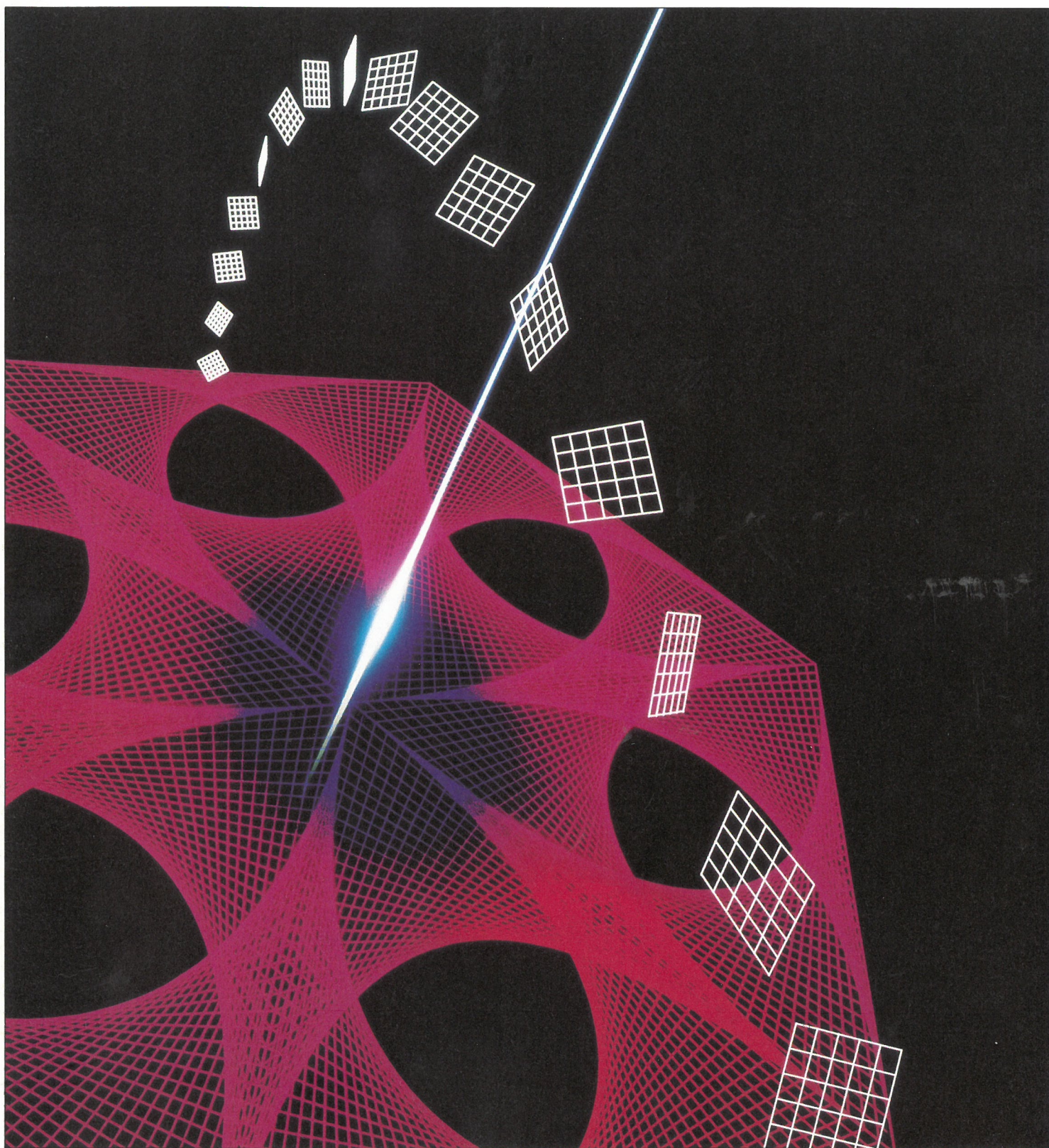
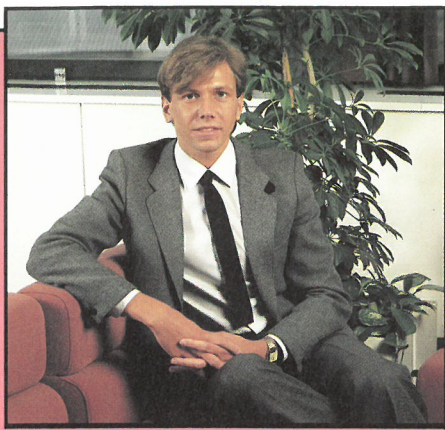


Softari

NOKIAN OHJELMISTOLEHTI 1/86





Tervetuloa luovuus

Luovuus on aina tarpeen. Varsinkin tietotekniikassa.

Luovuudesta on viime aikoina pidetty meteliä. Eikä syyttä. Logiikka ja järkeily eivät enää riitä. Uudet innovaatiot vaativat luovuutta avuksi.

On vaikeaa löytää uusia ideoita yrittämällä katsoa kahta tiukemmin siihen suuntaan, josta on jo etsinyt ratkaisua. Pään kääntäminen on paljon järkevämpää. Uudet ideat syntyvät helpommin asettamalla entiset periaatteet kyseenalaiseksi.

Toiset ovat luonnostaan luovempia kuin toiset. Mutta ei hätää. Luovuuttakin voi oppia. Sitä varten on oikein omia kursseja.

Joissakin yrityksissä käytetään musiikkia luovuuden kirvoittajana ja toisessa rentouttavaa työpaikkaliikuntaa. Kolmannessa pyritään edistämään luovuutta esimer-

kiksi idearyhmillä, joissa eri organisaatiotoimien työntekijät voivat esittää vapaasti ajatuksiaan ja ideoitaan. Kaikissa on kuitenkin sama päämäärä — koko aivokapasiteetin käyttöönotto tietoiseksi työkaluksi.

Myönteinen ilmapiiri on välttämätöntä luovuudelle. Se vapauttaa ajatukset kahleistaan. Päinvastoin kuin ryppyotsaisuus, joka kahlitsee uudet ideat tiukasti alitajunnan vangeiksi.

Parasta mitä voit alkaneelta vuodelta toivoa onkin luovuus. Murra raja-aidat ja tee kaikki toisin kuin ennen. Ole luova!

Luovaa Vuotta 1986!

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

Osoitemuutokset:

Outi Kuusela, puh.(90) 567 2017

Tässä numerossa

Tervetuloa luovuus	2
INFO tehokkaaseen käyttöön asiantuntijan avulla	3
Uusi WordPerfect käy verkkoon	4
Pascal-kurssi — Osa 4	7
CP-8 — Uusi valtti korttipeliin	10
Symbolics 3645 ja 3675 — Kaksi uutta tehokasta Lisp-konetta	12
Nokia tekoälyn kokonaistoimittajaksi	14
Adan listankäsittelyrutiinit	17
Ada eilen ja tänään	20
Uusi myymäläpääte Nokialta	23
Grafiikkaa PC-INFOon	23
Uusia Honeywell-emuja MikroMikko 2:een	23
MikroMikko TopTen	23
21. pölynimuri liikaa	24

INFO tehokkaaseen käyttöön asiantuntijan avulla

HMa

*"INFO-sovelluskehittimellä on helppo tehdä toimivia ohjelmia. Ne eivät kuitenkaan välttämättä toimi kovin tehokkaasti. INFOlla saadaan kyllä tehokkaita järjestelmiä, mutta se edellyttää hyvää INFO:n tunte-
musta tai asiantuntijan apua", sanoo RDB-Systemsin toimitusjohtaja Markku Aarikka.*

RDB-Systems Oy (Relational Database Systems), entiseltä nimeltään Unisoft Oy, on vuonna 1983 perustettu ohjelmistotalo Espoossa. Yritys on erikoistunut ohjelmointiin INFO-sovelluskehittimellä. RDB-Systemsin onkin paras INFO-tietämys Suomessa. Tietämys on saavutettu pitkälti kantapään kautta monivuotisella INFO-kokemuksella. RDB-Systems työllistää tänään viisi INFO-asiantuntijaa.

RDB-Systems järjestää lisäksi yrityskohtaista INFO-koulutusta. Ja viimeisen puolentoista vuoden ajan RDB-Systems on myös pitänyt lähes kaikki Nokian INFO-kurssit.

RDB-Systems on rakentanut INFO-sovelluksia lähinnä suurille yrityksille. RDB-Systemsin asiakkaita ovat mm. OKO, Yhtyneet paperitehtaat, Nokia, Veikkaus ja Raute.

INFO-sovellukset ovat olleet pääosin normaaleja kaupallishallinnollisia sovelluksia, kuten erilaisia myyntitilousjärjestelmiä, valuutta- tai henkilöstöhallintoa. RDB-Systemsin INFOlla toteutettuja paketteja ovat mm. yritysten lainojenhoitojärjestelmä ja kassatoimintojen hoitojärjestelmät.

Rajoitetusti loppukäyttäjille

"INFO:n mainostetaan soveltuvan hyvin myös loppukäyttäjien työkaluksi. Näin minäkin ajattelin ennen. Mutta kokemukset ovat osoittaneet, että INFO:n käyttö loppukäyttäjillä on kuitenkin hyvin rajoittunutta. Ehkä suurin syy on se, että loppukäyttäjillä ei kuitenkaan ole aikaa päivittäisistä rutineista päästä tekemään sovelluksiaan", Aarikka sanoo.

INFOssa voidaan erottaa komento- ja ohjelmointitaso. Aarikan mukaan vain komentotaso soveltuu loppukäyttäjälle. Sen sijaan ohjelmointitason käyttö edellyttää kehittimen syvällisempää tuntemusta.

Suunnittelua ei saa unohtaa

Kehittimillä rakennetaan nopeasti laajojakin järjestelmiä. Kehittimellä tehty ohjelmat eivät kuitenkaan välttämättä toimi tehokkaasti. Koneaikaa kuluu, kun tiedostot kasvavat. Ei ole kiinnitetty huomiota tiedostosuunnitteluun eikä ohjelmien rakentamiseen. Vaikka kehittimillä pystyy tekemään sovelluksia nopeasti, ne vaativat yhtä huolellisen tiedosto- ja ohjelmistosuunnittelun kuin vanhat menetelmät.

"Vaikka suunnitteluun käyttää enemmän aikaa, on ohjelmointi silti huomattavasti nopeampaa kehittimillä kuin perinteisillä tavoilla. Mitä monimutkaisemmasta sovelluksesta on kyse, sitä pienemmäksi ero muodostuu. Yleisesti voidaankin sanoa, että INFOlla pystyy tekemään sovelluksia 3-10 kertaa nopeammin kuin perinteisesti ohjelmoiden. Mutta ylläpidossa kehitin on täysin ylivoimainen. Voidaan puhua jopa 50-kertaisesta nopeudesta.

Ohjelmoinnin edelleen nopeuttamiseksi ja laadun parantamiseksi RDB-

Systems käyttää kehittämiään ohjelmakomponentteja ja ohjelmointistandardeja", Aarikka korostaa.

"Jotkut asiakkaat valittelevat INFO:n hitautta. On tapauksia, joissa listaus on kestänyt useita tunteja. Paria käskyä muuttamalla ajo on saatu jopa 50-kertaa nopeammaksi. INFOlla on siis helppoa tehdä toimivia, mutta hitaita ohjelmia. Sen sijaan asiantuntemuksella saadaan myös INFOlla nopeita järjestelmiä", hän lisää.

Ei kaikkea INFOlla

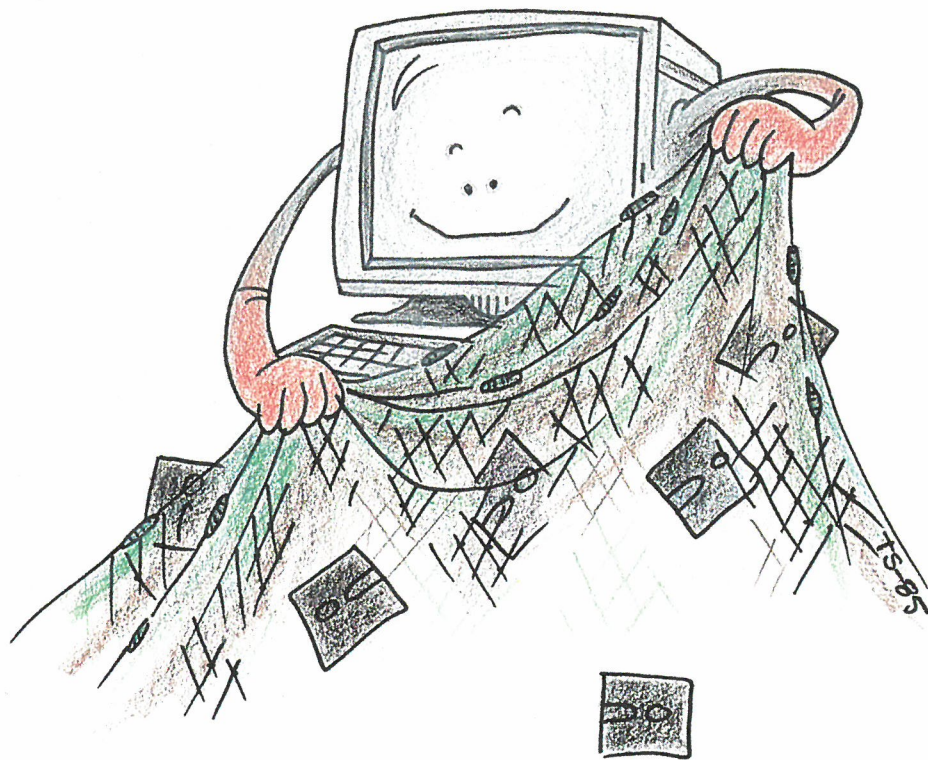
INFOlla pystytään hallitsemaan myös suuria rekistereitä. RDB-Systemsin kokemusten mukaan oikein käytettynä INFO pystyy käsittelemään rekistereitä, joissa on satoja tuhansia tietueita.

