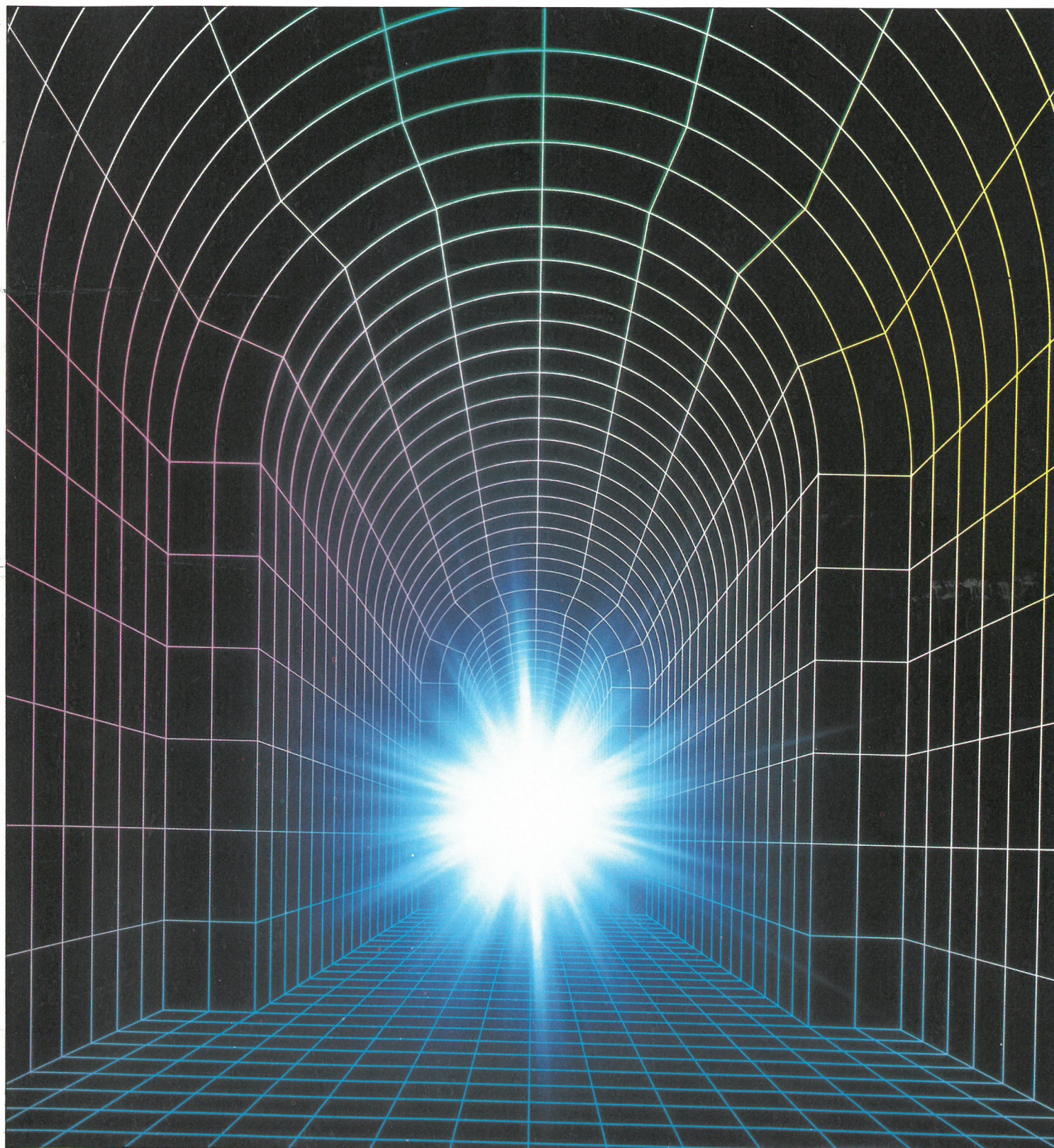
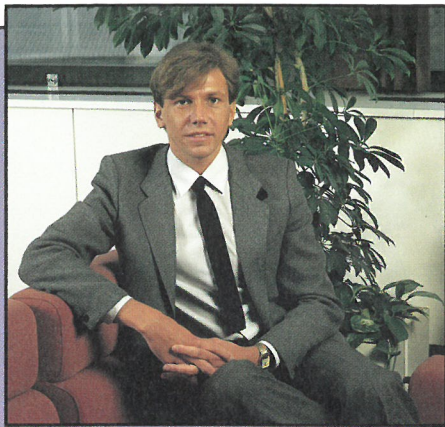


Softari

NOKIAN OHJELMISTOLEHTI 3/85





Ohjelmointi muuttuu

Kaikki muuttuu, ei entiselleen mikään jää ...

Niin myös ohjelmointi. Ensimmäisinä muutoksen merkkeinä ovat ns. neljännen sukupolven sovelluskehittimet. Niiden käyttö on valtaamassa alaa.

Arvellaan, että sovelluskehittimillä ohjelmointityön tuottavuus nousee kymmenkertaiseksi.

Uusien ohjelmointivälineiden käyttöön on muitakin syitä. Yksi tärkeimmistä on ohjelmointikustannusten kurissapito. Koska ohjelmistotuotanto on työvoimavaltaista, on ohjelmien teko kallista. Uusin välinein voidaan ohjelmointityötä nopeuttaa ja siten säästää kustannuksissa. Toisaalta puute atk-ammattilaisista pakottaa uusien menetelmien käyttöön. Lisäksi sovelluskehittimet helpottavat ohjelmien yllä-

pitoa, joka perinteisessä ohjelmoinnissa on niin työlästä ja hidasta.

Kehittimet soveltuvat pitkälti myös loppukäyttäjien työkaluiksi. Niiden käyttö ei vaadi yhtä perusteellista atk-koulutusta kuin perinteiset ohjelmointimenetelmät.

Sovelluskehittimien käytön yleistyessä niitä on kehitetty voimakkaasti. Mikrojen tehon ja muistikapasiteetin kasvu sekä grafiikan parantuminen ovat osaltaan vaikuttaneet siihen, että erityisesti mikroiin on saatavana toinen toistaan tehokkaampia sovelluskehittämiä.

Monet mikrojen kehittimet ovat tunnettuja aiemmin lähinnä keskuskoneympäristöstä. Syynä mikroversioiden syntyyn on ollut tarve liittää mikrot keskuskoneiden tietovarastoihin.

Huolimatta neljännen sukupol-

ven ohjelmointivälineiden kehitymisestä ei niillä pystytä vielä rakentamaan kaikkia sovelluksia. Laajat monimutkaiset tapahtumankäsittelyjärjestelmät joudutaan edelleen rakentamaan perinteisillä ohjelmointikielillä. Toki niihinkin on kehitetty monenlaisia apuvälineitä.

Unohtaa ei sovi myöskään uusia ohjelmointikieliä, joista tunnetuin lienee Ada. Eikä tekoälyä. Mitä kaikkea se tuokaan sovellusohjelmointiin, sen aika näyttää.

Ehkä vasta silloin kaikki todella muuttuu. Ken tietäis sen?

Hannu Markus

Softari

Nokian ohjelmistolehti

Julkaisija:
Nokia Informaatiojärjestelmät

Painosmäärä: 10 000
Ladonta: Kaukoladonta Oy
Kirjapaino: Vientipaino Oy

Toimitus:
Keskuskatu 7, 9. kerros
PL 780, 00101 HELSINKI 10

Päätoimittaja:
Hannu Markus, puh.(90) 187 4125

Ulkoasu:
Taina Sonninen, puh.(90) 567 2980

Osoitemuutokset:
Outi Kuusela, puh.(90) 567 2017

Toimitusneuvosto:
Taru Kuhanen, Tuija Matikka, Pentti Mäkipää, Ossi Syrjä ja Hannu Markus

Tässä numerossa

Ohjelmointi muuttuu	2
Helposti ylläpidettäviä sovelluksia DIMSillä	3
Pascal-kurssi — Osa 3	4
Kestävätkö elämänlankasi	8
ORACLE-sovelluskehitin — Yhteensopivia sovelluksia eri laitteisiin	10
PC-Focus tuo keskuskoneen tietokannan mikrolle	12
Mikro-Mercur — Taloussuunnittelua mikrolla	14
MPSINDEXillä monikäyttäjäsovelluksia	16
MikroMikko 2 -lähiverkko	16
MPSFORMS-näytönkäsittelijä — Ohjelmointityökalu	18
MikroMikko 2:een	18
Nyt Lotus 1-2-3 ja MikroMikkokin ymmärtävät toisiaan	20
System-80 generoi Cobol-ohjelmia	23
Adan merkinkäsittelyrutiinit	24
Unto ja Mikko	27
Uusia DPSiä markkinoille	27
MikroMikko TopTen	27
Erittäin opettavaista!	28

HMa

Helposti ylläpidettäviä sovelluksia DIMSillä

DIMS on ohjelmisto tapahtumankäsittelysovellusten tekoon. Se pakottaa tekemään sovellusohjelmista rakenteeltaan yhtenäisiä. Ja näin sovelluksista tulee myös helposti ylläpidettäviä.

DIMS (Display Management System) on DM4-TP-tapahtumankäsittelysovellusten tekoon tarkoitettu ohjelmisto. DIMS on saatavissa kaikkiin DPS 8-laitteistoihin ja se on muokattavissa erilaisten käyttäjäympäristöjen vaatimusten mukaiseksi.

DIMS on tanskalaisen ohjelmistotalon Sofcon'in valmistama. Ensimmäiset versiot DIMSistä julkistettiin syksyllä 1984. OKO on ensimmäinen DIMS-käyttäjä Suomessa.

DIMS osana sovellusta

DIMS on sovellusohjelmioijan työkalu. Osaa DIMS-ohjelmistosta tarvitaan vain sovelluksen rakentamiseen, esimerkiksi näyttöjen määrittelyyn. Osan DIMSistä tulee sisältyä myös sovelluksen tuotantoversioon.

Jotta DIMSiä voitaisiin hyödyntää sovellusohjelmasta, on sovellusohjelmassa käytettävät näytöt määriteltävä ja tapahtumat kuvattava ennen sovelluksen ohjelmointia. Näyttö määritellään maalaamalla sekä kuvaamalla sen kentät. Näytön kentille voidaan asettaa

sekä DIMSin sisältämiä että sovellusohjelmioijan määrittelemiä monipuolisia tarkistus- ja muotoilusääntöjä.

DIMS pakottaa tekemään sovellusohjelmat rakenteeltaan yhtenäisiksi. Kaikki tietoliikenteeseen ja tapahtumien käsittelyyn ja tarkistukseen liittyvät yhteiset toiminnot hoidetaan DIMSillä. Sovellusohjelmiin jää sovelluksen kannalta vain olleellisin koodi. Näin sovellusohjelmista tulee myös lyhyempiä ja niitä on helppo ylläpitää ja hallita.

DIMS-ohjelmisto muodostaa eräänlaisen loogisen kerroksen loppukäyttäjän ja sovellusohjelman väliin. Se keskustelee käyttäjän kanssa tarkistaen sanomia ja antaen virheilmoituksia. Esimerkiksi syöttövirheen havaitessaan DIMS palauttaa käyttäjälle näytön korjausta varten. Sovellusohjelmaan palataan takaisin vasta, kun syöttötiedot ovat virheettömät.

DIMS mahdollistaa ns. HELP-toiminnot ilman ohjelmointia. Käyttäjän kirjoittaessa ruudulle HELP-pyynnön

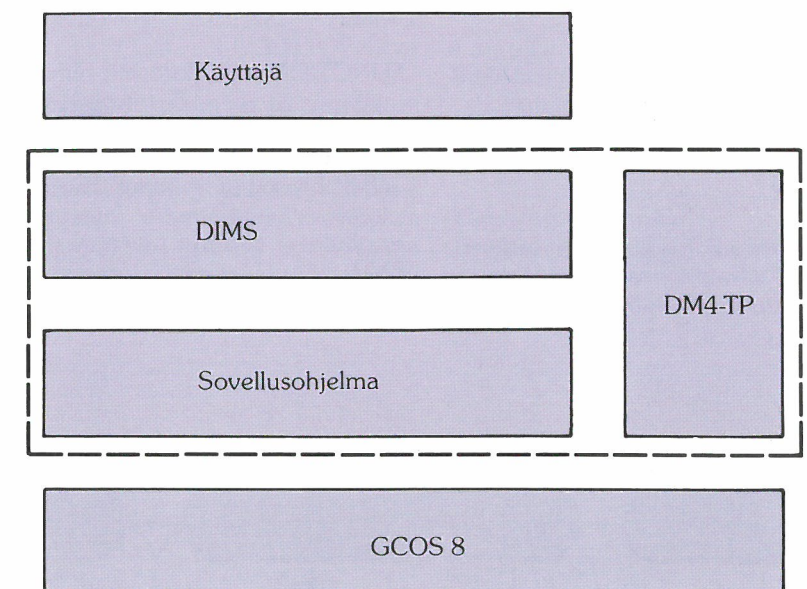
tai painaessa erityistä HELP-toimintonäppäintä DIMS antaa ko. tilanteeseen liittyvät ohjeet. Itse sovellukseen ei tarvitse ohjelmoida HELP-toimintoja, vaan ne ovat valmiina DIMSissä. Sovellusohjelmioijan tarvitsee vain määrittellä DIMSillä kuhunkin tilanteeseen liittyvät HELP-näytöt sovelluksen teko vaiheessa. HELP-näytöt voivat olla monisivuisia.

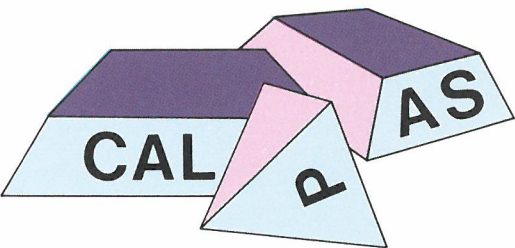
Testausta helpotettu

DIMS sisältää useita testausta helpottavia piirteitä.

Näyttöjen toimintaa voidaan testata maalauksen ja kenttien määrittelyn jälkeen jo ennen sovellusohjelmien tekoa.

Testauksen kuluessa on mahdollista tallettaa testirivejä (display-rivejä) sekä selata niitä tapahtumankäsittelyjärjestelmästä poistumatta ja häiritsemättä muita DM4-TP:ssä työskenteleviä. □





Tässä Pascal-kurssimme kolmannessa osassa paneudumme rakenteisiin tietotyyppeihin ja dynaamisiin tietorakenteisiin. Ennen kuin jatkat opiskelua, tutustunut edellisen osan harjoitustehtävien mallivastauksiin ja vertaillet niitä omiin ratkaisuihisi.

Vastaukset kurssin 2. osan harjoitustehtäviin

```
Tehtävä 1.
PROCEDURE pyydaluku(VAR luku: Integer);
BEGIN
  REPEAT
    Write('Anna luku (0-100): ');
    Readln(luku)
  UNTIL luku >= 0 AND luku <= 100;
END;

Tehtävä 2.
FUNCTION jako(osoit, nimit: Integer): Integer;
{Toimii vain positiivisilla luvuilla}
```

Pascal-kurssi
Osa 3

```
VAR
  osamaara: Integer;
  miinus: Integer;
BEGIN
  osamaara := 0;
  WHILE osoit >= nimit DO
    BEGIN
      osoit := osoit-nimit;
      osamaara := osamaara + 1;
    END;
  jako := osamaara;
END;

FUNCTION modulo(osoit, nimit: Integer): Integer;
{Toimii vain positiivisilla luvuilla}
BEGIN
  WHILE osoit >= nimit DO
    osoit := osoit-nimit;
    modulo := osoit;
  END;
```

6. Rakenteiset tietotyypit

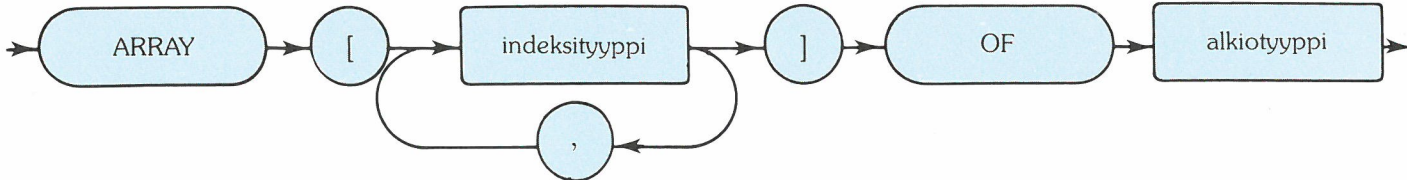
Luvussa 3 käsitelimme yksinkertaisia tietotyyppejä. Rakenteiset tyypit ovat useista samanlaisista tai erilaisista alkioista koostuvia tietorakenteita.

