Tekniska Högskolan
i Stockholm

Stockholms Universitet

NUMERISK ANALYS OCH DATALOGI

TAMINO - ETT PROGRAM FÖR STATISTIKINSAMLING VID MINSVEPNING

av

Patrik Fältström

TRITA-NA-E8868

NADA (Numerisk analys och datalogi)
KTH
100 44 Stockholm

Department of Numerical Analysis
and Computing Science
Royal Institute of Technology
S-100 44 Stockholm, Sweden

TAMINO - ett program för
statistikinsamling vid minsvepning

av

Patrik Fältström

TRITA-NA-E8868

TAMINO

Introduction

This program was written by order of the 6th mine clearance squadron in the Swedish Naval Forces. The work was originally leaded by commodore Dick Börjesson and later by liutenant-commander Thomas Ekvall and commander Rolf Blomqvist. Almost all programming was done during the winter 1986-87, but some tuning of the system have been made in cooperation with the crue of HMS Blidö, HMS Arkö and HMS Styrsö.

How it all started

Some words about the problem with minesweeping

During a minesweeping you don't really know where you have been, and if you know that, you have problem to know if you have swept the route so many times that it is free from mines. One idea (the idea that TAMINO follows) is to divide the route into small rectangular areas, and count for each rectangel how many times you have passed it with the sweep. If you make some stastistical calculation on that number, you can get a percentual value of how safe the route is for a specific type of ship.

Until TAMINO came, all this calculations was done by hand on a chart by an old salt who had years of experience of minesweeping. It occupied two or maybe three officers on board the ship and it took about one day if it was an ordinary route. To be sure that the route was clear, you made a few rounds for safety's sake.

One help came with the Motorola system. It is a navigational system based on position finding by radio. In some way the system gives the position with an error which is incredible small whenever you want it. The idea was to collect all this positions, save them for the future, present where you have been and make some calculations witch should end with some sort of map where the nonacceptable parts of the route is marked. All this points to a special designed program with database functions, but also powerful capabilities of computing.

The idea of writing a program was born several years ago by commodore Dick Börjesson. The first version was written 1984 in BASIC on a Luxor ABC-80 but this program didn't comply with the wishes. It was slow, the database part of the program didn't work very well, and the evaluation was irritating slow, even if you didn't have as many points as you have in real life. Therefore the project fell asleep.

The birth of TAMINO

I did my military service on The Swedish Naval Staff in Stockholm. My task was to do some service on the computers, like backup, and to write a program that carried out most of the wishes that I listed above. This program ended up in TAMINO.

The first version was written in C, but problems with the part of the program that handeled the database made me write it in Progress. Computing parts of the program, like the evaluation and presentation of the map-alike picture, was still written in C and called from Progress. Progress is a so called 4GL-tool, fourth generation language, which is much more similar to ALGOL than SQL. You can see the capabilities of handeling the database as an extension, just as the extension from Pascal to Object Pascal. If you want more information of Progress see appendix one.

How is TAMINO working?

The database

All information is stored in a database which Progress handles automatically. In the database information about the operation, the route, the sweep and all positions delivered from the Motorola system. Today all this is entered by hand, but TAMINO has capabilities to do some of this automaticly in the future. It is some limitations on the MS-DOS operating system that prevents it, but it is possible on UNIX (or XENIX).

The database is structured like this: Each operation is unique by its number and the date that it starts and an index is made upon this. In the route is the routes number, the positions of the transmitters and the fairways length stored. An index is built on the number. An operation and a route together makes a 'sweeped route'. On each swept route, an operation can make several runs. A run shall follow a straight line which is placed a fixed number of meters from the middle of the route. That ideal line is called a course. When you register a run, you have to give the number of your ship and the number of the course. These two numbers plus a consecutive number makes a number of the run which (together with all fields from the index of the swept fairway) makes the unique key. While you are sweeping this course the Motorola system gives you the number of meters you are of course, that is the distance from the middle of the route, and how far away from the start of the route you are. This information makes one point. If you find a mine, this can also be registred in TAMINO.

What happens with the data?

You choose one route to evaluate and two dates that bounds the runs you want to choose, and start the evaluation. In the evaluation the route is divided into small rectangels with variable sizes. Between each pair of points a straight line is drawn, and the width of the run is fetched from the database. This makes a parallellogram, and in each small rectangle that is into this paralellogram, the program increases a counter. A better way to do can be to draw a spline, or circular arcs between the points, but there is a lot of other errors that are worse, like the problem that the extension of the magnetic field is not the same all the time.

When this is done with all pairs of points, the result is presented on a file. The file can be printed or just displayed on the screen.

You can find more information of the function of TAMINO in appendix 2.

The connection between my work and my courses

What I had read before I wrote TAMINO

When I wrote TAMINO I had finished my second year at the mathematician line in Stockholm. I must say that none of the courses that I had made was about writing such a large program. The only good thing was that I was used to write programs. I don't think the special knowledge about a special version of a language or computer is very useful. It is the ability to quickly absorbe information abuot a complete new system, and also present some result very quick.

The third year at the university

The technique of programming has been trained a lot more in the third year where one meets several new languages. One course in the third year, 'Databasteknik', was very good. I could with my experience from the TAMINO project find the solutions of several problems I had with TAMINO during the writing. Some examples are the diffrent normal forms and the problem with inconsistence and redundance. Another very good cource was the 'Människa-Dator-Interaktion' where I recognized several problems that I had when a person, an end-user, should work with my program. It was really a big part of the total amount time that I had to put on the interface between the user and the program. The problem for me, as the programmer, was to understand the users conceptuation of the whole system. This involved for example moving data from one object in the database to another that was more natural for the people that really did sweep routes.

Stockholm 880519
Patrik Fältström

4GL
Myt eller möjlighet

En uppsats skriven av Patrik Fältström som ett moment i kursen Datorns Möjligheter och Begränsningar som ingår i Matematikerlinjens fjärde årskurs. Dessutom medtagen i mitt examensarbete på Matematikerlinjens dataloginriktning.

Det som kommer behandlas är 4GL-verktyg i allmänhet och PROGRESS i synnerhet.

Innehållsförteckning

1 Var befinner sig 4GL bland relationsdatabaserna?

1.1 Vad är en databas?

C.J.Date[1] beskriver en databas som "En förvaringsplats för en mängd av datoriserad information".