Aarikan mukaan kaikkea ei ole kuitenkaan syytä tehdä INFOlla. "Meille ei ole tullut vastaan vielä yhtään sellaista järjestelmää, jota ei olisi voinut tehdä INFOlla. Mutta paljon sellaisia, joita ei olisi ikinä kannattanut tehdä INFOlla. Jos raskasta tapahtumankäsittelyä hoitamaan järjestelmään tarvitaan kymmeniä päätteitä ja vastausaikavaatimukset ovat kovat, ei ole järkevää lähteä soveltamaan INFOa", Aarikka sanoo.

Kannattaako tehdä INFOlla vai ei? Tässäkin kysymyksessä on INFO-asiantuntijan apu usein tarpeen.

INFOlla tehty ohjelmakoodi on helposti siirrettävissä koneesta toiseen. Niinpä laitteistoa laajennettaessa ei ohjelmistoa tarvitse uusia. Esimerkiksi MikroMikko 2:lla tehty järjestelmä voidaan sinällään siirtää DPS 6 -laitteistoihin. Ja päinvastoin. Pieniä eroavaisuuksia kyllä löytyy. Esimerkiksi DPS 6:lla on indeksitiedosto, joka taas puuttuu MikroMikon INFOsta. □

Lisää sovelluksia MikroMikko 2 -lähiverkkoon



Uusi WordPerfect käy verkkoon

MikroMikko 2 -lähiverkkoja asennetaan täyttä häkää. Ja samalla verkon sovellusvalikoima laajenee jatkuvasti. Viimeisin tulokas on uusi WordPerfect-tekstinkäsittelyohjelma.

MikroMikko 2 -lähiverkon eri työasemissa voidaan käyttää samanaikaisesti eri MS-DOS -sovelluksia. Esimerkiksi samalla kun yhdessä työasemassa käytetään Multiplania, voidaan toisessa tehdä vaikkapa tekstinkäsittelyä. Sovellukset saavat käyttää samoja massamuisteja sekä yhteistä kirjoitinta.

Tällä hetkellä verkossa toimivia MS-DOS -yleisohjelmia ovat WordStar ja Edix/Wordix -tekstinkäsittelyohjelmat, Multiplan-tilukkolaskenta, InfoStar, SuperSort, dBASE II, Quickcode sekä K-mies. Ja viimeisimpänä verkkoon on istutettu WordPerfect-tekstinkäsittelyohjelman uusin versio 4.1. Uusi WordPerfect toimii tietysti myös yksittäisessä MikroMikko 2:ssa.

WordPerfectin valmistaja on amerikkalainen Satellite Software International. Ohjelmiston on suomentanut sen maahantuojat Businessman Oy.

Helppo käyttää

WordPerfect on komentopohjainen. Ohjelmiston toiminnot käynnistetään yksinkertaisilla komennoilla ilman valikkojen selausta. Komennot annetaan joko toimintonäppäimillä tai shift- ja ctrl-näppäinten avulla.

MikroMikossa komennot on sijoitettu myös pehmonäppäimiin. Eli voit käyttää ohjelmistoa joko normaalinäppäimistöltä tai pehmonäppäimistä, tai ristiin - ihan miten haluat.

Koska näytöllä ei tarvita valikkoja, WordPerfectin näyttö on selkeä. Näytöllä on vain yksi ohjausrivi, joka ilmoittaa käyttäjälle tekstin numeron sekä kohdistimen paikan sivun, rivin ja sarakkeen tarkkuudella.

WordPerfectissä teksti näkyy kuvaruudulla valmiiksi muotoiltuna eli sellaisena kuin se tulostuu paperille. Ainoa poikkeuksena on oikean reunan tasaus. Oikea reuna ei kuvaruudulla ole vielä tasattua, vaan vasta kirjoittimelle tulostettaessa. Tai sitten ei - ihan miten käyttäjä haluaa.

Sen sijaan alleviivaukset, lihavoinnit, keskitykset ja sisennykset näkyvät näytöllä heti oikeanlaisina. Uudessa versiossa myös monipalstatekstin palstat näkyvät jo kuvaruudulla rinnakkain, kun ne aiemmassa versiossa tulostuivat rinnakkain vasta paperille. Niin ikään sivurajat tulostuvat näytölle.

Oppimista ja käyttöä helpottaa ohjelmiston suomenkielisyys. Ja lisäksi pulmatilanteissa auttaa HELP-toiminto. Painamalla ns. HELP-näppäintä saat näytölle tarvittavat käyttöohjeet.

Monipuolinen toiminnoiltaan

WordPerfect on monipuolinen. Sen ominaisuudet riittävät ammattimaiseenkin tekstinkäsittelytyöhön. Ohjelma on ainakin perustoiminnoiltaan melko helposti opittavissa.

WordPerfectistä löytyvät kaikki tekstin muotoilun ja muokkauksen perustoiminnot. Näitä ovat esimerkiksi ylä- ja alaindeksit, päällelyönti, ylä- ja alätunnisteet, alaviitteet, automaattinen sivunumerointi, 'etsi ja korvaa' -toiminto, 'poista sana/rivi/sivun loppuun' -toiminnot, tekstin keskitys, vasen ja oikea sisennys jne. Ala- ja ylätunniste-rivejä voi olla yhteensä enintään neljä. WordPerfectin päällelyöntiä käyttäen saadaan aikaan esimerkiksi aksentit.

Reunukset, sarkaimet ja riviväli ovat muutettavissa missä tekstinkohdassa tahansa. Tuolloin kyseisestä kohdasta alkaen teksti muotoutuu uusien asetusten mukaisesti. Ohjelmistosta löytyy myös desimaalisarkainus.

Tekstilohkon käsittely on WordPerfectissä monipuolista. Ensin on merkittävä käsiteltäväksi otettava tekstilohko. Tekstilohkon voi sitten siirtää, kopioida, poistaa tai vaikka tulostaa. Myös lohkon alleviivaus, vahvennus ja muuttaminen isoiksi/pieniksi kirjaimiksi käy kätevästi yhdellä komennolla.

Kohdistinta pystytään liikuttelemaan tekstissä sivuittain, riveittäin ja sanoittain. Kohdistin liikkuu näppärästi myös tekstin alkuun ja loppuun, sekä suoraan tietyn sivun alkuun.

Aiemmassa versiossa oli jo mahdollista käsitellä kahta tekstiä vuorottain, mutta nähdä näytöllä yhdellä kertaa vain toinen niistä. Sen sijaan uudessa WordPerfectissä molemmat tekstit saadaan samalle näytölle, sillä näytön voi jakaa kahdeksi ikkunaksi. Ikkunoista on hyötyä lähinnä kopioitaessa tekstilohkoja tekstistä toiseen.

WordPerfect -version 4.1 uusia piirteitä:

- automaattinen tekstin talletus
- viivagrafiikka
- otsikoiden numerointi ja sisällysluettelon generointi haluttuun paikkaan
- ikkunointi (2)
- toimintojen ryhmittelyä parannettu
- kaikkiin toimintoihin yhden käden sormilla
- sanakirjaa laajennettu
- saatavissa verkkoversiona
- uusittu käsikirja

WordPerfectin plussat ja miinukset:

- + teksti tallettuu automaattisesti
- + moniriviset ylä- ja alatunnisteet
- + rivivälien vaihtelut samassa tekstissä
- + alaviitteet automaattisesti
- + viivagrafiikka
- + monipalstatulostus
- + aritmetiikka
- + makrot
- + suomenkielinen
- + taustatulostus
- automaattinen tavutus puuttuu

Suojaus huomioitu

WordPerfectillä voidaan nimetä uudelleen, poistaa sekä tallettaa tekstitiedostoja.

Automaattinen tekstin talletus halutuun aikaväleihin on mahdollista. Näin voidaan välttää tekstin täydellinen katoaminen esimerkiksi laitehäiriön sattuessa.

Tekstit voi tallettaa myös salasanoihin, jolloin ulkopuoliset eivät pääse teksteihin käsiksi. Verkkoversiossa on lisäksi tekstitiedostojen samanaikainen päivitys estetty tiedostolukituksella.

Talletus voi tapahtua joko WordPerfectin omassa tekstiformaatissa tai ASCII-muodossa. Niinpä se soveltuu myös muokkaimeksi ohjelmien kirjoittamiseen. Niin ikään se pystyy käsittelemään muiden ohjelmien, mm. WordStarin, ASCII-tiedostoja.

WordPerfectissä on komennot levykkeen vapaan työtilan sekä tekstitiedoston koon tarkistukseen.

Aritmetiikkaa, makroja, ...

Tekstinkäsittelyohjelmistoista löytyy harvemmin aritmetiikkaa. WordPerfect sen sijaan hallitsee sarakepohjaista laskentaa. Ohjelmistolla voidaan lisätä, vähentää, kertoa ja jakaa numerosarakkeittain. Varsinaisesta taulukkolaskennasta ei voida kuitenkaan puhua. Sarakkeissa voi olla myös tekstiä. Niinpä WordPerfectillä on mahdollista rakentaa esimerkiksi taseita ja tuloslaskelmia.

Yksi WordPerfectin hienouksista on monipalstarakenne. Se pystyy muotoilemaan tekstin jopa viiteen palstaan.

Kun joudutaan toistamaan tiettyjä komentoja usein, on järkevää hyödyntää WordPerfectin makromahdollisuutta. Käyttäjä voi tallettaa levykkeelle peräkkäisiä näppäinpainalluksia (merkit, komennot tai niiden yhdistelmät) ns. makrokseksi. Jos esimerkiksi käytetään usein samanlaista ylätunnistetta, määritellään ylätunniste makrokseksi. Ja aina kun halutaan kyseinen ylätunniste, viitataan vain makroon. Makroja voidaan myös ketjuttaa.

WordPerfect osaa viivagrafiikan. Eli tekstiin voidaan piirtää nuolinäppäimillä vaak- ja pystyviivoja. Niinpä esimerkiksi organisaatio- ja lohkokaaviot saadaan tekstiin jo kirjoitusvaiheessa. Tosin viivagrafiikan tulostus paperille edellyttää, että käytettävä kirjoitin pystyy tulostamaan tarvittavat viiva-alkiot.

Automaattinen tavutus puuttuu

Suurin ongelma WordPerfectin käytössä on tavutus. Se tuntee vain puoliautomaattisen tavutuksen.

WordPerfect ehdottaa rivin lopussa tavutuskohdan, mutta aina suoraan oikean reunuksen kohdalle ilman mitään tavutussääntöjä. Jos sen ehdottama tavutuskohdan on väärä, siirretään kohdistin haluttuun tavutuskohdan ja painetaan ESC-näppäintä.

Tekstiä voi kirjoittaa myös tavuttamatta, jolloin se rivittyy automaattisesti. Eli kun teksti menee oikean reunuksen yli, WordPerfect siirtää heti viimeisen sanan seuraavan rivin alkuun. Tekstin voi haluttaessa tavuttaa jälkikäteen.

Ohjelmassa on mukana englanninkielisen tekstin oikeinkirjoituksen tarkistus. Sen mahdollistaa WordPerfectin englanninkielinen sanakirja. Kirja sisältää valmiina 80 000 sanaa ja käyttäjä voi lisätä sinne omia sanojaan.

Tulostus taustalla

Tekstit tulostuvat WordPerfectissä taustatulostuksena. Näin käyttäjä voi tulostuksen aikana käsitellä toista tekstiä.

Tulostus on toteutettu jonotusperiaatteella eli tulostettavaksi voidaan laittaa useampia tekstejä peräkkäin. Ohjelma tulostaa tekstit vuorollaan jonon mukaisessa järjestyksessä.

WordPerfect pystyy tulostamaan lähes kaikille kirjoittimille. MikroMikko 2:n WordPerfectissä on valmiina kirjoittimääritys yli 200 kirjoittimelle, mm. eri laserkirjoittimille. Ja jos ei ole valmiina, voidaan kirjoitin määritellä helposti ohjelmiston mukana toimitettavalla PRINTER -apuohjelmalla.

WordPerfect tulostaa tarvittaessa myös joukkokirjeitä ja osoitetarroja osoiterekisterien mukaan. □

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

Pascal-kurssi

Osa 4:

Vastaukset kurssin 3. osan harjoitustehtäviin

Tehtävä 1.

a) Korttipakan kortti:

```
TYPE
  kortti = RECORD
    maa: (risti,ruutu,pata,hertta);
    arvo: 1..13;
  END;
```

b) Tiedot myytävästä huoneistosta:

```
TYPE
  huoneistotiedot = RECORD
    osoite: PACKED ARRAY [1..30] OF Char;
    rakvuosi: Integer;
    omistaja: PACKED ARRAY [1..20] OF Char;
    hinta: Integer;
  END;
```

c) Ruotsin kielen vokaalien joukko ja esimerkki joukon käytöstä:

```
TYPE
  kirjainjoukko = SET OF 'a'..'ä'
  {ASCII-merkistössä 'ä' on viimeinen kirjain}
VAR
  vokaalit: kirjainjoukko;
  c: Char;
```

```
...
vokaalit := ['a','e','i','o','u','y','ä','ö']
{kätevä tapa testata onko c vokaali:}
IF c IN vokaalit THEN
  ...
```

Tehtävä 2.

Kokonaislukulistan rakentaminen ja tulostaminen syöttöjärjestyksessä:

```
PROGRAM lista(Input, Output);
TYPE
  osoitin = ^alkio;
  alkio = RECORD
    arvo: Integer;
    seuraava: osoitin;
  END;
VAR
  alku,x,viim: osoitin;
  i: Integer;
BEGIN
  New(alku);
  alku^.seuraava := Nil;
  viim := alku;
  Read(i);
  WHILE i <> 0 DO
    BEGIN
      New(x);
      x^.arvo := i;
      x^.seuraava := Nil;
      viim^.seuraava := x;
      viim := x;
      Read(i);
    END;
  x := alku^.seuraava;
  WHILE x <> Nil DO
    BEGIN
      Writeln(x^.arvo);
      x := x^.seuraava;
    END;
  END.
```

8. Syöttö ja tulostus

Ohjelmissa tarvitaan syöttöä tai tulostusta, yleensä kumpaakin. Olemme jo esimerkeissämme käyttäneet syöttö- ja tulostusproseduureja Read ja Writeln.

