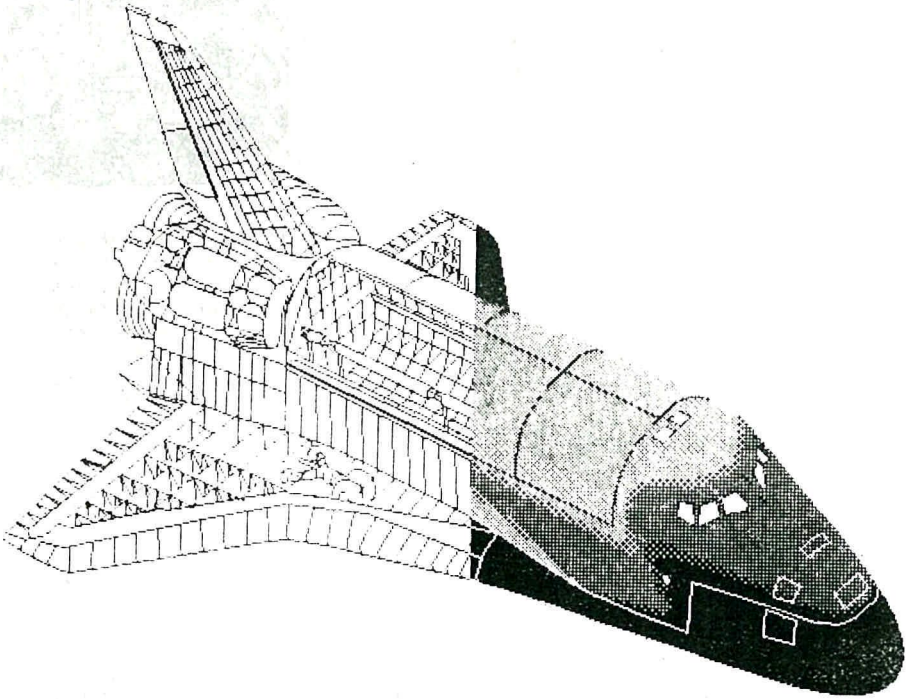


STACKPOINTER
NR 1 - 81
HUSORGAN FÖR
DATOR-
FÖRENINGEN
STACKEN

ORDFÖRANDEN HAR ORDET

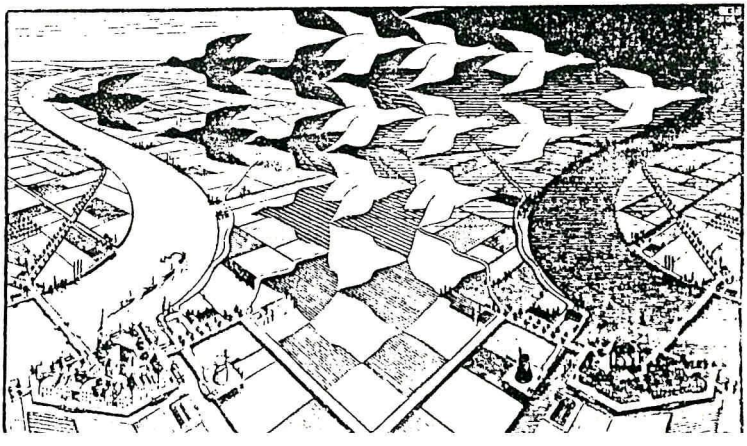
Så var det dags för ett nytt nummer av STACKPOINTER. Den här gången blev det mer teoretiskt nummer är förut. Det ska bli intressant att se om trenden håller i sig. Det är faktiskt på dig, kære läsare, det beror. Det är nämligen dina manus som utgör stackpointer. Så fatta tangentbordet (eller pennan!) och s-k-r-i-v! Det går t.ex. utmärkt att lämna manus i form av en textfil på NADJA. Naturligtvis tar dock vi emot manus i vilken form som helst! Vad ska då lilla JAG skriva om, frågar du dig kanske? Svar: vad du sysslar med och tycker är intressant! Chansen att andra likasinnade i STACKEN också ska finna det intressant är större än du tror!

VERKSAMHETEN

Verksamheten över årsslutet har väl varit lägre än normalt, kanske beroende på på att jultraditionerna kommit mellan. Någon gång i februari börjar det dock hända saker: då blir det dags att inviga vår nya lokal! En tilldragelse vi ser fram emot. Kaffe/te-bryggare är redan undanställd, kylskåp har kassörn sagt att vi har råd med, tidskrifter har Dag Rende lovat ordna, möbler kan vi få, (fast om det är någon som har en skön fåtölj eller soffa, så...). Sedan kanske vi vågar ta emot ett officiellt besök av vår systerorganisation LYSATOR i Linköping!

HEDERSMEDLEMMAR

STACKEN har fått sin första hedersmedlem! Till första hedersmedlem i STACKENS korta historia valdes på mötet den 4 december Lars S. Ljungdahl till hedersmedlem. Mötets motivering var hans insatser för föreningen, främst införskaffandet av vår egen SEVEN-S dator.