Ullman[1] tycker att en databas är "En abstraktion av verkligheten, eller en approximation av verkligheten".

Codd[1]: "En mängd relationer som representerar någon modell av verkligheten genom att lagra information om de objekt modellen ska representera i relationerna".

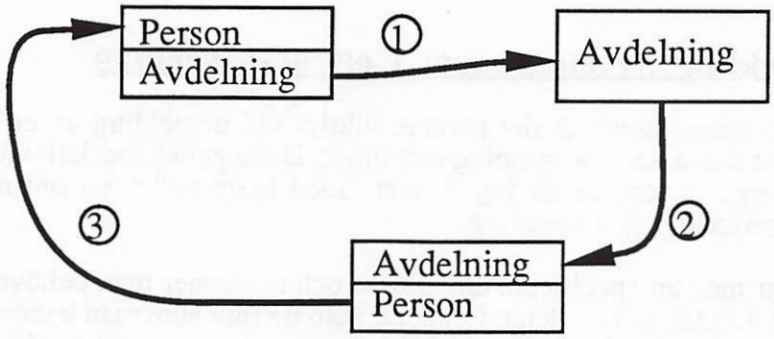
Detta är tre åsikter om vad en databas är. Sannolikt finns det lika många åsikter om detta som det finns användare av databaser, men det intuitiva synsättet med en databas som en kortlåda som är ordnad på något sätt (tex i bokstavsordning) tycker jag ger en bra bild. Man har definierat utseendet på ett kort där man skriver personens namn, adress och telefonnummer. När man ska lägga till en person i sitt kortregister tar man fram ett tomt kort, fyller i detta samt sorterar in det i kortlådan. Borttagning gör man genom att leta fram rätt kort, ta upp detta ur lådan, radera personens namn, adress och telefonnummer och till slut lägga tillbaka det tomma kortet i högen med lediga kort.

1.2 Vad är en relationsdatabas?

Exemplet med kortlådan där man förvarar adresser och telefonnummer till sina bekanta är lätt att realisera. Den enda lådan beskriver ett personregister, sorterad efter efternamnet på personerna. För att ge exempel på vad en relationsdatabas är lägger vi till ännu en låda vid sidan om den första. Antag att den första lådan står på kameralavdelningen i ett mindre företag och innehåller namn och adress för alla anställda och beteckning på den avdelning den anställde arbetar på. Den andra lådan står till höger om den första och innehåller information om varje avdelning. Nu vill man veta vilka som är anställda på varje avdelning, och därför inför man en tredje låda. I denna tredje låda har vi ett kort för varje anställd. På det tredje kortet skriver vi personens namn och avdelningens beteckning. Korten i denna tredje låda sorterar vi efter avdelningsnamn, och inom avdelningarna efter de anställdas efternamn.

Här nedan kan vi se hur dessa tre kort kan realiseras.

Detta tredje kort ger oss möjlighet att leta efter vilka anställda det finns på varje avdelning. Om vi vill ha adressen letar vi därefter i den första asken där de fullständiga adresserna finns. Denna tredje låda, liksom uppgiften om avdelning på person-kortet, skapar en så kallad relation.



På denna andra bild ser man lätt hur de olika sökningarna går till.

Om man vill veta vem som är chef för en speciell anställd letar man först i första kortlådan, följer relation 1, letar i låda två och ser på kortet över avdelningen vem som är chef.

Om man vill veta vilka som arbetar på en speciell avdelning letar vi först i låda två och kollar om avdelningen verkligen finns. Därefter följer vi relation 2 och plockar fram alla kort för den avdelningen. Om vi också vill ha adresserna till de anställda följer vi relation 3 till den första asken och letar där fram de anställda.

Detta samband kallas i relationsdatabassammanhang för ett ett- till n-samband[2]. En person arbetar på en avdelning, men varje avdelning har flera (n st) anställda.

Varje låda betecknas fil och varje kort i lådan en post. De fält i filen som man sorterar efter bildar tillsammans ett så kallat index. De olika sambanden i en relationsdatabas beskrivs endast genom att jämföra olika fält i olika filer och detta görs i programmet. Skaparen av databasen bestämmer alltså vilka fält han vill jämföra.

Denna enkla databas skulle dock inte realiseras genom tre olika filer. Det beror på att man i datorn kan ha flera index definierade på samma fil, men olika fält. Man kan alltså, genom att använda olika index, ha filen sorterad på flera fält samtidigt. Den första asken med anställdas namn och adresser kan sorteras dels på namn och dels på avdelningsbeteckning. Därmed behövs inte den tredje kortasken.

För att bestämma utseendet på de olika filerna och vilka fält som ska ingå i filernas index använder man ett Data Dictionary.

1.3 4GL-verktyg

4GL-verktyg kan ses som en utvidgning av begreppet relationsdatabas[3]. Beteckningen 4GL kommer från engelskans "Fourth Generation Language"; fjärde generationens språk. Till tredje generationens språk räknar man ordinära programspråk som tex C eller PASCAL. Det som gör att de kallas fjärde generationens språk är att de innehåller kommandon för att påverka en databas och definiera bildens utseende på bildskärm och skrivare. Vart och ett av dessa kommandon motsvarar många rader i tex C. De flesta 4GL-verktyg gör endast denna översättning till ett högnivåspråk. Därmed behövs även en kompilator för detta språk. Andra verktyg håller sig hela tiden med en egen kod och kompilerar denna istället. Se i övrigt del två.

4GL-verktyg var för ett par år sedan den stora innegrejen bland programmerare. Alla skulle hålla på med 4GL, men efter ett tag mattades detta av. I och med att utvecklingstiden förkortades så markant trodde man att det inte behövdes några kunskaper för att skapa en databas i ett 4GL-verktyg. Ack så fel man hade. Det är snarare så att man verkligen behöver tänka igenom situationen, strukturera det hela ordentligt, och börja från rätt ände. Uppbyggnaden av en enkel modell går på en eller ett par dagar och detta gör att man lätt kan "springa ifrån" hela projektet under programkodningen. Det krävs disciplin och att man noggrant följer kravspecifikationen utan att hoppa in på något intressant sidospår. Detta grundar jag på egna erfarenheter efter ett års arbete med 4GL.