Syöttö ja tulostus liittyvät kiinteästi tiedostojen käsittelyyn. Standardi-Pascalissa käsiteltävät tiedostot ovat peräkkäistiedostoja. Kun tiedosto on luotu tai avattu, sen alkioihin päästään käsiksi vain peräkkäisjärjestyksessä. Tiedostotyyppin määrittely esitettiin kohdassa 6.5.

Tässä kurssissa emme käsittele harvoin käytettyjä tiedosto-operaatioita, kuten esimerkiksi Get ja Put.

8.1 Tiedoston luonti

Tiedosto luodaan standardiproseduurilla Rewrite, joka myös tuhoaa tiedoston mahdollisen vanhan sisällön.

```
TYPE
  alkio = ...; {mielivaltainen tyyppi}
VAR
  x: alkio;
  f: FILE OF alkio;
  ...
Rewrite(f);
```

Uusia alkioita liitetään edellisten perään standardiproseduurilla Write:

```
Write(f, x);
```

Seuraavassa kirjoitetaan tyyppiä FILE OF Integer olevaan tiedostoon 25000 kokonaislukualkiota numerojärjestyksessä:

```
Rewrite(f);
FOR i:= 1 TO 25000 DO
  Write(f,i);
```

8.2 Tiedostosta lukeminen

Olemassa olevan tiedoston sisällön käyttäminen Pascal-ohjelmassa edellyttää, että tiedosto ensin avataan standardiproseduurilla Reset:

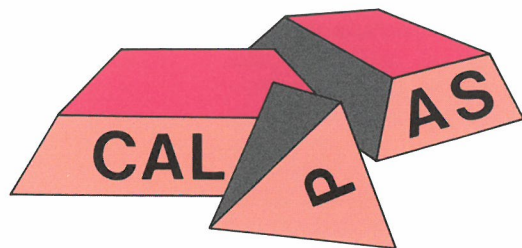
```
Reset(f);
```

Tämän jälkeen peräkkäiset alkiot voidaan lukea proseduurilla Read:

```
Read(f, x);
```

Tiedoston loppumista voidaan tarkailla standardifunktiolla Eof(f). Se saa arvon True, jos ollaan tiedoston lopussa. Jos tiedostosta yritetään lukea, vaikka Eof(f) saa arvon True, seuraa useimmissa Pascal-järjestelmissä ajonaikainen virhe. Funktiota Eof on siis syytä käyttää aina ennen lukukäskyä, ellei tiedetä kuinka monta alkioita tiedosto sisältää.

```
Reset(f);
WHILE NOT Eof(f) DO
  BEGIN
    Read(f, x);
    ... {x:n käsittely}
  END;
```



8.3 Tekstitiedostot

Pascal-kielessä on tekstitiedostoilla erityisasema.

Tekstitiedosto on muodollisesti tyyppiä FILE OF Char oleva tiedosto, joka voidaan jakaa riveihin. Rivi koostuu merkeistä ja rivinlopettimesta. Tekstitiedostoille on varattu oma tyyppintunnus Text.

```
VAR
  txtied: Text;
```

Tekstitiedostojen erikoispiirteisiin kuuluu se, että niihin voidaan tallettaa myös lukuja, tai oikeammin merkeistä koostuvia lukujen esityksiä.

Tekstitiedostoon tulostetaan prosedureilla Write ja Writeln, joista jälkimmäinen kirjoittaa rivinlopettimen tulostettujen parametrien perään.

```
Ohjelmanpätkä
```

```
i:= 3;
Write(txtied,2);
Writeln(txtied,i);
Writeln(txtied,i);
```

```
tulostaa
```

```
2      3
3
```

Standardiproseduurilla Page(txtied) saadaan tiedostoon aikaan sivunvaihto.

Tulostettaessa tekstitiedostoon, voidaan Write-lauseessa määritellä tulostuskentän leveys merkitsemällä parametrin jälkeen kaksoispiste ja kokonaisluku. Kentän leveys määrää montako merkkiä tulostuu kutakin parametria kohden. Jos tulostettava arvo on lyhyempi kuin kentän leveys, kenttä täytetään välilyönneillä joko vasemmalta (luvut) tai oikealta (merkkijonot).

```
Ohjelmanpätkä
```

```
i:= 45
str:= ' KANISTERI'
Write(txtied, i:4, str:5, 'A');
```

```
tulostaa
```

```
45 KANIA
```

Kentän leveyden oletusarvo vaihtelee eri Pascal-järjestelmissä. Samoin ohjelman toiminta silloin, kun tulostettava luku ei mahdu määritettyyn kenttään.

Reaaliluvut tulostetaan oletusarvoisesti eksponenttiesityksenä (ks. kohta 3.2). Ne saadaan myös kiinteän pilkun muotoon, jos kentän leveyden jälkeen merkitään tulostettavien desimaalien lukumäärä kaksoispisteellä erotettuna.

```
Ohjelmanpätkä
```

```
r:= 234.567
Writeln(r);
Writeln(r:8);
Writeln(r:6:1);
```

```
tulostaa
```

```
2.345670000E+02
2.3E+02
234.6
```

Olkoon i tyyppiä Integer. Tällöin lause

```
Read(txtied, i)
```

lukee merkkejä (hyläten mahdolliset välilyönnit ja rivinlopettimet) ja tulkitsee ne kokonaisluvuksi, joka sijoitetaan i:n arvoksi. Sen sijaan Readln-proseduuri lukee ensin tiedostosta parametrien arvot ja hylkää sitten rivin loppuosan.

Jos esimerkiksi tekstitiedostossa txtied on seuraava sisältö:

```
4
7      4.2
3.5
```

```
niin ohjelmanpätkä
```

```
VAR
  a,b: Integer;
  r: real;
```

```
Reset(txtied);
Read(txtied, a);
Readln(txtied, b);
Read(txtied, r)
```

antaa muuttujille seuraavat arvot:

```
a = 4
b = 7
{arvo 4.2 hylätään}
r = 3.5
```

Vain harvoissa tapauksissa tekstitiedostosta luetaan yksittäisiä merkkejä. Jos niin kuitenkin tehdään, on standardifunktio Eoln(ttied) usein tarpeen. Se saa arvon True jos rivillä ei enää ole luettavia merkkejä. Rivinlopetinta ei muulla tavoin voi havaita, sillä se näkyy Read-lauseelle välilyöntinä.

8.4 Input ja Output

Olemme aikaisemmin käyttäneet Read- ja Write-lauseita, joissa ensimmäinen parametri on ollut muu kuin tiedostomuuttuja. Tällöin tiedostooperaatiot kohdistuvat standarditiedostoihin Input (luku) ja Output (kirjoitus). Ne ovat tekstitiedostoja, jotka esitellään ohjelman otsikossa, mutta niitä ei tarvitse erikseen avata.

Tavallisesti käyttäjän pääte toimii sekä Input- että Output-tiedostona.

8.5 Ulkoiset ja sisäiset tiedostot

Ohjelman otsikossa eli PROGRAM-lauseessa voidaan esitellä myös muita tiedostoja kuin Input ja Output.

```
PROGRAM (Input, Output, ifile);
VAR
  ifile: FILE OF Integer;
```

Tällaisia tiedostoja kutsutaan ulkoisiksi tiedostoiksi. Muut ohjelmassa määritelty tiedostot ovat sisäisiä tiedostoja.

Tapa, jolla ohjelmalle ilmoitetaan mitkä fyysiset tiedostot (levyllä oleva nimetty tiedosto, kirjoitin jne.) liittyvät mihinkin sisäiseen tai ulkoiseen tiedostoon, vaihtelee Pascal-järjestelmästä toiseen. Standarditapaa ei valitettavasti ole. Levytiedostojen käsittelyssä voidaan tiedoston nimi esittää esimerkiksi Pascal-standardiin kuulumattoman Assign-proseduurin kutsulla.

```
Assign(ifile, 'LUVUT.DAT');
Reset(ifile);
Read(ifile, i);
...
```

9. Yhteenveto

Pascal on korkeantason ohjelmointikieli, jonka syntaksi on selkeä ja helppo oppia. Ei ole mikään ihme, että se on noussut tärkeimmäksi opetuskieleksi korkeakouluissa ja muissa koulutuslaitoksissa maassamme. Kun Pascal vielä kymmenen vuotta sitten oli 'akateemisten ohjelmointien leikkikalu', olemme tänään siinä tilanteessa, että suurin osa maamme nuoresta atk-väestöstä on saanut sillä peruskoulutuksensa.

Kuten jo kurssin alussa todettiin, Pascal lähes pakottaa ohjelmoijan kirjoittamaan selkeitä ohjelmia. On havaittu, että ohjelman ylläpito vie suurimman osan siihen käytettävistä resursseista. Mitä selkeämpiä ja helpompia ohjelmia tehdään, sitä helpompaa niitä on ylläpitää. Varsinkin muiden kuin alkuperäisen tekijän. Tällaisiin ohjelman laatua parantaviin seikkoihin on alettu kiinnittää yhä enemmän huomiota.

Pascal on yleiskäyttöinen ohjelmointikieli. Nykyisin on sekä suurkoneisiin että mikroiin saatavana kehittyneempiä työkaluja tietojärjestelmien rakentamiseen. Näissä ns. neljännen sukupolven kielissä yleiskäyttöisyys on vaihdettu mahdollisuuteen luoda nopeasti räätälöity tietojärjestelmä.

9.1 Käytännön Pascal-järjestelmistä

Nicklaus Wirthin määrittelemä Pascal ei ole aivan käyttökelpoinen vaativaan ohjelmatuotantoon. Sen puutteita on pyritty poistamaan kielen laajennuksilla. Tavallisimmat laajennusten kohteet ovat merkkijonojen ja tiedostojen käsittely sekä liitännät käyttöjärjestelmään.

Pascal-järjestelmien tekijät ovat itse toteuttaneet tarpeellisiksi näkemänsä laajennukset. Samalla he ovat kuitenkin luopuneet yhteisen standardin antamista eduista. Yhdellä Pascal-versiol-

la kirjoitettua ohjelmaa ei yleensä voi muuttamatta siirtää käännettäväksi toisella Pascal-kääntäjällä.

USCD-Pascalilla on eräänlainen standardin asema (UCSD = University of California San Diego). Siinä ohjelmaa käännetään ensin välikielelle, ns. pseudo- eli p-koodiksi, jota kone sitten tulkkaa käsky käskyltä. P-koodi voidaan myös siirtää toiseen koneeseen. P-koodin tulkkajia on saatavana useimmille tietokoneille.

Mikrotietokoneille tarkoitettuista Pascal-versioista mainittakoon tässä kaksi. Turbo-Pascal ja MS-Pascal.

Borlandin Turbo-Pascal sisältää kääntäjän lisäksi ohjelmatekstin muokausohjelman ja muita apuvälineitä ohjelmien nopeaan kehittämiseen. MS-Pascal taas kuuluu Microsoftin kieliperheeseen, jossa eri kielillä kirjoitettuja moduleita voidaan yhdistää. Se soveltuu suurempiinkin ohjelmistoprojekteihin.

9.2 Pascal-kielen tulevaisuus

Ohjelmointikielen valinta on tärkeä strateginen päätös, jolla saattaa olla vaikutuksia kymmeniksi vuosiksi eteenpäin. Ohjelmointitottumuksia, varsinkin huonoja, on vaikea muuttaa.

Pascalia huomattavasti vanhemmat ohjelmointikielien FORTRAN ja COBOL ovat vieläkin laajalti käytössä, koska niillä tehtyihin ohjelmakirjastoihin on investoitu suuria rahasummia.

Pascal-kielen oppiminen on kuitenkin henkinen investointi, jonka tuotto näkyy ohjelmien laadun parantumisena, vaikka kieltä myöhemmin vaihdettaisiin.

Pascal ei tietenkään ole ohjelmointikielissä viimeinen sana. Jatkuvasti kehitetään uusia kieliä, jotka pystyvät paremmin hyödyntämään uusien tehokkaampien tietokoneiden tarjontaa mahdollisuuksia. Pascalin jälkeen onkin syntynyt useita varteenotettavia ohjelmointikieliä. Yksi niistä on Ada, jonka syntaksi muistuttaa hyvin paljon Pascalia.

9.3 Kirjallisuutta

Korpela, Larmela, Planman: Pascal-ohjelmointikieli, OtaData. Katava Pascal-hakuteos, runsaasti esimerkkejä.

Rodney Zaks: Introduction to Pascal, SYBEX. Tunnetun kirjoittajan selkeä oppikirja.

Cooper, Clancy: Oh Pascal!, NORTON. Paljon mielenkiintoisia esimerkkejä eri aloilta.

Proessorilla varustettu muovikortti CP8 on Korttipelin uusin valtti



Muovikortit ovat tämän päivän maksuvälineitä. Ja kortteja riittää.

Ranskalainen huipputeknologia on lyönyt pöytään uuden valttikortin, mikroproessorilla varustetun CP8-muovikortin. Kortti soveltuu ostosten maksuvälineeksi, käyttäjätunnistukseen ja moneen muuhun.

CP8-muovikortti on ranskalaisen Bull'in kehittämä. Kortti on ollut koekäytössä Ranskassa jo muutaman vuoden, mutta nyt ollaan siirtymässä sen laajamittaiseen käyttöön.

CP8-kortin leviäminen yleiseen käyttöön Ranskassa on jo varmaa tehtyjen päätösten perusteella. "Ranskassa on päätetty pankkien tukemana levittää vuoden -86 loppuun mennessä yleisölle 2,5 miljoonaa korttia. Ja vuoden -88 tavoite on 12 miljoonaa korttia. Kauppiaille puolestaan toimitetaan kymmeniä tuhansia korttien lukulaitteita", kertoo Heikki Kalliola Nokia Informaatiojärjestelmistä.