FÖRSLAG

Alla större organisationer har ju en förslagsbrevlåda. Vi ska väl inte vara sämre, och uppmanar härmed alla medlemmar att komma in med FÖRSLAG till verksamhet. Det är ju trots allt vi själva som avgör hur STACKENS verksamhet ska se ut. För att föregå med gott exempel och sätta fart på ide´kläckandet föreslår jag själv att vi anordnar ett semiarium eller diskussionsafton om datorkonst, något som jag tror borde intressera ett flertal. Vi skulle också vilja se fler projekt av AMIS´ karaktär, där flera STACKERS går samman och skriver 1) ihop något tillsammans. Vi har t.ex. fått tillgång till något som på NADJA som kallas STAGE2 och lär vara en macro-processor som enligt rykten skall innehålla bl.a. en Z80-assembler...

NYA MEDLEMMAR

Det är roligt att se att det strömmar till nya medlemmer till STACKEN. Till nykomlingarna vill jag framföra ett "välkommen i gänget", och be om en smula tålamod när det gäller ev. bekräftelser på medlemsskap. Det har varit lite si och så med uppdateringen av registret. En annan sak som bör nämnas när det gäller nya medlemmar gäller STACKENS verksamhet och spel. På LYSATOR har man haft problem med personer som vill bli medlemmar bara för att få tillgång till det ena eller andra spelprogrammet. Där har man blivit tvungna att ta till drastiska åtgärder för att stoppa sånt. Så i STACKEN gäller numera den oskrivna regeln, att spel är till för att SKRIVAS, men helst inte KÖRAS.

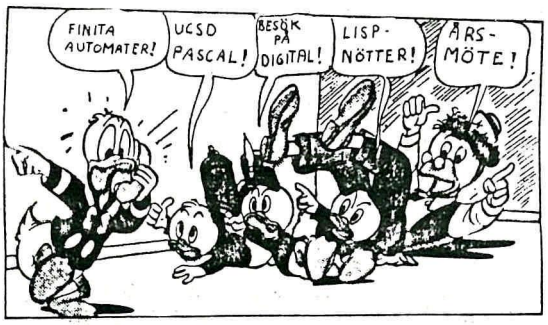
NY ADRESS

Vi har också fått en ny adress till föreningen, när det gäller saker av icke-ekonomisk karaktär. Ett postfack har installerats på NADA 2) med adressen

Datorföreningen STACKEN
c/o NADA, KTH
100 44 Stockholm

När det gäller medlemsavgifter betalas dessa fortfarande på vårat postgiro nummer 433 01 15 - 9, och för post som ska nå vår KASSÖR, använd vår BOX-adress: Box 5079, 141 05 Huddinge.

Computare necesse est! /Perre L.



1) eller skruvar...

2) förkortning för **N**umerisk **A**nalys och **DA**talogi

LITE BERÄKNINGSBARHETSTEORI

Teori, det är svårt det, brukar man ju tycka, speciellt när den är abstrakt. Trots det så tror jag att STACKENs medlemmar kan behöva litet av den varan att lyfta själen med. Därför skall jag nu helt kort tala om Datorprogrammens Matematiska Grundvalar[1]. Jag har försökt undvika att fylla artikeln med en massa avancerad matematik, så du kan lugnt läsa på i stället för att bläddra till nästa artikel (som antagligen handlar om Zilog-68000, kontaktstudsare eller annat konkret larv). För de som inser skönheten i och uppskattar uppbyggligheten hos abstrakta matematiska övningar så har jag ett komplett bevis i slutet av artikeln. De som inte gör detta har förhoppningsvis fått ett ökat intresse för ämnet när de läst färdigt.

Man brukar ju säga att den matematiska grunden till datamaskiner utgörs av logik och Boolesk algebra. Alla har vi väl läst om hur man med hjälp av enkla grindar kan bygga upp komplexa kretsar som kan addera, multiplicera, avkoda adresser, implementera virtuellt minne och annat. Eftersom detta endast berör hälften av en dator[2], hårdvaran, så undrar man kanske vad som är den matematiska grundvalen för programmen, mjukvaran.

Efter en stunds funderande kommer man kanske fram till att det som är grundläggande för mjukvaran är beräkningen. Ett program ges viss indata och efter att ha beräknat en stund så presenterar det viss utdata. En matematisk teori för program borde alltså starta med att tala om vad en beräkning är.

På 1930-talet (alltså långt innan det fanns några datorer) arbetade den engelske matematikern Alan Turing på att formalisera (det vill säga exakt beskriva) vad som menas med en beräkning. Hans arbete ledde fram till den så kallade Turingmaskinen (TM), en matematisk "maskin" som, med lämplig programmering, kunde fås att beskriva en beräkning. Turing påstod att hans maskin kunde beskriva alla beräkningar som överhuvud taget gick att utföra. Det skulle alltså inte finnas några beräkningar, vare sig av människa eller maskin, som Turingmaskinen inte skulle kunna efterlikna. Detta påstående brukar kallas Turings Tes.