2 Utveckling av ett system i ett 4GL-verktyg

Som jag tidigare nämnt är det mycket viktigt vid utveckling av en relationsdatabas att man specificerar databasens utformning ordentligt. Detta gäller speciellt vid utveckling med hjälp av 4GL-verktyg. Observera att jag skriver "med hjälp av" även om man i vissa 4GL-verktyg utvecklar program i själva verktyget.

Man börjar med att specificera alla objekt och relationer man behöver för att lösa det aktuella problemet. Utgående från detta definierar man de filer som man behöver, och hur dessa filer ska vara uppbyggda. Hur detta går till går jag inte in på här, men det behandlas ingående i ett flertal olika böcker, se litteraturlistan. Det man får fram registrerar man i Data Dictionary.

Nästa steg är att definiera menyer och bilder. Med bilder menas dels det som syns på skärmen under inmatning och dels det som syns på skärm eller vid utskrift vid en rapportgenerering. I en del 4GL-verktyg definieras dessa bilder och menyer i Data Dictionary, i andra definieras de i programkoden.

I det tredje steget skriver man en förenklad modell av databasen (med databas menas ibland även tillhörande programkod, vilket inte är så lyckat). Därmed låter man snabbt användarna av den färdiga produkten få prova menyer, inmatning av poster och enkla rapportvarianter så att de kan säga till i tid om man vid specifikationen tex har glömt något fält i en fil.

Tiden det tar att få igång en sådan förenklad modell av den färdiga produkten varierar naturligtvis på problemets komplexitet, men tar från en vecka till ett par månader. I COBOL skulle motsvarande utveckling ta mycket längre tid. En del tillverkare av 4GL-verktyg talar om flera hundra gånger snabbare utveckling med 4GL. En stor skillnad mellan att arbeta i ett 4GL-verktyg och i ett ordinärt programspråk är att programmeringen kan börja redan från början.

Man kan likna utvecklingen av en produkt med hjälp av 4GL med det filosofiska synsättet på utvecklingen. Programmeraren skapar en modell, en tes som beställaren får prova. Beställaren i sin tur kommer med en antites, d.v.s synpunkter på tesen som programmeraren tar åt sig. Av detta, modellen och synpunkterna, tesen och antitesen, skapar programmeraren en ny modell, en syntes, som ånyo övergår i en ny tes. Tiden mellan varje sådant steg rör sig vid utveckling i ett 4GL-verktyg om en eller ett par veckor mot att det i ett annat programspråk kan vara en betydligt längre tidsperiod. Därmed kan hela utvecklingen ses ske iterativt mellan programmeraren och beställaren. Beställaren får mycket snabbt reaktion på sina synpunkter och programmeraren får också ett snabbt resultat vilket i sig skapar en gynnsam miljö för programutveckling.

3 En närmare titt på 4GL-verktyget PROGRESS

3.1 En allmän bild av PROGRESS

PROGRESS är skrivet av Data Language Cooperation i USA och är det verktyg i Sverige som har näst flest installationer, c:a 520 st. Det verktyget som överträffar detta är MIMER som har c:a 640 st. Övriga 4GL-verktyg har betydligt färre installationer, mindre än 100 st.

PROGRESS är ett komplett verktyg. Det består av[4]:

1) En editor där program kan skrivas och testköras; felaktigheter markeras vid kompilering.
2) Ett hjälpsystem där man får fullständiga hjälptexter, dels av valfri felkod, dels av de fel som har uppkommit vid kompilering, i omvänd tidsordning. Dessutom en fullständig beskrivning av PROGRESS syntax.
3) Ett Data Dictionary där filer definieras.
4) Ett hjälpsystem där man kan dumpa data från databasen i ASCII format på valfri fil, samt återläsning av detsamma.
5) Ett system för inläsning av datadefinitioner och data från dBase-filer vilket medför att man lätt kan överföra databaser från dBase till PROGRESS.
6) Ett behörighetssystem som sköter behörighet på fältnivå.
7) Ett eget filhanteringssystem som inte använder sig av operativsystemet, detta för att snabba upp läsning/skrivning på disken.
8) Naturligtvis även en databas. Denna är uppbyggd som ett balanserat B-träd.

De definitioner som man lägger upp i data-dictionary finns även de med som poster i en databas. Den kan man använda sig av precis på samma sätt som av "sin egen". Som grädden på moset är alla de funktioner som nämndes ovan skrivna i PROGRESS eget språk, varför man kan göra egna specialvarianter genom att kopiera valda delar av hjälpsystemet.

3.2 Programspråket PROGRESS

PROGRESS eget programspråk är nära besläktat med ALGOL-språken. Idén med BEGIN...END finns liksom REPEAT, DO och IF.. ..THEN.. ..ELSE. Man avslutar varje rad med punkt och inte semikolon som annars är vanligt. De speciella kommandon som anropar databasen är mycket enkla. Här kommer ett par exempel med kommentarer. Som grund har vi det i del ett nämnda registret över anställda i ett företag.

Först vill vi kunna mata in nya personer. Detta ska fortsätta ända tills vi trycker på END-knappen <F4>. Tidigare har vi definierat databasen i data-dictionary. Så här ser det ut på skärmen:

```
Namn       : Patrik Fältström
Adress     : Åkervägen 21
Postadress: 135 53  TYRESÖ
Avdelning : K31B
```

```
REPEAT:
   CREATE person.
   UPDATE person WITH CENTERED SIDE-LABELS FRAME f-person.
END.
```

Sedan vill vi mata in ett avdelningsnamn varpå alla anställda på den avdelningen ska skrivas ut på skrivare. Om man matar in ett felaktigt värde skall det meddelas.

```
Avdelning: K31B
```

och på skrivaren:

Namn	Adress	Postadress	Avdelning
Patrik Fältström	Åkervägen 21	135 53 TYRESÖ	K31B
Per Hägglund	Jungfruns Gata 414	136 60 HANDEN	K31B
Malek Yayo	Värdshusvägen 20	145 50 NORSBORG	K31B

```
DEFINE VARIABLE avd LIKE avdelning.namn.

UPDATE avd WITH CENTERED SIDE-LABELS FRAME f-inmatning.

FIND avdelning WHERE avdelning.namn = avd NO-ERROR.
IF AVAILABLE avdelning THEN DO:
   OUTPUT TO PRINTER PAGED PAGE-SIZE 64.
   FOR EACH person WHERE person.avdelning = avdelning.namn:
      DISPLAY person.
   END.
   OUTPUT CLOSE.
END.
ELSE DO:
   BELL.
   MESSAGE "Det fanns ingen avdelning som hette" avd.
END.
```