6.1 Taulukot

Taulukko on samaa tyyppiä olevien alkoiden joukko. Taulukkotyyppin esittelyn syntaksi on kuvassa 6.1. Taulukon koko on kiinteä, sillä alkoiden lukumäärä määräytyy taulukon indeksityypin perusteella. Indeksityyppi voi olla mikä tahansa yksinkertainen tietotyyppi, paitsi Integer ja Real.

```
TYPE
  indeksi = 1..100;
```

Kuva 6.1



1. Johdanto osa 1
2. Pascal-ohjelman rakenne 1
3. Yksinkertaiset tietotyypit 1
4. Lauseet ja ohjausrakenteet 2
5. Aliohjelmat 2
6. Rakenteiset tietotyypit 3
7. Dynaamiset tietorakenteet 3
8. Syöttö ja tulostus 4
9. Yhteenveto 4

```
rtaulu = ARRAY [indeksi] OF Real;
itaulu = ARRAY [5..12] OF Integer;
VAR
  t1: itaulu;
  t2: rtaulu;
```

Muuttujat t1 ja t2 ovat taulukkotyyppiä. Niiden alkioihin viitataan muuttujan nimen ja alkion indeksin avulla. Muuttujan t1 ensimmäiseen alkioon viitataan lausekkeella t1[1]. Muuttujan t2 ensimmäinen alkio sen sijaan on t2[5], sillä indeksityypin ensimmäinen arvo on 5.

```
CONST
  koko = 5;
TYPE
  indeksi = 1..koko
  vtaulu = ARRAY [indeksi] OF
    (pun, vih, sin);
```

```
VAR
  t3: vtaulu
  i: indeksi
...
FOR i:= 1 to koko DO
  t3[i] := pun
```

Indeksoitua muuttujaa t3[i] voidaan käyttää kuten tavallista yksinkertaisen tyyppin muuttujaa. Laittoman eli esimerkiksi taulukon ulkopuolelle viittaavan indeksin käytöstä antaa kääntäjä yleensä virheilmoituksen.

tuksen. Jos indeksinä käytettävä muuttuja saa ohjelmasuorituksen aikana laitoman arvon, seuraa ajonaikainen virhe. Pascalissa voidaan määritellä myös monidimensioisia taulukoita.

```
VAR
  kuutio: ARRAY [1..3,1..3,1..3] OF Boolean;
...
kuutio[2,2,2] := True
```

6.2 Merkkijonot

Merkkijonot ovat tavallisimpia ohjelmissa käytettäviä tietorakenteita. Pascalissa ne esitetään ns. pakattuina taulukkoina jossa alkiot ovat tyyppiä Char.

```
TYPE
  miono20 = PACKED ARRAY [1..20] OF Char
```

Merkkijonon alkioihin eli yksittäisiin merkkeihin viitataan normaalisti indeksitujen muuttujien avulla. Merkkijonomuuttujan arvoksi voidaan sijoittaa samanpituinen merkkijonovakio. Ja merkkijonon voi tulostaa kokonaisuena standardiproseduureilla Write ja Writeln.

```
VAR
  mj: PACKED ARRAY [1..13] OF Char;
...
mj := 'Blaise Pascal'
```

Kahta samanpituista merkkijonoa voidaan verrata toisiinsa käyttämällä vertailuoperaattoreita =, <>, <, >, <= ja >=. Merkkijonojen "järjestys" määräytyy aakkosjärjestyksen mukaan.

```
'Virtanen' > 'Salminen'
'Aalto' < 'Aaltonen'
'ABC' <> 'abc'
```

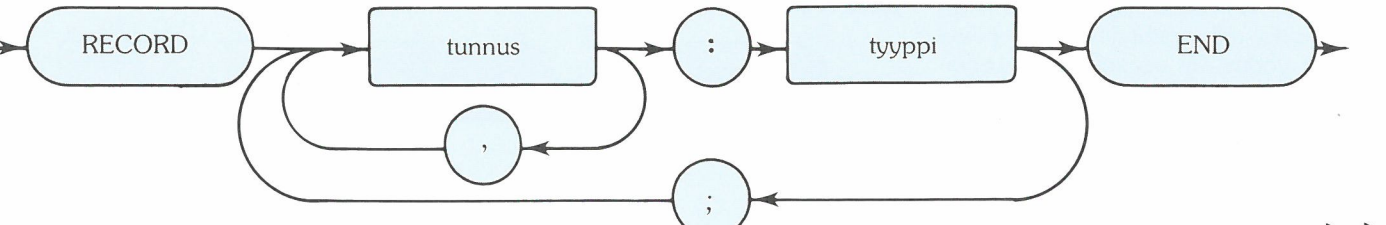
Useisiin Pascal-järjestelmiin on lisätty erillinen merkkijonotyyppi STRING, jonka käsittely on huomattavasti joustavampaa kuin standardin mukaisten merkkijonojen.

Varattua sanaa PACKED voidaan käyttää muidenkin rakenteisten tietotyyppien määrittelyjen kanssa, jos halutaan säästää muistitilaa. Useat Pascal-kääntäjät jättävät PACKED-sanankuitenkin huomiotta, joten sen käyttö on em. merkkijonotyyppien määrittelyä lukuunottamatta yleensä turhaa. Pakkaamaton taulukko voidaan kopioida pakattuun taulukkoon ja päinvastoin käyttäen proseduureja Pack ja Unpack.

```
VAR
  taulu: ARRAY [1..100] OF Char;
  ptaulu: PACKED ARRAY [1..50] OF Char;
...
Pack(taulu,51,ptaulu);
{taulun alkioita kopioidaan ptauluun alkaen alkioista taulu[51]}

Unpack(ptaulu,taulu,1);
{ptaulun kaikki alkiot kopioidaan tauluun alkaen alkioista taulu[1]}
```

Kuva 6.2



6.3 Tietueet

Tietue eroaa taulukosta siinä, että alkioiden ei tarvitse olla samaa tyyppiä. Tietueeseen kootaan yleensä loogisesti yhteenkuuluvia tietoja, joita kutsutaan kentiksi. Kuvassa 6.2 on tietuetyypin esittelyn syntaksi.

```
TYPE
  htieto = RECORD
    nimi: PACKED ARRAY [1..20]
      OF Char;
    ika: Integer;
    palkka: Real;
  END;
  ptieto = RECORD
    passi: Integer;
    sos: PACKED ARRAY [1..11]
      OF Char;
  END;
VAR
  aiti: htieto;
  isanp: ptieto;
```

Tietueen kenttiin viitataan antamalla tietuemuuttujan ja kentän nimet pistellä erotettuna.

```
aiti.ika := 35;
aiti.palkka := aiti.palkka + 300.0;
isanp.sos := '120349-763G';
```

Usein joudutaan viittaamaan saman tietueen eri kenttiin, jolloin pistenotaatio käy työlääksi. Tietue voidaan avata WITH-DO-lauseella, minkä jälkeen viittaamiseen riittää pelkkä kentän nimi.

```
{Sama toiminta kuin edellä}
WITH aiti, isanp DO
  BEGIN
    ika := 35;
    palkka := palkka + 300.0;
    sos := '120349-763G'
  END
```

Pascalin erikoisuuksiin kuuluu niin sanottu **vaihteleva tietue**. Tietuemäärittelyssä voi edellä kuvatun kiinteän osan lisäksi olla vaihteleva osa, joka sisältää yhden tai useampia muunnelmia eli rakennevaihtoehtoja. Tällöin tietueeseen voidaan tallettaa tapauskohtaisesti aivan erilaista tietoa. Vaihtelevien tietueiden käyttö on hankalaa, eikä siihen tämän kurssin puitteissa syvennystä tarkemmin.

6.4 Joukot

Joukko koostuu samaa kantatyyppiä olevista alkioista. Kantatyyppi voi olla jokin ordinaalityyppi (ks. luku 3.1). Joukon alkiot ovat kaikki erilaisia eikä niiden järjestyksellä ole väliä. Niinpä kaksi joukkoa ovat samoja, jos ne sisältävät samat alkiot missä tahansa järjestyksessä. Joukkotyyppin esittelyn syntaksi on kuvassa 6.3.

Joukon kantatyyppi ei saa olla liian laaja, sillä mahdollisten alkioiden lukumäärä on useimmissa Pascal-järjestelmissä rajoitettu.

Joukko esitetään luettelemalla alkiot hakasuluissa pilkuilla erotettuna.

```
TYPE
  numerojoukko = SET OF [0..9];
VAR
  a, b: numerojoukko
```

```
...
a := [1,3,5,7,9];
b := [0..4] { = [0,1,2,3,4] }
```

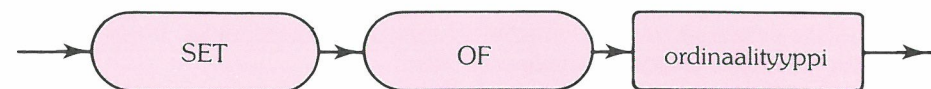
Joukoille on määritelty seuraavat joukko-opista tutut operaatiot:

```
Yhdiste eli unioni +
Leikkaus *
Erotus -
Joukkoon kuuluminen IN
Vertailu = ja < >
Osajoukko <= ja >=
```

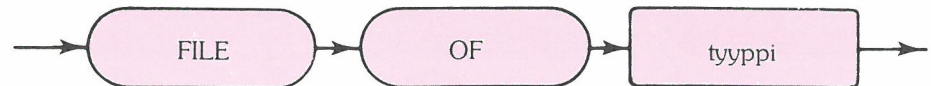
```
n2 = [1..3,7..9]
n3 = [3,5,7,9]
n2+n3 = [1..3,5,7..9]
n2*n3 = [3,7,9]
n2-n3 = [2,8]
3 IN n2 = True
n2 < > n3 = True
[3,5] <= n3
```

IN-operaattorin avulla voidaan testata kuuluko operandi joukkoon. Sen käyttö on kätevää esimerkiksi syöttötiedon oikeellisuuden tarkistuksessa.

Kuva 6.3



Kuva 6.4



```
VAR
  c: Char;
  REPEAT
    Read(c)
  UNTIL c IN '0'..'9'
```

6.5 Tiedostot

Tiedoston alkiot ovat kaikki samaa tyyppiä, mutta niiden lukumäärä ei ole ennalta määrätty. Tiedostotyyppin esittelyn syntaksi on kuvassa 6.4.

Tiedostotyyppinen muuttuja täytyy ohjelmassa avata ennen kuin sen alkioihin päästään käsiksi. Tiedoston alkioita käsitellään peräkkäisjärjestyksessä yhtä kerrallaan.

Tiedostoja käsitellään lähemmin kurssin seuraavassa osassa.

7. Dynaamiset tietorakenteet

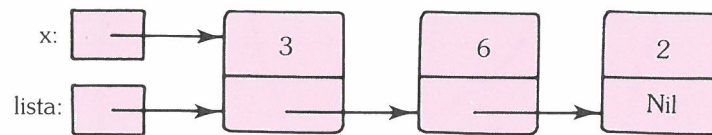
Seuraavaksi käsitellään erästä Pascalin erityispiirrettä, jota yksinkertaisten ohjelmien tekijä ei yleensä tarvitse.

Kuten olemme todenneet, on ohjelman muuttujat esiteltävä ennen kuin niitä voidaan käyttää. Jokainen muuttujan esittely varaa kuitenkin muistista tilaa.

Taulukko on tietorakenteena siinä mielessä kankea, että sen koko on tunnettava etukäteen. Jos esimerkiksi haluamme tehdä ohjelman, joka lajittelee ennalta määrittelemättömän määrän lukuja, meidän täytyy määritellä niin suuri taulukko, että kaikki luvut varmasti mahtuvat siihen. Tällainen taulukoiden käyttö tuhlaa muistia.

Dynaamiset muuttujat ovat muuttujia, jotka luodaan tarvittaessa ohjelman suorituksen aikana ja hävitetään, kun niitä ei enää tarvita.

Kuva 7.1



7.1 Dynaamisen muuttujan luonti.

Dynaamista muuttujaa ei esitellä tavalliseen tapaan nimellä, vaan siihen päästään viittaamaan ns. osoitinmuuttujan kautta. Tyyppiä T olevan dynaamisen muuttujan osoitin on tyyppiä $\hat{T}$. Osoitinmuuttuja on määriteltävä ennen kuin dynaaminen muuttuja voidaan luoda.

Dynaamisen muuttujan luonti tapahtuu standardiproseduurilla New.

```
TYPE
  lukupari = ARRAY 1..2 OF Integer;
VAR
  p:  $\hat{\text{lukupari}}$ ;
...
New(p);
```

Proseduurin New kutsu luo uuden tyyppiä lukupari olevan muuttujan ja asettaa osoitinmuuttujan p osoittamaan siihen. Voimme nyt viitata tähän luotun muuttujan lausekkeella $p^{\wedge}$, jolla tarkoitetaan sitä dynaamista muuttujaa, johon p osoittaa.

```
 $p^{\wedge}[1] := 3;$ 
 $p^{\wedge}[2] := 5;$ 
```

Osoitinmuuttujan arvoksi voidaan sijoittaa vakio Nil, joka ilmaisee, että osoitin ei osoita mihinkään dynaamiseen muuttujaan.

Dynaamisilla muuttujilla voidaan tehdä monimutkaisia linkitettyjä lista- ja puurakenteita. Niissä dynaaminen muuttuja on tyyppiltään tietue, jonka yksi tai useampi kenttä on osoitin samaan tietuetyyppiin.

Seuraavassa on ohjelma, joka lukee päätteeltä reaalityypin muodostaen niistä linkitetyn listan. Lukeminen päättyy nollaan, minkä jälkeen luvut tulostetaan käänteisessä järjestyksessä.

```
PROGRAM lista(Input, Output);
TYPE
  osoitin =  $\hat{\text{alkio}}$ 
  alkio = RECORD
    arvo: Real;
    seuraava: osoitin;
  END;
VAR
  lista, x: osoitin;
  r: Real;
  BEGIN
```

```
lista := Nil;
Read(r);
WHILE r < > 0.0 DO
  BEGIN
    New(x);
    x. $\hat{\text{arvo}}$  := r;
    x. $\hat{\text{seuraava}}$  := lista;
    lista := x;
    Read(r);
  END;
x := lista;
WHILE x < > Nil DO
  BEGIN
    Writeln(x. $\hat{\text{arvo}}$ );
    x := x. $\hat{\text{seuraava}}$ ;
  END;
END.
```

Kuvassa 7.1 on näytetty muuttujien arvot sen jälkeen kun ohjelmalle on syötetty luvut 2, 6 ja 3.

7.2 Dynaamisten muuttujien hävittäminen

Jokainen New-proseduurin kutsu varaa uutta muistitilaa. Jottei dynaamiseen muistinhallintaan varattu muistialue pääsisi loppumaan, pitäisi ohjelmassa tarpeettomat dynaamiset muuttujat hävittää eli niiden varaama tila vapauttaa. Dynaaminen muuttuja hävitetään standardiproseduurin Dispose kutsulla. Parametrina on osoitinmuuttuja, joka sisältää osoittimen hävitettävään muuttujaan.

```
VAR
  p:  $\hat{\text{ARRAY}} [1..1000] \text{ OF Integer};$ 
...
New(p); {Luodaan uusi taulukko}
p. $\hat{[23]}$  := 0; {Nollataan taulukon alkio}
...
Dispose(p); {Vapautetaan taulukon varaama muistitila}
```

Harjoitustehtäviä

- Keksi sopivat tietotyypit seuraaville tietorakenteille:
 - Korttipakan kortit (ilman jokereita)
 - Tiedot myytävästä huoneistosta (osoite, rakentamivuosi, omistaja, hinta).
 - Ruotsin kielen vokaalien joukko.

- Laadi ohjelma, joka lukee päätteeltä lukuja (viimeinen on 0), laatii niistä linkitetyn listan, ja tulostaa lopuksi luvut syöttöjärjestyksessä.