Man kan visa att en traditionell datamaskin (med godtyckligt mycket minne) kan utföra precis samma beräkningar som en TM kan. Mot varje datamaskinprogram svarar alltså ett Turingmaskinprogram och tvärtom - detta gör att man kan använda Turingmaskiner för att visa egenskaper hos vanliga datorprogram. För att uttrycka att datorer och TM kan beräkna samma sak brukar man säga att en dator "är

---

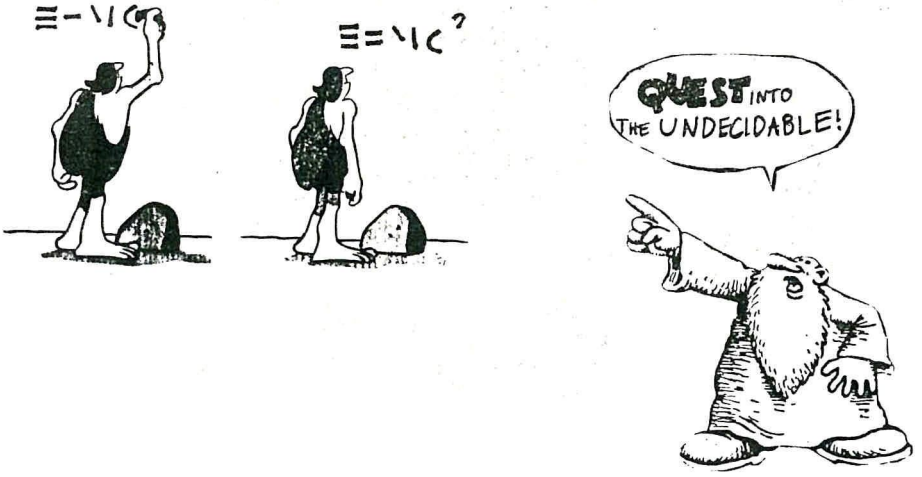
[1] Om någon expert på rekursionsteori eller närliggande områden skulle läsa detta så ber jag i förväg om ursäkt för det sätt, på vilket jag misshandlat ämnet.

[2] ..den mindre viktiga halvan.

ekvivalent med" en TM, eller till och med att den "är" en TM.

Turings Tes väckte så småningom stor uppmärksamhet, eftersom det visade sig finnas saker som man tyckte skulle gå att beräkna, men i själva verket inte gör det. Det finns alltså problem som vi aldrig på något tänkbart sätt, ens i teorin, kan få lösningen till, även om en lösning finns! Dessa problem brukar kallas "oberäkningsbara" eller "oavgörbara". Analogt kallas sådana problem som man kan beräkna en lösning till för "avgörbara", "beräkningsbara" eller "Turing-beräkningsbara".

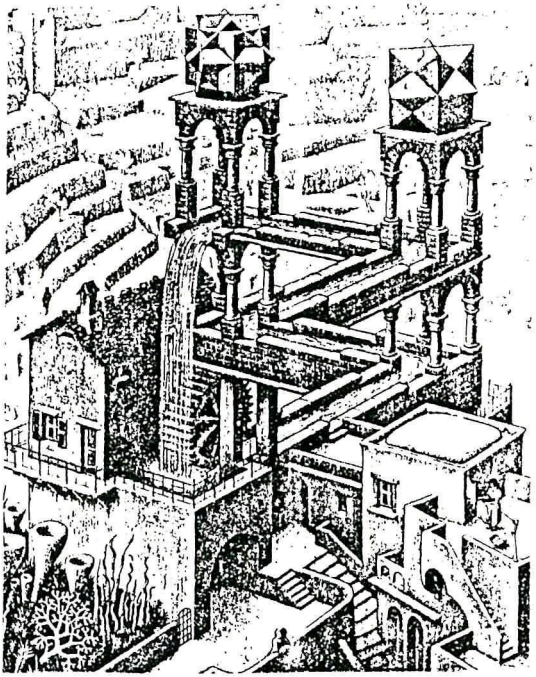
Ett av de mest kända av de "oberäkningsbara" problemen är Stopproblemet för Turingmaskiner. Precis som en beräkning som en dator gör kan pågå i oändlighet utan att någonsin ta slut (om programmet loopar, alltså), så kan det också finnas "loopar" i Turingmaskiner som aldrig tar slut. I datorprogrammering skulle det förstås vara av stort värde att kunna avgöra i förväg, genom att undersöka ett program, ifall det skulle komma att loopa eller inte. Både datortid och programmerarmöda skulle kunna sparas genom att loopar kunde hittas utan programtestning. Nu kan man visa (det gör jag formellt vid artikelns slut) att det inte finns något sätt att beräkna om en TM stannar (och alltså inte heller ifall ett datorprogram gör det).



Andra exempel på oavgörbara problem är t.ex. huruvida två olika program räknar ut samma saker från samma indata eller ifall en logisk sats är sann eller falsk. Naturligtvis kan man ofta hitta ett svar på dessa frågor, men det finns alltid situationer när man inte kan det och - värst av allt - ibland vet man inte ens om det är möjligt att finna en lösning eller inte. Det kan alltså till och med vara oavgörbart om ett problem är avgörbart eller inte.