Om detta var invecklat så kan det tilläggas att PROGRESS har c:a 190 olika kommandon! Vissa kan speciellt nämnas:

- Datatypen DATUM existerar. Man kan subtrahera två datum och få antalet dagar däremellan, addera ett heltal till ett datum och få ett nytt datum, ta reda på vilken veckodag det är ett visst datum m.m. Almanackan sträcker sig från c:a 38000 fKr till 38000 eKr.
- Algebraiska stränghanteringskommandon. STRÄNG1 = STRÄNG1 + "G" lägger till bokstaven G i slutet av strängen STRÄNG1.
- Automatiska transaktionsnivåer som medger att man ångrar sig och allt blir ogjort (inom vissa gränser).
- Postlåsning vid fleranvändarsystem, som programmeraren kan påverka i programmet.
- Fullständiga terminalhanteringsrutiner med en egen "termcap" där terminaltypen definieras.
- Fullständig kompatibilitet av programvara mellan olika operativsystem. Bara man har rätt version av PROGRESS så...
- Lätta åtkomster av andra program. I UNIX sker det antingen genom kommandot UNIX <UNIX-kommando> eller genom piping med kommandot INPUT THROUGH eller OUTPUT THROUGH. I MS-DOS med kommandot MS-DOS <MS-DOS-kommando>.
- Ett helt knippe variabler sätts av PROGRESS. Ett urval: användarens namn, datum, tid, databasens namn, operativsystemets namn (om det är UNIX eller MS-DOS), favoritskrivarens namn, minnesuppdelning m.m. Alla dessa kan läsas av om man bara anger variabelns namn.

3.2 Är PROGRESS bra?

Min erfarenhet[5] av PROGRESS är enbart positiv. Det gäller bara att tänka på att det är en databashanterare man håller på med, och en sådan är skriven för att skyffla omkring stora mängder data. Stora tunga beräkningar som inte har med databasen att göra gör man fördelaktigast i ett annat programspråk och anropar detta program från PROGRESS. PROGRESS själv verkar vara en mycket väl genomtänkt produkt. tex kan man ändra samtliga värden i data-dictionary efter det att man har börjat lägga in data i databasen. Eftersom det är lätt att vara efterklok, så blir detta en användbar utväg när man helt plötsligt ska registrera en person som har ett efternamn som är ett tecken "för långt".

4 Slutsats

4.1 Myt eller möjlighet?

Kanske har 4GL-verktyg glorifierats på sista tiden. Det gäller att se dessa programspråk för vad de är. Ska man ha **ett** specialicerat system som sätter stora krav på hastighet vid beräkning av mycket stora datamängder passar ett ordinärt programspråk bättre. Jag tror dessutom inte att det kommer säljas program skrivna för 4GL-språken. Det enda som skulle kunna vara gångbart är ett programskelett för menyer, databashantering och mer speciella men allmänna funktioner som reskontra och registerfunktioner. Sedan skall var och en kunna skriva till den delen av programmet som som just han behöver. Tex behöver en bokhandlare kunna kolla checksiffran på ISBN-nummer.

4GL-verktyg är en mycket bra produkt när det visar sig att man behöver en speciell variant av databas. Ett problem man kan ha är att tex lagerhållning och bokföring sker på olika sätt på olika företag. Då ger 4GL-verktygen en möjlighet att lätt skapa en egen, effektiv arbetsmiljö för slutanvändarna. Om ett företag kan optimera ett program efter sitt eget specifika behov, så behöver detta inte bli dyrare för företaget även om själva programutvecklingen blir dyrare. Vinsten kommer istället tillbaka i minskad arbetsåtgång för de anställda. Arbetsglädjen gör kanske t.o.m. att de anställda stannar längre på företaget. Därmed tjänar man in utbildningskostnader också.

Kostnaden för ett 4GL-verktyg är förhållandevis högt. En installation av PROGRESS i ett UNIX system kostar tex 65000:-. Men den stora kostnaden får man förhoppningsvis tillbaka i minskad programmeringstid, bättre program och ett snabbare resultat vid programmeringen. Kostnaderna för att programmera i tex C blir mycket högre då det kräver mer programmeringstid och redan där tjänar man kanske igen den högre kostnaden för ett 4GL system.

4.2 Egna erfarenheter

Min erfarenhet grundar sig enbart på PROGRESS och arbete på Marinstabens ADB-avdelning under tiden aug 1986 - maj 1987. Det framkom att de olika avdelningarna (grupper om 4-10 personer) har egna behov av specialiserade databaser, med eller utan anknytning till andra beräkningsverktyg. Det är då som ett 4GL-verktyg verkligen kommer till sin rätt. Under den tiden producerade jag tre adressregister för Marinnytt, KA-nytt samt HMS Carlskronas besättning. Dessutom finputsning av ett system för planering av inryckandebehov och utplacering av värnpliktiga i Kustartilleriet. Dessa fyra projekt gjordes vid sidan av att ett komplett system för bokföring av, samt utvärdering vid minsvepning gjordes. Det sistnämnda systemet, TAMINO, är numer i produktion ombord på kustminsvepare på bärbara datorer som använder sig av MS-DOS varianten av PROGRESS.

Detta visar ungefär hur kraftfullt detta programspråk är. Dock kan man invända på att tunga beräkningar som inte har med databasen att göra bör skrivas i ett annat programspråk, vilket tidigare nämnts.

På grund av att kommandouppsättningen är så komplex och omfattande som den är, tog det ett tag innan jag klarade av att påverka vissa standardfunktioner hos PROGRESS som fönsterhanteringen. Innan jag klarade detta fick jag acceptera de standardinställningar som jag fick. Detta gav ibland oanade effekter vilket i sin tur gjorde att jag efter varje ändring i programmet undrade hur det skulle te sig på skärmen. Detta var dock inte en nackdel, för man kommer lättare igång med programmeringen om man inte behöver ange en mängd konstiga parametrar utan får vissa standardvärden.