Kestävätkö elämänlankasi



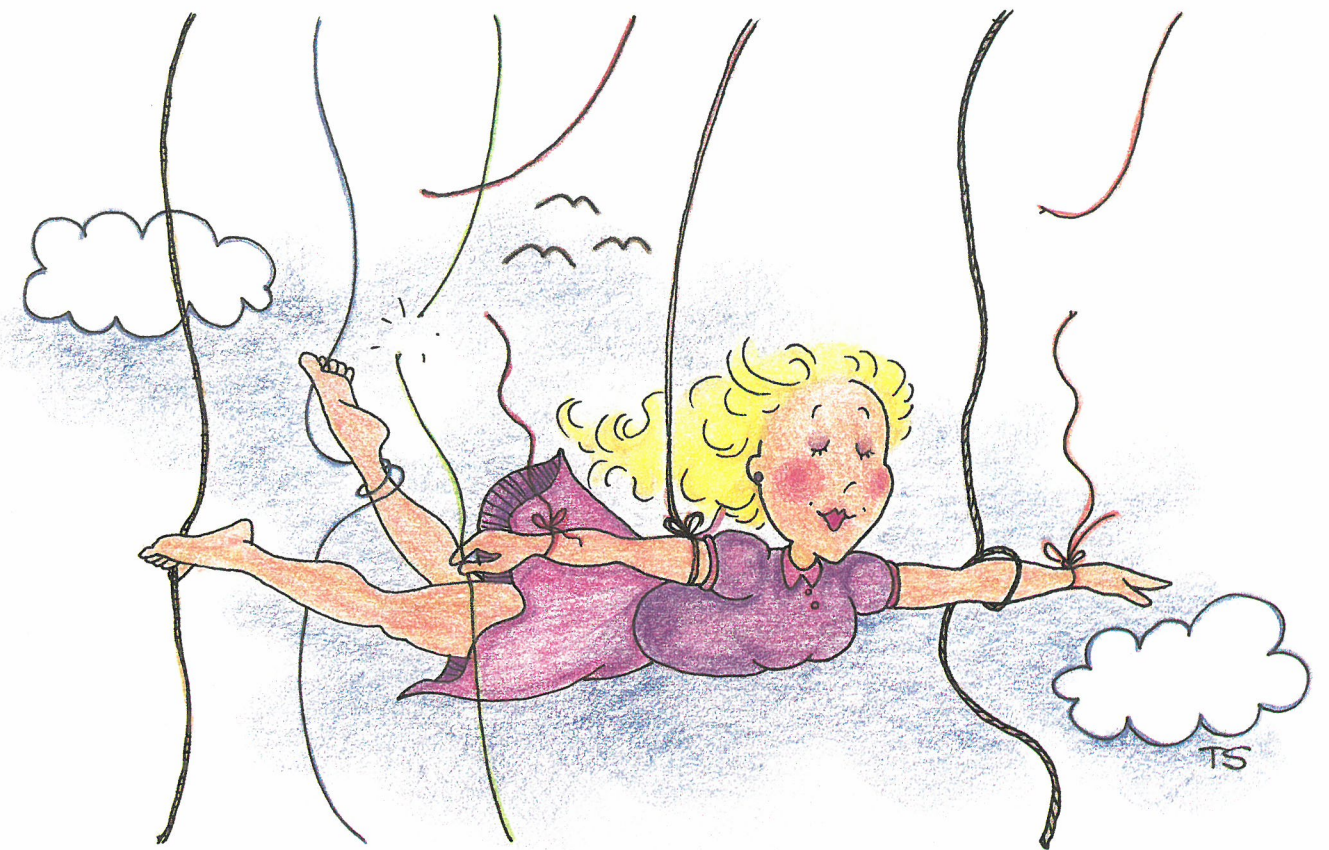
Elisabeth Lukas tutki yli tuhannen ihmisen henkisen hyvinvoinnin. Tulokset olivat yllättäviä. Kurjat elinolosuhteet, vaikea lapsuus tai raskaatkaan menetykset eivät näyttäneet olevan henkisen pahoinvoinnin syynä. Jakavana tekijänä oli se, tunsiko ihminen, että juuri hänen elämällään on jokin ainutlaatuisen tarkoitus vai oliko elämä päämäärätöntä ajelehtimista.

Hyvinvoinnin pahoinvointia

Voimiensa äärimmilleen ponnistamisella on huono kaiku ihmisten mielissä. Ankarat luonnonolot, sairaudet ja sodat ovat pakottaneet ihmisen venymään yhä uudelleen sietokykynsä ääri-rajalle. Yrittämisestä ja kamppailusta on tullut kiroksena. Ainaiseen olemassaolon taisteluun tuskastuneet sukupolvet ovat tehneet kaikkensa, jotta he tai edes heidän jälkeläisensä voisivat viettää mahdollisimman huolehtontaa ja yltäkylläistä elämää.

Kohonnut bruttokansantuote ei ole avannut portteja kauan kaivattuun onnelaan. Pikemminkin päinvastoin. Yhä useammat ovat alkaneet elämisen helppoutuessa kärsiä epämääräisestä masennuksesta sekä turhuuden ja tyhjyyden tunteesta. Logoterapian kehittäjä Viktor Frankl nimittää tätä kulovalkean nopeudella etenevää 'kansantautia' noogeeniseksi neuroosiksi. Ennen tästä riesasta näyttivät kärsivän vain vanhennevat elintasorouvat.

Vaikka elämänhalun puuttuminen voi aluksi näyttää mitättömältä pulmalta 'todellisten' katastrofien aiheuttaman hädän rinnalla, se on itse asiassa hengenvaarallinen tila. Ihminen pyrkii väistomaisesti hankkimaan elämänsä mielekkyyttä; joskus keinolla millä hyvänsä. Esimerkiksi monien nuorten arvelaan käyttävän huumeita, koska se antaa heille selkeän ja konkreettisen elämänsisällön. Viranomaisten jallitus, rahan hankkiminen rikoksella ja seuraavan 'matkan' suunnittelu antavat jatkuvaa, kaikki voimavarat nielevää tekemistä. Se puolestaan auttaa pakenemaan ahdistavia ajatuksia ja kaoottiseksi koettua maailmaa.



Elämässä pitää olla tsemppiä

Voidakseen hyvin ihminen siis tarvitsee ainakin yhden riittävän arvokkaan asian, jonka vuoksi ponnistella.

Monelle työ on ympärille kaikki muu elämänmeno rakentuu. Toisille lapset ovat kaikki kaikessa, jolloin elämänrytmi määräytyy näiden pikkukenkeiden tarpeiden sanelemana. Erilaiset urheilumuodot tarjoavat monelle areenan oman suorituskäytönsä ääri-rajoiden tutkailmiseen. Henkisemmät ihmiset omistautuvat koko sielullaan uskonnolle tai tieteelle. Jotkut taas löytävät taiteen ja politiikan ympäriltä mielekkään elämänsisällön.

Lähin tavoite pitää yllä piristävää odotusta. Tilanteen kehittymistä seurataan henkeä pidätellen. Tavoitetasoa nostetaan välittömästi, kun edellinen etappi on saavutettu. Henkinen tasapaino ja tyytyväisyys ovat itsestäänselvyyksiä niin kauan kuin edistymistä tapahtuu.

Ei kaikkea yhden kortin varaan

Elämällä on kuitenkin tapana generoida enemmän tai vähemmän satunnaisia yllätyksiä, jotka voivat hetkessä muuttaa koko elämän. Ennen pitkää työn sankari 'pääsee' eläkkeelle, lapsille omistautunut äiti huomaa jälkikasvun lähteneen maailmalle, himourheilija saa kohtalokkaan polvivamman, uskovainen menettää uskonsa, poliitikon edustama puolue joutuu skandaaliin.

Jos elämä on vain yhden langan varassa ja tuo lanka katkeaa, edessä ammottaa epätoivon kuilu.

Yksinomaan työhönsä uppoutuneen atk-ammattilaisen kannattaisi silloin tällöin vakavasti pohtia, mitä hän tekisi, jos ura yhtäkkiä katkeaisi sairauden tai jonkin muun ylivoimaisen syyn vuoksi. Esimerkiksi Ronald Reagan saattaa syystä tai toisesta päättää, että Suomi kuuluu tästä lähtien niihin maihin, joihin ei sotilaallisista syistä viedä tietotekniikkaa. Reaganille ja sikäläiselle teollisuudelle päätös on pikkujuttu, mutta monelle atk-ammattilaiselle se voi merkitä täydellistä romahdusta.

Jos hankittu taitopohja on kyllin laaja, uran katkeaminen aiheuttaa aluksi ilkeältä tuntuvan putoamisen kohti epätoivoa. Mutta melko pian riiputaan jo uuden kiinnostuksen varassa.

On tärkeää tajuta, että toimivaa varaköyhtä ei tarvitse pitää välttämättä 'kireällä' eli sen ylläpito ei välttämättä vaadi kovin paljoa aikaa. Pelkkä tietoisuus sen olemassaolosta mahdollistaa työhön keskittymisen turvallisilla mielin.

Pelastava turvaverkko

Lukasin tukimuksen mukaan kaikkein vahvimmita ovat ne, joilla on useita vankkoja kiinnekohtia elämään; monipuolinen ammattitaito, kiinteät ja rikkaat ihmissuhteet, antoisa harrastus jne. Kun muutoksia ja kriisejä sattuu yhdellä elämän alueella, muut kiinnepaikat pitävät kokonaistilanteen vakana. Voimia riittää suurenkin vaikeuden voittamiseen tai jopa tilanteen eduksikäntämiseen.

Suurimmassa vaaravyöhykkeessä ihminen on silloin, kun useat elämänlangat katkeavat samanaikaisesti. Tilastojen mukaan sairastumiset ajoittuvat usein suurten elämänmuutosten (läheisen kuolema, eläkkeelle jääminen, työpaikan vaihdos, avioliiton solmiminen, opiskelun lopettaminen jne.) yhteyteen. On äärimmäisen tärkeää, että elämänlankojen joukossa olisi ainakin yksi köysi, joka kestää kaikissa olosuhteissa. Esimerkiksi viime maailmansodan keskitysleireissä ihmiset menettivät miltei kaiken ulkoisen. Näistä kauhuista selvisivät yksinomaan ne, jotka tunsivat, että heillä on jokin elämäntehtävä, joka piti ehdottomasti suorittaa.

Aiheesta enemmän:

Lukas Elisabeth: Sinunkin elämälläsi on tarkoitus, KY 1984
Frankl Viktor: Ihmissyyden rajalla □

ORACLE-sovelluskehitin: Yhteensopivia sovelluksia eri laitteisiin

Tänään kehitettävien sovellusten on toimittava myös tulevaisuudessa. Eräs käypä ratkaisu tällaiseen sovelluskehitykseen on ORACLE.

ORACLElla voidaan kehittää helposti ylläpidettäviä laiteriippumattomia sovelluksia, jotka perustuvat voimakkaasti yleistyvään SQL-tiedonhallintakieleen.

ORACLE on relationaalinen tiedonhallintajärjestelmä, johon liittyy myös välineet sovellusten kehittämiseen. Se soveltuu erityisesti hajautettujen ja laiteriippumattomien järjestelmien kehittämiseen.

ORACLE perustuu IBM:n määrittelemään SQL-kieleen (Structured Query Language, lausutaan see-quel), joka on tiedonhallintajärjestelmien de facto -standardikieli sekä ANSI standardiehdotus relationaalisten tiedonhallintajärjestelmien kieleksi. SQL-kielellä voidaan mm. kysellä, ylläpitää, määritellä ja suojata tietokannan tietoja.

ORACLEN valmistaja on amerikkalainen Oracle Corporation. Yritys perustettiin vuonna 1977 välittömästi IBM:n SQL-tutkimusraporttien julkistamisen jälkeen ja ensimmäinen versio ORACLEsta julkistettiin 1979. Honeywell ja useat muut tietokonevalmistajat ovat valinneet ORACLEN omaksi tiedonhallintajärjestelmäkseen sen sijaan, että olisivat kehittäneet itse oman järjestelmänsä.

ORACLE on suunniteltu mahdollisimman laiteriippumattomaksi. Se on saatavissa moniin mikro-, mini- ja suurkankeisiin, mm. Stratukseen. Tähän mennessä ORACLEja on installoitu mini- ja suurkankeisiin maailmalla yli tuhat. ORACLE on myös yhteensopiva IBM:n SQL/DS ja DB2 -tiedonhallintajärjestelmien kanssa.

Tietokantana taulut

ORACLEN tietokanta koostuu tauluista (table), joille määritellään sarakkeita (column). Jokaiselle sarakkeelle määritellään tyyppi ja tarvittaessa enimmäispituus sekä 'pakollisuus'. Sarake voi olla tyypiltään joko merkkijono, numero tai päiväys. Tauluun lisätään tietoja riveinä (row), jotka koostuvat taulun sarakkeista vastaavista kentistä (field).

Kenttä on tietokannan pienin osa, johon voidaan suoraan viitata SQL-lauseella. Kenttiin viitataan aina nimen eikä sijainnin perusteella. Sen sijaan riveihin viitataan aina kenttien arvojen perusteella. Näin ollen SQL-lauseet ja niistä rakennetut sovellukset ovat riippumattomia tietojen fyysisestä sijainnista.

Aiemmin luodun taulun sarakkeita voidaan leventää sekä liittää tauluun uusia sarakkeita. Ja ilman, että taulun tietoja tarvitsisi ladata tai ORACLElla tehtyjä sovelluksia tarvitsisi muuttaa tai kääntää.

Mikä tahansa SQL-kyselylause on määriteltävissä virtuaaliseksi tauluksi (view), jota voidaan käyttää muissa kyselyissä tavallisten taulujen tapaan. Virtuaalista taulua ei päästä kuitenkaan päivittämään, sillä sitä ei ole talletettu kuten tavallista taulua.

SQL-kieliset lauseet määrittelevät vain mitä tietokannalle halutaan tehdä, ja SQL-tulkki päättää miten lause suoritetaan. SQL-tulkki eristää loogisen tietokannan fyysisestä tietokannasta ja mahdollistaa tietokannan virittämisen niin, että SQL-lauseita ei tarvitse muuttaa.

Hakemistot nopeuttavat hakuja

Taululle voidaan luoda hakemistoja (index), jotka nopeuttavat tietojen etsimistä taulusta. Hakemiston avain voi koostua useasta sarakkeesta mielivaltaisessa järjestyksessä. Voidaan myös määritellä, salliiko hakemisto samaavaimisuuden (synonyymiavaimet) vai ei. Hakemistorakenteena on tasapainotettu binääripuu.

Taulut, joiden tietoja käytetään usein yhdessä, voidaan haluttaessa tallettaa fyysisesti samoihin lohkoihin (cluster). Esimerkiksi osasto- ja työntekijä-tilut on järkevää tallettaa siten, että yhteen fyysiseen lohkoon talletetaan sekä osaston että sen työntekijöiden tiedot. Tällöin tietoja haettaessa tarvitaan mahdollisimman vähän levyhakuja. Tällä tavalla tiedot on yleensä talletettu vain tehokkaissa hierarkkisissa ja verkkorakenteisissa tiedonhallintajärjestelmissä.

ORACLE minimoi levyhakujen määrää pitämällä usein käytettyjä tietoja puskureissaan. ORACLE hallitsee täysin käyttöjärjestelmästä riippumatta tietojen tallettamismenetelmät ja kykenee tehokkaasti hakemaan ja tallettamaan tietoja sekä optimoimaan levytilan käyttöä. Tiedot talletetaan pakatussa muodossa.

ORACLE sallii useiden käyttäjien käsittelevän samaa taulua yhtaikaisesti, mutta lukitsee muutettavat rivit muutostapahtuman ajaksi. Poikkeuksena ORACLEN MS-DOS -versio PC-ORACLE, joka sallii vain yhden käyttäjän kerrallaan.

Monipuoliset suojaukset

Jokaisella käyttäjällä on käyttäjätunnus ja salasana. Tietokannan taulua voi käyttää vain taulun luoja, ellei hän anna muille käyttäjille oikeuksia käyttää taulua. Oikeudet saa määrittellä erikseen tietojen kyselyyn, lisäämiseen, muuttamiseen ja poistoon.

Käyttäjän tietokantaan tekemät muutokset jäävät pysyvästi voimaan ja muille käyttäjille näkyviksi vasta, kun tietokannalle annetaan hyväksymiskäskey. Jos tietokannalle annetaan perumiskäskey tai jos järjestelmä kaatuu ohjelmisto- tai laitevirheeseen, perutaan edellisen hyväksymiskäskyn jälkeiset muutokset, jotta tietokanta pysyisi johdonmukaisessa tilassa. ORACLEN lukitukset estävät kahta käyttäjää päivittämästä samaa kyselyyn, lisäämiseen, muuttamiseen ja poistoon.

Käyttäjän tietokantaan tekemät muutokset jäävät pysyvästi voimaan ja muille käyttäjille näkyviksi vasta, kun tietokannalle annetaan hyväksymiskäskey. Jos tietokannalle annetaan perumiskäskey tai jos järjestelmä kaatuu ohjelmisto- tai laitevirheeseen, perutaan edellisen hyväksymiskäskyn jälkeiset muutokset, jotta tietokanta pysyisi johdonmukaisessa tilassa. ORACLEN lukitukset estävät kahta käyttäjää päivittämästä samaa riviä yhtaikaisesti.

Tietokannasta voidaan ottaa ajoittain varmuuskopio ja tulostaa loki tietokantaan tehdyistä muutoksista. Varmuuskopion ja lokin avulla pystytään tietokanta esimerkiksi levyn tuhoutuessa palauttamaan tuhoutumista edeltäneeseen tilaan.

Soveltuu hajautukseen

ORACLE tarjoaa palvelut hajautettujen sovelluksien rakentamiseen.

ORACLE LINK -apuohjelmalla siirretään tietoja tietokannasta toiseen. Tietokannat voivat sijaita jopa erilaisilla laitteistoilla.

ORACLE NET mahdollistaa tietojenkäsittelyn hajauttamisen usealle laitteelle. Tällöin sovellus voi sijaita eri laitteella kuin sen käyttämä tietokanta. ORACLE ei ole vielä kuitenkaan täydellinen hajautettu tietokanta, koska sovelluksen tai käyttäjän täytyy tietää missä tietokannassa tiedot sijaitsevat.