Viime aikojen yleinen kiinnostuksen herääminen ja eräiden elektroniikan suurvalmistajien sitoutuminen CP8-standardiin lupaavat kortille valoisa tulevaisuutta myös muualla.

CP8 soveltuu moniin arkipäivän sovelluksiin. Tavallisia sovelluskohteita lienevät maksuliikennejärjestelmät, turvallisuus- ja kauttakulkujärjestelmät sekä erityisesti tietopankkien käyttölu-pajärjestelmät. Kortti soveltuu myös mukanakannettavaksi tiedostoksi.

Tietokone lompakkoon

Tavallisen luottokortin muovihihnon on CP8-kortissa upotettu mikroproessori muisteineen. Proessorina on 8-bittinen Motorola 6805 ja muistia kortissa on 1600 merkkiä. CP8-kortti on siis tietokone lompakkokoossa.

Kortti saadaan kommunikoimaan ulkomailman kanssa CP8-liitäntöin varustettujen lukulaitteiden avulla. Kortilla on sähkökontaktit, joiden avulla se lukulaitteeseen työnnettynä saa virran ja kommunikoi ulkomailman kanssa.

CP8-kortissa on haluttaessa myös ISO:n standardeihin perustuva magneettijuova, kuten tavallisissa luottokorteissa.

CP8-kortti on tällä hetkellä hinnaltaan n. 50 mk yksinkappalein. Massalevitys kuitenkin pudottaa hinnan kilpailukykyiseksi kuvallisen pankkikortin kanssa.

Turvallinen käyttää

Kortin muistiin on sen myöntäjä kirjoittanut sovelluskohtaista tietoa. Haluttu osa tiedoista voidaan määritellä sellaiseksi, ettei sitä pysty lukemaan kortilta. Kortin prosessori pääsee tietoihin käsisiksi, muttei luovuta niitä ulos. Niinpä kortin käyttäjätunnuksen tarkistuksen (ns. PIN-tarkistuksen) tekee itse kortti, eikä vaaraa tarkistuksen pohjana olevien avaintietojen vuotamisesta ole. Tähän perustuu kortin ehdoton turvallisuus esimerkiksi maksuvälineenä.

Löydettyä tai varastettua korttia ei voi käyttää hyväksi, ellei tiedä käyttäjätunnusta. Tätä ei kortin avulla saa selville eikä lukulaitteen haltijakaan pysty kokeilemalla etsimään oikeaa tunnusta. Kolmen väärän yrityksen jälkeen kortti vetäytyy kuoreensa kuin simpukka eikä reagoi enää mihinkään. Tilanne vastaa nykyisten pankkiautomaattien kolmen virheellisen käyttäjätunnuksen syöttämistä seuraavaa kortin 'nielaisua'.

Samoin kuin takavarikoitu kortti voidaan hakea pankin konttorista, on CP8-kortti käytävä pankissa erikoislaittein avaamassa lukkiutuneesta tilasta.

Elektronisena kukkarona

Kortti soveltuu erinomaisesti elektroni- sen rahan säilytykseen. Asiakas voi pankissa latauttaa korttinsa sen muistiin kirjoitettavalla 'elektronisella rahasmallalla'. Aivan kuin täyttäisi kukkaron kolikoilla.

Kortinlukijalla varustettu laite, esimerkiksi yleisöpuhelin, kuluttaa vähin erin tätä 'rahalatausta' vastaten kolikkojen pudottelua automaattiin. Ostovoiman ehdyttyä käydään kortti täyttämässä pankin konttorissa.

Ostosten maksuvälineenä

Kortti on turvallinen maksuväline kaupassa asioitaessa. Kortin ja henkilökoh- taisen käyttäjätunnuksen avulla esittä- dytään kortin lailliseksi haltijaksi ja val- tuutetaan kauppias välittämään ostos- tapahtuma pankkiin. Pankki puolestaan veloittaa asiakkaan tiliä ja hyvittää kauppiaan tiliä.

"Mikäli kauppialla on tarkoitukseen suunnitellut erikoislaitteet, kaikki ta- pahtuu automaattisesti ilman paperito- sitteiden täyttöä ja lähetystä pankkiin", lisää Kalliola.

Pienkauppias tulee sen sijaan toi- meen pienellä käyttäjätunnuksen tar- kistuslaitteella. Kun laitteella on varmis- tuttu kortin olevan oikeissa käsissä, jatketaan kuten tällä hetkellä pankki- kortilla maksettaessa.

Henkilökorttina

Korttia voidaan käyttää myös henkilö- korttina, jonka muistiin on talletettu henkilökohtaisia tietoja. Koska tietoja ei voi lukea tietämättä käyttäjätunnus- ta, henkilökortin tietojen käyttö on täy- sin kortin haltijan valvonnassa.

Kalliolan mukaan eräessä Pariisin yliopistossa on opiskelijoiden opintokir- ja korvattu CP8-kortilla. Vastaavanlai- nen esimerkki on terveydentilaa ja hoi- tohistoriaa koskevien tietojen talletus kortille. Tällöin korttia voi käyttää avuksi diagnoosin teossa. Lukumah- dollisuus on vain salaisen käyttäjätun- nuksen tietäjillä, joten intimitettisuoja on turvattu.

Teleyhteyksien turvaajana

Tietopankkien väärinkäytökset este- tään tehokkaasti soveltamalla CP8- korttia yhteydenottajan tunnistami- seen.

"Korttien luovuttaja voi muodostaa korteista 'perheen'; pitää itse äitikortin ja jakaa laillisille päätekäyttäjille tytär- korteja. Äiti tuntee tyttärensä ja hy- väksyy vain näiden lähentymisyrityk- set", Kalliola täsmentää.

Koko tunnistusoperaatio tapahtuu automaattisesti. Päätteet ja keskuskone on varustettu lukulaittein, joiden kautta kortit vuoropuhelulla tunnistavat toi- sensa. Vaaraa kadotetun kortin väärin- käytöstä ei ole, sillä yhteydenoton al- kuun liittyy käyttäjätunnuksen syöttö ja tarkistus. Lisäksi teleyhteydellä siirrettä- vät tiedot voidaan salata mahdollisilta 'kuunteluyrityksiltä'. Lähettäjäkortti se- koittaa tiedon ja vastaanottajakortti purkaa takaisin selväkieliseksi.

"Erilaisten teletietopalvelujen yleisty- minen vaatii Suomessakin kiinnittä- mään huomiota yhteyksien turvallisu- den varmistukseen. Kriittisten tietokan- tojen käsittely voisi edellyttää CP8-tar- kistusta. Muihin pääsisi mahdollisesti helpommin tai toisenlaisin varmistuk- sin", Kalliola painottaa. □



Symbolics 3645 ja 3675:

Kaksi uutta tehokasta Lisp-konetta

Nokia on julkistanut kaksi uutta Lisp-tietokonetta, Symbolics 3645 ja 3675. Ne ovat suorituskyvyltään jopa 50 prosenttia tehokkaampia kuin edeltäjänsä 3640 ja 3670.

Lisp-tietokoneet on kehitetty erityisesti symbolien käsittelyyn ja päättelyyn. Ne pystyvät käsittelemään tehokkaasti objekteja, listoja, tietorakenteita ja tietämyskantoja. Niinpä ne soveltuvat mainiosti tekoälysovellusten kehitysjärjestelmiksi.

Lisp-koneiden markkinajohtaja on Symbolics, jonka laitteita edustaa Suomessa ja Skandinaviassa Nokia Informaatiojärjestelmät.

Lisp-kone syntyi tarpeesta

Lisp-tietokoneiden kehitys alkoi 1970-luvun puolivälissä. Tuolloin MIT:n (Massachusetts Institute of Technology)

tekoälytutkimus ajautui vaikeuksiin. Laitteiden muisti- ja tehokkuusrajoitukset olivat tulleet tekoälyohjelmien pulonkaloiksi. Syntyi tarve kehittää kone, joka tehokkaasti tukisi Lisp-ohjelmointikieltä. Koneen tuli olla riittävän tehokas laajojenkin ohjelmistojen ajamiseen. Sillä tuli olla myös kehittynyt ohjelmointiympäristö.

Niinpä MIT:hen perustettiin 15 henkilön projektiryhmä, jonka tavoitteena oli rakentaa Lisp-tietokone. Ja viiden vuoden työskentelyn jälkeen koneen prototyyppi oli valmis.

Projektiryhmän mielestä kehitetty tekniikka oli liian hyvää heitettäväksi roskakoriin ja siitä oli syytä tehdä tuote. Prototyyppiä kaupattiin useille yrityksille. Mutta yksikään ei ollut kiinnostunut siitä.

Koska tekniikalle ei löytynyt ostajaa, projektiryhmän jäsenet päättivät perustaa oman yrityksen vuonna 1980. Näin syntyi yritys nimeltä Symbolics.

Tällä hetkellä yrityksessä työskentelee noin 600 henkilöä. Ja Symbolics on toimittanut tähän mennessä yli 1500 Lisp-konetta noin 300 asiakkaalle, joihin kuuluu suuryrityksiä, yliopistoja, korkeakouluja jne.

Ensimmäinen Symbolicsin tuote, Lisp-kone LM-2, tuli markkinoille vuonna 1981. Jatkokehityksen tuloksena valmistui edelleen Symbolics 3600 -tietokone, joka julkistettiin vuonna 1982. Kesällä 1984 Symbolics julkisti kaksi uutta tuotetta, Symbolics 3670 ja 3640 -koneet. Ja nyt on julkistettu taas kaksi uutta laitteistoa, Symbolics 3675 ja 3645.

Symbolics-koneiden merkittävin sovellusalue on asiantuntijajärjestelmät. Koneista noin 60 prosenttia on käytössä asiantuntijajärjestelmissä. Ohjelmistoprototyyppien suunnittelussa on peräti 21 prosenttia Symbolics-koneista. Muita tärkeitä sovellusalueita ovat CAD/CAE (16 %), robotiikka (17 %), luonnollinen kieli (14 %) ja suunnittelu (26 %).

Lisää suorituskykyä

Symbolics-koneissa on 36 bitin suoritin. Ne ovat tehokkaita ja hyvin rinnastettavissa supermineihin.

Uudet mallit 3645 ja 3675 perustuvat malleihin 3640 ja 3670, mutta ne ovat jopa 50 prosenttia edeltäjiään tehokkaampia. Tehon kasvun ovat mahdollistaneet IFU- ja ECS-yksiköt. IFU (Instruction Fetch Unit) esikäsittelee ohjelmakäskyä, ja se toimii rinnakkain pääsuorittimen kanssa. ECS (Extended Control Store) on laajennettu mikrokoodimuisti, jota hyödynnetään mm. Prolog-kielessä.

Symbolics 3645 ja 3675 -koneisiin käy sama ohjelmisto. Laitteiden erot ovat lähinnä massamuistikapasiteetissa.

Symbolics 3645 ja 3675 -koneiden peruskokoonpanossa on keskusmuistia neljä megatavua, mutta se on laajennettavissa 28 megatavuun. Virtuaalimuistin enimmäiskoko on 261 megatavua 3645:ssa ja 1000 megatavua 3675:ssa.

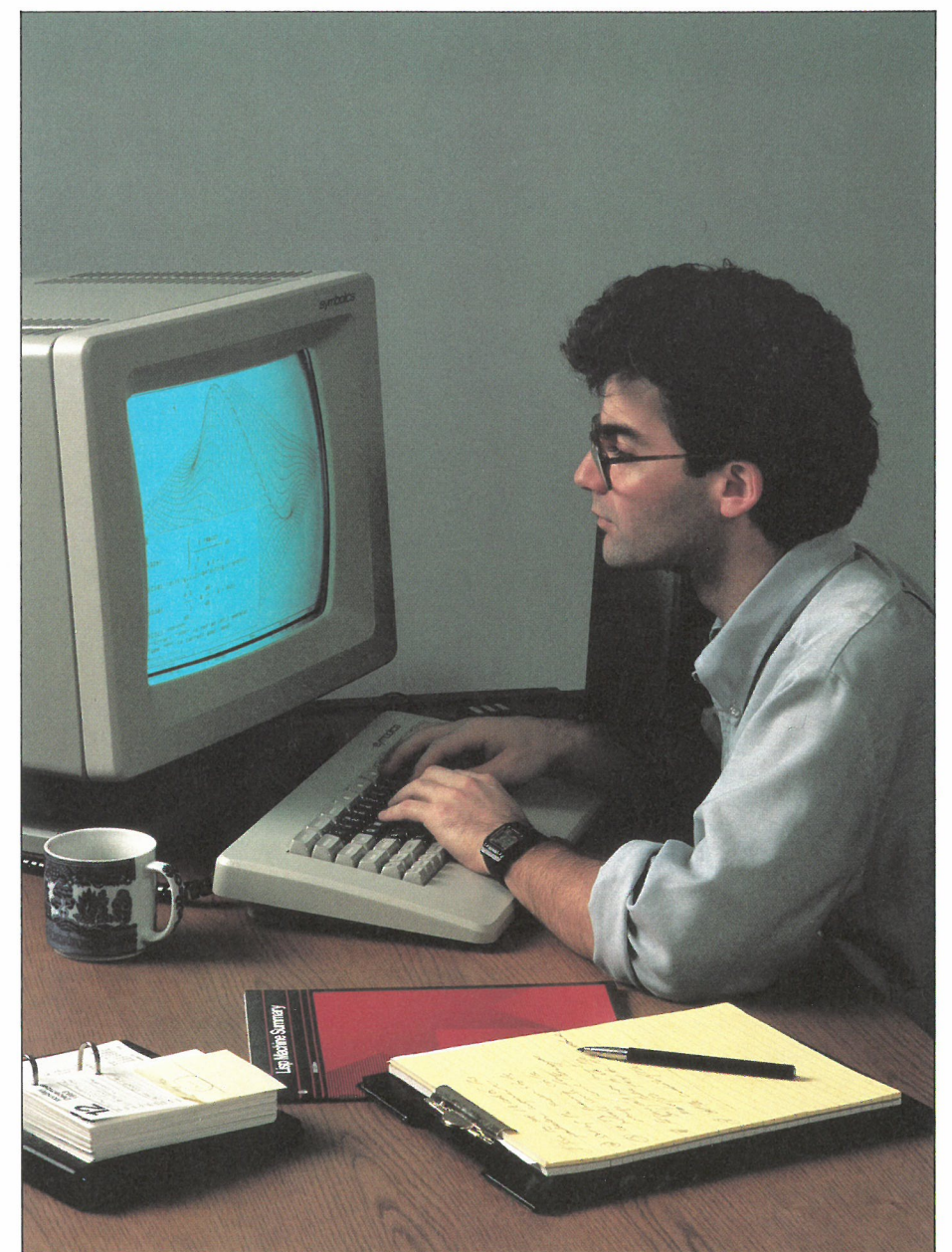
Levykapasiteettia on 3645:ssa 190-380 megatavua ja 3675:ssa 300-3800 megatavua. Molempiin koneisiin on saatavana mm. 45 megatavun nauha-ase. Vakiovarustukseen kuuluvat myös osoitinhiiiri ja Ethernet-liitäntä.

