bara skriva ut alla superprimtal under en miljon. Undrar hur mycket långsammare 59:an är?)

17. Skrivkodstabell.

Gör ett program som skriver ut en skrivkodstabell som någon av tabellerna i figur 3. Programmet skall vara så snabbt som möjligt, bekvämt att använda och få plats på en kortsida. (Lars Hedlund.)

18. Beräkning av a^x mod n i dubbel precision.

Gör om pgm 8 så att det klarar av dubbelt så stora tal.

På återhörande
Björn Gustavsson ■

MAGNETKORT TILL PROGRAMMEN I "UTMANINGEN"

Från och med nu kan Du hos programförmedlare Bo Nordlin beställa magnetkort till samtliga program i ett nummer av "Utmaningen" för normalt programpris 30 (+10) kr.

Egen modul!

TI PPC Notes sista nummer för året har tyvärr kommit utan att den utlovade sammanställningen av program som var lämpliga för modulen kom med. I numret fanns emellertid följande program av en ej helt okänd programmerare, nämligen Björn Gustavsson. Maurice Swinnen skriver: "This is a good candidate for our own module. I mean the data packing routine, not Björn himself, of course."

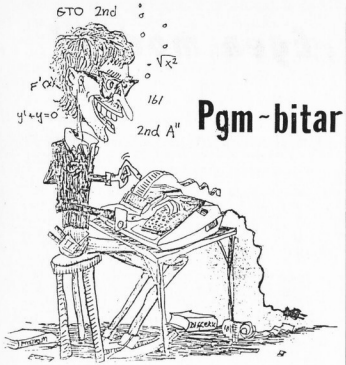
Det handlar alltså om datapackning, nämligen heltal av samma maximistorlek. Om man väljer t ex 12 multipliceras varje värde med en dignitet av 13 och summeras till ett register (idén kommer från 52-Notes V2N11P5). Programmet är nu gjort så att lagringen sker i RO2 och framåt. Numreringen för "pseudoregistren" är löpande från 0 och framåt. Man får själv hålla reda på totalt tillgängligt minnesutrymme - man vet att det går in 12/log(max+1) värden i ett verkligt register.

Bruksanvisning:

1. Ge maxvärde och anropa/tryck A.
2. För lagring, mata in talet och tryck x:t. Ge "pseudoregistrets" nummer och tryck B.
3. För återkallning, ge "pseudoregistrets" nummer och tryck C.
4. För att byta innehåll i x-registret och ett "pseudoregister", tryck x:t. Ge "pseudoregistrets" nummer och tryck D.

Detta program är snabbare än MU-08.

```
000  76 LBL    032  73 RC*    064  59 INT    096  17  17
001  13  C      033  01  01    065  75  -      097  76 LBL
002  53  (      034  75  -      066  53  (      098  11  B
003  82 HIR     035  53  (      067  24  CE     099  14  D
004  08  08     036  32 HIR     068  55  ÷      100  53  (
005  55  ÷      037  18  18     069  43 RCL     101  53  (
006  53  (      038  22 INV     070  00  00     102  32 XIT
007  01  1      039  59 INT     071  54  )      103  75  -
008  02  2      040  29 CP      072  59 INT     104  32 XIT
009  55  ÷      041  67  EQ      073  65  ×      105  13  C
010  43 RCL     042  00  00     074  43 RCL     106  82 HIR
011  00  00     043  47  47     075  00  00     107  07  07
012  28 LOG     044  43 RCL     076  54  )      108  54  )
013  54  )      045  00  00     077  32 XIT     109  65  ×
014  59 INT     046  65  ×      078  76 LBL     110  82 HIR
015  42 STD     047  53  (      079  12  B      111  18  18
016  01  01     048  43 RCL     080  53  (      112  54  )
017  54  )      049  00  00     081  53  (      113  74 SM*
018  53  (      050  45 Y^x     082  32 XIT     114  01  01
019  59 INT     051  82 HIR     083  82 HIR     115  82 HIR
020  65  ×      052  18  18     084  07  07     116  17  17
021  48 EXC     053  59 INT     085  75  -      117  76 LBL
022  01  01     054  54  )      086  32 XIT     118  15 LBL
023  54  )      055  52 EE      087  13  C      119  11  A
024  82 HIR     056  22 INV     088  54  )      120  42 STD
025  58  58     057  52 EE      089  65  ×      121  00  00
026  02  2      058  33 X^2     090  82 HIR     122  69 OP
027  44 SUM     059  54  )      091  18  18     123  20  20
028  01  01     060  82 HIR     092  54  )      124  92 RTN
029  82 HIR     061  08  08     093  74 SM*
030  68  68     062  54  )      094  01  01
031  53  (      063  53  (      095  82 HIR
```



UTVIDGAT DECIMALPUNKTSKNEP

Knepet demonstreras i det korta programmet nedan. Tryck A när ett tal står i displayen. Om displayen är mjuk skrivs "MJUK" och vice versa. Programmet skiljer mellan mjuk och hård nolla. Tryck t ex

O A så skrivs MJUK
O = A så skrivs HARD.

Det finns två labels A'. Om man har en hård display anropas den på steg 000. Om den däremot är mjuk anropas A' på steg 025! Man kan naturligtvis använda vilken label man vill, men den måste stå första gången på steg 000. (Koden 21 fordrar som de flesta vet syntetisk programmering t ex STO 21 BST BST Del SST).

000	76	LBL	009	01	1	018	16	A'	027	25	CLR
001	16	A'	010	06	6	019	69	DP	028	03	3
002	25	CLR	011	92	RTN	020	04	04	029	00	0
003	02	2	012	76	LBL	021	69	DP	030	02	2
004	03	3	013	11	A	022	05	05	031	05	5
005	01	1	014	21	2ND	023	25	CLR	032	04	4
006	03	3	015	93	.	024	92	RTN	033	01	1
007	03	3	016	21	2ND	025	76	LBL	034	02	2
008	05	5	017	21	2ND	026	16	A'	035	06	6
									036	92	RTN

Björn Gustavsson

I PB 81-1 sidan 24 finns en artikel om möjligheten att anropa ett modulprogram genom att trycka på R/S under ett programs exekvering. Problemet är att om modulen omedelbart hoppar tillbaka till det ordinarie programminnet hinner man ofta inte släppa R/S fort nog, utan programmet stannar. Detta kan man dock undvika om man placerar den label som modulen anropar långt bak i programminnet. Om den finns på steg 470 har man en hel sekund på sig, även om modulen anropar den såfort man trycker på R/S.

Anders Persson ■

Beräkning av KONTROLLSIFFRA

av Lars Kristiansson Mölnlycke

Beräkningen av kontrollsiffran i folkbokföringsnumret går till så att man (enligt figur) multiplicerar varannan siffra i numret med 2 (utom kontrollsiffran, vilken för övrigt kallas SCD: Self Check Digit). Därefter lägger man ihop tvärsumman av alla resultaten samt även de orörda siffrorna. Därefter drar man resultatets entalssiffra från 10 och tar heltalssiffran av det (om resultatet skulle bli 10).

6	3	0	3	0	7	5	0	5	
·2		·2		·2		·2		·2	
12	3	0	3	0	7	10	0	10	
1+2	+3		+3		+7	+1		+1	= 18

10
−18
2

630307-5052

Programmets bruksanvisning:

1. Mata in hela personnumret och tryck A. Om skrivare är ansluten skrivs det ut.
2. Efter en stund visas endera den kontrollsiffra som matades in som sista siffra i personnumret (ej blinkande), vilket betyder att kontrollsiffran är riktig. Eller visas den riktiga kontrollsiffran blinkande (och skrivs ut om skrivare är ansluten). Detta betyder alltså att det inmatade personnumret var felaktigt.

I PB 80-2 fanns ett program för "Viktors kvadrat". Där förfar man enligt samma princip, men det programmet beräknar kontrollsiffra för ett tal med inte nödvändigtvis 10 siffror och det är betydligt längre.

000	76	LBL	024	59	INT	048	22	INV	072	01	1
001	11	A	025	42	STO	049	67	EQ	073	00	0
002	98	ADV	026	01	01	050	00	00	074	95	=
003	99	PRT	027	95	=	051	53	53	075	22	INV
004	55	÷	028	22	INV	052	09	9	076	59	INT
005	09	9	029	87	IFF	053	22	INV	077	94	+/-
006	42	STO	030	00	00	054	86	STF	078	85	+
007	00	00	031	00	00	055	00	00	079	01	1
008	22	INV	032	54	54	056	44	SUM	080	95	=
009	28	LOG	033	67	EQ	057	00	02	081	22	INV
010	52	EE	034	00	00	058	01	1	082	59	INT
011	95	=	035	53	53	059	00	0	083	85	+
012	42	STO	036	85	+	060	49	PRD	084	01	1
013	01	01	037	24	CE	061	01	01	085	00	0
014	86	STF	038	75	-	062	97	DSZ	086	95	=
015	00	00	039	53	(	063	00	00	087	67	EQ
016	25	CLR	040	24	CE	064	00	00	088	00	00
017	42	STO	041	55	÷	065	20	20	089	93	93
018	02	02	042	09	9	066	43	RCL	090	69	DP
019	32	XIT	043	54	)	067	01	01	091	99	99
020	43	RCL	044	59	INT	068	32	XIT	092	99	PRT
021	01	01	045	65	×	069	43	RCL	093	92	RTN
022	00	00	046	09	9	070	02	02	094	81	RST
023	22	INV	047	95	=	071	55	÷			

Programladdare

av Björn Gustavsson

Det här är ett program som skulle kunna vara med i modulen. Det gör det möjligt att mata in program utan att gå in i programmeringsläge!

Programmet bygger på samma princip som Sven Östberg redogjorde för i "Något om TI-59's datorliknande egenskaper" i 79-3&4 och 80-2, nämligen att programkoder vid ändrad uppdelning bildar ett minnesregister och tvärtom - alltså att programkoder kan fås av innehållet i ett register.

Bruksanvisning

1. Mata in ett registernummer och tryck eller anropa E.
2a. Mata in en kod och tryck eller anropa A.
ELLER
2b. Mata in en till fem koder och tryck/anropa B. Koderna kommer inte att separeras.
3. När alla koder är inmatade, tryck/anropa C. Uppdelningen kan nu ändras och det inmatade programmet kan köras med SBR nnn.

Exempel

Vi skall mata in följande program: LBL A STO 00 LBL CE RCL 00 X DSZ 0 CE 1 = RTN. Vi vill att programmet skall börja vid steg 480, så vi matar in 59 som registernummer och trycker E. Sedan matar vi in koderna så här:

7611 B 4200 B 7624 B 4300 B 65 A 970024 B 1 A 95 A 92 A C

För att köra det inmatade programmet, tryck först 2nd CP för att radera laddningsprogrammet. Ändra uppdelningen genom att trycka 5 2nd Op 17. Mata sedan in t ex 5 och tryck A. 120 kommer att visas (du kanske har upptäckt att det är ett fakultetsprogram).

Programmet kan listas genom att trycka GTO 480 2nd List. Som synes finns det en massa "2nd" i det (kod 26, alltså 2nd 2nd). De kommer inte att störa programmet, eftersom de fungerar som NOP. (Med den ytterligare fördelen att koden kan placeras i alla steg i minnet.)

A-rutinen används för att mata in enstaka koder. (T ex 95, 92, 00.) B-rutinen används för koder som inte får skiljas åt av "2nd", som t ex 4201 (STO 01) eller 970024 (DSZ 0 CE). Om nödvändigt kommer B-rutinen att fylla delar av ett register med "2nd" och placera koderna i nästa register för att undvika separering. (Det går även att mata in enstaka koder med B-rutinen, men ibland upptas då mer utrymme än om inmatningen hade gjorts med A.)

Användningsområden: Programmet skulle kunna användas som subrutin i en TI-57 kompilator som tar emot tangent-koder för TI-57 och gör om dessa till koder för TI-59 samt laddar dessa direkt och kör programmet!!! Det skulle också vara möjligt att göra en BASIC-kompilator, men en sådan måste då troligen finnas i en modul.

000	76	LBL	024	07	7	048	02	2	072	54	)
001	12	C	025	67	EQ	049	06	6	073	53	(
002	43	RCL	026	00	00	050	44	SUM	074	94	+/-
003	01	01	027	40	40	051	02	02	075	65	×
004	29	CP	028	32	XIT	052	43	RCL	076	01	1
005	67	EQ	029	48	EXC	053	02	02	077	00	0
006	00	00	030	01	01	054	94	+/-	078	00	0
007	39	39	031	44	SUM	055	64	PD+	079	75	-
008	02	2	032	02	02	056	00	00	080	03	3
009	42	STO	033	93	.	057	69	DP	081	54	)
010	11	A	034	00	0	058	30	30	082	22	INV
011	61	GTO	035	01	1	059	01	1	083	28	LOG
012	13	C	036	49	PRD	060	53	(	084	52	EE
013	76	LBL	037	02	02	061	48	EXC	085	22	INV
014	11	A	038	00	0	062	01	01	086	52	EE
015	69	DP	039	92	RTN	063	65	×	087	72	ST×
016	21	2ND	040	01	1	064	93	.	088	00	00
017	48	EXC	041	00	0	065	01	1	089	00	0
018	01	01	042	32	XIT	066	42	STO	090	92	RTN
019	32	XIT	043	43	RCL	067	02	02	091	44	SUM
020	01	1	044	01	01	068	75	-	092	02	02
021	67	EQ	045	77	GE	069	59	INT	093	43	RCL
022	00	00	046	00	00	070	49	PRD	094	02	02
023	60	60	047	91	91	071	02	02	095	94	+/-

096	64	PD+	120	52	EE	144	82	HIR	480	26	2ND
097	00	00	121	22	INV	145	54	54	481	76	LBL
098	69	DP	122	52	EE	146	11	A	482	11	A
099	30	30	123	54	)	147	01	1	483	42	STO
100	00	0	124	82	HIR	148	00	0	484	00	00
101	42	STO	125	04	04	149	00	0	485	76	LBL
102	01	01	126	69	DP	150	82	HIR	486	24	CE
103	92	RTN	127	23	23	151	44	44	487	26	2ND
104	76	LBL	128	43	RCL	152	97	DSZ	488	26	2ND
105	12	B	129	03	03	153	03	03	489	43	RCL
106	53	(	130	32	XIT	154	01	01	490	00	00
107	29	CP	131	53	(	155	41	41	491	65	×
108	55	÷	132	06	6	156	00	0	492	97	DSZ
109	67	EQ	133	75	-	157	92	RTN	493	00	00
110	01	01	134	43	RCL	158	76	LBL	494	24	CE
111	15	15	135	01	01	159	15	E	495	26	2ND
112	54	)	136	54	)	160	42	STO	496	26	2ND
113	28	LOG	137	77	GE	161	00	00	497	01	1
114	59	INT	138	01	01	162	00	0	498	95	=
115	42	STO	139	41	41	163	42	STO	499	92	RTN
116	03	03	140	13	C	164	01	01	500	26	2ND
117	22	INV	141	82	HIR	165	92	RTN	501	26	2ND
118	28	LOG	142	14	14				502	26	2ND
119	53	(	143	59	INT				503	26	2ND

Avancerade programmeringsmetoder -1

av BJÖRN GUSTAVSSON

I den här artikelserien tänker jag presentera olika avancerade programmeringsmetoder. De flesta programmen kommer inte, i sig själva, att ha någon praktisk användning. Men de kommer förhoppningsvis att visa metoder som används på datorer, och kan kanske även hjälpa till att optimera program för TI-59.

Den här första artikeln skall handla om sökning efter ett tal i en tabell (ett antal register). I följande artiklar har jag bland annat tänkt ta upp BASIC och BASIC-kompilatorer.

SÖKNING

Vi skall här titta på hur man snabbt kan leta upp information i en dators eller räknedosas minne. Ju större minne som finns, desto viktigare blir det att välja rätt sökmetod. Även om man har ett relativt litet minne, som i TI-58/59 lönar det sig att välja rätt metod. I det här numret skall två metoder beskrivas; några till kommer i nästa nummer. Först några beteckningar:

Informationen är uppdelad i poster (eng "record"). I ett personregister kan varje post bestå av t ex personnummer, namn, adress m m. I varje post finns också en nyckel (eng "key"). Vi förutsätter att alla nycklar är olika. I vårt exempel med personregistret är det lämpligt att personnumret är nyckeln.

Vårt problem är att från ett s k argument K hitta den post som har K som nyckel. När det gäller personregistret skall man alltså kunna mata in personnummer och få fram allt som är lagrat om personen.

En sökning är antingen lyckad (posten innehållande nyckeln som är lika med K hittad), eller misslyckad (det finns ingen sådan post).

Att lägga upp personregister med hjälp av TI-59 är ju litet svårt, så det skall vi inte göra. I programmen i den här artikeln består nycklarna av heltal större än noll med högst 10 siffror. De som har skrivare bör betrakta nycklarna som skrivkoder. De som inte har skrivare kan t ex betrakta nycklarna som personnummer.

Trots att målet för sökningen är att finna informationen som är förknippad med K, kommer vi endast att bry oss om nycklarna. Detta för att göra programmen lättförståeligare. Programmen kommer endast att visa i vilket register som den funna nyckeln fanns i. Om sökningen misslyckades visas 0 (ingen nyckel är lagrad i R_0).

I praktiken, om man vet i vilket register nyckeln finns, är det också lätt att hitta motsvarande post. Vi kan t ex ha alla nycklar i R_1 - R_{49} och motsvarande poster i R_{51} - R_{99}. Postens registernummer blir då nyckelns registernummer plus 50. Om en nyckel finns i R_{10} finns motsvarande post i R_{60}.

Det är också möjligt att packa information i samma register som nyckeln. Lämpligen består då informationen av decimaldelen av talet och nyckeln av heltalsdelen. Om sådan lagring används kan man sätta in INT efter alla RC∎ i programmen.

Sekventiell sökning

Vi skall börja med att titta på det enklaste sättet att söka efter något: nämligen sekventiell sökning. Det innebär helt enkelt att man börjar i ena änden av tabellen och jämför K med alla nycklar tills man hittar K eller tills tabellen är slut.

Mer in i detalj kan metoden beskrivas som en algoritm:

Algoritm S (sekventiell sökning). Den här algoritmen söker efter ett argument K i en tabell med N stycken nycklar K_1, K_2, K_3... K_N. Vi antar att $N \geq 1$.

S1. [Initiera.] Sätt i←N.
S2. [Jämför.] Om K=K_i, skriv ut i och stanna. (Sökningen lyckad.)
S3. [Nästa nyckel.] Minska i med 1.
S4. [Slut?] Om i>0, gå tillbaka till steg S2. I annat fall skriv ut i (i = 0) och stanna. (Sökningen misslyckad.)

```
000  76 LBL     009  11  A      018  43 RCL     027  97 DSZ
001  10 E'      010  98 ADV     019  99  99     028  00  00
002  25 CLR     011  69 DP      020  42 STD     029  00  00
003  01  1      012  00  00     021  00  00     030  22  22
004  00  0      013  69 DP      022  73 RC+     031  43 RCL
005  69 DP      014  02  02     023  00  00     032  00  00
006  17  17     015  69 DP      024  67  EQ     033  99 PRT
007  92 RTN     016  05  05     025  00  00     034  92 RTN
008  76 LBL     017  32 XIT     026  31  31
```

Pgm 1a. Sekventiell sökning.

En sak bör kanske förklaras, nämligen "←". Pilen betyder: Sätt variabeln (register på TI-59) till vänster lika med värdet till höger. "i←N" betyder "sätt i lika med värdet av N". Det kan också stå "i←i-1". Det betyder: "Beräkna i-1 och ge i det värdet." i minskas alltså med 1.

Med hjälp av Algoritm S är det ganska enkelt att skriva ett program för TI-59 (med några ändringar går det in på TI-58/58C). Varje variabel i algoritmen tilldelas ett register. R_0 är i, R_{99} är N, och tabellen är R_1 - R_N. (Se pgm 1a.)

Först i programmet finns en initieringsrutin inledd med LBL E' som ställer in rätt uppdelning. Efter LBL A kommer sedan sökrutinen. ADV DP 00 DP 02 DP 05 skriver ut argumentet K tolkad som skrivkod. Därefter läggs K in i T-registret, och i (R_0) får värdet av N (R_{99}) genom RCL 99 STO 00. (S1)

RC∎ 00 X=T 0 31 jämför K med K_i och hoppar till utskriftsrutinen om lika. (S2)

DSZ 0 0 22, slutligen, exekverar S3 och S4 samtidigt. Om R_0=0, sker inget hopp; i stället utförs RCL 00 PRT (skriver noll), och programmet stoppas.

Hur lång tid tar det att köra programmet? Söktiden för sökning efter en nyckel som inte finns är ungefär 37,5 s på min TI-59 (N=97). Detta är det värsta tänkbara fallet, hela tabellen måste ju sökas i genom. Man kan visa, om alla nycklar har lika stor sannolikhet att bli sökta för, att i genomsnitt söks ungefär halva tabellen i genom. Det tar ungefär 37,5/2=18,75 s för N=97.

Innan vi lämnar sekventiell sökning skall vi titta på hur programmet kan snabbas upp. Det finns faktiskt en bättre metod. En allmän princip för att snabba upp program lyder: När en inre loop i ett program testar två eller flera villkor, bör man försöka nedbringa det till ett villkor.

Detta kan göras så här:

Algoritm Q (snabb sekventiell sökning). Den här algoritmen är densamma som Algoritm S, förutom att nycklarna kallas K_2, K_3, K_4... K_N.

