

Softari

NOKIAN OHJELMISTOLEHTI 2/86



Työt puoleen — tuottavuus tuplaksi

Ohjelmistojen laatukysymykset ovat viime aikoina saaneet runsaasti huomiota osakseen. Julkinen sana on populistisesti keskittynyt ohjelmistovirheiden aiheuttamiin vahinkoihin. Se on tietysti tärkeä, ehkä jopa tärkein näkökulma. Toinen mielenkiintoinen näkökulma on virheiden vaikutus työn tuottavuuteen.

Me ohjelmistoammattilaiset käytämme liian paljon, jopa kymmeniä prosentteja, työpanoksestamme virheiden korjaamiseen ja asioiden uudelleen tekemiseen. Tämä on periaatteessa täysin turhaa ja tuottamatonta työtä. Eihän ole järkevää tehdä ensin virheitä ja sitten työllä tuskalla korjata niitä. Tässä piilee valtava tuottavuuspotentiaali; tekemällä vähemmän virheitä käyttäisimme vähemmän aikaa niiden korjaamiseen ja voisimme kohottaa työn tuottavuuden jopa kaksinkertaiseksi.

Otetaan esimerkiksi ohjelman testaus. Ohjelma saattaa vaatia kymmeniä käännöskertoja ja virheen korjauksia ennen kuin se on 'valmis'. Ja silti — tai juuri siitä johtuen — ohjelmasta saattaa käytön aikana löytyä lisää virheitä. Jos ohjelma tehtäisiin kerralla kuntoon niin, että käännöskertoja tarvittaisiin vain yksi, ohjelmointi- ja testausvaiheen tuottavuus nousisi moninkertaiseksi.

Ohjelmointivirheet näemme selkeästi, mutta muut virheet voivat olla piilevämpiä. Kun teemme ohjelmiston 'jatkokehitystä', esimerkiksi lisäämme vaikkapa uuden toiminnon, kyseessä saattaa olla määrittelyvirheen korjaaminen. Kun 'viritämme' ohjelmistoa esimerkiksi nopeammaksi, saatammekin korjata suunnitteluvirhettä. Määrittely- ja suunnitteluvirheet ovat kuitenkin työn tuottavuuden kannalta kaikkein ongelmallisimpia. Niiden korjaaminen on kymmeniä, jopa satoja kertoja kalliimpaa kuin ohjelmointivirheiden korjaaminen.

Pahinta virheissä on, että olemme niin tottuneita niihin, että pidämme niitä hyväksyt-

tävinä. 'Jokainenhan tietää, että on mahdollista tehdä virheetöntä ohjelmaa.' Tästä on tullut ohjelmistoihmisten luonnonlaki.

Virheiden tekeminen ei kuitenkaan ole luonnonlaki. Ei hammaslääkärinkään tarvitse porata kymmeniä reikiä ennen kuin osuu oikeaan hampaaseen. Ei kirurginkaan tarvitse tehdä kymmeniä viiltoja ennen kuin saa leikkaushaavan oikeaan kohtaan. Eikä palkanlaskijankaan tarvitse tehdä kymmeniä korjauksia ennen kuin saa maksettua työntekijän palkan oikean suuruisena. Maailma on täynnä asioita, jotka voidaan tehdä virheettömästi.

Kyse on ennen kaikkea asenteista ja virheiden henkilökohtaisesta vaikutuksesta niiden tekijään. Jos tekisimme liikenteessä yhtä paljon virheitä kuin ohjelmia tehdessämme, harvat meistä olisivat terveitä tai edes hengissä. Useimmat meistä ovat kuitenkin sen verran kiinnostuneita terveydestään ja hengestään, että pyrimme välttämään liikennevirheitä.

Näin meidän tulisi suhtautua virheisiin myös työssämme. Jokainen virhe on kuin puukonisku suoraan sydämeemme tai vähintäänkin ikävä tahra ammattitaitomme kiiltävään kilpeen. Meidän olisi pikku hiljaa alettava asennoitua niin, että ohjelmiakin voi tehdä kerralla kuntoon.

Kannattaa muistaa myös se, että mitä enemmän teemme virheitä ohjelmantuotantoprosessin aikana, sitä enemmän niitä jää 'valmiiseen' ohjelmaan ja sitä tyytymättömämpi ohjelman käyttäjä on. Kuitenkin meidän ohjelmistoammattilaisten ainoa tehtävä on palvella asiakastamme, käyttäjää — tuottaa hänelle hyötyjä, ei haittoja.

Jos siis haluamme merkittävästi kohottaa työme laatua ja tuottavuutta, meidän ei tarvitse tehdä yhtään enempää. Päinvastoin, meidän pitää tehdä vähemmän, vähemmän virheitä.

Softari

Nokian ohjelmistolehti

Päätoimittaja:
Hannu Markus
puh. (90) 567 2982

Toimitussihteeri:
Outi Kuusela
puh. (90) 567 2017

Piirroksat:
Taina Sonninen

Osoitemuutokset:
Jaana Lamberg
puh. (90) 567 2980

Toimituksen osoite:
Nokia Informaatiojärjestelmät
Softari-lehti
Kutomotie 16-18
PL 780
00101 HELSINKI

Painosmäärä: 10 000
Ladonta: Tamprint
Kirjapaino: Vientipaino Oy

Julkaisija:
Nokia Informaatiojärjestelmät

ISSN 0782-8209



Kansikuva:

Tietojärjestelmien rakentaminen on toisinaan kuin purjehtimista läpi myrskyjen. Mutta kun tukena on alansa osaava toimittaja, myrskyt ovat aina voitettavissa. Se nähtiin tämänvuotisessa maailmanympäri-purjehduksessakin, jossa voittoon purjehti Honeywell Bullin, Nokian yhteistyökumppanin, tukemana L'Esprit d'Equipe -vene (kuva kannessa).

NOKIA

Laatupalkinto Softplanille



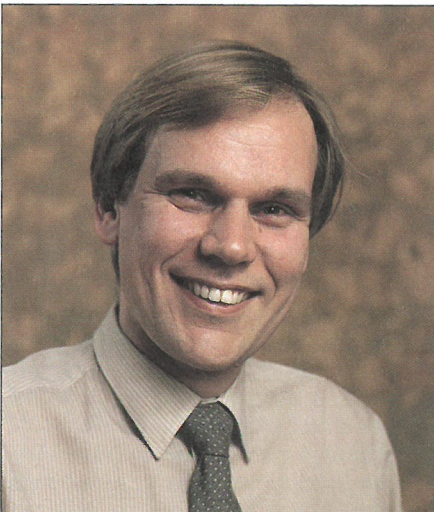
Suomen Laatu yhdistys jakoi 15.4.1986 vuosittaisen laatupalkintonsa. 20-vuotisen toimintansa kunniaksi yhdistys myönsi tänä vuonna kolme palkintoa. Yksi palkinnon saajista oli Oy Softplan Ab, Nokia-yhtymään kuuluva ohjelmistotalan palveluyritys.

Vuodesta 1976 jaettu laatupalkinto myönnetään yrityksille, yhteisöille ja laitoksille tai näiden itsenäisille osille, jotka ovat toteuttaneet ansiokasta laatutoimintaa ja edistäneet laatuajattelua myös oman toimialueensa ulkopuolella. Laatupalkintoa ei koskaan myönnetä tuotteelle.

Ohjelmistotyön laatua kohottavien menetelmien, työtapojen ja apuvälineiden kehittämiseen on Softplanissa jo vuosia kiinnitetty erityistä huomiota. Siitä osoituksena Suomen Laatu yhdistys myönsi jo viime vuonna Softplanin laatu päällikölle Pasi Kanteliselä erityisen tunnustuspalkinnon.

Nyt Softplanille myönnettiin laatupalkinto tunnustukseksi mm. ihmislähtöisiin menetelmiin perustuvasta laatu toiminnasta sekä laatu kysymysten yleisestä tunnetuksi tekemisestä. Softplanin toimitusjohtaja Hannu Lehesaari muistutti palkinnonjakotilaisuudessa, että asiakkaiden yhä kovenevat vaatimukset edellyttävät yritykseltä jatkuvaa ja entistä suurempaa panosta laadun hyväksi.

Ja laatuun on Nokialla muuallakin panostettu. Saihan Nokia Elektroniikka vastaavan laatupalkinnon kaksi vuotta sitten tunnustukseksi laatu tekniikan ja laatuun liittyvien asioiden tunnetuksi tekemisestä.



Viereisen sivun pääkirjoituksen on kirjoittanut fil.maist. Pasi Kantelinen (kuvasa). Hän toimii laatu päällikkönä Softplanissa ja on työskennellyt ansiokkaasti laatu järjestelmien kehittämiseksi ohjelmistotuotannossa. Hänen kehittämänsä menetelmät ohjelmistojen laadun parantamiseksi ovat herättäneet laajaa kiinnostusta.

Kohti ohjelmistojen integraatiota

Useimmat ohjelmistot ovat edelleen yhteen tehtävään, esimerkiksi tekstinkäsittelyyn, suunniteltuja. Ohjelmien erilaiset käyttäjäliitynnät ja tietojen hankala siirrettävyys ohjelmien välillä aiheuttavat harmaita hiuksia käyttäjille.

Käyttäjillä on kuitenkin oikeus vaatia toimintojen saatavuuden niveltyä toisiinsa. Siihen on pyritty viime vuosina ohjelmien integroinnilla.

Ohjelmistojen integroinnilla tarkoitetaan niiden yhteensovittamista siten, että eri ohjelmat liittyvät toisiinsa joustavasti ja että niiden käyttäjäliityntä on yhdenmukainen. Täten integroinnilla pyritään luomaan käyttäjälle yhtenäisen työskentely-ympäristö.

Ohjelmistojen integroitumisen paras tarkastelu-ympäristö on yleismikrojen yhden tehtävän käyttöjärjestelmä MS-DOS. MS-DOSin vakiintuminen mikrotietokoneiden ja vähitellen myös verkkojärjestelmien työasemien sovellysympäristöksi on edistänyt integroitumista.

Yhden toiminnon sovellukset

Ensimmäiset mikrotietokoneiden sovellysohjelmat eivät juurikaan sisältäneet integraatioita. Kukaan ohjelma soveltui vain yhteen tietojenkäsittelytehtävään.

Eri syistä johtuen kävi vielä niin, että tärkeimmille sovellysalueille nousi markkinaykkösiksi eri toimittajien tuotteet. WordStar hallitsi tekstinkäsittelyä, Multiplan taulukkolaskentaa, ja kortistointiin yleistyi dBase II. Näiden käyttäjien kannalta yksitoimisovellusten kohdalla oli - ja on osin edelleen - siirtymisen ohjelmasta toiseen tehtävä levykettä vaihtamalla tai ainakin ladattava seuraava ohjelma levyltä. Ja monasti tiedot on syötettävä uudelleen.

Tilanteeseen on vähitellen kehitetty erilaisia integraatioon pyrkiviä ratkaisus-

ja. Näistä mainittakoon tiedostomuunnosohjelmat eri sovellusten välillä sekä tulostuslaajennukset. Esimerkiksi Nokian valmistamalla Convix-ohjelmalla voidaan WordStarilla tehtyt tekstit siirtää suoraan käsiteltäväksi Edix/wordixillä ja päinvastoin. Niin ikään Nokia on kehittänyt Multigraph-ohjelman, jonka avulla Multiplanin tietoa-ineisto voidaan esittää graafisessa muodossa. Multigraph on laajennettu sittemmin myös Kmies-tietokannan tulostukseen. Samoin Nokia tarjoaa Infographin vastaavasti PC-Infon käyttäjille.

Yksitoimisovellusten myötä yleiseen käyttöön on tullut muutamia toisistaan poikkeavia tiedostorakenteita, jotka ovat keskenään yhteensopimattomia. Useat viime aikoina markkinoille tulleet ohjelmatuotteet onkin suunniteltu siten, että ne pystyvät käsittelemään eri tiedostoformaatteja, esimerkiksi ASCII-, DIF-, SYLK- tai dBase II-tiedostoja. Tietojen siirto ohjelmasta toiseen on siten yhä useammassa tapauksessa mahdollista. Voidaan puhua tiedostokäsittelyn integroitumisesta.

Toisenlainen tiedon integroituminen on kyseessä kun joukko yksitoimisovellyksia toteutetaan yhteiselle tiedonhallintajärjestelmälle. Sovellukset ovat samalta toimittajalta ja keskenään yhdenmukaisia käyttäjä. Puhutaan tuoteperheistä. Esimerkkinä tuoteperheestä mainittakoon PFS-File, PFS-Write ja PFS-Calc -ohjelmaperhe. Käyttäjä voi hankkia vain tarvitsemiaan toimintoja vastaavat ohjelmat tuoteperheestä.

Monen toiminnon sovellukset

Monitoimisovellukset, joista ensimmäinen merkittävä oli Lotus 1-2-3, ovat pidemmälle vietyä integraatiota. Teknisesti kyseessä on usean eri toiminnon toteuttaminen yhteen ohjelmaan.

Monitoimisovellus rakentuu yleensä yhden päätoiminnon varaan. Lotus 1-2-3:n päätoiminto on taulukkolaskenta ja sen ohella siinä on grafiikka ja tiedonhallinta. Toisenlaista toimintojen integraatiota edustaa Kmies, jossa päätoimintona voidaan pitää tiedonhallintaa ja jossa lisäksi on taulukkolaskenta, kyselyt, raporttien tulostaminen ja ohjelmointimahdollisuudet.

Sekä Lotus 1-2-3 että Kmies ovat menestyneet hyvin ja niistä on tullut markkinoille uudet monipuolisemmat versiot. Myös Symphony on menestynyt hyvin eräillä markkina-alueilla. Symphonyssa on vastaavat toiminnot kuin Lotus 1-2-3:ssa ja lisänä tekstinkäsittely ja asynkroninen tietoliikenne.

Monitoimisovellusten etuina ovat eri toimintojen varsin hyvä yhtenäisyys ja saman tietoa-ineiston käyttö eri toimintoihin ilman erillisyttöä. Monitoimiso-vellyus on myös hinnaltaan edullisempi kuin vastaavat toiminnot erillisinä sovellyksina.

Monitoimisovellusten kaupallinen menestys on epäilemättä johtunut myös siitä, että niillä on — vaikka ovatkin kiinteitä paketteja — joustavuutta eri käyttötilanteisiin. Laskentalustahan on oiva pohja mitä erilaisimpiin analyysihin eri toimialoilla. Lisäksi ohjelmoitavuus mahdollistaa toimintojen laajentamisen käyttäjän tarpeiden mukaisesti.

Monitoimisovellusten menestys on riippunut ja riippuu paljolti myös siitä, miten hyvin ne välttävät keskeisen ongelman — liian monen toiminnon haittamisen yhteen ohjelmaan. Muistia ja levytilaa kuluu helposti liikaa ja toimintojen nopeus ei tyydytä käyttäjää. Liian suuri toimintojen määrä voi näkyä käytön monimutkaisuutena.



Onkin nähtävissä, että monitoimiso-
vellusten toimintojen lukumäärää ei
tulla nostamaan. Parhaimmillaan kol-
men tai neljän eri toiminnon oikea
yhteennivominen on sopuustainen
ratkaisu. Tästä kehityssuunnasta esi-
merkkeinä mainittakoon Reflex sekä
Excel, jonka MS-DOS -version huhu-
taan olevan kehitteillä.

Monitoimiso-
vellustenkin kohdalla
esiintyy pyrkimyksiä kasvattaa integroi-
tumisastetta. Esimerkiksi eräät monitoi-
misovellukset pystyvät käsittelemään
usean eri ohjelman tuottamia tiedostoja
ja muodostamaan tietoaineistosta mo-
nenlaisia tulosteita. Tyypillisiä tällaisia
tuotteita ovat Microsoft Rbase sekä
Reflex.

Reflex on tarkoitettu tiedon analy-
sointiin ja esittämiseen. Sen sijaan re-
laatiotietokanta Rbase sisältää mm. vä-
lineet tietokannan luontiin, ylläpitoon
ja kyselyihin, vuorovaikutteisen sovel-
luskehittimen sekä ohjelmointimahdol-
lisuuden. Muun muassa Multiplanin,
dBase II:n, Lotus 1-2-3:n ja Sympho-
nyn tietoaineisto saadaan Rbasen käyt-
töön samoin kuin ASCII- ja DIF-tiedos-
tot.

Eräänä monitoimiohjelmien erikois-
tyyppinä on syytä mainita ohjelmat,
jotka määritellään käyttöjärjestelmään
ja joita voidaan käyttää rinnakkain var-
sinaisen sovellusohjelman kanssa. Täl-
lainen ohjelma sijoittuu keskusmuistiin
samanaikaisesti käyttäjän varsinaisen
sovellusohjelman kanssa.