Symbolics-työaseman näytössä on jopa 1280 x 1024 pisteen tarkkuusgrafiikka. Mustavalkonäytön lisäksi on saatavissa erilaisia värinäyttöjä. Ja näyttö voidaan jakaa ikkunoihin.

Kehittynyt ohjelmointiympäristö

Integroidun ohjelmointiympäristön ansiosta Symbolics-koneet soveltuvat hyvin asiantuntijajärjestelmien rakentamiseen ja yleensä vaativaan ohjelmistokehitystyöhön. Koneissa ei ole erillistä käyttöjärjestelmää, vaan konetta ohjataan suoraan Lisp-kielillä.

Symbolics-järjestelmien pääkielenä on ZetaLisp, joka on yksi Lisp-kielen murteista. Se pohjautuu MIT:ssä kehitettyyn MacLispin, mutta sisältää lisäksi monia Lisp-koneympäristössä tarvittavia laajennuksia. Symbolics-koneissa on mahdollista käyttää myös muita Lisp-murteita, kuten esimerkiksi Common Lispia tai Interlispia.



Symbolics-koneisiin on saatavana Lisp-kielen lisäksi Fortran-77, Pascal, Prolog ja Ada-kielet. Ne on rakennettu Lispin päälle. Esimerkiksi jollain toisella laitteistolla tehty Fortran-ohjelma voidaan siirtää Symbolicsiin ja tehdä siihen lisäyksiä Lispillä.

Symbolicsilla ohjelmointiin on käytettävissä joukko hyödyllisiä apuohjelmia, kuten mm. Zmacs-näyttömuokain (editori) ja virheenetsintävälineistö.

Tieteellistekniseen työskentelyyn soveltuvia valmisohjelmistoja ovat esimerkiksi Macsyma ja S-Geometry. Macsyma on asiantuntijajärjestelmä matemaattisten ongelmien analyttiseen ratkaisemiseen. Macsymalla voi myös generoida matemaattisia yhtälöitä Fortran-ohjelmakoodiksi. S-Geometry puolestaan soveltuu mm. tietokoneavusteiseen suunnitteluun eli CADiin.

Liitettävissä muihin koneisiin

Symbolics-koneet tukevat älykkäisiin työasemiin ja nopeaan paikallisverkkoon perustuvaa hajautusta. Symbolics-koneet voi liittää monen eri valmistajan laitteisiin.

Symbolics-koneet ovat liitettävissä esimerkiksi VAX-laitteisiin DECnet-ohjelmistolla. Tällöin käyttäjän tiedostot voivat sijaita VAXilla ja varsinainen ohjelmistokehitys tapahtuu Symbolicsilla. DECnet-ohjelmisto mahdollistaa tiedostojen siirron lisäksi mm. elektronisen postin ja toisen koneen nauha-aseman käytön.

Symbolics-koneet pystyvät kommunikoimaan Ethernet-väylän kautta myös TCP/IP tai Chaosnet-liikennekäytäntöä tukevien tietokoneiden kanssa. Chaosnet on tarkoitettu Lisp-koneiden väliseen liikennöintiin. □

Asiantuntemus, Symbolics, Apollo, Mikko-Lisp, ...



Nokia tekoälyn kokonaistoimittajaksi

Nokia on tänään tekoälyjärjestelmien kokonaistoimittaja.

"Voimme tarjota asiakkaillemme täyden kokonaisjärjestelmätoimituksen alkaen asiantuntijajärjestelmäprojektin suunnittelusta sen toteutukseen. Tämän mahdollistaa toisaalta uusi kattava tuoteperheemme ja toisaalta riittävä asiantuntemuksemme", sanoo myyntipäällikkö Matti Virtanen Nokia Informaatiojärjestelmistä.

Tekoälytutkimus sai alkunsa 1950-luvun puolivälissä USA:ssa. Mutta vasta viime vuosina tekoälyä on alettu kaupallistaa. Ja tällä hetkellä se elää voimakasta nousukautta.

Nokialla tekoälytoiminta käynnistyi vuonna 1983, jolloin ryhdyttiin etsimään tekoälytutkimukseen soveltuvien Lisp-koneiden edustusta. Keväällä 1984 Nokia solmi edustusopimuksen Lisp-työasemakoneita valmistavan Symbolics Incorporated'in kanssa. Sopimus kattaa koko Skandinavian.

Tekoälyn kokonaistoimittajaksi

Ensimmäinen tehtävä oli rakentaa organisaatio tekoälytoimintaa varten. Nokia Informaatiojärjestelmissä tekoälytoiminnasta vastaa myyntipäällikkö Virtanen. Informaatiojärjestelmien tytäryhtiöissä Ruotsissa ja Tanskassa on myös omat tekoälyorganisaationsa.

Nokia Elektronikan Tutkimus- ja Kehitysosastolla toimii tekoälysovelluksien rakentamiseen perehtynyt ryhmä, jonka vastuulla on tekoälytuntemuksen ylläpito. Ryhmää vetää suunnittelu-päällikkö Pertti Lounamaa, joka on aiemmin työskennellyt Stanfordin yliopistossa tekoälytutkimuksen parissa. Hän on tehnyt töitä myös useissa tekoäly-yrityksissä.

Kevästä 1983 alkaen tavoitteena on ollut myydä tekoälytutkimuksen työvälineitä eli lähinnä Symbolics-koneita. Symbolics-järjestelmiä on toimitettu tähän mennessä kymmenkunta. Merkittävin kauppa on ollut Symbolics 3640 -laitteiston toimitus EC:n ESPRIT-projektiin.

"Nyt tavoitteena on tulla tekoälyalueen kokonaistoimittajaksi Skandinaviassa. Ensimmäinen edellytys on, että meillä on riittävän hyvä oma asiantuntemus tältä alueelta. Toiseksi meillä on nykyiselle tuotepohjallemme, Symbolics-Apollo-MikroMikko, hyvät ohjelmistot. Voimme tarjota asiakkaillemme tekoälyjärjestelmiä koulutuskäyttöön, vaativaan ohjelmistokehitystyöhön sekä sovelluskäyttöön.

Toisaalta tavoitteenaamme on tulla tekoälyn hyväksikäyttäjäksi eli käyttää tekoälyä omassa tuotannossamme", Virtanen sanoo.

Mitä tekoäly on?

Ei ole olemassa mitään yhtä ja ainoaa oikeaa määritelmää tekoälylle, vaan se voidaan määritellä monella eri tavalla riippuen lähestymistavasta.

Lounamaa luonnehtii tekoälyä lähinnä unelmaksi koneesta, joka pystyy kaikkeen mihin ihminenkin. "Kaikki se, mitä ei ole saavutettu ohjelmointitekniestä, on tekoälyä. Mutta kun se saavutetaan, se ei ole enää tekoälyä. Esimerkiksi älykäs shakkiohjelmisto oli aiemmin tekoälyä, muttei enää. Nyt se on vain tietokoneohjelma", Virtanen lisää.

Yksi tapa määritellä tekoäly on eritellä ne osa-alueet, joilla tekoälytutkimusta tapahtuu. Tekoälyn sovellusalueita ovat asiantuntijajärjestelmät, puheentunnistus eli signaalianalyysi, luonnollisen kielen tulkinta, hahmontunnistus (vision), robotiikka ja pelit.

Korvaako tekoäly ihmisen?

Virtanen empii ja vastaa: "Ei koskaan. Tietyissä tilanteissa kyllä. Siihen hän ollaan menossa. Esimerkiksi robotiikalla pyritään siirtämään fyysistä työtä koneille. Nyt puhutaan paljon asiantuntijajärjestelmistä, joiden avulla pyritään henkistä työtä siirtämään koneille."

Asiantuntijajärjestelmä tietämyksen käsittelyä

Asiantuntijajärjestelmistä on tullut varsin nopeasti yksi tekoälyn tärkeimmistä sovellusalueista.

Asiantuntijajärjestelmä on ohjelma, joka tietyllä kapealla alueella toimii asiantuntijan tavoin. Se koostuu tyypillisesti sovelluskohtaisesta tietämyskanasta ja päättelymekanismista.

"Asiantuntijajärjestelmä ei välttämättä eroa tavallisesta ohjelmasta käyttäjän kannalta. Tyypillistä asiantuntijajärjestelmälle on, että siltä voidaan kysyä, miksi se päätyi kulloiseenkin johtopäätökseen. Eli pyytää ohjelmaa perustelemaan omaa toimintaansa. Tähän normaali ohjelma ei tee. Ainoa tapa normaalisti on selvittää ohjelmakoodista, miksi näin tapahtui", Virtanen valaisee.

Ohjelmatekniikaltaan asiantuntijajärjestelmä eroaa tavallisesta ohjelmasta.

"Asiantuntijajärjestelmät käyttävät dynaamisia tietorakenteita. Niinpä ohjelmalle ei tarvitse ennalta varata käytettävän muistin kokoa. Itse asiassa myös ohjelmat ovat listarakenteita, joten ohjelma voi jopa muuttaa itseään.

Tyypillistä asiantuntijajärjestelmälle on myös loogikkaohjelmointi. Ohjelmalle määritellään ongelma. Ohjelman päättelymekanismi ratkaisee ongelman logiikan todistusmekanismeja käyttäen", Lounamaa kertoo.

Asiantuntijajärjestelmä merkitsee tietämyksen parempaa hyväksikäyttöä. Yrityksessä oleva tietämys on yleensä keskittynyt harvoille. Asiantuntijajärjestelmillä saadaan tietämys talteen ja kaikkien käytettäväksi. Yritys ei ole enää riippuvainen muutaman ihmisen tietämyksestä. Asiantuntijajärjestelmiä kutsutaankin tietämyspohjaisiksi järjestelmiksi.

Asiantuntijajärjestelmän kannalta tietämys ei ole pelkkä tietoa, vaan siihen liittyy tiettyjä lisäominaisuuksia. "Ensimmäkin tiedon rakenne on monimutkainen ja dynaamista. Tieto on myös tapauskohtaista. On tarkkaan tiedettävä tilanne, jossa tieto on pätevää. Tieto ei siis ole yleispätevää kuten tavallisissa ohjelmissa.

Koska tieto on epävarmaa ja kokemuseräistä, voidaan jokaiseen kriittiseen tietoon liittää numeerinen arvo, joka kuvaa sen epävarmuutta. Johtopäätöksiä tehdessään ohjelma huomioi nämä luvut ja ilmoittaa myös johtopäätöksen varmuuden", Lounamaa kuvaillee tiedon olemusta asiantuntijajärjestelmässä.

Asiantuntijajärjestelmiä jo käytössä

Asiantuntijajärjestelmiä on sovellettu käytäntöön mm. geologian, lääketieteen ja tietokonetekniikan alueella.

Asiantuntijajärjestelmiä käytetään tänään esimerkiksi laitteistovikojen etsintään. Niin ikään laitteistojen konfigurointi voidaan tehdä asiantuntijajärjestelmällä. Asiakas ilmoittaa, mitkä ominaisuudet hän haluaa laitteistoltaan. Järjestelmä kertoo sitten, kuinka paljon keskusmuistia tarvitaan, mitä kaapeleita tarvitaan jne.

Asiantuntijajärjestelmän voi sijoittaa ohjelmointikielen kääntäjään, jolloin kääntäjällä saadaan todella tehokasta koodia. Esimerkiksi Ada- ja Fortran-kielen on jo saatavana älykkäitä kääntäjiä.

Muista käytännön asiantuntijajärjestelmistä mainittakoon mm. mineraaliesiintymien analysointi ja monimutkaisen instrumenttien käytön neuvonta.



Suunnittelupäällikkö Pertti Lounamaa ja myyntipäällikkö Matti Virtanen luonnehtivat tekoälyä unelmaksi koneesta, joka pystyy kaikkeen mihin ihminenkin.

Projektit kouluttavat asiantuntijoita

Tekoälyasiantuntijoiden koulutus on Suomessa toistaiseksi puutteellista. Korkeakouluissa ja yliopistoissa koulutus rajoittuu lähinnä Lisp- ja Prolog-kielten opetukseen sekä erillisiin asiantuntijajärjestelmäseminaareihin. Varsinaisia tekoälyn koulutusohjelmia, joita järjestetään esimerkiksi Ruotsissa Uppsalassa ja Linköpingissä, ei Suomen korkeakouluissa vielä ole.

Asiantuntijoiden koulutus on lähinnä tekoälytoimintaa harjoittavien yritysten varassa. Erinäiset projektit lisäävät tekoälytuntemusta kuitenkin nopeasti. "Esimerkiksi ensimmäiset tekoälyprojektit Nokialla on käynnistetty lähinnä koulutusmielessä, jotta saisimme ihmisiä, joilla on jo kokemusta tekoälyn alueelta. Kaikki meneillään olevat Nokian sisäiset projektit ovat ensisijaisesti koulutusprojekteja. Sitten katsotaan, onko niissä mahdollisesti mitään tuoteistettavaa", Virtanen kertoo.