Q1. [Initiera.] Sätt i←N och sätt K_1←K.
Q2. [Jämför.] Om K=K_i gå till steg Q4.
Q3. [Nästa nyckel.] Minska i med 1 och gå till steg Q2.
Q4. [Lyckad sökning?] Om $i \geq 2$ har sökningen lyckats; annars har den misslyckats.

Pgm 1b har gjorts efter Algoritm Q. Den inre loopen är ett steg kortare än i Pgm 1a. Exekveringstiden är ungefär 35 s. Sparad tid är alltså knappt 2,5 s.

Varför tog jag då över huvud taget upp Algoritmen Q? För det första för att det visar hur man kan ersätta två villkor med ett. För det andra för att tidsskillnaden mellan algoritmerna kan väntas vara större på andra räknare/datorer.

Anledning till den ringa tisskillnaden på TI-59 är dels att loopen genomlöps för få gånger, dels den kraftfulla DSZ-instruktionen (som gör två steg på en gång) som snabbar upp Pgm 1a.

(Samma knep att byta ut två villkor mot ett används i Pgm 4 i Utmaningen i det här numret.)

Låt oss till slut lämna sekventiell sökning och gå över till effektivare metoder!

```
000  76 LBL     011  69 DP      022  42 STD     033  43 RCL
001  10 E'      012  00  00     023  00  00     034  00  00
002  25 CLR     013  69 DP      024  73 RC+     035  69 DP
003  01  1      014  02  02     025  00  00     036  20  20
004  00  0      015  69 DP      026  69 DP      037  67  EQ
005  69 DP      016  05  05     027  30  30     038  00  00
006  17  17     017  42 STD     028  22 INV     039  42 STD
007  92 RTN     018  01  01     029  67  EQ     040  43 RCL
008  76 LBL     019  32 XIT     030  00  00     041  00  00
009  11  A      020  43 RCL     031  24  24     042  99 PRT
010  98 ADV     021  99  99     032  69 DP      043  92 RTN
```

Pgm 1b. Snabb sekventiell sökning.

```
000  76 LBL     018  08  08     036  16  16     054  65  65
001  10 E'      019  43 RCL     037  95  =      055  77  GE
002  25 CLR     020  99  99     038  55  ÷      056  00  00
003  01  1      021  85  +      039  02  2      057  27  27
004  00  0      022  01  1      040  95  =      058  43 RCL
005  69 DP      023  95  =      041  59 INT     059  00  00
006  17  17     024  82 HIR     042  42 STD     060  82 HIR
007  92 RTN     025  06  06     043  00  00     061  06  06
008  76 LBL     026  76 LBL     044  67  EQ     062  61 GTD
009  11  A      027  43 RCL     045  00  00     063  00  00
010  98 ADV     028  00  0      046  66  66     064  31  31
011  69 DP      029  82 HIR     047  73 RC+     065  43 RCL
012  00  00     030  05  05     048  00  00     066  00  00
013  69 DP      031  82 HIR     049  32 XIT     067  99 PRT
014  02  02     032  15  15     050  82 HIR     068  92 RTN
015  69 DP      033  85  +      051  18  18
016  05  05     034  32 XIT     052  67  EQ
017  82 HIR     035  82 HIR     053  00  00
```

Pgm 2. Binär sökning.

Binär sökning

Om vi lagrar nycklarna i stigande nummerordning eller alfabetisk ordning kan vi snabba upp sökningen. Tänk dig att någon ger dig en stor telefonkatalog och säger: "Sök upp den person som har nummer 31 35 02." Enda sättet att klara den uppgiften är att göra en sekventiell sökning, dvs börja på första sidan och söka tills numret hittas (eller katalogen tar slut och numret alltså inte fanns; hemska tanke).

Tänk sedan på hur man hittar ett namn i katalogen, vilket kan göras betydligt snabbare. Det viktiga är alltså att ordning har en stor betydelse.

En tabell i en räknare/dator kan sökas med följande metod: Jämför K med den mittersta nyckeln i tabellen. Om K är lika med nyckeln har sökningen lyckats. Om K är mindre än nyckeln kan hela övre halvan av tabellen uteslutas och vi vet att den sökta nyckeln finns i nedre tabellhalvan. Å andra sidan, om K är större än nyckeln måste av samma skäl den sökta nyckeln ligga i övre halvan av tabellen.

Efter ett enda test har nalva tabellen eliminerats! Genom att jämföra K med den mittersta nyckeln i den kvarvarande tabellen kan på nytt hälften de kvarvarande nycklarna uteslutas. Förfarandet upprepas tills nyckeln har hittats eller man vet att den inte finns i tabellen.

När man konstruerar algoritmen för sökningen måste man vara noggrann med detaljerna. Ett problem som uppkommer är att om en tabell har ett jämnt antal nycklar finns ingen unik mittersta nyckel. Det är inte heller självklart när man skall sluta om nyckeln inte finns i tabellen.

Följande algoritm fungerar:

Algoritm B (binär sökning). Den här algoritmen söker efter ett givet argument K i en tabell med nycklarna $K_1 < K_2 < K_3 \ldots K_N$.

B1. [Initiera.] Sätt v←0, h←N+1.

B2. [Beräkna mittpunkt.] Sätt i←⌊(v+h)/2⌋. (⌊⌋ betyder matematisk heltalsdel.)

B3. [Misslyckat?] Om i=v, skriv 0 och avbryt.

B4. [Jämför.] Om K=K_i, skriv i och avbryt (lyckat). Om K<K_i, sätt h←i och gå till steg B2. Om K>K_i, sätt v←i och gå till steg B2.

Från Algoritm B har jag gjort Pgm 2. Observera följande trick:

```
026 LBL
027 RCL
028  0
```

Om körningen görs från steg 026 ignoreras steg 026-027 (LBL RCL) och resultatet blir en nolla i displayen. Om däremot inhopp görs vid steg 027 blir resultatet RCL 0. Ett liknande trick används vid steg 065-066, där ibland inhopp till steg 066 görs med noll som resultat.

Exekveringstiden för en misslyckad sökning är 10 s, en klar förbättring. En lyckad sökning går något snabbare.

Det var allt för den här gången. Nästa gång tänker jag fortsätta med en ännu snabbare sökmetod. Den som har synpunkter eller frågor får gärna skriva till mig. (Adressen finns i Utmaningen.)

Litteratur

D. E. Knuth, The Art of Computer Programming, Vol. III (Sorting and Searching), Addison-Wesley 1975.

D. E. Knuth, Algorithms, Scientific American april 1977, s 63-80.

Knuth har skrivit tre böcker i serien The Art of Computer Programming. (Sju stycken är planerade.) Den som är intresserad av att köpa böckerna bör kontakta Föreningen Programbiten för eventuellt samköp utomlands. (I Sverige fick jag betala 263:- för del III enbart.) ■

VI GICK PÅ ETT APRILSKÄMT!!!

I förra numret efterlyste vi gamla TI-58:or och 59:or med märkliga egenskaper. Vi hade tagit uppgiften från vår beundrade kollega TI PPC Notes utan att begripa att dess eljest så omdömesgille redaktör Maurice Swinnen gått på ett aprilskämt! Maurice skriver att han lämnat över Worthington & Regelmans brev till några granskare för utlåtande och när de dröjde länge med sitt svar publicerade Maurice det hela ändå. Emil Regelman har sedan dess ej avhörts och Maurice tror han är på sjukhus för skrattkramper och urledskrattad käke, Kanske är han nu bättre så berätta inte för honom att vi också tog in hans "nyhet".

Diofantiska ekvationen
ax + by = c

av GÖSTA BLUME

Den obestämda ekv ax + by = c, där vi antar att a>0, kan skrivas ax = c + y·(-b), vilket uttryck kan uppfattas som en kongruens x·a ≡ c (mod b), dvs en okänd multipel av a är kongruent med c (mod b). Symboliskt kan x då skrivas x ≡ c/a (mod b), alltså som en "kongruenskvot", vilket inte är detsamma som en algebraisk kvot, men som visar sig ha vissa egenskaper som påminner om en sådans. Den viktigaste av dessa egenskaper är, att (cs)/(as) ≡ c/a, dvs "bråket" kan förkortas och förlängas efter samma regler som vanliga bråk. Dessutom är (c+b)/a ≡ c/a, där b är modulen, dvs täljaren (eller nämnaren) kan ökas (eller minskas) med (en multipel av) modulen.

Dessa båda egenskaper kan man nu utnyttja för att steg för steg minska nämnaren till 1, varvid täljaren ger värdet på x.

Ex. 1024 x - 15625 y = 8404 ⇒ 1024x ≡ 8404 (15625)
x ≡ 8404/1024 = 2101/256 ≡ 17726/256 = 8863/128 ≡
≡ 24488/128 = 3061/16 ≡ 18686/16 = 9343/8 ≡ 24968/8 =
=3121 (som ger y ≡ 204).

Trots de rätt stora talvärdena gick reduktionen här ganska snabbt tack vare att nämnaren var uppbyggd av många små faktorer. För nämnare som är stora primtal eller produkter av sådana primtal kan metoden knappast användas, eftersom körtiderna då blir för långa. I sådana fall fordras en något omständligare och mer aggressiv behandling av problemet än den enkla sökningsmetod som här tillämpas och som med marginal ryms i TI-57.

Pgm fungerar så här. Först bestäms med hjälp av Euklides' algoritm den största gemensamma faktorn i c och a varefter bråket c/a förkortas med denna faktor. Detta sker i pos 000 - 032. Nämnaren i det förkortade bråket visas under pausen i pos 037. Skulle nämnaren vara =1 är problemet löst och resultatet meddelas i Lbl B. I annat fall ökas täljaren med b (pos 042 - 049) och det nya bråket behandlas på samma sätt.

I Lbl B visas det erhållna x-värdet (pos 054 - 055), varefter motsvarande y-värde beräknas ur den givna ekv (pos 057 - 070).

Lbl A är inmatningsrutinen: a, b och c slås in i denna ordning (med R/S emellan).

-o0o-

Vid användning av pgm måste följande iakttas:

1. Endast heltalslösningar är av intresse och pgm är gjort mot denna bakgrund
2. Om i ekv ax + by = c, a och b har en gemensam faktor, så måste också c innehålla samma faktor för att ekv skall vara lösbar i hela tal
3. Om ekv har någon heltalslösning alls, så har den oändligt många sådana, bland vilka pgm ger en, säg (x_o, y_o)
4. Den allmänna lösningen kan då skrivas

$$\left.\begin{aligned} x &= x_o \pm b\cdot k \\ y &= y_o \mp a\cdot k \end{aligned}\right\}$$

där k är ett godtyckligt heltal. Med hjälp av dessa samband kan man ofta beräkna k-värden som ger lämpliga lösningar. I regel ger dock pgm direkt minsta värden, resp minsta positiva värden (om sådana lösningar finns)

Slutligen må erinras om vikten av att vid användning av detta ganska primitiva pgm välja lämpligaste möjliga värde på a (små ingående faktorer eller, i brist på sådana, minsta möjliga a). Är t ex den givna ekv 37x -400y = 129, tar det endast tredjedelen så lång tid att få svar om jag skriver ekv som 400y - 37x = = -129 och slår in koeff i denna ordning (dvs beräknar Y först), som om jag tar koeff i den ursprungliga ordningen. Är både a och b stora primtal, blir körtiden enligt detta pgm orimligt lång.

Ex 12x + 19y = 35
Slå in 12 på A, 19 på R/S, 35 på R/S
x_o = -5, R/S ger y_o = 5

$$\left.\begin{aligned} x &= -5 + 19k \\ y &= 5 - 12k \end{aligned}\right\} \quad k = 1 \text{ ger } x = 14,\ y = -7$$

Ex 17x - 18y = 11 ⇒ 18y - 17x = -11
Slå in 18 på A, -17 på R/S, -11 på R/S
y_o = 6, R/S ger x_o = 7

$$\left.\begin{aligned} x &= 7 + 18k \\ y &= 6 + 17k \end{aligned}\right\} \quad k = 3 \text{ ger } x = 61,\ y = 57$$

```
000  43 RCL     024  43 RCL     048  42 STO     072  11  A
001  03  03     025  03  03     049  03  03     073  42 STO
002  75  -      026  50 IX1     050  29 CP      074  02  02
003  53  (      027  22 INV     051  81 RST     075  42 STD
004  24 CE      028  49 PRD     052  76 LBL     076  04  04
005  55  ÷      029  01  01     053  12  B      077  42 STD
006  43 RCL     030  22 INV     054  43 RCL     078  06  06
007  04  04     031  49 PRD     055  01  01     079  50 FIX
008  54  )      032  02  02     056  91 R/S     080  00  00
009  52 EE      033  01  1      057  65  ×      081  29 CP
010  22 INV     034  32 X:T     058  43 RCL     082  91 R/S
011  52 EE      035  43 RCL     059  06  06     083  94 +/-
012  65  ×      036  02  02     060  75  -      084  42 STD
013  43 RCL     037  66 PAU     061  43 RCL     085  05  05
014  04  04     038  67  EQ     062  07  07     086  91 R/S
015  42 STD     039  12  B      063  54  )      087  42 STD
016  03  03     040  42 STD     064  55  ÷      088  01  01
017  95  =      041  04  04     065  43 RCL     089  42 STD
018  42 STD     042  43 RCL     066  05  05     090  03  03
019  04  04     043  05  05     067  95  =      091  42 STD
020  22 INV     044  44 SUM     068  22 INV     092  07  07
021  67  EQ     045  01  01     069  58 FIX     093  81 RST
022  00  00     046  43 RCL     070  91 R/S
023  00  00     047  01  01     071  76 LBL
```

BRIDGE-givar

av MARKÚS S MARKÚSSON ISLAND

Från Markús S. Markússon i Garðabær i Island har vi fått det här bidraget. Han skrev det på engelska och det skulle väl ha förståtts av majoriteten av våra läsare, men för "principens" skull har vi ändå översatt det. Dessutom har vi gjort ett litet ingrepp i programmet - på Björn Gustavssons förslag har vi satt in den ändrade uppdelningen i programmet.

I Programbiten 80-2 finns det ett program med bridgegivar av Agne Swerin. När jag såg det blev jag förtjust och knappade omedelbart in det i min TI-59 och började använda det mycket. Några månader senare hade jag blivit ganska trött på utskriftsformatet och på att renons i klöver inte skrevs ut när den förekom. Jag bestämde mig därför att rätta till det hela. Under de följande dagarna lyckades jag skriva ett program som visar fördelningen av kort på nord, syd, öst och väst. Det visar också alltid renons när sådan uppträder. En ytterligare finess är att giv och vilka som är i riskzonen anges.

Uppenbarligen får med denna utskrift antalet kort i en färg inte överstiga tio. Det besvärar mig emellertid inte särskilt mycket, fastän jag en gång hade tio kort i en färg (med det enda saknade honnörskortet, spader dam, hos partnern!). Jag väntar inte att det skall inträffa igen, ännu mindre att det skall bli mer än tio i samma färg. Om jag inte tar fel är chansen att få fler än tio kort i samma färg bara $\binom{4}{1}\binom{13}{11}\binom{13}{2}/\binom{52}{13} = 1/26.093.588!$

Programmet använder samma metod som Agne Swerins program, dvs mod 13-metoden. Riskzon och giv alternerar i en cykel på 16 givar så att första gången nord är giv och ingen i riskzonen. Så vitt jag vet är därför programmet i enlighet med internationella tävlingsnormer.

Jag skall nu ge en kort beskrivning av programmet.

000 - 023 bestämmer och trycker giv och
112 - 162 riskzon

024 - 084 ger utskrift av händerna
085 - 104 initiering
163 - 200 giv
201 - 257 sortering
258 - 388 gör i ordning händerna för utskrift
389 - 393 avslutning

Registerinnehåll:

64 textkod (NS),(OV) 3136.3242
65 " (ALLA) 1327271300
66 0,1408418
67 0,80140841
68 textkod (N),(OSV) 31.323642
69 " (9TJQKA)x1EE-32, får framställas med summering och multiplikation (eller enligt red. anm. som program), listas som 2.1372534-21.

Som synes har vi behållit de engelska beteckningarna för korten, dvs A(ce), K(ing), Q(ueen) och J(ack). Den som vill ha något annat (vi kunde inte hitta något bättre) kan ju lätt ändra det med hjälp av ovanstående förklaringar. - Registren 66 och 67 reglerar vilka som är i riskzonen (steg 138 - 162) och där står 0 för OV i riskzonen, 1 för NS, 4 för alla och 8 för ingen. Register 68 bestämmer given.

Körinstruktion:

1. Läs kortsida 1 och 2.
2. Lägg ett slumptalsfrö mellan 0 och 1 i t-registret.
3. Ange önskat antal givar.
4. Tryck A.

Programbitens kommentar till inmatning av programmet:

Vi har valt att ge program och registerinnehåll i minnena 64 - 69 som en sammanhängande listning i standarduppdelning. Både i programmet (särskilt vid dsz-funktionerna) och i stegen 400-447 som ju motsvarar minnena kommer att fordras en hel del syntetisk programmering. En kod som inte kan programmeras direkt kan ju alltid kodas med STO nn BST BST Del SST. Se också upp med en kod som "27 INV" dvs "2nd INV"!