Esimerkiksi Sidekick on tämäntyyppi-
nen ratkaisu. Se tarjoaa käyttäjälle
mm. kalenterin, laskimen ja muistileh-
tiön. Muutamalla näppäinpainalluksella
käyttäjä pääsee varsinaisesta sovelluk-
sesta Sidekickiin ja samoin takaisin so-
vellukseen. Kyseessä on ratkaisu, jon-
ka laajennettavuus ja toimivuus seuraa-
vien ohjelmaversioiden kanssa on käyt-
täjän omalla vastuulla. Sidekickin käyt-
töönnotto edellyttääkin käyttäjältä hyvää
käyttöjärjestelmän tuntemusta.

Vaikka siirtyminen toiminnosta toi-
seen on joustavaa monitoimiso-
velluk-
sissa, erilliset ohjelmat voivat olla to-
teutukseltaan monipuolisempia. Niinpä
eräät yksitoimiso-
vellukset ovat menes-
tyneet ja menestynevät jatkossakin hy-
vin. Näin on käynyt esimerkiksi Word-
Perfectille.

Kohti avointa ohjelmaympäristöä

Jatkossa integroituminen kehittyi to-
dennäköisimmin siten, että sovellusoh-
jelman sijasta käyttöjärjestelmä hoitaa
keskeisiä integroitumisen edellytyksiä.
Näitä ovat mm. tietoaineiston siirto
ohjelmien välillä, näytön hallinta, toi-
mintojen valinta ja mahdollisuus ajaa
useampaa ohjelmaa samanaikaisesti.
Tuloksena on avoin ympäristö erillisille
ohjelmille.

Avoin ympäristö toteutuu laajennuk-
sena MS-DOSin päälle. Samalla saa-
vutetaan se tärkeä etu, että uusi sovel-
lusympäristö toimii jo olemassa olevien
sovellusten kanssa. Esimerkiksi Micro-
soft Windows on tällainen avoin ohjel-
maympäristö.

Avoimeen ohjelmaympäristöön siir-
tyminen helpottaa mikrotietokoneen
käyttöä ja nostaa työskentelyn tehok-
kuutta. Mikrot saadaan tuottavampaan
käyttöön. Näin ohjelmien integroitumi-
nen kehittyi oikeaan suuntaan.

Windows

Microsoft Windows sisältää kosolti edis-
tyksellisiä piirteitä. Se on MS-DOSin
laajennus, jonka avulla ohjelmien in-
tegroitumiskehitys astuu ison askeleen
eteenpäin.

Windowsilla voi työskennellä käyt-
tämällä samanaikaisesti useaa ohjelmaa.
Näytön käyttö laajenee, sillä näytön eri
osiin eli ikkunoihin saadaan yhtäaika-
näkyviin eri ohjelmien tietoa. Uusi näy-
tölle tuotu ikkuna jakaa näyttöruudun
tilan edellisten kanssa.

Käyttäjän on helppo vaihtaa toimin-
nosta toiseen. Windowsia voi käyttää
joko näppäimistöltä tai hiiren avulla;
hiiri helpottaa käyttöä.

Ohjelma käynnistetään valitsemalla
se luettelosta. Käynnistämiseen riittää
ohjelman nimen osoittaminen. Käyn-
nissä oleva ohjelma voidaan sulkea ja
samalla jättää se aktiiviseksi, jolloin siitä
jää näkyviin ns. ikoni näytön alareu-
naan varatulle ikonialueelle. Aktiivinen
ohjelma voidaan käynnistää siirtämällä
ikoni takaisin näytölle. Tällöin ohjelma
saa itselleen ikkunan ikonin osoittamal-
ta alueelta. Jokainen käynnissä oleva
ohjelma näkyy joko näytöllä tai ikonina
näytön alalaidassa.

Windowsin komennot ovat näkyvis-
sä näytön ylälaidassa. Komento anne-
taan sitä osoittamalla ja valitsemalla
esiintulevasta vetovalikosta haluttu
vaihtoehto. Eräiden kommentojen anta-
miseen käytetään lisäksi erillistä ko-
mentoikkunaa (dialogbox).

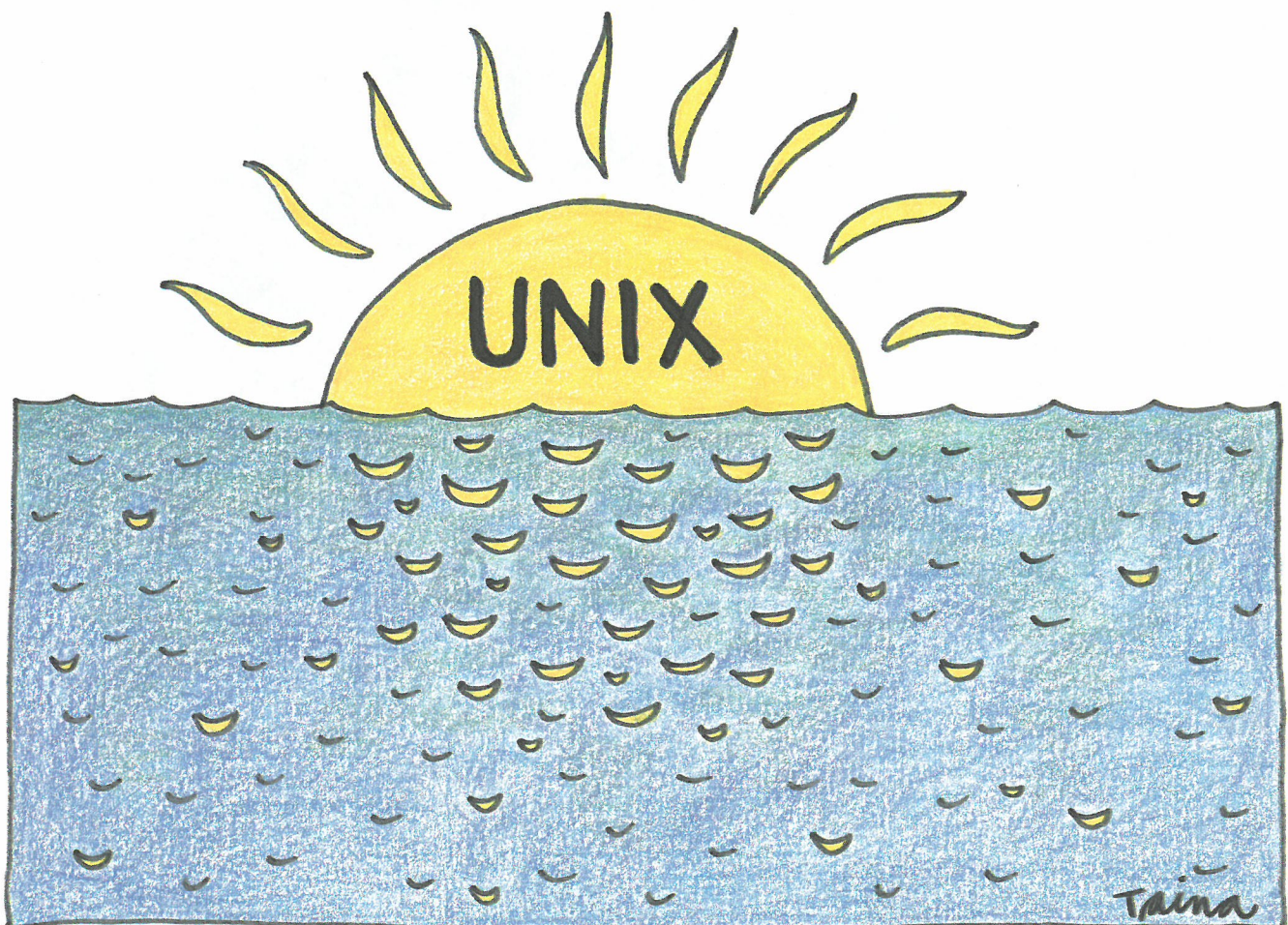
Käyttäjän haluamat toiminnot toteu-
tetaan usean ohjelman avulla. Näytöllä
ikkunassa meneillään oleva toiminto
on zoomattavissa koko näytön kokoi-
seksi. Niin ikään tietoa voidaan siirtää
eri toimintoja suorittavien ohjelmien
välillä. Windows sisältää valmiina eräitä
perustoimintoja, mm. kalenterin, kel-
lon, muistion, laskimen ja kortiston.

Windowsin edut tulevat parhaiten
esiin levymuistilla ja korkealuokkaisella
näytöllä varustetussa suorituskykyises-
sä mikrossa. □

Moni uskoo Unixiin

Tuskin millään muulla käyttöjärjestelmällä on niin monta jopa fanaattista kannattajaa kuin Unixilla. Toisaalta Unixilla on aina ollut paljon myös kritisoivia vastustajia. Unix ei jätä ketään välinpitämättömäksi.

Puolesta tai vastaan, mutta tosiasia lienee, että Unixista on muodostumassa supermikrojen käyttöjärjestelmä-standardi. Unix on monen käyttäjän järjestelmä, jonka valtteja ovat mm. siirrettävyys ja runsas työkaluohjelmien valikoima.



Käyttöjärjestelmä on tietokoneen ohjelma, joka huolehtii koneen resurssien hallinnasta. Käyttäjä 'näkee' koneensa käyttöjärjestelmän kautta. Sen tehtävänä on tarjota yhtenäinen ja helppo-käyttöinen liitäntä laitteiston, käyttäjän sekä hänen ohjelmiansa välillä. Hyvä käyttöjärjestelmä on sellainen, jossa liitäntä on samanlainen laitteiston kokoonpanosta riippumatta.

Ripeästi kehittyvä tekniikka on ajanut monen käyttöjärjestelmän ohi. Käyttöjärjestelmät ovat yleensä perustuneet liiaksi tiettyyn tietokonearkkitehtuuriin. Niitä ei ole voitu siirtää uusille tehokkaammille suorittimille.

Unix onkin ainoa varteenotettava ehdokas laiteriippumattomaksi käyttöjärjestelmäksi.

Unixin synty ja kehitys

Unix ei ole mikään uusi keksintö. Se syntyi jo 1970-luvun alussa, kun amerikkalaisen tietoliikennejätin AT&T:n Bellin laboratorioioden ohjelmoijat Ken Thompson ja Dennis Ritchie työskentelivät erään suuren ohjelmointiprojektin kimpussa. He alkoivat kyllästyä surkeaan ohjelmankehitysympäristöön ja kehittivät oman yksinkertaisen tiedostojärjestelmän sekä useita käteviä ohjelmankehityksen työkaluja. Näistä kehittyi vähitellen kokonainen käyttöjärjestelmä. Projektiin liittynyt Brian Kernighan keksi järjestelmälle nimen Unix.

Unix kehittyi nopeasti. Siihen lisättiin 'putket' eli mekanismi prosessien väliselle kommunikoinnille. Laiteriippumattomuutta paranneltiin, koska järjestelmä haluttiin siirtää muille koneille. Tähän tarkoitukseen Thompson ja Ritchie kehittivät uuden systeemiohjelmointiin soveltuvan C-ohjelmointikielen. Unixin uusi versio kirjoitettiin tällä korkean tason kielellä vuonna 1973. Aikaisemmin vastaaviin tehtäviin oli käytetty konekieltä.

Unixiin ihaustuivat monet myös Bellin laboratorioioden ulkopuolella. Järjestelmä levisi nopeasti erityisesti akateemisessa maailmassa. Unixia onkin jaettu yliopistojen käyttöön nimellistä maksua vastaan.

Unixin pääpiirteitä

Unix on avoin ja joustava käyttöjärjestelmä. Siihen on helppo lisätä uusia piirteitä. Unixin keksijät tekivät aikanaan monia näppäriä ratkaisuja, joiden käyttökelpoisuus on säilynyt aina tähän päivään saakka.

Unixin keskeisimpiä piirteitä ovat mm. tiedostojen ja laitteiden yhdenmukainen käsittely, runsas työkaluohjelmien valikoima, siirrettävyys, tiedostojen yhdenmukainen rakenne sekä prosessikäsite.

Unixissa syöttö ja tulostus kohdistuu aina tiedostoihin. Erilaisia oheislaitteita kuten päätteitä, kirjoittimia, telex-päätteitä jne. käsitellään tiedostoina ja niihin voidaan kohdistaa samoja käskyjä kuin muihinkin tiedostoihin.

Myös tiedostojärjestelmän hakemistot ovat muodollisesti tiedostoja.

Unix-ympäristöön on tehty pieniä ohjelmia, työkaluja, jotka suorittavat vain yhden tehtävän, mutta jotka tekevät sen hyvin.

Työkalut suunnitellaan siten, että ne on helppo yhdistää toisiin työkaluihin Unixin putkimekanismin avulla. Näjn voidaan nopeasti rakentaa uusia komentoja. Koska yksittäiset komponentit ovat hyvin testattuja, on niistä koottu uusi komentokin helppo saada toimimaan.

Unix on kirjoitettu korkean tason kielellä eli C:llä. Käyttöjärjestelmän istuttaminen uuteen koneeseen vaatii yleensä vain valmiin C-kielisen koodin uudelleenkäntämisen sekä muutaman konekielisen ohjelmanosan kirjoittamisen.

Unixin ydin on hyvin pitkälle riisuttu kaikista hienouksista. Sellaiset piirteet, joita ei välttämättä ole pakko sijoittaa kiinteästi käyttöjärjestelmän sisään, on jätetty sen ulkopuolelle. Toisaalta Unixiin kuuluu standardinomaisesti kymmeniä erillisiä työkaluja, joilla puuttuvat piirteet saadaan käyttöön.

Kaikilla tiedostoilla on Unixissa samanlainen sisäinen rakenne. Ohjelmat näkevät tiedoston aina merkkien virtana, jonka ne voivat tulkita haluamallaan tavalla (esimerkiksi henkilörekisterin tietueiksi). Hyöty saavutetaan sillä,

että eri ohjelmat voivat helposti käyttää ja muokata samoja tiedostoja.

Prosessi on eräs Unixin keskeisimpiä käsitteitä. Sillä tarkoitetaan koneen muistissa suorituksessa olevaa tai suoritukseen pääsyä odottavaa ohjelmaa. Prosessit voivat kommunikoida keskenään tai luoda uusia prosesseja. Prosessien avulla on Unixissa luonnollisella tavalla toteutettu esimerkiksi usean ohjelman samanaikainen ajaminen.

Hierarkkinen tiedostojärjestelmä

Unixin tiedostojärjestelmä muistuttaa paljon MS-DOS 2.0 -käyttöjärjestelmän tiedostojärjestelmää. Sen hakemistorakenne on hierarkkinen eli puumainen.

Koska Unix on monen käyttäjän järjestelmä, jokaisella käyttäjällä on oma kotihakemistonsa. Omat tiedostot voidaan suojata muilta käyttäjiltä.

Monipuolinen komentotulkki

Käyttäjä kommunikoi käyttöjärjestelmän kanssa ns. komentotulkin eli shellin kautta. Se on ohjelma, joka lukee päätteeltä käyttäjän syöttämiä komentoja, tulkitsee ne ja käynnistää sitten vaadittavat prosessit.

Unixin komentotulkki on erittäin monipuolinen. Sen käyttö on vaikeahko oppia, mutta osaavan käsissä se on erittäin tehokas väline tietokoneen hyväksikäyttöön. Pienen opettelun jälkeen käyttäjä pystyy ilman ohjelmointikieliä rakentamaan melko näyttäviä sovelluksia vain komentotulkkia ja työkaluohjelmia käyttämällä.

Runsaasti valmiita työkaluja

Unix-järjestelmän mukana seuraa suuri määrä käteviä työkaluohjelmia mitä erilaisimpiin tarpeisiin. Monet niistä ovat sellaisia, joista muissa järjestelmissä saisi maksaa pitkän pennin.

Käyttöalue laajenee

Unix tunnetaan lähinnä ohjelmankehitysympäristönä. Sitä on käytetty yliopistoissa pääasiassa koulutus- ja tutkimustarkoituksiin.

Tänä päivänä Unixin käyttöalue on jo varsin laaja. Unixin käyttö graafisten työasemien käyttöjärjestelmänä on kovaa vauhtia yleistymässä. Niin ikään Unix soveltuu runsaan työkaluohjelmistonsa ansiosta tekniseen laskentaan.

Unix soveltuu myös kaupallishallinnollisten sovelluksien käyttöjärjestelmäksi. Markkinoille onkin jatkuvasti tulossa valmiita kaupallishallinnon ohjelmapaketteja.

Tulevaisuus näyttää valoisalta

Miksi Unixiin panostetaan toden teolla vasta nyt, 15 vuotta sen keksimisen jälkeen?

Eräs syy Unixin nihkeään tulemiseen (verrattuna esimerkiksi mikroissa käytettävän MS-DOSin suosioon) lienee se, että Unix on joutunut kilpailemaan olemassa olevien järjestelmien kanssa. Laittevalmistajat eivät tietenkään ole halunneet luopua omista järjestelmistään.

Tänä päivänä kuitenkin lähes kaikki suuret laitevalmistajat tarjoavat Unixia käyttöjärjestelmäksi johonkin koneeseensa. Laitteistoja löytyy kaikkia ko-koja, mikroista suurtietokoneisiin asti.