Tietohakemisto

Tietohakemisto on tiedonhallintajärjestelmän omistama tietokannan osa. Se sisältää tietoja järjestelmän käyttäjästä, käyttäjien luomista tauluista, taulujen sarakkeista ja hakemistoista sekä käytöluvista.

Tiedonhallintajärjestelmä vastaa tietohakemiston ylläpidosta, mutta käyttäjät voivat kysellä sieltä itseään koskevia tietoja SQL-kieltä käyttäen.

Tietohakemiston ansiosta SQL-tulkki näkee jatkuvasti ajan tasalla olevan kuvauksen tietokannan rakenteesta. Niinpä tietokannan rakenteeseen tehtävät muutokset ovat välittömästi käytettävissä.

Sovelluskehitysvälineet

ORACLEen sisältyy joukko sovelluskehitysvälineitä, joita ovat näytönhallintavälineet, kyselykieliliitännä, raporttigeneraattori ja esikääntäjä. ORACLElla tehty sovellukset ovat aina riippumattomia tietokannan fyysisestä tallettamisesta, koska kaikki sovelluskehitysvälineet käyttävät SQL-kieltä.

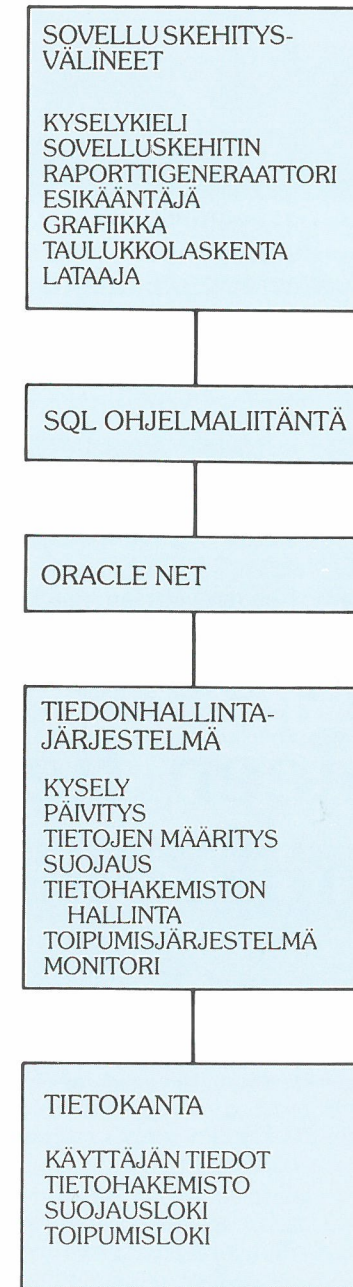
Näytönhallintavälineellä määritellään helposti näyttöjä, joilla voi kysellä, selata, lisätä, muuttaa ja poistaa tietokannan tietoja. Näytön kentille voi määritellä tarkistuksia, muotoiluja, otsikoita ja ohjetekstejä.

Näyttö voi muodostua useasta lohokosta, joista kukin vastaa yhtä tietokannan taulua. Lohkojen kenttien välillä on mahdollista välittää tietoja. Näytön lohkolle voidaan määrittellä liipaisimia (trigger), jotka ovat lohkolle tehtävän kyselyn, lisäyksen, muutoksen tai poiston yhteydessä suoritettavia SQL-lauseita. Ohjauslohko on näytön lohko, jolla annetaan muuttuvia oletusarvoja muiden lohkojen kentille. Lohkosta toiseen siirrytään joko suoraan tai valikon kautta.

Näyttö voi olla myös monisivuinen, mutta vain yksi sivuista on näkyvissä kerrallaan. Näkyvillä oleva sivu määrittyy kohdistimen sijainnin mukaan.

Kyselykieliliitännällä voidaan tehdä SQL-kielellä tietokantakyselyjä, muotoilla kyselyjen tuloksia raporteiksi sekä rakennella komentomakroja. Raporttigeneraattori puolestaan soveltuu monipuolisten muotoiltujen raporttien tekoon.

Esikääntäjällä pystytään kääntämään ohjelmointikielellä kirjoitettuja ohjelmia, joihin on upotettu SQL-lauseita. Esikäännöksen tuloksena saadaan käännettävä ohjelma, jossa SQL-lauseet on muutettu aliohjelmakutsuiksi. Tällä hetkellä esikäännökseen soveltuvia ohjelmointikieliä ovat C, Cobol, Fortran, Pascal ja PL/1. □



ORACLE-KONSEPTI

Mikro-Mercur:

Taloussuunnittelua mikrolla

Mercuria on käytetty taloussuunnitteluun jo useita vuosia suurtietokoneilla. Mercurin hyväksi koetut piirteet on nyt siirretty mikeroon. Ja myös MikroMikko 2:lle.

Mercurin kehitys käynnistyi Ruotsissa 70-luvun puolivälissä. Mercurin valmistaja on Mercur Planeringsspråk AB. "Mercur on suunniteltu ensisijaisesti Pohjoismaisten yritysten taloussuunnitteluun. Pohjoismaissa on yli tuhat Mikro-Mercurin käyttäjä. Suomalaisia käyttäjiä ovat mm. SKOP, Partek, Kemi Oy, Veho, Helvar, Ahlström, Åbo Akademi ja Turun kauppakorkeakoulu", kertoo Nokia Informaatiojärjestelmien Mercur-tuotepäällikkö Mikko-Jussi Suonenlahti.

Mikro-Mercur on käytettävissä paitsi yksin MikroMikko 2:lla myös yhdessä Nokian Laskentakeskuksen suurtietokoneiden kanssa. Tällöin tietoja voidaan siirtää mikrolta suurtietokoneelle ja takaisin. Ja siten hajauttaa tietojenkäsittelyä.

Koska taloussuunnittelussa tarvitaan pohjaksi toteutuneita lukuja ja toisaalta verrataan toteutunutta suunnitelmiin, on Mercuriin tehty liittymät tietojen siirtämiseksi Mercurin ja muiden tietojärjestelmien välillä. Mercur keskustelee esimerkiksi yrityksen laskentatoimen järjestelmien sekä vaikka Lotuksella tai Multiplanilla tehtyjen taulukoiden kanssa.

Suonenlahti sanoo Mikro-Mercurin toimivan 512 kilotavun keskusmuistilla varustetussa MikroMikko 2:ssa. Mercurin käyttöön suositellaan umpilevyä (Winchester). MS-DOS -käyttöjärjestelmästä edellytetään versiota 2.0. "Tietysti tarvitaan myös kirjoitin paperitulostusta varten. Laskelmien ja analyysien tulokset on pystyttävä tulostamaan paitsi kuvaruudulle myös kirjoittimelle tai piirturille", Suonenlahti lisää.

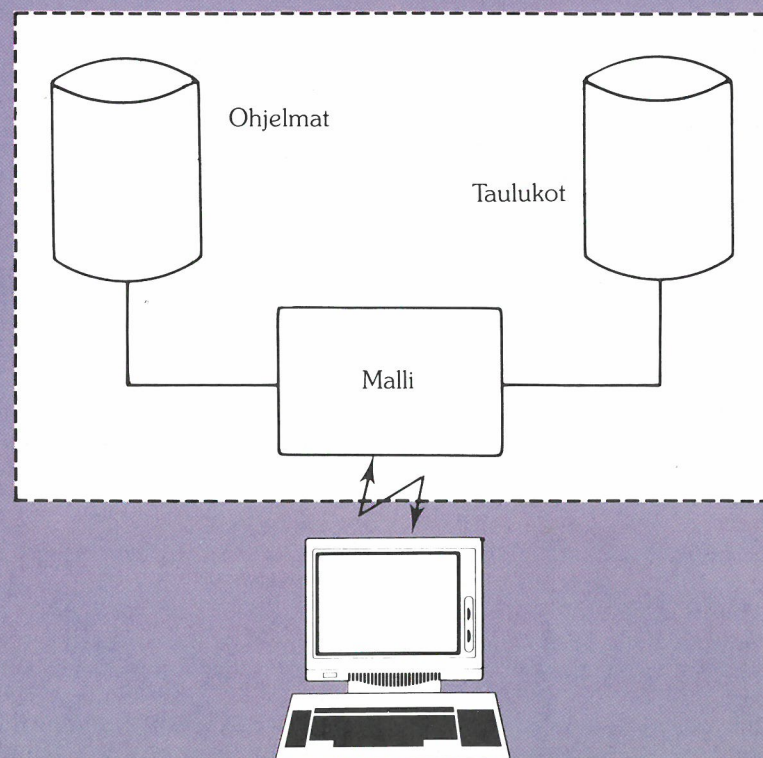
Kieli sovelluksien tekoon

Mercur on itse asiassa taloussuunnittelu-kieli, jonka tehokkailla komennoilla käsitellään Mercurin tietokantaa. Mercuria voi pitää myös neljännen sukupolven ohjelmointikielenä tai sovellushittimenä.

Mercur perustuu matriisitietokantaan, joka on relaatiotietokannan erikoismuoto. Mercurin tietokanta koos-

tuu taulukko- ja ohjelmatietokannoista. Yhteen taulukkotietokantaan voi Mikro-Mercurilla tallettaa 512 taulukkoa, joissa kussakin voi olla 6000 elementtiä. Ja automaattinen haku on mahdollista kohdistaa kuuteen tietokantaan samanaikaisesti.

"Mikro-Mercur-sovelluksissa voi olla käytössä yhtäaikaa yli 3000 taulukkoa ilman, että taulukoiden suuri määrä vaikeuttaisi sovelluksen hallintaa. Tämä on mahdollista siksi, että Mercur



käyttää Winchester-levyä virtuaalisesti", Suonenlahti kertoo.

Jokaisella käyttäjällä voi olla oma taulukkotietokantansa, johon hän taltioi omia taulukoitaan. Taulukoissa on erotettavissa rivit, sarakkeet, otsikot, tekstiosa ja alleviivaukset. Niinpä taulukko vastaa tuttua raporttia, mikä osaltaan auttaa käyttäjää suoriutumaan tietojen syötöstä ja päivityksestä.

Taulukkotietokannan käsittely on helppoa Mercurin komennoilla. Mercurin komennoilla luot uusia taulukoita, päivität taulukoitasi, selaillet taulukoita näytöllä, yhdistelet taulukoita sekä tulostat haluamiasi raportteja, ja jopa graafisessa esitysmuodossa ja värein. Raportit tulostetaan joko suoraan taulukosta tai kokoamalla tietoja yhteen useammasta eri taulukosta. Laskelmat ja simuloinnit ovat mahdollisia suoraan kuvaruudulla.

Taulukoiden sisältöjä pystyy muuttamaan yksittäisillä komennoilla. Taulukon voi myös hakea näytölle ja tehdä siinä tarvittavat päivitykset. Taulukon ei tarvitse mahtua kuvaruudulle yhdellä kertaa. Suuria taulukoita selataan MikroMikko 2:n toimintonäppäimiä käyttäen.

Mercurin komennoilla voidaan tehdä erilaisia laskutoimituksia taulukoiden tiedoille. Tietoja saatetaan laskea yhteen, vähentää toisistaan, kertoa ja jakaa. Mercuriin on rakennettu lisäksi monia laskelmien teossa tarvittavia funktioita. Funktioita on valmiina esim. lineaariseen regressioon, matriisikäsitelyyn, kuljetusten optimointiin, sekä annuutein, nykyarvon ja sisäisen koron laskemiseen.

Mercurin analysointipiirteitä hyödyn-täen saadaan nopeasti vastaus esim. seuraavanlaisiin kysymyksiin:

- 'mitä jos' -kysymykset: miten hinnanmuutos 200,- vaikuttaa ennustettuun myyntiin?
- herkkyysoanalyysit: mikä on yrityksen tulos, jos tuotteen hinta vaihtelee + tai - 5 % budjetoidusta?
- iterointi taaksepäin: millä myyntillä saavutetaan 30 %:n myyntikate nykyisellä kustannusrakenteella?

Yhdistelemällä Mercurin komentoja ohjelmoidaan ns. Mercur-sovelluksia eri käyttötarkoituksiin. Sovellukset sijaitsevat omassa ohjelmatietokannassaan, erillään taulukkotietokannoista. Sama sovellusohjelma on siten useiden eri käyttäjien käytettävissä. Esimerkiksi eri kustannuspaikat tai osastot käyttävät samaa budjetoitsovellusta.

Käyttäjäystävällisyyttä

Mercur sisältää tulkin, joka tarkistaa käyttäjän komennon oikeellisuuden ja ilmoittaa virheistä välittömästi. Näin

säästetään aikaa sovellusten teossa.

Sovellukset on mahdollista rakentaa hyvinkin helpokäyttöiseksi suomenkielillä valikkorakenteilla. Tällöin Mercurin päivittäinen käyttö on yksinkertaista. Osoitetaan valikolta vain seuraavaksi tehtävä työ. Niinpä sovelluksen käyttäjän ei tarvitse tuntea Mercurin varsinaisia käskyjä lainkaan.

Mercuriin on rakennettu suomenkieliset ohjeet, jotka saadaan näytölle HELP-näppäimellä. Tätä elektronista opasta käyttäjä voi selata ja löydettyään haluamansa tiedot palata jatkaamaan työskentelyään siitä mihin jäi.

Soveltuu moneen

Vaikka Mikro-Mercur on erikoistunut taloussuunnitteluun, se soveltuu myös monenlaiseen analysointiin ja taulukkolaskentaan.

Taloussuunnittelu on yleensä kokeilua eri vaihtoehdoilla. Asetetaan tulokseen vaikuttaville tekijöille erilaisia arvoja ja katsotaan, millaiseksi tulos muodostuu.

Toisaalta voidaan asettaa ensin tulos ja katsoa, millaisia tulokseen vaikuttavien tekijöiden tulee olla. Mercur mahdollistaa tällaiset kaksisuuntaiset laskelmat. Mercur-taloussuunnittelu pystyy hallitsemaan myös yrityksen kaikki tulosot. Tyypillisin esimerkki lienee budjetointi.

Yritysten budjetointikierros on yleensä monimutkainen ja aikaa vievä. Budjetointiin osallistuvat henkilöt syöttävät omat budjettitietonsa päätteelle, minkä jälkeen budjettia voidaan simuloida päätteellä ja analysoida tuloksia. Jokaista osastoa tai kustannuspaikkaa voi vastata yksi taulukko. Mercur konsolidoi taulukot halutulle hierarkiatasolle asti. Yhtä helposti käy myös yrityksen kustannuslajikohtaisten raporttien, kuten palkkaraporttien, laatiminen.

Organisaation hierarkian esittäminen ja muuttaminen on joustavaa Mercur-sovelluksissa, koska Mercurissa on erillinen taulukko kuvaamassa organisaation hierarkiaa. Niinpä organisaatiomuutokset voidaan päivittää ilman hankalia ohjelmistomuutoksia. Niin ikään rinnakkaisten organisaatorakenteiden hallitseminen on mahdollista ilman eri ohjelmistoja.

Tuotelaskelmien teossa saadaan yhdellä Mercur-käskyllä laskelma tietylle tuotteelle. Kun annetaan Mercurin tehdä sama laskelma niin monelle tuotteelle kuin halutaan, voidaan lopuksi liittää laskelmat yhteen yksinkertaisella taulukkokäsittelyllä.

Mercurilla voidaan toteuttaa yritys-malli, joka sisältää esimerkiksi yrityksen budjetin, PTS-mallin, likviditeettisuunnitelman, tunnusluvut, investointilaskelmat, kassavirta-analyysit, tulos- ja tase-ennusteet. Mercurilla voi tehdä

erilaisia analyysejä yritysmallista ja samalla tutkia mallin riippuvuuksien vaikutuksia eri tilanteissa.

Suonenlahden mukaan jokaisella yrityksellä on tulevaisuudessa jokin Mercurin kaltainen taloussuunnittelu-kieli päätöksenteon tukijärjestelmien osana. "Mieluimmin tietysti Mercur", Suonenlahti lisää.

Koulutusta saatavissa

Mercur-koulutuksen saa joko NITECistä (Nokian tietotekniikan koulutuskeskuksesta) tai yrityskohtaisilta kursseilta. Jo kahden päivän koulutuksen jälkeen käyttäjä pystyy tekemään suomenkielisiä valikko-ohjattuja Mercur-sovelluksia.

Mercurin mukana toimitetaan sekä käyttäjän käsikirja että oppikirja. Kirjat on suunniteltu nopeasti omaksuttaviksi. Käyttäjän käsikirja on laaja sisältäen runsaasti esimerkkejä. Oppikirja puolestaan sisältää harjoituksia itseopiskeluun.

Vuonna 1981 perustettiin Ruotsissa Mercur-käyttäjähdistys, jonka keskeisin tehtävä on olla Pohjoismaiden Mercur-käyttäjien keskustelufoorumi. Käyttäjät kokoontuvat säännöllisesti käsittelemään erilaisia teemoja. Näissä tilaisuuksissa heillä on mahdollisuus solmia kontakteja sekä saada tietoutta Mercurin käytöstä. □

MPSINDEXillä monikäyttäjäsovelluksia MikroMikko 2 -lähiverkkoon

Lähiverkoissa on pystyttävä käsittelemään samaa tiedostoa yhtäaikaisesti usealta eri työasemalta. Lähiverkkoihin soveltuvista tiedonhallintajärjestelmistä on kuitenkin huomattava pula markkinoilla. Niinpä Nokia on kehittänyt oman MPSINDEX-tiedostojenkäsittelyohjelmiston MikroMikko 2 -lähiverkkoon.