.5072049537

1. N/-

AQT74
QJ87
K653
-

83 K965
- K6532
AT9 87
AKQJ8754 . 62

J2
AT94
QJ42
T93

J7
7
AJ82
JT8632

AKT62 953
A4 QJT98532
Q3 T7
AKQ4 -

Q84
K6
K9654
975

2. O/NS

3. S/OV

K982
Q743
QJT5
2

QT J7
J8 AK52
A42 K63
QJT863 AK54

A6543
T96
987
97

J32
T8632
QJ4
97

KQ9 854
AJ754 9
8 T9752
8642 AQT3

AT76
KQ
AK63
KJ5

4. V/ALLA

5. N/NS

K95
Q93
AK7
AKT9

AJ743 T862
7654 J82
85 T
Q3 J8742

Q
AKT
QJ96432
65

87
J532
T8
KQJ73

96532 KQ
T8 K9764
KJ973 642
T A62

AJT4
AQ
AQ5
9854

6. O/OV

000	22 INV	060	92 BTN	120	01 1	180	35 I-X	240	97 DSZ	300	77 77	360	97 DSZ	420	01 1
001	59 INT	061	76 LBL	121	00 0	181	22 INV	241	58 58	301	22 INV	361	62 62	421	18 C'
002	61 GTD	062	19 D'	122	00 0	182	59 INT	242	02 02	302	59 INT	362	02 02	422	64 DP+
003	10 E'	063	32 X:T	123	49 PRD	183	82 HIR	243	27 27	303	65 ×	363	82 82	423	40 40
004	43 RCL	064	63 EX+	124	68 68	184	04 04	244	73 RC+	304	01 1	364	17 B'	424	14 D
005	65 ×	065	51 51	125	85 +	185	65 ×	245	55 55	305	00 0	365	82 HIR	425	00 0
006	61 GTD	066	97 DSZ	126	52 EE	186	43 RCL	246	63 EX+	306	32 X:T	366	12 12	426	00 0
007	10 E'	067	51 51	127	08 8	187	57 57	247	56 56	307	01 1	367	29 CP	427	00 0
008	02 2	068	51 BST	128	94 +/-	188	95 =	248	72 ST+	308	03 3	368	67 EQ	428	40 IND
009	76 LBL	069	92 BTN	129	44 SUM	189	42 STO	249	55 55	309	85 +	369	03 03	429	01 1
010	10 E'	070	69 DP	130	68 68	190	58 58	250	97 DSZ	310	03 3	370	72 72	430	18 C'
011	52 EE	071	03 03	131	06 6	191	73 RC+	251	54 54	311	95 =	371	18 C'	431	84 DP+
012	04 4	072	32 X:T	132	03 3	192	57 57	252	02 02	312	52 EE	372	97 DSZ	432	90 90
013	22 INV	073	69 DP	133	95 =	193	63 EX+	253	10 10	313	22 INV	373	03 03	433	00 0
014	52 EE	074	04 04	134	22 INV	194	58 58	254	97 DSZ	314	77 GE	374	03 03	434	00 0
015	59 INT	075	19 D'	135	52 EE	195	72 ST+	255	57 57	315	03 03	375	77 77	435	00 0
016	03 03	076	19 D'	136	69 DP	196	57 57	256	02 02	316	30 30	376	98 ADV	436	13 C
017	69 DP	077	69 DP	137	03 03	197	97 DSZ	257	06 06	317	22 INV	377	05 5	437	27 INV
018	04 04	078	01 01	138	43 RCL	198	57 57	258	98 ADV	318	28 LOG	378	09 9	438	27 INV
019	69 DP	079	32 X:T	139	66 66	199	01 01	259	86 STF	319	52 EE	379	42 STO	439	13 C
020	05 05	080	69 DP	140	48 EXC	200	77 77	260	40 IND	320	33 X2	380	51 51	440	30 TRN
021	22 INV	081	02 02	141	67 67	201	04 4	261	61 61	321	65 ×	381	22 INV	441	00 0
022	59 FIX	082	69 DP	142	52 EE	202	42 STO	262	03 3	322	43 RCL	382	86 STF	442	00 0
023	92 BTN	083	05 05	143	01 1	203	57 57	263	82 HIR	323	69 69	383	40 IND	443	00 0
024	76 LBL	084	92 BTN	144	75 -	204	42 STO	264	02 02	324	95 =	384	61 61	444	42 STO
025	17 B'	085	76 LBL	145	22 INV	205	61 61	265	01 1	325	22 INV	385	97 DSZ	445	32 32
026	82 HIR	086	11 A	146	59 INT	206	01 1	266	03 3	326	59 INT	386	61 61	446	36 PGM
027	16 16	087	82 HIR	147	42 STO	207	02 2	267	42 STO	327	52 EE	387	02 02	447	31 31
028	32 X:T	088	03 03	148	66 66	208	42 STO	268	62 62	328	94 +/-	388	59 59		
029	82 HIR	089	07 7	149	95 =	209	54 54	269	65 ×	329	97 DSZ	389	97 DSZ		
030	17 17	090	69 DP	150	42 STO	210	43 RCL	270	43 RCL	330	65 ×	390	60 60		
031	22 INV	091	17 17	151	00 00	211	57 57	271	61 61	331	82 HIR	391	01 01		
032	67 EQ	092	82 HIR	152	52 EE	212	65 ×	272	75 -	332	18 18	392	05 05		
033	00 00	093	13 13	153	08 8	213	01 1	273	01 1	333	29 CP	393	91 R/S		
034	03 39	094	98 ADV	154	94 +/-	214	03 3	274	95 =	334	77 GE	394	00 0		
035	76 LBL	095	60 DEG	155	44 SUM	215	95 =	275	42 STO	335	03 03	395	00 0		
036	18 C'	096	42 STO	156	66 66	216	42 STO	276	63 63	336	42 47	396	00 0		
037	02 2	097	60 60	157	25 CLR	217	56 56	277	69 DP	337	82 HIR	397	00 0		
038	00 0	098	32 X:T	158	43 RCL	218	33 X2	278	00 00	338	16 16	398	00 0		
039	77 GE	099	82 HIR	159	64 64	219	52 EE	279	08 8	339	82 HIR	399	00 0		
040	00 00	100	04 04	160	71 SBR	220	43 RCL	280	82 HIR	340	07 07	400	14 D		
041	43 RCL	101	99 PRT	161	40 IND	221	54 54	281	08 08	341	00 0	401	02 2		
042	32 X:T	102	00 0	162	00 00	222	42 STO	282	82 HIR	342	82 HIR	402	13 C		
043	87 IFF	103	82 HIR	163	05 5	223	58 58	283	12 12	343	06 06	403	26 2ND		
044	02 02	104	03 03	164	01 1	224	01 1	284	32 X:T	344	08 8	404	34 FIX		
045	00 00	105	01 01	165	42 STO	225	44 SUM	285	73 RC+	345	82 HIR	405	25 CLR		
046	70 70	106	01 1	166	00 00	226	58 58	286	63 63	346	08 08	406	37 F-R		
047	87 IFF	107	33 33	167	42 STO	227	97 DSZ	287	55 ÷	347	22 INV	407	12 B		
048	03 03	108	82 HIR	168	57 57	228	56 56	288	01 1	348	28 LOG	408	10 E'		
049	00 00	109	13 13	169	43 RCL	229	51 BST	289	03 3	349	95 =	409	00 0		
050	62 62	110	98 ADV	170	00 00	230	73 RC+	290	95 =	350	42 STO	410	00 0		
051	69 DP	111	99 PRT	171	72 ST+	231	56 56	291	77 GE	351	82 HIR	411	00 0		
052	00 00	112	43 RCL	172	00 00	232	77 GE	292	03 03	352	36 36	412	32 X:T		
053	69 DP	113	68 68	173	97 DSZ	233	02 02	293	01 01	353	25 CLR	413	31 I-R		
054	02 02	114	69 DP	174	00 00	234	40 40	294	17 B'	354	02 2	414	42 STO		
055	32 X:T	115	00 00	175	01 01	235	32 X:T	295	01 1	355	82 HIR	415	36 36		
056	69 DP	116	65 ×	176	69 69	236	43 RCL	296	82 HIR	356	58 58	416	14 D		
057	03 03	117	22 INV	177	82 HIR	237	56 56	297	52 52	357	97 DSZ	417	00 0		
058	69 DP	118	44 SUM	178	14 14	238	42 STO	298	61 GTD	358	63 63	418	00 0		
059	05 05	119	68 68	179	30 TRN	239	55 55	299	02 02	359	51 BST	419	00 0		

En TI 59 användares

Prov med HP-41C

och en HP-41C-användares kommentar

PROV: HEWLETT-PACKARD HP-41C

Föreningens huvudsakliga inriktning till trots har det då och då kommit önskemål om stoff om HP. Eftersom de flesta medlemmar säkert har Texas börjar vi med lite information om hur det är att använda HP-41C när man är van vid TI-59.

För att undvika allt eventuellt gruff om partiskhet på ena eller andra hållet vill jag påpeka från början att detta prov är subjektivt i hög grad. Men det är också meningen! Denna artikel ska nämligen försöka svara på frågan: "Om jag har en TI-58/59, hur är det då att börja med HP-41C?" Detta kan ju även vara intressant för ägare av mindre apparater som tänkt byta upp sig. Den som istället vill läsa om "bara" 41C hänvisas till t.ex. Radio & Television 3-1980.

BEHÄKNINGSLÄGE

Den största skillnaden vid vanliga beräkningar är givetvis den omvända polska notationen (RPN). För en inbiten Texas-brukare är det bakvänt i början, men man vänjer sig ganska snabbt. Personligen anser jag att AOS är bäst vid manuell räkning, speciellt med komplicerade uttryck, medan däremot RPN - rätt använd - kan spara in utrymme vid programmering (det extra tänkandet som jag anser att RPN kräver spelar inte någon roll då, eftersom man ändå får tänka en hel del). Att HP har hela fyra register i stacken är inget att hänga upp sig på - motsvarande siffra för TI är ju nio, (58/59). Last X-registret som ingår i RPN, men inte i stacken, kan dock vara bra att ha ibland, eftersom vinkeln sparas i detta vid beräkning av sinus etc.

I övrigt har HP-41C de vanliga matematiska funktionerna (logaritmer, trigonometri, P/R mm) på tangentbordet. Indikeringsformat kan också bestämmas direkt. Det finns dock inte möjlighet till flytande decimalredovisning, utan man får alltid ha fast avrundning. Antingen får man dras med extra nollor eller också vet man inte om man har alla visningsbara decimaler. Andra funktioner såsom omvandling mellan grader och radianer, modulo, absolutbelopp etc. kan man åstadkomma genom att mata in dessa bokstav för bokstav via de alfanumeriska funktionerna på tangentbordet. Varje tangent har nämligen två vanliga och två alfanumeriska funktioner.

Om man använder någon av dessa dolda funktioner ofta kan den få en egen tangent. Tangentbordet kan på detta sätt varieras efter eget behag med upp till sex funktioner per tangent, inkl, de förprogrammerade. Det finns alltså tre lägen som tangentbordet kan vara i: Standard, användardefinierat eller alfanumeriskt. Genom att använda 2nd-tangenten får man då totalt sex funktioner per knapp. För att detta inte ska bli alltför rörigt finns de alfanumeriska andrafunktionerna på en skylt på undersidan av räknaren. De egna man definierar kan anteckna på en mall liknande den Key Code Overlay som finns till TI 58/59.

PROGRAMMERING

Nåväl, den största frågan för de flesta är väl hur den är att programmera. När man trycker på PRGM (LRN) lägger man direkt märke till att 41C anger hur många outnyttjade register (7 bytes) det finns tillgängligt. En annan bra sak är att ett program kan visas i klartext på skärmen, inte bara som sifferkoder. Själva inmatningen av program är lite annorlunda, då HP visar den nyss inmatade raden i stället för nästa tomma. Vidare skjuter den eventuella existerande rader framför sig, som en automatisk "insert"-funktion. För att undvika olägenheten att inte kunna skriva över långa ändrade programavsnitt finns det en "delete"-instruktion som kan ta bort upp till 1999 rader åt gången. Sådan radering minskar dock inte programmets minnesbehov, utan inför "osynliga" tomstruktioner. Detta gäller även vid radering av enstaka rader. För att sedan minska ner programmet finns en packningsinstruktion.

Själva programminnet disponerar man på ett lite annorlunda sätt, eftersom flera olika, helt självständiga program kan finnas lagrade samtidigt. Dessa hålls åtskilda av en END-markering. Inom varje program kan man flytta programpekaren på vanligt sätt, men när man kommer till END börjar den om från början i samma program igen. Trots detta kan ett program anropa subrutiner i ett annat, men mer om det senare. En bra detalj är att räknaren själv kan sätta ut END efter ett program, packa hela minnet och göra klart för inmatning av ett nytt med bara ett kommando.

Dataminnena har möjligheter till de sedvanliga beräkningarna in i minnet. Vid återkallning av något minne lyfts stacken automatiskt, varför man inte behöver göra några speciella åtgärder för att behålla det som fanns på skärmen innan. 41C har ingen möjlighet motsvarande Op 20-39 på TI 58/59, utan man får ha en etta i display och använda minnesaritmetiken.

Minnesuppdelningen är mera flexibel än hos TI, då kommandot SIZE kan ges vilket värde som helst, inte bara jämna tiotal. Vid lagring av många siffror finns det dock inte möjlighet att utnyttja skyddssiffrorna som hos TI 58/59.

När det gäller märkning av programmen saknas det minsann inte kapacitet. Genom att man kan konstruera labels som man vill, med två till sju tecken, är antalet programrader en droppe i havet jämfört med det totala antalet labels. Vidare finns 15 st lokala labels kallade A-F aamt a-e. Om dessa finns med i det program stegräknaren för tillfället befinner sig är definieras motsvarande alfanumeriska tangenter som användardefinierad label. Detta är alltså en direkt motsvarighet till A-E' på Texas. Man kan även använda numeriska etiketter, 00-99, lokalt inom ett program.

Genom att labelsökningen på HP börjar vid anropet, kan man ofta (om man tänker sig för) använda samma label flera gånger i ett program. Labels med låga nummer tar mindre plats i minnet.

Adressering direkt till programsteg är dock inte möjligt, utom vid redigering. För att ändå få en godtagbar snabbhet har konstruktörerna fiffigt nog låtit 41C komma ihåg på vilket steg varje label finns, bara den har anropats en gång under körningen. Sedan går ett hopp mycket fortare. Den verkar visserligen glömma detta så snart körningen avbryts, men om man gör en loop 100 gånger går ändå de 99 snabbare.

Vill man ha tillgång till någon label direkt från en tangent kan man definiera vilken alfanumerisk label som helst till denna tangent.

All adressering, t.ex. GTO och XEQ (SBR) kan använda vilken typ av label som helst. Om man vill köra en subrutin i ett annat självständigt program måste man dock ha en alfanumerisk adress, eftersom de andra är lokala inom programmet och inte i hela minnet.

Indirekt adressering kan ske via valfritt register, även i stacken eller Last X.

Vid hämtning av värde i tabeller t.ex. kan man alltså låta den beräknade adressen stanna i display och kommendera RCL IND ST X. Eftersom man med ASTO kan lagra strängar om max sex tecken i vilket minne som helst, kan även alfanumeriska labels anropas indirekt. Villkorliga hopp som beror på värdet av X-registret (display) finns det hela tio stycken. Hälften av dessa kan dock enbart testa mot noll, vilket i och för sig är ganska vanligt. De övriga använder Y-registret som testvärde. Just detta faktum att det inte finns något självstständigt testregister - motsvarande t-registret på TI - anser jag vara en beklaglig miss på HP. Så fort man utför en beräkning åker talen i stacken upp och ner, och jämförelsevärdet kan lätt komma på avvägar eller tappas bort helt!

Av HP:s broschyr kan man få uppfattningen att man har hela 56 flaggor att manipulera med. Detta är en sanning med modifikation, eftersom dessa 56 inkluderar alla systemflaggor som 41C behöver. I själva verket finns det 10 flaggor som man huserar över helt själv. 26 av de andra flaggorna går inte alls att påverka, bara att testa. Bland dessa finns t.ex. flaggor som håller reda på om man är i beräkningsläge eller programmerar, om man har tryckt på 2nd etc. Bland de andra 20, som alltså kan påverkas, finns information om valt vinkelmått, om användardefinierat tangentbord är tillkopplat mm. Här finns vissa intressanta möjligheter. Genom att ha flagga 11 hissad när HP 41C stängs av, kommer den att börja köra det program som programpekaren pekade på när den stängdes av, så fort den sätts på igen. Om man programmerar åt någon annan kan denne alltså få en helt automatisk start av programmet. Ett bra påfund nu när räknarna är så avancerade att många köper programvara konstruerad av specialister. Andra användbara flaggor är de som reagerar på inmatning från tangentbordet samt de som möjliggör olika åtgärder i händelse av felindikering.

När det gäller loopar ger 41C möjlighet att välja på upp- eller nedräkning av varvräknarregistret. Den stora skillnaden är att HP41C kan ha valfria start-,slut- och stegvärden. De olika värdena matas in som ett tal där olika positioner ger olika information, varför värdena siffermässigt måste ligga inom vissa gränser.

För att man ska kunna hålla reda på alla lagrade program och funktioner finns tre olika kataloger. Två visar förprogrammerade funktioner i räknaren resp. tillbehör. Den tredje visar samtliga alfanumeriska labels i minnet, samt om dessa ingår i ett stort program eller är självständiga små program. All katalogvisning kan ske snabbt, långsamt (med en tangent nedtryckt) eller steg för steg med SST resp. BST. När man går igenom programminnet följer programpekaren med, så att man lätt ska kunna placera den i något program.

Körning av program i förprogrammerade moduler liknar den "vanliga" programkörningen. Eftersom HP 41C inte är utrustad med två skilda programpekare som 58/59 kan den dock inte komma ihåg var man lämnade ett modulprogram, utan anrop måste ske på nytt varje gång. På TI kan man ju börja köra ett modulprogram från stoppunkten igen, även om man har hörjat i det ordinarie minnet under tiden. En nackdel här är att man inte kan anropa ett visst programsteg som en subrutin från ett annat program såvida det inte finns en passande alfanumerisk label just där. Egna program kan ju förses med sådana, men inte modulprogram. TI däremot kan hoppa in var som helst i ett modulprogram.

Detta var de stora funktionerna vad det gäller programmering samt några detaljer. Men det finns lite mer som är värt att ta upp.

Statistiken t.ex. har en möjlighet att fritt placera de sex minnen som behövs var som helst i minnet. De måste visserligen ligga efter varandra, men den valfria placeringen kan ändå underlätta ibland. HP 41C saknar dock den kraftfulla instruktionsrepertoar som finns hos 58/59, då det inte finns någon linjär regressinn. 41C är endast kapabel att beräkna medelvärde och standardavvikelse. Datainsamlingen fungerar dock exakt likadant som hos TI 59, varför det är ganska lätt att göra ett program med dessa funktioner. Att det tar lite minne kommer man förstås inte ifrån.

Räknaren är även utrustad med en liten tongenerator, vilken är kapabel att åstadkomma ett kort tut med 10 olika frekvenser (TONE 0-9) samt en liten truddelutt, kallad BEEP.

Avsikten med denna ska, enligt Hewlett-Packard, vara att t.ex. tala om när ett program har räknat färdigt. Nog så riktigt tänkt, men man får sannerligen inte vara lomhörd för att höra HP 41c signalera. Ljudstyrkan överträffas av vilket tutande armbandsur som helst. Troligtvis är detta en följd av ambitionen att få ner strömförbrukningen. Tonerna påstås vara energikrävande ändå.

Den stora skillnaden, eller den som märks först i alla fall, är naturligtvis den alfanumeriska kapaciteten. Omedelbart uppstår då frågan: Är den praktiskt användbar?

Svaret får bli ett jodå, men kanske inte riktigt så som HP vill göra gällande i sin broschyr, där det talas om en "funktionell dialog även i ord". 41C kan visserligen jämföra två inmatade strängar och se om de är lika eller inte, men denna jämförelse måste ske sex tecken i taget. Lagring av text i register göres också med sex tecken i varje. Man kan visserligen svara "JA" eller "NEJ" på frågor som räknaren ställer, men att kalla detta en funktionell dialog är allt att ta i. Däremot är den alfanumeriska kapaciteten alldeles utmärkt när det gäller korta instruktioner, ställa frågor samt märka svar vid programkörning. Att funktionen är nödvändig för att komma åt de olika "extra" funktionerna är en helt annan sak. Detta kunde åstadkommits med sifferkode - se på TI 58/59 (Op nn) - även om dessa givetvis är svårare att komma ihåg.

TILLBEHÖR

De moduler som används för att expandera programminnet ser likadana ut som de förprogrammerade modulerna. Nu finns dock även en expansionsmodul som ger maximalt minne med en gång, utan att lägga beslag på mer än en expansionskontakt. Denna är dock - naturligt nog - betydligt större. För den som vill ha mycket minne finns numera även möjligheten att anskaffa HP 41CV, vilken redan från "födseln" har fullt utbyggt minne. Det ges dock INTE möjlighet att utvidga denna ännu mer, antagligen pga bristande adresseringsmöjlighet. Maximalt arbetsminne blir alltså 319 register (2233 bytes). Det finns egentligen ett register till, men det används av räknaren.

Till 41C finns en intressant liten skrivare, som även går att använda portabelt med batteridrift. Skrivaren är lätt att använda i och med att de flesta (och vanligaste) tecknen finns tillgängliga på tangentbordet och inte behöver matas in som

sifferkoder. Två flaggor kan påverkas för att göra alla versaler som inmatas till gemena bokstäver vid utskriften. Man kan också skriva samtliga tecken med dubbel bredd. Då får givetvis bara 12 plats på varje rad mot normala 24.

Om man vill använda något tecken som inte finns på tangentbordet, såsom å,ä,ö,π,(,), α,β etc. får man mata in sifferkoder för ett tecken i taget direkt in i utskriftsbufferten. Hittar man inte heller här något passande tecken kan man konstruera ett eget genom att tala om vilka av punkterna i 7x7 matrisen som ska användas. Dessa tecken kan man sedan lagra i något minne i väntan på utskrift. Genom att man kan definiera hela 7x7 matrisen kan man få två tecken att helt gå ihop, om man vill ha ett stort.

Skrivaren har också möjlighet till plottning av funktioner som definieras i programminnet. Plottningen sker helt interaktivt med instruktioner i display. Man kan också köra utan instruktioner om man vill anropa rutinen från ett eget program. Det program som plottar är skrivet i samma språk som man programmerar med, och kan kopieras över till det vanliga minnet om man vill inspektera eller ändra något. Detta motsvarar alltså den möjlighet som ges med modulprogram på TI 58/59. Man kan också plotta enstaka värden som med PC-100c.

Skrivaren har möjlighet till reglering av intensiteten på "trycket", vilket kan behövas eftersom 24 tecken på en smal remsa lätt blir otydliga. Läsbarheten är ändå klart sämre än vad PC-100C presterar.

Med stor stil blir det något annat, men då går det åt dubbelt så lång tid för att skriva lika mycket.

Hur det är med arkivbeständigheten är inte lätt att säga, men en enkel jämförelse mellan HP:s och TI:s rekommenderade pappersorter - vilka efter någorlunda samtidig tryckning placerades i ett fönster med södersol - gav vid handen att Hewlett-Packards text bleknar snabbare samtidigt som pappret gulnar. Skrivaren skrev med maximal intensitet. Vetenskapligheten kan kanske ifrågasättas i detta test, men det ger en liten antydan.

Magnetkortsläsare finns också att ansluta. Denna arbetar med fyra olika sorters kort, även om de ser likadana ut. Man skiljer nämligen på program-, data-, status- och totalkort. Fördelen med detta är att man inte kan läsa ett datakort och få siffrorna tolkade som programsteg, vilka i regel blir helt vanvettiga. Även 59:an skyddar sig mot detta genom att vara noga med uppdelningen vid kortläsning.
Statuskort lagrar flaggornas läge, vinkelmått, indikeringsformat mm samt alla egna tangenttillordningar. Totalkort lagrar alltihop som finns i minnet, inklusive flaggor och liknande.

Liksom på TI-59 kan man skydda magnetkort mot redigering och kopiering. Det går även att skrivskydda magnetkort genom att klippa av ett hörn. Skulle man sedan ångra sig kan man ställa flagga 14 och spela in på kortet ändå. Texas SR-52 hade också denna möjlighet, men där klistrade man på resp. pillade bort en liten tejpbit.
Om man har för avsikt att använda magnetkortsläsaren flitigt gör man klokt i att investera i laddare och ackumulatorer till sin HP 41C, eftersom dess energibehov är våldsamt (jämfört med räknarens CMOS-elektronik.

Den som tänker lagra mycket information får också bereda sig på att anskaffa en försvarlig kortbunta. En kortsida rymmer nämligen endast 16 register på vardera 7 bytes. Detta ska jämföras med TI 59, vilken lagrar 30 register om 8 bytes på en sida. Själv har jag 229 kort med program och data på till min 59:a. Det gör bortåt 450 för en 41C...

SAMMANFATTNING

Vilken ska man då välja, när vi nu har plöjt igenom de väsentliga funktionerna. För det första är det en fråga om AOS kontra RPN. Kan man absolut inte med det ena systemet blir valet enkelt.
I övrigt borde det vara uppenbart att HP 41C har ett klart övertag beträffande den alfanumeriska kapaciteten, speciellt i "fönstret" men även på papper. Möjligheten att ansluta valfria tillbehör är naturligtvis också bra. (Det finns för övrigt även en streckkodsläsare som tillbehör, men den har inte varit tillgänglig.)
När det gäller den rent numeriska kapaciteten har 41C däremot inte så mycket mer att erbjuda. Det finns visserligen modulo, decimal-oktalomvandling samt några andra

funktioner, men däremot inte linjär regression. Vidare är TI:s datorminne mer "värda" i vissa tillämpningar genom att man kan utnyttja skyddssiffrorna på ett enkelt sätt. När det gäller programkörning är HP långsammare, åtminstone i enklare slingor som är direkt jämförbara. En större instruktionsrepertoar kan naturligtvis möjliggöra bättre och snabbare lösningar.