Unixia on myös vaivannut erilaisten versioiden kirjo, mutta tilanne on lopulta selkiytymässä. Tämän päivän tärkein standardi on System V, ja lähes kaikki muut rinnakkaiset versiot ovat kehittyneet sen suuntaan.

Hankalaksi moitittu käyttäjäliityntä voidaan korvata valikko- tai ikkunointiohjelmilla, jolloin Unixin oma komentotulkki piilotetaan kauniin kuoren alle.

Unix elää uusien kehitysvirtausten mukana. Joustavuutensa ansiosta se soveltuu hyvin esimerkiksi hajautettujen järjestelmien käyttöjärjestelmäksi. Monipuoliset tietoliikennepalvelut ovat jo kauan olleet Unix-järjestelmän osana.

Huonosti toteutetut toiminnot ovat vuosien mittaan vähitellen karsiutuneet pois kriittisten käyttäjien painostuksen seurauksena. Unix on lopulta päässyt 'miehen ikään' ja sen kaupallinen läpimurto näyttää nyt Suomessakin olevan käsillä.

Yhä useammat käyttäjät liittyvät Unixin kannattajiin huomattuaan sen tärkeimmät edut: tehokkuuden ja joustavuuden. □

Useimmat Unixin työkaluista on suunniteltu ohjelmoiden käyttöön. Niinpä Unixia pidetään erityisesti hyvänä ohjelmankehitysympäristönä.

Työkaluohjelmien eli komentojen nimet ovat tyypillisesti lyhyitä, eivätkä kovinkaan selvästi kerro mitä niiden taakse kätkeytyy. Seuraavassa on lueteltu eräitä Unixin tärkeimpiä komentoja.

adb	Debuggeri
awk	Yleiskäyttöinen raporttogene-raattori
cat	Yhdistelee tiedostoja
cc	C-kielen kääntäjä
cmp	Tiedostojen sisältöjen vertailu
cp	Tiedostojen kopiointi
crypt	Tiedoston koodaaminen sala-kirjoitukseksi
diff	Kahden tiedoston eroavuuk-sien tulostus
find	Tiedostojen etsiminen nimen, muutosajan, koon, tms. pe-rusteella
grep	Tiettyjen rivien poimiminen tiedostosta
kill	Prosessin tuhoaminen
lpr	Tulostus kirjoittimelle
ls	Hakemiston sisällön listaus
mail	Elektroninen posti
man	Käsikirjan sisällön tulostus päätteelle
nroff	Tekstin muotoilu kirjoittimelle
passwd	Salasanan muuttaminen
ps	Ajossa olevien prosessien tu-lostus
sort	Tiedoston sisällön lajittelu
spell	Englanninkielisen oikeinkirjoi-tuksen tarkistus
troff	Tekstin muotoilu ladontalait-teelle
vi	Näyttömuokkain
wc	Tiedoston rivien, sanojen, ja merkkien lukumäärän tulos-tus
write	Keskustelu toisen käyttäjän kanssa

Nokia täyd Honeywell-Unix-konee

Nokian ja Honeywellin yhteistyö lujittuu. Näkyvänä merkinä tästä on Nokian tuotevalikoiman täydentäminen Italian Honeywellin valmistamalla Unix-koneella. Laite on Unix System V tai Xenix -käyttöjärjestelmällä varustettu supermikrotietokone. Ja sitä markkinoidaan tuotenimellä Nokia UX-1.

Nokialla on tuotteen markkinointioikeudet paitsi Suomessa myös Ruotsissa ja Norjassa. Laitetta markkinoivat Nokian lisäksi hyvän Unix-tietämyksen omaavat jälleenmyyjät.

Nokia UX-1 on monen käyttäjän moniajokone. Tekniikaltaan se edustaa uusinta mikroelektroniikkaa ja käyttöjärjestelmältään yhä kasvavan suosion saavaa sovellusympäristöä.

Laitteiston arkkitehtuuri

Laitteisto on toteutettu 16/32 -bittisillä Motorolan 68000 mikrosuorittimilla.

Laitteisto on arkkitehtuuriltaan 2-suoritinjärjestelmä. Erillisten pää- ja I/O-suorittimen avulla saadaan lisää suorituskyyä. Pääsuoritin (Motorola 68010) huolehtii käyttöjärjestelmän sisäisistä toiminnoista sekä ohjelmien suorituksesta. Sen sijaan I/O -toiminnot hoidetaan erillisellä I/O -suorittimella (Motorola 68000).

Koneessa on myös erillinen aritmeettikkasuoritin. Se lisää edelleen laitteiston suorituskyyä.

Keskusmuistia laitteistossa on peruskokoonpanossaan yksi megatavu, mutta sitä voidaan laajentaa aina 3,5 megatavuun asti.

entää tuotevalikoimaa lla



Laitteistoon kuuluu vakiona 60 megatavun umpilevy (Winchester). Massamuistikapasiteettia voidaan laajentaa lisäämällä toinen umpilevyasema, jolloin päästään 120 megatavuun.

Levyasemista saadaan helposti varmuuskopio Streamer-nauha-asemalle, jonka kapasiteetti on myös 60 megatavua. Nauhayksikkö on saatavana optiona.

Ohjelmistojen siirtoa varten on laitteistossa oma levykeasema, jonka kapasiteetti on 720 kilotavua.

Työasemat ja kirjoittimet

Nokia UX-1 -järjestelmässä on neljä tai kahdeksan asynkronista sarjaporttia, joista yksi tai kaksi voi olla vaihtoehtoisesti synkronisia. Porttien kautta kone pystyy kommunikoimaan muiden tietokoneiden kanssa.

Niin ikään työasemat liitetään sarjaporttien kautta. Niinpä laitteisto voi olla peruskokoonpanossaan jopa kahdeksan käyttäjän järjestelmä. Työasemiksi sopivat Nokian VDU- tai muut asynkroniset näyttöpäätteet. Myös mikrotietokoneet asynkronisten pääte-emulaattoreiden avulla käyvät laitteiston työasemiksi.

Koneessa on valmiina myös kirjoitinliitäntä, johon on liitettävissä centronics-liitännällä varustettu kirjoitin. Sen sijaan sarjaliitännällä varustetut kirjoittimet on liitettävä sarjaporttien kautta kuten työasematkin.

Laitetekoonpanoa voidaan laajentaa erillisellä Multibus-väylällä ja siihen sopivilla laajennuskorteilla. Esimerkkinä Multibus-korteista mainittakoon HDLC/SDLC -ohjaimet sekä Ethernet/IEEE 802.3, jolla UX-1 voidaan liittää Bridge-verkkoon. Multibus-korteilla voidaan kasvattaa myös esimerkiksi työasemien määrää.

Ohjelmisto

Käyttöjärjestelmävaihtoehtoja ovat joko Unix System V tai Xenix -käyttöjärjestelmä. Molemmat käyttöjärjestelmät ovat yleistyneet mikroteknologian kehityksen myötä. Mikrojen parantunut suorituskky mahdollistaa monen käyttäjän moniajoympäristön. Eli useampi käyttäjä voi samanaikaisesti suorittaa eri tehtäviä samalla koneella.

UX-1 -järjestelmään Nokia tarjoaa valmiita työkaluohjelmia, kuten kieliä ja tietokantoja. Lisäksi Nokian Laskentakeskuksella on oma taloushallinnon sovellusohjelmisto UX-1 -koneelle.

Unix System V käyttöjärjestelmänä

Amerikkalaisen Unisoftin tekemä Unix System V on yleisin Unix-versio maailmalla. Se on ollut käytössä jo yli kymmenen vuotta erityisesti yliopistomaailmassa. Sitä on käytetty lähinnä tieteellisiin sovelluksiin minikoneilla.

Viime vuosina useimmat laitevalmistajat ovat valinneet Unix System V:n standardikseen. Niipä siihen on saatavissa jo nyt melkoisesti sovelluksia myös toimistokäyttöön.

Unix System V -ympäristöön on Nokialta saatavissa ohjelmointikieliksi RM-Cobol, SVS-Basic, SVS-Pascal, SVS-Fortran 77 sekä C-kieli. Tietokantaohjelmista UX-1:een saa Unify- ja Progress-ohjelmat. Tietoliikenneohjelmista mainittakoon 2780/3780 BSC, 3270 BSC sekä 3270 SNA/SDLC. Lisäksi Nokia tukee Unigraph-grafiikkaohjelman käyttöä.

Vaihtoehtona Xenix

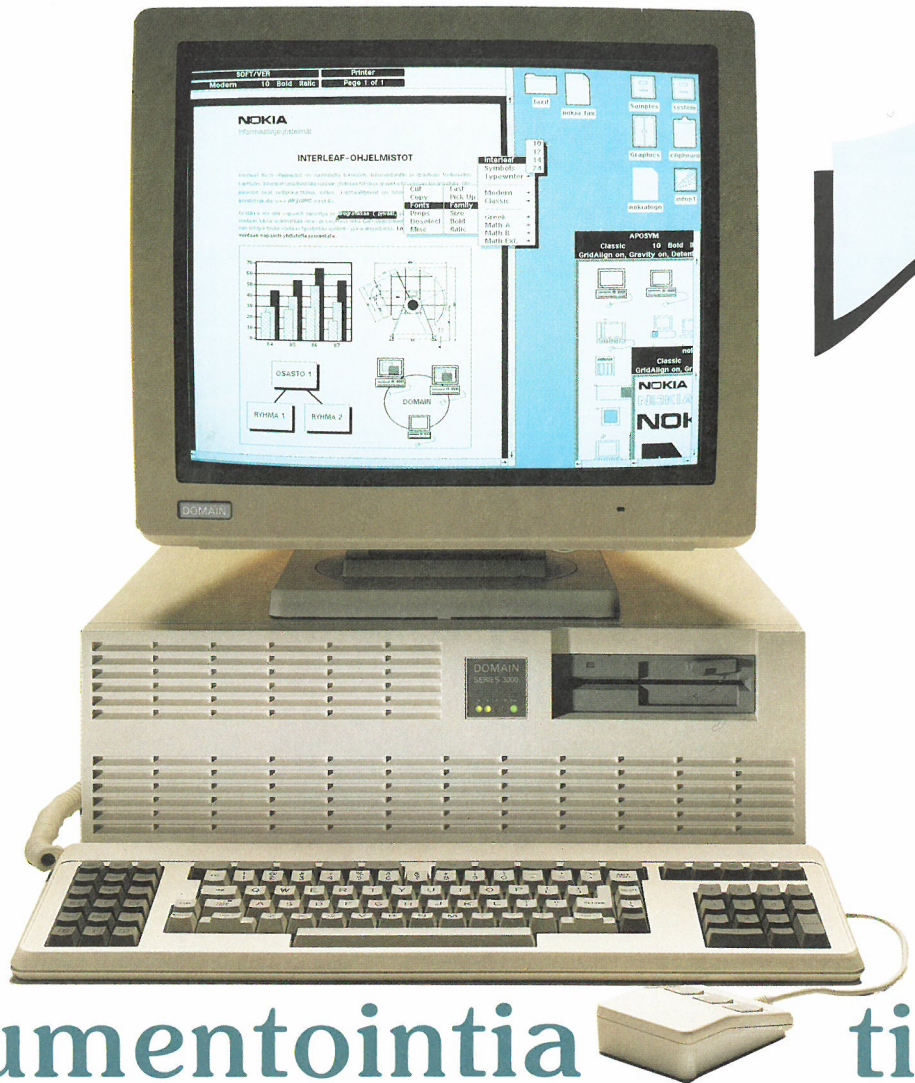
Microsoftin tekemää Xenixiä on käytetty tähän asti pääasiassa kaupallishallinnollisissa sovelluksissa. Suomessa Xenixin käyttö on ollut yleisempää pk-yrityksissä kuin Unixin. Xenix on kylläkin voimakkaasti kehittymässä Unix System V:n suuntaan.

Xenixiin on Nokialta saatavissa RM-Cobol, MS-Basic, HCR-Pascal, ACE-Fortran 77 ja C-ohjelmointikielet, Informix- ja C-ISAM -tietokantaohjelmat, Unigraph-grafiikkaohjelma, sekä BSC 2780/3780 ja BSC 3270 -emulaattorit. Muina työkaluohjelmoina UX-1:een tuetaan Uniplex-tekstinkäsittelyä ja Multiplan-taulukkolaskentaa.

Lisätietoja: Vesa Kousa,
puh. (90) 187 4170.

Korkealaatuisten, tekstiä ja kuvia sisältävien asiakirjojen teko on perinteisin menetelmin työlästä. Toistuvaa leikkaamista, liimaamista ja kopiointia. Jos lopputulos ei tyydytä, koko työ on useimmiten tehtävä uudestaan. Lisäksi työhön kuluu paljon kallista aikaa ja materiaaleja.

Tänä päivänä tietotekniikka tulee avuksi myös asiakirjojen laadinnassa.



Dokumentointia tiet

Näytöllä jo lopullisessa muodossaan

Nokian Interleaf-dokumentointijärjestelmällä pystytään asiakirjat tekemään alusta loppuun tietokoneella. Työaseman kuvaruudulla käyttäjä voi yhdistää tekstin ja grafiikan valmiiksi asiakirjaksi nopeasti ja vaivattomasti.

Nokia markkinoi järjestelmää Suomen lisäksi Ruotsissa, Norjassa ja Tanskassa. Interleaf-ohjelmisto on amerikkalaisen Interleaf Inc:n kehittä-mä.

Interleaf-järjestelmä toimii 32-bittisillä Apollo-työasemakoneilla. Järjestelmä on laajennettavissa joustavasti paikallisverkolla yhdestä aina useaan saatan työasemaan siten, että tiedostot ja ohjelaitteet ovat yhtäläillä kaikkien käyttäjien käytettävissä.

Työaseman 19 tuuman kuvaruudulla voidaan A4-kokoista asiakirjaa käsitellä lopullisessa muodossaan ja oikean kokoisena. Työaseman grafiikanäytön erotuskyky (resoluutio 1280 x 1024) riittää pienienkin yksityiskohtien tarkkaan esittämiseen.

Asiakirjojen laadinnassa on käytettävissä useita kirjasintyyppejä eri kokoisina, kursivina, vahvennettua sekä matemaattisia merkkejä. Ja nämä kaikki nähdään kuvaruudulla jo lopullisessa muodossaan eli samanlaisina kuin ne tulostuvat paperille.

Järjestelmän ikkunointitekniikka mahdollistaa jopa 16 eri asiakirjan samanaikaisen käsittelyn.

Tekstin kirjoittamiseen ja muokkaamiseen on käytettävissä tavanomaiset tekstinkäsittelytoiminnot, kuten tavutus, tekstin tasaus, sekä tekstilohkojen kopiointi ja siirto eri asiakirjojen välillä.

Tekstiin grafiikkaa

Tekstiin on mahdollista yhdistää grafiikkaa haluttuihin kohtiin. Grafiikka voi olla vapaasti piirrettyä, kuten organisaatio- ja lohkokaavioita, symboleja tai vaikkapa lay-out -piirustuksia.

Interleaf-ohjelmistolla käyttäjä pystyy esittämään haluamansa tietoaineiston businessgrafiikan pylväsdiagrammeina, piirakoina tai käyrinä.

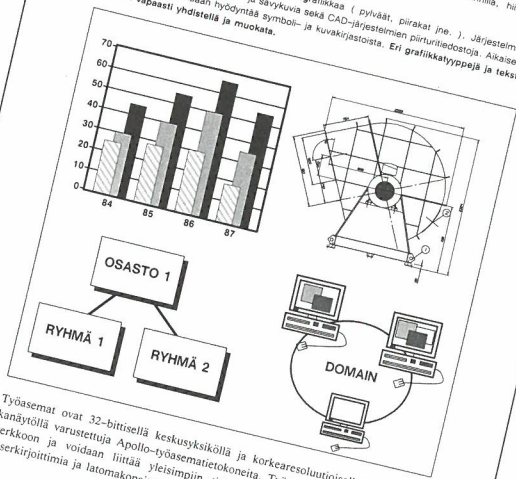
Asiakirjoihin voidaan siirtää valmiita CAD-piirustuksia yleisimmistä CAD-järjestelmistä. CAD-piirustuksia voidaan suurentaa, pienentää, peilata tai kiertää haluttuun kulmaan. Esimerkiksi käyttöohjeiden laadinnassa voidaan jo suunnitteluvaiheessa syntyneet kokoonpanokuvat ja piirustukset yhdistää sellaiseen tekstiin sekaan ja näin nopeuttaa ohjeiden valmistumista.

NOKIA
Informaatiojärjestelmät

INTERLEAF-OHJELMISTOT

Interleaf Inc:n ohjelmistot on suunniteltu tekniseen dokumentointiin ja graafisen teollisuuden käyttöön. Interleaf-ohjelmistolla voidaan yhdistää teksti ja grafiikka työaseman kuvauksella. Ohjelmistot ovat helppokäyttöisiä, joiden käyttöä helpottaa WYSIWYG-ruudulla.