Millainen MPSINDEX sitten oikein on?

MS-DOS -yleisohjelmistot toimivat periaatteessa sellaisenaan paitsi MikroMikko 2:ssa myös MikroMikko 2 -lähiverkossa. Nämä yleisohjelmistot ovat kuitenkin yhden käyttäjän järjestelmiä. Ne pystyvät käsittelemään yhteistä massamuistia siten, että kukin käyttäjä käsittelee eri tiedostoa.

Jotta MikroMikko 2 -lähiverkosta olisi todella hyötyä, tarvitaan monikäyttäjäsovelluksia. Tällaisten verkkosovellusten tekoon on Nokia kehittänyt MPSINDEX-ohjelmiston. MPSINDEX yhdessä MPSFORMS-näytönkäsittelijän kanssa muodostaa tehokkaan työvälineen verkkoympäristöön.

MPSINDEX on tiedostojenkäsittelyjärjestelmä, joka mahdollistaa yhteisten moniavaintiedostojen käytön. MPSINDEXissä voidaan tiedostot lukita sekä tiedosto- että tietueitasolla ja siten estää yhtäaikaiset päivitykset.

Työkalu sovellusohjelmointiin

MPSINDEX on sovellusohjelmoijan apuväline. MPSINDEXiä käytetään sovellusohjelmista proseduurikutsuin. Tällä hetkellä MPSINDEXiä voi kutsua mm. Pascalista, Basicista, Cobolista, Fortranista ja C-kielestä.

Sovellusohjelman kanssa linkitetään käytetyn ohjelmointikielen MPSINDEX-liitäntämoduli (MPSxxx.OBJ), joka sisältää ko. kieliliitännän mukaiset proseduurikutsut. Liitäntämodulit toimitetaan MPSINDEXin jakelulevykkeellä.

Varsinainen MPSINDEX-ohjelmisto on laiteohjainohjelman tyyppinen keskeytyspohjainen ohjelma, joka ladataan muistiin aina käyttöjärjestelmän mukana. Niinpä MPSINDEX on installoituva käyttöjärjestelmään ennen kuin se voidaan ottaa käyttöön. Verkkokäyttöjärjestelmä NETOS on tilattavissa MPSINDEXillä varustettuna, jolloin MPSINDEXiä ei tarvitse installoida.

MPSINDEX voidaan installoida myös MS-DOSiin. Tämä ns. yhden käyttäjän MPSINDEX on toiminnallisesti täysin yhteensopiva verkkoversion kanssa. Näin ollen kaikki MPSINDEXiä käyttävät sovellukset toimivat sellaisenaan sekä yksittäisessä MikroMikko 2:ssa että MikroMikko 2 -lähiverkossa.

MPSINDEXin avulla voidaan lukea tiedoston tietueita avaimen mukaan sekä perättäisesti että hajasaantisesti. MPSINDEXillä on mahdollista myös mm. päivittää, lisätä ja poistaa tietueita sekä poistaa ja luoda uusia avaimia.

Jopa 16 käyttäjää samanaikaisesti

MPSINDEX-tiedostolla voi samanaikaisesti olla enintään 16 käyttäjää. Ja yhdellä käyttäjällä voi samaan aikaan olla avoinna korkeintaan 16 tiedostoa. Tosin yhden käyttäjän MPSINDEXissä MS-DOS rajoittaa avoinna olevien tiedostojen määrän viiteen.

MPSINDEX-tiedosto on käyttäjän kannalta looginen kokonaisuus. Rakenteellisesti se koostuu kuitenkin datatiedostosta, hakemistoista eli ns. indeksitiedostoista sekä määrittelytiedostosta.

Määrittelytiedosto (.DEF) sisältää data- ja indeksitiedostoja koskevaa määrittelytietoa.

Datatiedosto (.DAT) koostuu varsinaisista 'käyttäjän tietueista'. Fyysisesti tietueet ovat tiedostossa aikajärjestyksessä eli siinä järjestyksessä kuin niitä on tiedostoon lisätty. Tietueiden enimmäispituus on 2048 tavua ja tietueita voi yhdessä datatiedostossa olla enintään n. 8,4 miljoonaa. Käytännössä levyaseman kapasiteetti on kuitenkin ensimmäinen tiedoston kokoa rajoittava tekijä.

Datatiedoston tietueella voi olla enintään sata kenttää. Kenttä voi olla tyyppiltään numeerinen tai merkkijono.



Enintään kahdeksan avainta

Kullekin datatiedostolle voidaan määrittellä kahdeksan avainta. Ja avain voi koostua kolmesta kentästä, joiden ei tarvitse sijaita tietueella peräkkäin. Avaimen enimmäispituus on 128 tavua.

Kutakin datatiedoston avainta vastaa yksi indeksitiedosto (.IX1 - .IX8), johon talletetaan avaimen mukainen tietuejärjestys. Indeksitiedostossa käytetään B-puurakennetta.

Avainta määriteltäessä kerrotaan MPSINDEXille, sallitaanko synonyymiavaimet eli voiko useammalla datatiedoston tietueella olla sama avaimen arvo. Jos sama-avaimisuutta ei sallita ja ollaan lisäämässä sama-avaimista tietuetta, MPSINDEX estää lisäyksen.

Monipuolista tiedostonkäsittelyä

Sovelluksen MPSINDEX-proseduurikutsuilla saadaan MPSINDEX tekeään erilaisia tiedosto-operaatioita. Proseduurikutsuihin liittyvillä parametreilla täsmennetään kutsua.

Sovellusohjelma voi MPSINDEXillä luoda kokonaan uuden MPSINDEX-tiedoston tai tehdä jo olemassa olevalle uusien hakemistojen (indeksitiedostojen). Kutsumalla MPSINDEXiä sovellusohjelma voi myös poistaa kerralla koko MPSINDEX-tiedoston tai vain yksittäisiä hakemistoja. Niin ikään MPSINDEXin kutsulla saadaan vioittunut MPSINDEX-tiedosto uudelleen käyttöön.

Ennen kuin MPSINDEX-tiedostoa aletaan sovelluksessa käsitellä, se on aina ensin avattava. Avauksessa tiedoston voi varata omaan käyttöön, jolloin muut eivät voi siihen koskea.

Avauksen lisäksi MPSINDEX-palveluihin kuuluvat myös tiedoston sulkeminen, tietueen luku, kirjoitus ja poisto. Tietueen voi lukea joko hajasaantisesti tai peräkkäisesti avaimen mukaan nousevassa tai laskevassa järjestyksessä. Tietueen luvun yhteydessä voidaan tietue lukita muilta käyttäjiltä.

Tehtäessä muutoksia tiedostoon MPSINDEX päivittää automaattisesti hakemistot.

Apuohjelmia

MPSINDEX-järjestelmään kuuluu joukko MPSINDEX-tiedostojen käsittelyyn liittyviä apuohjelmia.

Apuohjelmat on yhdistetty MPSMENU-ohjelmaksi, jonka valikolta käyttäjän on helppo käynnistää haluamansa toiminto. Apuohjelmia voi tuki kutsua myös suoraan MS-DOS -tasolta.

MPSFUTIL-ohjelmalla luodaan uusia MPSINDEX-tiedostoja tai rakennetaan MPSINDEX-tiedostolle uusia hakemistoja. Ohjelmalla saadaan vioittuneet MPSINDEX-tiedostot uudelleen käyttökuntoon. Lisäksi MPSFUTILilla voidaan poistaa käytöstä koko MPSINDEX-tiedosto tai vain yksittäisiä hakemistoja.

MPSDUMP-ohjelmalla on mahdollista tutkia datatiedoston sisältöä, joka tulostetaan näytölle tai kirjoittimelle.

MPSLOAD-ohjelma konvertoi ASCII- ja binääritiedostoja MPSINDEX-tiedostoiksi. Ohjelmalla voi myös luoda jo olemassa olevasta MPSINDEX-tiedostosta rakenteeltaan uuden MPSINDEX-tiedoston.

Kaikki apuohjelmien toiminnot sisältävät FILEDEF-ohjelmaan. FILEDEF on MPSFORMS-näytönkäsittelijän apuohjelma MPSINDEX-tiedoston luontiin ja rakenteen muuttamiseen. FILEDEF-ohjelmassa voidaan hyödyntää suoraan ns. tietotyyppikuvauksia, joita määritellään DICTDEF-ohjelmalla. □

MPSFORMS-näytönkäsittelijä: Ohjelmointityökalu MikroMikko 2:een

Näytönkäsittelyn ohjelmointi sovelluksiin on työlästä perinteisillä ohjelmointikielillä. MikroMikko 2:een on kehitetty avuksi oma näytönkäsittelijä, MPSFORMS. Se on mainio työkalu sovellusohjelmoijalle.

MPSFORMS on kehitetty Nokia Informaatiojärjestelmissä ja se on suunniteltu ensisijaisesti näytönkäsittelyn apuvälineeksi. MPSFORMSin näytönkäsittelytoimintoja pyydetään sovellusohjelmasta proseduurikutsuin. MPSFORMSia voi kutsua mm. Pascalista, C-kielestä, Basicista sekä Cobolista.

Näytönkäsittelijä on ohjelmoitu pääosin C-kielellä. Ainoastaan muutama alimman tason proseduurit on toteutettu assembler-kielellä.

Näytönkäsittelijän suunnittelussa olivat lähtökohtina mm. monipuolisuus ja joustavuus. MPSFORMSin oli pystyttävä kaikkiin tavanomaisiin näytönkäsittelytoimintoihin. Käsittelijään oli voitava helposti liittää myös erikoistoimintoja. Käsittelijän piti olla myös nopea, pienikokoinen ja ennen kaikkea helpokäyttöinen. Nämä vaativat ja ristiriitaisiltakin tuntuvat tavoitteet on toteutettu MPSFORMSissa.

Näyttökuvaukset erillään sovelluksesta

MPSFORMSin peruskäsite on näyttö-

lomake. Näyttölomake koostuu ohjeteksteistä (ns. näyttölomakkeen pohja) ja datakentistä. Varsinainen tiedon syöttö ja tulostus näytölle tapahtuu näyttölomakkeen datakenttien kautta siten, että kohdistin liikkuu ainoastaan datakentissä.

Eräs näyttölomakkeen erikoistapaus on ns. HELP-lomake, joka sisältää vain ohjetekstejä. HELP-lomakkeen voi liittää tavallisen näyttölomakkeen datakenttään. Tällöin tiedon syötössä ko. datakentän kohdalla tietyn näppäimen (ns. HELP-näppäimen) painallus tuo näytölle vastaavan HELP-lomakkeen. Seuraava näppäinpainallus poistaa HELP-lomakkeen palauttamalla alkuperäisen näytön. Näin HELP-lomakkeilla voidaan välittää käyttäjälle kuhunkin tilanteeseen liittyviä ohjeita.

Ennen kuin MPSFORMSia voidaan käyttää sovellusohjelmasta on käytettävät näyttölomakkeet kuvattava ns. lomaketiedostoihin. Kaikki näyttölomakkeiden tekstit, joita sovellus käyttää, sijaitsevat lomaketiedostossa. Esimerkiksi yhteen lomaketiedostoon voidaan sijoittaa suomenkieliset ja toiseen vastaavat ruotsinkieliset näytöt. Niinpä

pelkästään lomaketiedostoa vaihtamalla saadaan sovelluksesta erikielisiä versioita.

Näyttölomakkeet kuvataan helppokäyttöisellä FORMSDEF-näytönmaalausohjelmalla. Ohjelmalla määritellään erilaisia näyttölomaketta koskevia tietoja, esimerkiksi lomakkeen ohjetekstit ja datakentät attribuutteineen (ominaisuuksineen).

Sovellusohjelma voi käyttää useampia näytölle tulostettuja näyttölomakkeita samanaikaisesti. Näyttölomakkeet voidaan jopa limittää näytölle siten, että aktiivinen näyttölomake tulostetaan 'nipun päällimmäiseksi'. Tällöin aikaisemmin tulostetut näyttölomakkeet saattavat jäädä kokonaan tai osittain näkyviin.

Koska sovelluksessa käytettäviä lomakkeita pidetään sovellusohjelman työmuistissa, saadaan lomakkeet näytölle nopeasti.

Eri tason toimintoja

MPSFORMSin toiminnot voidaan jakaa eri tasoihin, jotka ovat keskenään yhteensopivia.

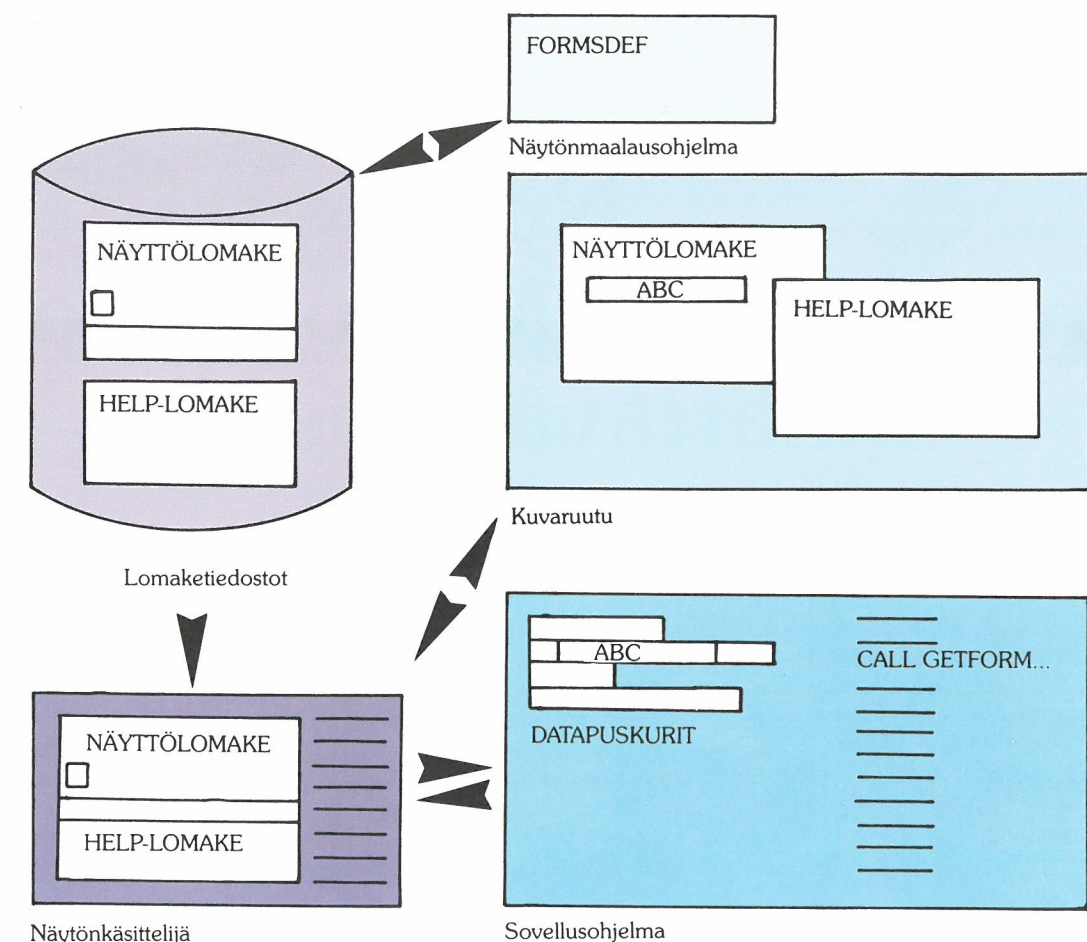
Alimmalla tasolla on joukko valikoiduja näytönkäsittelyn perustoimintoja, jotka on toteutettu mahdollisimman tehokkaasti. Perustoimintoja on vähän, ja koko näytönkäsittely on rakennettu niiden päälle. Näytönkäsittelyn alin taso perustuu INT 10H-keskeytykseen.

Kun alemman tason toiminnot tukevat normaalia vuorovaikutteista näytönkäsittelyä, tukevat ylemmän tason toiminnot näyttölomakkeiden käsittelyä. Ylemmän tason toiminnot kutsuvat tarvittaessa alemman tason toimintoja.

Näytönkäsittelijän toimintoja vastaavat selkeät proseduurikutsut, joilla sovellusohjelmat pyytävät käsittelijältä haluamia toimintoja. Niinpä proseduurikutsut voidaan jakaa vastaavasti eri tasoihin.

Ylemmän tason proseduurikutsuilla voidaan esimerkiksi tulostaa tyhjä lomake näytölle, tulostaa lomakkeen datakenttiin tiedot sovelluksen datapuskurista, lukea näyttölomakkeelta datakenttiä tai tyhjentää datakentät. Alemman tason proseduurikutsuilla voidaan tehdä mm. vastaavia tulostus-, luku- ja tyhjästointitoimintoja yksittäiselle näytön kentälle. Sovellusohjelmat voivat käyttää samassa ohjelmassa eri tasojen proseduurikutsuja.