Turings Tes har också tagits som argument både för och emot artificiell intelligens. Motståndare till AI har ansett att Turings Tes inte gäller människor och att det därför finns saker som människor kan göra som maskiner inte kan. På motsatt sätt har förespråkare för AI hävdat att eftersom Turings Tes klart säger att både människor och maskiner har samma begränsingar så är verklig artificiell intelligens möjlig.

Många andra matematiker har försökt hitta på egna formaliseringar av beräkningar, t.ex. Generellt Rekursiva Funktioner (GRF), Markovalgoritmer och LAMBDA-kalkylen (vilken som bekant utgör grunden till LISP). När man jämfört dessa med Turingmaskinen så har det visat sig att alla dessa modeller kan beskriva exakt samma beräkningar som en TM kan, varken mer eller mindre. Att flera olika forskare på olika sätt har kommit till exakt samma resultat är det viktigaste tecknet på att Turings Tes är riktig.



MATEMATISK MASKIN

Slutligen kan jag nämna att det finns andra typer av matematiska maskiner som är mindre kraftfulla än Turingmaskiner. En av dessa är så kallade "Finite-State Machines" (FSM). Den viktigaste skillnaden mellan en TM och en FSM är att den senare har ändligt minne, medan en TM har oändligt. (Eftersom en dator också har ändligt minne, är en dator egentligen bara en FSM och inte en TM. I de flesta praktiska sammanhang har dock datorn så mycket minne att man kan betrakta den som en TM, fast den inte är det.)

Matematisk avslutning:

Bevis för att stopproblemet för Turingmaskiner är oavgörbart

I det följande betraktar vi Turingmaskiner som funktioner från mängden av de naturliga talen, N, till N. Eftersom både argumentet och värdet kan betraktas som en Gödelkodning[3] av något element tillhörande en annan uppräknerlig mängd[4], utgör detta inget principiellt hinder.

Man kan visa att det finns en (beräkningsbar) uppräkning av alla Turingmaskiner så att vi kan ange en godtycklig TM som f_n för något heltal n. (Detta kan ske t.ex. genom att låta n vara ett Gödeltal för f_n.)

Vi antar nu existensen av en beräkningsbar funktion h(n,x) sådan att om Turingmaskinen f_n stannar när den ges indata x (d.v.s. om $f_n(x)$ stannar), så har h värdet 0, annars 1.

Vi definierar nu funktionen d(M) på följande sätt:

```
d(M) =    (h(M,M) = 0 -> (L: goto L);
            annars -> 1).
```

Här skall "(L: goto L)" tolkas som att beräkningen d(M) går i en oändlig loop, d.v.s. aldrig stannar. d(M) stannar alltså inte ifall beräkningen $f_M(M)$ gör det. Ifall $f_M(M)$ inte stannar, stannar d(M) (och får värdet 1). Eftersom h var beräkningsbar (antagandet) så kan man visa att även d är det. Det finns alltså ett heltal D sådant att $f_D = d$).

Vi undersöker nu resultatet av beräkningen d(D)! Definitionen av d säger att "d(D) inte stannar ifall $f_D(D)$ stannar" och "d(D) stannar ifall $f_D(D)$ inte gör det). Eftersom $f_D(D) = d(D)$, har vi alltså kommit till resultatet att d(D) stannar precis när den inte gör det och tvärtom! Denna motsägelse visar att vårt första antagande, d.v.s. att h är beräkningsbar, var felaktigt.

Slutsats: det existerar ingen beräkningsbar funktion som kan avgöra ifall en godtycklig Turingmaskin stannar.

/Lars-Henrik Eriksson

[3]Gödelkodning är en metod att tilldela ett objekt en kod, ett Gödeltal, på liknande sätt som ett datorprogram kan ges en kod bestående av alla bitar som bygger upp programmet, tolkade som ett heltal.

[4]En uppräknerlig mängd är en mängd där elementen kan numreras. T.ex. är heltalen och de rationella talen uppräknerliga, men inte de reella talen.

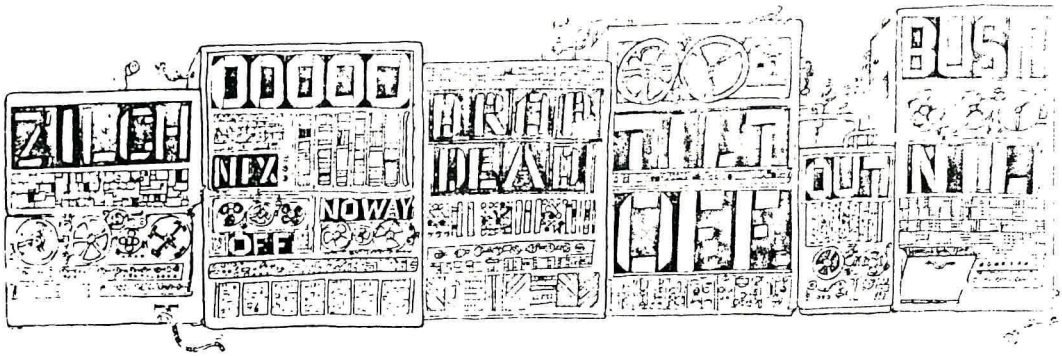
Referenser

Den, vars intresse skulle ha väckts av denna artikel, kan läsa mera om ämnet i (t.ex.) någon av följande böcker:

The Theory of Computer Science, J.M. Brady, Chapman and Hall, London 1977.
En omfattande framställning av olika områden i program- och programmeringsteori.