Grafiikka voi olla vapaasti piirrettyä ja liikegrafiikkaa (pylväät, piirakat jne.). Järjestelmään voidaan lukea skannattuja viivo- ja sävykuvia sekä CAD-järjestelmien piirustustietoja. Aikaisemmin tehty työ voidaan hyödyntää symboli- ja kuvakirjastoista. Eri grafiikkatyyppisiä ja tekstejä voidaan vapaasti yhdistellä ja muokata.



Työasemat ovat 32-bittisellä keskusyksiköllä ja korkearesoluutiolla 19" grafiikkakäytöllä varustettuja Apollo-työasematietokoneita. Työasemat kytketään paikalliseen verkkoon ja voidaan liittää yleisimpiin tietojärjestelmiin. Tulostimena käytetään laserkirjoittimia ja latomakoneita.

Teknisiä asiakirjoja ja painotuotteita

Nokian dokumentointijärjestelmä on suunniteltu erityisesti tekniseen dokumentointiin. Sillä laaditaan esimerkiksi varaosakirjat, kokoonpano-, asennus-, käyttö- ja huolto-ohjeet, sekä tarjousasiakirjat.

Teknisten asiakirjojen teossa CAD-kuvien siirto on varsinkin käyttökelpoinen ominaisuus. Järjestelmällä päästäänkin saumattomaan kokonaisuuteen suunnittelun alkuvaiheista aina viimeiseen osatekijään eli dokumentointiin.

Interleaf-ohjelmisto soveltuu myös erilaisten painotuotteiden tekoon tietokoneavusteisesti. Yritysten sisäiseen ja ulkoiseen viestintään liittyy paljon tekstiä ja kuvia sisältäviä painotuotteita. Organisaatiokaaviot, sisäiset tiedotteet ja lehdet syntyvät työasemalla vaivattomasti. Samaten markkinointimateriaali, kalvot, raportit ja vuosikertomukset voidaan suurelta osin tehdä Interleaf-ohjelmistolla.

Aikaa ja rahaa säästyy

Dokumentointijärjestelmän edut perustuvat asiakirjojen laadinnan nopeuteen, laadun ja ymmärrettävyyden parantamiseen.

Järjestelmästä saadut kokemukset osoittavat merkittäviä ajansäästöjä. Samoin kustannukset per sivu ovat pienentyneet jopa kymmenenteen osaan verrattuna perinteiseen 'leikkaa-liimaa'-menetelmään. Erityisen suuriin aikajä-kustannussäästöihin päästään, kun asiakirjat tulostetaan omalla laserkirjoittimella ulkopuolisen palvelun sijasta.

Pitkällä aikavälillä saadaan hyötyä asiakirjojen elektronisesta arkistoinnista. Lisäkopioid ovat nopeasti saatavissa. Samaten asiakirjojen muuttaminen ja päivittäminen on helppoa.

Kaiken lisäksi järjestelmän käytön oppii muutamassa päivässä. Ja jo viikon harjoittelun jälkeen käyttäjä pystyy hyödyntämään järjestelmää täysitehokkaasti.

Lisätietoja: Juha Mäntylä,
puh. (90) 455 7299.

Yhteydet muihin tietojärjestelmiin

Työasemaan liitetyn tutkaimen (scanner) tai videokameran avulla voidaan asiakirjoihin yhdistää myös viiva- ja valokuvia.

Eri grafiikkatyytit ovat vapaasti yhdisteltävissä. Niin ikään niihin on mahdollista lisätä tekstiä ja niitä voidaan muokata. Asiakirjoihin liitettyjä kuvia ja kaavioita voidaan myös tallettaa symboli- ja kuvakirjastoihin, joista ne ovat myöhemmin uudelleen käytettävissä.

Valmiit asiakirjat tulostetaan yleensä laserkirjoittimella, jonka erottelukyky on 300 pistettä tuumalla. Tulostimeksi käy myös valolatokone.

Järjestelmän voi liittää useimpiin markkinoilla oleviin tietojärjestelmiin ja erityisesti tekstinkäsittelyjärjestelmiin tekstitiedon siirtoa varten.

Mikrotietokone soveltuu mainiosti liitettäväksi Apollo-työasemakoneeseen. Tällöin käyttäjä voi kirjoittaa raakatekstit mikrolla. Sen jälkeen siirretään tekstitiedostot Apollo-työasemalle, jossa tekstiä voidaan vielä muokata ja lisätä siihen grafiikkaa.

Esimerkiksi MikroMikko 2 on liitettävissä Apollo-työasemaan. Niinpä MikroMikon tekstinkäsittelyohjelmilla, mm. WordPerfectillä, tehdyt tekstit voidaan siirtää suoraan Interleaf-ohjelmistolle jatkokäsitteltäväksi. Vastaavasti Lotus 1-2-3 ja Multiplan-taulukot saadaan siirrettyä Interleafin asiakirjoihin ja pystytään esittämään ne businessgrafiikalla esimerkiksi piirakoina.

okoneavusteisesti

Kodistako tietoyhteiskunnan keskipiste?

Kotipiirissä työskentelyn pitäisi olla tuttua. Olihan se vallitseva elämäntapa vielä pari sukupolvea sitten. Työpaikan ja kodin tiukka erottaminen on vasta teollisuusyhteiskunnan seurausta.

Säilyykö nykyasetelma vai tuoko tietoyhteiskunta muutoksen tullessaan? Jos tietoyhteiskunnassa työskennellään myös kotona, niin miten se voidaan hoitaa käytännössä? Ja tuleeko kodista taas elämisen keskipiste?

Kotona tapahtuvaa työntekoa pidetään edelleen harvojen, kuten maalaisten, kotirouvien tai taiteilijoiden, elämäntapana.

"Minäkö työskentelin kotona? Ei ikinä! Et kai kuvittele, että nyhraisin 20 neliön yksiössäni kaikki päivät, kun jo vapaa-aikanakin meinaan tulla hulluksi siinä kopissa. Sitä paitsi melkein kaikki kotona työskentelystä tehdyt kokeilut osoittavat, että touhusta ei tule mitään — ei ainakaan pitkän päälle. Ihminen tarvitsee kontakteja työtovereihin."

Voiko sen enää sattuvammin sanoa? Kysymys kotona työskentelystä herättää meissä voimakkaita tunteita pikemmin kuin viileää harkintaa. Miksi?

Jo koulussa meidät opetetaan säännölliseen työskentelyyn kodin ulkopuolella. Ei siis ihme, että tunnetaan levottomuutta, kun aletaan puhua kotona työskentelystä. Eihän normaalikansalaisen elämänrytmi ole sitten kouluaikeiden miksikään muuttunut. Joka aamu verstaalle ja opettajan roolin saanutta pomoa totellaan mukisematta.

Teollisuusyhteiskunnassa ihmiselämä on ketju ohjattuja päivähoitopaikkoja; seimestä vanhustentaloon. Putkesta ulos astuminen sisältää aina suuren riskin. Elämä voi mennä pilalle eli joutua raiteiltaan, kun ei ole ketään valvomassa ja ohjaamassa.

kiinteä kahdeksan tunnin työputki vietetään kokonaan joko kotona tai työpaikalla. Mieleen ei ole hevin juolahtanut, että osa työtehtävistä, esimerkiksi paneutumista vaativat paperihommat, sujuvat parhaiten kotona. Sen sijaan epämääräisten ongelmavyyhtien selvittely sujuu parhaiten nokikkain palaverissa työpaikalla.

Epäonnistuneissa kokeiluissa koehenkilöksi on valittu tai valikoitunut yleensä äiti, jolla on päivähoito-ongelmia tai joka ei halua olla erossa lapsistaan.

Kokeilujen mukaan yksilön täytyy olla todellinen rautahermo, jos hän kestää pitkään moista menoa jossain nukkumalähiön kaksiossa. Työn takia hän menettää nekin aikuiskontaktit, joita hän ennen sai keskustellessaan muiden äitien kanssa hiekkalaatikolla. Työpaikasta hän saa työpaineet, mutta ei sen ihmiskontakteja. Vaikka hän kestäisi päivän yksinolon, täytyisi illaksi olla toimintaa ja tapaamisia. Nukkumalähiö on iltaisin hiljainen. 'Työssä' käyneet ihmiset haluavat levätä ylivirikköisen työpäivän jälkeen. Eli juuri silloin, kun kotona työskentelijällä on halua ja energiaa tavata ihmisiä, muut haluavat katsella telkkaria.

'Joko tai' -kokeiluja

Nopeaa vauhtia väistymässä olevan teollisuusyhteiskunnan malli työnteosta on lähes yksioikoinen. Sinulla on joko 'kahdeksasta neljään' -työpaikka tai olet kokonaan työtön. Jos kuulut työtä tekeviin, aikasi jaetaan minuutin tarkkuudella työ- ja vapaa-aikaan. Työt tehdään työpaikalla ja kotona levätään.

'Joko tai' -ajattelu näkyy eräissä kotona työskentelyä koskeneissa kokeiluissa. Niissä on lähdetty siitä, että

'Sekä että' -vaihtoehto

Kun muutama vanhakantaiselle 'joko tai' -ajattelulle pohjautunut kokeilu on



mennyt myttyyn, tehdään rautainen johtopäätös: kotona ei voi työskennellä.

Mikäli kotona työskentelyn mahdollisuuksia halutaan tutkia vakavasti, selvitystyö tulee aloittaa tutkimalla niitä, joille kotona työskentely (keskittymisrauha ja joustavuus) on aikaansaannosten elinehto. Heitä ovat esimerkiksi tiedemiehet ja tietotyön tekijät.

Erilaisten ihmisten kokemuksia yhdistelemällä ja kokeilemalla jokainen yksilö voisi kehittää itselleen elämäntavan, joka on työtehtävien hoidon kannalta mahdollisimman tehokas ja samalla miellyttävä. Joidenkin kohdalla tämä merkitsee työn ja vapaa-ajan ja/tai kodin ja työpaikan tiukkaa erotamista. Elämänmeno noudattelee vanhaa käytäntöä.

Kuitenkin enemmistölle tietoyhteiskuntaan siirtyminen merkitsee kodin ja työn erasteista sekoittumista. Ajatus-työtään voi tehdä ajasta ja paikasta riippumatta.

Tietotyötä tekevät voivat aivan hyvin tehdä suuren osan työstä kotona. Kotoa voidaan olla pääteyhteydessä toimistoon sekä muihin työntekijöihin.

Kun toimittajalla on paperityöhön tarvittava 'valkoinen ruuti' lopussa ja piipussa on vain 'punaista ihmissuhderuutia', voi pistäytyä naapurissa katso-
massa päivähoitossa olevaa lasta tai hölkkätä parin korttelin päähän jututtamaan tutkimustyötä tekevää tiedemies-
tä. Voisihan siitä irrota seuraavan jutun aihe.

Mitä useampi ihminen työskentelee valtaosan ajastaan kotona, sitä enemmän naapuristossa ollaan kotona päivisin. Tietenkään kaikki eivät ole aina paikalla, koska asiantuntijalla on pääkonttorissa puolen päivän mittaisessa palaverissa, kirurgi on sairaalassa leikkaamassa ja joku naapuri on tehtaalla valvomassa robottien työskentelyä jne.

Ympäri vuorokautistuminen

Tietoyhteiskunnassa riippuvuus ajasta hämärtyy. Alvin Tofflerin mukaan yhteiskunnan elämänrytmi ympärivuorokautistuu. Tämä taas edellyttää, että mahdollisimman monet palvelut ovat kellonajasta riippumattomia.

Pankkiautomaatista saa jo nyt rahaa 24 tuntia vuorokaudessa. Kotitietokoneella voi maksaa laskut pankin tietokoneen kautta. USAssa yhä useampi tilaa mm. lento- ja teatteriliput suoraan kotitietokoneella, haluamanaan vuorokaudenaikana.

Tietotekniikan elämäntapavaikutuksia pohtinut Paul Strassmann ennustaa, että kuvatus kaltaiset suoran käytön järjestelmät yleistyvät ja monipuolistuvat nopeasti — kannattavuussyistä. Yhä useammat asiakkaat eivät ole halukkaita maksamaan heitä palvelevien toimistotyöntekijöiden aiheuttamia kustannuksia, jos he voivat saada asiansa hoidetuksi tietokoneen avulla kotoa ja merkittävästi halvemmalla.

High tech plus high touch

Strassmann ennustaa, että pikkuhiljaa jokaiselle asuinalueelle syntyy ns. välittäjien (midleman) ammattikunta.

Miten järjestelmä toimisi käytännössä?

Olettakaamme, että naapuriin on syntynyt pienokainen. Ylpeä äiti ja isä tulevat näyttämään perillistään. Kahvinjuonnin ja ihastelevan hämmästelyn jälkeen vanhemmat voivat välittäjän (naapurin) avustuksella hoitaa monta käytännön asiaa.

Ehkä ensimmäiseksi todetaan, että perheen vakuutuksia täytyy vauvan syntymän johdosta muuttaa. Välittäjä ottaa tietokoneellaan yhteyden vakuutusyhtiön tietokoneeseen ja muuttaa perheen vakuutustiedot halutuiksi. Sit-
ten voidaan ottaa yhteys pankin tietokoneeseen ja maksaa mahdollinen vakuutusmaksu, jonka maksamiseksi vakuutusyhtiön kone teki elektronisen maksulapun.

Ja tosiaan, täytyyhän lapselle avata oma tili, johon sukulaiset voivat lähettää ristiäislahjan. Lapsilisät voidaan määrätä maksettavaksi suoraan lapsen tilille.

Istunnon kuluessa ilmoitetaan yksi ja toinen tieto siihen ja tähän virastoon ja laitokseen. Niin hoitui lapsen tuoma paperisota kahvikupposen äärellä jutustellessa, ilman luukulta luukulle juoksemista.

Tällaisessa toimintatavassa John Naisbittin peräänkuuluttamat High Tech (korkea teknologia, tehokkuus, nopeus, tarkkuus) ja High Touch (inhimillinen lämpö, elämykset, joustavuus, yksilöllinen kohtelu) yhdistyvät ihan-
teellisella tavalla.

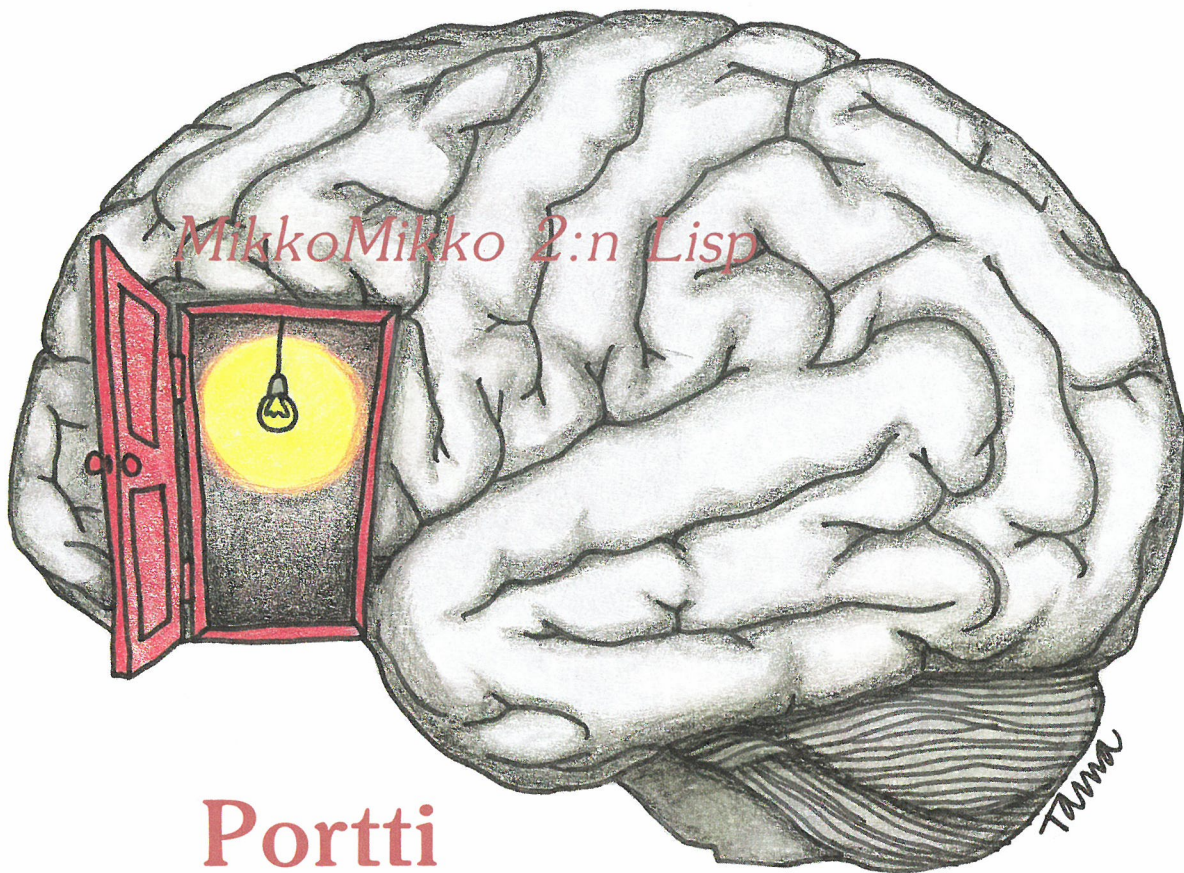
Tietenkin isä ja äiti menettävät pankkineidin pikaisen hymyn ja rauhoittavat 'tosi on' -vakuuttelut tiskin äärellä jonnottaessaan. Mutta vastapainoksi he saavat pitkän ja häiriöttömän juttutuokion, jonka aikana perheen tilannetta voidaan käsitellä kokonaisuutena.