Sovellusohjelmien käytössä olevia proseduurikutsuja on suhteellisen vähän, mikä helpottaa niiden käyttöä. Lisäksi proseduurikutsuissa tarvitaan ainoastaan muutamia parametreja, koska näyttölomakkeet kuvataan sovellusohjelmien ulkopuolella FORMSDEF-näytönmaalausohjelmalla.



Tiedon syöttö datakenttään ...

Esimerkkinä MPSFORMSin käytöstä sovellusohjelmoijan näkökulmasta tarkastellaan tiedon syöttöä.

Varsinainen tiedon syöttö näyttölomakkeiden datakentiltä tapahtuu proseduurikutsulla GETFIELD tai GETFORM.

Tarkastellaan ensin proseduuria GETFIELD eli tiedon syöttöä tiettyyn näyttölomakkeen datakenttään. Aluksi näytönkäsittelijä siirtää kohdistimen ko. datakentän alkuun, jonka jälkeen kenttään voidaan syöttää tietoa. Syötön aikana syöttötietoa voi muotoilla, koska näytönkäsittelijän sisäinen muokaus (editointi) on automaattisesti käytössä kaikissa datakentissä.

Tiedon syöttöön liittyy automaattinen sisäinen tarkistus- ja virhehälytysjärjestelmä eli ns. implisiittiset tiedon tarkistukset. Käsittelijä tekee datakenttien virhetarkistukset näyttölomakkeen määrittelyvaiheessa asetettujen attribuuttien mukaan.

Jo tietoa syötettäessä näytönkäsittelijä tekee tiedolle tarkistuksia ja ilmoittaa syöttövirheestä äänimerkillä. Näin käy esimerkiksi syötettäessä kirjainta numeeriseksi määritellyyn datakenttään.

Kentän täyttyminen tai lopetusnäppäimen painaminen lopettaa syötön. Lopetusnäppäinten sijainnit ja lopetus-

koodit ovat konfiguroitavissa. Syötön lopetuksen jälkeen näytönkäsittelijä tekee automaattisesti joukon sisäisiä toimintoja.

Aluksi tehdään syötetylle tiedolle implisiittisiä tarkistuksia. Jos käsittelijä havaitsee virheen, se antaa äänimerkin ja tulostaa selväkielisen virheilmoituksen, sekä siirtää kohdistimen takaisin kentän alkuun odottamaan korjaussyöttöä.

Seuraavaksi tehdään mahdolliset sisäiset tiedon tarkistukset. Niiden kuvaukset virheilmoituksineen on talletettu lomaketiedostoon näytönmaalausyhteydessä.

Ja lopuksi tehdään vielä ulkoiset tiedon tarkistukset itse sovellusohjelmassa, mikäli näyttölomakkeen määrittelyssä on datakenttään liitetty tarkistuskoodi osoittamaan ulkoisen tarkistuksen pyyntöä. Tällöin GETFIELD-proseduuri keskeyttää toimintansa ja välittää sovellusohjelmalle lopetuskoodin sijasta negatiivisen tarkistuskoodin arvon. Tästä sovellusohjelman pitäisi havaita, että siltä odotetaan nyt ulkoista tiedon tarkistusta.

Ulkoisissa tarkistuksissa sovellusohjelma voi tutkia esimerkiksi datakenttien välisiä riippuvuuksia. Se voi myös muuttaa näytönkäsittelijän toimintaa tilapäisesti näytönkäsittelyn yleisten ohjaustietojen avulla tai pyytää virhehälytystä. Tarkistusten jälkeen sovellusohjelman pitää kutsua uudelleen GET-

FIELD-proseduuria samoilla parametreilla, jotta proseduurit voisivat jatkaa keskeytynyttä toimintaansa.

Kun syötetty tieto on näin todettu virheettömäksi, näytönkäsittelijä jatkaa edelleen syöttötiedon käsittelyä. Se mahdollisesti muotoilee tietoa ja lopuksi siirtää tiedon sovellusohjelman datapuskuriin.

Lopuksi tapahtuu vielä syötetyn datakentän automaattinen takaisintulostus näytölle. Numeerisen tiedon syötössä esimerkiksi tarpeettomat etumerkit ja -nollat poistetaan.

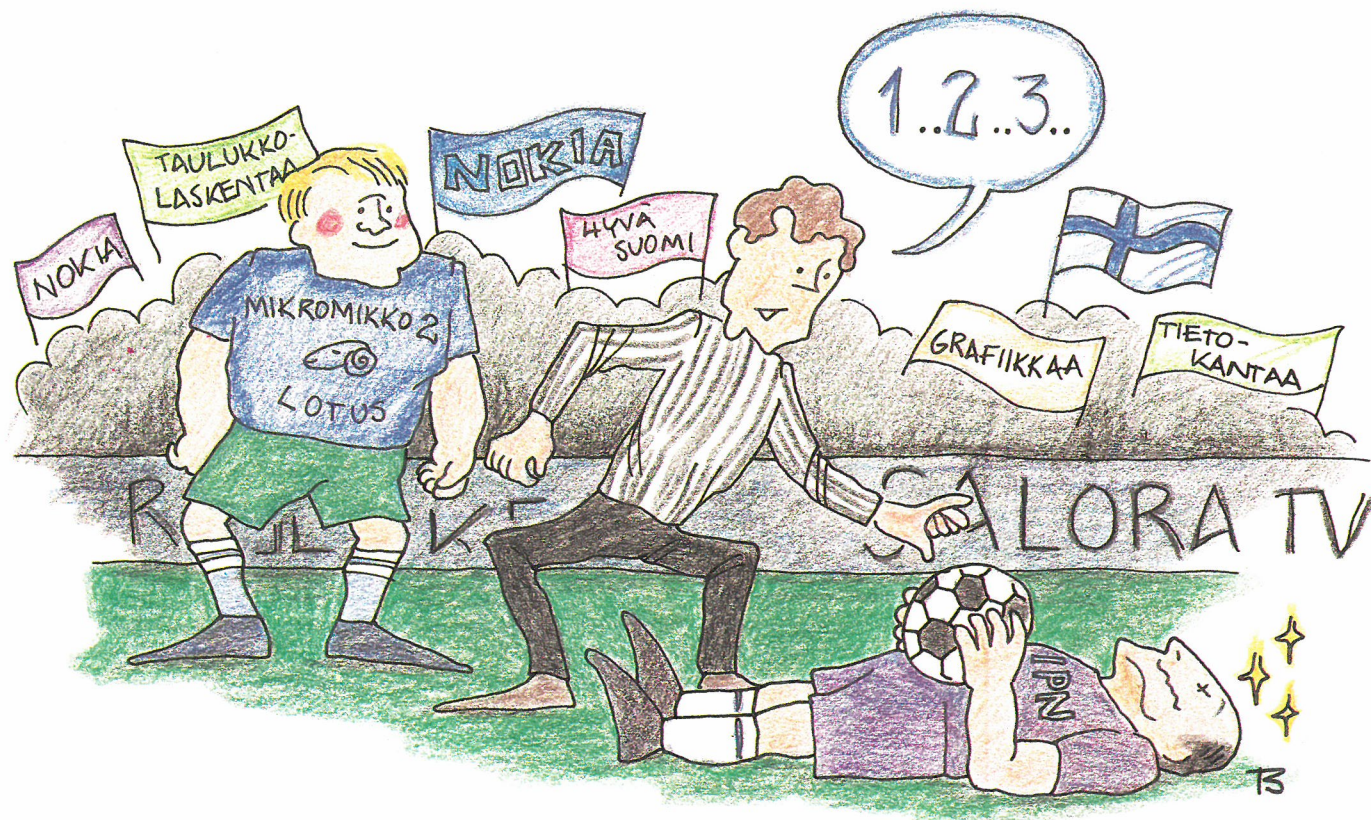
... ja lomakkeelle

Kun proseduurit GETFIELD lukee syötötiedon yksittäisestä datakentästä, kohdistuu tiedon syöttö proseduurilla GETFORM puolestaan lomakkeen kaikkiin datakenttiin. Datakenttien linkkiketjut, jotka asetetaan lomakkeen määrittelyn yhteydessä, ohjaavat kenttien käsittelyjärjestystä. Linkkiketjun päätyminen tai lopetusnäppäimen painaminen lopettaa syötön.

GETFORMissa nuolinäppäimiin on liitetty tiettyjä erikoistoimintoja. Jos esimerkiksi näyttölomakkeita on ketjutettu, tulostaa SHIFT + nuolinäppäin automaattisesti uuden näyttölomakkeen ja aloittaa syötön ko. lomakkeelta.

GETFORM on rakennettu käyttäen proseduuria GETFIELD. □

Nyt Lotus 1-2-3 ja MikroMikkokin ymmärtävät toisiaan



Lotus 1-2-3 on pitkään ollut mikrojen suosituin ohjelmisto maailmalla. Nyt tämä kysytty ohjelmisto on istutettu myös MikroMikko 2:een.

Millainen Lotus 1-2-3 oikein on ja mitkä ovat sen erityispiirteet MikroMikossa?

Lotus 1-2-3:n MikroMikko 2 -versio syntyi Nokian ja Lotus 1-2-3:n maahantuojaan, Tietoväylä Oy:n, yhteistyönä. Lotus 1-2-3:n alkuperäinen valmistaja on amerikkalainen Lotus Development Corporation.

Lotus 1-2-3 tunnetaan ensimmäisenä ns. integroituna ohjelmistona. Ohjelmistossa yhdistyvät taulukkolaskenta, taulukkorakenteeseen perustuva

tietokanta ja grafiikka. Lisäksi ohjelmistoon kuuluu kolme apuohjelmaa, joilla voidaan mm. formatoida levykkeitä, kopioida ja poistaa tiedostoja levykkeeltä, sekä tulostaa kirjoittimelle Lotus 1-2-3:n grafiikalla tehtyjä kuvatiedostoja.

Lotus 1-2-3 -ohjelmisto on englanninkielinen.

Taulukkolaskenta keskeisenä

Lotus 1-2-3:n vahvin puoli on taulukkolaskenta, joka on toimintoiltaan samankaltainen kuin esimerkiksi Multiplan. Lotus 1-2-3:een sisältyvät kaikki taulukkolaskennan perustoiminnot, kuten otsikointi, taulukon lokeroiden väliset laskutoimitukset, kaavojen ja luku- ja tallentaminen lokeroihin, sarakkeiden ja rivien lisäys ja poisto, lokeroiden muotoilu, tietojen kopiointi muista taulukoista jne.

Lotus 1-2-3:ssa taulukko voi olla jopa 2048 rivin ja 256 sarakkeen kokoinen. Taulukon lokeroihin voit kirjoittaa kaavoja, lukuja ja tekstiä. Kaavoissa on käytettävissä tavallisten laskutoimitusten (+, -, *, /) lisäksi monenlaisia matemaattisia funktioita.

Lotus 1-2-3 pystyy yhdistelemään taulukoita toisiinsa. Taulukkoon voidaan kopioida suoraan tietoja toisesta taulukosta. Taulukon lukuihin voit lisätä tai niistä vähentää toisen taulukon vastaavat lukuarvot. Tätä piirrettä ovat monet Multiplanin käyttäjät kaipaillleet. Lotus 1-2-3 kopioi tietoja myös muiden ohjelmien tiedostoista, mm. muilla taulukkolaskentaohjelmilla tehdyistä taulukoista.

Yksi Lotus 1-2-3:n erikoispiirteistä on kalenteriaritmetiikka. Lokeroihin tallennettuja päivämääriä voidaan vertailla päivän, kuukauden ja vuoden tarkkuudella. Päivämääriin voi kohdistaa myös laskutoimituksia.

Näppäistömakrojen käyttö on toinen hyödyllinen erikoispiirre Lotus 1-2-3:n taulukkolaskennassa. Käyttäjä antaa komennot yleensä yksitellen näppäimistöltä. Lotus 1-2-3:ssa peräkkäin suoritettavat komennot voidaan niputtaa näppäimistömakroksi ja tallentaa se johonkin taulukon lokeroon. Kukin makro nimitään omalla kirjaimellaan. Makron sisältämäni komentojonon voi nyt suorittaa vain viittaamalla makron nimeen. Makrojen käytöstä on hyötyä silloin, kun joudutaan kirjoittamaan samat komennot monta kertaa peräkkäin. Makrojen käyttö vähentää työmäärää olennaisesti.

Taulukkopohjainen tietokanta

Lotus 1-2-3 pystyy käsittelemään taulukkoa tai sen osaa tietokantana. Tällöin yksi taulukon rivi muodostaa tietueen ja sarakkeet tietueen kentän. Tietokannan ensimmäisenä tietueena on sarakkeotsikoiden muodostama rivi. Sarakeotsikoiden nimiä käytetään tietueen kenttien niminä. Näitä nimiä tarvitaan useissa tietokantakomennoissa. Niinpä sarakkeiden nimeäminen on pakollista. Ja kahdella sarakkeella ei myöskään saa olla samaa nimeä.

Taulukkolaskennan toiminnoilla voit lisätä uusia kenttiä (sarakkeita) tietueisiin (riviin) sekä uusia tietueita (rivejä). Samaten voidaan muuttaa tietueiden kenttien järjestystä siirtämällä sarakkeita, muuttaa kenttien leveyksiä sekä päivittää kenttien sisältöjä. Tietokanta tai osa siitä on tulostettavissa kirjoittimelle taulukkolaskennan tulostuskomennoilla.

Varsinaisilla Lotus 1-2-3:n tietokantakomennoilla voidaan mm. lajitella tietueita ja poimia tietyin kriteerein tietoja tietokannasta.

Sort-komento lajittelee tietokannan tietueet nousevaan tai laskevaan järjestykseen haluamasi avainkentän mukaan. Lajittelussa voidaan käyttää samanaikaisesti myös toista avainkenttää. Ja lajittelun voi rajoittaa haluttuun tietokannan osaan.

Komennon Query eri alikomennoilla (Find, Extract, Unique ja Delete) voi tietokannalle tehdä useampia erilaisia toimenpiteitä.

Find-komento etsii tietokannasta tietyt ehdot täyttävät tietueet. Ohjelma tuo näytölle ensimmäisen ehdot täyttävän tietueen. Nuolinäppäimellä pääset seuraavaan ehdot täyttävään jne. Voit selata näitä tietueita myös taaksepäin. Jos esimerkiksi tietokantana on puhelinluettelosi, voit henkilönimellä etsiä haluamasi henkilön puhelinnumeron.

Extract-komento kopioi määrittelemäsi kentät niiltä tietueilta, jotka täyttävät asettamasi ehdot. Kopioituvat tiedot tallentuvat määrittelemäsi taulukon osaan. Näin voit poimia tietokan-

nasta uuden suppeamman tietokannan.

Unique-komento on toiminnoiltaan muuten samanlainen kuin Extract päitisi, että määrittelemäsi taulukon osaan ei kopioidu kahta samanlaista tietuetta.

Delete-komento poistaa tietokannasta ne tietueet, jotka täyttävät asetetut ehdot.

Grafiikalla tiedot kuvalliseen muotoon

Taulukkolaskennan ja siihen pohjautuvan tietokannan lisäksi Lotus 1-2-3:een on yhdistetty grafiikka. Lotus 1-2-3:lla voidaan määrittellä kuvia, tulostaa kuvia näyttöruudulle ja kirjoittimille sekä tallentaa kuvia levykkeelle/levylle.

Ennen kuin kuvia voidaan piirtää, on kuva määriteltävä. Ensimmäisenä on määriteltävä kuvan tyyppi. Kuva voi olla viiva-, pylväs-, päällekkäispylväs-, xy- tai piirakkakuvi. Kuvatyyppin jälkeen kerrotaan Lotus 1-2-3:lle, mikä rivin tai sarakkeen osa vastaa x-akselia, mikä ensimmäisenä piirrettävää muutujaa, mikä toisena piirrettävää jne. Piirakkakuvion määrittelyssä tarvitsee antaa vain yksi muuttuja, jonka arvojen mukaan piirakka muodostetaan. Muissa kuvatyypeissä voidaan vertailla yhtäaikaa useampaa muuttujaa.

Kun kuva on määritelty, piirretään se näytölle View-komennolla. Näytölle piirtyvän kuvan lukuarvot ohjelma hakee muistissa olevasta taulukosta. Jos kuva halutaan tulostaa kirjoittimelle tai piirturille, se on ensin tallennettava ns. kuvatiedostoksi levykkeelle/levylle. Kuvatiedostosta kuvan voi myöhemmin tulostaa kirjoittimelle tai piirturille käyttäen PrintGraph-apuohjelmaa.

Myös pelkät kuvan määrittelytiedot voidaan tallentaa tiedostoksi levykkeelle/levylle ja lukea ne myöhemmin sieltä muistiin. Kun piirretään kuvaa tiedostosta luetuilla määrittelyillä, lukuarvot haetaan kuvaan sillä hetkellä muistissa olevasta taulukosta. Täten samoilla määrittelyillä saadaan eri taulukoista vertailukelpoisia kuvia.