Vissa matematiska instruktioner är snabbare hos 41C. Till saken hör att 41C-program ofta tappar i snabbhet pga de alfanumeriska meddelanden man lägger in i programmet. Då kan man förstås säga att strunta i dessa, så går det ju fortare. Jovisst, men vad finns det då för anledning att lägga pengar på en maskin med dessa funktioner? Nu kan man kanske tycka att HP 41C är en i de flesta avseenden överlägsen apparat. Vilket i så fall inte vore helt onaturligt, då den är betydligt nyare.

Här kommer dock en, åtminstone för privata kunder, viktig faktor med i bilden: Priset. Utan att gå in på detaljer (priserna ändras ändå titt och tätt) kan man se en klar skillnad på prisnivån, och man vill ha ett HP-system som inte saknar väsentligheter i motsvarande Texasutrustning. Skillnaden är i storleksordningen 3000/7000 kronor. Här avses då TI 59 med PC-100C kontra HP 41C med magnetkortsläsare, skrivare, en minnesmodul och en programmodul (t.ex. Math Pac).

TI 59 har ju magnetkort, standardmodul och dubbelt så stort minne som 41C från början. Ett "minsta möjliga"-system skiljer givetvis inte så mycket i pris, men då får ju HP köparen vara utan denna utrustning. (HP 41C är alltså fortfarande dyrare.)

Instruktionen till räknaren håller sedvanlig Hewlett-Packard standard - dvs god - och vinner ytterligare på att vara sparsam med de fullkomligt idiotiska exempel som förekommer i boken till HP 34C bl.a. Ändå finns det bara ett färdigt program som någon kan tänkas ha glädje av i 41C-boken, nämligen lösning av andragradsekvationer. Frågan kvarstår alltså: Vilken ska man välja? Det finns inget kategoriskt svar på den frågan. Dessutom kommer det säkert ett svar från Texas Instruments så fort de hinner. Se bara lite tillbaka: HP-65, Texas SR-52, HP-67, TI-59, HP-41C(V) och sen?? TI-88C/GT med fartrand???

Anders Persson, Örkelljunga

En HP-41C användares kommentar

När just med intresse läst Anders Perssons prov av HP-41C, och kommer här med några synpunkter:

Jag vet inte vad TI:s Op 20-39 gör för något, men av sammanhanget att döma gäller det upp- resp nedräkning av minnesregister. HP 41 har två instruktioner för detta, ISG och DSE, som kan användas för att räkna upp eller ner såväl minne 00-99 som stacken. Indirekt kan man komma åt alla minnen, 00-319. I samband med upp/nedräkningen görs en test, och beroende på resultatet fortsätter programmet med nästa steg, eller hoppar över nästa steg. Om man inte vill utnyttja testen, bör man lägga en "dummy"-instruktion efter, t.ex. en label man ej använder, X<>X, DSG eller något liknande. Tyvärr saknar HP 41 instruktionen NOP (No operation), åtminstone så länge man nöjer sig med att göra det som står i instruktionsboken. Men man kan - likasom med TI-räknare - gå bortom instruktionsboken.

Att man inte kan utnyttja HP-41:ans skyddssiffror beror på att HP-41 inte har några sådana. Aritmetiken görs internt med 11 (en del påstår 13) siffror, och resultatet avrundas till två till 10 siffror. Givetvis blir då alla 10 siffrorna då också riktiga.

Det finns inte bara 15 utan 115 lokala labels: 00-99 samt A-J och a-e. När man hoppar till en lokal label kan man inte hoppa över en END-instruktion, dvs man kan bara hoppa inom samma program. Detta medför att olika program kan använda samma lokala labels utan risk för förväxling, även om ett program använder ett annat program som subrutin.

När man kör ett HP-41-program kommer 41:an ihåg var varje label finns, bara den har letat efter den en gång (och hittat den). Denna information skrivs in i GTO- resp XEQ-instruktionerna, dvs om två GTO-instruktioner hoppar till samma label måste både GTO-instruktionerna leta efter labeln första gången. I motsats till vad Anders Persson påstår försvinner inte denna information när körningen stoppas. Den finns kvar även om man stänger av och sätter på maskinen, och spelar man in programmet på magnetkort kommer informationen även med där (den finns ju inne i GTO- resp XEQ-instruktionerna). En HP-41-ägare bör därför köra sitt program innan han spelar in det på kort - programmet går då med "full fart" även i fortsättningen när man läser det från kortet. Men om man ändrar i sitt program, tar bort eller lägger till någon eller några rader, då försvinner informationen om var alla labels finns, och samma sak händer om man PACKar programminnet. Det är egentligen inte så konstigt - allt flyttas ju om i minnet vid dessa tillfällen, så att all information om var labels finns blir ogiltig. Lenske jag här borde göra klart skillnaden mellan "programsteg" och "minnesadresser". I alla TI:s räknare, och i HP:s tidigare modeller, har varje programsteg innehållit bara en byte, och varje programsteg har fysiskt haft samma position i minnet, det har m.a.o. haft samma minnesadress.

I HP-41 kan "programsteger" vara av olika längd, allt mellan 1 och 16 bytes. Men varje programsteg i HP-41C är alltid en fullständig instruktion. Jämför följande HP-41 och TI-59-program, som båda adderar talen 1.234 och 5.678 samt stoppar dem i minne 89:

TI-59:	HP-41C:
01: 1	01: 1.234
02: .	02: ENTER
03: 2	03: 5.678
04: 3	04: +
05: 4	05: STO 89
06: +	
07: 5	
08: 6	01: 1.234
10: 7	02: 5.678
11: 8	03: +
12: =	04: STO 89
13: STO	
14: 89	

HP-41-programmet finns i två versioner. Vilket av dessa 3 program är längst? Svaret är att alla är lika långa, 4 bytes räknat. HP-41C-program nr 2 kan inte matas in som det står i listan, man måste göra något mellan de två siffer-inmatningarna (t.ex. stoppa in ett ENTER som man sedan tar bort). HP-41C-program nr 2 ser egentligen ut så här:

```
01:  1.234
    (NULL)
02:  5.678
03:  +
04:  STO 89
```

NULL-instruktionen är "osynlig", men tar ändå upp plats i minnet. PACK-instruktionen är till för att just ta bort onödiga NULLs - men i detta fall behövs NULL för att skilja de två talen åt, och PACK låter denna NULL vara.
Antag nu att vi tar bort programrad 2 och 3. Programmet ser då ut så här:

```
01:  1.234
    (NULL)
    (NULL)
    (NULL)
    (NULL)
    (NULL)
    (NULL)
02:  STO 89
```

Och efter en PACK blir det:

```
01:  1.234
02:  STO 89
```

Om man glömmer att PACKa, och matar in mer och mer i programminnet tills det är fullt, vad händer då? Jo HP-41:an packar automatiskt, och uppmanar en sedan att TRY AGAIN.

Vad är nu anledningen till allt detta trassel med PACK och NULLs? För att förstå det måste man titta på tidigare HP-räknare. De har i regel fungerat så att när man matat in ett programsteg har man "knuffat ner" alla efterföljande programsteg ett steg. Man har också bara kunnat hoppa till labels, och för varje hopp har räknaren då fått leta igenom hela programminnet, tills den hittat labeln. Detta är ur användarens synpunkt bekvämt - man kan fritt stoppa in och ta bort rader utan att behöva bekymra sig om att ändra hoppadresser. På HP-65 med 100 programrader var det inget problem - programminnet var så litet att det gick snabbt att söka igenom. På HP-67 med 224 programrader var det en fullt märkbar fördröjning när man matade in en programrad i början och då "knuffade ner" alla följande rader ett steg. Och att leta efter en label kunde ta en fullt märkbar bråkdel av en sekund. Men på HP-41C skulle motsvarande fördröjningar ha blivit helt oacceptabla. I ett program på mellan 1000 och 2000 programrader skulle det ta upp till ca 5 sekunder att "knuffa ner efterföljande rader ett steg", och lika lång tid att leta efter en label. Detta är givetvis helt oacceptabelt, och därför gjordes systemet mer komplicerat så att det skulle kunna bli snabbare. Nu knuffas programraderna ner 7 bytes i taget, och ledigt utrymme fylls med NULL. P.g.a. HP-41:ans konstruktion går det mycket snabbare att flytta ner programrader 7 bytes än att flytta det 1 byte, men det kan i värsta fall fortfarande ta ca 1 sekund. När man tar bort programrader sker ingen uppflyttning, de borttagna instruktionerna skrivs över med NULL. Efter tillräckligt antal ändringar ser programmet ut som en schweizerost, med NULLs lite här och där. Dessa NULLs syns inte när man listar programmet, men märks ändå på så sätt att programmet tar onödigt stor plats. Så när man är klar (eller tror man är klar) gör man PACK för att få bort alla NULLs (för stora program kan PACK ta 30s eller mer). Sen bör man till sist köra det igen så att alla GTO och XEQ hittar sina labels, innan man sparar det på magnetkort.

Man kan påverka alla 56 flaggorna hos HP-41. Enligt instruktionsboken kan 26 av flaggorna bara testas, men med speciella "knep" kan man hissa och hala även dessa. Resultatet är nog mer roligt än användbart, man kan t.ex. när man vill sätta på "low battery"-indikatorn, man kan få program att utöka sig själv, man kan se ganska lustiga kataloger etc.

Ytterligare en praktisk sak med flagga 11: Om man spelar in ett program på magnetkort med flagga 11 hissad, så startar programmet automatiskt när man läser in kortet igen.

Att man inte kan hoppa in var som helst i ett modulprogram är ganska besvärande med HP-41, i synnerhet som många modulprogram inte är skrivna så att de kan användas som subrutiner, fast man skulle vilja det (visst kan de rent tekniskt köras som subrutiner, men de vill att man matar in siffror från tangentbordet när man själv kanske skulle vilja att de tog siffrorna direkt från minnet istället). Det finns två lösningar på detta: Använd COPY för att kopiera över det i det vanliga RAM-minnet, och ändra det där så det passar. Men då tar det förstås plats i RAM-minnet. Eller också kan man med speciella "knep" som inte står i instruktionsboken faktiskt hoppa in på vilken programrad som helst. Det är en smula besvärligt - men det går.

Linjär regression finns visserligen inte som inbyggd funktion i HP-41, men det finns i en statistikmodul. Och i HP-41 kan man plugga in upp till 4 programmoduler (men då har man förstås inte plats att ansluta vare sig kortläsare eller skrivare. Den som tycker om att pula själv kan dock göra expansionsportar. Men kan då - samtidigt - ansluta 4 minnesmoduler, 4 programmoduler, kortläsare, skrivare och ljuspenna till sin HP-41:a).

Enligt min mening är den alfanumeriska kapaciteten inte användbar att föra en "dialog" med, så där håller jag helt med Anders Persson. När jag programmerar min egen HP-41:a brukar jag använda den alfanumeriska kapaciteten till att skriva ut korta meddelanden i samband med sifferresultatet, så att man meddetsamma ser vilken siffra man har på displayen eller printern. Men om man ska svara på en fråga eller välja olika alternativ, gör man det lättare för sig om man svarar med siffror istället för bokstäver. Och om man skriver program som innehåller mycket text, märker man att minnet snabbt äts upp av texten - varje bokstav tar ju en byte. Om man huvudsakligen är intresserad av konversation med ord med sin räknedosa, bör man nog inte satsa på HP-41, utan istället se sig omkring bland de BASICdosor (eller "fickdatorer") som nu börjat dyka upp på marknaden.

Printerpapperet hos HP-41:ans skrivare är ett sorgligt kapitel. Det finns dock "piratpapper" som ger svarta bokstäver istället för blå. Bokstäver och siffror är klart mindre på HP-41:s skrivare än på TI:s skrivare PC-100 - men å andra sidan får det då plats fler tecken per rad (24 st). Man kan även skriva ut flera tal bredvid varandra på samma rad på skrivaren, med korta textsnuttar emellan.

Slutligen ställer jag mig lite frågande till Anders Iersons påstående att HP-41 var långsammare än TI-59. Jag har nämligen inbillat mig att HP-41C är det f.t. snabbaste räknedosan som existerar (se Programbiten 81-1 sid 3 högra spalten). Att ett program blir långsammare när man lägger in extra funktioner, som t.ex. alfanumeriska meddelanden, är väl ganska självklart. Och om långsamheten besvärar kan man göra meddelandena så korta som möjligt, t.ex. förkortningar på 2 eller 3 bokstäver. Själv använder jag sparsamt med alfanumeriska meddelanden när jag skriver mina egna HP-41-program. Anledningen till att jag själv valde HP-41C var inte främst den alfanumeriska kapaciteten, utan minneskapaciteten! Fullt utbyggd har HP-41C 319 minnesregister (+ 4 bytes) som man fritt kan disponera som program- eller dataminne. HP-41C har m.a.o. mer än 3 ggr så mycket minne som TI-59! Lägg därtill att HP med sitt RPN-system gör att programmen blir kortare! Samt att HP-41C har flera typer av "korta" instruktioner - STO 89, RCL 89 tar t.ex. båda 2 bytes i minnet, medan STO 11, RCL 11 bara tar 1 byte. Samt att HP-41C är snabbare - kanske inte i en liten slinga alla gånger, men väl i ett lite större program som uträttar något nyttigt.

Men den som vill ha en HP-41C får också vara beredd att betala för den. Ett HP-41C-system kostar mer än dubbelt så mycket som ett TI-59-system. Så vad man ska välja beror nog väldigt mycket på vad man vill göra med räknaren. För en del kan valet vara nog så enkelt - de kanske inte har råd med något annat än TI-59. Andra kanske visserligen har råd med HP-41C, men kanske ändå inte behöver 41:ans räknekapacitet - en TI-59 kan då vara ett bra val. Och den som tycker lilla om RPN väljer förstås inte alls HP, vare sig han har råd med det eller inte. För min egen personliga del var valet enkelt - jag behövde helt enkelt den räkne- och minneskapacitet HP-41 erbjöd. Och jag var sen tidigare van vid HP-räknare (den saken spelar nog en genteka stor roll för de flesta).

Den frågan som alla slutligen ställer sig är: "Vad kommer i framtiden?" För ca 2 år sedan, då HP-41C var ganska ny, väntade jag mig att TI inom ca ett år skulle släppa ut en räknare med kanske 1000 minnesregister som var klart snabbare än HP-41C. Som alla vet har det inte inträffat, och jag är nu mer och mer böjd att tro att nästa räknedosa inte kommer att vara som HP-41C eller TI-59, m.a.o. att dessa två räknedosor kommer att vara de sista i sitt slag. Nästa räknedosa av större format kommer antagligen att vara programmerbar i BASIC. Det finns redan en sådan att köpa. Ibland heter den Sharp, ibland TRS-80 Pocket Computer, men under skalet finns i bägge fallen samma maskin. Den är långsammare än t.ex. HP-67 (som i sin tur är långsammare än TI-59, som i sin tur är långsammare än HP-41), och den har mindre minne än TI-59. Den är programmerbar i BASIC. Den är ingenting för den som vill ha hög räknekapacitet (i så fall är både TI-59 och HP-41 bättre val), men om man tycker om BASIC, eller vill lära sig BASIC, eller inte har så höga krav på räknekapacitet, är den kanske ett bra val. Men den är bara en leksak jämfört med som troligen kommer. Det får finnas en annan BASIC-programmerbar dosa, av fabrikat Quasar/Panasonic, med en 6502 som CPU, och med minne utbyggbart till 32K (HP-41 har drygt 2K i fullt utbyggt skick), som är ungefär lika snabb som Apple och Pet - och samtidigt så liten att den kan stoppas i ficken. Det ryktas f.ö. att HP snart ska släppa en BASIC-programmerbar räknedosa. Om kanske 5 år, då dessa "fickdatorer" bör ha blivit vanliga, då kommer AOS slutligen att ha segrat över RPN.

Paul Schlyter
Nybrogatan 75 E
114 40 Stockholm

Den som har skaffat sig en HP-41C, och som vill veta mer om de "tricks" som inte står i bruksanvisningen, kan kontakta:

PPC - Norden
c/o Dan Bellander
Karlbergsv. 84
113 35 STOCKHOLM

Anders Perssons svar

Efter att ha tagit del av Paul Schlyters kommentar till mitt prov av HP-41C vill jag gärna själv förtydliga och utveckla några av mina egna påståenden.

Paul har påpekat att det finns några fel i min artikel, vilket tyvärr stämmer. Bl.a. har jag av misstag räkat påstå att den tyska storleken på en QUAD-minnesmodul skulle vara större än en vanlig modul. Det är den alltså inte.

För att anknyta till Pauls kommentar ställer jag mig lite undrande till det här med skyddssiffror. Om man trycker 3 1/x visas helt naturligt 0.333... En multiplikation med 3 ska ge resultatet 1, vilket den gör. Men för att detta ska kunna ske krävs minst en skyddssiffra utanför display. Annars skulle svaret bli 0.999... Denna borde man kunna komma åt genom att subtrahera 1/3 med 0.333... inmatat från tangentbordet. Så kan man göra på TI-59. Men på HP-41 går det inte! Försöker man blir resultatet 0. Kan man av detta dra slutsatsen att det finns skyddssiffror som bara används vid vissa operationer?

Beträffande lokala labels har jag med uppgiften att det finns 15 stycken avsett de användardefinierade tangenter som definieras av de lokala labels A-J samt a-e. Några rader längre ner har jag nämt de 100 olika numeriska labels, vilka också är lokala, som man kan utnyttja.

Att HP-41 kommer ihåg var varje label är placerad även efter ett avbrott i programkörningen trodde jag däremot inte. Jag tycke det gick sakta i början varje gång man körde ett program, men jag har kanske varit inne i minnet och editerat mellan gångerna.

Antalet programmoduler i en TI-59:a kan faktiskt också vara 4 på en gång. En förgreningskontakt har tillkännagivits i PB.

Jag vidhåller att HP-41C är långsammare när det gäller enkla "benchmarks". Det behöver ju inte vara något hinder för att den kan vara snabbare när det gäller nyttiga program. Dessa är dock svåra att jämföra, eftersom skillnaden då ofta beror minst lika mycket på programmeraren. Att P/R är snabbare på HP-41 än på TI-59 beror kanske på att den rutinen är skriven i samma språk som man programmerar med när det gäller TI-59. Om den är mikroprogrammerad på HP-41 är det inte konstigt om den är snabbare där. För övrigt tycker jag inte att det är underligt om räknare uppbyggda i CMOS-teknik är långsammare än de som använder mer konventionella MOS-kretsar. De brukar vara snabbare, till priser av högre strömförbrukning. De TI-58C som jag har jämfört min 59:a med har varit klart "segare".

Minneskapaciteten hos TI-59 har Paul Schlyter fått lite om bakfoten. Det är inte så lätt alltid när man inte har erfarenhet av maskinen ifråga, vilket jag är den förste att erkänna. Det är visserligen halt riktigt att TI-59 har 100 dataregister som mest, jämfört med 319 för HP-41CV. Men samtidigt har TI-59 20 register programminne, vilka tyvärr inte kan omvandlas till dataminne. (Vem blir den förste som kommer på hur man kringgår den regeln?) Totalt sett alltså 120 register á 8 bytes, vilket ger 960 bytes minne. Detta räkar vara lika med maximala antalet programsteg, eftersom varje steg omfattar 8 bitar. En byte avser här alltså oegentligt 8 bitar, trots att den ALU som används är en fyrabitars.

Dessa 960 bytes ska jämföras med de 2233+4 (jag antar att de sista fyra kommer från den fasta .END.-instruktionen) i ett maximalt HP-41-system. 2237/960=2,33 ungefär.

Internminnet i TI-59 är alltså något större än det i två vanliga expansionsmoduler till 41C.

Beträffande val av räknare köpte jag min TI-59 av samma anledning som Paul Schlyter skaffade sig HP-41C: Det var den mest avancerade som fanns på marknaden då. Närmaste konkurrenten var HP-67/97.
Den BASIC-programmerbara Sharp som finns tycker jag mest är en leksak, just på grund av den låga räknekapaciteten. Att övernuvudtaget ge en så liten maskin ett så utrymmeskrävande och ändå "svagt" språk som BASIC anser jag mindre lämpligt. Den andra maskin som han (Paul) nämner verkar avsevärt vettigare. När en dylik apparat med t.ex. Pascal som språk ser dagens ljus köper jag nog en sådan...

Någon tycker kanske att vi har hamnat långt vid sidan om programutveckling till TI-58 respektive TI-59, men jag hoppas att det ändå har varit intressant.

Anders Persson
Örkelljunga

Musse Pigg på omslaget

Är gjord med ett program ur TI PPC Notes V6N6/7P14 av Frederic de Mees. Vi avstår från att publicera det eftersom det möjligen var några tryckfel i registerlistningen. Dessutom undras om Frederics program verkligen är det effektivaste. Kanske en extra "Utmanings"-uppgift? Vi gjorde tyvärr omslaget i sista minuten och har därför inte hunnit pröva andra metoder för programmet. Men idéer har vi. Skriv och "utmana" oss! Själva skrivmetoden, dvs en (halv) åtta och ett (halvt) mellanslag (op 00 op 05) i Graphics Mode, anses dock given. Det är datalagringen problemet gäller.

Tippa med TI 59

av BERTIL WALLIN · Karlskoga

För att köra programmet, krävs visst förarbete, de indata som skall användas måste kodas. Detta för att ej behöva göra onödigt många "inputs".
Om vi antar att det finns tillgång till följande inf. om den spelomg. som är aktuell (de flesta tidningar):

1. information typ "10 tidningars tips".
2. lagens poängställning i tabell.
3. annan godtyckligt parameter från tidningarnas statistiska tabeller.
4. egen bedömning av lagens styrka. (i %).