Jos sittenkin

Vaikka esitetty visio saattaa tuntua kaukaiselta, on syytä löysätä pipoa ja tiukan kielteisiä asenteita. Varovaisinkin voi varmuudella sanoa, että työnte-
ko kotona ja erilaisten juoksevien asioiden hoito kotoa tietokoneella ovat todellisuutta joillekin jossain määrin jonkin ajan kuluttua.

Tietoyhteiskunnan ainoa käytäntö taitaa olla se, että jokaisella ihmisellä on yksilölliset työjärjestelyt. Nykyinen 'kahdeksasta neljään' on vain yksi niistä. □





Portti tekoälyn maailmaan

Lisp-kielestä on tullut nopeasti tekoälysovellusten valta-kieli. Teknologian kehitys on mahdollistanut myös Lispin mikrototeutukset. Nyt MikroMikollekin on saatavissa oma Lispinsä, Mikko Lisp.

Mikko Lisp ei ole pelkästään kieli, vaan täydellinen ohjelmointiympäristö. Se soveltuu tekoälyn koulutukseen ja protoiluun.

Mikko Lispin on tehnyt Entity Systems, Suomen ensimmäinen tekoälyn erikoistunut ohjelmistotalo, yhteistyössä Nokia Informaatiojärjestelmien Knowledge Engineering-ryhmän kanssa. Kieltä markkinoi Nokia Informaatiojärjestelmät.

Mikko Lisp mahdollistaa tekoälyn ja sen ohjelmointitekniikoihin perehtymisen jo pienin kustannuksin. Mikko Lisp on yhteensopiva Nokian edustamien Symbolics-koneiden Lispin kanssa. Niinpä se soveltuu myös Symbolics-käyttäjien koulutukseen.

Mikko Lisp on tulkki ja se toimii 512 kilotavun keskusmuistilla varustetussa MikroMikko 2:ssa. MS-DOS -käyttöjärjestelmästä edellytetään versiota 2.0.

Tehokas työkalu

Ohjelmistokustannusten jatkuva nousu vaatii ohjelmointityön tuottavuuden parantamista. Tarvitaan yhä tehokkaampia ohjelmointityökaluja. Lisp on tällainen työkalu.

Lisp on erinomainen kieli monimutkaisiin ohjelmointitehtäviin. Mitä monimutkaisempi tehtävä on ja mitä monimuotoisempia tieto- ja ohjausrakenteita siihen sisältyy, sitä paremmin Lisp siihen soveltuu.

Lisp on symbolinen ohjelmointikieli. Sen voima on rakenteeltaan monimutkaisen tiedon käsittelyssä. Siksi Lispistä on tullut tekoälyn tärkein ohjelmointikieli.

Useimmat Lispin toteutukset ovat kuitenkin olleet tehottomia. Niin voimakkaan kielen kuin Lispin toteuttami-

nen vaatii tinkimätöntä panostusta tehokkuuteen.

Mikko Lisp -projektissa keskeisenä suunnittelutavoitteena on ollut erityisesti tehokkuus. Tavoitteessa onkin onnistuttu hyvin, sillä Mikko Lisp -tulkki on miltei yhtä tehokas kuin Symbolics 3600 Lisp-koneen tulkki. Tosin Symbolicsin kääntäjä on huomattavasti tulkkia tehokkaampi.

Dynaaminen muistinhallinta

Monet vaativat ohjelmointitehtävät edellyttävät dynaamisia tietorakenteita. Siksi Lispin sisältyy automaattinen, täysin dynaaminen muistinhallinta, sisältäen ns. roskien keruun. Lisp-ohjelmat käsittelevät tyypillisesti dynaamisia tietorakenteita, joiden laajuus ja muoto määräytyvät vasta ajoaikana.

Dynaamisten rakenteiden käsittely Lispillä on yksinkertaista, koska ohjelmoijan ei tarvitse mieltää rakenteita samalla tavalla kuin perinteisissä ohjelmointikielissä, osoittimina.

Lispissä tietorakenteet on käsiteltävissä puhtaasti käsitteellisellä tasolla ilman, että toteutukseen tarvitsee sen enempää puuttua. Ohjelmat puhuvat olioista, ei olioiden esitysmuotoja sisältävistä muistipaikoista.

Sen sijaan perinteisillä kielillä ohjelmoija joutuu aina uudelleen luomaan tietorakenteita käsittelevät mekanismit ja huolehtimaan siitä, että muisti vapautuu ajallaan. Tämä pidentää ohjelmia sekä johtaa helposti vaikeisiin virheisiin.

Lisp-koodin sisäinen esitysmuoto on lista, joka on kielen perustietorakenne. Itse asiassa Lisp on lyhenne sanoista 'List Processor'. Täten Lisp-koodia voi käsitellä kuten mitä tahansa tietorakennetta. Ohjelma voi muodostaa lausekkeita ja suorittaa niitä. Niinpä Lisp-ohjelmalla voi helposti generoida uusia Lisp-ohjelmia.

Avoin kieli

Lisp on laajennettavissa itsellään. Eli kieleen pystytään tarvittaessa lisäämään ongelmakohtaisia rakenteita. Myös uusia ohjausrakenteita ja syntaktisia laajennuksia on mahdollista tehdä esimerkiksi makroilla.

Lispillä on helppo toteuttaa edelleen muita kieliä rakentamalla ne sen päälle. Miltei kaikki tekoälyssä käytetyt kielit ja AJ-kehittimet on rakennettu Lispin päälle.

Kun uusi kieli tai järjestelmä rakennetaan Lispin päälle, se perii perusjärjestelmän ilmaisuvoiman ja kaikkia Lispin valmiita välineitä voidaan hyödyntää myös uudessa järjestelmässä. Jos

esimerkiksi rakennetaan Fortran Lispin päälle, voidaan järjestelmässä ajaa Fortran-ohjelmia ja tehdä niihin lisäyksiä Lispillä.

Common Lisp standardina

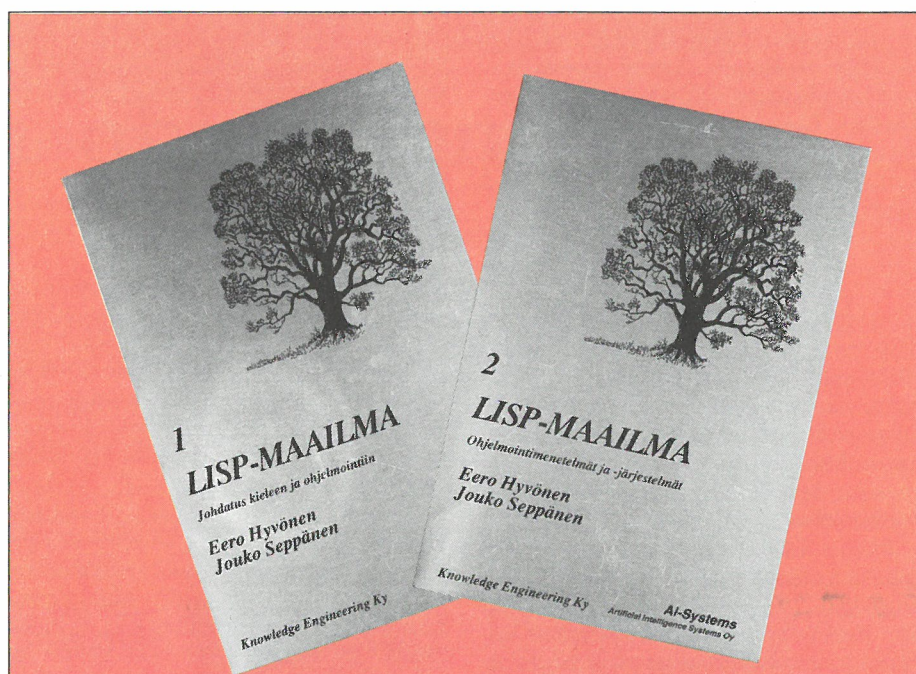
Lispin kaupallisen käytön kasvaessa syntyi tarve teollisuusstandardiin. Common Lisp on tämä standardi.

Common Lisp määrittelee noin 800 valmista standardifunktiota. Koska monimutkaisetkin operaatiot ovat standardoituja, ohjelmista tulee helposti luettavia. Mikko Lisp sisältää noin 600 Common Lisp -standardissa määriteltyä funktiota sekä lukuisia laajennuksia.

Common Lisp on 25 vuoden jatkuvan kehityksen tulos. Se on synteesi

aiempien Lisp-murteiden hyvistä piirteistä. Siihen ovat vaikuttaneet hyvin monet aiemmat Lispit. Common Lisp on hyvin lähellä Symbolicsin Lisp-koneissa ennen Common Lispä käytettyä Zetalisp-murretta ja MIT:ssä kehitettyä suorkoneilla toimivaa MacLispia. Common Lisp on laajuudestaan huolimatta johdonmukainen ja säännöllinen kieli.

Common Lisp on täysipainoinen yleiskieli. Se ei rajoitu pelkästään symboliseen tietojenkäsittelyyn, kuten monet aiemmat Lispit, vaan sisältää välineitä myös muunlaiseen ohjelmointiin. Esimerkiksi aritmetiikka on erittäin laajaa. Se tukee rajoitettuja sekä äärettömän pitkiä kokonaislukuja, murtolukuja, neljän eri tarkkuuden liukulukuja



Eero Hyvönen — Jouko Seppänen:

LISP-MAAILMA

Suomalaisten alan huipputekijöiden kirja tarjoaa johdatuksen Lisp-kielen käsitteistöön ja välineisiin sekä symbolien käsittelyyn ja tekoälyn ohjelmointitekniikan yleisiin perusteisiin. Keskeisessä asemassa ovat funktionaalinen ohjelmointi ja oliokeskeinen ajattelu tietotyypeissä ja -rakenteissa.

Kirja on laadittu oppikirjakäyttöön yliopistoissa ja korkeakouluissa sekä ammatillisessa täydennys- ja jatkokoulutuksessa. Se soveltuu myös itseopiskeluun harrastajille. Kirja tarjoaa myös katsauksen tekoälyn uusimmista tutkimus- ja kehitysnäkymistä.

Kirja toimitetaan Mikko Lispin mukana. Lisäksi kirjaa on saatavissa kirjakaupoista.

sekä kompleksilukuja. Näille kaikille on määritelty suuri joukko funktioita, jotka sisältävät peruslaskutoimitusten lisäksi useita kymmeniä transsendentti ym. funktioita.

Common Lispissa on systeemiohjelmointiin joustava bittikenttien käsittely, hash-tauluja sekä bittivektoreita.

Common Lisp sisältää kehittyneitä syöttö- ja tulostusfunktioita, joilla vuorovaikutteisten, käyttäjäystävällisten sovellusohjelmien kirjoittaminen on helpoa.

Vasta Common Lisp on tehnyt Lispistä kilpailukykyisen teollisessa ja kaupallisessa käytössä. Yhtenäinen standardi mahdollistaa helpon ohjelmistojen siirron. Niin ikään ohjelmistojen ei tarvitse opetella aiempien lukemattomien eri Lisp-murteiden piirteitä.

Yhdysvaltain puolustusministeriö sekä alan teollisuus, kuten Symbolics, ovat omaksuneet Common Lisp -standardin. Common Lisp on noussut Adan rinnalle toiseksi Yhdysvaltain puolustusministeriön hyväksymäksi kieleksi.

Ohjelmointiympäristö

Ohjelmankehitys Mikko Lisp -järjestelmällä on mukavaa, koska perinteinen käännös-linkkaus -sykli puuttuu kokonaan; Mikko Lisp on tulkki. Ohjelmia voidaan testata jokaisen pienen muutoksen jälkeen välittömästi.

Mikko Lisp sisältää laajan valikoidun ohjelmistotyökaluja, kuten mm. muokkain, kuuntelija, askellin ja jäljitin. Kaikki työkalut on integroitu samaan ympäristöön ja ne ovat käytettävissä rinnakkain ja sisäkkäin. Esimerkiksi muokkain on kutsuttavissa ohjelmien sisältäkin. Niinpä Lisp-istunnossa voi olla useita sisäkkäisiä muokkaus- ja ajovaiheita.

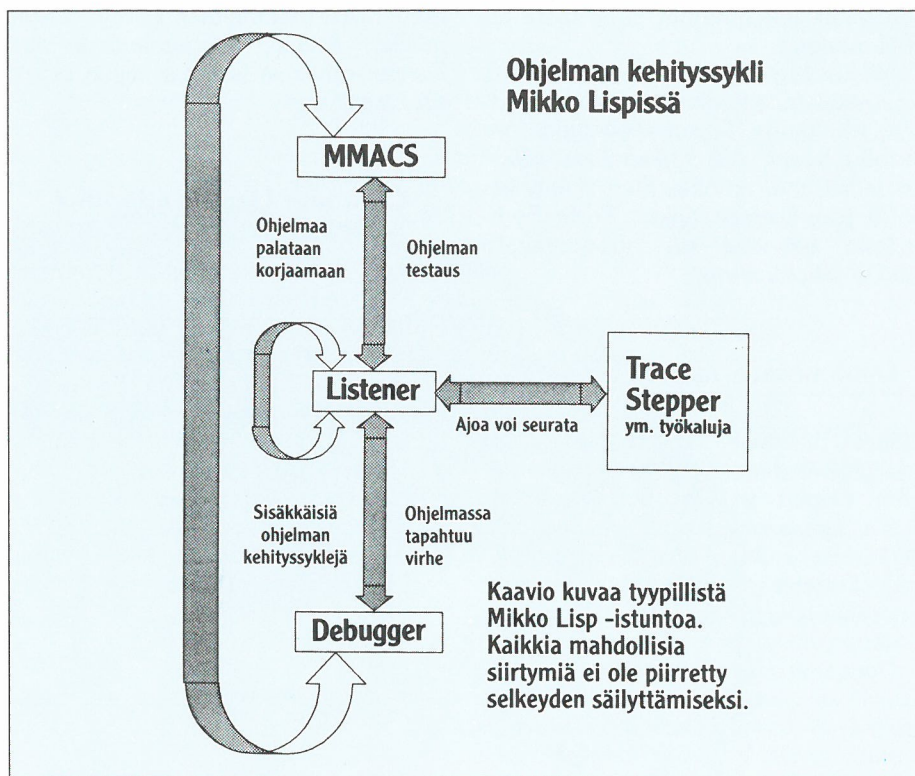
Mikko Lisp -ohjelmointiympäristö muistuttaa rakenteeltaan paljon Symbolics-koneiden Lisp-ympäristöä. Lisp-kuuntelija on pääasiallinen keino kommunikoida Lisp-tulkin kanssa. Kuuntelija toimii komentotulkin tavoin.

Lukuunottamatta Lisp-tulkkia ja muokkaimen perusosia järjestelmä on toteutettu Lispillä.

Ohjelman kehityssykli

Mikko Lisp -järjestelmän rakenne mukaillee ohjelman kehityssykliä. Kutakin vaihetta vastaa oma työkalu.

Ohjelmat kirjoitetaan MMACS-muokkaimella. Kutakin ohjelmaa tai tiedostoa vastaa MMACSissä ikkuna.



Kun ohjelma on kirjoitettu, voidaan siirtyä testaamaan sitä 'Lisp-ikkunaan', Lisp-kuuntelijaan, joka lukee ja suorittaa Lispin lausekkeita vuorovaikutteisesti.

Kun ohjelmassa syntyy virhetilanne, Lisp ilmoittaa virheestä ja siirtyy debuggeriin, korjausohjelmaan, jossa voidaan tutkia järjestelmän tilaa. Debuggerilla voidaan tarkastella virhettä edeltäneitä funktiokutsuja, funktioparametrien sekä paikallisten muuttujien arvoja.

Virhettä voidaan palata korjaamaan MMACSiin. Tai mikäli virhettä ei pystytä heti paikallistamaan, voidaan sitä etsiä tarkkailemalla, miten ohjelma siihen joutuu. Tässä ohjelmiojaa auttavat jäljitin ja askellin. Ja kun virhe on jäljitetty, se voidaan korjata MMACSissa ja testata uudelleen.