Kolme apuohjelmaa

Lotus 1-2-3:n mukana toimitetaan kolme apuohjelmaa: File Manager, Disk Manager ja PrintGraph.

Disk Manager ja File Manager -ohjelmilla voidaan suoraan Lotus 1-2-3:sta tehdä samoja toimenpiteitä kuin käyttöjärjestelmän varusohjelmilla. Itse asiassa Lotus 1-2-3:n käyttäjän ei tarvitse tietää mitään käyttöjärjestelmätason komennoista.

Disk Managerilla käyttäjä mm. formatoi levykkeitä, kopioi levykkeen sisällön toiselle levykkeelle yksi yhteen ja tarkistaa levykkeen käyttökelpoisuuden. File Managerilla voidaan puolestaan tulostaa näytölle levykkeen hakeisto, nimetä tiedostoja uudelleen, sekä kopioida ja tuhota tiedostoja.

PrintGraph-ohjelmalla tulostetaan kirjoittimelle levykkeelle tallennettu kuvatiedosto. Ohjelma kyselee käyttäjältä kaikki tarpeelliset tiedot, kuten kuvatiedoston nimen ja tulostuslaitteen.

Lotus 1-2-3 MikroMikko 2:ssa

Lotus 1-2-3 toimii MikroMikon MS-DOS -käyttöjärjestelmän 2.12 tai sitä uudemmassa versiossa.

Lotus 1-2-3 vaatii keskusmuistia vähintään 192 ktavua, joten ohjelma toimii kaikissa MikroMikko 2 -malleissa. Ohjelmisto pystyy käyttämään keskusmuistia aina 544 ktavuun. Täten 768 ktavun muistilla varustetussa MikroMikossa voidaan käyttää ohjelmiston koko kapasiteetti, mikä käytännössä merkitsee mahdollisuutta rakentaa suurempia taulukoita.

Koska Lotus 1-2-3:ssa ei käytetä pehmonäppäimiä, saadaan MikroMikon näytölle samanaikaisesti 23 taulukoriviä normaalin 20 rivin sijasta. Kolme lisärivää näytöllä parantaa ohjelman käyttömukavuutta olleellisesti.

MikroMikon Lotus 1-2-3 pystyy tulostamaan n. 50 erimerkkiselle kirjoittimelle, mm. Facit ja Mannesmann Tally-kirjoittimille.

Mitä ohjelman mukana

Käyttäjälle toimitetaan Lotus 1-2-3 -ohjelmalevykkeiden mukana suomenkielinen tukipaketti, englanninkielinen käsikirja ja opetusohjelma (tutorial).

Suomenkieliseen tukipakettiin kuuluva käyttöopas tutustuttaa käyttäjän Lotus 1-2-3:n perustoimintoihin nopeasti. Ja pian käyttäjä antaa mieluummin ohjelman opettaa kuin selaa käsikirjan sivuja. Englanninkielisestä käsikirjasta onkin eniten hyötyä vaativalle käyttäjälle, joka haluaa hyödyntää ohjelmiston kaikki hienoudet. □

HMa



System-80 -ohjelmageneraattorilla rakentelet tiedoston kysely- ja päivitysohjelmia sekä raporttiohjelmia DPS-laitteille. Ja se käy helposti, sillä System-80 ei edellytä ohjelmointikokemusta.

System-80 on amerikkalaisen Phoenix Systems'in valmistama ohjelmageneraattori, jonka myyntioikeudet Honeywell on ostanut.

System-80 toimii GCOS 6 ja 8 -käyttöjärjestelmissä ja se on käytettävissä sekä DPS 6 että DPS 8 -laitteistoilla. System-80 -ohjelmistosta on tulossa versio, jonka avulla voidaan rakentaa DM4-tietokantaa käyttäviä ohjelmia.

Generaattorin suunnittelussa oli lähtökohtana ajatus, että useimmat kaupallishallinnolliset sovellukset ovat luonteeltaan samankaltaisia. Ne keräävät tietoa käyttäjiltä ja tiedostoista, luovat ja ylläpitävät tiedostoja, sekä raportoivat tietoa käyttäjille. Olennaisimmat erot ohjelmissa ovat niiden suorittamisessa rutiineissa ja niiden käyttämissä tiedostoissa.

Ohjelmakoodi standardoitua

System-80 generoi ANSI-standardin mukaisia Cobol-74 -ohjelmia. Koska sen generoima ohjelmakoodi on hyvin jäsenneltyä ja standardoitua, on koodin muuttaminen helppoa. Standardoidun koodin ansiosta ohjelmat ovat myös helposti siirrettävissä koneelta toiselle.

Ohjelmageneraattori keskustelee käyttäjänsä kanssa valikoilla. Näin ollen sen käyttö ei edellytä ohjelmointikokemusta. Cobol-terminologian tunteminen on toki hyödyksi.

System-80:llä generoitavat ohjelmat voidaan jakaa tyypeittäin seuraavasti:

- reaaliaikainen tiedon syöttö/päivitysohjelma
- tiedostoa päivittävä eräohjelma
- raportin tulostava eräohjelma
- tiedostoa päivittävä sekä raportin tulostava eräohjelma

Tieto- ja näyttöhakemistot

Ennen ohjelmien generointia on kuvattava ohjelmien käyttämät tiedostot tietohakemistoon sekä määriteltävä tarvittavat näytöt näyttöhakemistoon. Tietohakemistoon talletetuista tiedostokuvauksista System-80 generoi Cobol-ohjelmaan FD- ja SELECT COPY-kappaleet.

Tiedostot kuvataan tietohakemistoon kenttä kerrallaan. Tiedostokuvauksen voi myös kopioida tietohakemistoon apuohjelmalla Cobol-74 -ohjelman FD-kuvauksesta tai erillisestä COPY-kirjastosta. Kullekin tietokentälle tehtävät tarkistukset määritellään tietohakemistoon. Samoin pikaraportteissa oletusarvoina käytettävät tietokenttien otsikot voidaan määritellä tietohakemistoon.

Näyttöhakemistoon talletettu näyttökuvaus sisältää näytön ja sen mahdollisten jatkonäyttöjen nimet sekä näytön avulla päivitettävän tietueen nimen. Myös näyttöalkioiden, kuten ohjetekstien ja datakenttien, sijaintipaikat näytöllä sekä datakenttien suojaukset on määriteltävä näyttökuvauksessa.

System-80:een sisältyy joukko apuvälineitä, joilla voidaan mm. muuttaa ohjelmageneraattorin oletusarvoja, tyhjentää hakemisto tai ottaa hakemistosta varmuuskopio. □

Ada-ohjelmakomponentti-kirjasto

Osa 1: Merkinkäsittelyrutiinit

Ohjelmistotyön tuottavuuteen vaikuttaa merkittävästi valmiiden ohjelmakomponenttien käyttö. Nyt alkavassa artikkelisarjassa tarkastellaan niiden käyttöä Ada-ympäristössä, joka mahdollistaa ohjelmakomponenttikirjaston tehokkaan käytön.

Tässä artikkelisarjassa käsitellään Adan yleiskäyttöisiä rutiineja sekä teoreettisesti että käytäntöön soveltaen. Ada/MPS:n erityisominaisuudet otetaan myös huomioon. Kirjoitussarja käsittelee seuraavat aiheet: merkkijono, lista, jono, pino, joukko, taulukko, matriisi, binääripuu, verkko ja B-puu. Tässä ensimmäisessä osassa esitellään joukko yleisimpiä merkinkäsittelyrutiineja.

Ada ja pakkaukset

Ada-ohjelmointikieli mahdollistaa ohjelman kokoamisen osista eli pakkauksista, kuten niitä Adassa kutsutaan. Pakkaukset sisältävät joukon proseduureja, funktioita ja tyyppien määrittelyjä. Rakentamalla yleiskäyttöisiä pakkauksia ja kokoamalla ne ohjelmakirjastoksi saadaan tehokas ohjelmankehitysympäristö. Tavoitteena on, että suuri osa ohjelmasta muodostuu valmiista komponenteista ja vain pieni osa on itse kirjoitettua koodia. Tämä on vähitellen toteutumassa.

Eräs yhteinen ominaisuus ohjelma-komponenteilla on se, että niissä huomioidaan erilaiset poikkeus- ja virhetilanteet. Näin ohjelmoija voi keskittyä tehtävän kannalta oleellisiin asioihin ja unohtaa tarpeettomat yksityiskohdat. Samalla ohjelmointityön tuottavuus kasvaa. Tämän 'information hiding'-periaatteen mukaan ohjelmoija voi työskennellä korkeammalla abstraktotasolla.

Merkkijonot kiinteämittaisia

Merkkijonot ovat varsin yleisesti käytetty tietorakenne melkein missä tahansa ohjelmassa. Monissa lausekielissä voidaan määritellä merkkijonomuuttujia ja niille on olemassa tarvittavat operaatiot. Adassa merkkijonotyyppi on määritelty taulukkona.

```
type STRING is array
(POSITIVE range <>)
of CHARACTER;
```

Adassa merkkijonomuuttuja määritellään esimerkiksi:

```
JONO: STRING (1..10);
```

Merkkijonoille on määritelty sijoitus-, vertailu- ja yhdistelyoperaatiot. Operaatioiden käyttöä rajoittaa se, että operandien pitää olla samaa tyyppiä eli yhtä pitkiä. Siten esimerkiksi seuraava ohjelmanpätkä on käyttökelvoton, koska ehto ei ole koskaan tosi:

```
A: STRING (1..10);
B: STRING (1..5);
```

```
...
if A = B then ...
```

Adassa merkkijono on kiinteämittainen. Ohjelmoijan kannalta on usein toivottavaa, että merkkijonot ovat loogisesti vaihtelevan mittaisia eli kiinteämittaisen merkkijonon lopputyhjät jätetään huomiotta. Tähän itse Ada-kieli ei anna keinoja. Mutta merkkijonojen käsittelyn helpottamiseksi on rakennettu joukko aliohjelmaa. Aliohjelmat mahdollistavat loogisten merkkijonojen helpon käsittelyn ja vapauttavat ohjelmoijan miettimästä turhaan Adan asettamia rajoituksia ja sääntöjä.

Aliohjelmia avuksi

Merkkijonojen käsittelyyn tarvitaan ainakin siirto- ja vertailuoperaatiot sekä merkkijonon osien etsintä- ja muutospoperaatiot. Lisäksi tarvitaan merkkijonon paloittelu- ja yhdistelyoperaatiot.

Siihen, millä tavalla operaatiot määritellään, on lukematon määrä tapoja. Tässä esitetään niistä eräs. Merkkijonon tyhjäys Adassa käy esimerkiksi seuraavasti:

```
A: STRING (1..10);
```

```
...
for I in A'range loop
  A (I) := ' ';
end loop;
```

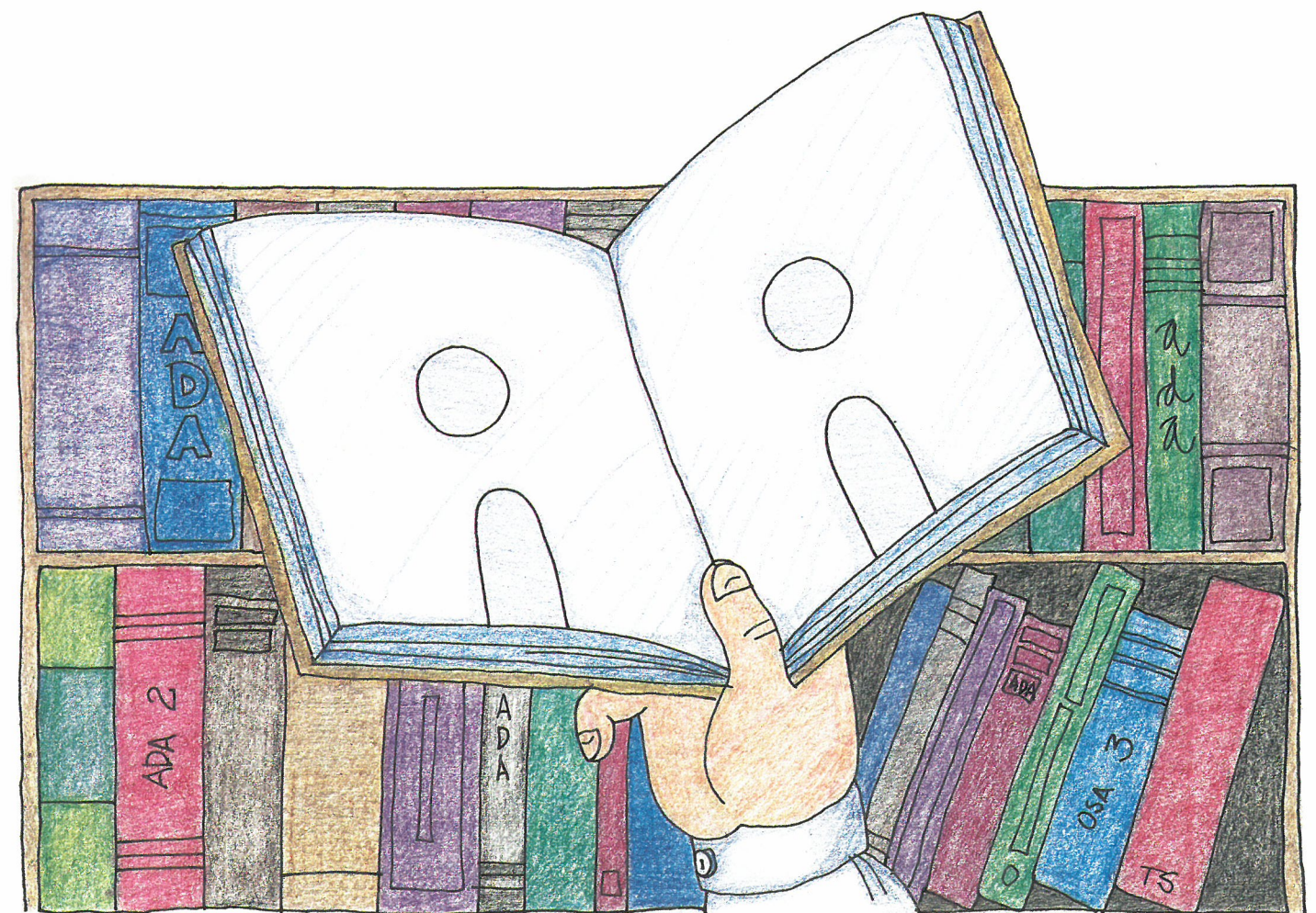
Tässä ohjelmoija joutuu kirjoittamaan 'luupin' ja itse asia, muuttujan tyhjäys, jää sivuseikaksi. Ohjelmasta tulee luetavampi, jos käytetään aliohjelmaa FILL. Se sijoittaa merkkijonon jokaiseksi meriksi halutun merkin. Oletusarvona on tyhjämerkki. Esimerkiksi:

```
FILL (A); -- A täytetään tyhjällä
```

```
...
FILL (A, '0'); -- A nollataan
```

FILL-käskyn jälkeen muuttujan A sisältö on "0000000000".

Aliohjelman käyttö on tulkittavissa ohjelmointikielen laajennukseksi. Tämä onkin varsin tärkeä muistaa käytettäessä tässä esiteltyjä rutiineja. Ohjelman



suunnittelun abstraktiotasoa voidaan nostaa, kun on olemassa 'käskyjä' ha-luttujen toimintojen toteuttamiseksi. Mikä tahansa aliohjelma ei kuitenkaan ole ohjelmointikielen laajennus. Joskus aliohjelman sivuvaikutukset saattavat olla hyvin haitallisia käytettäessä aliohjelmaa muualla kuin alkuperäisessä ympäristössään.

Oleellista käskyvalikoiman laajentamisessa on laajennettujen käskyjen käyttökelpoisuus ja luotettavuus. Usein toistuviin tapauksiin on helpompaa käyttää valmiita rutiineja kuin kirjoittaa omia ohjelmanpätkiä. Ohjelmakomponenttikirjaston perustaminen vaatiikin aivan erityistä panostusta laatuun. Ohjelmakomponentin pitää olla virheetön, tehokas ja helppokäyttöinen.

Merkkijonon 'normalisoimiseksi' hyödyllinen aliohjelma on tyhjämärkien siirtäminen oikeaan laitaan. Esi-merkiksi:

```
A: STRING := " a d a ";
...
BLANKS—TO—RIGHT (A);
-- A sisältää
-- nyt merkkijonon "ada "
```

Merkkijonon osien käsittely

Ada/MPS -kielessä ei ole määritelty merkkijonoviipaleita. Kun halutaan siirtää vain osa merkkijonosta on paikallaan aliohjelma MOVE—SLICE. Seuraava ohjelmanpätkä siirtää ensimmäisen merkkijonon alusta 8 merkkiä toisen merkkijonon alkuun.

```
A: STRING := "A:TIEDOT.TXT";
B: STRING := "????????LST";
...
MOVE—SLICE (A, B, 8); ...
```

MOVE—SLICE -käskyn jälkeen muut-tujan B sisältö on "A:TIEDOT.LST".