Valfrihet finns att välja godtyckliga parametrar, om man beaktar följande:
enl. pkt. 4 skall %-tal inmatas.
enl. pkt.1 kommer division att ske med talet N. För 10 tidningars tips är N=10.

Ett konkret exempel: För match nr 1 antas följande gälla,
10 tidn. tips: 7 st 1:or, 2 st x och 1 st 2:a.
Egen bedömning: hemmavinst 65% bortavinst 15%.
Poängställning: hemmalag 32, bortalag 23.
Vinnarförmåga: hemmalagets vunnena hemmamatcher 7st, bortalagets vunnena bortamatcher 2st.

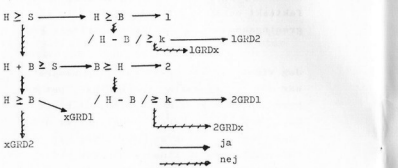
Kodningen för match 1 blir då enl. följande:
1....(0701.6515) och 2....(0702.3223), dvs. 2 st.
Koder skall inmatas för varje match! Mer generellt kan vi skriva: 1...(1122.EE_hEE_b),2...($VV_hVV_b.PP_hPP_b$).
Indexen anger hemma resp. bortalag. (h/b).
1122 avser 10 tidn. tips. P
EE_hEE_b avser egen bedömning (%)! Q
VV_hVV_b avser vinnarförmåga. R
PP_hPP_b avser poängställning. S
Om man vill ändra ing. parametrar möter det inget hinder för kod nr 2. Betr. kod nr 1 måste hänsyn tas till anm. enl. punkt 1 och 4 ovan. Ordningen får ej omkastas för kod nr 1 ,(EE_hEE_b,1122 är inte tillåtet). Indexen h och b får ej omkastas!

Något om matematiken

För enkelhets skull användes symbolerna P,Q,R och S för de ing. parametrarna. Relationerna för dessa indata är då för detta program enl. följande:

$W_1 = \frac{P+Q}{2}$ $W_2 = \sqrt{RS}$ $T = \frac{W_1 + W_2}{2}$

T representerar 2 variabler, och anger således styrkan på hemmalag respektive bortalag, (H resp. B).
Ett slumptal S genereras och testas mot H och B talen enl. följande strategi.



Konstanten k har värdet 0.1
Slumpgeneratorn är enl. följande:

$s = e^{10x}$frac.

Starttalet hämtas från register nr. 25, dvs. styrketalet för bortavinst i match nr.3.
Sannolikhetsrelationerna för de olika kombinationerna kan skrivas enl. följande:

P(1) = H·a
P(2) = B·(1- a)
P(x1) = (1 - H - B)·a
P(x2) = (1 - H - B)·(1 - a)
P(12) = H·b·(1 - a)
P(1x) = H·(1 - a)·(1 - b)
P(21) = B·a·b
P(2x) = B·a·(1 - b)

$a = \frac{1}{1 + B/H}$ $b = \frac{1}{1 + k/D}$ D = / H - B /

anm. P(12) = P(21) !
Dessa sannolikheter gäller för den ideella SLT - generatorn!

Körning av program.

	tryck	printout
1. initiera	E	TIPS, V NR
2. vecka nr. xy	R/S	N = ?, xy
3. antal tidn. tips	R/S	N

anm! N = antalet tidningar som lämnat sitt expert-tips, ex. enligt sid 1 har N = 10. N är alltså DIVISOR.

4. input kod 1 match 1 (1122.EE_hEE_b)
enl. ex. (0701.6515) D
input kod 2 match 1 ($VV_hVV_b.PP_hPP_B$)
enl. ex. (0702.3223) D
anm. efter att kod 1 inmatats för varje match visar display 0, när sedan kod 2 inmatats visas nästa matchnummer.
När kod 2 för match nr 13 inmatats startar beräkningen och utskrift sker.
5. om ny rad önskas (med annat starttal s)

tryck		
starttal s (0 - 1)	R/S	ny rad, s

Utskriften ger först en lista på T(H,B) för de 13 matcherna. DVS. sannolikheterna för hemma resp. borta vinst. Dessa värden är rent statistiska om man bortser från slumpen vid den egna bedömningen.
Sedan skrives en halvgarderad rad ut, enl. tidigare beskrivna strategi. Om man ej är nöjd med raden kan man antingen utnyttja pkt. 5 tills man får en godtagbar rad betr. ex.vis antal garderingar, eller så stryker man de "överflödiga" garderingarna med ledning av T(HB)-listan. Listan och tipsraden börjar med match 1.

Något om hård/mjukvaran.
Normal uppdelning
Printer
Block 1 och 2 pgm. Block 3 textregister.
Pgm.register 00 - 19, dataregister 20 - 45 och textregister 46 - 59.
FLG 0

anm. VV_h och VV_b får ej samtidigt vara 0! detta gäller även för P. Om så är fallet, sätt $VV_h = VV_b$ = godtyckligt tal (även för P).

Ett exempel.

Ur tipstabell tar vi 2 matcher. Om match 1 gäller Flen - Örebro och match 2 Eskilstuna - Sandviken fås följande.

DIVISION II NORRA

	hemma	borta	sammanlagt
Flen	4 0 1 8-- 6	3 1 1 8--5	10 7 1 3 17--11 15
Örebro	3 1 2 7--5	4 1 0 12--4	8 6 2 3 19--9 14
Eskilstuna	3 2 1 8--3	3 3 0 10--3	10 5 4 1 18--6 14
Sandviken	3 1 1 7--5	4 0 1 15--5	10 5 2 3 22--10 12

Match 1: VV_h =04, VV_b = 04, PP_h = 15, PP_b = 14
Match 2: VV_h =02, VV_b = 04, PP_h = 14, PP_b = 13
Från 10 tidningars tips fås följande:
Match 1: 5 st ettor, 2 st x och 3 st tvåor.
Match 2: 3 -"- 4 -"- 3 -"-
Egen bedömning:
Match 1: 60% hemmavinst, 30% bortavinst.
Match 2: 40% -"- 50% -"-
Kod match 1: kod 1 =(0503.6030), kod 2=(0404.1514).
Kod match 2: kod 1 =(0303.4050), kod 2=(0204.1413).
Sedan fyller vi på med simulerade värden för de övriga matcherna enl. nedan:

match	kod 1	kod 2
3	(0104.3050)	(0504.1317)
4	(0305.6030)	(0101.0309)
5	(0604.7515)	(0401.1510)
6	(0503.7020)	(0101.1107)
7	(0901.8010)	(0101.1513)
8	(0403.8010)	(0100.0806)
9	(0106.1080)	(0003.0714)
10	(0402.6010)	(0201.1107)
11	(0304.6030)	(0203.1515)
12	(0005.0099)	(0205.0101)
13	(0404.4060)	(0504.0101)

När dessa koder inmatats erhålles en halvgarderad rad enl. följande. Då den första raden (systemet) blev för stor (64 rader), söktes en ny. Med s = 0.789 erhölls då ett system om 8 rader, som bättre passar för en sliten plånbok.

Registerinnehåll i R90-R99 (t h)			
		330220.	90
		330203.	91
		330266.	92
		330220.	93
		330302.	94
		330366.	95
Fem exempel på		336602.	96
körningar (nedan)		336603.	97
		0.	98
		0.	99

001 15 E

0.56 H
0.13 B
0.1 K

.4544927536	P1-	
.0856002188	P12	
.0199070276	P1X	
.0244927536	P2-	
.0856002188	P21	
.0199070276	P2X	
.2515942029	PX1	
.0584057971	PX2	

0.13 H
0.56 B
0.1 K

.0244927536	P1-	.0000000133 P1-
.0856002188	P12	.0000882221 P12
.0199070276	P1X	.0000117645 P1X
.4544927536	P2-	.7499000133 P2-
.0856002188	P21	.0000882221 P21
.0199070276	P2X	.0000117645 P2X
.0584057971	PX1	.0000333156 PX1
.2515942029	PX2	.2498666844 PX2

0.0001 H osb H*D
0.75 B
0.1 K

0.5 H
0.5 B
0.1 K

0.25	P1-	0.75 P1-
0.	P12	0. P12
0.25	P1X	0. P1X
0.25	P2-	0. P2-
0.	P21	0. P21
0.25	P2X	0. P2X
0.	PX1	0.25 PX1
0.	PX2	0. PX2

000	76	LBL	040	94	+/-	080	01	01	120	02	02
001	15	E	041	42	STD	081	65	×	121	65	×
002	32	X↔T	042	04	04	082	43	RCL	122	43	RCL
003	25	CLR	043	43	RCL	083	04	04	123	07	07
004	42	STD	044	02	02	084	65	×	124	65	×
005	06	06	045	55	÷	085	43	RCL	125	43	RCL
006	09	9	046	43	RCL	086	08	08	126	05	05
007	00	0	047	01	01	087	95	=	127	95	=
008	42	STD	048	85	+	088	42	STD	128	71	SBR
009	00	00	049	01	1	089	10	10	129	01	01
010	32	X↔T	050	95	=	090	71	SBR	130	46	46
011	42	STD	051	35	1/X	091	01	01	131	43	RCL
012	01	01	052	42	STD	092	46	46	132	09	09
013	99	PRT	053	07	07	093	43	RCL	133	65	×
014	91	R/S	054	75	-	094	01	01	134	43	RCL
015	42	STD	055	01	1	095	65	×	135	07	07
016	02	02	056	95	=	096	43	RCL	136	95	=
017	99	PRT	057	94	+/-	097	08	08	137	71	SBR
018	91	R/S	058	42	STD	098	65	×	138	01	01
019	99	PRT	059	08	08	099	43	RCL	139	46	46
020	32	X↔T	060	01	1	100	05	05	140	43	RCL
021	53	(	061	75	-	101	95	=	141	09	09
022	43	RCL	062	43	RCL	102	71	SBR	142	65	×
023	01	01	063	01	01	103	01	01	143	43	RCL
024	75	-	064	75	-	104	46	46	144	08	08
025	43	RCL	065	43	RCL	105	43	RCL	145	95	=
026	02	02	066	02	02	106	02	02	146	44	SUM
027	54	)	067	95	=	107	65	×	147	06	06
028	50	I↔I	068	42	STD	108	43	RCL	148	32	X↔T
029	55	÷	069	09	09	109	08	08	149	73	RC*
030	32	X↔T	070	43	RCL	110	95	=	150	00	00
031	85	+	071	01	01	111	71	SBR	151	69	DP
032	01	1	072	65	×	112	01	01	152	04	04
033	95	=	073	43	RCL	113	46	46	153	32	X↔T
034	35	1/X	074	07	07	114	43	RCL	154	69	DP
035	42	STD	075	95	=	115	10	10	155	06	06
036	05	05	076	71	SBR	116	71	SBR	156	69	DP
037	75	-	077	01	01	117	01	01	157	20	20
038	01	1	078	46	46	118	46	46	158	92	RTN
039	95	=	079	43	RCL	119	43	RCL	159	00	0

Kostnad för telefonsamtal

av ANDERS PERSSON Örkelljunga

Eftersom min upplysning om att jag har konstruerat ett program för beräkning av samtalskostnad (PB 81-3 sid 28) har rönt ett visst intresse kommer här nu en mer utförlig beskrivning av metoden. Observera att programmet inte är generellt användbart eftersom det bara fungerar från vissa nummer i 0435-området.

För att använda programmet i sin nuvarande form måste Math/Utilities-modulen vara isatt. Mig veterligen reagerar TI-58 likadant som 59:an på ett s.k. vilande modulanrop, varför även 58 bör kunna användas.

Så här fungerar det:

1. Tryck A. Noll ska visas.
2. Mata in ev. riktnr. Om samtalet är ett när- eller lokalsamtal (utan riktnummer) ska du inte mata in någonting.
3. Tryck R/S.
4. Mata in abonnentnummer. Det har visserligen ingen betydelse för långväga samtal, men har det för kortare avstånd. Tag som regel att alltid ange abnr.
5. Tryck R/S igen. Nu ska ett visas, vilket betyder att det är klart för samtalsmätning.
6. Slå telefonnumret på telefonen. När den uppringde svarar, tryck på R/S.
7. Tryck på R/S en gång till när samtalet är slut. Här ska programmet inte stanna, utan bara hoppa in i modulen och sedan fortsätta med att räkna ut kostnaden. Om displayen indikerar att exekveringen har avbrutits har du haft R/S nertryckt för länge. Tryck en gång till.
   Om däremot ingenting alls händer är det värre, eftersom räknaren i så fall inte har hunnit uppfatta tangenttryckningen. Detta är lite svårare att se, då det ska ta några sekunder innan kostnaden beräknats.
   Detta verkar kanske avskräckande, men det är enklare i praktiken än vad det låter. Min räknare bryter inte programkörningen förrän efter 0,68 s, och det är fullt tillräckligt för att man ska hinna göra en tydlig tangentnedtryckning och sedan släppa igen.
   Nu kommer kostnaden att visas i kronor och ören. Om man föredrar att få antalet markeringar ges de med D, kostnaden visas igen med C.
8. För ett nytt samtal, börja om med p. 1.

Ja, hade nu alla bott i Åsljunga som jag kunde jag ha slutat här. För er andra ska jag försöka förklara hur det fungerar.

Vi kan börja med den tabell som finns lagrad i minnena 06-29.
Här finns samtliga (så när som på vissa specialfall) riktnummer grupperade med avseende på taxorna till varje område.
Register 6 anger att samtal till område med riktnummer i intervallet (011,0149)kan pågå i 12 sekunder per markering. Minne 7 anger att man får prata i 10 sekunder per markering till områdena (0150,0299). Talet 999 i register 29 är inte ett riktnummer, utan bara slutet på tabellen. Det högsta riktnumret är 0981 (Vittangi). OBS! Riktnummerna är sorterade i nummerordning på Televerkets vis. 08 räknas som 800, 042 som 420 etc. Den geografiska fördelningen blir mer trivsam på det sättet. Hur får man då ihop en sådan här tabell? Ja, det är det som är det jobbiga. Beväpna dig med en telefonkatalog över ditt område (inte någon liten lokalkatalog) och försök hänga med i beskrivningen.

Börja med att slå upp uppslagsdelen i katalogen. Det är den gröna delen längst bak. Slå upp sidan som handlar om taxor för samtal. Läs den. Det underlättar förståelsen för programmet. Studera speciellt skillnaden mellan lokal- och rikssamtal. Studera också kopplingen mellan "Avgift per minut" och "Tidsmellanrum mellan markeringarna".

Bläddra nu till början av ditt eget riktnummerområde. Där finns information om vad samtalen kostar till olika riktnummerområden. Börja med att sätta ihop en tabell med olika kolumner för 540, 45, 24, 15, 12 samt 10 sekunder per markering. Uppgifterna får du ur en tabell som kallas "Samtalskostnad till olika riktnummerområden (öre/minut)". Tag hjälp av uppslagsdelen om du har hunnit glömma hur ofta markering sker vid en viss minutkostnad. VIKTIGT! Om det inte står att en markering kostar 20 öre i din katalog, utan t.ex. 18 öre, är den föråldrad. Detta spelar dock ingen roll ännu, men kommer att göra det.
Nåväl, när du är färdig med tabellen kommer du förmodligen att upptäcka att det inte finns något riktnummer i kolumnerna för 10 och 540 sekunder per markering. Titta nu på nästa tabell angående närliggande områden. Avgiften här är i regel olika beroende på vilket abonnentnummer man ringer till. De riktnummer som har den egenheten är de specialfall som nämts tidigare. En särskild testprocedur måste läggas in i programmet för att ta hand om dem. Den tabell som finns i minnena kommer alltså inte att omfatta dessa nummer, såvida inte avgiften till ett av dessa områden råkar vara lika i hela området från just din telefon. Då behövs inget speciellt testande för dessa.

Till sist har vi de dyraste områdena. De är i regel inte tabellerade, utan anges som övriga. Titta i början av katalogen. Där finns samtliga riktnummer från 011 (Norrköping) till 0981 (Vittangi). Nu är det lämpligt om du har numren i tabellen du nyss har gjort i växande ordning. Använd sorteringsprogrammet i MU-modulen. Sedan är det bara till att pricka av i den stora tabellen i katalogen. Alla riktnummer som inte finns med i din egen tabell ska in i kolumnen för 10 sekunder per markering.

Har du gjort rätt nu ska du ha en tabell med 275 olika riktnummer. Bortse från de där kostnaden varierar inom riktnummerområdet, om du nu har tagit med dem alls i tabellen. Se till att resten är i nummerordning inom varje kolumn. Nu får du gruppera riktnummerna så att det inte finns något nummer inom en grupp med en taxa skild från resten av gruppen. Om t.ex. samtliga nummer inom intervallet (0150, 0297) utom (0227,0246) har samma tid mellan två markeringar får du tre grupper av dessa nummer, nämligen (0150,0226), (0227, 0246) och (0247,0297) Det är de högre av dessa nummer som ska matas in i tabellen i minnena tillsammans med en uppgift om hur många sekunder det går på en markering inom intervallet. För att komplicera sammanställningen finns inte alla nummer med. Mellan bl.a. 0253 och 0258 finns ingenting. Detta framgår av tabellen i början av katalogen. A andra sidan underlättas grupperingen av att områden i norra Sverige har högre nummer än de i södra Sverige. Tänk på att 08 är 800, inte 8.

När denna intervallindelning är färdig ska den anpassas till programmet. Det är dock tämligen enkelt. Tag det sista riktnumret i varje intervall och lägg till antalet sekunder per markering inom intervallet dividerat med tusen. Om vi i exemplet ovan antar att markeringslängden är 10 sekunder utom för (0227,0246) där den är 12 sekunder, ska följande lagras i minnena:

226.010	XX
246.012	XX+1
297.010	XX+2

Vilka minne det blir beror givetvis på övriga grupper av nummer. I den version av programmet som finns listad här är (011, 0144), (0150,0297) och (0300,0340) de tre första grupperna med 12, 10 respektive 15 sekunder per markering.
Kommen så här långt har du klarat av de stora bitarna. Nu kan vi börja intressera oss för hur programmet fungerar. Samtidigt kommer de speciella riktnumren att behandlas.
Börja vid Lbl A (140). Här lagras riktnummer (145) och abonnentnummer (148). Slutligen initieras den loop som tar tiden på samtalet. Denna ska ju avbrytas genom att R/S trycks ner från tangentbordet och då hoppa in i modulen. På steg 072 börjar en sekvens i matematikmodulen som innehåller RTN LBL A' PGM 00 A' RTN. Denna gör alltså ingenting utan hoppar tillbaka till programmet igen. Tidmätningen utförs i början av programmet (005-009). När man trycker ned R/S - motsvarande punkt 7 i instruktionen - hoppar körningen till Lbl A' (sist). RST utförs för att döda de återhoppsadresser som finns lagrade. Sedan fortsätter programmet med Lbl INV (164). Här görs antalet varv i tidmätningsloopen om till sekunder. Konstanten i minne 01 anger antalet sekunder per varv på min räknare. Denna får man prova ut enskilt för varje räknare. Programmera in steg 000-009, se till att moduiens PGM 16 steg 072 nyss har anropats, lägg in Lbl A' och INV någonstans (A' lämpligen långt bak i minnet) och se hur många varv räknaren hinner på en viss tid. Gärna 10 minuter för att få bra noggrannhet. Jag har använt OP 20, men det verkar som om SUM 00 är snabbare (en etta finns ju i x-reg.). I så fall ger det en ännu större upplösning.
Nåväl, låt oss anta att denna kalibrering är utförd. Efter tidmätningen omvandlas riktnummer så att de får tre siffror (169-181).

Fotnot. Ett mycket enkelt program för samma ändamål finns under "Idébörsen" på s.37.

Om inget riktnummer matats in (dvs noll) hoppas till 057. Där tas telefonnumrets två första siffror fram (057-069). Om dessa är 60 hoppas till steg 200 där kostnaden för en markering visas. Detta innebär nämligen samtal till abonnent ansluten till samma telefonstation som jag är ansluten till (Mitt nummer: 60259). Annars är taxan 540 sekunder - 9 minuter - per markering (076-081).
Om ett riktnummer var inmatat anropas steg 010 (187). Där sållas speciella riktnummer ifrån - 0430, 0433 och 0451 - innan sökning i tabellen börjar. De taxor som speciella riktnummer har plockas fram på steg 093-138. Här är det första siffran i den uppringdes telefonnummer som är avgörande. (082-090) tar fram den första siffran i ett tal.
Om det nu inte var något speciellt telefonnummer genomsökes tabellen (028-045). När rätt nummer hittats (037) hoppar programmet till steg 049. Där tas information om hur länge en markering varar fram.
Sedan skuttar pekaren ner till 190 där kostnaden beräknas. Observera att det gäller antalet påbörjade perioder, därav (196-198).
Beträffande riktnummer kan man se att till det speciella numret 0430 kan man prata 540 sekunder för en markering om den uppringdes nummer börjar på 3,4 eller 5, annars 45 sekunder. Dessa regler finns i den andra tabellen längst fram vid ditt eget område i telefonkatalogen.

På steg 201-202 och 225-226 finns kostnaden per markering lagrad. Denna är alltså 20 öre när detta skrives. Genom att programmet bygger på markeringsintervall och inte minutkostnad är dessa steg de enda som behöver ändras om televerket ändrar (läs höjer) taxan. Så skedde för en tid sedan, från 18 till 20 öre.
Inom det egna riktnummerområdet är taxan oftast en markering var 540:e sekund, såvida inte de två första siffrorna i abonnentnummerna stämmer överens. Om så är fallet får man i regel ha linjen öppen hur länge som helst för en markering.
För dig som inte har matematikmodulen rekommenderar jag en studie av Reaktionstestprogrammet i PB 81-1 sidan 24. Där finns ett program som använder knepet med vilande modulanrop listat för både standardmodulen och matematikmodulen. Den störande utskriften i det exemplet spelar ingen roll här, eftersom skrivaren inte används.