5 Litteraturlista

1 Föreläsningsanteckningar från kursen Databaser, Serafim Dahl, NADA, KTH

2 An Introduction To Database Systems av C.J.Date, Addison-Wesley 1986

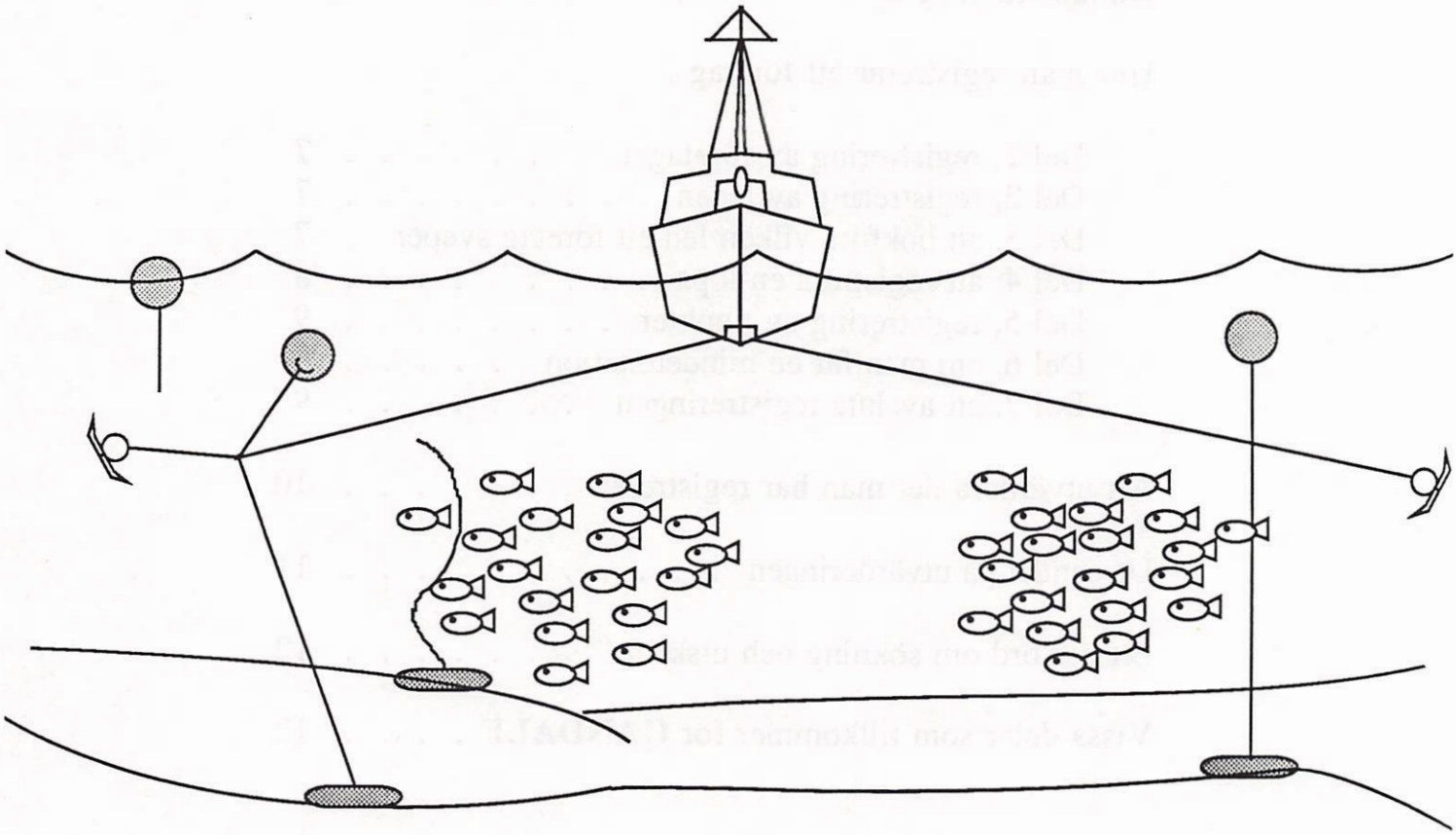
3 Industriell Datateknik, en artikelserie om 4GL-verktyg av Lars Karlander, nummer 13-16 1987 (Framför allt nummer 13)

4 PROGRESS Reference Manual, DLC, USA (om programspråket PROGRESS)

5 Dokumentation och utvärdering av projektet TAMINO, Patrik Fältström och ÖrlKn Thomas Ekvall AATAK 6:e minröjavdelningen

ANVÄNDARHANDLEDNING VERSION 4.1

TAMINO



MS-DOS versionen på 6:e minröjavdelningen

Patrik Fältström 1987

INNEHÅLLSFÖRTECKNING

INLEDNING:

TAMINO har skrivits i programspråket *PROGRESS* med vissa delar i **C** under operativsystemet **UNIX V** på en NCR TOWER-XP på Marinstabens ADB-avdelning i Stockholm[1].

Synpunkter på programmet har inhämtats från berörda i 6.mröjavd under bl.a. TMRÖ 4 1987. Denna version skiljer sig från tidigare versioner främst genom att begreppet Svept Ledavsnitt ej längre finns. Dessutom kan man numer begära utvärdering av endast en del av en led.

Ännu fler synpunkter mottages tacksamt hem till mig eller till Thomas Ekvall 6.mröj, nu när jag har slutat på MS/ADB.

Tyresö 1 november 1987

Patrik Fältström
Åkervägen 21
135 53 TYRESÖ

[1]Nu är hela TAMINO överfört till MS-DOS, UNIX-versionen finns ej mer.

Start av TAMINO (på 6.mröj):

Se till att det INTE sitter i några disketter i något av diskettfacken. Slå på datorn och skrivaren. Om allt går bra kommer du efter ett tag få upp en meny där ett flertal alternativ finns tillgängliga.