Vastaavia koulutusprojekteja ovat käynnistäneet VTT sekä Teknologian Kehittämiskeskus TEKES yhteistyössä eri yritysten kanssa. "Suomi on sen verran pieni maa ja tämä on sen verran vaikeaa teknologiaa, että tekoälytutkimuksessa tarvitaan yhteistyötä, jotta pysytään mukana", Lounamaa korostaa.

Huolimatta koulutuksen jälkeensäneisyydestä suomalainen tekoälytuntemus on eurooppalaisella tasolla. USA:ssa tekoälyteknologia on kyllä selvästi edellä. Samoin tietyiltä osin Ranskassa.

"Suomessa on lähdetty ennakkoluelottomasti soveltamaan asiantuntijajärjestelmiä käytäntöön. Esimerkiksi Ruotsissa on pitkään ollut tekoälytutkimusta, mutta siellä kiinnostuksen kohteena on ollut laitteistojen ja ohjelmointikielten kehittäminen. Uppsalassa suunnitellaan uudentyyppisen tietokoneen kehittämistä. Vastaavanlainen projekti on suunnitteilla myös Linköpingissä. Tekniikan soveltaminen käytäntöön on ollut Suomen lähtökohta tekoälyprojekteissa", Virtanen lisää.

Tekoälyprojekteja Nokialla

Tällä hetkellä Nokialla on meneillään kuusi sisäistä tekoälyprojektia.

Telenokiassa kehitellään puhelinkeskuksille asiantuntijajärjestelmää vianetsintään.

Nokia Elektroniikan Tutkimus- ja Kehitysosastolla on käynnistynyt projekti, jonka tavoitteena on kehittää asiantuntijajärjestelmä häiriötilanteiden selvittämiseen ydinvoimalassa. Ensimmäinen prototyyppi järjestelmästä pitäisi olla valmis ensi maaliskuussa.

Kolmannen projektin tarkoituksena on kehittää asiantuntijajärjestelmän rakentamistyökalu. Projektilla pyritään saamaan syvällisempää tietotaitoa päättelemekanismeista ja tietämyksen esittämistavoista.

Extend-projektissa rakennetaan tarjousten tekoon soveltuvaa asiantuntijajärjestelmää. Projekti on yksi TEKESin kokeiluprojekteista ja se tehdään yhteistyössä Teknillisen korkeakoulun kanssa.

Markkinat kasvavat

Tänä vuonna tekoälymarkkinoiden kooksi arvioidaan 100-200 miljoonaa dollaria. Vuosikymmenen loppuun mennessä markkinoiden ennustetaan kasvavan jo neljän miljardin suuruisiksi. Vuosittainen kasvuvauhti on siis lähes sata prosenttia.

Tällä hetkellä tekoälykauppa on etupäässä laite- ja työvälinekauppaa. Mutta sitä mukaa kun käytännön sovelluksia tulee markkinoille, tekoälymarkkinat muotoutuvat järjestelmäkaupaksi.

Vuosikymmenen lopulla tekoälyn markkinakakusta suurimmat palat lohkeavat ennusteiden mukaan tekoälykoneille, hahmontunnistukselle ja asiantuntijajärjestelmille. Sovelluskohteiden alueelliset erot voivat olla merkittäviä. Esimerkiksi USA:ssa hahmontunnistus on tärkein sovellusalue. Siellä hahmontunnistukseen sisältyy paljon sotilaallisia sovelluksia, satelliittikuvien tunnistusta, sääennusteiden tekoa sekä mineraalietäisyttä. Suomessa painopiste on sen sijaan asiantuntijajärjestelmissä.

Kattava tuotepohja

Nokian tekoälyjärjestelmien laitevalikoima on kattava.

Järjestelmien kehitystyöhön soveltuvat Symbolics 3600 -koneet, joissa käytetään pitkälle kehitettyä rakentamistyökaluja. Valmiit järjestelmät siirretään Apollo-työasemakoneisiin, jotka ovat puolestaan valmiiden tekoälyjärjestelmien käyttöympäristöjä.

MikroMikko 2 mahdollistaa tekoälyn perehtymisen jo pienin kustannuksin. Mikko-Lisp ja MikroMikko 2 soveltuvat mainiosti protoiluun ja tekoälyn koulutukseen. □

Ada-ohjelmakomponentti-kirjasto

Osa 2: Listankäsittelyrutiinit

Edellisessä Softarissa käsittelimme Ada-ohjelmakomponenttikirjaston merkinkäsittelyrutiineja. Tässä toisessa osassa esitellään listankäsittelyyn liittyvät rutiinit.

Ohjelmassa voidaan tarvita dynaamisia tietorakenteita. Tietorakenne on dynaaminen, jos sen koko vaihtelee ohjelman suorituksen aikana. Yksinkertaisin ja tavallisin dynaaminen tietorakenne on lineaarinen lista /2, s.55/.

Lineaariset listat

Lineaarinen lista määritellään joukoksi alkioita, joilla on tietty arvo ja paikka listassa. Esimerkiksi (a, b, c, d) on lista, jonka alkioita ovat a, b, c ja d. Alkiosta b tiedetään, että se seuraa alkiota a ja edeltää alkiota c.

Lineaarinen lista toteutetaan joko taulukon avulla tai ketjuttamalla. Kumpaa määrittelyä käytetään riippuu siitä, onko listan koon yläraja tiedossa vai ei. Jos yläraja on tiedossa, niin yksinkertaisin on toteuttaa lista taulukon avulla. Tämä tapa säästää sekä muistitilaa että suoritusaikaa.

Lista taulukon avulla

Adassa lista voidaan esittää taulukon avulla esimerkiksi seuraavasti:

```
generic
  type ELEMENT—TYPE is private;
  MAX—SIZE: NATURAL;
package LIST—PACKAGE is
  type POSITION—TYPE is new
    POSITIVE range 1..MAX—SIZE;
  type LIST—CONTENTS—TYPE is
    array (POSITION—TYPE)
      of ELEMENT—TYPE;
  type LIST—TYPE is
    record
      SIZE: NATURAL := 0;
      CONTENTS:
        LIST—CONTENTS—TYPE;
    end record;
end LIST—PACKAGE;
```

Lista talletetaan taulukkoon siten, että listan ensimmäinen alkio on taulukon alussa ja listan koon ilmoittaa muuttuja SIZE.

Uuden alkion lisäys listaan siirtää listan loppua yhden askeleen taulukon loppuun päin. Näin ollen alkion lisäys samoin kuin poistokin voi olla suhteellisen hidasta. Sen sijaan seuraavan tai edellisen alkion haku on nopeata.

Taulukolla toteutettu lista on itse asiassa valedynaaminen, koska ohjelmoijan kannalta listan looginen koko muuttuu, mutta sen fyysinen koko keskusmuistissa pysyy vakiona.

Lista ketjuttamalla

Todellinen dynaaminen lista voidaan määritellä ketjuttamalla esimerkiksi seuraavasti /1, s.104/:

```
generic
  type ELEMENT—TYPE is private;
  DUMMY: NATURAL := 0;
package LIST—PACKAGE is
  type TAG—TYPE is
    (HEADER, ATOMNODE);
  type NODE—TYPE
    (KIND: TAG—TYPE
     := HEADER);
  type LIST—TYPE is access
    NODE—TYPE;
  type POSITION—TYPE is new
    LIST—TYPE;
  type NODE—TYPE
    (KIND: TAG—TYPE
     := HEADER) is
  record
    LINK: LIST—TYPE;
  case KIND is
    when HEADER =>
      SIZE: NATURAL := 0;
    when ATOMNODE =>
      DATA: ELEMENT—TYPE;
  end case;
end record;
end LIST—PACKAGE;
```

Tässä rakenteessa alkion lisäys ja poisto eivät edellytä muiden alkiodien siirtämistä. Muistitilan varaus uudelle alkioille sen sijaan saattaa olla käyttöjärjestelmälle raskas toimenpide. Listaan kohdistuvien operaatioiden toteutus on helpompaa, kun listan ensimmäinen alkio ei sisällä varsinaista tietoa, vaan listan koon ja osoittimen varsinaiseen listaan.

Tässäkin rakenteessa tulee ongelmia tilankäytössä. Kun listasta varataan ja vapautetaan alkioita, täytyy ottaa kantaa siihen mitä vapautetuille muistipaikoille tehdään. Adassa ei ole automaattista 'roskankeruuta', vaan ohjelman on huolehdittava siitä. Jos muistitilasta ei ole pulaa, ohjelma saadaan toimimaan nopeammin, kun vapautettujen muistipaikkojen uudelleenkäyttöä ei huolehdita.

Listankäsittelyn tehokkuutta voidaan kohottaa lisäämällä tietorakenteeseen linkki edelliseen alkioon, jolloin taaksepäin selaus nopeutuu. Jos alkion koko on pieni, niin ylimääräisistä linkkikentistä voi aiheutua muistitilan tuhlausta. Adassa voi useampi prosessi käyttää samaa listaa yhtäaikaisesti. Tällöin on

huolehdittava siitä, ettei kaksi prosessia päivitä listaa samanaikaisesti, mikä sotkisi rakenteen. Mutta Adassa rakenne voidaan määritellä tehtävässä eli taskissa, joka automaattisesti estää samanaikaisen päivityksen. Tämä on kuitenkin hitaampaa kuin tavallinen määrittely. Lista voidaan määritellä myös siten, että taulukon alkio sisältää ketjun seuraavaan alkioon.

Listankäsittely voidaan siis toteuttaa usealla eri tavalla, joilla kullakin on hyvät ja huonot puolensa. Taulukolla toteutettu listarakenne on nopea, mutta listan koolla on ennalta määritetty yläraja. Jos listan koon yläraja on tiedossa, lista kannattaakin toteuttaa taulukkona. Jos taas ylärajaa ei tiedetä ja lista toteutetaan ketjuttamalla, niin muistitilaa saattaa tuhlaantua ketjuihin. Näin varsinkin listoissa, joiden koko ei muutu.

Ohjelmoijan on etsittävä sopiva kompromissi rajoitusten, ajankäytön ja muistitilan suhteen. Listan toteutukseen tarvitaan siis useita erilaisia pakkauksia, joiden käyttö riippuu kulloisestakin tilanteesta.

Operaatiot

Lineaariseen listaan liittyvät operaatiot ovat listan luonti, alkion lisäys ja poisto listasta, tietyn alkion haku ja muuttaminen sekä listan koon tutkiminen.

Listankäsittely on ohjelmoijalle joustavampaa kuin esimerkiksi tiedoston käsittely, koska listaa voidaan käsitellä sekä eteenpäin että taaksepäin. Lisäksi ohjelma voi tallettaa osoittimia mihin tahansa listan alkioon myöhempää käyttöä varten.

Kun listankäsittelyyn on tehty standardioperaatiot, ohjelmoijan ei tarvitse miettiä uudelleen ja uudelleen samoja yksityiskohtia. Myös ohjelmakoodin lukeminen on helpompaa, kun käytetään samoja standardioperaatioita. Käyttäjän kannalta on mukavaa, että operaatioiden muoto ei riipu listan toteutustavasta.

Listan tyhjentäminen

Esiteltäessä esimerkiksi kokonaislukuja sisältävä lista

```
package LISTA is new
  LIST—PACKAGE (INTEGER, 100);
```

se saa maksimikookseen 100 alkioita, mutta sen looginen pituus on nolla. Jos käytetään ketjutettua toteutusta, listan koolla ei ole merkitystä kunhan vain laitteiston resurssit riittävät. Samassa ohjelmassa voidaan määritellä useita erilaisia listoja, joiden alkuarvoksi tulee tyhjä lista. Jos kesken suorituksen lista pitää tyhjentää, niin proseduri EMPTY (L) tekee sen.

Listan kokoa voidaan tutkia funktiolla IS—EMPTY (L) ja IS—FULL (L) sekä funktiolla LENGTH (L), joka palauttaa arvonaan listan alkiodien lukumäärän. IS—EMPTY ja IS—FULL palauttavat arvonaan BOOLEAN tyyppiset muuttujat, TRUE tai FALSE.

Alkion paikantaminen

Kun listan alkioihin kohdistetaan operaatioita, täytyy tietää sen listan alkion paikka, jonka suhteen operaatiot toteutetaan. Niinpä tarvitaan operaatiot, joilla päästään kiinni listan alkuun ja loppuun. Funktio FIRST (L) palauttaa arvonaan listan ensimmäisen alkion paikan ja funktio LAST (L) palauttaa listan viimeistä alkioita seuraavan alkion paikan. Tyhjän listan käsittely ei eroa muiden listojen käsittelystä. Jos siis tyhjään listaan kohdistetaan operaatio FIRST (L), niin funktio palauttaa arvonaan LAST (L):n.

Funktiolla NEXT (L, P) saadaan P:tä seuraavan alkion paikka ja siis listan loppuessa paikka LAST (L). Tämän menettelyn etuna on se, että ohjelmoija voi käsitellä listaa samalla tavalla kuin tiedostoa. Listan taaksepäin selailuun on funktio PREVIOUS (L, P), joka palauttaa arvonaan P:tä edeltävän alkion paikan. Taaksepäin selailu on varsin hidasta, jos lista on toteutettu ketjuttamalla. Tätä voidaan nopeuttaa määrittelemällä toinen ketju, joka osoittaa kunkin alkion edelliseen alkioon. Tämä kaksinkertainen ketjutus vie kuitenkin enemmän muistitilaa.

Listan päivitys

Proseduuri INSERT (L, P, A) lisää listaan L alkion A paikassa P olevan alkion eteen. Niinpä, jos P = LAST (L), alkio sijoitetaan listan viimeiseksi. Jos taas P = FIRST (L), niin alkio lisätään listan alkuun.

Alkio poistetaan listasta paikasta P proseduurilla DELETE (L, P). Operaation jälkeen P osoittaa poistettua alkioita seuraavaa alkioita.