Gödel, Escher, Bach; Douglas R. Hofstaedter, Basic Books, New York.
Om du inte redan har köpt/lånat denna underbara bok så bör du göra det snarast! Hofstaedter ger en lättfattlig och underhållande introduktion till J.S. Bachs musik, beräkningsbarhetsteori, M.C. Eschers konst, Gödels Bevis, Artificiell Intelligens och mycket annat. GEB har varit en bestseller i USA, vilket är mycket ovanligt för ett populärvetenskapligt verk. Den har också blivit föremål för en veritabel kult.

Algoritmteori, Yngve Sundblad, Studentlitteratur, Lund.
Inledning till området skriven på svenska.



Trancendence

K A L L E L S E

TILL ÅRSMÖTE I DATORFÖRENINGEN **STACKEN** TORS 2 APRIL 1981
KLOCKAN 19:00 I SAL E7 PÅ KGL. TEKNISKA HÖGSKOLAN.

Förslag till dagordning

£ 1. Mötet öppnas
£ 2. Val av 2 personer att jämte ordförande justera mötesprotokollet. Justeringsmännen skall tillika tjänstgöra som rösträknare under mötet.
£ 3. Val av ordförande för mötet.
£ 4. Val av sekreterare för mötet.
£ 5. Tillkännagivande av vid mötet uppgjord röstlängd.
£ 6. Frågan om mötet är stadgeenligt utlyst.
£ 7. Frågan om dagordningens godkännande.
£ 8. Framläggande av kassaberättelse & styrelseberättelse.
£ 9. Framläggande av revisionsberättelse.
£10. Frågan om styrelsens ansvarsfrihet under det gångna arbetsåret.
£11. Val av styrelsemedlemmar:
- Ordförande
- Vice Ordförande
- Sekreterare
- Kassör
- Hexmästare
- Suppleant
- Redacteur för STACKPOINTER

samt revisorer med suppleant.
£12. Val av ledamöter till styrelsevalberedningen.
£13. Behandling av inkomna motioner (utgår p.g.a. brist på sådana).
£14. Behandling av styrelsens förslag:
14.1 Föreningens medlemmar indelas i 2 kategorier förutom hedersmedlemmar: 1. Aktiva medlemmar
2. Externmedlemmar
Där "externmedlem" motsvarar nuvarande "medlem", och "aktiv medlem" är medlem som på något sätt aktivt bidrager till föreningens verksamhet. För aktiv medlem föreslås en sänkt medlemsavgift.
14.2 Inköp av utrustning till STACKENs lokal.
14.3 Mötesdagar: 1:a torsdagen i varje månad, dock ej i Juli och Augusti 1981 eller Januari 1982.
£15. Behandling av budget för innevarande år.
£16. Fastställande av medlemsavgifter för 1982.
£17. Synpunkter på verksamheten för innevarnade år & övriga frågor.
£18. Mötets avslutande.

* * * * * **VÄLKOMNA!** * * * * *

BESÖKET PÅ digital

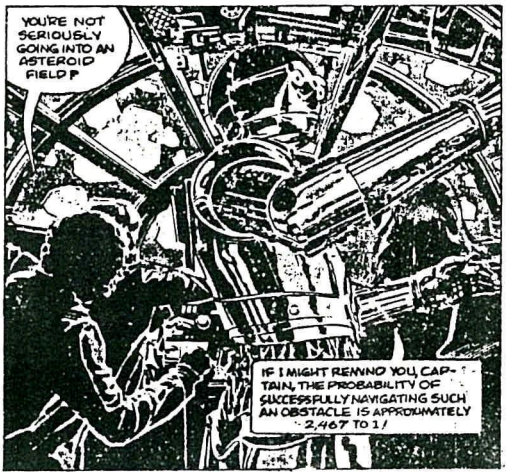
Den 20 November var Stacken på besök hos Digital Equipment (DEC) i Solna. Ett 30-tal personer från Stacken och Lysator deltog, fler än vid något tidigare studiebesök som Stacken gjort. Vi togs emot av några försäljare på Digital, som visade lite intressanta siffror om hur många anställda företaget har (och så vidare), och tog oss sedan med på en rundvandring.