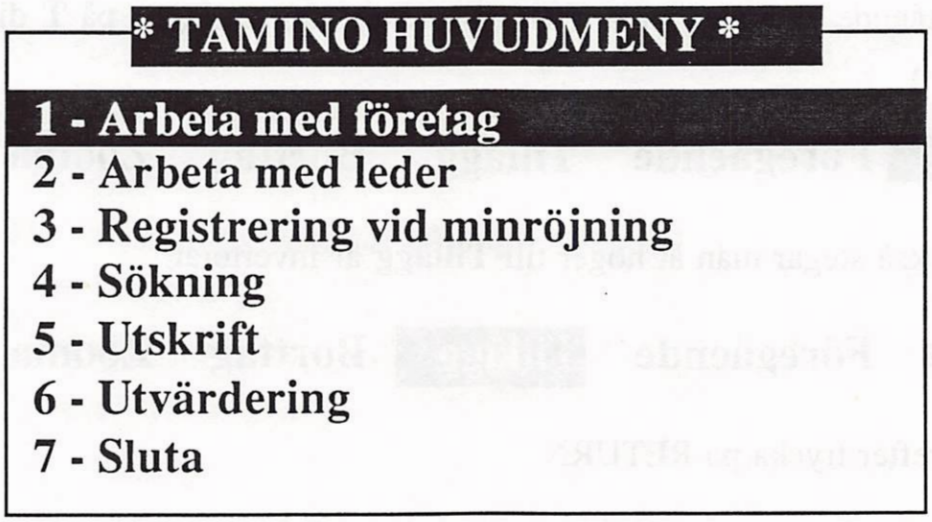
Välj alternativet TAMINO i menyn genom att stega dit med hjälp av piltangenterna och sedan trycka på RETURN-tangenten.

Om det uppstår något onormalt kan det bero på att föregående körning avslutades på ett felaktigt sätt. Det kan tex ha blivit strömavbrott. Detta fixar *PROGRESS* själv när TAMINO startas. Du får då se texten 'TAMINO.LK fanns, tas nu bort'. Om det ändå krånglar, ring Marinstabens ADB-avdelning om inte någon i närheten kan tolka och ordna upp felet.

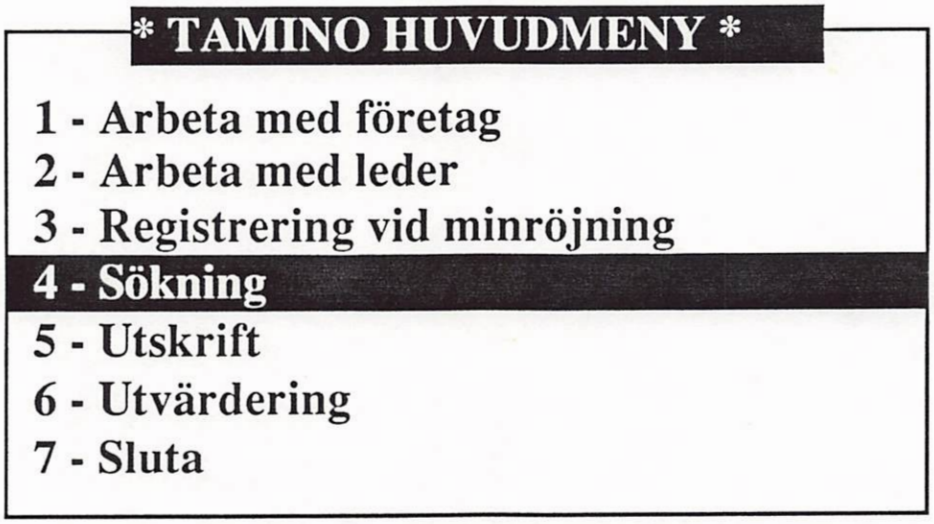
För att lösa detta problem så gör följande (**OBS VAR MYCKET NOGRANN!!!**):
1) Gå till biblioteket C:\TAMINO och ge kommandot PRO -F TAMINO. Du kommer då till PROGRESS egna editor. Ge kommandot DICT (skriv DICT och tryck på <F1>), och du kommer till PROGRESS databas-meny. Där väljer du alternativ 3-Load/Dump Data Files. I den nya meny du kommer till väljer du först alternativ 1-Dump Data Definitions. Ange att du vill spara ALL(a) genom att trycka på <RETURN>. Acceptera det val av filnamn, TAMINO.DF, som föreslås (tryck på <RETURN>). Tryck på mellanslag när det är färdigt. Välj nu alternativ 2-Dump File Contents i menyn. Ange att du vill spara ALL(a) återigen. Tryck på mellanslag igen, när det är klart. Gå ur menyerna genom att välja alternativ 5-Exit samt 6-Exit. Gå ur PROGRESS editor genom att ge kommandot QUIT <F1>. Du hamnar nu i MS-DOS igen.
2) Flytta den gamla databasen med upprepade användningar av kommandot COPY och DELETE:
COPY TAMINO.DB DUMP\TAMINO.DB
DELETE TAMINO.DB
COPY TAMINO.DB DUMP\TAMINO.LG
DELETE TAMINO.LG
3) Skapa en ny databas med kommandot PRODB TAMINO EMPTY.
4) Starta denna med kommandot PRO -i TAMINO
5) Ge kommandot DICT<F1> igen, och även alternativ 3-Load/Dump Data Files. Välj i den menyn nu först alternativ 3-Load Data Contents. Ange att du vill ladda från filen TAMINO.DF. Tryck på mellanslag när allt har gått bra. Välj nu alternativ 4-Load File Contents. Välj att du vill läsa in data från ALL(a) filer. Nästa sak att ange är ett acceptabelt fel, i detta fall är 10% OK. Dessutom ska du svara YES på att du använder 'no-integrity option' (det går mycket snabbare på detta sätt). Om det blir något fel under laddningen så anges det i vilken fil som felet är beskrivet, titta i denna med tex EDLIN. Avsluta på samma sätt som du gjorde tidigare (alternativ 5,6 samt EXIT<F1>).
6) Du kan nu ge följande kommandon:
DELETE *.DF
DELETE *.D
DELETE DUMP/TAMINO.*

Menyerna:

I menyerna kan man välja alternativ på två olika sätt. Som exempel visas här hur man väljer alternativet **4 - Sökning** i huvudmenyn.



Man väljer alternativ antingen genom att trycka på den siffra som motsvarar det alternativ man vill välja, eller genom att stega med piltangenterna till rätt alternativ är inverterat[2] och trycka på RETURN. Tryck alltså på **4** eller stega ned och tryck på RETURN.



[2]Med inverterat menas att texten skrivs med bakgrundens färg och i en ruta kring texten har bakgrunden antagit textens originalfärg.

Kommandoraderna:[3]

Alternativ väljs på ett likartat sätt, men här trycker man på första bokstaven i det kommando man vill utföra, eller stegar man höger-vänster till rätt kommando och konfirmerar med RETURN. För att i nedanstående exempel välja **Tillägg** trycker man antingen på **T** direkt (eller **t**),

Nästa **Föregående** **Tillägg** **Borttag** **Zooma** **Sluta**

eller också stegar man åt höger till **Tillägg** är inverterat:

Nästa **Föregående** **Tillägg** **Borttag** **Zooma** **Sluta**

och därefter trycka på RETURN.