Listan alkion sisältöön päästään käsiiksi funktiolla RETRIEVE (L, P), joka palauttaa arvonaan paikassa P olevan alkion sisällön.

Alkion arvoa voidaan muuttaa proseduurilla CHANGE. Esimerkiksi CHANGE (L, P, A) korvaa paikassa P olleen alkion alkioilla A.

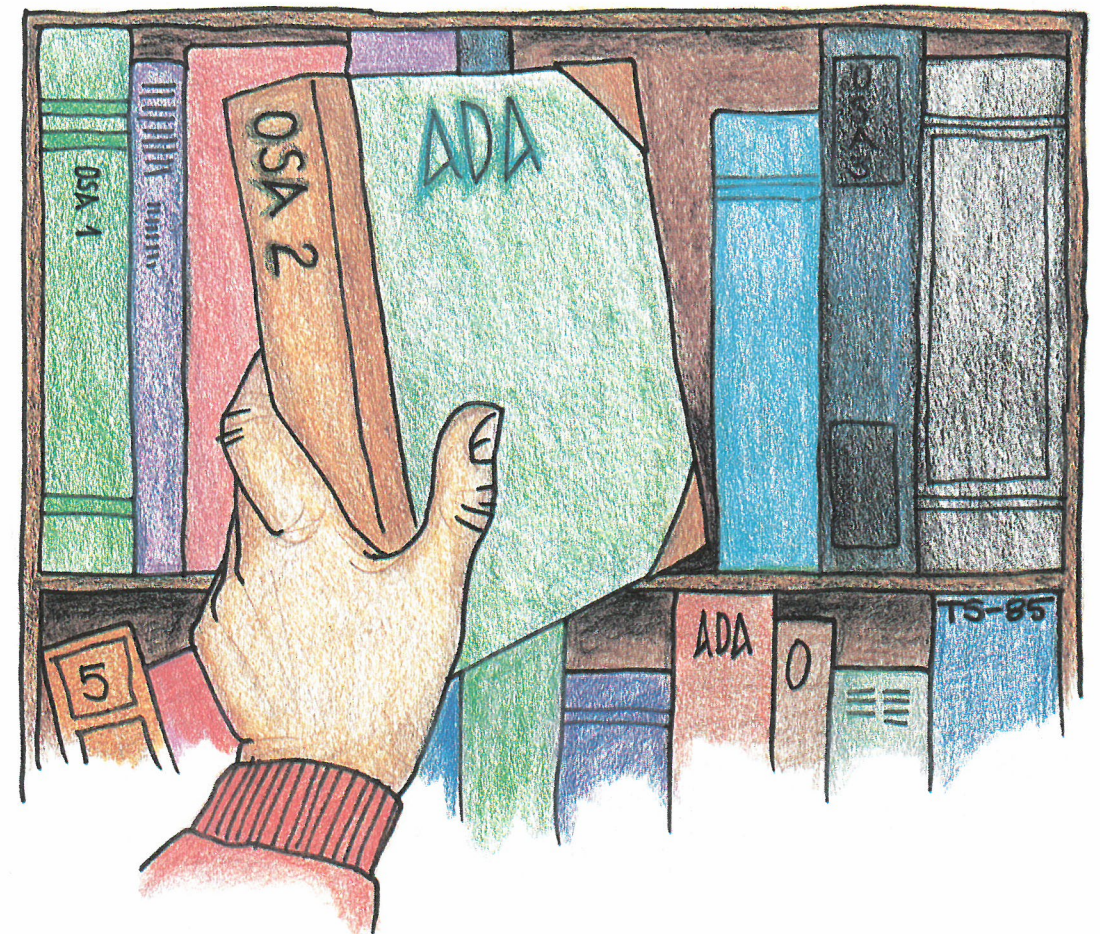
Poikkeusmekanismi virhetilanteissa

Listankäsittelyssä on varauduttava myös virhetilanteisiin. Adan poikkeusmekanismi on sopiva toteutusväline.

Kutsuttaessa jotain aliohjelmaa virheellisellä parametriarvolla kyseinen aliohjelma 'nostaa poikkeuksen'. Tällöin aliohjelman suoritus keskeytyy ja palataan kutsuvaan ohjelmaan. Poikkeus voidaan havaita kutsuvassa ohjelmassa ja toimia tilanteen vaatimalla tavalla. Hyvä ohjelma voidaan laatia siten, että poikkeuksia ei nouse ajoikana.

Kirjallisuusviitteet

1. Olsen, E.W. & Whitehill, S.B., Ada for Programmers. Reston Publishing Company, Inc. Virginia 1983.
2. Räihä, K.-J., Tietorakenteet. Helsingin Yliopisto, Tietojenkäsittelyopin laitos. Helsinki 1983. □



Adasta ja sen tulosta on puhuttu pitkään. Nyt Ada on tullut.

Mutta miten Ada syntyi? Ja miten laajasti sen käyttö on jo yleistynyt?

Ada eilen ja tänään



Ohjelmointikielen kehittämistä voidaan verrata talon rakentamiseen. Jos olet rakennuttamassa taloa itsellesi, sinulla on varmasti tiettyjä toiveita ja vaatimuksia talollesi. Vain sinä voit tietää ne. Ei arkkitehtisi. Sen sijaan hän voi sitten suunnitella talon, joka täyttää vaatimuksesi.

On siis tärkeää pitää vaatimusten määrittely ja varsinainen suunnittelu erillään. Näin on menetelty Adaa kehitettäessä. Aiempien ohjelmointikielten, esimerkiksi Pascalin, suunnittelussa sen sijaan samat henkilöt päättivät, millainen kielen pitäisi olla ja miten kieli toteutetaan. Aivan kuin antaisit arkkitehdin suunnitella itsellesi hänen tarpeitaan vastaavan talon.

Seuraavissa Softareissa käsittelemme Adan vaikutuksia ohjelmointityön laatuun ja Adan käyttökokemuksia.

Ohjelmistokustannukset kaiken alku

Jo 1970-luvun alkupuolella Yhdysvaltain puolustusministeriö, DoD (Department of Defense), alkoi kiinnittää huomiota nopeasti kasvaviin ohjelmistokustannuksiinsa. Vuonna 1973 ministeriö käytti ohjelmistoihin yli kolme miljardia dollaria. Se oli noin 46 prosenttia ministeriön koko atk-kustannuksista.

DoD analysoi tilanteen ja havaitsi käyttävänsä noin 400 eri ohjelmointikieltä. Kielten runsaus vaikeutti mm. ohjelmistojen ylläpitoa. Ohjelmissa ilmeni jatkuvasti virheitä, joita korjailtaessa syntyi taas uusia. Ohjelmointimenetelmissä oli niin ikään parantamisen varaa. Strukturoitu ohjelmointi näytti olevan tuntematon käsite.

Ministeriössä päätettiin, että ohjelmistokustannukset oli saatava kuriin. Ohjelmointikielten määrä oli rajoitettava järkevälle tasolle. Kieliä ei saanut olla montaa ja niiden tuli olla korkean tason ohjelmointikieliä. Sallittavilla kielillä kirjoitettujen ohjelmien tuli olla mm. helposti ylläpidettäviä, tehokkaita ja toimintavarmoja.

Määrittelyä lähes kolme vuotta

Tammikuussa 1975 DoD asetti työryhmän, ns. HOLWG-ryhmän (High Order Language Working Group), jonka tehtävänä oli laatia suositus puolustusministeriön ohjelmistohankkeissa sallittavista ohjelmointikielistä.

Ryhmän ensimmäinen tavoite oli määrittellä kielten laadulliset vaatimukset. Määrittelyvaihe kesti lähes kolme vuotta.

Ryhmän työn tuloksena syntyivät v. 1975 STRAWMAN- ja WOODENMAN-raportit, v. 1976 TINMAN-, v. 1977 IRONMAN- sekä v. 1978 STEELMAN-raportti. Raporteissa määriteltiin yhä tarkemmin ne vaatimukset, jotka ministeriön suosituksen hyväksyttävien kielten tuli täyttää.

Yli 200 eri henkilöä, jotka edustivat 85 USA:n puolustusministeriön osastoa, 26 teollisuuslaitosta, 16 yliopistoa ja 7 muuta organisaatiota, osallistui vaatimusten määrittelyyn. Voisi kuvitella, että niin monen henkilön osallistuessa prosessiin tuloksena olisi syntynyt epäyhtenäinen kokoelma erilaisia vaatimuksia. Mutta näin ei ollut, vaan hämmästyttävää kyllä vaatimukset olivat hyvin yhteneväiset.

Kieliltä vaadittiin mm. ohjelmakoodin luettavuuden ja ylläpidon helpoutta, luotettavuutta, siirrettävyyttä sekä tehokkuutta. Kielien piti tukea strukturoitua ohjelmointia. Kielien oli sovellettava laajojen reaaliaikaisten järjestelmien ohjelmointiin.

Vaatimuksissa pidettiin tärkeämpänä ohjelmien luettavuutta kuin kirjoitettavuutta. Aiemmin ohjelmointikieliä kehitellessä oli suoraviivaisesti pyritty siihen, että ohjelmien koodaus olisi nopeaa. Koska puolustusministeriön ohjelmat ovat pitkäikäisiä ja niitä joudutaan usein muuttamaan, luettavuus onkin ensiarvoisen tärkeää.

Jo varhain määrittelyvaiheessa todettiin, ettei mikään senaikaista ohjelmointikielistä täyttänyt vaatimuksia. Oli siis kehitettävä uusi kieli.

Oliko sitten käytännössä ylipäättään mahdollista kehittää vaatimukset täyttävä kieli? Sitä selviteltiin vuonna 1977 kahdessa erillisessä tutkimuksessa. Molempien tulokset olivat positiiviset. Lisäksi selvitysten mukaan sitoutuminen uuden kielen käyttöön merkitsi DoD:lle satojen miljoonien dollareiden säästöä vuosittain.

Kehittely kilpailuna

Vuonna 1977 päätettiin aloittaa uuden kielen kehittäminen. Kielelle annettiin työnimeksi DoD-1.

Koska kielestä piti tulla korkealuokkainen ja USA:n puolustusministeriön ulkopuolellakin hyväksytty, ministeriö järjesti kansainvälisen suunnittelukilpailun. Kilpailuun kutsuttiin 20 suunnitteluryhmää eri puolilta maailmaa.

Ryhmistä valittiin neljä varsinaiseen suunnittelukilpailuun. Näitä neljää ryhmää, jotka edustivat eri yrityksiä, nimettiin väreillä sininen, keltainen, punainen ja vihreä:

- SofTech (sininen)
- SRI International (keltainen)
- Intermetrics (punainen)
- Cii Honeywell Bull (vihreä)

Suunnitteluryhmät kehittivät kieltä puoli vuotta alkaen elokuussa 1977. Helmi-maaliskuussa 1978 ryhmien aikaansaannokset joutuivat puntarointiin. Noin 400 asiantuntijan raati kelpuutti vihreän ja punaisen ryhmän 'fiinaaliin' eli toiseen kehitysvaiheeseen.

Keväällä 1979 vihreän ja punaisen ryhmän jatkokehityksen tulokset otettiin tarkasteluun. Tuolloin tehtiin lopullinen ratkaisu suunnittelukilpailun voitosta. Voittajaksi nimettiin vihreä ryhmä. Sen ratkaisu, kuten muidenkin ryhmien, pohjautui pitkälle Pascal-kielen.

Samana keväänä ristittiin kieli Adaksi kreivitär Augusta Ada Lovelacen mukaan. Ada Lovelace (1815—1851) oli matemaatikko, joka työskenteli Charles Babbagen kanssa. Ada keksi, kuinka Babbagen differenssikoneita voitiin ohjelmoida. Niinpä Adaa pidetään ensimmäisenä ohjelmoijana.

Kohti standardointia

Suunnittelukilpailun ratkeaminen ei vielä merkinnyt sitä, että kieli olisi ollut valmis. Silloin todellinen kielen kehittyly vasta alkoi.

Vihreä ryhmä jatkoi kehitystyötään tiiviissä yhteistyössä noin sadan ympäri maailmaa olevan asiantuntijan kanssa. Tätä vuonna 1979 alkanutta kolmatta kehitysvaihetta kutsutaan 'Test and Evaluation' -vaiheeksi.

Vuonna 1980 valmistui ensimmäinen versio Adasta. Ja lopulta helmi-kuussa 1983 Ada standardoitiin.

Tänään useita kääntäjiä

Adan standardoinnin jälkeen on valmistunut parisen kymmentä virallisesti hyväksyttyä eli validoitua Ada-kääntäjää, ja uusia on jatkuvasti tulossa.

Ada-kääntäjien valmistumista on hidastuttanut validoituille kääntäjille asetetut korkeat vaatimukset. Tullakseen virallisesti hyväksytyksi kääntäjän on suoriuduttava yli 2000 testiohjelmasta. Ada-kääntäjän pitää tarkistaa paljon enemmän kuin tavallisen kääntäjän. Toisaalta tämä edesauttaa ohjelmien testausta, koska ohjelmointivirheet paljastuvat jo käännösvaiheessa.

Ensimmäinen Ada-kääntäjä valmistui New Yorkin yliopistossa keväällä 1983. Kääntäjällä oli enemmän historiallista kuin käytännön merkitystä. Se oli tulkkipohjainen ja siten hidas. Niinpä se soveltui lähinnä vain opetustarkoitukseen.

Toisella kääntäjällä, jonka tekijöinä olivat Rolm ja Data General, oli jo käytännön arvoa. Kääntäjä validoitiin kesäkuussa 1983 ja se toimi Data Generalin Eclipse -minikoneella.