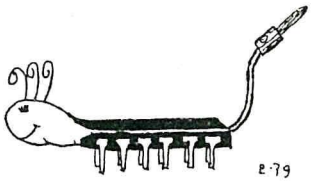
Rundvandring

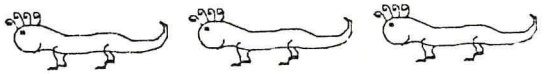
Dom visade oss Digitals experiment med text-TV, några exempel på grafik med matrisskrivare på PDP-11: bl.a. ett program med vars hjälp man kunde definiera tecken och layout för tangentbord, som sedan ritades med en matrisskrivare. (På väggen i samma rum satt en enorm karta över Zork, The Great Underground Empire.) Innanmätet på en VAX-11 visades, Digitals senaste underverk med 32 bits adressbuss, och med en liten låda som kan ringa upp USA och berätta om sina eventuella fel och krämpor.

Digitalt manualspråk

Efter rundvandringen blev vi bjudna på öl och mackor under en liten fråge/ pratstund. Vi fick bl.a. höra om DECs projekt "Digital English", en definition av en delmängd av det engelska språket, i vilket alla manualer skall skrivas i framtiden. På så sätt hoppas man att manualerna skall bli mer exakta och lättförståeliga. 1)







1) Invändningar om att detta är ett sätt att utarma språket kom från några personer. Våra värdar parerade med att en manual inte precis skrivs i skönlitterärt syfte, utan det är tvärtom en fördel att den är lättbegriplig.

LSI-11

Fastän sortimentet nu innehåller datorer av alla storlekar, så är ju Digital känt som företaget som uppfann minidatorn, och minidatorn PDP-11 tycks vara deras populäraste produkt. Vi fick titta och känna på LSI-11 (som den minsta PDP-11an heter) -kort, som fick vandra runt bland publiken. (DEC tog en risk där, men alla korten kom tillbaka...) Några initiativrika Stacken-medlemmar försökte tjata till sej ett PDP-11 system för Stackens räkning, och vi väntar med spänning på att få se om deras ansträngningar var till någon nytta.

/Björn Danielsson (LISP HACKER)

JAG SA INTE ATT GUSTAV VAR PÅ DEKIS, JAG SA ATT HAN VAR MED I DECUS.

STACKPOINTER

Är organ för datorföreningen **STACKEN.**

Redaktör: Björn Ahlén
Ansv. utgivare: Per Lindberg
I Redaktionen: Per Lindberg,
Lars-Henrik Eriksson, Björn Danielsson

Medlemskap i STACKEN kostar 70 SEK för 1981, och betalas in på pg 433 01 15 - 9.

STACKENS LISP-NÖTTER

Här får du en chans att visa din skicklighet i LISP-programmering, högre skolan! De tre uppgifter som följer, den ena värre än den andra, kan du prova att lösa. Om du lämnar dina lösningar till undertecknad, så skall jag bedöma dem och publicera de vackraste exemplaren i nästa nummer av STACKPOINTER.

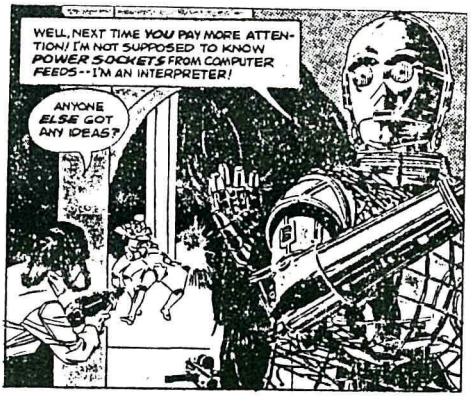
Alla problemen försigår helt och hållet i Pure Lisp, det vill säga endast funktionerna ATOM, EQ, NULL, CAR, CDR, CONS, COND och QUOTE samt funktionsformerna LAMBDA och LABEL är tillåtna[1] . För att göra problemen mera intressanta (läs: svåra) så får man dessutom inte heller använda LABEL.

En sak till: när man löser problemen så får man inte använda hjälpfunktioner!

Om någon skulle ha sett/hört mig tala om lösningen till någon av dessa problem tidigare så är han (hon finns tyvärr inga i Stacken, eller?) naturligtvis diskvalificerad!

Ha det så trevligt!

/Lars-Henrik Eriksson



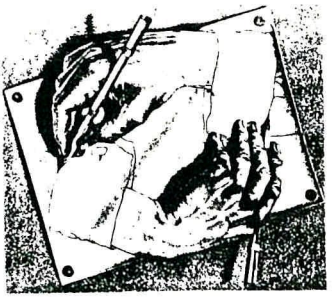
[1]Skall man vara riktigt puristisk (och det skall man ju egentligen) så gäller ytterligare restriktioner på Pure Lisp, men dessa behöver vi inte ta upp här.

Och här kommer Problemen:

1. Skriv ett LISP-uttryck (ej funktion, alltså) som evaluerar till sig självt! Snygga lösningar kommer att premieras. Uttryck av typen "1", "-4711" evaluerar visserligen till sig själva, men gillas inte!

2. Skriv en funktion som gör samma sak som funktionen REVERSE, utan att använda några LAMBDA-uttryck inne i funktionen och, som sagt, utan hjälpfunktioner! (Jo, det går i alla fall!)