[3]Dessa stöter du endast på i alternativet **Registrering vid minsvepning**

Hur man registrerar ett Företag

Del 1, registrering av företaget:

Välj alternativet **1 - Arbeta med företag** i huvudmenyn. Du kommer till en undermeny där man kan påverka ett företag på olika sätt.

Välj alternativet **1 - Skapa nytt företag** i denna meny. Ange företagsnamn (endast siffror) och sedan datum (på formen ÅÅMMDD). Detta par av uppgifter kommer att vara företagets UNIKA identitet. Endast ett företag i hela databasen kan ha en kombination av dessa två. Om det redan fanns ett företag med denna kombination i databasen kommer det upp ett litet felmeddelande om detta. Ange även röjstyrka och ev tidskod. Dessa bägge värden är dock ej viktiga för utvärderingen.

Återvänd till företagsmenyn genom att trycka på <ESC>. Härifrån kan du även söka efter ett företag och se bla vilken röjstyrka som svepte i det företaget.

När du är färdig med företaget återvänder du till Huvudmenyn.

Del 2, att registrera leden:

Om inte Leden är upplagd tidigare skall även detta göras. Välj alternativet **2 - Arbeta med leder** i huvudmenyn och sedan **1 - Skapa led** i ledmenyn. Ange lednamnet (endast siffror). Detta lednamn är unikt. Om leden redan fanns får du ett felmeddelande. Återgå till ledmenyn.

Om du är osäker på vilka leder du har registrerat kan du välja **Sökning leder** i ledmenyn.

Del 3, att bokföra vilken led ett företag sveper:

Välj alternativ **3 - Registrering vid minsvepning** i huvudmenyn. Ange företagets namn och datum (det som var unikt för ett företag). Som du ser kommer det upp två rutor till upp på skärmen så att det nu är tre stycken. Den översta visar vilket företag du håller på att registrera. Den mellersta visar efterhand vilken led, löpa, punkt som för närvarande är **aktuell**. Den aktuella posten visas i det nedre fönstret

och det är denna som du påverkar med kommandoraden som syns nederst på sidan.

Om det inte tidigare är svept en led i det företag du angav kommer du automatiskt till kommandot **Tillägg** i kommandoraden. Ange ledens namn. Då skapas det en sk **Svept Led** som man ska skilja från en **Led.** En svept led är kopplingen mellan ett företag och den led som företaget sveper. Om företaget 4151, 871017 sveper leden 12345-34 så skapas en svept led bestående av 4151, 871017 och 12345-34. Dessa tre värden blir den svepta ledens unika nyckel i databasen. Lägg märke till att om ett annat företag, 4152, 871017, sveper samma led kommer endast en ny svept led bildas. Denna gång av 4152, 871017 och 12345-34. **En ny led skapas således inte.**

Du kan bläddra mellan de leder som företaget svept med kommandona **Nästa** och **Förra** i kommandoraden.

Ledens början är det mycket viktigt att man håller reda på. Det måste man definiera innan man börjar svepa pga att alla avstånd man anger ska anges från denna punkt i meter oavsett från vilken ända av leden man sveper. Sveper man i negativ (se del 4, registrering av löpa) riktning ska alltså avstånden minska vartefter man sveper löpan.

Del 4, registrering av löpa:

Bläddra bland de svepta lederna med **Nästa** och **Förra** tills du har rätt led aktuell. Ge sedan kommandot **Zooma**, och du kommer till den del av TAMINO där du registrerar en löpa på en viss svept led.

Precis som när du kommer till registrering av en led på ett speciellt företag så hamnar du automatiskt i kommandot **Tillägg** om det tidigare ej fanns några löpor registrerade i den svepta leden.

Här fungerar kommandot **Tillägg** på ett lite speciellt sätt. Längst ned till vänster på skärmen kommer det upp ett par frågor som du måste besvara för att kunna skapa löpnumret. Först anger du ett tvåsiffrigt fartygsnummer, sedan en ensiffrigt stråknummer och sist löpans nummer i stråket. Dessa bildar löpnumret: **56:1:02** om fartyget med nummer 56 (M56 antagligen) sveper i stråk nr 1 och där gör sin andra (02) löpa. Företagsnummer, företagsdatum, lednummer samt detta löpnummer är unikt för en löpa.

Om man sveper i **pos** (positiv) riktning kommer man att börja svepa i ledens startpunkt och tvärtom för **neg** (negativ) riktning.

Del 5, registrering av punkter:

Bläddra bland löpen och välj sedan alternativet **Zooma** i kommandoraden. Du hamnar direkt i den del där du registrerar dina punkter. Dessa har punktkoderna A,B,...,Z,AA,AB,...,ZZ och varje punktkod bör för en och samma led bindas till ett avstånd från ledens början.

Två saker är **mycket** viktiga:

1) Om man sveper i negativ riktning ska man börja med sista punkten i leden (tex AF) och sluta med punkt A. Avstånden kommer då att minska.

2) Man behöver inte svepa hela leden utan man kan börja och sluta precis var man vill. Kom bara ihåg att ange rätt avstånd till rätt punkt.

När man sveper i positiv riktning betyder ett positivt läge att man ligger för långt åt babord i förhållande till stråkets mitt. Man har alltså stråkets mitt på sin styrbordssida. Naturligtvis blir det tvärtom om man sveper i negativ riktning.

Del 6, om man får en mindetonation:

Välj alternativet **Mina** när du registrerar punkter. Ange läget för minan i stråket med avstånd från ledens början och avståndet från stråkets mittlinje på samma sätt som du registrerar en punkt.

Del 7, att avsluta registreringen:

Ge upprepade gånger kommandot **Sluta** i kommandoraden (eller tryck på <ESC>) så kommer du till slut tillbaka till det läge när man ska ange vilket företag man vill arbeta med. Där måste du sluta med <ESC>, varvid du åter hamnar i huvudmenyn.

Att utvärdera det man registrerat:

Välj alternativet **Utvärdering** i huvudmenyn.