Ja, nu hoppas jag att de som har visat intresse för mitt program ska kunna konvertera det till sin egen telefon. Det finns ju utvecklingsmöjligheter också. Man kan t.ex. tänka sig att lägga in en Pause-instruktion som kan visa den löpande kostnaden medan samtalet pågår. Man kan ju vilja sluta innan kostnaden blivit alltför våldsam. En annan möjlighet vore kapacitet att ta emot landsnummer också, så att man kan beräkna kostnaden för utlandssamtal också. Dessa taxor varierar visserligen med klockslaget, varför även tiden måste matas in i ett sådant fall. Det tycks för övrigt vara meningen att det ska bli så inom Sverige också, om några år eller så.
Jag hoppas att ni som konstruerar något nytt i den här stilen låter mig och alla andra ta del av det!

0.	00	414.015	15
.1963627238	01	415.045	16
0.	02	417.024	17
0.	03	439.54	18
0.	04	453.024	19
0.	05	455.015	20
149.012	06	456.024	21
299.01	07	459.015	22
344.015	08	469.024	23
345.024	09	475.015	24
349.015	10	476.024	25
359.024	11	478.015	26
399.015	12	489.024	27
412.024	13	529.012	28
413.045	14	999.01	29

```
000 61 GTO   060 53 (    120 01 01   180 02 02
001 22 INV   061 28 LOG  121 11 11   181 95 =
002 36 PGM   062 59 INT  122 05 5    182 32 X:T
003 16 16    063 75 -    123 67 EQ   183 25 CLR
004 51 BST   064 01 1    124 01 01   184 67 EQ
005 69 OP    065 54 )    125 11 11   185 00 00
006 20 20    066 22 INV  126 04 4    186 57 57
007 61 GTO   067 28 LOG  127 05 5    187 71 SBR
008 00 00    068 95 =    128 92 RTN  188 00 00
009 05 05    069 59 INT  129 71 SBR  189 10 10
010 04 4     070 32 X:T  130 00 00   190 35 1/X
011 03 3     071 06 6    131 82 82   191 65 ×
012 00 0     072 00 0    132 04 4    192 43 RCL
013 67 EQ    073 67 EQ   133 67 EQ   193 00 00
014 00 00    074 02 02   134 01 01   194 95 =
015 93 93    075 00 00   135 11 11   195 59 INT
016 04 4     076 05 5    136 04 4    196 85 +
017 03 3     077 04 4    137 05 5    197 01 1
018 03 3     078 00 0    138 92 RTN  198 95 =
019 67 EQ    079 61 GTO  139 76 LBL  199 65 ×
020 01 01    080 01 01   140 11 A    200 93 .
021 15 15    081 90 90   141 22 INV  201 02 2
022 04 4     082 43 RCL  142 58 FIX  202 00 0
023 05 5     083 03 03   143 25 CLR  203 95 =
024 01 1     084 55 ÷    144 91 R/S  204 58 FIX
025 67 EQ    085 28 LOG  145 42 STO  205 02 02
026 01 01    086 59 INT  146 02 02   206 42 STO
027 29 29    087 22 INV  147 91 R/S  207 04 04
028 02 2     088 28 LOG  148 42 STO  208 91 R/S
029 04 4     089 95 =    149 03 03   209 76 LBL
030 42 STO   090 59 INT  150 25 CLR  210 13 C
031 04 04    091 32 X:T  151 42 STO  211 43 RCL
032 06 6     092 92 RTN  152 00 00   212 04 04
033 42 STO   093 71 SBR  153 01 1    213 58 FIX
034 05 05    094 00 00   154 36 PGM  214 02 02
035 73 RC*   095 82 82   155 16 16   215 91 R/S
036 05 05    096 03 3    156 71 SBR  216 76 LBL
037 77 GE    097 67 EQ   157 00 00   217 14 D
038 00 00    098 01 01   158 72 72   218 43 RCL
039 49 49    099 11 11   159 91 R/S  219 04 04
040 69 OP    100 04 4    160 61 GTO  220 55 ÷
041 25 25    101 67 EQ   161 00 00   221 93 .
042 97 DSZ   102 01 01   162 02 02   222 02 2
043 04 04    103 11 11   163 76 LBL  223 00 0
044 00 00    104 05 5    164 22 INV  224 95 =
045 35 35    105 67 EQ   165 43 RCL  225 22 INV
046 25 CLR   106 01 01   166 01 01   226 58 FIX
047 35 1/X   107 11 11   167 49 PRD  227 91 R/S
048 91 R/S   108 04 4    168 00 00   228 76 LBL
049 22 INV   109 05 5    169 02 2    229 16 A'
050 59 INT   110 92 RTN  170 75 -    230 81 RST
051 65 ×     111 05 5    171 43 RCL
052 03 3     112 04 4    172 02 02
053 22 INV   113 00 0    173 28 LOG
054 28 LOG   114 92 RTN  174 59 INT
055 95 =     115 71 SBR  175 95 =    140 11 A
056 92 RTN   116 00 00   176 22 INV  164 22 INV
057 43 RCL   117 82 82   177 28 LOG  210 13 C
058 03 03    118 03 3    178 65 ×    217 14 D
059 55 ÷     119 67 EQ   179 43 RCL  229 16 A'
```


Beräkningar med

KOMPLEXA TAL

med RPN och stack

av G JENNINGS 'BTIUC'

För beräkningar med komplexa tal finns det som bekant tre program i standardmodulen, nämligen ML-04, ML-05 och ML-06. Ett av dessa program måste anropas, beroende på vilket problem man vill lösa, och sedan får man trycka någon av tangenterna B - E' enligt instruktionsboken. G. Jennings i den tyvärr numera nedlagda British TI Users' Club har för att förenkla behandlingen gjort detta ganska charmiga program där man dessutom enligt HP-modell har tillgång till en stack med fyra register. OBS att stackens fjärde register kallas "T-register" (med stort T) till skillnad från räknarens t-register. Observera också att programmet använder ML-modulen som alltså måste vara isatt.

Programmet presenterades ursprungligen i två nummer av BTIUC:s medlemsblad på det sättet att första numret kunde användas av både TI-58 och TI-59-ägare medan kompletteringen i följande nummer bara kunde användas av TI-59 ägare. Vi gör på liknande sätt, men i stället för komplettering ger vi en hel listning för TI-59 och undgår på så sätt några otympligheter.

I rutinen A' som är en intern subrutin och alltså inte skall användas från tangentbordet förekommer en finess som beskrivs i detta nummers "Programbitar". Genom användning av den syntetiska koden "21" kan man få räknaren att välja mellan två labels D beroende på om displayen är "mjuk" eller "hård".

Vid inmatning av tal slås förs talets realdel följt av + (eller -!) och därefter imaginärdelen följd av A. Programmet känner ju efter om någon operation är vilande! Negativt tecken hos endera av talets delar måste ges med +/- (minustecknet ovan är här betydelselöst!).
För 2 + 3i mata in 2 + 3 A
" 2 - 3i -"- 2 + 3 +/- A

Stacken manövreras på följande sätt:
A Mata in ett tal i stacken (ENTER ↑)
B Rulla upp stacken (R ↑)
C Rulla ned stacken (R ↓)
D Visa innehållet i x-registret
E Visa innehållet i y-registret
D' -"- z-registret
E' -"- T-registret
SBR x:t Byt innehållen i x- och y-registren
SBR CE Rulla ned stacken och lägg in 0 i T-registret
SBR CLR Nollställ stacken
SBR INV Användes med inversa funktioner - se följande.

A', B' och C' är alltså interna subrutiner. A' avkodar inmatningen från tangentbordet och är likvärdigt med D om ingen inmatning har gjorts. B' rullar upp stacken och C' rullar ned den.

Grundläggande aritmetiska funktioner:
Det följande programavsnittet (Lbl + till Lbl Lbl) gör de här sakerna:
SBR + Adderar x och y och rullar ner stacken
SBR - Subtraherar x från y och -"-
SBR x Multiplicerar x och y och -"-
SBR ÷ Dividerar y med x och -"-
SBR y^x Upphöjer y till x:te digniteten och -"-
SBR INV SBR y^x Tar x:te roten ur y och -"-.
I samtliga dessa fall läggs resultatet i x-registret.

Minnesfunktioner
för TI-58 (endast ett minne):
Enkla minnesfunktioner (Lbl STO - Lbl RCL)
SBR STO Lagrar x i minnet
SBR RCL Rullar upp stacken och lagrar minnesinnehållet i x-registret.
Andra minnesfunktioner (Lbl Exc - Lbl ( )
SBR Exc Byter innehåll i minne och x-register
SBR SUM Adderar x-registrets innehåll till minnet.
SBR INV SBR SUM Subtraherar x-registrets innehåll från minnet.

(Minnesfunktioner för TI-59, se längre fram.)

Trigonometriska funktioner (Lbl sin - Lbl Prt)
SBR sin Ersätter x med sin x.
SBR INV SBR sin Ersätter x med arcsin x.
Samma sak gäller de övriga trigonometriska funktionerna. Vinkelmått radianer väljs automatiskt av biblioteksrutinerna. Ovanstående rutiner bevarar inte innehållet i T-registret som sätts lika med z-registret.

Andra funktioner (Lbl P/R - Lbl +/-)

SBR P/R Visar x under formen (r, Θ) med r i displayen och Θ (i radianer) i (räknarens) t-register.

SBR x^2 Ersätter x med x^2.

SBR $\sqrt{x}$ -"- $\sqrt{x}$.

SBR 1/x -"- 1/x.

SBR lnx -"- ln x (principalvärde).

SBR INV SBR lnx -"- e^x.

SBR +/- -"- x^* (konjugerat komplext).

FÖRÄNDRINGAR OCH KOMPLETTERINGAR FÖR TI-59 (se separat listning)

Det finns 25 minnen numrerade 0 - 24. Dessutom kan man adressera stackens register med -1 för T, -2 för z, -3 för y och -4 för x-registret. Om minnesnumret är n ger

n SBR STO	lagring av x i minne n.
n SBR RCL	återkallning av minne n i x-registret och upprullning av stacken.
n SBR Exc	byte av innehåll i x och minne n.
n SBR SUM	addition av x till minne n.
n SBR INV SBR SUM	subtraktion av x från minne n.
n SBR CMs	nollställning av minnena fr o m 0 t o m n.
SBR CLR	nollställning av alla register i stacken.

Det skulle också gå att skriva rutiner för minnesmultiplikation och do. division men man måste då vara försiktig med minne -4 (x-registret) då dess innehåll skulle ändras under beräkningens gång.

ALLMÄNT:

Uppdelningen är för TI-58 399.09 och för TI-59 standarduppdelning.

Felaktiga siffervärden kan naturligtvis tas bort med CE och CLR så länge de inte matats in i stacken. Om detta gjorts kan man trycka SBR CE som rullar ner stacken så att x-registret förloras.

Det enda SBR INV gör är att sätta flagga 0. Alla funktioner som kan ha INV tar också bort flagga 0 efteråt, dock ej de andra. Om SBR INV tryckts av misstag rekommenderas alltså "INV Stf 0".

Efter de flesta rutiner stannar programmet med x-registrets realdel i display och imaginärdelen i (räknarens) t-register. Om inga tangenttryckningar görs är det inte nödvändigt att ta tillbaka realdelen i display innan man fortsätter.

ANVÄNDNINGSEXEMPEL 1

Beräkna $\arctan\{[2+3i - \ln(1-2i)]^{(4-5i)}\}$

Tryck	Display	Räknarens t-register
2 + 3 A	2	3
1 + 2 +/- A	1	-2
SBR lnx	0.8047189562	-1.107148718
SBR -	1.195281044	4.107148718
4 + 5 +/- A	4	-5
SBR y^x	-108629.5454	-178892.9745
SBR INV SBR tan	1.570798807	-0.000004084

ANVÄNDNINGSEXEMPEL 2

Beräkna $\sqrt[i]{-1}$.

Tryck	Display	Räknarens t-register
1 +/- + 0 A	-1	0
0 + 1 A	0	1
SBR INV SBR y^x	23.14069263	0

Resultatet är alltså (något överraskande för den oinvigde) reellt. Med hjälp av sambandet $e^{ix} = \cos x + i \sin x$ får man ju för $x = \pi$ $e^{i\pi} = -1 + 0$ och om man drar i:te roten ur båda sidorna får man alltså $\sqrt[i]{-1} = e^{\pi}$.

Programmets stora nackdel är den långa exekveringstiden för stackoperationer. Tyvärr går det inte att använda indirekt adressering - delvis därför att stacken måste kunna "rotera" men mest därför att biblioteksprogrammen arbetar med fasta minnen.

Den längsta exekveringstiden är ungefär 15 s för tan z, där stacken rullas upp för att spara y-registret (modulprogrammet använder R03 och R04), modulprogrammet anropas och stacken därefter rullas ner igen.

Programmet går i nästan alla displayformat. Enda undantaget är subrutinerna SUM och INV SUM för TI-58 som innehåller en INV EE-instruktion för att komma runt problemen med HIR-funktionerna. Normal representation och Eng-format behålls men EE-format ändras till normal representation.

Om reella tal matas in måste imaginärdelen 0 matas in samtidigt.

2 A matar in 0+2i

2 + A räknas ej som inmatning

2 + 0 A matar in 2+0i dvs 2.

Den märkliga rutinen A' fungerar endast om den första "Lbl D" står på steg 000. Lägg märke till den finurliga sekvensen "INV Lbl INV Stf 0" som ger motsatt effekt om den tas rakt igenom mot om man trycker "SBR INV".

I det allra senaste numret av TI PPC Notes fanns ett annat program för komplexa tal. Det utnyttjar inte ML-modulen men kan göra detsamma som den utom P/R. Behandlingen blir även där RPN-lik men med endast ett register i stacken. Om det finns tillräckligt många önskemål kan vi naturligtvis ta in även detta.

Registerdispositionen är

(variabel)	realdel	imaginärdel
x-register	R01	R02
y-register	R03	R04
z-register	R05	R06
T-register	R07	R08
Minne 0 / Minne 24	R09 / R57	R10 / R58

```
000 76 LBL   065 16 A'    130 24 CE    195 87 IFF   260 13 C     325 00 00    390 01 01    455 00 00
001 14 D     066 17 B'    131 29 CP    196 00 00    261 92 RTN   326 92 RTN   391 69 OP    456 69 OP
002 43 RCL   067 61 GTO   132 00 0     197 79 X̄     262 76 LBL   327 76 LBL   392 30 30    457 30 30
003 02 02    068 14 D     133 18 C'    198 11 A     263 34 √X    328 42 STO   393 74 SM*   458 44 SUM
004 32 X:T   069 76 LBL   134 42 STO   199 36 PGM   264 36 PGM   329 71 SBR   394 00 00    459 02 02
005 43 RCL   070 12 B     135 02 02    200 06 06    265 05 05    330 70 RAD   395 92 RTN   460 73 RC*
006 01 01    071 10 E'    136 32 X:T   201 12 B     266 14 D     331 43 RCL   396 76 LBL   461 00 00
007 22 INV   072 17 B'    137 42 STO   202 61 GTO   267 92 RTN   332 02 02    397 61 GTO   462 69 OP
008 76 LBL   073 61 GTO   138 01 01    203 95 =     268 76 LBL   333 72 ST*   398 43 RCL   463 30 30
009 22 INV   074 14 D     139 92 RTN   204 76 LBL   269 35 1/X   334 00 00    399 02 02    464 44 SUM
010 86 STF   075 76 LBL   140 76 LBL   205 79 X̄     270 71 SBR   335 32 X:T   400 22 INV   465 01 01
011 00 00    076 13 C     141 85 +     206 11 A     271 37 P/R   336 69 OP    401 74 SM*   466 97 DSZ
012 92 RTN   077 71 SBR   142 36 PGM   207 36 PGM   272 33 X²    337 30 30    402 00 00    467 10 10
013 76 LBL   078 32 X:T   143 04 04    208 06 06    273 35 1/X   338 43 RCL   403 32 X:T   468 69 OP
014 16 A'    079 15 E     144 12 B     209 17 B'    274 49 PRD   339 01 01    404 69 OP    469 61 GTO
015 21 2ND   080 18 C'    145 76 LBL   210 61 GTO   275 01 01    340 72 ST*   405 30 30    470 14 D
016 93 .     081 61 GTO   146 95 =     211 95 =     276 94 +/-   341 00 00    406 43 RCL
017 21 2ND   082 14 D     147 10 E'    212 76 LBL   277 49 PRD   342 92 RTN   407 01 01
018 21 2ND   083 76 LBL   148 18 C'    213 39 COS   278 02 02    343 76 LBL   408 22 INV
019 14 D     084 15 E     149 61 GTO   214 87 IFF   279 61 GTO   344 43 RCL   409 74 SM*   001 14 D
020 92 RTN   085 43 RCL   150 14 D     215 00 00    280 14 D     345 71 SBR   410 00 00    009 22 INV
021 76 LBL   086 04 04    151 76 LBL   216 89 π     281 76 LBL   346 70 RAD   411 92 RTN   014 16 A'
022 14 D     087 32 X:T   152 75 -     217 11 A     282 23 LNX   347 73 RC*   412 76 LBL   022 14 D
023 32 X:T   088 43 RCL   153 71 SBR   218 36 PGM   283 87 IFF   348 00 00    413 47 CMS   028 17 B'
024 00 0     089 03 03    154 32 X:T   219 06 06    284 00 00    349 32 X:T   414 71 SBR   048 18 C'
025 95 =     090 92 RTN   155 36 PGM   220 13 C     285 78 Σ+    350 69 OP    415 70 RAD   064 11 A
026 92 RTN   091 76 LBL   156 04 04    221 61 GTO   286 36 PGM   351 30 30    416 42 STO   070 12 B
027 76 LBL   092 19 D'    157 17 B'    222 95 =     287 05 05    352 73 RC*   417 09 09    076 13 C
028 17 B'    093 43 RCL   158 61 GTO   223 76 LBL   288 16 A'    353 00 00    418 00 0     084 15 E
029 48 EXC   094 06 06    159 95 =     224 89 π     289 92 RTN   354 17 B'    419 76 LBL   092 19 D'
030 01 01    095 32 X:T   160 76 LBL   225 11 A     290 76 LBL   355 61 GTO   420 80 GRD   100 10 E'
031 48 EXC   096 43 RCL   161 65 ×     226 36 PGM   291 78 Σ+    356 14 D     421 72 ST*   108 32 X:T
032 03 03    097 05 05    162 36 PGM   227 06 06    292 36 PGM   357 76 LBL   422 09 09    115 25 CLR
033 48 EXC   098 92 RTN   163 04 04    228 18 C'    293 05 05    358 48 EXC   423 97 DSZ   130 24 CE
034 05 05    099 76 LBL   164 13 C     229 61 GTO   294 17 B'    359 71 SBR   424 09 09    141 85 +
035 48 EXC   100 10 E'    165 61 GTO   230 95 =     295 61 GTO   360 70 RAD   425 80 GRD   146 95 =
036 07 07    101 43 RCL   166 95 =     231 76 LBL   296 14 D     361 43 RCL   426 92 RTN   152 75 -
037 32 X:T   102 08 08    167 76 LBL   232 30 TAN   297 76 LBL   362 02 02    427 76 LBL   161 65 ×
038 48 EXC   103 32 X:T   168 55 ÷     233 87 IFF   298 50 I×I   363 63 EX*   428 52 EE    168 55 ÷
039 02 02    104 43 RCL   169 71 SBR   234 00 00    299 71 SBR   364 00 00    429 43 RCL   177 45 YX
040 48 EXC   105 07 07    170 32 X:T   235 99 PRT   300 37 P/R   365 42 STO   430 09 09    187 76 LBL
041 04 04    106 92 RTN   171 36 PGM   236 11 A     301 42 STO   366 02 02    431 85 +     194 38 SIN
042 48 EXC   107 76 LBL   172 04 04    237 36 PGM   302 01 01    367 32 X:T   432 42 STO   205 79 X̄
043 06 06    108 32 X:T   173 18 C'    238 06 06    303 00 0     368 69 OP    433 10 10    213 39 COS
044 48 EXC   109 36 PGM   174 61 GTO   239 14 D     304 42 STO   369 30 30    434 85 +     224 89 π
045 08 08    110 04 04    175 95 =     240 61 GTO   305 02 02    370 43 RCL   435 01 1     232 30 TAN
046 92 RTN   111 10 E'    176 76 LBL   241 95 =     306 61 GTO   371 01 01    436 44 SUM   243 99 PRT
047 76 LBL   112 61 GTO   177 45 YX    242 76 LBL   307 14 D     372 63 EX*   437 10 10    251 37 P/R
048 18 C'    113 14 D     178 87 IFF   243 99 PRT   308 76 LBL   373 00 00    438 01 1     257 33 X²
049 48 EXC   114 76 LBL   179 00 00    244 11 A     309 94 +/-   374 42 STO   439 02 2     263 34 √X
050 07 07    115 25 CLR   180 76 LBL   245 36 PGM   310 01 1     375 01 01    440 95 =     269 35 1/X
051 48 EXC   116 36 PGM   181 36 PGM   246 06 06    311 94 +/-   376 92 RTN   441 42 STO   282 23 LNX
052 05 05    117 01 01    182 04 04    247 19 D'    312 49 PRD   377 76 LBL   442 00 00    291 78 Σ+
053 48 EXC   118 71 SBR   183 14 D     248 61 GTO   313 02 02    378 44 SUM   443 11 A     298 50 I×I
054 03 03    119 25 CLR   184 61 GTO   249 95 =     314 61 GTO   379 71 SBR   444 00 0     309 94 +/-
055 32 X:T   120 42 STO   185 95 =     250 76 LBL   315 14 D     380 70 RAD   445 42 STO   317 70 RAD
056 48 EXC   121 07 07    186 76 LBL   251 37 P/R   316 76 LBL   381 87 IFF   446 01 01    328 42 STO
057 08 08    122 42 STO   187 76 LBL   252 36 PGM   317 70 RAD   382 00 00    447 42 STO   344 43 RCL
058 48 EXC   123 08 08    188 36 PGM   253 05 05    318 65 ×     383 61 GTO   448 02 02    358 48 EXC
059 06 06    124 22 INV   189 04 04    254 12 B     319 02 2     384 43 RCL   449 76 LBL   378 44 SUM
060 48 EXC   125 86 STF   190 15 E     255 92 RTN   320 85 +     385 02 02    450 69 OP    397 61 GTO
061 04 04    126 00 00    191 61 GTO   256 76 LBL   321 01 1     386 74 SM*   451 36 PGM   413 47 CMS
062 92 RTN   127 29 CP    192 95 =     257 33 X²    322 00 0     387 00 00    452 04 04    420 80 GRD
063 76 LBL   128 92 RTN   193 76 LBL   258 36 PGM   323 95 =     388 32 X:T   453 13 C     428 52 EE
064 11 A     129 76 LBL   194 38 SIN   259 05 05    324 42 STO   389 43 RCL   454 73 RC*   450 69 OP
```