3. Skriv en funktion som gör samma sak som funktionen MEMQ (samma som MEMBER, men använder EQ i stället för EQUAL i jämförelserna). Utan att den använder sitt eget namn för rekursionen! (Det skall bli ett s.k. LAMBDA-namn för MEMQ, alltså.) Som nämnts ovan får man inte heller använda hjälpfunktioner eller LABEL för att få till rekursionen! (Jo, detta går faktiskt också!)

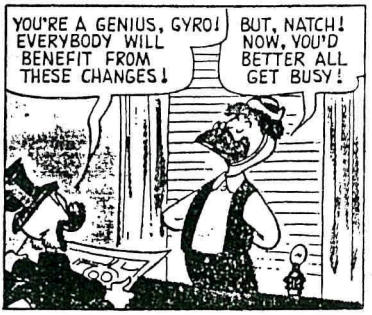


P.S.

Extrafråga: på sidan två i förra numret av STACKPOINTER fattas två parentespar och två quotar i Uppfinnar-Jockes program. Kan någon tala om var dessa skall sättas in?

UCSD PASCAL

För den som till äventyrs inte känner till så mycket om UCSD PASCAL, vill jag gärna berätta lite om detta eminenta mjukvarusystem.



Pascal

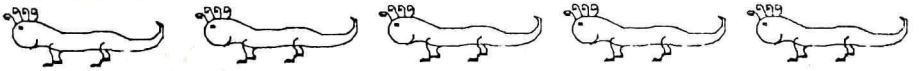
Att Pascal är ett jättebra programspråk, ja det visste väl de flesta redan tidigare. I Pascal slipper man alla eländigheter med de vanliga ostrukturerade språken som BASIC och F-N. I Pascal har man med blockstruktureringsfaciliteterna möjlighet att utan besvär bygga sina program så att det klart och tydligt framgår vad de gör. Ja, man får faktiskt ANSTRÄNGA sig för att göra ett Pascal-program oläsligt, vilket inte är någon konst alls i t.ex. BASIC, det är bara att gröta på som vanligt med en massa GOTO, och ve den som senare försöker fatta vad programmet gör! I Pascal finns även en hel klase datatyper som inte återfinns i t.ex. BASIC. Pekare, mängder, skalära typer, boolska variabler, köer och sammansatta typer, för att nu nämna några.

UCSD PASCAL

På UCSD 1) skrevs under 1978-1979 ett Pascal-system i syfte att ge BASIC och F-N lite konkurrens ute i industrin. I strategin ingick även att den skulle gå att köra på en mikrodator. För att åstadkomma detta, gjorde man inte bara en Pascal-kompilator. Ånej, man gjorde ett komplett OS (OperativSystem) med filhantering, kompilator, länkare, biblioteksrutiner, editorer och så vidare. Och det allra starkaste knepet var, att man inte skrev systemet för någon existerande dator, utan en som man hittat på helt själva, och som inte ens existerade i sinnevärlden! Maskinen i fråga kallas P-maskin 2) och exekverar "P-kod".

P-maskinen

Detta till synes helkorkade påhitt visade sig vara ett genidrag! Ty nu skrev man en INTERPRETATOR för olika existerande processorer som låtsades köra p-kod. Och simsalabim, hela systemet gick att köra på en massa olika maskiner! Eftersom det hela utvecklades på en LSI-11, var den första p-kods interpretatorn på LSI-11:an, men snart fanns också interpretatorer till 8080, Z80, 6502, 6800, och så vidare.

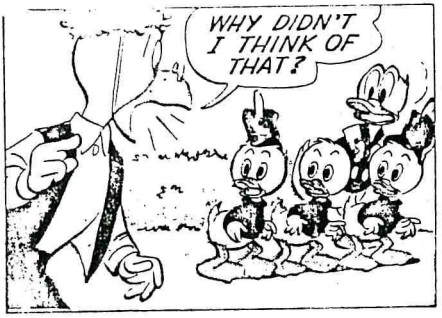


Ett universellt system

Nu fanns ett avancerat högnivå programmeringsSYSTEM till nästan alla mikrodatorer, och intresset för USCD PASCAL blev därefter. Nu, ett par år efteråt, har UCSD PASCAL blivit något av en industristandard, så spritt är det. Nuförtiden säljs det inte längre direkt av UCSD, utan av SofTech, som fått överta distributionen och underhållet av hela paketet. 3)

Något för dig?