Aliohjelmalla ASSIGN siirretään eri-mittaisia merkkijonoja. Jos lähetävä merkkijono on pidempi kuin vastaanot-tava, niin merkkijono katkaistaan jo-non jommasta kummasta päästä. Jos taas lähetävä merkkijono on lyhyem-pi, niin loput täytetään tyhjämärkillä. Seuraavassa esimerkissä tätä ominai-suutta käytetään merkkijonon paloitte-luun.

```
A: STRING := "A:TIEDOT.TXT";
B: STRING (1..8);
C: STRING (1..3);
...
ASSIGN (B, A); ...
ASSIGN (C, A, RIGHT); ...
```

ASSIGN-käskyjen jälkeen muuttujan B sisältö on "A:TIEDOT" ja muuttujan C sisältö "TXT".

Osamerkkijonon etsimiseen merkki-jonosta on aliohjelma SEARCH. Jos halutaan tietää, missä kohtaa on " " merkkijonossa A, niin se käy seuraa-vasti:

```
A: STRING := "A:TIEDOT.TXT";
L: NATURAL;
...
SEARCH (A, ' ', 1, L); ...
```

SEARCH-käskyn jälkeen muuttujan L sisältö on 9. Tätä voidaan hyödyntää siirrettäessä " "n jälkeinen osa:

```
B: STRING (1..4);
...
MOVE—SLICE (B, A, 3, L + 1); ...
```

B:n sisältö on nyt "TXT ". Näin alioh-jelmilla merkkijono voidaan helposti paloitella osajonoiksi.

Merkkijonon osittainen muuttami-nen onnistuu REPLACE aliohjelmalla. Toiminta on samanlaista kuin missä tahansa muokkaimessa. Seuraavassa esimerkissä "TXT" muutetaan "LST":ksi.

```
A: STRING := "A:TIEDOT.TXT";
...
REPLACE (A, "TXT", "LST"); ...
```

REPLACE-käskyn jälkeen A:n sisältö on "A:TIEDOT.LST".

Merkkijonojen vertailu

Erimittaisten merkkijonomuuttujien vertailussa on ongelmana tilanne, jossa pidemmän merkkijonon alku on sama kuin lyhyempi merkkijono. Onko "A:" yhtäsuuri, suurempi vai pienempi kuin "A:TIEDOT"? Tässä on sovittu seuraa-vasta periaatteesta: Jos merkkijonot ei-vät ole yhtä pitkiä, niin lyhyempi mer-kijono oletetaan pidennetyksi tyhjä-merkeillä pitemmän merkkijonon pitui-seksi ja näin saatuja merkkijonoja ver-rataan. Käytäntö on sama kuin Cobol-kielessä. Kahden erimittaisen merkkijo-non vertailuun sopivat funktiot LESS, EQUAL ja GREATER. Esimerkiksi:

```
A: STRING := "A:TIEDOT.TXT";
...
if LESS (A, "B:") then ...
```

Tämä ehto on tosi, kuten seuraavakin:

```
if GREATER (A, "A:") then ...
```

Pienet kirjaimet isoiksi

Joskus on tarpeen muuttaa pienet kir-jaimet isoiksi. Tällöin on hyötyä alioh-jelmasta UPPERCASE. Tätä valaisee seuraava esimerkki:

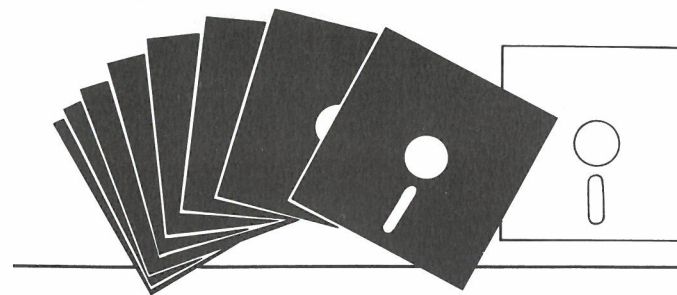
```
A: STRING := "A:tiedot.txt";
...
UPPERCASE (A); ...
```

UPPERCASE-käskyn jälkeen merkki-jonon A sisältö on "A:TIEDOT.TXT".

Numeerisen tiedon muuttaminen merkkijonoksi ja päinvastoin tapahtuu rutiineilla FROM—INT ja TO—INT. Esimerkiksi:

```
A: STRING := "-126 ";
B: INTEGER;
...
B := TO—INT (A); ...
```

Käskyn suorituksen jälkeen kokonaislu-kutyypin muuttujan B looginen si-sältö on miinus satakaksikymmentä-kuusi. □



MikroMikko TopTen

Tämän kertainen MikroMikkojen ohjel-mistolista perustuu kuluvaan vuoden heinä-elokuun myyntiluvuista laskettui-hin suhteellisiin indekseihin.

Multiplan-taulukkolaskenta jatkaa edelleen listan kärjessä. Huomatta-koon, että Multiplanin pahin kilpailija Lotus 1-2-3 nousi heti pelkällä elokuun myynnilään listalle. Se on ehkä seu-raavalla listalla jo kärkisijoilla.

MikroMikko TopTen

1. (1)	Multiplan	56
2. (5)	Multigraph	44
3. (3)	Edix	40
4. (4)	Wordix	36
5. (8)	InfoStar	31
6. (7)	K-mies	30
7. (6)	MS-Basic -tulkki	26
8. (2)	WordStar	21
9. (-)	dBase II	16
10. (-)	Lotus 1-2-3	15

Suluissa sijoitus edellisessä TopTenissä (Softari 2/85).

DECnet-tietoliikenne Symbolics'iin

Symbolics Inc. julkisti DECnet-tieto-liikenneohjelmiston Symbolics 3600 -sarjan tietokoneille Los Angelesissa elokuun lopulla.

DECnet-ohjelmistolla Symbolics 3600 -koneet pystyvät kommunikoi-maan kaikkien DECnet-VAX -liikenne-käytäntöä tukevien tietokoneiden kans-sa. Ohjelmisto mahdollistaa mm. tie-dostojen siirron, elektronisen postin ja toisen koneen nauha-aseman käytön.

Tilaa ohjelmistoluettelo

Tiedot markkinoilla olevista Mikro-Mikko-ohjelmistoista on nyt kerätty yh-teen MikroMikko-ohjelmistoluetteloksi.

Luetteloon on koottu valtuutettujen MikroMikko-jälleenmyyjien ja riippu-mattomien ohjelmistontekijöiden Mik-roMikolle tekemiä ohjelmia. Luettelos-sa on kustakin ohjelmasta myyjän nimi ja osoite, yhdyshenkilön nimi ja puhe-linnumero, laitteistovaatimukset sekä lyhyt kuvaus ohjelmasta, sen sovellusa-lueesta ja laajennettavuudesta. Luette-lon käyttöä helpottaa hakemisto, jossa ohjelmat on ryhmitelty toimialoittain.

Ohjelmistoluetteloita voi tilata joko tämän lehden palvelukortilla tai soitta-malla Marja Kulangolle, puh. 90-567 2039.

Unto ja Mikko

Elokuun lopulla julkistettiin uusi Mikro-Mikko 2:lle kehitetty taloushallinnon ohjelmisto Unto. Tämä pk-yritysten tarpeisiin suunniteltu ohjelmisto käsit-tää varastokirjanpidon, laskutuksen, osto- ja myyntireskontran, kirjanpidon, kustannuslaskennan, palkanlaskennan, henkilöstöhallinnon, tilinpäätössuunnit-telun ja pankkiyhteydet.

Unto on Uicin ja Efactorin yhdessä kehittämä ohjelmisto. Sen suunnitte-lussa on hyödynnetty molempien yri-tysten pitkäaikaisia kokemuksia yli 3000 suomalaisen yrityksen taloushal-linnosta.

Unto-ohjelmat soveltuvat useimmille yrityksille ilman kalliita räätälöintejä. Ja Unto keskustelelee käyttäjänsä kanssa suomeksi. Niinpä käyttäjän ei tarvitse tuntea atk-slangia, sovellusalueen tuntemus riittää.

Uusia DPSiä markkinoille

Nokia on täydentänyt edustamiensa Honeywell DPS-tietokoneiden valikoi-maa julkistamalla DPS 88/42T, DPS 90, DPS 6/22 sekä DPS 6/85 -laitteis-tot.

Tandem-versio DPS 88/42T täy-dentää 88-suurkonesarjaa. DPS 90 eri malleineen edustaa puolestaan uutta kokoluokkaa suurkoneiden sarjan ylä-päässä. Molemmissa laitteissa käyttö-järjestelmänä on GCOS 8. Tämä mer-kitsee yhteensopivuutta, sillä samat työkalut ja ohjelmistot ovat käytettävis-sä kaikissa DPS 8 -laitteistoissa.

DPS 6/22 on 16-bittinen supermik-ro. Peruskokoonpanossaan se sisältää 512 kilotavua keskusmuistia, 68 mega-tavun umpilevyn (Winchester) sekä 650 kilotavun levykeaseman. Ja siihen voidaan liittää viisi työasemaa. DPS 6/22 on laajennettavissa aina kymme-nen työaseman koneeksi. Maksimiko-koonpanossaan DPS6/22 käsittää 1,7 megatavua keskusmuistia, kaksi umpi-levyä ja streamer nauha-aseman.

DPS 6/85 on 32-bittinen supermini, johon on liitettävissä 64 työasemaa. Keskusmuistia laitteeseen saa 2-4 me-gatavua. DPS 6/85 soveltuu järjestel-miin, joissa vaaditaan erittäin suurta tehokkuutta.

Sekä DPS 6/22:n että 6/85:n käyt-töjärjestelmänä on GCOS 6. Kaikki DPS 6 -sarjan koneisiin tehty ohjelmis-to toimii niissä ilman uudelleenohjel-mointia.

Mikromyymälä muuttanut

Nokian mikromyymälä on muuttanut Presidentin talosta Keskuskatu seitse-mään.

Myymälästä saa suoraan hyllyltä mikroja, päätteitä, kirjoittimia, ohjel-mistoja ym. tietokonetarvikkeita. Ja jos kaipaat neuvoja erilaisissa laitteiden käyttöön liittyvissä asioissa, voit turvau-tua myymälän ystävälliseen henkilös-töön.

Nokia
Softari-lehti
PL 780
00003 HELSINKI

Vastauslähetyk-
sen Hki 10 Lupa 2509

Nokia
maksaa
postimaksun

Palvelukortti

Softari 3/85

☐ Haluan esitteitä seuraavista
Nokian ohjelmistoista

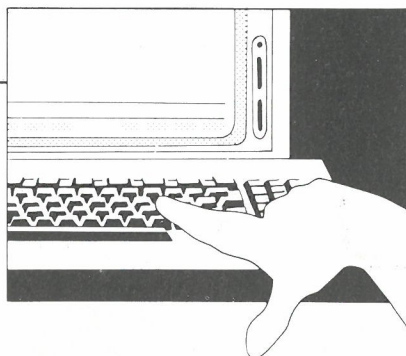
☐ Pyydän, että minuun otetaan
yhteyttä.
Olen kiinnostunut

☐ Korjaan yllä olevat osoitetiedot
Poistan yllä olevat tiedot
☐ Softarin ☐ NETin osoitteistosta
Lisään alla olevat tiedot
☐ Softarin ☐ NETin osoitteistoon

nimi _____
tehtävä _____
yritys _____
osoite _____

JOUKKOJULKAISU

Erittäin



opettavaista!

Loman jälkeen on taas kiva olla töissä, vaikka alku näytti pahalta. Heti maanantaiaamuna pomo marssi työpöytäni viereen innosta puhkuen. Joku kaveri oli saanut hänet uskomaan, että hänen kannattaisi opetella taulukkolaskenta. Nyt hän vaati minua opettajaksi. Mietin kuumeisesti, mitä tehdä. Aikaa voittaakseni pyysin häntä odottamaan kaksi päivää, jotta 'kiireellisimmät' työt saadaan järjestykseen'.

Onneksi pomo on viimeisen päälle pihi mies. Yleensä en ole kiitellyt tätä luonteenpiirrettä, koska jopa välttämättömyimpien laitehankintojen läpivieminen on ollut kiven alla. Mutta nyt olin erittäin tyytyväinen, että hän päätti tinkiä kurssimaksuissa ja kääntyi luottamuksella minun puoleeni. Minulle on kokemuksen myötä kehittynyt selkeä kuva siitä, mitä kelpo pomon tulee tietää mikroista ja mitä ei. Niinpä käytin kaksi työntäyteistä päivää opetuksen valmisteluun.

Päätin opettaa hänelle Lotus 1-2-3:n, koska siinä on enemmän vaihtoehtoja ja hienouksia. Muisti oli päässyt vähän rapisemaan, koska itse käytän Multiplania. Kävin siis läpi kaikki mahdolliset yksityiskohdat, erikoissovellukset ja tein myös muutaman esimerkin, jotka demonstroivat korkeamman matematiikan teoreettisia erityiskysymyksiä. Teoria ja käytäntö ovat tunnetusti laadukkaan opetuksen kulmakivi.

Lopulta olin saanut Lotuksen käskyt juurrutetuksi 'aivojeni RAM -muistiin' niin hyvin, että osasin kaiken ulkoa. Oli aika lähteä noutamaan oppilasta.

Matkalla mikron luo kerroin, kuinka ilahtunut olin hänen edistysellisestä päätöksestään. Nauroin huolettomasti, kun hän ääni epävarmuudesta värähtäen tiedusteli, onko Lotuksen oppiminen vaikeaa. Rauhoitin häntä vakuuttamalla, että normaaliälyinen ihminen oppii sekä mikron käytön että Lotuksen perusteellisesti muutamassa tunnissa. Tarkoitin tietysti, että jossain NITECin viimeisen päälle viritetyillä kursseilla sen voi oppia muutamassa päivässä, mutta miten satuinkaan sekoittamaan päivät tunteihin. Viimeisetkin epäilyksen ripaukset poistuivat kun kerroin, miten videoajan lapset oppivat mikroilun ilman ohjekirjoja ja opastusta, puhtaasti luonnon menetelmällä. En tietenkään tahtonut masentaa häntä kertomalla, että tämä 'luonnon menetelmä' sisältää tuhansia tunteja 'yritys ja erehdys' -tyyppistä harjoittelua.

Täynnä kutkuttavaa odotusta istutin hänet mikron ääreen ja aloitin opetuksen aidolla 'bittinikkari' -menetelmällä, joka perinteisesti käynnistyy termien pikaesittelyllä. Pian alkoi näyttää siltä, että hän koki komplisoiduksi koherentin struktuurin abstrahoinnin annetusta spesifistä formulaatioblokista. Kenties koko kognitiivinen kapasiteetti hupeni termien denotatiivisten ja konnotatiivisten funktioiden identifiointiin. Kun sanoma ei näyttänyt menevän perille, toistin käsite-esittelyni. Tosin tällä kertaa oppilaan hitauden johdosta kärsimättömästi tuhahtellen. Toisto oli kuitenkin ilmeisen tehokas, sillä pomo ei, tavoistaan poiketen, kaivannut lisäselvityksiä.

Nyt voitiin siirtyä käytännön harjoitteluun. Näytin hänelle hetkeäkään tuhlaamatta esimerkkini ja ohjelman toimintaperiaatteet em. termejä runsaasti viljellen. Mahdollisimman kattavan kuvan antamiseksi selitin vielä jokaisen mieleen tulleen yksityiskohdan ja poikkeuksen.

Demonstraation jälkeen katkaisin mikrosta virran ja yllytin häntä yrittämään omin päin.

Kun noin puolen tunnin kuluttua pistäydyin katsomassa opiskelun etenemistä, pomo oli mielestään oppinut tarpeeksi. Hän poistui huoneesta hartaana hiljaisena. Opiskelu näytti ottaneen koville. Eikä mikroilu mitään lastenleikkiä olekaan. Kuvaruudulla komeilivat työn tulokset A> sekä jotain omaperäistä.

Nykyään pomo pitää minua poikkeuksellisenä nerona eikä hänen päähänsä pälkähdä puuttua asioihini. Sain roiman palkankorotuksenkin, tosin sillä ehdolla, että pidän kaikki mikrokoulutusprojektiimme liittyvät asiat ehdottoman salaisina. Samalla koko yrityksen kattava mikrokoulutushanke hautautui vähintään muutamaksi vuodeksi.

Sinänsä sääli, että koko yritys kärsii korvaamattomuuteni varmistamisesta. Koulutuksen jumiutuminen saattaa välillisesti heijastua koko yrityksen tuottavuuteen ja kilpailukykyyn. Mene ja tiedä. Toisaalta minun asiani ei ole ottaa siitä päänsärkyä, sillä enhän minä omista ensimmäistäkään osaketta. Jos vaikeuksia ilmenee, voi hyvissä ajoin siirtyä toiseen firmaan. Mikrotaitoisilla ihmisillä on aina kysyntää.

Softaria