Här nedan en förkortad listning för TI-58. Stegen 000 - 192 är gemensamma med listningen för TI-59 (överst).

```
193 76 LBL   226 01 01    259 22 INV   292 95 =     325 92 RTN   358 16 A'    001 14 D     321 37 P/R
194 42 STO   227 82 HIR   260 52 EE    293 76 LBL   326 76 LBL   359 92 RTN   009 22 INV   327 33 X²
195 43 RCL   228 05 05    261 14 D     294 89 π     327 33 X²    360 76 LBL   014 16 A'    333 34 √X
196 02 02    229 14 D     262 92 RTN   295 11 A     328 36 PGM   361 78 Σ+    022 14 D     339 35 1/X
197 82 HIR   230 92 RTN   263 76 LBL   296 36 PGM   329 05 05    362 36 PGM   028 17 B'    352 23 LNX
198 06 06    231 76 LBL   264 38 SIN   297 06 06    330 13 C     363 05 05    048 18 C'    361 78 Σ+
199 32 X:T   232 44 SUM   265 87 IFF   298 18 C'    331 92 RTN   364 17 B'    064 11 A     368 50 I×I
200 43 RCL   233 52 EE    266 00 00    299 61 GTO   332 76 LBL   365 61 GTO   070 12 B     379 94 +/-
201 01 01    234 87 IFF   267 79 X̄     300 95 =     333 34 √X    366 14 D     076 13 C
202 82 HIR   235 00 00    268 11 A     301 76 LBL   334 36 PGM   367 76 LBL   084 15 E
203 05 05    236 53 (     269 36 PGM   302 30 TAN   335 05 05    368 50 I×I   092 19 D'
204 92 RTN   237 43 RCL   270 06 06    303 87 IFF   336 14 D     369 71 SBR   100 10 E'
205 76 LBL   238 02 02    271 12 B     304 00 00    337 92 RTN   370 37 P/R   108 32 X:T
206 43 RCL   239 82 HIR   272 61 GTO   305 99 PRT   338 76 LBL   371 42 STO   115 25 CLR
207 82 HIR   240 36 36    273 95 =     306 11 A     339 35 1/X   372 01 01    130 24 CE
208 16 16    241 32 X:T   274 76 LBL   307 36 PGM   340 71 SBR   373 00 0     141 85 +
209 32 X:T   242 43 RCL   275 79 X̄     308 06 06    341 37 P/R   374 42 STO   146 95 =
210 82 HIR   243 01 01    276 11 A     309 14 D     342 33 X²    375 02 02    152 75 -
211 15 15    244 82 HIR   277 36 PGM   310 61 GTO   343 35 1/X   376 61 GTO   161 65 ×
212 17 B'    245 35 35    278 06 06    311 95 =     344 49 PRD   377 14 D     168 55 ÷
213 14 D     246 22 INV   279 17 B'    312 76 LBL   345 01 01    378 76 LBL   177 45 YX
214 92 RTN   247 52 EE    280 61 GTO   313 99 PRT   346 94 +/-   379 94 +/-   187 76 LBL
215 76 LBL   248 92 RTN   281 95 =     314 11 A     347 49 PRD   380 01 1     194 42 STO
216 48 EXC   249 76 LBL   282 76 LBL   315 36 PGM   348 02 02    381 94 +/-   206 43 RCL
217 82 HIR   250 53 (     283 39 COS   316 06 06    349 61 GTO   382 49 PRD   216 48 EXC
218 16 16    251 43 RCL   284 87 IFF   317 19 D'    350 14 D     383 02 02    232 44 SUM
219 48 EXC   252 02 02    285 00 00    318 61 GTO   351 76 LBL   384 61 GTO   250 53 (
220 02 02    253 82 HIR   286 89 π     319 95 =     352 23 LNX   385 14 D     264 38 SIN
221 82 HIR   254 56 56    287 11 A     320 76 LBL   353 87 IFF                275 79 X̄
222 06 06    255 43 RCL   288 36 PGM   321 37 P/R   354 00 00                283 39 COS
223 82 HIR   256 01 01    289 06 06    322 36 PGM   355 78 Σ+                294 89 π
224 15 15    257 82 HIR   290 13 C     323 05 05    356 36 PGM               302 30 TAN
225 48 EXC   258 55 55    291 61 GTO   324 12 B     357 05 05                313 99 PRT
```

Presentation av

TI 55-II

av Björn Gustavsson

TI-55-II är en av de senaste räknarna från TI. Den har en del nya, intressanta egenskaper. Den sägs också vara programmerbar, men den är ingen efterföljare till TI-57.

TI-55-II har som alla nya TI-räknare flytande kristall-display och kontinuerligt minne. Displayen är vinklad uppåt för att underlätta avläsning. Den har 45 tangenter. precis som TI-59, och tre skifttangenter, nämligen 2nd, INV och hyp (den senare för att beräkna hyperboliska funktioner).

Nästan alla tangenter har andrafunktioner. Det finns således ovanligt många funktioner för en räknare i den här prisklassen.

Som vanligt finns AOS (Algebraiskt OperationsSystem) med fyra vilande operationer och 15 parentesnivåer. En nyhet är en sorts "Last x"-register, som gör det möjligt att skifta talet på skärmen, x, med talet som matades in före förra funktions-tangenten, y.

Exempel: Antag att vi har tryckt 3 - 2, men egentligen skulle vi trycka 2 - 3. Vi kan då trycka x:y (x skift y) och skifta om talen. Trycker vi sedan = visas rätt resultat (-1). Ännu större nytta av tangenten har man vid upphöjning. x- och y-register utnyttjas även vid P/R-omvandling.

Vidare har räknaren kvadratrot, kvadrat (som andrafunktion), invertering (också som andrafunktion), trigonometriska funktioner, logaritmer, procent. Omvandlingar omfattar DMS-omvandling och den tidigare nämnda P/R-omvandlingen samt några enhetsomvandlingar främst för amerikaner, nämligen gallon-liter, pound-kg, tum-cm och Fahrenheit-Celsius. Fakultet och även kombinationer och permutationer finns (alltså samma som ML-16 på TI-59). Typiska "programmeringsfunktioner" finns också, alltså Intg (heltalsdel), Frac (decimaldel), absolutbelopp, Sgn (signumfunktion, ger -1 för $x < 0$ och 1 för $x \geq 0$, där x är talet i fönstret). Teknisk notation finns också, vilket jag aldrig har sett på en "liten" räknare.

Räknaren är också välförsedd med statistiska funktioner, t ex linjär regression. Inmatningen av statistiska data är förbättrad.

Programmering då? Tyvärr måste jag säga att det är räknarens svagaste punkt, trots att det finns stora möjligheter att ändra program, för SST, BST, Del och Ins finns. (Det går också väldigt snabbt, både att skjuta in/ta bort steg och att använda SST och BST; annars brukar flytande kristaller göra att det går långsamt, men så inte här.) Pause finns också.

Men RST fungerar så att stegräknaren återställs till första programsteget och programmet stoppas!!! Det gör det alltså omöjligt att använda loopar!!! Det finns förstås heller inga GTO-instruktioner eller testinstruktioner. Hur stor nytta har man då av Pause?

Totalt kan man ha 56 programsteg. Men då har man bara ett register. Det går också att få 8 register och inga programsteg eller vilken blandning som helst däremellan (varje register tar upp 8 programsteg). Uppdelning görs med 2nd Part n, där n är antalet register man vill ha.

En nyhet ligger också i registeraritmetiken. På TI-59 används SUM för att summera till ett register och Prd för att multiplicera in i ett register. På TI-55-II används samma smarta system som på HP-räknarna. Med STO görs allting. Om man vill summera till register 1 skriver man STO + 1; vill man multiplicera skriver man STO x 1. Till och med "upphöjt till" och rotutdragning fungerar!

En sådan instruktion tar emellertid i ett program upp tre programsteg, eftersom koderna inte är kopplade. I ett TI-57-program tar motsvarande instruktion upp ett programsteg (dock finns inte "upphöjt till" eller rotutdragning direkt till register).

Förutom att minnet inte är särskilt rymligt är det lätt att förlora ett program. Visserligen finns kontinuerligt minne som bevarar programmet när räknaren är avslagen, men om man använder statistikfunktioner raderas programminnet (statistikfunktioner kan alltså inte användas i ett program)! Också om man ändrar uppdelning så att programsteg försvinner raderas dessa steg. Råkar man trycka 2nd Part 8 har man 8 register - och programmet är borta.

Räknaren har dock ännu en nyhet. Det finns en speciell integrationstangent som integrerar en funktion som är inmatad som ett program. Så här gör man för att integrera en funktion:

Mata in funktionen som ett program. Avsluta med R/S. Lagra gränserna i register 1 och 2. Tryck på integreringstangenten. Räknaren frågar efter antalet intervall med Int 00. Mata in antalet intervall (högst 99) och tryck R/S. Metoden som används är Simpsons formel. Tar man många intervall tar det några minuter att slutföra beräkningen.

Sammanfattning:

TI-55-II har en mängd funktioner för tekniska beräkningar. De statistiska funktionerna är de bästa som jag har sett på någon TI-räknare. Tyvärr är den inte särskilt lämpad för programmering, förutom för att beräkna funktionsvärden eller integrera funktioner. Om man väljer att betrakta den som en icke-programmerbar räknare är den toppen!

Man får hoppas att efterföljaren till TI-59, som alla ivrigt väntar på, har några av de nya egenskaperna, t ex registeraritmetiken och "Last x"-register. ■

KONTROLL-SUMMA

av Björn Gustavsson

OBS! Denna artikel införes endast som information. Om kontrollsumma kommer att införas som regel vid införande av program kommer vi då att ange vilket kontrollsummeprogram som gäller. Det kan nämligen tänkas att nedanstående program inte är det slutgiltiga.

För att kontrollera om man knappat in ett program rätt från en listning vore det givetvis bra om något slag av kontrollberäkning kunde göras på det inknappade programmets koder.

I TI PPC Notes' senaste nummer (V6N9/10P3) fanns ett sådant program av Jules Bell. Det tvångsläser den programsida man vill undersöka i block 4 och gör sedan om detta minnesutrymme till minnesregister. Logaritmen av ett register i taget summeras till en kontrollsumma. Emellertid utnyttjas varken exponenten eller talets tecken. Detta i kombination att en kod som slutar på 8 eller 9 i visst läge ger "overflow"-beteckning i motsvarande register gör att det fortfarande finns luckor där fel kan slinka igenom.

Björn Gustavsson har visserligen inte lyckats täta dessa luckor i den här omgången, men han har gjort ett eget program som är användarvänligare och som visas här.

Det används så här. Läs in block 1, tryck A. Läs in block 1 igen. Mata in numret på det block du vill testa. Tryck R/S. Sätt in kortet. Upprepa för alla block. För att få kontrollsumman för hela programmet, tryck R/S. Resultatet skrivs som två tal för att få med alla 13 siffrorna.

PPX Exchange har i V5N5P6 haft ett annat program där man klarat "overflow"-problemet genom att rekommendera ett manuellt "Insert" fyra gånger före en förnyad körning, vilket ger ny kontrollsumma. Programmet blev emellertid på så sätt inte särskilt användarvänligt. Vi återkommer. ■

```
CHECKSUM:

BANK 1
      31.437
   0.4433885

PROG
      31.437
   0.4433885

003  76 LBL
004  11  A
005  36 PGM
006  02  02
007  71 SBR
008  02  02
009  39  39
010  09  9
011  00  0
012  22 INV
013  58 FIX
014  22 INV
015  57 ENG
016  98 ADV
017  01  1
018  05  5
019  02  2
020  03  3
021  01  1
022  07  7
023  01  1
024  05  5
025  02  2
026  06  6
027  69 OP
028  00  00
029  69 OP
030  01  01
031  03  3
032  06  6
033  04  4
034  01  1
035  03  3
036  00  0
037  06  6
038  02  2
039  58 FIX
040  02  02
041  69 OP
042  02  02
043  69 OP
044  05  05
045  01  1
046  04  4
047  01  1
048  03  3
049  03  3
050  01  1
051  02  2
052  06  6
053  69 OP
054  00  00
055  69 OP
056  01  01
057  98 ADV
058  22 INV
059  58 FIX
060  25 CLR
061  91 R/S
062  29 CP
063  67  EQ
064  01  01
065  64  64
066  85  +
067  01  1
068  95  =
069  58 FIX
070  08  08
071  69 OP
072  02  02
073  69 OP
074  05  05
075  22 INV
076  58 FIX
077  04  4
078  94 +/-
079  22 INV
080  96 WRT
081  02  2
082  09  9
083  93  .
084  01  1
085  42 STO
086  47  47
087  00  0
088  42 STO
089  48  48
090  73 RC*
091  47  47
092  50 I×I
093  29 CP
094  67  EQ
095  01  01
096  88  88
097  32 X:T
098  01  1
099  32 X:T
100  67  EQ
101  01  01
102  85  85
103  77  GE
104  01  01
105  07  07
106  35 1/X
107  28 LOG
108  22 INV
109  59 INT
110  65  ×
111  03  3
112  22 INV
113  28 LOG
114  54  )
115  42 STO
116  33  33
117  59 INT
118  55  ÷
119  03  3
120  22 INV
121  28 LOG
122  54  )
123  44 SUM
124  48  48
125  44 SUM
126  59  59
127  43 RCL
128  33  33
129  22 INV
130  59 INT
131  44 SUM
132  48  48
133  44 SUM
134  59  59
135  97 DSZ
136  47  47
137  00  00
138  90  90
139  25 CLR
140  43 RCL
141  48  48
142  65  ×
143  03  3
144  22 INV
145  28 LOG
146  54  )
147  42 STO
148  33  33
149  59 INT
150  55  ÷
151  03  3
152  22 INV
153  28 LOG
154  54  )
155  99 PRT
156  43 RCL
157  33  33
158  22 INV
159  59 INT
160  99 PRT
161  61 GTO
162  00  00
163  57  57
164  03  3
165  03  3
166  03  3
167  05  5
168  03  3
169  02  2
170  02  2
171  02  2
172  00  0
173  00  0
174  69 OP
175  00  00
176  69 OP
177  01  01
178  69 OP
179  05  05
180  43 RCL
181  59  59
182  61 GTO
183  01  01
184  42  42
185  93  .
186  04  4
187  85  +
188  93  .
189  01  1
190  95  =
191  61 GTO
192  01  01
193  31  31
```

MR MAGIC

av Robert & Richard Snow
ur TI PPC Notes v5n3p10
översättning Björn Gustavsson

Som alla andra kortkonster, kräver även den här illusionen lite fingerfärdighet. 59:an själv står för fingerfärdigheten. Du talar om att din räknare kan lista ut alla kort. Till de åhörandes förvåning utför räknaren den prestationen med ungefär sex gissningar. Vad det stackars "offret" inte inser är att det är du som sätter räknaren på rätt spår.

När du trycker B, för BÖRJA, skriver skrivaren: READY? PICK A CARD - alltså en uppmaning att välja ett kort. "Offret" plockar upp ett ur en kortlek och lägger det på bordet med rätsidan upp. Titta nu på displayen och lägg märke till att den blinkar omväxlande 9999 och 9999. (med decimalpunkt). När du trycker ned R/S och håller den nere i ungefär en sekund finns det alltid två möjliga utfall, beroende på om niorna följs av en decimalpunkt eller inte. Men det vet bara du!

Ditt offer, även någon som känner till 59:an, lägger inte alltid märke till den lilla skillnaden. Du kan enkelt förutsäga vad skrivaren kommer att skriva, genom att följa, och förhoppningsvis lära utantill, det binära trädet nedan. De stora svarta punkterna anger närvaron av decimalpunkt. Således får du BLACK första gången du trycker R/S med decimalpunkt. (Övers. anm.: På slutet kommer en översättning av engelska korttermer.) Nästa gång blir resultatet SPADE.

Från och med nu betyder decimalpunkt höga kort. Med lite övning kommer du att märka att det är väldigt enkelt att erhålla rätt resultat. Men om du gör ett misstag - och det gör ju alla då och då - misströsta inte! Detta djävulska program har en fantastisk förmåga att släta över fel: Om du gör en tavla, och skriver fel färg eller kort, börja prata snabbt: "Konstigt, jag måste fortfarande kontrollera det här programmet. Ibland skriver det ut motsatsen. Kanske, den... låt se..." Och se, programmet skriver I CHANGED MY MIND ("Jag ändrade mig") och skriver sedan motsatsen till sin gissning!

Den här felbehandlingen äger rum var tionde sekund, och bör för trovärdighetens skull inte missbrukas. Det betyder att du bara har ungefär tio sekunder på dig varje gång för att ge rätt resultat, så öva och sätt fart!

Åhörarna är vanligen förbluffade över hur räknaren klarar av det. De misstänkte något sorts knep förstås, men få lyckas peka ut det. Några av mina offer höll ögonen på mina fötter: de kanske trodde att jag styrde räknaren med fotpedaler! Andra bad mig till och med att inte upprepa högt valören och färgen på kortet. Trodde de verkligen att 59:an kunde höra det?

Bruksanvisning:

1. Tryck B för att starta. READY? PICK A CARD skrivs.
2. Offret väljer ett kort.
3. Tryck ned R/S inom tio sekunder.
4. Tryck A för att fortsätta. Skrivaren skriver räknarens (eller är det ditt?) val.
5. Gå tillbaka till 3 tills hela kortet är avslöjat.

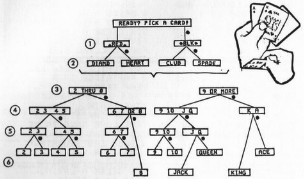
Översättarens not: Programmet har gjorts av de i TI PPC mycket berömda bröderna Snow. Som vanligt är programmet välprogrammerat, med en del knep. Observera t ex skrivkoden i R_{55}. Om ett skrivregister laddas med innehållet med hjälp av OP nn blir resultatet "I ", men laddas det med HIR nn blir resultatet " MIND"! Detta underlättar vid utskrift av I CHANGED MY MIND. Bara tre register behöver återkallas (se steg 012-027).

Det finns också en rutin i programmet så att kortet skall kunnas listas ut med färre antal gissningar. Men då måste man trycka A', och så ser ju åhörarna att man trycker på olika tangenter och förstår direkt att det är något fiffel på gång.

Lars Hedlund föreslog en kanske bättre metod: Ett pekminne räknas upp samtidigt med displayväxlingen. Det pekar på t ex tre register som alla innehåller 9999. (med punkt). I display ser man då en gång 9999 utan punkt och tre gånger 9999. med punkt. Genom att räkna 9999. (med punkt) klarar man alltså fyra valmöjligheter per R/S. Man klarar då rätt färg direkt och valören i tre "gissningar".

Läsarna inbjuds skicka in program som bygger på den principen. Det måste gå att ångra sig även i det programmet. (Skicka programmen till Utmaningen.)

Förklaring av engelska korttermer: Färgerna heter Diamond (förkortat DIAMD), Heart, Club och Spade, vilket alltså motsvarar rutter, hjärter, klöver och spader. Knekt heter Jack, dam heter Queen, kung heter King och ess heter Ace.



121.4571003324	30		11.00012003235	45	
Y? PI			9 OR		
118.15260013	31		11.00030323517	46	
CK A			MORE		
3.007535171675	32		15.00012000201	47	
▲RED▲			9 10		
-25.01624133016	33		15.000250034	48	
DIAMD			J Q		
-25.02317133537	34		30.000260013	50	
HEART			K A		
3.005114272651	35		25131526.	51	
BLK			JACK		
-25.00015274114	36		3441171731.	52	
CLUB			QUEEN		
-25.03633131617	37		26243122.	53	
SPADE			KING		
11.00003003723	38		13151700.	54	
2 TH			ACE		
11.03541000973	39		2400.030243116	55	
RU 8?			I MIND		
15.000030004	40		1523133122.	56	
2 3			CHANG		
15.000050006	41		1716003045.	57	
4 5			ED MY		
15.000070008	42		35171316.	58	
6 7			READ		
15.032350009	43		1513351673.	59	
OR 8			CARD?		