Ensimmäinen eurooppalainen Ada-kääntäjän tekijä oli Dansk Datamatik Center (DDC), joka on kehittänyt Ada-kääntäjiä mm. Honeywellin DPS 6 ja 8 -laitteistoille sekä DECin VAX-koneille. Tämä tanskalainen ohjelmistotalo on tehnyt koneriippumattoman osan myös Nokian MPS 10 -minikoneen ensimmäiseen Ada-kääntäjään. Koneriippuva osa oli sen sijaan Softplanin tekoa. Nyt MPS 10:ssä toimii Softplanin koneaan tekemä Ada/MPS-kääntäjä.

Nyt kun markkinoilla alkaa olla kääntäjiä, Ada-kehityksen painopiste on siirtymässä ohjelmankehitysvälineisiin.

Koska Adan kehittämisessä pääpaino oli koodin luotettavuudessa, ylläpidettävyydessä ja tehokkuudessa eikä niinkään ohjelmoinnin nopeudessa, ei uusien Ada-ohjelmien teko ole välttämättä nopeaa. Niinpä ohjelmoinnin tuottavuuden nostamiseksi tarvitaan erilaisia ohjelmakehitysvälineitä sekä Ada-ohjelmistokomponentteja.

Käyttö yleistyy

Adan käyttöönotto on ollut odotettua hitaampaa. Suurimpana syynä on ollut validoitujen kääntäjien puute. Kääntäjien yleistyessä Adan käyttö on vilkastumassa.

Niin ikään Adan tuloa on hidastuttanut sen uutuus. Vuosikaudet Cobol-koodia kirjoittanut ohjelmoija ei hevin halua siirtyä uuteen kielimaailmaan.

Tietysti koulutuksen puuttumista voidaan pitää osasyllisenä. Koulutus ei ole enää tänään ongelma. Sitä on saatavissa, jos vain löytyy haluja.

Ja toistaiseksi on paljon vanhoilla kielillä tehtyjä ohjelmia, joita ei ole syytä laittaa uusiksi.

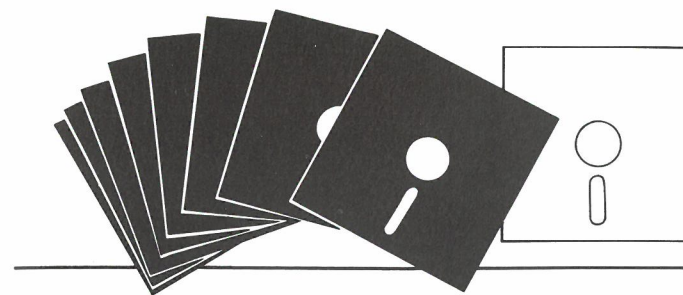
Tällä hetkellä ei Adalla kirjoitettuja sovellusohjelmia ole vielä markkinoilla yleisesti saatavana. Lähivuosina ohjelmistokauppoihin odotellaan jo yleiskäyttöisiä Ada-sovelluksia. Näille Ada-ohjelmapaketeille povataan hyviä markkinanäkymiä; ovathan Ada-ohjelmat koneriippumattomia.

Yhdysvaltain puolustusministeriössä on käytetty Adaa tammikuusta 1984 alkaen kaikkien kriittisten ohjelmien tekoon. Vuosikymmenen lopulla ministeriössä uskotaan käytettävän Ada-ohjelmistoihin vuosittain 15 miljardia dollaria.

Äskettäin myös Yhdysvaltain ilmailu- ja avaruushallinto, NASA, on päättänyt käyttää Adaa 1990-luvun avaruusasemien tietokoneiden ohjelmointiin. New Yorkin St. Luke's-Rooseveltin sairaalassa monimutkainen munuaisdialyysi tehdään Ada-ohjelmitoilla. Ja myös USA:n teollisuus on innokkaasti ottamassa Adaa käyttöönsä sekä teknistieteellisten että kaupallishallinnollisten sovellusten ohjelmoinnissa. Esimerkiksi Moog Electronics on toteuttanut Adalla muovin valmistuksen ohjausjärjestelmänsä.

Pankit ja vakuutuslaitokset ovat olleet maailmanlaajuisesti kiinnostuneet Adasta. Se on ymmärrettävää, sillä Ada soveltuu laajojen, toimintavarmojen tosiaikajärjestelmien ohjelmointiin. Tärkeimpiä Adan käyttökohteita Suomessa ovat olleet juuri pankkien uudet toimistojärjestelmät.

Ruotsissa merkittävin Adan käyttöönotto on meneillään paikallisessa puhelinlaitoksessa Televerketissä. □



MikroMikko TopTen

Tämänkertainen MikroMikkojen ohjelmistalista perustuu kuluvan vuoden lokakuun myyntiluvuista lasketuihin suhteellisiin indekseihin.

Listan kärjessä on tapahtunut suuria muutoksia. WordStar ja K-Mies ovat nousseet ohi pitkäaikaisen ykkösen, Multiplanin. Kuten viimeksi ennakoitiin, Lotus 1-2-3 on aloittanut nousun. Uusina tulokkaina listalla ovat WordPerfect-tekstinkäsittely sekä VT-100 -emulaattori.

MikroMikko TopTen

1. (8)	WordStar	56
2. (6)	K-mies	51
3. (1)	Multiplan	32
4. (7)	MS-Basic -tulkki	28
5. (-)	VT-100 emulaattori	27
6. (10)	Lotus 1-2-3	25
7. (-)	WordPerfect	21
8. (3)	Edix	20
9. (4)	Wordix	16
10. (2)	Multigraph	14

Suluissa sijoitus edellisessä Top-Tenissä (Softari 3/85).

Grafiikkaa PC-INFOon

MikroMikko 2:n INFO-tietokannasta voi nyt tehdä graafisia esityksiä. Tämä käy Infograph-ohjelmalla. Infographilla voidaan myös välittää PC-INFO:n tiedot Multiplan-ohjelman käyttöön.

Infograph piirtää Multigraph-tyyppisiä kuvia, jotka voivat olla piirakoita, pylväs- tai viivakuviota. Se tulostaa valmiit kuvat grafiikalla varustetulle kirjoittimelle tai piirturille.

Uusi myymäläpäätte Nokiaalta

Nokia on julkistanut uuden kassapäätteen, CRS 300. Nokia on myös kehittänyt uusiin kassapäätteisiin perustuvan myymäläjärjestelmän, joka soveltuu pieniin päivittäistavarakauppoihin.

CRS 300 -kassapäätte voidaan liittää NETNET-paikallisyhteyteen yhdessä MikroMikkojen kanssa. Kassapäätteeseen kuuluu pienikokoinen kuittikirjoitin, matala näppäimistö sekä erilliset myyjä- ja asiakasnäytöt. Sen oheislaitteina voivat olla samanaikaisesti kassa-vaaka, pöytäscanner, lukukynä sekä magneettikortin lukija.

Uusi myymäläjärjestelmä koostuu NETNET-väylästä sekä siihen liitetyistä CRS 300 -päätteistä ja MikroMikko 2:ista. Yleisin kokoonpano on 4—5 kassapäätettä ja kaksi MikroMikko 2 -keskysyksikköä, joissa on varsinainen myyntiohjelmisto. Järjestelmään sisältyy myös yhteydet raha- ja luottolaitoksiin.

Järjestelmään liitettyä MikroMikko 2 -laitteistoa voi käyttää muihin tehtäviin kassatoimintojen häiriintymättä. Esimerkiksi MikroMikko 2:n valmisohjelmistot ovat hyödynnettävissä.

Summer lisää Multiplanin käyttömahdollisuuksia

Summer on ohjelma, jolla voidaan siirtää tietoja MikroMikko 2:n tiedostoista Multiplan-taulukoihin. Summer on saanut Softarissa 2/84 tapahtuneen esittelyn jälkeen monia käyttäjien toivomia lisäominaisuuksia.

Summerilla voidaan nyt laskea yhteen ja vähentää toisistaan samanmuotoisia Multiplan-taulukoita. Summer siirtää tarvittavat luvut ja merkkijonomuotoiset tiedot suoraan ulkopuoliseen tiedostosta Multiplan-taulukon alkioihin. Siirron yhteydessä se lisäksi summaa tietoa valmiiksi yhteenvedoksi (ristintaulukointi).

Tiedostonimissä voi käyttää sekä hakemistoreittejä että yleistäviä viitteitä. Uusi Summer toimii myös MikroMikko 2 -lähiverkossa.

Lisätietoja: Tmi Mikrola, puh. 90-515159

Uusia Honeywell-emuja MikroMikko 2:een

MikroMikko 2:een on saatavissa uudet työasemaohjelmistot WS7300 ja WS7800. Ohjelmistoilla MikroMikko 2 saadaan toimimaan Honeywellin VIP7300 ja VIP7800-päätteiden tavoin.

WS7300-työasemaohjelmisto emuloi sekä VIP7301- että VIP7303-päätettä. Yhteys isäntäkoneeseen on asynkroninen. Ohjelmistoon voidaan liittää käyttäjän omia aliohjelmia ja näppäimistö on käyttäjän määriteltävissä.

Asynkroninen WS7800-päätte-emulaattori emuloi sekä VIP7801- että VIP7803-päätettä. Sen sijaan synkroninen päätte-emulaattori WS7800 emuloi VIP7804-päätettä.

WS7800-ohjelmisto vaatii 512 kilotavua keskusmuistia, kun taas WS7300 toimii jo 256 kilotavun muistilla varustetussa MikroMikko 2:ssa. Molemmat työasemaohjelmistot mahdollistavat tiedostojen siirron MikroMikon ja DPS 6:n välillä.

Sekä WS7300- että asynkronisella WS7800-emulaattorilla voidaan MikroMikkoa käyttää TOIMI-ohjelmiston työasemana. TOIMIn toiminnot on sijoitettu MikroMikon pehmonäppäimiin.

Nokia
Softari-lehti
PL 780
00003 HELSINKI

Vastauslähetyks
HKI 10 Lupa 2509

Nokia
maksaa
postimaksun

Softari 1/86

Palvelukortti

☐ Haluan esitteitä seuraavista
Nokian ohjelmistoista

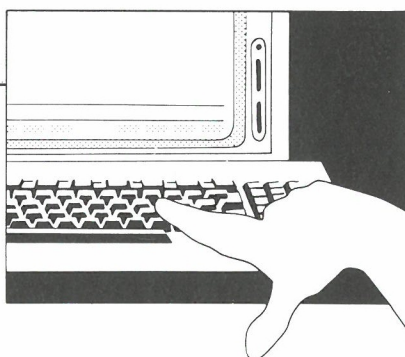
☐ Pyydän, että minuun otetaan
yhteyttä.
Olen kiinnostunut

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

nimi _____
tehtävä _____
yritys _____
osoite _____

JOUKKOJULKAISU

21. pölynimuri



liikaa

Maailmassa on niin paljon ydinaseita, että ihmiskunta voitaisiin tuhota ainakin 20 kertaa. On pelit ja vehkeet, jos joskus tulee tarvetta saada koko jengi hengiltä. Samalla nollautuu kaikki se, mitä ihmiset ovat saaneet aikaan. Kätevää vai mitä?

Kun kuulemme, että ydinsota on syttynyt kaikki suuret ja pienet elämän ongelmat vaihtuvat yhteen vaivaiseen pulmaan. Mistä hitosta saada nopeasti ja kivuttomasti tappavaa myrkkyä, jota voi nauttia ennen saastepilven tai ydintalven ennättämistä.

Ennen sota oli kiduttavampaa. Vankileireillä viruneet vangit, juoksuhaudoissa värjötelleet sotaurhot tai pommi-suojissa pelkäävät siviilit eivät voineet lopettaa kärsimyksiään hyvän sään aikana. Aina oli pientä toivoa eloonjäämisestä. Nyt ei ole. Kenenkään ei enää tarvitse kärsiä suotta. Viisaat päättäjät ovat ajatelleet ihmisten parasta — loppuun asti.

Miksi pommeja tarvitaan kipeästi lisää, vaikka riittävä tuhovoima on saavutettu ja moneen kertaan varmistettu? Jos minulla on kaapissani 20 toimivaa pölynimuria, miksi minun pitäisi ostaa uusi, vaikka on muutenkin tiukkaa rahan kanssa?

Yhä useampien, ns. pehmojen, mielestä 20 imuria riittää. Kyllä niillä jo puhdasta tulee tarvittaessa. Kuoleman laatu ei uusista aseista niin paljoa parane, että niihin uhratut miljardit olisivat alkuunkaan kannattavaa.

Kernaasti pehmot vaihtaisivat sukellusveneosuutensa autoon tai tiskikoneeseen, koska niiden olemassaolo vaikuttaa elämisen tasoon joka ainoa päivä eikä vain sen yhden kerran. Ei silti, etteikö ydinase vaikuttaisi vahvasti. Ja ei vähiten lapsiin.

Pehmot sanovat haluavansa maksaa vain 'hyödyllisten' laitteiden kehittämisestä. Esimerkiksi kerroskuvauslaitteisto, joka kuvaa ihmisen sisuskalut olisi sellainen. Jokainen tahtoo säästää hipiänsä viimeiseen asti kirurgin ilkeästi vihlovalta veitseltä. Pehmot korostavat, että tämä siunauksellinen laite olisi käyttämättömänäkin täsmälleen yhtä tuottava kuin uusi pommi. Heidän mielestään olisi mukavampaa ostaa videot tai autopuhelin kuin se 21. pölynimuri.

Saas nähdä, montako imuria ihmiset aikovat vielä kustantaa.

Niin kauan kun palkka juoksee, insinöörit jatkavat suunnittelua; vaikka ruudin keksiminen yhä uudestaan voi alkaa kyllästyttää. Tosi kauppamiehet myyvät vaikka mitä kunhan joku vain maksaa. Ja poliitikot ostavat kiltisti aseita niin kauan kun saavat aseellisuudelta sopivaa täydennystä vaalikassansa ja ulkomaanmatkoja Geneven aserajoitusneuvotteluihin.

Kun ihmiset kieltäytyvät ostamasta samaa asejärjestelmää 21. kertaa, voidaan sama imuri myydä vielä monta kertaa uudestaan. Tosin nyt puolustusjärjestelmän nimikkeellä. Ja taas mäntii ostaa innolla.

Softaja