Jäljitin eli Trace sallii funktioiden kutsuparametrien ja palautusarvojen tarkastelun ohjelman ajon aikana. Ohjelmoija voi valita ne funktiot, joita jäljitetään.

Askellin eli Stepper mahdollistaa Lisp-lausekkeiden suorituksen askeltamisen lauseke kerrallaan. Tällöin jokainen ohjelman operaatio tulee näkyviin. Vasta kun ohjelmoija tahtoo, ohjelma etenee seuraavan askeleen. Mikko Lispissä on mahdollista asettaa myös halutut funktiot askellustilaan. Tällä tavalla on mahdollista asettaa ohjelmaan keskeytyskohtia (breakpoint).

Milloin tahansa Lisp-ohjelman ajaessa sen voi pysäyttää ja esimerkiksi mennä debuggeriin katsomaan ohjelman tilaa tai mennä uuteen Lisp-kuuntelijaan suorittamaan Lisp-lausekkeita ja vaikkapa muita ohjelmia. Keskeytettyä ohjelmaa voidaan tämän jälkeen jatkaa.

MMACS, Mikko Lispin tekstimuokkain, on suosittu EMACSin sukulainen. Se on suunniteltu erityisesti Lisp-ohjelmien kirjoittamiseen. Koska se sisältää Lisp-lausekkeita automaattisesti, ohjelmien kirjoitusasu säilyy yhdenmukaisena ja selkeänä. Se laskee myös sulkuja. Kohdistimen ollessa sulkeutuvan sulun kohdalla vastaava aukeava sulku vilkkuu. Niinpä ohjelmoijan on helppo hallita sulut, joita Lisp-koodissa on runsaasti.

Mikko Lisp -ohjelmalevykkeen mukana toimitetaan seuraavat kirjat:

- Hyvönen-Seppänen: Lisp-maailma
- Mikko Lisp Tutorial
- Mikko Lisp User's Guide
- MMACS Manual
- Mikko Lisp Reference Manual

Flavors-järjestelmä

Tärkein Mikko Lispin Common Lisp -laajennuksista on oliokeskeistä ohjelmointia tukeva Flavors-järjestelmä. Se on Lispin mikrototeutuksissa ainutlaatuinen piirre.

Oliokeskeisyys tekee suurista ohjelmistoista helposti hallittavia. Näin Flavors edistää ohjelmien modulaarisuutta.

Flavors-lähestymistapa soveltuu erinomaisesti simulointiin ja todellisen maailman mallittamiseen.

Lisätietoja: Matti Virtanen,
puh. (90) 455 7238, ja
Tauno Korpela,
puh. (90) 455 7233

ADA

parantaa ohjelmien laatua

Adaa tarkastellaan usein pelkkänä ohjelmointikielenä. Adan taustalla on kuitenkin monia ohjelmistotuotannon uusia periaatteita ja menetelmiä.

Adan suunnittelussa on pidetty kiinni sellaisista periaatteista kuin ohjelmien luotettavuus, ylläpidettävyys ja ymmärrettävyys. Ada-ohjelmointiympäristössä on huomioitu erityisesti ohjelmien ylläpito ja jatkokehitys, sillä voivathan nämä vaiheet haukata valtaosan ohjelmiston elinkaaren kustannuksista.

Ada kielenä

Adan perusrakenteet, kuten lauseet, tietotyypit ja aliohjelmat, ovat periaatteiltaan hyvin samanlaiset kuin vastaavat rakenteet esimerkiksi Pascalissa. Nämä ominaisuudet ovat kuitenkin vasta pieni osa Adasta. Adan varsinaisen voima on sen uusissa piirteissä.

Pakkaus (package) on Ada-ohjelmien perusrakennuspalikka. Pakkaus on itsenäinen ohjelmayksikkö, joka koostuu joukosta yhteenkuuluvia olioita kuten tietotyyppejä, muuttujia ja aliohjelmiä. Pakkauksessa voidaan erottaa kaksi osaa, pakkauksen rajapinta eli määrittely ja runko eli pakkauksen toteutus. Pakkauksen ominaisuuksiin päästään käsiksi pelkän pakkausmäärittelyn avulla. Niinpä muun ohjelmiston ei tarvitse tietää mitään rungon sisällöstä.

Ohjelmointivirheet saattavat aiheuttaa korvaamattomiakin vahinkoja. Siten ohjelmointikielen luotettavuus, luettavuus ja ylläpidon helppous ovat ohjelmistotyössä ensiarvoisen tärkeitä. Ada-ohjelmointiympäristö perustuu juuri näihin ohjelmiston laatuun vaikuttaviin tekijöihin.

Pakkaus voi olla myös parametroitu eli ns. geneerinen malli. Tällainen malli soveltuu eri käyttötarkoituksiin. Oteetaan esimerkiksi yleinen listarakenteen käsittelijä, jonka parametreina ovat listan alkion tietotyyppi ja listan alkiodien järjestyksen määrittelevä aliohjelma. Mallia voi erilaisilla parametreilla käyttää esimerkiksi henkilötietueita tai tilitahtumia sisältävien listojen käsittelyyn.

Useimmissa ohjelmointikielissä rinnakkaisten prosessien hallinta on jätetty käyttöjärjestelmäpalvelujen varaan. Adassa sen sijaan rinnakkaisuuden hallinta sisältyy itse kieleen, sillä Adan tehtävät (tasks) ja niihin liittyvät lauseet mahdollistavat rinnakkaisten toimintojen ohjelmoinnin hallitusti. Tehtävien käynnistys, niiden välinen kommunikointi ja synkronointi, sekä tarvittavat jonotusmekanismit on määriteltävä kielellä ja toteutettu sen ajoaikaisessa tukijärjestelmässä.

Virhetilanteiden havaitseminen ja niistä toipuminen on aina huomioitava ohjelmissa. Virhekäsittelyn testaaminen on kuitenkin usein kaikkein vaikeinta ohjelmatestauksessa ja siten se on altteinta virheille. Adassa virhekäsittely hoituu poikkeusmekanismilla. Sen avulla voidaan varautua ja hoitaa ajoaikaiset virhetilanteet. Adassa virhetilanne eli poikkeus voi olla kieleen kuuluva

standardipoikkeus tai itse ohjelmoijan määrittelemä. Standardipoikkeuksia ovat esimerkiksi indeksointi taulukon ulkopuolelle, aritmeettinen ylivuoto tai tulostuksessa tapahtunut laitevirhe. Poikkeustilanteet käsitellään erillään ohjelman päälogiikasta, joka näin säilyy yksinkertaisena ja helpopolukuisena.

Ei pelkkä kieli

Ada on enemmän kuin pelkkä ohjelmointikieli. Siihen liittyy oleellisena osana kokonainen ohjelmointiympäristö APSE (Ada Programming Support Environment).

APSE on eräänlainen Ada-ohjelmoijan työkalupakki. Se muodostuu ohjelmakirjastosta ja erilaisista ohjelmointityökaluista. APSEen sisältyy myös konekohtainen liitäntä KAPSE, joka mahdollistaa koneriippumattomuuden.

APSEn ja koko Ada-ohjelmiston siirto koneelta toiselle vaatii siten vain KAPSEn uudelleenkirjoittamisen.

APSEn työkaluja ovat mm. muokkain (editor), kääntäjä, yhdistelijä (linker) ja jäljitysohjelma (debugger). Kehittyneestä APSEsta löytyy työkaluja kaikkien ohjelmointiprojektin osa-alueiden käyttöön. Erityistä huomiota APSEssa on kiinnitetty ohjelmiston ylläpidon ja jatkokehityksen tukemiseen.

Ada-ohjelman rakennuspalikat, pakkaukset ja aliohjelmat, voidaan kääntää erillään, vieläpä käännösyksikön määrittely erillään vastaavasta rungosta. Erilläänkääntäminen on projektiryhmän työskentelyn kannalta tärkeää. Koska käännösyksiköiden rungot tarvitaan vasta testausvaiheessa, ohjelmoijat voivat toteuttaa niitä rinnakkain ja toisistaan riippumatta.

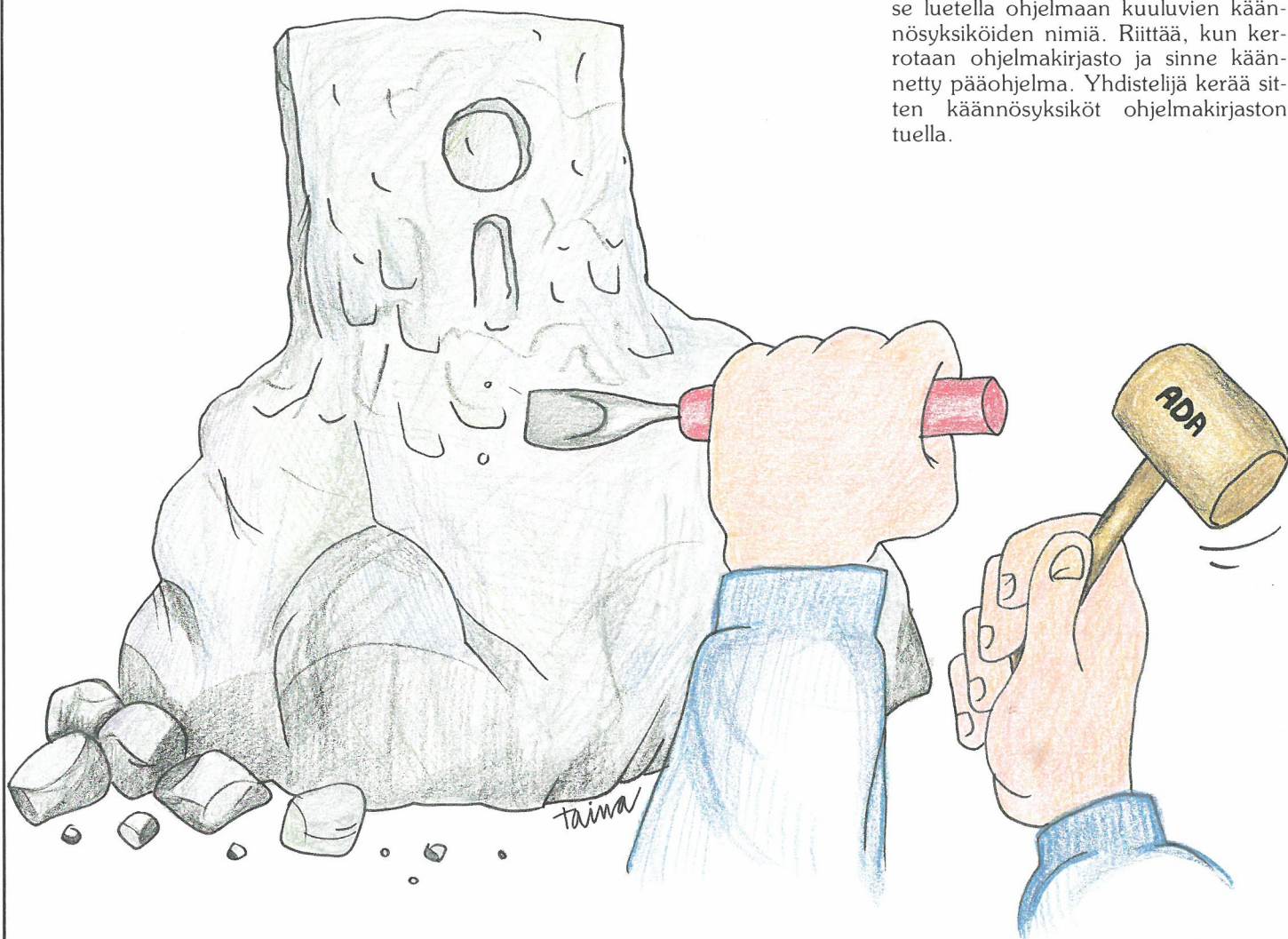
Ada-ohjelmakirjasto on ohjelmistoprojektin tietokanta, joka sisältää keskitetysti kaiken projektissa syntyvän informaation. Kirjasto hallitsee ohjelmis-

ton erillään käännettäviä osia ja niiden välisiä riippuvuuksia. Siten Ada-kielen standardi määrittelee kirjastolle ne ominaisuudet, jotka ovat välttämättömiä Ada-kääntäjän kannalta. Muut mahdolliset palvelut ovat sen sijaan kunkin APSE-toteutuksen määrättävissä.

APSEn työkalut on integroitu toisaalta Ada-kieleen ja toisaalta ohjelmakirjastoon.

Integroituminen Ada-kieleen tarkoittaa sitä, että työkalut keskustelevat käyttäjän kanssa Adan termin; eivät, kuten perinteisissä järjestelmissä, laitteiston ja käyttöjärjestelmän termin. Niinpä esimerkiksi Ada-jäljitysohjelman (debugger) käyttö ei enää edellytä koneen käskykannan tai muistiosoitteiden tuntemista.

Integroituminen ohjelmakirjastoon näkyy esimerkiksi niin, että ohjelmoija ei rakentaessaan suurtakaan ohjelmistoa käsittele erillisiä tiedostoja (lähde-ohjelma, objektikoodi jne.), vaan yhtä ohjelmakirjastoa. Kirjasto hallitsee vastaavat tiedot automaattisesti. Esimerkiksi Ada-yhdistelijälle (linker) ei tarvitse luetella ohjelmaan kuuluvien käännösyksiköiden nimiä. Riittää, kun kerrotaan ohjelmakirjasto ja sinne käännetty pääohjelma. Yhdistelijä kerää siten käännösyksiköt ohjelmakirjaston tuella.



Luotettavuutta lisää

Tänä päivänä tietokonejärjestelmien on toimittava pitkiä aikoja ilman ihmisen valvontaa. Ohjelmistot ovat kriittisessä asemassa esimerkiksi ydinvoimalan valvontajärjestelmässä tai pankkijärjestelmässä. Tällöin ohjelmistojen ehdoton luotettavuus on välttämätöntä.

Ohjelmoinnissa, kuten inhimillisessä työssä yleensäkin, syntyy virheitä. Ohjelmointikielten kehityksessä se on johtanut kielen sääntöjen lisääntymiseen. Ja samalla kääntäjien on täytynyt tarkastaa yhä enemmän.

Ensimmäisen sukupolven ohjelmointikielissä säännöt liittyivät pelkästään ohjelman kirjoitusasuun eli syntaksiin. Uudemmat kielet, ennen muuta Pascal, määrittelevät syntaksisääntöjen lisäksi joukon kielen merkitykseen, semantiikkaan, liittyviä sääntöjä. Pascalissa tällaisia ovat esimerkiksi tietotyyppien yhteensopivuuteen liittyvät säännöt. Adassa sääntöjen joukkoa on laajennettu vielä melkoisesti verrattuna Pascaliin. Niinpä Adassa inhimilliset huolimattomuusvirheet löytyvät mahdollisimman usein jo ennen testausta.

Kaikkia virheitä ei voida kuitenkaan havaita vielä ohjelmaa käännettäessä. Adassa tällaiset ajoaikaiset virhetilanteet hoidetaan poikkeuskäsittelymekani-

nismin avulla. Mekanismi auttaa ohjelmoijaa varautumaan mahdollisiin virhetilanteisiin rikkomatta ohjelman peruslogiikkaa. Ada-ohjelmisto onkin luotettava erilaisissa virhe- ja poikkeustilanteissa.

Ylläpidettävyys helpottuu

Ohjelmisto kirjoitetaan vain kerran ja yleensä vielä suhteellisen lyhyessä ajassa. Sen sijaan sitä muutetaan ja ylläpidetään vuosia. Niinpä suurimmat säästöt ohjelmiston elinajan kustannuksissa voidaan saavuttaa ylläpitovaiheessa.

Kun yhteen ohjelmiston kohtaan tehdään muutos, se aiheuttaa usein muutostarpeita muihinkin ohjelmiston osiin. Ja myös sellaisiin osiin, joihin muutoksella ei etukäteen näyttänyt olevan mitään vaikutusta. Oikein käytettynä Adan pakkaus rakenne vähentää merkittävästi tätä ilmiötä.

Hyvin suunnitellussa Ada-ohjelmassa useimmat muutokset rajoittuvat yhteen tai muutamaan pakkausruntoon. Koska vastaavat pakkausmäärittelyt säilyvät muuttumattomina, muutoksen vaikutukset eivät voi levitä muualle ohjelmistoon.

Jos jotakin pakkausmäärittelyä joudutaan muuttamaan, kertoo Ada-järjestelmä kaikkien muuttunutta pakkausta käyttävien osien nimet. Näin järjestelmä osaa automaattisesti rajoittaa muutoksen vaikutusalueen sekä vaatia sen edellyttämät korjaukset ja uudelleenkäännökset.

Ymmärrettävyys paranee

Ohjelmiston ylläpidosta vastaavat usein eri henkilöt kuin sen alkuperäiset tekijät. Näin ollen on tärkeää, että ohjelmakoodia pystyvät lukemaan ja ymmärtämään muutkin kuin sen kirjoittajat.