000	76	LBL	048	11	A	096	69	OP	144	95	=
001	66	PAU	049	24	CE	097	10	10	145	44	SUM
002	59	INT	050	69	OP	098	85	+	146	05	05
003	09	9	051	10	10	099	43	RCL	147	18	C'
004	09	9	052	42	STO	100	09	09	148	65	×
005	09	9	053	04	04	101	75	-	149	03	3
006	09	9	054	32	X:T	102	03	3	150	95	=
007	66	PAU	055	09	9	103	08	8	151	17	B'
008	97	DSZ	056	42	STO	104	95	=	152	44	SUM
009	03	03	057	03	03	105	99	PRT	153	05	05
010	00	00	058	43	RCL	106	76	LBL	154	44	SUM
011	01	01	059	05	05	107	12	B	155	05	05
012	43	RCL	060	42	STO	108	25	CLR	156	85	+
013	55	55	061	06	06	109	43	RCL	157	04	4
014	82	HIR	062	43	RCL	110	58	58	158	03	3
015	08	08	063	09	09	111	69	OP	159	75	-
016	69	OP	064	32	X:T	112	01	01	160	67	EQ
017	01	01	065	83	GO*	113	43	RCL	161	01	01
018	43	RCL	066	00	00	114	59	59	162	72	72
019	56	56	067	76	LBL	115	69	OP	163	04	4
020	69	OP	068	18	C'	116	04	04	164	05	5
021	02	02	069	73	RC*	117	42	STO	165	95	=
022	43	RCL	070	09	09	118	03	03	166	17	B'
023	57	57	071	82	HIR	119	03	3	167	44	SUM
024	69	OP	072	06	06	120	00	0	168	05	05
025	03	03	073	01	1	121	42	STO	169	01	1
026	69	OP	074	76	LBL	122	00	00	170	01	1
027	05	05	075	17	B'	123	42	STO	171	32	X:T
028	69	OP	076	44	SUM	124	09	09	172	43	RCL
029	00	00	077	09	09	125	18	C'	173	05	05
030	43	RCL	078	73	RC*	126	65	×	174	22	INV
031	01	01	079	09	09	127	07	7	175	77	GE
032	22	INV	080	82	HIR	128	85	+	176	01	01
033	44	SUM	081	07	07	129	42	STO	177	05	05
034	00	00	082	69	OP	130	05	05	178	04	4
035	43	RCL	083	05	05	131	03	3	179	00	0
036	06	06	084	69	OP	132	08	8	180	44	SUM
037	42	STO	085	00	00	133	95	=	181	05	05
038	05	05	086	44	SUM	134	42	STO	182	73	RC*
039	32	X:T	087	00	00	135	09	09	183	05	05
040	42	STO	088	42	STO	136	18	C'	184	69	OP
041	09	09	089	01	01	137	65	×	185	03	03
042	01	1	090	69	OP	138	02	2	186	69	OP
043	75	-	091	29	29	139	44	SUM	187	05	05
044	43	RCL	092	81	RST	140	05	05	188	12	B
045	04	04	093	76	LBL	141	85	+			
046	54	)	094	16	A'	142	44	SUM			
047	76	LBL	095	24	CE	143	09	09			

■

IDÉBÖRSEN

Anders Perssons idé från förra numret om ett program för att beräkna kostnad för telefonsamtal gjorde många intresserade. Det presenteras därför på annan plats i det här numret. Innan jag fick det gjorde jag ett eget som fordrar besked om markeringsintervall eller minutkostnad. Bruksanvisning: Slå in endera av dessa och när abonnenten svarar, tryck A vid markeringsintervall och B vid minutkostnad. Kostnaden uppdateras var 10:e sekund och visas i fönstret till nästa uppdatering. Då samtalet slutar trycker man R/S. Skulle kostnaden då ej stå i fönstret, tryck E.

000	43	RCL	018	43	RCL	036	81	RST
001	01	01	019	01	01	037	76	LBL
002	66	PAU	020	91	R/S	038	12	B
003	97	DSZ	021	76	LBL	039	58	FIX
004	00	00	022	11	A	040	02	02
005	00	00	023	58	FIX	041	47	CMS
006	00	00	024	02	02	042	55	÷
007	43	RCL	025	47	CMS	043	06	6
008	02	02	026	35	1/X	044	95	=
009	44	SUM	027	65	×	045	42	STO
010	01	01	028	02	2	046	02	02
011	01	1	029	95	=	047	01	1
012	03	3	030	42	STO	048	02	2
013	42	STO	031	02	02	049	42	STO
014	00	00	032	01	1	050	00	00
015	81	RST	033	02	2	051	81	RST
016	76	LBL	034	42	STO			
017	15	E	035	00	00			

Ett annat tips till program finns i TI PPC Notes V6N9/10P28. Jules Bell fann att han kunde spara över en halvtimma per vecka i det stora snabbköp där han handlar. Han skrev en lista på de 100 vanligaste artiklar han köpte och sorterade deras skrivkoder skrivna som decimalbråk (.1314151617 t ex) i bokstavsordning med M/U-modulen. Han spelade in denna lista på magnetkort. Sedan tog han den ordnade listan med sig till snabbköpet och antecknade en tvåsiffrig kod för läget i affären så att nummerordning motsvaras av lämplig gångväg (t ex monter, gång nr). Denna kod adderas sedan till decimalbråket enligt ovan. Han spelade in en omgång magnetkort även den här gången. När man skall gå och handla slår man in registernumret för de varor man vill ha så får man med programmets hjälp en lista med varorna i den ordning de skall plockas i butiken.

Kom gärna med nya idéer så att vi inte behöver lägga ner den här avdelningen p g a idébrist! ■

Twelve days of

CHRISTMAS

julramsa med TI 59

Den här ramsan presenterades i TI PPC Notes till förra julen (V5N9/10P27). Programmet var då skrivet av Bill Skillman. I år har en Fast mode-version av Lem Matteson publicerats (V6N9/10P15). Ca 2000 nedslag på 2 min 12 s! I normalmode startas med A. I Fast mode läses först sida 1 och tryckes RST R/S. Läs därefter alla fyra kortsidorna utan att röra någon tangent. (Standardmodulen isatt förstås.) Granska den snygga programmeringen, t ex hur "subrutiner" används i Fast mode och hur en operation (steg 090-091) kan ligga vilande medan all världens andra operationer pågår.

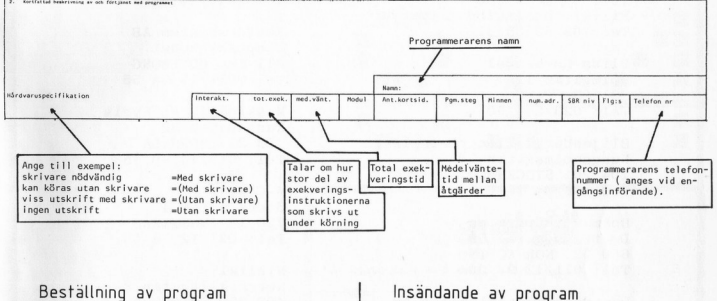
```
  NUMERIC  REG ALPHA    3122570000. 44 NG,
1300331335. 01 A PRR     203002701. 45 I2 LO
3735241622. 02 TRIDG    3516360013. 46 RDS A
1700243100. 03 E IN     2027171323. 47 -LEAP
1300000000. 04 A        2431225700. 48 ING,
3743010037. 05 TWO T    3735411700. 49 TRUE
4135372717. 06 URTLE    2701421700. 50 LOVE
1601421736. 07 DOVES    2213421700. 51 GAVE
5700133116. 08 , AND    3701003017. 52 TO ME
3723351717. 09 THREE    1613450001. 53 DAY O
  21351731. 10  FREN    2100152335. 54 F CHR
1523002317. 11 CH HE    2436373013. 55 ISTMA
3136570000. 12 NS,      3600304500. 56 S MY
2101413500. 13 FOUR     3317133500. 57 PEAR
1513227724. 14 CALLI    3735171740. 58 TREE.
3122001424. 15 NG BI     203001613. 59 12 DA
3513635700. 16 RDS,     4536000121. 60 YS OF
2124421700. 17 FIVE       15233524. 61  CHRI
2201271600. 18 GOLD     3637301336. 62 STMAS
3524312236. 19 RINGS    2124353637. 63 FIRST
5700000000. 20 ,                    64
3624440022. 21 SIX G    3617150131. 65 SECON
1717361700. 22 EESE     1600000000. 66 D
1320271345. 23 A-LAY    3723243516. 67 THIRD
2431225700. 24 ING,                 68
 800364313. 25 7 SWA    2101413537. 69 FOURT
3136001320. 26 NS A-    2300000000. 70 H
3643243030. 27 SWIMM    2124213723. 71 FIFTH
2431225700. 28 ING,                 72
 900301324. 29 8 MAI    3624443723. 73 SIXTH
1636001320. 30 DS A-                74
3024272624. 31 MILKI    3617421731. 75 SEVEN
3122570000. 32 NG,      3723000000. 76 TH
1200163541. 33 9 DRU    1724222337. 77 EIGHT
3030173536. 34 MMERS    2300000000. 78 H
  163541330. 35  DRUM    3124313723. 79 NINTH
3024312257. 36 MING,                80
3717310033. 37 TEN P    3717313723. 81 TENTH
2433173536. 38 IPERS                82
  33243324. 39  PIPI    1727174217. 83 ELEVE
3122570000. 40 NG,      3137230000. 84 NTH
 202002713. 41 11 LA    3743172721. 85 TWELF
1624173600. 42 DIES     3723000000. 86 TH
1613311524. 43 DANCI      37231700. 87  THE
```

```
12 DAYS OF CHRISTMAS

 THE FIRST
DAY OF CHRISTMAS MY
TRUE LOVE GAVE TO ME
A PARTRIDGE IN A
PEAR TREE.

 THE SECOND
DAY OF CHRISTMAS MY
TRUE LOVE GAVE TO ME
TWO TURTLEDOVES, AND
A PARTRIDGE IN A
PEAR TREE.

 THE THIRD
DAY OF CHRISTMAS MY
TRUE LOVE GAVE TO ME
THREE FRENCH HENS,
TWO TURTLEDOVES, AND
A PARTRIDGE IN A
PEAR TREE.

 THE FOURTH
DAY OF CHRISTMAS MY
TRUE LOVE GAVE TO ME
FOUR CALLING BIRDS,
THREE FRENCH HENS,
TWO TURTLEDOVES, AND
A PARTRIDGE IN A
PEAR TREE.

 THE FIFTH
DAY OF CHRISTMAS MY
TRUE LOVE GAVE TO ME
FIVE GOLD RINGS,
FOUR CALLING BIRDS,
THREE FRENCH HENS,
TWO TURTLEDOVES, AND
A PARTRIDGE IN A
PEAR TREE.

 THE SIXTH
DAY OF CHRISTMAS MY
TRUE LOVE GAVE TO ME
SIX GEESE A-LAYING,
FIVE GOLD RINGS,
FOUR CALLING BIRDS,
THREE FRENCH HENS,
TWO TURTLEDOVES, AND
A PARTRIDGE IN A
PEAR TREE.

 THE SEVENTH
DAY OF CHRISTMAS MY
TRUE LOVE GAVE TO ME
7 SWANS A-SWIMMING,
SIX GEESE A-LAYING,
FIVE GOLD RINGS,
FOUR CALLING BIRDS,
THREE FRENCH HENS,
TWO TURTLEDOVES, AND
A PARTRIDGE IN A
PEAR TREE.

 THE EIGHTH
DAY OF CHRISTMAS MY
TRUE LOVE GAVE TO ME
8 MAIDS A-MILKING,
7 SWANS A-SWIMMING,
SIX GEESE A-LAYING,
FIVE GOLD RINGS,
FOUR CALLING BIRDS,
THREE FRENCH HENS,
TWO TURTLEDOVES, AND
A PARTRIDGE IN A
PEAR TREE.

 THE NINTH
DAY OF CHRISTMAS MY
TRUE LOVE GAVE TO ME
9 DRUMMERS DRUMMING,
8 MAIDS A-MILKING,
7 SWANS A-SWIMMING,
SIX GEESE A-LAYING,
FIVE GOLD RINGS,
FOUR CALLING BIRDS,
THREE FRENCH HENS,
TWO TURTLEDOVES, AND
A PARTRIDGE IN A
PEAR TREE.

 THE TENTH
DAY OF CHRISTMAS MY
TRUE LOVE GAVE TO ME
TEN PIPERS PIPING,
9 DRUMMERS DRUMMING,
8 MAIDS A-MILKING,
7 SWANS A-SWIMMING,
SIX GEESE A-LAYING,
FIVE GOLD RINGS,
FOUR CALLING BIRDS,
THREE FRENCH HENS,
TWO TURTLEDOVES, AND
A PARTRIDGE IN A
PEAR TREE.

 THE ELEVENTH
DAY OF CHRISTMAS MY
TRUE LOVE GAVE TO ME
11 LADIES DANCING,
TEN PIPERS PIPING,
9 DRUMMERS DRUMMING,
8 MAIDS A-MILKING,
7 SWANS A-SWIMMING,
SIX GEESE A-LAYING,
FIVE GOLD RINGS,
FOUR CALLING BIRDS,
THREE FRENCH HENS,
TWO TURTLEDOVES, AND
A PARTRIDGE IN A
PEAR TREE.

 THE TWELFTH
DAY OF CHRISTMAS MY
TRUE LOVE GAVE TO ME
12 LORDS A-LEAPING,
11 LADIES DANCING,
TEN PIPERS PIPING,
9 DRUMMERS DRUMMING,
8 MAIDS A-MILKING,
7 SWANS A-SWIMMING,
SIX GEESE A-LAYING,
FIVE GOLD RINGS,
FOUR CALLING BIRDS,
THREE FRENCH HENS,
TWO TURTLEDOVES, AND
A PARTRIDGE IN A
PEAR TREE.
```

```
000  00  0      040  61 GTO    080  30  30      120  02  02
001  00  0      041  01  01    081  69 DP       121  69 DP
002  00  0      042  30  30    082  30  30      122  05  05
003  00  0      043  98 ADV    083  09  9       123  98 ADV
004  00  0      044  01  1     084  00  0       124  97 DSZ
005  36 PGM     045  02  2     085  42 STO      125  09  09
006  02  02     046  42 STO    086  99  99      126  00  00
007  71 SBR     047  89  89    087  61 GTO      127  48  48
008  02  02     048  01  1     088  01  01      128  25 CLR
009  39  39     049  44 SUM    089  30  30      129  91 R/S
010  00  0      050  88  88    090  04  4       130  73 RC*
011  00  0      051  73 RC*    091  85  +       131  00  00
012  00  0      052  88  88    092  42 STO      132  69 DP
013  00  0      053  69 DP     093  00  00      133  30  30
014  00  0      054  00  00    094  01  1       134  69 DP
015  00  0      055  69 DP     095  00  0       135  04  04
016  22 INV     056  02  02    096  02  2       136  73 RC*
017  58 FIX     057  01  1     097  42 STO      137  00  00
018  25 CLR     058  44 SUM    098  99  99      138  69 DP
019  91 R/S     059  88  88    099  61  GTO     139  30  30
020  25 CLR     060  73 RC*    100  01  01      140  69 DP
021  91 R/S     061  88  88    101  30  30      141  03  03
022  25 CLR     062  69 DP     102  02  02      142  73 RC*
023  91 R/S     063  03  03    103  00  0       143  00  00
024  76 LBL     064  43 RCL    104  07  7       144  69 DP
025  11  A      065  87  87    105  42 STO      145  30  30
026  16 1/X     066  69 DP     106  99  99      146  69 DP
027  00  0      067  01  01    107  97 DSZ      147  02  02
028  69 DP      068  69 DP     108  00  00      148  73 RC*
029  17  17     069  05  05    109  01  01      149  00  00
030  06  6      070  05  5     110  30  30      150  69 DP
031  02  2      071  06  6     111  69 DP      151  01  01
032  42 STO     072  42 STO    112  00  00      152  69 DP
033  00  00     073  00  00    113  43 RCL     153  05  05
034  42 STO     074  08  8     114  57  57      154  83 GD+
035  88  88     075  01  1     115  69 DP      155  99  99
036  04  4      076  42 STO    116  01  01
037  03  3      077  99  99    117  43 RCL
038  42 STO     078  61 GTO    118  58  58
039  99  99     079  01  01    119  69 DP
```


PROGRAMBIBLIOTEKET

Nedanstående mall kan rekvireras från föreningen.
Använd den då Du skickar in program!



Beställning av program

Beställning av program sker genom inbetalning av aktuellt belopp till föreningens postgiro 430 01 59 - 3. Utom programmens pris betalas för varje hel beställning en beställningsavgift på 10 kr.
Två alternativ finns för svenska program till 58/59:
med magnetkort 30 kr
utan magnetkort 15 kr. } + best.avg. 10 kr
För övriga program (utom de tyska i nr 81-2) finns ej magnetkort och enhetspriset är 15 kr.
De tyska kostar 25 kr utan magnetkort.
I samtliga fall tillkommer beställningsavgift på 10 kr per beställning.
Endast svenska och engelska program lagerförs. Övriga beställes av klubben till den utländska klubben en gång per månad, och det kan alltså bli viss leveranstid.

Komplett programbibliotek med beskrivning av programmen enligt mall kan beställas från föreningen genom insättande ev 10 kr på postgiro 430 01 59 - 3.

Insändande av program

1. Programmet skänks till Föreningen Programbiten.
   För varje program Du skänker har Du rätt att välja 2 valfria program ur Programbiblioteket på magnetkort eller 3 st utan magnetkort.
   (Valda program får ej stå under rubriken Engångsinförande.)
   Mall för insändande av program kan gratis erhållas från föreningen.
2. Engångsinförande.
   Om Du önskar marknadsföra Dina privata program kan Du göra det under denna rubrik. Programbeskrivning (enligt mall) införs en gång i Programbiblioteket, utan kostnad för Dig. Du svarar för att leverans sker till eventuella beställare.

Program till Programbiblioteket insändes till Programförmedlare Bo Nordlin
Stora Björnens gata 70
136 64 HANDEN

Rekommenderade fackhandlare

Under den här rubriken låter vi de återförsäljare som säljer Texas Instruments och som är medlemmar i föreningen annonsera:

Karl D. Björkmans AB
Stensätravägen 5-7
127 03 SKÄRHOLMEN
Tel: 08/97 95 50
Filial: Humlegårdsgatan 22
Tel: 08/62 65 81

Blids Bokhandel
Holmgatan 11
791 23 FALUN
Tel: 023/280 50

Blijenburgh Electronics
Kungsholmsgatan 20
104 20 STOCKHOLM
Tel: 08/54 18 75

Bokman i Spiralen
Drottninggatan 59
600 02 NORRKÖPING
Tel: 011/12 22 00

Dawidssons Maskinaffär AB
Aspholmsvägen 1
702 27 ÖREBEO
Filial: Stortorget 8
Tel: 019/13 64 50

Elikon
Regeringsgatan 30
111 53 STOCKHOLM
Tel: 08/21 93 00

Gumperts Universitetsbokhandel
Norra Hamngatan 26
401 25 GÖTEBORG
Tel: 031/23 54 80
Filial:
Västra Storgatan 16
552 55 JÖNKÖPING
Tel: 036/11 92 40

Kontorskonsult AB
Ågatan 23
581 02 LINKÖPING
Tel: 013/13 01 75

Kontorssystem AB
Skeppsbrohuset
411 18 GÖTEBORG
Tel: 031/17 44 55

Maskinaffären Fyris
Kungsgatan 32
753 21 UPPSALA
Tel: 018/14 90 15

M-Center
Stora Gatan 4
720 13 VÄSTERÅS
Tel: 021/12 38 00

Minital
Norra Allégatan 8
402 32 GÖTEBORG
Tel: 031/11 01 54

Wettergrens Bokhandel
V:a Hamngatan 22
Avenyn 21
Vasagatan 22
Frölunda Torg
Wieselgrensplatsen
GÖTEBORG
Tel: 031/17 00 90
Kungsmässan
KUNGSBACKA
Tel: 0300/160 90

AB Montins Bokhandel
Hovrättsesplanaden 15
65 100 VASA 10
FINLAND
Tel: 961/24 25 00

K-TRYCK AB, STOCKHOLM 1981

RÄTTELSE

till "BRIDGE-GIVAR" på s. 17.

Listningen av steg 400 - 447 som motsvarar registren 64 - 69 är delvis fel. Registren ändras nämligen vid körning av programmet och den listning som återgavs gjordes efter körning. Till höger ges rätt listning som motsvarar registerinnehållet i texten.

400	14	D	424	14	D
401	02	2	425	00	0
402	13	C	426	00	0
403	26	2ND	427	00	0
404	34	fX	428	80	GRD
405	25	CLR	429	41	SST
406	37	P/R	430	08	8
407	12	B	431	14	D
408	10	E'	432	90	LST
409	00	0	433	00	0
410	00	0	434	00	0
411	00	0	435	00	0
412	42	STO	436	13	C
413	36	36	437	27	INV
414	32	X:T	438	27	INV
415	31	LRN	439	13	C
416	14	D	440	30	TAN
417	00	0	441	00	0
418	00	0	442	00	0
419	00	0	443	00	0
420	41	SST	444	42	STO
421	08	8	445	32	32
422	14	D	446	36	PGM
423	80	GRD	447	31	31

Årsmöte Medlemsträff

(se s. 2)

Gunnar Svanborg från Texas Instruments kommer och visar den nya versionen av hemdatorn TI-99/4 A med tillbehör.

Vidare kan det tänkas att även någon nyhet på räknarsidan kan visas!

Tack till

FOLKE JOHANSON INGENJÖRSBYRÅ AB
för att vi fått utnyttja lokaler och kontorsutrustning vid montering av tryckoriginal

och till
K-TRYCK, JOHANNESHOV
för ett utomordentligt gott och snabbt arbete!