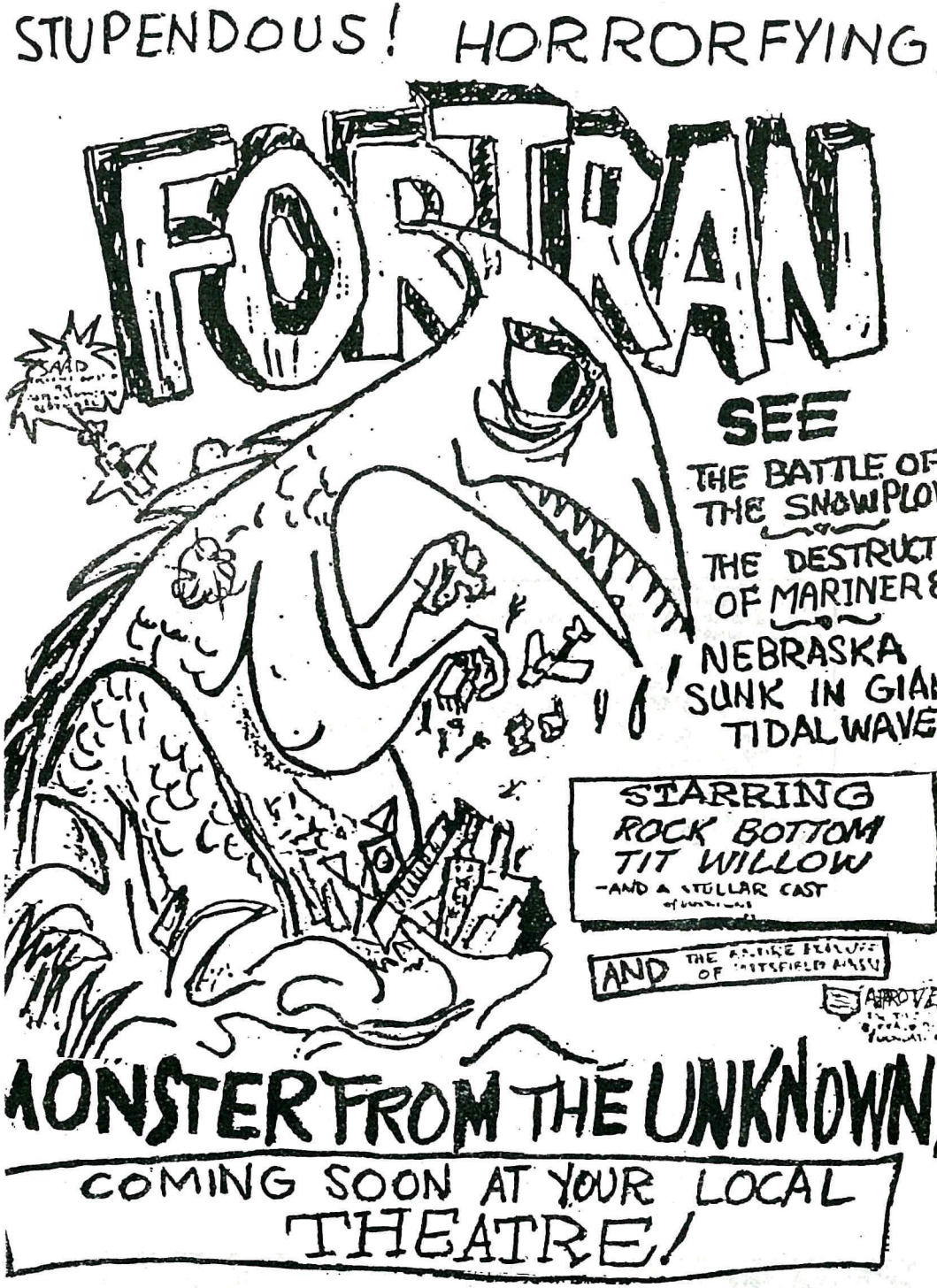
UCSD PASCAL finns att få fix och färdigt att laddas upp från operativsystemet CP/M. Har man inte CP/M, är inte detta någon katastrof, utan det enda man behöver göra själv är att skriva ett par primitiva I/O-rutiner och sedan länka ihop det på något sätt. Det följer med instruktioner hur detta skall gå till. För att läsa CP/M-disketterna är det dock förstås bra om man åtminstone har tillgång till ett system som har CP/M. Ett annat alternativ är om man kan läsa Zilogs eget diskformat, där finns också UCSD-systemet implementerat.



1) **UCSD** är förkortning för **U**niversity of **C**alifornia, **S**an **D**iego

2) "P" i "P-maskin" står för PSEUDO, d.v.s. den är inte riktigt riktig. Men den är ordentligt genomtänkt, och fungerar ungefär som vilken processor som helst. De maskininstruktioner den har, är valda så att det ska gå att göra (kompilera) effektiva p-kodsprogram av ett Pascal-program.

3) Ryktet förtäljer att Ken Bowles som var projektledare för Pascalsystemet numera sitter och snickrar på en ADA (det där nya språket som förväntas bli jättebra och jättespritt någon gång omkr. 1983). Med lite tur, ska det kanske gå att köra på en mikrodator... även om det kanske krävs en 16-bitsprocessor.


STUPENDOUS! HORRORFYING
FORTRAN
SEE
THE BATTLE OF
THE SNOWPLO
THE DESTRUCT
OF MARINER 8
NEBRASKA
SUNK IN GIA
TIDAL WAVE
STARRING
ROCK BOTTOM
TIT WILLOW
-AND A STELLAR CAST
AND
MONSTER FROM THE UNKNOWN
COMING SOON AT YOUR LOCAL
THEATRE!