Du kommer då till en till synes obegriplig inmatningsprocedur där du ska ange vilket tecken som ett givet antal svepningar ska ge vid utvärderingen. Om du har svept en del av leden 6 gånger kommer det tecken som står under sexan att skrivas ut. Som du kan se får du ett förslag på vad som ska skrivas ut. Om du tycker detta duger trycker du på <F1>. Man kan genom att ändra här ändra utseendet på utvärderingen till det som passar. T.ex. kan man om man är nöjd med 8 svepningar ändra alla tecken som står för 8, 9, 10, 11 osv till . (punkt). Se nedan.

Ange sedan vilken led du vill utvärdera. Om du inte ger något lednamn så får du återigen se resultatet av den förra utvärderingen du gjorde. Sedan är det dags att tala om vilken del av leden du vill utvärdera. Detta anger du genom att ge start och slut (i meter från ledens början) och mittlinje (avstånd från ledens mitt). Dessutom ska du ange vilken skala du vill ha utvärderingen i (märk att om man ger en stor skala så kan man missa vissa brister i svepningen) och begränsningen i vilka företag som ska räknas med. När allt detta är gjort startar första delen av utvärderingen. Den tar upp till en minut, så det är bara att vänta. Efter denna första del ska du tala om om du vill ha det utskrivet på papper eller bara se det på skärmen. Man kan första gången se det på skärmen och andra gången (genom att slå blankslag när man anger lednamn) skriva ut det på skrivare. Efter detta startar den 'tunga' delen av utvärderingen. Den kan för ett stort antal punkter och en liten skala ta flera tiotals minuter, så det passar bra för en kafferast. Man kan alltså lämna datorn för sig själv i detta läge.

Om man har valt att få det hela på skrivare kommer det naturligtvis ut där, men om man har valt att få det på skärmen?

Jo, mellan varje skärmsida kommer bläddringen att stoppa och det skrivs i nedre vänstra hörnet en uppmaning: **--MORE--** vilken betyder att du ska trycka på blankslaget för att se nästa sida. Det är alltså ingen fara att sidorna rinner iväg och du inte hinner se vad som står.

Utseendet på utvärderingen:

Först kommer en skala. I mitten står hur många meter från ledens start som den ligger, och vid sidorna hur många meter åt sidorna från utvärderingens mitt som ***siffrans vänsterkant*** *är.*

```
-400    -300     -200     -100      0      200      300      400
                            1123489AAA877611981
                           11235888|A988866119911
```

Man ser att det i mitten står ett '|' eller någon bokstav. Om det är en bokstav är det bokstavskoden som står för just det avståndet från ledens början. Även markeringen för en mindetonation kan du bestämma i början av programmet. Här markeras den av ett ''.*

```
                           112223489|AA877*11981
                            11235888|A9888559971
                           111123888BAAA884449541
-400    -300     -200     -100      240    200      300      400
```

*Här ser vi att denna skala kom efter 240 meter. Strax före denna (efter 230 meter, ty här är varje ruta 10*10 meter) kom punkt B.*

Om man hade valt att markera alla punkter där svepningen hade varit bättre än 7 med punkt hade det sett ut så här:

```
                           1122234..|...77*11..1
                            11235...|.....55..71
                           111123...B.....444.541
-400    -300     -200     -100      240    200      300      400
```

Då ser man mycket lätt var bristerna finns.

Ett par ord om sökning och utskrift:

Funktionerna under Sökning och de under Utskrift i huvudmenyn är lika. Man kan söka efter ett vissa företag eller vissa leder. Detta görs i alternativen **Sökning Företag xx-yy** resp **Sökning Leder xx-yy**. Som exempel på hur dessa båda fungerar kan vi titta på **Sökning Leder xx-yy**. Man anger start och stopp för det intervall där ledens namn ska ligga för att komma ut på utskriften. Lederna sorteras alfanumeriskt, vilket innebär att led 2 kommer efter 199 men före 219128. Om man har lederna 111, 123, 215, 225 och 301 så får man om man anger start:1 och stopp:21 ut lederna 111, 123 och 215 på utskriften. **Sökning Företag xx-yy** fungerar på motsvarande sätt.

Företag kan man dessutom söka efter med hjälp av datum, röjstyrka eller något annat. Detta görs i alternativet **Sökning Företag**. Ett likartat förfaringssätt finns i alternativet **Sökning Svepta Leder** och **Sökning Minor** som med hjälp av företag, röjstyrka, år el.dyl. ger de svepta leder som uppfyller kraven.

All utskrift sker på stående A4.

Glöm inte slå på skrivaren innan du ger kommandot om utskrift. Om man glömmer det så kan det hela krångla till sig.

Vissa delar som som tillkommer för GANDALF:

Följande kommandon tillkommer för den som startar med valet **GANDALF** i MS-DOS menyn istället för TAMINO.

Backup:
Det kanske viktigaste kommandot är backup. Med hjälp av detta kommando kan du spara eller läsa tillbaka delar av databasen till/från filer som ligger på biblioteket C:\TAMINO\DUMP*.D

Om någon post är felaktig vid återläsning hamnar denna post på filerna C:\TAMINO\DUMP*.E. Som felaktig post räknas även en post som redan finns i databasen.

Att flytta data från en dator till en annan med hjälp av en diskett:

Sätt i en tom diskett i diskettenhet A: (eller ev B:).

Ge följande kommandon.
C>DEL \TAMINO\DUMP*.*
C>GANDALF
I TAMINO ger du kommandona backup, spara data och sedan sluta i huvudmenyn så att du kommer tillbaka till MS-DOS.
C>COPY \TAMINO\DUMP*.D A:
...eller B: om det var den diskettenheten du valde att använda.
Tag ur disketten och flytta den till den andra datorn.
C>DEL \TAMINO\DUMP*.*
C>COPY A:*.D C:\TAMINO\DUMP
C>GANDALF
I TAMINO ger du kommandona backup, läs data, ta INTE bort allt, och sluta senare i huvudmenyn så att du kommer till MS-DOS. Om det stod "...felfil **DUMP\FORETAG.E**" eller något liknande så ska du kolla vad det var för fel.
C>TYPE \TAMINO\DUMP\FORETAG.E | MORE
Eller vad nu filen hette. Där står precis vad det var för fel.

Byta lösenord:

Du anger det gamla lösenordet och om detta var rätt det nya. För att du inte ska slå fel får du slå det nya även en andra gång. Detta får du göra för både TAMINO och GANDALF.