Uusissa kielissä, ensimmäisten joukossa Pascalissa, on huomioitu kielen perusrakenteiden, lauseiden ja aliohjelmarakenteiden, selkeys ja ymmärrettävyys. Strukturoitu ohjelmointi 1970-luvulla kehitti standardin näille rakenteille.

Lausetason ymmärrettävyyttä tärkeämpi on koko ohjelmiston ymmärrettä-

vyys. Adan ratkaisu tähän kysymykseen on pakkaus. Kun aliohjelma on tapa jakaa ohjelmakoodi ymmärrettäviksi osiksi, on pakkaus tapa jakaa edelleen aliohjelmat ja niille yhteiset oliot (muuttujat, tietotyypit, tiedostot, jne.) selvästi määritellyiksi ymmärrettäviksi joukoiksi.

Suurtakin Ada-ohjelmistoa voidaan tarkastella pakkaus kerrallaan. Pakkauksen ymmärtämiseksi riittää vain pakkauksen rungon ja sen käyttämien muiden pakkausten määrittelyjen tunteminen. Kokonaiskuva ohjelmistosta saadaan, kun tunnetaan sen keskeisimpien pakkausten määrittelyosat ja niiden väliset suhteet. Ohjelmiston yksityiskohtainen tarkastelu ja ymmärtäminen voi tapahtua pienempien osien, pakkausrunkojen, tasolla.

Ada on yleiskäyttöinen

Ohjelmistot koostuvat perinteisesti sovellusohjelmaosista ja systeemiohjelmien kaltaisista osista. Lisäksi niihin voi liittyä vielä reaaliaikapiirteitä, rinnakkaisten prosessien hallintaa ja vaikkapa epästandardien laitteiden käsittelyä. Erilaiset osat on ohjelmoitu yleensä eri kielillä.

Yleiskäyttöistä Adaa voidaan käyttää kaikkien osien ohjelmointiin. Sovellusosissa käytetään Adan peruspiirteitä, kuten lauseita, aliohjelmaa ja pakkauksia. Systeemiohjelmaosissa voidaan käyttää lisäksi tehtäviä, keskeytysmekanismeja ja esitysmuotomäärittelyksiä. Saman ohjelmointikielen käyttö kaikkialla pienentää koulutuskustannuksia sekä auttaa eri osien tekijöiden välistä kommunikointia.

Adan pakkaus rakenne ja siihen liittyvä geneerisyys mahdollistavat yleiskäyttöisten komponenttien rakentamisen. Esimerkkeinä komponenteista mainittakoon erilaisten tietorakenteiden (listat, jonot, pinot jne.) käsittelijät, tiedostojen käsittelijät, hakemistorakenteet ja lajitteluohjelmat. Ohjelmiston rakentaminen valmiista testatuista ohjelmakomponenteista on tehokkaampaa ja luotettavampaa kuin vastaavien osien suunnittelu ja teko alusta alkaen aina uudelleen.

Ohjelmakomponentteja on käsitelty tarkemmin Softarissa omassa artikkelisarjassaan. □

Ada yhdessä ohjelmistotuotannon menetelmien ja niitä tukevan ohjelmointiympäristön kanssa muodostaa uuden, tehokkaamman ja laadukkaamman pohjan ohjelmistoprojekteillemme. Tätä kokonaisuutta, jota kutsutaan Ada software engineering -kulttuuriksi, tarkastellaan tässä artikkelissa ohjelmistojen laadun kannalta. Seuraavassa Softarissa jatkamme aiheita käsittelemällä Adan vaikutuksia ohjelmointiprojektin eri vaiheisiin.

Ada-ohjelmakomponentti-kirjasto

Osa 3: Jonot ja pinot

Ada-ympäristössä voidaan ohjelmistotyön tuottavuutta kohottaa merkittävästi käyttämällä valmiita ohjelmakomponentteja. Tässä komponentteja esittelevän sarjan kolmannessa osassa paneudumme jonojen ja pinojen käsittelyyn.

Artikkelisarjan aiemmissa osissa olemme jo käsitelleet merkkijonoja sekä erilaisia listoja.

Jonot

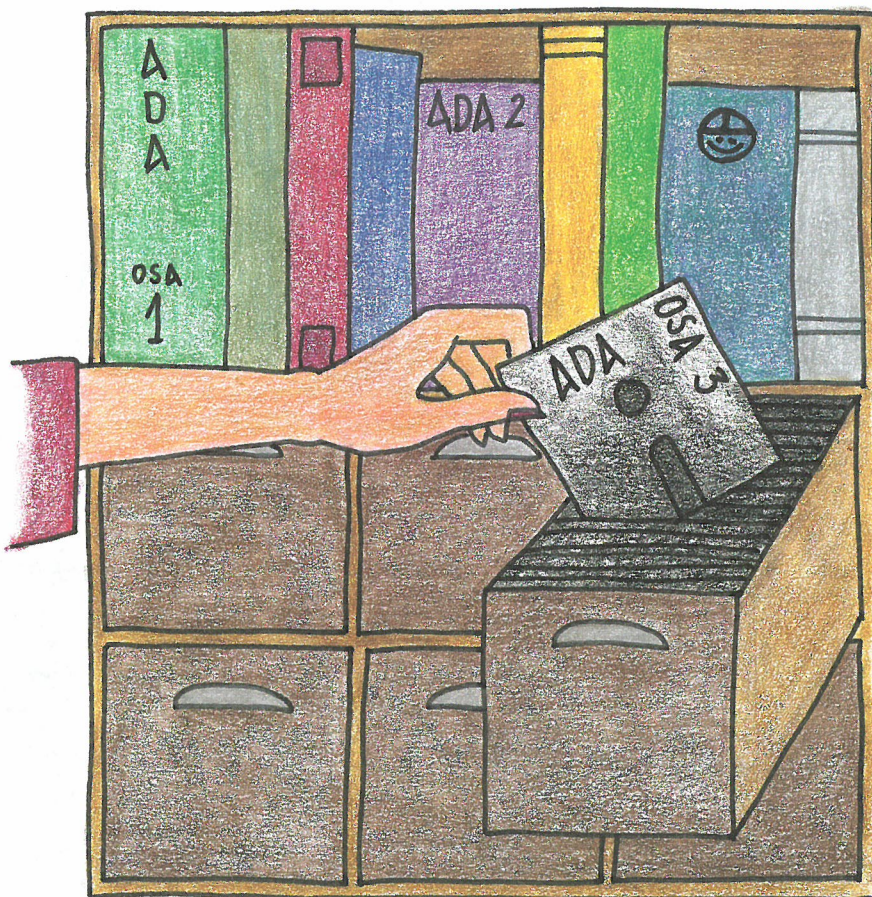
Jono on lineaarinen lista (ks. osa 2), jossa alkiot lisätään ja poistetaan listan eri päistä. Yleensä lisäykset kohdistuvat jonon loppuun ja poistot jonon alkuun. Jonolla on siten First-In-First-Out (FIFO) luonne.

Jonossa on 0 — n alkioita. Toteutettaessa jono on huomioitava jonon maksimipituus. Jonot on yleensä toteutettu siten, että ne ovat tietokoneen keskusmuistissa. Ohjelman koon kannalta on tärkeää, että jonon maksimipituus voidaan määrittellä lähelle suurinta todellista pituutta. Tällöin jonolle voidaan varata kiinteän mittainen muisti-alue. Todellista maksimipituutta ei kuitenkaan usein pystytä tietämään etukäteen. Tällöin on varauduttava ohjelmassa siihen, että muisti loppuu. Jos käytetään ulkoista tiedostoa jonon toteutukseen, maksimipituuden voi määrittellä huomattavasti suuremmaksi. Myös virtuaalimuisti mahdollistaa pitkät jonot.

Jonon pituus vaihtelee ohjelman suorituksen aikana. Mitä tehdään vapautuneille muistipaikoille?

Jos ei tehdä mitään, muisti saattaa loppua. Mikäli jonoon lisätään vain rajallinen määrä uusia alkioita, tämä tapa saattaa olla tehokkain. Useimmin kuitenkin tästä 'roskankeruusta' joudutaan huolehtimaan.

Helpoin tapa on lisätä vapautuneet alkiot niin sanottuun vapaiden alkioiden listaan. Jonon uudelle alkioille saadaan varattua tilaa vapaiden alkioiden listasta. Vasta kun tämä lista on tyhjä,



TAIMA

varataan uusi alkio muistista. Tällöin muistin käyttö on tehokkainta.

Jono tulee toteuttaa riittävästi ja täydellisesti. Riittävällä tarkoitetaan sitä, että ohjelmoijalla on käytössään riittävästi operaatioita jonon käsittelemiseksi. Täydellisyys puolestaan merkitsee jonon kaikkien ominaisuuksien huomioon ottamista.

Jos useampi rinnakkainen ohjelma käsittelee samaa jonoa, tulee huolehtia, että ohjelma on suorittanut jono-operaationsa loppuun ennen kuin toinen ohjelma päivittää jonoa. Tähän käytetään Adan tehtävä-käsitettä (task), jolla voidaan estää ohjelmia häiritsemästä toisiaan.

Jonopakkausessa on huomioitava myös erilaiset virhetilanteet. Adassa virhetilanteet hoidetaan poikkeuksien avulla. Kun aliohjelma havaitsee virhetilanteen, se aiheuttaa poikkeuksen. Poikkeus keskeyttää aliohjelman suorituksen ja seuraavaksi siirrytään erityiseen poikkeuskäsittelijään tai kutsuvaan ohjelmaan. Poikkeusten käyttö selkeyttää ohjelman rakennetta.

Jos jonoon yritetään lisätä uusi alkio, eikä muistissa ole enää tilaa, aiheutetaan OVERFLOW -poikkeus. Jos tyhjistä jonosta puolestaan yritetään poistaa alkio, aiheutetaan UNDERFLOW -poikkeus. Jos viitataan tyhjän jonon alkuun tai osoitetaan olematonta alkioa, aiheutetaan ELEMENT_NOT_FOUND -poikkeus.

Jonon alkion tyyppi pitää olla ohjelmoijan määrättävissä. Jonon käsittelyrutiinit ovat samoja tyypistä riippumatta. Riittää, kun kirjastossa on yksi jonopakkaus, josta voidaan muodostaa tarvittavia jononkäsittelijöitä. Kun pakkaus on valmiiksi testattu, ei ohjelmoijan aikaa kulu turhaan näiden standardirutiinien testaukseen.

Jonon operaatiot

Jono voidaan toteuttaa paitsi lineaarisen listan avulla myös suoraan Adalla. Kummassakin tapauksessa jonoon kohdistuvat operaatiot ovat samat.

Jonon käsittelyyn tarvitaan useita operaatioita. Alkion lisäämisen ja poistamisen lisäksi tarvitaan operaatiot jonon pituuden tutkimiseen, yksittäisen alkion osoittamiseen sekä uuden jonon luontiin, jonon kopiointiin ja kahden jonon vertailuun.

Operaatio ENQUEUE (Q; E) lisää alkion E jonon Q loppuun. Muistitilan loppuessa kesken aiheutuu OVERFLOW -poikkeus.

Operaatio DEQUEUE (Q, E) kopioi jonon Q ensimmäisen alkion muuttu- jaan E ja poistaa sen jonosta. Yritettäessä poistaa alkio tyhjistä jonosta aiheutuu UNDERFLOW -poikkeus. Siksi tarvitaankin funktio IS_EMPTY (Q), joka palauttaa arvon TRUE, jos jono on tyhjä, muuten arvon FALSE. Funktio ei aiheuta poikkeuksia.

Funktiolla FRONT (Q) saadaan jonon ensimmäisen alkion arvo. Tämä eroaa proseduurista DEQUEUE siinä, ettei alkioa poisteta jonosta. Tyhjä jono aiheuttaa ELEMENT_NOT_FOUND -poikkeuksen.

Jonon muiden alkioden arvo saadaan funktiolla PEEK (Q, N), joka palauttaa arvonaan jonon N:n alkion arvon. Jos jono on lyhyempi kuin N, aiheutetaan ELEMENT_NOT_FOUND -poikkeus.

Proseduurilla EMPTY (Q) luodaan uusi jono Q.

Jonon Q pituutta voidaan kysyä funktiolla NUMBER_OF_ELEMENTS (Q), joka palauttaa arvonaan jonossa kutsuhetkellä olleiden alkioden lukumäärän.

Kahta jonoa voidaan verrata keskenään funktiolla EQUAL (Q1, Q2), joka palauttaa arvon TRUE, jos jonoissa Q1 ja Q2 on täsmälleen samat alkiot samassa järjestyksessä.

Jonon voi monistaa kopioimalla se toiseen jonoon proseduurilla COPY (Q1, Q2), joka sijoittaa jonon Q2 arvoksi jonon Q1.

Pinot

Pino, samoin kuin jono, on lineaarinen lista. Pinossa alkiot lisätään ja poistetaan listan samasta päästä, ns. pinon huipusta. Pinon pohjalla on vanhin pinon viety alkio. Pinolla on siten Last-In-First-Out (LIFO) luonne.

Pinon maksimikorkeuteen pätevät samat säännöt kuin jonon maksimipituuteen. Pinon korkeus vaihtelee ohjelman suorituksen aikana. Eräs tapa on varata pinolle kiinteän mittainen alue muistista.

Myös pinopakkausessa on huomioitava erilaiset virhetilanteet. Jos pinoon yritetään lisätä uusi alkio, eikä muistissa ole enää tilaa, aiheutetaan OVERFLOW -poikkeus. Samoin, jos tyhjistä pinosta yritetään poistaa alkio, aiheutetaan UNDERFLOW -poikkeus. Jos viitataan tyhjän pinon alkuun tai osoitetaan olematonta alkioa, aiheutuu ELEMENT_NOT_FOUND -poikkeus.

Pinon operaatiot

Alkion lisäämisen ja poistamisen lisäksi tarvitaan operaatiot pinon pituuden tutkimiseen, yksittäisen alkion osoittamiseen sekä uuden pinon luontiin, pinon kopiointiin ja kahden pinon vertailuun. Useat pinon operaatiot ovat samoja kuin jonossa.

Operaatio PUSH (S; E) lisää alkion E pinon S huipulle. Muistitilan loppuessa kesken aiheutuu OVERFLOW -poikkeus.

Operaatio POP (S, E) kopioi pinon S päällimmäisen alkion muuttu- jaan E ja poistaa sen pinosta. Yritettäessä poistaa alkio tyhjistä pinosta aiheutuu UNDERFLOW -poikkeus. Siksi tarvitaankin funktio IS_EMPTY (S), joka palauttaa arvon TRUE, jos pino on tyhjä, muuten arvon FALSE. Funktio ei aiheuta poikkeuksia.

Funktiolla TOP (S) saadaan pinon päällimmäisen alkion arvo. Tämä eroaa proseduurista POP siinä, ettei alkioa poisteta pinosta. Tyhjä pino aiheuttaa ELEMENT_NOT_FOUND -poikkeuksen.

Pinon muiden alkioden arvo saadaan funktiolla PEEK (S, N), joka palauttaa arvonaan pinon N:n alkion arvon. Jos pino on lyhyempi kuin N, aiheutetaan ELEMENT_NOT_FOUND -poikkeus.

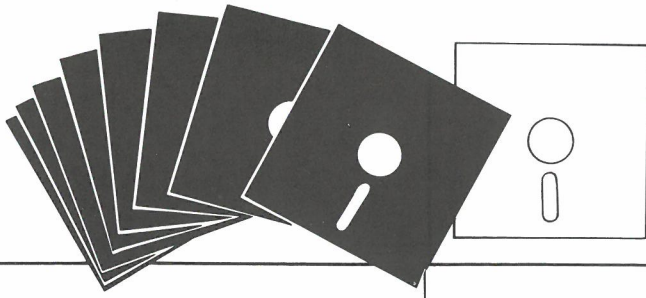
Proseduurilla EMPTY (S) luodaan uusi pino S.

Pinon S pituutta voidaan kysyä funktiolla NUMBER_OF_ELEMENTS (S), joka palauttaa arvonaan pinossa kutsuhetkellä olleiden alkioden lukumäärän.

Kahta pinoa voidaan verrata keskenään funktiolla EQUAL (S1, S2), joka palauttaa arvon TRUE, jos pinoissa S1 ja S2 on täsmälleen samat alkiot samassa järjestyksessä.

Pino voidaan monistaa kopioimalla se toiseen pinoon proseduurilla COPY (S1, S2), joka sijoittaa pinon S2 arvoksi pinon S1.

Jono ja pino mahdollistavat 'Data Structured Programming' -ajattelutavan ohjelman rakennetta suunniteltaessa. □



MikroMikko TopTen

Tämäkertainen MikroMikkojen ohjelmistolista perustuu kuluvan vuoden huhtikuun myyntiluvuista laskettuihin suhteellisiin indekseihin.

WordPerfect on noussut listan ykköseksi ollen samalla suosituin tekstinkäsittelyohjelmisto tällä hetkellä. Kmies näyttää vakiinnuttaneen asemansa tietokantaohjelmien ykkösenä.

Uusina tulokkaina listalla ovat MikroMikko 2 -lähiverkon sovellusohjelmoinnin työkalut MPS-Index (sija 4) ja MPS-Forms (sija 10).

Yllättävää listalla on Multiplanin putoaminen pois kymmeniköstä. Ilmeisesti tämä on kuitenkin vain tilapäistä. Sen sijaan Multigraphia myytiin hyvin huhtikuussa (sija 3).

MikroMikko TopTen

1. (7) WordPerfect	56
2. (2) Kmies	50
3. (10) Multigraph	49
4. (-) MPS-Index	48
5. (1) WordStar	36
6. (4) MS-Basic -tulkki	23
7. (6) Lotus 1-2-3	22
8. (8) Edix	21
9. (9) Wordix	20
10. (-) MPS-Forms	18

Suluissa sijoitus edellisessä TopTenissä (Softari 1/86).

Multigraph-liitanta Kmieheen

Versiosta 1.07B alkaen toimitetaan Kmiehen jakelulevykkeellä muunnosohjelma KMSYLK. Ohjelmalla pystytään muuntamaan halutut tietokentät Kmiehen kortistosta Microsoftin SYLK-formaattiin. Tämän jälkeen uudella Multigraph-ohjelmalla saadaan tietoaineisto graafiseen esitysmuotoon.

Multigraphista uusi versio

MikroMikko 2:n Multigraph-ohjelmasta on nyt saatavana uusi versio 3.0. Ohjelman käyttäjäystävällisyyttä on parannettu ja siihen on lisätty monia uusia ominaisuuksia.

Multigraph on businessgrafiikkaohjelmisto, jolla voidaan esittää Multiplan-taulukoita sektori-, pylvä- ja viivakuvioina. Uusi versio mahdollistaa graafisen tulostuksen myös Kmiehen kortistoista.

Kysyessään parametrejä uusi Multigraph näyttää käyttäjälle käsiteltävänä olevan laskenta-alustan. Käyttäjä voi antaa parametrikentät suoraan taulukonäytöstä kohdistimen avulla. Laskenta-alustaa voidaan selata näytöllä sekä sivuttain että riveittäin. Näin Mul-

tigraphin käyttö on entistä helpompaa.

IDC-näytönohjaimella varustetussa MikroMikossa pystytään Multigraphilla lisäämään tekstiä graafiseen kuvaan. Tavallisen tekstin lisäksi teksti voi olla alleviivattua ja vahvennettua. Kuva on myös muokattavissa siirtämällä pinta- ja viivakuvioita teksteineen paikasta toiseen tai vaikkapa poistamalla niitä kuvasta.

Versio 3.0 toimii myös MikroMikko 2 -lähiverkossa verkkokäyttöjärjestelmän versiosta 3.10 alkaen. Muita uusia ominaisuuksia ovat ohjelmiston kirjoitinvalikoiman monipuolistuminen ja ohjelmaparametrien muuttaminen suoraan ohjelmasta.

Maailmanennätyksiä tietokoneellakin

Maailmanennätyksiä tehdään nykyään mitä erilaisimmissa asioissa. Myös tietokoneilla. Viimeisin tällainen ME kantautui toimituksemme korviin Kaliforniasta.

Symbolics Inc:ssa työskentelevä matemaatikko ja tutkija William Gosper on laskenut Symbolics 3670 -koneella

17 526 100 milj. piin (π) desimaalia, mikä on uusi ME. Aiempi ennätys 16 777 216 milj. oli japanilaisilla.

Kouluajoilta muistamme, että pii määriteltiin ympyrän kehän ja sen halkaisijan suhteeksi. Piin lukuarvo on noin 3.14159265.

MikroMikko 2 -verkkoon uusi käyttöjärjestelmäversio

MikroMikko 2 -lähiverkon toisen vaiheen verkkokäyttöjärjestelmän toimitukset ovat käynnistyneet. Käyttöjärjestelmäversio 3.22, joka pohjautuu MS-DOS 2.0:aan, sallii usean massamuistillisen koneen liittämisen verkkoon. Versiossa on uusina ominaisuuksina taustatulostus (netprint), näytön kopio (hardcopy), ladattavat ulkoiset laiteohjaimet (esim. RS232-ohjain), viestien välitys työasemalta toiselle sekä grafiikka.

Käyttöjärjestelmä mahdollistaa tulostuksen kirjoittimelle taustalla samanaikaisesti, kun suoritetaan jotain toista ohjelmaa. Taustatulostus toimii kuten yksittäisessä MikroMikko 2:ssa.

Näytön kopiolla saadaan näytön kuva sekä teksteineen että graafisine esityksineen kirjoittimelle. Tätä ominaisuutta käytetään mm. Multigraph-ohjelmassa hyväksi. Kirjoitin voi sijaita missä tahansa verkon koneessa.

Microsoftilta tehokkaampia kääntäjiä

Microsoftin ohjelmointikieliä on kehitetty entistä tehokkaammiksi ja toistensa kanssa yhteensopivammiksi. Pascal, Fortran ja C -kielillä kirjoitettuja aliohjelmita voidaan linkata yhteen. Parametrit voidaan välittää sekä arvo- että muuttujaparametreina.

Pascalin uusi versio 3.31 on ISO-standardin mukainen ja sisältää useita laajennuksia, jotka mahdollistavat tehokkaan järjestelmäohjelmoinnin. Se tukee käyttöjärjestelmän MS.DOS 3.x uusien piirteiden hyväksikäyttöä.

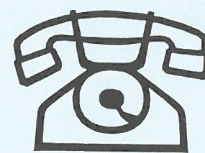
Fortran-versio 3.31 on laaja alijoukko X3.9-1978 ANSI:n määrittelemästä FORTRAN-standardista. Myös se tukee käyttöjärjestelmän MS.DOS 3.x uusien piirteiden hyväksikäyttöä. Tässä versiossa on lisäksi parannettu merkkijonojen käsittelyä.

Microsoftin C-kääntäjän versio 3.00 on täydellinen toteutus Kernighanin ja Ritchien määrittelemästä C-kielestä. C-kielinen ohjelma voi kutsua Fortran 3.3, Pascal 3.3 tai Macro Assemblerilla käännettyjä aliohjelmita. C-kääntäjään sisältyy yli 200 funktiota käsittävää XENIX ja UNIX System V -yhteensopiva aliohjelmakirjasto.

Microsoftin Macro Assemblerin uu-

sin versio on 4.0. Se sisältää edellistä versiota jopa kolme kertaa nopeamman assembler-kääntäjän (MASM), joka tukee myös suorittimien 80186 ja 80286 käskykanta. Samaten uuteen versioon sisältyy entistä nopeampi linkitysohjelma (LINK), ohjelmakirjastojen ylläpito-ohjelma (LIB), symbolinen debuggeri (SYMDEB) sekä työkalu helpottamaan ohjelmien ylläpitoa. Tämä työkalu (MAKE) päivittää ohjelman, jos yhteen tai useampaan lähdetiedostoon on tehty muutoksia. Assemblerin mukana seuraa myös työkalu EXE-tiedostojen tiivistämiseen (EXEPACK), jolla ohjelmien latausajat saadaan lyhyemmiksi.

Microsoftin Cobol-kääntäjän uusin versio 2.1 on ANSI X3.23-1974 -standardin mukainen. Kieleen sisältyy mm. tiedostojen lukitusmahdollisuus monen käyttäjän järjestelmissä, indeksoitujen tiedostojen (ISAM) moniavainkäsitteily sekä sisäänrakennettu lajittelu- ja yhdistelytoiminto. Kääntäjän versiolle 1 kirjoitetut ohjelmat voidaan kääntää sellaisenaan uudella versiolla. Niin ikään muistia säästävä dynaaminen aliohjelmien lataus ja poisto on mahdollista uudella versiolla.



MikkoInfo avoinna pitempään

MikkoInfo -puhelinneuvontapiste on nyt avoinna arkipäivisin kello 15.00 asti.

MikkoInfosta MikroMikkojen käyttäjät voivat saada puhelimitse neuvoja erilaisissa ongelmatilanteissa. Numerossa 90-567 2200 neuvoja antavat kello 9.00–15.00 Joel Ortila, Helena Brännare ja Tarja Mäntyjärvi.

Uusia DPS 6 -julkistuksia

DPS 6 -käyttäjäpäivillä huhtikuussa julkistettiin uusi DPS-malli DPS 6/50, kiinteä levyasema FSD413, nauha-asema MTU9635, puskuroitu levyasemaohjain DIRAM 32, RS-323 -sovitin, sekä uusia ohjelmistoversioita, kuten DSA6 3.1.

DPS 6/50 täydentää hyvin nykyistä DPS 6 -laitesarjaa. Se on jatkoa matalakabinettisille koneille, DPS 6/20:lle, DPS 6/22:lle ja DPS 6/40:lle. Laite on täysin yhteensopiva DPS 6 -sarjan muiden koneiden kanssa.

DPS 6/50:ssä on DPS 6/75:n suoritin ja siten se on erittäin tehokas minikone. Maksimikokoonpanossaan se on 48 käyttäjän laite ja keskusmuistia siihen saadaan 2 megatavua. DPS

6:een voidaan liittää kuusi levy-yksikköä ja käyttämällä uusia kiinteitä 413 megatavun FSD313-levyasemia laitteen levykapasiteetti nousee aina 2400 megatavuun.

Pienikokoisen FSD413 winchesterlevyn formatoitu kapasiteetti on peräti 413 megatavua. Lisäksi levyn hinta/kapasiteetti -suhde on alhainen. Koska levyt ja luku/kirjoituspäät on asennettu ulkoilmalta suojattuun koteloon, levyaseman toiminta on myös luotettavaa. Uuden puskuroidun MSC9615-ohjaimen ansiosta levyaseman siirtonopeus on jopa 1,8 megatavua sekunnissa.

Uusista julkistuksista ja DPS 6 -laitesarjasta enemmän seuraavassa Softarissa.

Oraclen laitevalikoima laajenee

Oracle-tietokantaohjelma on nyt saatavissa Stratuksen lisäksi MikroMikko 2:een, Apollo-työasemakoneeseen sekä Vax-laitteistoihin.

Oracle on relationaalinen tiedonhallintajärjestelmä, johon liittyy välineitä

sovellusten kehittämiseen. Oracle perustuu SQL-kieleen, josta on muodostumassa ANSI-standardi relationaalisten tiedonhallintajärjestelmien kieleksi. Oracle esiteltiin Softarissa 3/85.

VDU 220 uudistunut

Uudessa VDU 220 -pääteessä on 15 tuuman näyttö ja tummat kirjaimet vaalealla pohjalla kuten MikroMikko 2:ssa ja PääteMikossa. Näppäimistö on myös samanlainen kuin MikroMikko 2:ssa, sisältäen pehmonäppäimet.

VDU 220 emuloi DEC:n VT220, VT100 ja VT52 -päätteitä ja tukee myös ANSI X3.64-standardia näytön ohjauksessa. Pääteparametrit annetaan käyttäjäystävällisellä asetusohjelmalla (set-up).

Muita pääteominaisuuksia ovat:

- 80 tai 132 merkkiä rivillä
- kaksinkertainen merkin leveys ja korkeus
- yhdistelmämerkit
- merkkijoukon lataus set-up -asetusohjelmalla tai linjalta
- pehmonäppäinsekvenssien määrittely set-up:lla
- pehmonäppäintekstien määrittely set-up:lla tai isäntäkoneelta
- tilarivit erikseen sovellukselle ja pääteelle

Nokia
Softari-lehti
PL 780
00003 HELSINKI

Vastaustähtäys
HKI 10 Lupa 2509

Nokia
maksaa
postimaksun

Softari 2/86

Palvelukortti

☐ Haluan esitteitä seuraavista
Nokian palveluista ja tuotteista.

☐ Pyydän, että minuun otetaan
yhteyttä.

puhelin: _____

Olen kiinnostunut _____

- ☐ Korjaan yllämainitut osoitetiedot.
☐ Poistan yllämainitut osoitetiedot.
☐ Lisään allämainitut osoitetiedot.
☐ Softarin ☐ NETin rekisteristä/rekisteriin.

yritys _____

osasto _____

lähiosoite _____

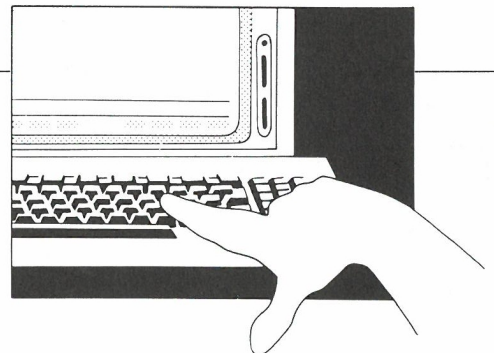
postinro _____

nimi _____

postitoimip. _____

JOUKKOJULKAISU

Tavallisen laadukasta palvelua



Jo aamun ensimmäinen tapaaminen saa atk-suunnittelu- ja myyntitoimisto Oy Semmel & Tre kumppar Ab:n johtajaomistajan hengästymään.

Herra Semmelin verenpainetta nosti suuren tietokoneyrityksen edustajan käynti. "Sivistymätön ja ylimielinen juippi", Semmel tuhisee.

"Tulee tänne kuin ulosottomies ja kaappaa kainaloonsa meidän ohjelmistokehityslaitteemme. Ei muka saada kauppaa eikä pystytä maksamaan ensimmäistäkään laskua."

"Poika olis varmaan vinyt viimeisenkin laitteiston, ellen olisi heti älynnyt tilata viittä uutta. Piti vähän näyttää että kyllä kauppa käy."

"Mutta sitten sen kirotn laukusta löytyi vielä muka meille sopivaa ohjelmistokehitystyötä. Autokorjaamo-ohjelmiston reskontran koodausta. Vaikka hyvin on tiedossa, että me ollaan erikoistuttu ratkaisemaan nimenomaan leipomoiden kokonaistietojenkäsittelyä laitteineen, ohjelmistoineen, kaikkineen."

Semmelin murheet on kertaheitolla poispyyhkäisty, kun sisään tupsahtaa asiakkaan näköinen tyyppi ja käyntikortti paljastaa tulijan leipomon johtajaksi.

Tosin asiakkaan likaiset kynnet viittaavat enemmän autokorjaamoon kuin leipomoon. Mutta täällä pohjolan perukoilla ei kaikilla yksityiskohdilla ole niin väliä, kuten jo Semmelin Saksasta Suomeen eksyneellä leipuriesi-isällä oli tapana sanoa.

Ja yhtäkkiä Semmel on vaudissa.

"Aikaansa seuraavan leipomon varma valinta on meidän firma. Leipurialan ja atk-alan täydellisesti osaavina ammattilaisina meidän nelihenkinen tehoryhmämme syvässä hanskassa on kaikki alan ongelmat. Ja koska minustakin alunpitäen piti tulla leipuri, kokonaisratkaisumme on lähdetty tekemään just leipureille."

"Mutta, kun ...", yrittää asiakas keskeyttää. Semmel sen kun jatkaa.

"Ratkaisumme perustuu uusinta huippuluokan makroteknologiaa hyödyntävään MakroMaija 10:een. Sen 512-bittinen SBC16-prosessori tukee erillistä aritmetiikkaprosessoria, mikä puolestaan mahdollistaa leivinuunin lämpötilan joustavan ohjauksen. Ja samanaikaisesti voit tulostella vaikkapa lähetykslistat ja laskut korkealuokkaisella laserilla, jonka tulostusnopeus on konfiguroitavissa setup-moodilla aina 21602 baudiin. Näin leivät paistuu tasaisesti ja nopeasti, ja päivittäiset paperirutiinit hoituu itsestään siinä sivussa."

"Äläkä usko mitä kilpailijat sanoo. Meillä on tämän alan kovin osaaminen myös huomisen vaatimuksia silmällä pitäen."

"Mutta, mutta ...", asiakas yrittää tämän tästä keskeyttää. Semmelin vuodatus kuitenkin jatkuu niin kauan kuin järjestelmän monipuolisuus, ohjelmistotuen laajuus, huollon korkealuokkaisuus, koulutuksen etevämyys ja laadun ylivoima on todistettu sen seitsemällä kalvolla.

Viimein parituntisen ylistyslaulun jälkeen asiakaskin saa suunvuoron.

"Sullahan on hyvä virma. Ja kovat tuntuvat olevan lääkkeetkin, mutta kun minä olen oikeastaan faktori ja omistan pienen LeipäSet-latomon. Ja olen kuullut, jotta tuota tekstiä voisi siirtää puhelinlinjoja pitkin. Niin tuln vaan kyselemään oikeen niinkun atk-ammattilaiselta, että miten se teksti niillä puhelinlinjoilla oikeen siirtyy, kun ne siitä lehdissäkin kirjoittaa."

Softaaja